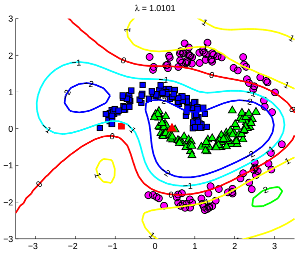
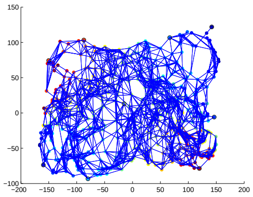
Regularization path algorithm for statistical learning

Karina Zapien Arreola Zapién Arreola

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES





Thèse

présentée par

Karina ZAPIÉN ARREOLA

pour obtenir le grade de

Docteur

de l'Institut National des
Sciences Appliquées de Rouen

Discipline : Informatique

Sujet :

Algorithme de Chemin de Régularisation pour l'apprentissage Statistique

Regularization Path Algorithm for Statistical Learning

Jury :

Stéphane Canu	(Directeur)	Professeur à l'INSA de Rouen (HDR)
Olivier Chapelle	(Rapporteur)	Chercheur Senior à Yahoo
Gilles Gasso	(Encadrant)	MdC à l'INSA de Rouen
Yann Guermeur	(Rapporteur)	Chargé de recherche CNRS du groupe ABC (HDR)
Hélène Paugam-Moisy	(Examinateur)	Professeur à l'Université Lumière Lyon 2 (HDR)
Fabrice Rossi	(Rapporteur)	MdC à TELECOM ParisTech (HDR)
Michel Verleysen	(Examinateur)	Professeur à l'Université Catholique de Louvain

Laboratoire d'Informatique, de Traitement de l'Information et des Systèmes (LITIS) - INSA de Rouen
Avenue de l'Université, 76801 Saint Etienne du Rouvray Cedex

Résumé

La sélection d'un modèle approprié est l'une des tâches essentielles de l'apprentissage statistique. En général, pour une tâche d'apprentissage donnée, on considère plusieurs classes de modèles ordonnées selon un certain ordre de « complexité ». Dans ce cadre, le processus de sélection de modèle revient à trouver la « complexité » optimale, permettant d'estimer un modèle assurant une bonne généralisation. Ce problème de sélection de modèle se résume à l'estimation d'un ou plusieurs hyperparamètres définissant la complexité du modèle, par opposition aux paramètres qui permettent de spécifier le modèle dans la classe de complexité choisie.

L'approche habituelle pour déterminer ces hyperparamètres consiste à utiliser une « grille ». On se donne un ensemble de valeurs possibles et on estime, pour chacune de ces valeurs, l'erreur de généralisation du meilleur modèle. On s'intéresse, dans cette thèse, à une approche alternative consistant à calculer l'ensemble des solutions possibles pour toutes les valeurs des hyperparamètres. C'est ce qu'on appelle le chemin de régularisation. Il se trouve que pour les problèmes d'apprentissage qui nous intéressent, des programmes quadratiques paramétriques, on montre que le chemin de régularisation associé à certains hyperparamètres est linéaire par morceaux et que son calcul a une complexité numérique de l'ordre d'un multiple entier de la complexité de calcul d'un modèle avec un seul jeu hyper-paramètres.

La thèse est organisée en trois parties. La première donne le cadre général des problèmes d'apprentissage de type SVM (Séparateurs à Vaste Marge ou Support Vector Machines) ainsi que les outils théoriques et algorithmiques permettant d'appréhender ce problème. La deuxième partie traite du problème d'apprentissage supervisé pour la classification et l'ordonnancement dans le cadre des SVM. On montre que le chemin de régularisation de ces problèmes est linéaire par morceaux. Ce résultat nous permet de développer des algorithmes originaux de discrimination et d'ordonnancement. La troisième partie aborde successivement les problèmes d'apprentissage semi supervisé et non supervisé. Pour l'apprentissage semi supervisé, nous introduisons un critère de parcimonie et proposons l'algorithme de chemin de régularisation associé. En ce qui concerne l'apprentissage non supervisé nous utilisons une approche de type « réduction de dimension ». Contrairement aux méthodes à base de graphes de similarité qui utilisent un nombre fixe de voisins, nous introduisons une nouvelle méthode permettant un choix adaptatif et approprié du nombre de voisins.

Mot clés : chemin de régularisation, sélection de modèle, classification, ordonnancement, parcimonie, réduction de dimension, graphe de similarité.

Abstract

The selection of a proper model is an essential task in statistical learning. In general, for a given learning task, a set of parameters has to be chosen, each parameter corresponds to a different degree of "complexity". In this situation, the model selection procedure becomes a search for the optimal "complexity", allowing us to estimate a model that assures a good generalization. This model selection problem can be summarized as the calculation of one or more hyperparameters defining the model complexity in contrast to the parameters that allow to specify a model in the chosen complexity class.
The usual approach to determine these parameters is to use a "grid search". Given a set of possible values, the generalization error for the best model is estimated for each of these values. This thesis is focused in an alternative approach consisting in calculating the complete set of possible solution for all hyperparameter values. This is what is called the regularization path. It can be shown that for the problems we are interested in, parametric quadratic programming (PQP), the corresponding regularization path is piecewise linear. Moreover, its calculation is no more complex than calculating a single PQP solution.

This thesis is organized in three chapters, the first one introduces the general setting of a learning problem under the Support Vector Machines' (SVM) framework together with the theory and algorithms that allow us to find a solution. The second part deals with supervised learning problems for classification and ranking using the SVM framework. It is shown that the regularization path of these problems is piecewise linear and alternative proofs to the one of Rosset [Ross 07b] are given via the subdifferential. These results lead to the corresponding algorithms to solve the mentioned supervised problems. The third part deals with semi-supervised learning problems followed by unsupervised learning problems. For the semi-supervised learning a sparsity constraint is introduced along with the corresponding regularization path algorithm. Graph-based dimensionality reduction methods are used for unsupervised learning problems. Our main contribution is a novel algorithm that allows to choose the number of nearest neighbors in an adaptive and appropriate way contrary to classical approaches based on a fix number of neighbors.

Keywords: regularization path, model selection, classification, ranking, sparsity, dimensionality reduction, neighborhood graph.

A mi familia
mama, papa, Erik

y a mi otra mitad
Olivier.

Contents

Remerciements

Je tiens à remercier en premier lieu les membres du jury, Olivier Chapelle, Yann Guermeur, Hélène Paugam-Moisy et Fabrice Rossi pour avoir pris le temps de s´intéresser à mes travaux. Ses commentaires m'ont définitivement aidé à mieux mettre en forme mes résultats et à soulever des questions très interessantes.

Cette thèse n'aurait, sans doute, jamais abouti sans Gilles Gasso, qui m'as donné l'opportunité de travailler avec lui. Tout au long de cette thèse, il n´a jamais perdu espoir et je l´en remercie. Je lui serai toujours très reconnaissante de sa précieuse aide lors des difficultés rencontrées, pour sa disponibilité permanente et sa grande patience.

Je remercie également Stéphane Canu pour m'avoir donné l'opportunité de faire cette thése. Il a mon admiration car c´est un chercheur passionné, dedié à la cause scientifique et qui arrive toujours à remonter le morale de son équipe.

To Olivier Durand-Viel, that has always fight to understand me and make everything he can to keep the sun shining in my heart. For helping me to be better. A mi p que sabe en todo momento contagiarme de su alegria de la vida. Der Mann dass Sinn und Kraft auf meinem Leben gibt. Je te remercie.

A mi hermano Erik, que siempre camina con su corazon en la mano y que en todo momento comparte sinceramente mis alegrias conmigo, por todo su apoyo invaluable, judicionso e infatigable a lo largo de estos largos años de estudio y trabajo.

A mis papas, Ramón Zapién Gaitán y Olga Arreola Zavala a quienes les debo gran parte de lo que soy, les agradezco su amor incondicional y esfuerzo incansable por buscar siempre lo mejor para su familia.

Je souhaite faire également une dédicace à mes collègues de bureau : à Elsa Planterose, qui possede une énergie de vivre contagieuse, et je la remercie pour sa bonne hummeur et ses conseils judicieux. A Grégory Mallet qui a contribué au bon déroulement de cette dernière année et qui a partagé les hauts et les bas de ma thèse.

A Brigitte qui fait bien plus qu'elle ne devrait pour nous aider à résoudre nos problémes, je serai toujours très contente d'avoir connu sa maniére de trouver la vie aussi optimiste.

Je remercie aussi tous les autres thesards, ex-thesards et amis que j'ai connus ici, qui m'ont entourés durant ces trois années et qui ont tous contribué a créer une atmosphère chaleureuse et conviviale: à Gaëlle Loosli, Vincent Guigue, Sarah Kahane, Olivier Francois, Sam Maes, Fréderic Suard, Karine Criner, Rémi Flamary, Benjamin Labbé, Firas Abou Latif, Yaquian Li, Xilan Tian et Alexandros Karatzoglou, je leur souhaite toute la réussite qu'ils méritent!

A tout le departement des professeurs ASI qui m'ont adopté en toute facilité : Nico Delestre, Nico Malandain, Seb-Seb Bonnegent, Alain Rakotomamonjy (qui sera toujours un ASI ;), Florence Aubry, Alex Pauchet et Romain Herault.

Für meine alles liebste WG und Freunde dass ich in Deutschland gelassen habe, aber immer in meinem Herzt bleiben werden, besonders die Karla, der Oscar, der Dani, Andrés, Katharina, Christophe, Dominik, Lea, Claudia und Volker.

Ich will auch bedanken der Thomas Gärtner, der mir sehr geholfen hat, ich habe sehr genissen mit Ihm gezuarbeiten.

Introduction

A large number of real life questions turn out to be far too complicated to be solved directly as real world problems. Additionally, in general, the whole phenomenon cannot be observed and the complete problematic structure has to be inferred with only partial observations. In order to find a solution the observed problem is reduced to be able to represent it into mathematical terms and notation in order to solve it with known theoretical tools.

When designing a mathematical model, different kinds goals have to be satisfied at the same time, which in general lead to opposite solutions on its own. Parameters are introduced in order to control the cost-benefit trade-off between all desired objectives to attain.

This thesis studies three different learning frameworks. These problems have the common characteristic that can be modeled as optimization problems consisting of conflicting goals. Extra parameters will be used to keep an equilibrium between all objectives. The efficient search of these parameters is the central issue on this thesis, since it will lead to a satisfactory or not model.

The first learning problem belongs to the supervised learning framework, where data samples are given with an attached label. This label can represent classes, relevance, dependent value, etc. Labels are used with the general structure of the known samples to build a decision function that will hopefully also work for unseen points. The aim is to build a function capable of correctly assigning the observed data to their corresponding label, which can be done by adjusting a decision function to these data. It is also desired to have a decision function able to generalize for unseen points, this is normally controlled by measuring the model difficulty and number of observed errors done by the built function. To achieve this goal and according to the now well known statistical learning theory, a trade-off between the errors made by the learned function on the training data and the complexity of the learner must exists. Along this work, efforts were directed toward the correct and efficient model search in terms of computational time and generalization error. The attention was focused on a class of learners represented by the Support Vector Machine. In this framework the loss function used to measure the precision of the learner during the training phase is a hinge-loss cost which is a piecewise linear function while the complexity of the decision function is expressed via its quadratic norm. Under these statements, it turns out that the choice of the trade-off parameter can be efficiently done by computing a regularization path [Ross 07b]. This computation consists essentially in tracking the evolution of the decision function according to the regularization parameter, having the advantage that this property can be extended to other formulations with different loss and complexity function. In this general framework, Hastie et al. [Hast 04] have proposed the regularization path for SVM for classification where the final results of the algorithms is a partition of the search interval into breakpoints where the linear relation between the optimal functions changes.

Contributions

In this thesis, we provide an **alternative view of Rosset's path formulation** [Ross 07b] by using a **functional formulation** (see Section 2.1). As far as our knowledge can go, this is the first time that such a formulation is proposed. We believe this formulation is **more general**. This formulation leads as well to the formulation of a bi-dual problem which is an equivalent problem of the primal optimization problem with the advantage that the form of the decision function is explicit. The aim of the expressed learning problem under these formulations is to achieve a satisfactory generalization ability. An analysis of the way the overfitting problem arises is given in Theorem 1.16.

We **extended the regularization path algorithm to the ranking problem** under the SVM framework [**Zapi 08b, Zapi 08a**]. The ranking problem occurs typically in search engines according to queries. The calculation of the whole solution set is a lot faster by means of the regularization path than by repeatedly solving the QP problem in a grid search, which is the traditional method.

To speed up the algorithm, we propose a procedure to **reduce the number of constraints** involved in the ranking problem by a **reformulation of the constraints graph**. Indeed, if we consider a set on n samples, the number of ranking constraints could be of order $\mathcal{O}(n^2)$. Even though the computation of the regularization path could be fast, the algorithm suffers from this high number of constraints. Therefore, we derive a pre-processing algorithm to generate a reduced graph of ranking constraints to alleviate the drawback. When we face a ranking problem with different levels of relevance, the principle of the algorithms consists in generating intra-level constraints (horizontal constraints) and inter-level constraints (vertical constraints). The vertical constraints impose that all the samples of higher level of relevance dominate one randomly chosen sample of lower level. Doing so, we have considerably reduced the complexity of the problem and reduced the computational cost, nevertheless, there is a reduction in the performances of the ranking decision function. This proposal is explained in depth in Section 2.4 and was published in [**Zapi 09**].

The aforementioned problem in the ranking framework concerning the number of constraints can impact the computation cost because the SVM algorithm expresses the ranking function as a linear combination of the constraints. Another way to reduce the complexity is to express the ranking function as a function of the samples instead of the constraints or using a sparsity penalty to control the number of elements involved in the solution. We investigate in Section 2.5 the **sparse ranking SVM** where the loss function is a quadratic one and the regularizer is the L_1-norm penalty. Experimental results are provided and compared to the classical ranking SVM.

The regularization path algorithm that was analyzed, tracks the evolution of the learner according to the training set. The same analysis can be carried out for a validation set. In Section 2.3 **the validation path was developed**, where it was shown that with high probability, if the training and validation sets follow the same distribution, it is enough to keep the best solution from the breakpoints given on the partition proposed by the regularization path on the learning set and with high probability, it is not necessary to make a more refined search to find the deeper partition proposed by the validation path. Nevertheless, if two breakpoints in the regularization path are kept as optimal solution, the breakpoints given by the validation set can also be efficiently calculated. This analysis brings a new insight in the properties of the regularization path algorithms.

Up to now, we have considered that all the samples come accompanied with their labels. However, labeling can be very costly or not possible for all data. In the semi-supervised learning framework, labels can be known for only a part of the dataset but not for the whole population. This information even though uncomplete can still be used to help unraveling the hidden information for the rest of the population and improve generalization capability of the learner [Burg 05]. As for the ranking problem, a desired model would be the one that manages to have a satisfactory generalization ability by using only the essential points. In Section 3.1, we consider the Laplacian SVM algorithm [Belk 06] to retrieve geometrical information from the unlabeled data. As a result, this algorithm expresses the learned function with all available samples (labeled or not). We conduct experiments showing that it is unnecessary to keep all the samples in the final solution. From this remark, we propose a **sparse version of the Laplacian SVM** where the necessary degree of sparsity is analyzed automatically via the computation of a regularization path, simplifying the model search process. The main results of this algorithm were published in **[Gass 07c], [Gass 08b] and [Gass 08c]**.

Another way to exploit the unlabeled data is to use them to build an appropriate representation of the data for supervised learning. The last part of this thesis deals with unsupervised learning. Working on this kind of problem can aim at cluster discovering or dimensionality reduction. The dimensionality reduction problem is approached by searching two objectives: elimination of superfluous information and preservation of the intrinsic relationships. That is, useless information has to be discarded keeping the essential relationship between sampled data. Many of the implemented algorithms consider a neighborhood graph. Unfortunately, if this graph is not properly built, the results will be disappointing: local properties such as distance preserving will not be conserved in the reduced space. Hence, efforts in Section 3.4 are focused in the **proper neighborhood graph construction** by taking into account local properties and the manifold assumption. These properties include the Euclidean distance and the distance to the tangent at each point and are used to create an appropriate neighborhood graph **[Zapi 07]**. The experimental results are compelling compared to the existing graph correction algorithms.

This work is structured as follows: an introduction to all necessary concepts is given in Chapter 1 and basic definitions are developed in the Appendixes. Especially emphasis is given to the connection between the type of regularized problem solved usually in machine learning and the constrained optimization. The links between notions of Pareto frontier and regularization paths are also highlighted. The rest of the chapter reviews the SVM framework formulation, together with the mathematical and optimization tools used to handle such an algorithm and the global picture of model selection. Chapter 2 is dedicated to model selection for fully supervised learning using the regularization path. This chapter exhibits our functional formulation and investigates the validation path as well as path computation for ranking problems. Chapter 3 considers semi-supervised and unsupervised learning and develops our contributions to model selection under these frameworks. Finally, the thesis provides some concluding remarks and discusses some extensions.

Résumé

L'objet principal de cette thèse est la sélection efficace de modèle dans le cadre de l'apprentissage statistique. Les problèmes d'apprentissage statistique, en particulier les problèmes de reconnaissance de formes font appel à des données totalement ou faiblement étiquetées. L'élaboration de modèles performants à partir de ces données procède en général de l'optimisation de critères antagonistes. En effet, l'absence de connaissances sur les lois statistiques ayant généré les données et l'impossibilité d'obtenir un nombre infini de données représentatives de ces lois amènent l'utilisateur à faire appel à la minimisation structurelle de risques [Vapn 79]. Le cadre théorique maintenant bien établi de l'apprentissage statistique [Vapn 79] stipule que les capacités de généralisation d'un modèle sont liées à sa précision sur les données d'apprentissage et à sa complexité. Par généralisation, nous entendons la capacité du modèle à prédire correctement les étiquettes des données n'ayant pas servi à son réglage. Un modèle relativement complexe aura tendance à s'adapter aux particularités des données d'apprentissage (notamment aux bruits) et généralisera mal sur des données de test et vice versa. Se pose alors le problème du choix du critère de mesures de la précision du modèle et la quantification de sa complexité. La précision est évaluée par une fonction de coût alors que la complexité peut prendre la forme d'un terme de régularisation [Schl 01]. En plus de ces choix, une question essentielle est la réalisation du compromis entre ces critères antagonistes. Ce travail présente une manière efficace d'appréhender la réalisation de ce compromis.

Pour illustrer nos propos, nous avons étudié dans un premier temps des problèmes de classification binaire en utilisant le formalisme des machines à vecteur support (SVM : Support Vector Machines) popularisées depuis les travaux initiaux de Boser et al. [Bose 92]. La fonction de décision est un hyperplan linéaire ou non-linéaire dans l'espace des données. Dans le cadre classique des SVM, la fonction de coût utilisée pour approximer la vraie erreur 0-1 (bonne ou mauvaise classification) est le coût charnière (hinge loss). Cette fonction est convexe et linéaire par morceaux. Pour mesurer la complexité du modèle, une autre fonction convexe représentée par la norme quadratique du modèle est considérée. Pour résoudre le problème d'optimisation multi-objectif, l'approche SVM minimise la combinaison linéaire de la fonction de coût et du terme de régularisation via un paramètre de régularisation $0 \leq \lambda < \infty$. La qualité du modèle final dépendra, toutes choses égales par ailleurs, du choix optimal de ce paramètre. Il a été établi par Rosset et al. [Ross 07b] que les paramètres de la fonction de décision varie de facon linéaire par morceaux lorsque le paramètre λ change : on obtient alors le chemin de régularisation [Hast 04]. Ce faisant, le choix de λ peut être automatisé car il est facile de déterminer l'évolution du modèle en fonction de λ et de retenir le meilleur modèle en fonction d'un critère de sélection de modèle. De surcroît, le calcul du chemin de régularisation a, en pratique, une complexité numérique légèrement plus élevée que la résolution d'un SVM avec une seule valeur de λ.

Dans cette thèse, nous proposons une vue alternative des conditions nécessaires pour qu'un algorithme d'apprentissage admette un chemin de régularisation linéaire par morceaux. Cette vue présente un caractère plus général et repose sur des outils de dérivation fonctionnelle. De manière similaire, une formulation alternative du chemin de régularisation pour les SVM a été proposée par dérivation directe du problème d'optimisation des SVM via des outils d'analyse convexe (voir Section 2.1). Le chemin de régularisation ne fournit pas seulement une méthode pour suivre efficacement l'évolution du modèle sur les données d'apprentissage. Son formalisme peut être étendu pour analyser le chemin de validation c'est-à-dire l'évolution de l'erreur de généralisation évaluée comme la précision du modèle sur des données de validation. Ainsi, on peut efficacement suivre les changements de l'erreur de validation en fonction du paramètre de régularisation. L'analyse empirique du chemin de validation montre que si les ensembles d'apprentissage et de validation sont statistiquement proches, avec une grande probabilité, l'évaluation de l'erreur de validation aux points de changements (en apprentissage) de la variation linéaire du modèle est suffisante pour réaliser une sélection pertinente de modèle. Cette analyse développée dans la Section 2.3 apporte

une nouvelle vision des propriétés du chemin de régularisation.

Dans un deuxième temps, nous avons étendu le formalisme du chemin de régularisation à des problèmes d'ordonnancement (ranking problems) résolus avec des modèles de type SVM [Zapi 08b, Zapi 08a]. Le calcul de l'ensemble des solutions est beaucoup plus rapide en utilisant le chemin de régularisation que si le problème de programmation quadratique induit par le SVM est résolu plusieurs fois pour différentes valeurs du paramètre de régularisation.

Pour accélérer l'algorithme, une réduction du nombre de contraintes participant au problème d'ordonnancement est proposée en passant par une reformulation du graphe de contraintes (voir Section 2.4). Si on considère un ensemble de n points, le nombre de contraintes peut être de l'ordre $\mathcal{O}(n^2)$. Le nombre de paramètres du problème de programmation quadratique étant du même ordre, les temps de calcul deviennent très rapidement prohibitifs pour des problèmes à grande échelle même pour un chemin de régularisation. La proposition d'un nouveau graphe permet de réduire le nombre de contraintes. Le principe de sa construction pour un problème d'ordonnancement avec différents niveaux de pertinence consiste à regrouper les points d'apprentissage en fonction de ces niveaux. Des contraintes d'ordonnancement horizontales (intra-niveaux) et verticales (inter-niveaux) sont ensuite générées. A chaque niveau, un point dit maximal est choisi aléatoirement et une relation d'ordre (supériorité) est imposée entre tous les points du niveau de pertinence supérieur et ce maximum. Le graphe est finalement complété par des contraintes horizontales en imposant des relations d'ordre entre le maximum de chaque niveau et tous les autres points partageant ce niveau. Cette construction réduit considérablement le nombre de contraintes et par conséquent les temps de calcul [Zapi 09]. Cependant, une légère réduction de la performance du modèle a été également observée.

La réduction de la taille du modèle final n'est pas seulement importante à cause du temps de calcul du temps, mais aussi parce que la taille du modèle optimal dépend du nombre de contraintes. Une autre manière de réduire sa complexité est d'exprimer directement la fonction d'ordonnancement en fonction des points. Cette démarche s'inspirant de la décomposition d'un signal sur un dictionnaire [Dono 03a] consiste à définir le modèle comme étant une combinaison linéaire de fonctions élémentaires (linéaires ou non-linéaires) définies en chaque point d'apprentissage. Un sélection des fonctions élémentaires pertinentes est réalisée en adjoignant au problème d'optimisation une contrainte de parcimonie définie comme la norme L_1 du vecteur de paramètres du modèle. Ce modèle a été appris avec une fonction de coût quadratique en utilisant un chemin de régularisation linéaire par morceaux. Dans la Section 2.5, une description détaillée de la formulation du chemin dans ce contexte et une comparaison empirique de ce nouveau modèle d'ordonnancement avec le modèle classique de type SVM sont exposées.

Dans notre exposé, nous avons jusqu'à maintenant considéré que les données d'apprentissage étaient toutes étiquetées. Cependant, dans certaines applications, il arrive que le coût d'étiquetage des données soit important et qu'il faille élaborer le modèle à partir d'un faible nombre de données avec étiquettes et un grand nombre de données sans étiquettes. Ce problème d'apprentissage dit semi-supervisé peut être résolu en extrayant l'information sur la distribution statistique marginale des données sans étiquettes et en intégrant cette information dans la résolution. Un autre type d'information pouvant être pertinent à l'apprentissage semi-supervisé est la structure géométrique sous-jacente des données. En effet si les données appartiennent à différentes classes pouvant être décrites comme des variétés géométriques, il est pertinent d'utiliser des graphes de similarité pour représenter cette structure. Le cadre flexible des algorithmes de type SVM permet d'intégrer aisément ce type d'information. Ainsi nous avons étudié l'algorithme du Laplacien SVM [Belk 06] qui inclut cette information de structure sous la forme d'une régularisation favorisant une variation lisse et régulière de la fonction de décision le long des variétés. Un inconvénient de la solution obtenue via cet algorithme est que le modèle a généralement autant de paramètres que de points (avec ou sans étiquettes) et est donc peu parcimonieux. Dans la Section 3.1 une solution consistant à ajouter une pénalisation supplémentaire de type L_1 sur les paramètres du modèle est proposée afin d'en réduire sa complexité. En se fondant sur les travaux de Wang [Wang 06b], un chemin de régularisation est présenté permettant de calculer efficacement la solution. En parcourant le chemin de régularisation, les paramètres du modèle varient linéairement par morceaux, ce qui en fait un algorithme très efficace pouvant être couplé avec une procédure de validation croisée pour sélectionner le meilleur modèle. L'application sur des données simulées et réelles montre les avantages d'un modèle parcimonieux où nous obtenons les mêmes niveaux de performance que le Laplacien SVM en réduisant significativement la complexité du modèle final.

La dernière partie de ce travail, pouvant servir d'étape de pré-traitement pour les problèmes d'apprentissage précédemment évoqués, concerne la réduction de dimension dans le cadre de l'apprentissage non-supervisé. Cette réduction de dimension peut être utile à plusieurs égards comme par exemple la visualisation des données, la réduction de la dimensionnalité du problème et vise à projeter les données dans un espace de dimension réduite tout en préservant les informations topologiques. Pour aborder cette problématique, nous nous sommes intéressés à des techniques basées sur les graphes de similarité entre les points. Ce type de graphes comme dans l'algorithme du Laplacien SVM sert à modéliser la proximité géométrique des points. En faisant l'hypothèse de variétés formées par les points, l'objectif de la réduction de dimension est la préservation de cette similarité après projection des points. Le souci majeur de ces techniques est qu'elles considèrent un nombre fixe de voisins pour tous les points, en ignorant la densité locale autour de chaque point. La conséquence est l'apparition de faux voisins et de raccourcis (shortcuts) dans le graphe induisant une mauvaise modélisation de la topologie locale de la variété. Nous avons alors proposé une procédure permettant d'éviter ces raccourcis [Zapi 07]. Elle est basée sur un choix adaptatif du nombre de voisins. Elle part de l'hypothèse de régularité des variétés à modéliser et suppose qu'en chaque point de la variété, la topologie locale peut être approximée par un hyperplan (dans un espace euclidien) tangent à la variété. Les points voisins sont ceux situés à proximité de l'hyperplan (c'est-à-dire ceux se trouvant à une distance inférieure à un certain seuil) et dans un certain cône angulaire. De ce fait, différents nombres de voisins sont envisageables pour chaque point d'apprentissage. L'évaluation empirique de la procédure montre une amélioration nette de la qualité de la projection en termes de préservation de similarité entre points.

Le reste du document est structuré de la manière suivante : le chapitre 1 présente les principales notions de l'apprentissage statistique et la manière classique d'appréhender le compromis optimal à assurer entre complexité et précision du modèle. Nous avons ensuite établi une passerelle cette formulation classique et l'optimisation multi-critères notamment la notion d'optimalité de Pareto. Une vue générale des procédures utilisées dans la littérature pour la sélection de modèle est aussi présentée. Dans le chapitre 1, le cadre des modèles de types SVM est développé ainsi que les outils théoriques et algorithmiques utilisés pour déterminer ces modèles. Le chapitre 2 présente l'arsenal théorique et algorithmique sous-tendant le calcul du chemin de régularisation pour la classification et l'ordonnancement. L'analyse du chemin de validation ainsi que la comparaison entre le graphe complet des contraintes d'ordonnancement et sa version réduite y sont développés. Ce chapitre se termine par la dérivation du chemin de régularisation pour l'ordonnancement sous la forme d'un développement parcimonieux sur un dictionnaire. Le dernier chapitre, chapitre 3, aborde le problème d'apprentissage semi-supervisé et sa résolution via une version parcimonieuse de l'algorithme du Laplacien SVM. La deuxième moitié de ce chapitre traite des méthodes de réduction de dimension à base de graphes de similarité des données. Finalement, ce document se conclut par quelques conclusions et perspectives.

1 Background

Consider a machine which receives a set of inputs. These will be called the *samples*, *input data* or *points*, and could correspond to an image on the retina, pixels in a camera or a sound waveform. It could also correspond to less obviously sensory data, for example the words in a news story, the list of items in a supermarket shopping basket or a taken survey. All this data can be reduced to $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \subset \mathcal{X}$, representing an image, a sound wave, or a word, etc.

Having a set representing the same kind of data, this data might be grouped according to particular characteristics. The corresponding group will be named the *label* or *class* which can be known or not. The label that corresponds to sample i, $\mathbf{x}_i$ will be denoted y_i. The aim is to build a decision function f able to distinguish which label corresponds to each sample. This decision function is the ***classifier*** or ***learner***.

In general, the goal of statistical learning is to learn the decision function from the knowledge of some samples $\mathbf{x}_i$ and y_i with the intent to perform as well on the unseen data as on the seen data. It is well known in the literature that without a clever design, one can build a complex decision function able to explain in depth the seen data (even the noisy data or outliers) and which fails to predict correctly the appropriate labels of the unseen samples. This phenomenon is known as ***overfitting***. The paradigm employed to tackle this issue in statistical learning consists in balancing at least two antagonist criteria: the adequacy to the learning data and the control of complexity. This control of complexity avoids minimizing only the given error in seen data, leading hopefully to a better generalization of the learned model for new data. The topic of this chapter is to provide some efficient tools to help the neophyte user to achieve the aforementioned trade-off in the particular context of Support Vector Machines (SVM) [Schl 01]. The SVM algorithms were broadly studied these twelve last years and had shown satisfactory state of the art results in most applications. They are based as well on a strong and sound mathematical theory and many numerical tools were developed to address the practical resolution of the optimization problem.

This chapter aims at settling the background of the used tools and is divided as follows: Section 1.1 introduces the learning problem and relates it to a multiple objective optimization. This part also introduces the notion of hypothesis space in which the decision function is sought. In particular, we will consider in Section 1.1.2 the kernel space used in SVM to extend the linear algorithms to nonlinear case. The notions of model adequacy (loss) and complexity (regularizer) are described together with the regularization path which is the way the trade-off behaves, deriving a link with the optimal Pareto frontier. This approach induces the problem of model selection which is studied in Section 1.2, where the different techniques of selecting a model are exposed and a brief analysis is summarized. The learning paradigm is illustrated in Section 1.3 on two particular cases: classification and ranking. Section 1.4 provides basic mathematical tools to address such a formulation. Optimality conditions to decide if a solution has been found are revised in this section. It will be shown in Section 1.5 that the handled problems can be embedded into a standard quadratic optimization problem thanks to duality. The chapter ends with a review of the most used methods to solve quadratic problems.

1.1. Statistical Learning Overview

We will be dealing with a general setting called a ***learning problem***, where sets $\mathcal{X}, \mathcal{Y}$ are considered and an unknown fixed joint probability measure $\mathbb{P}(\mathbf{x}, y), \mathbf{x} \in \mathcal{X}, y \in \mathcal{Y}$ is defined over $\mathcal{X} \times \mathcal{Y}$. We are particularly interested in the conditional probability $\mathbb{P}(y|\mathbf{x})$ since the aim is to be able to build an algorithm, called the learner. This last will take samples $S \subset \mathcal{X} \times \mathcal{Y}$ to *learn* the relation between points and labels to predict the appropriate label y for an observed $\mathbf{x}$, even though, this point has not been seen before by the learner.

Several methods have already been developed to realize supervised learning. These include neural networks, regression methods, boosting, etc. (see [Hast 01] for an overview). In this work, we will be focused on the support vector machines framework.

Support Vector Machines (SVM) [Vapn 99, Bose 92] are learning methods originally designed for binary classification and regression. Its flexible framework makes them applicable to different kind of learning problems. They possess several advantages as they have proved to have high generalization ability, they are based on solid mathematical principles, and the training time has been improved with efficient algorithms. However, as any other model, the generalization ability depends on the chosen parameters, that is, efficient model selection has to be done in order to obtain satisfactory results.

Hereafter, we develop the framework of statistical learning and the induced trade-off between approximation quality and complexity control which is the basis of the SVM.

1.1.1 Overview: Learning as a Multi-criteria optimization problem

The pursued objective is the design of a decision function $f \in \mathcal{H}$ under a known space $\mathcal{H}$ of functions $f : \mathcal{X} \to \mathcal{Y}$ that best predicts y from $\mathbf{x}$ according to the given probability. The space $\mathcal{H}$ will be called the ***hypothesis space*** The hope is that $\mathcal{H}$ is dense enough to include the best model according to $\mathbb{P}$.

In order to find the most appropriate function and evaluate its quality, a **loss function** will be given:

$$\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+ \cup \{0\}$$

that will measure the discrepancy between the predicted value $f(\mathbf{x})$ and the ground truth given by the space $\{(\mathbf{x}, y) | \mathbf{x} \in \mathcal{X}, y \in \mathcal{Y}\}$.

Then, from a statistical point of view, the searched function would be the one that minimizes ℓ over the space $\mathcal{H} \times \mathcal{X} \times \mathcal{Y}$ according to its probability measure. This is equivalent to minimizing its expectation with respect to $\mathbb{P}(\mathbf{x}, y)$, that is, the searched function is the one that solves:

$$\operatorname*{argmin}_{f \in \mathcal{H}} \mathbb{E}_{\mathcal{X} \times \mathcal{Y}}[\ell(f, \mathbf{x}, y)] = \operatorname*{argmin}_{f \in \mathcal{H}} \int\limits_{\mathcal{X} \times \mathcal{Y}} \ell(f, \mathbf{x}, \mathbf{y}) \mathbb{P}(d\mathbf{x}, dy) \tag{1.1}$$

The term $\mathbb{E}_{\mathcal{X} \times \mathcal{Y}}[\ell(f, \mathbf{x}, y)]$ is called the ***expected risk*** [Vapn 82].

Since the joint distribution $\mathbb{P}(\mathbf{x}, y)$ is unknown and in reality it is only possible to partially observe the domain $\mathcal{X} \times \mathcal{Y}$, the function f will be determined by using only a sample set $S = \{(\mathbf{x}_i, y_i)\}_{i=[\![n]\!]} \subset \mathcal{X} \times \mathcal{Y}$ which is assumed to be drawn i.i.d. according to $\mathbb{P}(\mathbf{x}, y)$. Let $S_X = \{\mathbf{x}_i\}_{i=[\![n]\!]}$ and $S_Y = \{y_i\}_{i=[\![n]\!]}$, where $[\![n]\!] = 1, 2, ..., n$. Then the expected risk will be approximated with:

$$\mathcal{L}(f, S) = \frac{1}{n} \sum_{i=1}^{n} \ell(f, \mathbf{x}_i, y_i),$$

$\mathcal{L}(f, S)$ is called the ***empirical risk***. The usefulness of this formulation lies in the fact that this is a ***consistent estimator***, that is, the empirical risk converges in probability to the searched expectation in Equation (1.1), where the unknow probability $\mathbb{P}(\mathbf{x}, y)$ is being approximated using the seen data S. This way of choosing a decision function is called the empirical risk minimization (***ERM***) method [Vapn 99].

The goal of the learner is to find a model that correctly predicts the corresponding label given a new input even though this has never been seen by the classifier. This is called the ***generalization ability***.

Nevertheless, since only a finite sample of the space $\mathcal{X}$ is seen, to minimize this criterion, we would only need a complete hypothesis space where one of the functions can perfectly adjust itself to a finite sample set. This function will then minimize the empirical risk, possibly resulting in a zero loss, but it will not guarantee to efficiently predict unseen samples. This phenomenon is known as ***overfitting***.

In general, the complexity of this function increases as it fits more precisely the sample set. This complexity can be measured by various ways. For instance, the VC dimension was proposed by Vapnik and Chervonenkis [Vapn 79] a complexity index of a class of functions. In a more empirical way, the complexity can be related to the norm L_2 or to the number of parameters or variables involved in the model. This concept will be included in the problem as additional criterion to avoid overfitting. This function is denominated ***penalization*** or ***regularization function***. In the sequel, we will denote the penalization of f by the term $\Omega(f)$ defined as:

$$\Omega : \mathcal{H} \to \mathbb{R}^+ \cup \{0\}.$$

The inclusion of a penalization in the minimization problem will lead to a strongly consistent method (see [Vapn 99] for more details) for choosing a decision function. This will form a multi objective problem: we aim at minimizing the loss function and in parallel controlling the complexity of f, that is, a compromise between simplicity and data fitting. These two objectives are in general opposite to each other as a simple model might not be able to express the complete relationship in the data as there might be missing explanatory variables or noise while a very complex model might risk to overfit the data. The stated problem is written as a trade-off of these two objectives, which is regulated by a parameter λ.

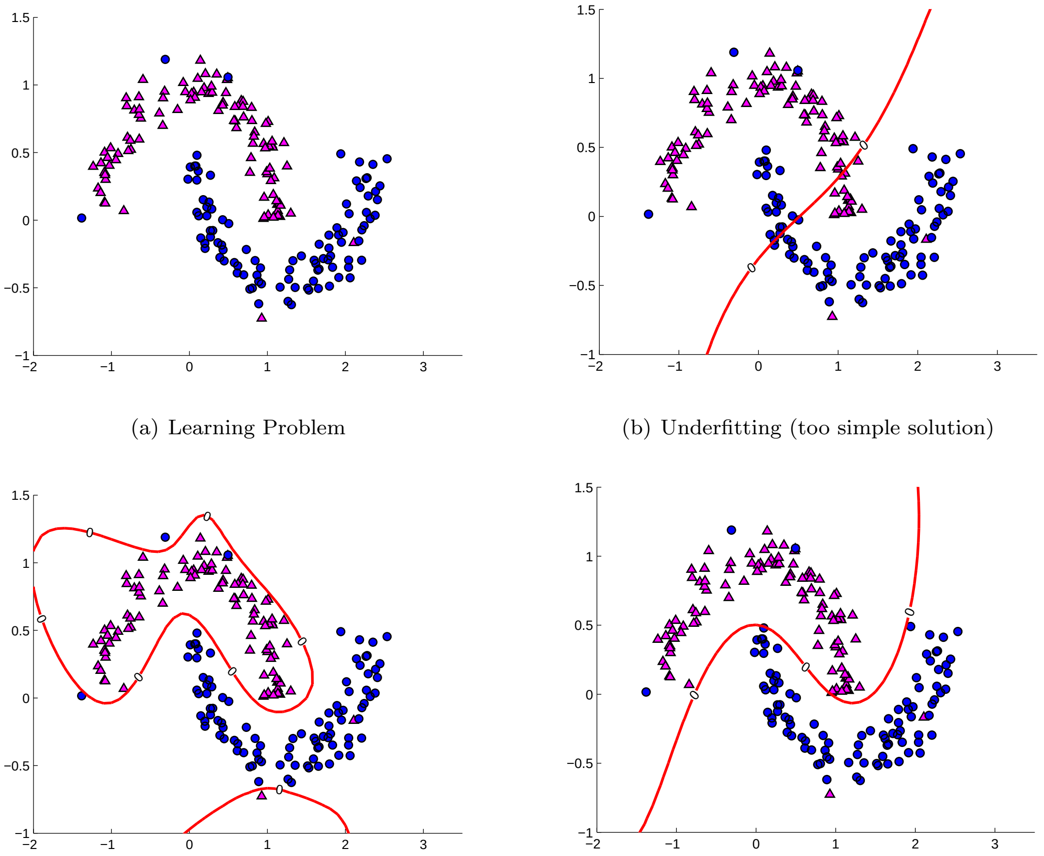
The type of problems we are interested in solving will have the form of a ***Parametric Programming***, which are about to be discussed in Subsection 1.1.5 under the setting of a multi-criteria optimization. Such a problem will look as follows:

Problem 1.1 (Multi-criteria optimization Problem)**.** *Each criteria is balanced by a weigh λ as follows:*

$$\hat{f}_\lambda = \operatorname*{argmin}_{f \in \mathcal{H}} \mathcal{L}(f, S) + \lambda \Omega(f)$$

where λ is a non-negative parameter that must be set by the user aiming at designing a learning function that performs well on the training sample set S and the unseen data as well. The example in Figure 1.1 illustrates how the trade-off influences the obtained solution. The aim in this example is to be able to find a decision function that keeps in one side all triangles and on the other side all circles. In Figure 1.1(b), λ in Problem (1.1) is very large and penalizes the complexity of the function. The obtained function will suffer from underfitting, that is, the output is then a very simple decision function that will have neither a satisfactory generalization ability nor a low number of observed errors in the training set. The opposite case is when more weight is given to the loss function, the decision function is rather complex and overfitting is produced as in Figure 1.1(c). Finally, if the appropriate value for parameter λ is found, we would obtain a solution like in Figure 1.1(d), where the compromise between complexity and accuracy is reached.

The possibility to attain this compromise will depend in the chosen hypothesis space which needs to be complete and in the chosen trade-off parameter. This will be translated in the possibility to obtain also an optimal solution of Problem 1.1 for all λ. In addition, this space must be able to provide a ***stable solution*** in the sense that a small change in S will induce a small change in f. The study of kernel spaces in the following section will show that they form a suitable option.



(a) Learning Problem

(b) Underfitting (too simple solution)

(c) Overfitting (bad generalization)

(d) Desired Solution

Figure 1.1: Illustrations of the trade-off between the loss and regularization functions. The red line defines the decision frontier between the two classes.

Given λ, an optimal solution can be found for the multi-criteria optimization problem. The consequences of this parameter change on f is the main interest of this dissertation.

There exists several theoretical frameworks that propose parametric programming which include statistical analysis [Vapn 99], regularization [Pogg 89], model selection [Mass 07], and Bayesian posterior maximization [Hast 01]. They all consist in an objective function that results from the combination of two measures: the data adjustment and the hypothesis regularity or complexity.

Economists have also studied these kinds of problems with convex objective functions, noting that a set of optimal solutions can be found where each solution corresponds to the result of a multi-objective optimization problem solved with a particular trade-off parameter. This set of optimal solutions is called the Pareto optimal set or ***Pareto frontier*** [Bi 03a, Bi 03b, Pare 97]. By definition, Pareto solutions are considered optimal because there are no better solutions in terms of all objectives functions [Steu 86, Bele 99, Miet 99]. In the statistical learning field, this set of optimal solutions is called the ***regularization path*** [Hast 04].

This connection between the Pareto frontier and the regularization path helps to understand and study derived properties that will be useful to efficiently track the evolution of the decision function in Problem 1.1 as λ changes. It will also clarify why piecewise linear paths are computed and the advantage of it. We will show that convexity of $\mathcal{L}$ and Ω and their piecewise linear or quadratic characteristics in the case of support vector machines play a central role in the derivation of the problem solution.

1.1.2 Hypothesis space induced by a kernel

Each element in the settled Problem 1.1 has an influence in the obtained result. In this section we will put particular interest in the kind of hypothesis space $\mathcal{H}$ the decision function belongs to. As it was mentioned, we need a complete and stable hypothesis space. It will be seen that the space produced by kernel functions possesses many convenient characteristics that make them a good choice of search space. Before delving into these characteristics, some notions of kernels must be defined.

Several loss functions are based on similarity measures between samples such as the inner product $\langle \mathbf{x}, \mathbf{z} \rangle = \mathbf{x}^\top \mathbf{z}$ for some $\mathbf{x}, \mathbf{z} \in \mathbb{R}^{\mathcal{D}}$, $\mathcal{D} \in \mathbb{N}$ that can be seen as a similarity measure between real vectors. Another more powerful similarity measure known as *kernel* will be introduced here, which will allow us to extend linear methods to larger and more complex spaces. Hence, the kernel function can be seen as a generalization of the similarity measure to objects which can not be necessary real vectors. Indeed, the kernel functions can be engineered for a large number of structures as graphics, texts, proteins, documents, etc. [Shaw 04].

Definition 1.2 (Kernel [Aron 50])**.** *Let $\mathcal{X}$ be a set, then a **kernel** $\mathbf{k}$ is a function of two variables from $\mathcal{X} \times \mathcal{X} \to \mathbb{R}$.*

For convenience, we will deal with ***symmetric kernels***, that is, we will suppose that $\mathbf{k}(\mathbf{x}, \mathbf{y}) = \mathbf{k}(\mathbf{y}, \mathbf{x})$ for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$.

Kernel Functions

A particularly desired property on a kernel is its positiveness, the space generated by ***positive kernel*** has been already studied by Mercer [Merc 09] who showed that these functions can be indeed seen as a similarity measure.

The kernel function for a finite set of samples $S_X = \{\mathbf{x}_i\}_{i=[\![n]\!]}, \mathbf{x}_i \in \mathcal{X}$ can be represented as a matrix $K \in \mathbb{R}^{n \times n}$ (called the ***Gram matrix***) with entries $K_{ij} = \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j)$. This kind of induced Gram matrices by positive kernels were already studied by Moore [Moor 16]. All results were later summarized by Aronszajn [Aron 50] after the second world war.

Definition 1.3 (Semi-Positive Definite Kernel (p.d. kernel))**.** *A kernel $\mathbf{k}$ is said to be a **semi-positive definite kernel** if for any positive integer n:*

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) \geq 0 \text{ for all } \{\alpha_i\}_{i=[\![n]\!]} \in \mathbb{R}, \quad \{\mathbf{x}_i\}_{i=[\![n]\!]} \in \mathcal{X}. \tag{1.2}$$

Definition 1.4 (Positive Definite Kernel)**.** *A kernel $\mathbf{k}$ is said to be a **positive definite kernel** if for any positive finite integer n, for all $\{\mathbf{x}_i\}_{i=[\![n]\!]}\mathbf{x}_i \in \mathcal{X}$ and for all $\{\alpha_i\}_{i=[\![n]\!]}\alpha_i \in \mathbb{R}$ with at least one $\alpha_i \neq 0$:*

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) > 0. \tag{1.3}$$

The positive definiteness of a kernel $\mathbf{k}(\mathbf{x}, \mathbf{y})$ over samples S_X can be tested by checking if the resulting eigenvalues of matrix K are strictly positive, that is, if for all $\mathbf{v}_i \in \mathbb{R}^n$ such that $K\mathbf{v}_i = \lambda_i \mathbf{v}_i$, $\lambda_i > 0, i = [\![n]\!]$. The Gram matrix provides also the pairwise comparison between samples S_X and we will see that it can be easily embedded in the learning algorithms.

Some examples of commonly used kernels with their properties for $\mathcal{X} = \mathbb{R}^{\mathcal{D}}$ are listed below.

Homogeneous kernel Also known as ***linear kernel***, defined as: $\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle = \mathbf{x}_i^\top \mathbf{x}_j$ for all $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^{\mathcal{D}}$. As the inner product is symmetric, we can see that it is a positive kernel because:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \mathbf{x}_i^\top \mathbf{x}_j = \left(\sum_{i=1}^{n} \alpha_i \mathbf{x}_i \right)^\top \left(\sum_{j=1}^{n} \alpha_j \mathbf{x}_j \right) = \left\| \sum_{i=1}^{n} \alpha_i \mathbf{x}_i \right\|^2 \geq 0.$$

Polynomial kernel For two samples $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^{\mathcal{D}}$, the polynomial kernel with parameter $d \in \mathbb{N}$ is defined as:

$$\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + b)^d.$$

The symmetry and positiveness of this kernel follows directly from the homogeneous kernel properties.

Gaussian kernel The Gaussian kernel [Bara 93, Jone 95], is a radial basis function (RBF). It is based on the Gaussian probability distribution, where parameter $\sigma \in \mathbb{R}$ would represent the square root of the variance. This form is also used in signal processing where parameter σ determines the bandwidth of the applied filter. It is defined as:

$$\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right).$$

This is a symmetric positive definite kernel, and it is illustrated in Figure 1.2.

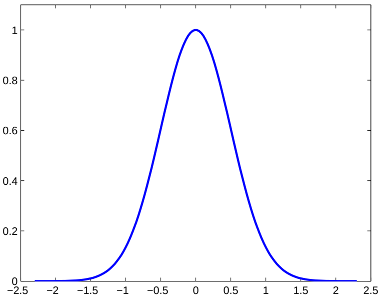


Figure 1.2: Example of the Gaussian kernel applied to all $\mathbf{x}_i \in \mathbb{R}$ with $\mathbf{x}_j = 0$ and $\sigma = 0.5$.

Rational quadratic kernel This kernel has experimentally proved that it provides as satisfactory results as the Gaussian Kernel [Kocs 04]. It was analyzed by Genton [Gent 02] and has the following form:

$$\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) \quad = \quad \frac{t}{t + \|\mathbf{x}_i - \mathbf{x}_j\|^2}.$$

Sigmoidal kernel This is not a positive kernel in general but it was inspired by the neural networks [Luss 08, Liu 08] and has been largely discussed [Schl 97, Sell 99, Guo 05, Haas 05, Lin 03]. There are two formulations of this kernel, the first one uses the hyperbolic formulation:

$$\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\kappa \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \theta),$$

parameter θ controls the center of the function, while parameter κ controls the slope. This is an approximation of the sign function: $\operatorname{sign}(x) = \left\{ \begin{array}{ll} -1 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \end{array} \right.$, and the advantage of the sigmoid function compared to this one is its continuity and differentiability with the limit case $\kappa = \infty$ corresponding to the sign function.

The other formulation is the exponential formulation:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{1 + e^{-\|\mathbf{x}_i - \mathbf{x}_j\|}}$$

which can be seen as an approximation of the Heaviside function (or zero-one error function) [Abra 74]

$$\Gamma(x) = \left\{ \begin{array}{ll} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{array} \right. .$$

The Gaussian function proposed by Boser, Guyon and Vapnik [Bose 92], [Guyo 93] and [Vapn 99] is usually one of the first choices in most applications.
All additional parameters in kernel definitions (usually called hyperparameters) are user determined, that is, these have to be a priori chosen, but there are several new works that intend to automatically find the best hyperparameters [Keer 07, Wang 06b, Bach 05].
We will be interested in positive definite kernels since it will be seen that these result in very nice hypothesis spaces. There are several kinds of positive kernels and an extended summary can be consulted in the work of Genton [Gent 02]. Even though there exists several other non-positive kernels with interesting properties, [Ong 04, Lin 03] they will not be further treated in this document.
Since kernels will help us to define a space of searched functions, the first question about these objects concerns the properties owned by the generated space.

Theorem 1.5 (Closure properties)**.** *Let* $\mathbf{k}_1$ *and* $\mathbf{k}_2$ *be two kernels over* $\mathcal{X} \times \mathcal{X}$*,* $a \in \mathbb{R}^+$*, and* f *a real value function on* $\mathcal{X}$*,* $\phi : \mathbb{R} \to \mathbb{R}$ *a polynomial with positive coefficients and* $\psi : \mathcal{X} \to \mathcal{X}$*. Then, the following functions are p.d. kernels for* $\mathbf{x}, \mathbf{z} \in \mathcal{X}$*:*

i) $\mathbf{k}(\mathbf{x}, \mathbf{z}) = \mathbf{k}_1(\mathbf{x}, \mathbf{z}) + \mathbf{k}_2(\mathbf{x}, \mathbf{z})$,

ii) $\mathbf{k}(\mathbf{x}, \mathbf{z}) = a\mathbf{k}_1(\mathbf{x}, \mathbf{z})$,

iii) $\mathbf{k}(\mathbf{x}, \mathbf{z}) = \mathbf{k}_1(\mathbf{x}, \mathbf{z})\mathbf{k}_2(\mathbf{x}, \mathbf{z})$,

iv) $\mathbf{k}(\mathbf{x}, \mathbf{z}) = f(\mathbf{x})f(\mathbf{z})$,
The proof of $i) - iv)$ *properties can be found in [Shaw 04, Cato 07].*

v) $\mathbf{k}(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{k}_1(\mathbf{x}, \mathbf{z}))$ *is a p.d. kernel.*
Proof sketch: as $\phi(\cdot)$ *is a polynomial, this result can be proved by induction using properties* $iii)$ *and* $ii)$*.*

vi) $\mathbf{k}(\mathbf{x}, \mathbf{z}) = \mathbf{k}_1(\psi(\mathbf{x}), \psi(\mathbf{z}))$
Proof: Since $\mathbf{k}_1$ *is a p.d. kernel over* $\mathcal{X}$ *and since the image* $\psi(\mathbf{x})$ *of any point* $\mathbf{x} \in \mathcal{X}$ *belongs as well to* $\mathcal{X}$*, the positive definiteness follows obviously.*

It can be seen that in fact if it is possible to find some positive definite kernels, an infinite number of them can be derived and they form a closed convex cone. These results lead to the conclusion that an infinite number of kernels can be generated and these kernels belong to a generated close space over a field.
For any $\mathbf{x} \in \mathcal{X}$, and for a fixed set $\{\mathbf{x}_i\} \in \mathcal{X}, i = [\![m_f]\!]$, with m_f a natural number, a linear combination of the form $f(\mathbf{x}) = \sum_{i=1}^{m_f} \alpha_i \mathbf{k}(\mathbf{x}, \mathbf{x}_i)$ defines a function $f : \mathcal{X} \to \mathbb{R}$. Let $\mathcal{H}$ be the space of these functions: $\mathcal{H} = \{f(\mathbf{x}) : f(\mathbf{x}) = \sum_{i=1}^{m_f} \alpha_i \mathbf{k}(\mathbf{x}, \mathbf{x}_i), \alpha_i \in \mathbb{R}, m_f \in \mathbb{N}, \mathbf{k}(\mathbf{x}, \mathbf{x}_i) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}\}$. Using the closure properties of the kernel functions, we know that the addition and multiplication of this kind of functions result in another function belonging to the same space. If the induced space by the previous functions is taken as the hypothesis space, a distance is needed to test consistency and stability. The following inner product will induce a norm between this kind of functions.

Definition 1.6 (Kernel inner product)**.** *If* $f, g \in \mathcal{H}$ *with* $f(\mathbf{x}) = \sum_{i=1}^{m_f} \alpha_i \mathbf{k}(\mathbf{x}, \mathbf{z}_i)$ *and* $g(\mathbf{x}) = \sum_{i=1}^{m_g} \beta_i \mathbf{k}(\mathbf{x}, \mathbf{y}_i)$ *with* $m_f, m_g \in \mathbb{N}, \mathbf{x}, \mathbf{y}_j, \mathbf{z}_i \mathcal{X}, \alpha_i, \beta_j \in \mathbb{R}, i = [\![m_f]\!], j = [\![m_g]\!]$*. A bilinear form called an* ***inner product under*** $\mathcal{H}$ *will be defined as follow:*

$$\langle f, g \rangle_{\mathcal{H}} = \sum_{i=1}^{m_f} \sum_{j=1}^{m_g} \alpha_i \beta_j k(\mathbf{z}_i, \mathbf{y}_j) \tag{1.4}$$

For simplicity, this inner product can also be denoted $\langle f, g \rangle_{\mathcal{H}} = \langle f, g \rangle$.
The next interest of the kernels lies on the fact that a function f issued from a linear combination of these kernels can be *reproduced* by the aforementioned inner product. The interesting part of this particularity of the positive kernels is that it will help us to calculate functional derivatives.

Definition 1.7 (Reproducing Property)**.** *Let $\mathcal{H}$ be a class of functions defined over $\mathcal{X}$, $\mathcal{H}$ forming a Hilbert space with inner product $\langle\cdot,\cdot\rangle_{\mathcal{H}}$. The function $\mathbf{k}(\mathbf{x},\mathbf{y}), \mathbf{x},\mathbf{y} \in \mathcal{X}$ is called a* ***reproducing kernel*** *[Aron 50] or is said to possess the* ***reproducing property*** *in $\mathcal{H}$ if*

1. *For every $\mathbf{x} \in \mathcal{X}$, $\mathbf{k}(\mathbf{x},\cdot)$ is a function that belongs to $\mathcal{H}$.*
2. *The* ***reproducing property*** *holds for every $\mathbf{x} \in \mathcal{X}$ and every $f \in \mathcal{H}$:*

$$f(\mathbf{x}) = \langle f(\cdot), \mathbf{k}(\mathbf{x},\cdot)\rangle_{\mathcal{H}}$$

Definition 1.8 (Reproducing kernel Hilbert space (RKHS))**.** *A Hilbert space $\mathcal{H}$ embedded with the inner product $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ is said to have with reproducing kernel if it exists a positive kernel $\mathbf{k}$ with the reproducing property over the elements in $\mathcal{H}$.*

Taking advantage of the p.d. kernel's properties, the hypothesis space $\mathcal{H}$ that will be used is built based on them. Let $\mathbf{k}$ be a p.d. kernel and let a hypothesis space $\mathcal{H}_0$ be defined as:

$$\mathcal{H}_0 = \{f | f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i \mathbf{k}(\mathbf{x},\mathbf{z}_i), \text{ with } m < \infty, \mathbf{x}, \mathbf{z}_i \in \mathcal{X}, \alpha_i \in \mathbb{R}, i = [\![m]\!]\}.$$

It can be verified that the inner product defined in Equation (1.4) induces a norm with the form $\|f\|^2_{\mathcal{H}} = \langle f, f\rangle_{\mathcal{H}}$. This norm will measure the complexity of function f, inducing a measure of overfitting.

Finally, the searched space will be the closure of $\mathcal{H}_0$ [Aron 50] noted:

$$\mathcal{H} = \overline{\mathcal{H}_0}.$$

This turns out to be a Hilbert space with a reproducing kernel, therefore, the hypothesis space we will use is a RKHS.

The hypothesis space built based on kernel functions forms a rich family of functions. Intuitively, when choosing a decision function f_1 as a sufficient expansion over the kernel function evaluated at $\mathbf{x}_i$, one can achieve a perfect fitting of the outputs y_i. If the complexity of f_1 is measured by any non-negative function of its norm $\|f_1\|_{\mathcal{H}}$ and we are able to find another function $f_2 \in \mathcal{H}$ with reduced complexity, the question is then how does this function perform on the training data and how will both functions perform on unseen samples.

In the next section, we review some usual loss functions according to the type of solved problems (classification or ranking) and common penalization functions as well. Although, the representer theorem (see Theorem 1.9) suggests the use of the norm of f as complexity measure, other kinds of measures can be proposed depending on the point of view adopted for f. Indeed, the measure can consist in taking a non negative penalty on the coefficients of the expansion of the decision function as described in Subsection 1.1.4.

Theorem 1.9 (Nonparametric Representer Theorem [Scho 01])**.** *Suppose we are given a positive definite real-valued kernel $\mathbf{k}$ on $\mathcal{X} \times \mathcal{X}$, a training set $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}, \mathbf{x}_i \in \mathcal{X}, y_i \in \mathbb{R}, i = [\![n]\!]$, a strictly monotonically increasing real-valued function g on $[0,\infty)$, an arbitrary cost function $\mathcal{L} : \mathcal{H} \times S \to \mathbb{R} \cup \{\infty\}$, and a class of functions*

$$\mathbf{F} = \{f \in \mathcal{H} | f(\cdot) = \sum_{i=1}^{\infty} \beta_i \mathbf{k}(\cdot, \mathbf{z}_i), \beta_i \in \mathbb{R}, \mathbf{z}_i \in \mathcal{X}, \|f\|_{\mathcal{H}} < \infty\}.$$

Here, $\|\cdot\|_{\mathcal{H}}$ is the norm in the RKHS $\mathcal{H}$ associated to $\mathbf{k}$. Then any function $f \in \mathbf{F}$ minimizing the regularized risk functional

$$\mathcal{L}(f, S) + g(\|f\|_{\mathcal{H}})$$

admits a representation of the form

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i \mathbf{k}(\cdot, \mathbf{x}_i).$$

1.1.3 Accuracy criteria: Loss function

An analysis of loss functions was already started in Section 1.1.1, in this section, we are interested in the different characteristics that form groups of them [Gass 07b], several properties and convergence characteristics have been studied [Bart 06, Stei 07, Rosa 04].

Definition 1.10 (Loss Function)**.** *Let $f \in \mathcal{H}$ be a decision function. A function $\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to [0, \infty)$ with the property $\inf_{\mathcal{X}\times\mathcal{Y}} \{\ell(f, \mathbf{x}, y)\} \geq 0$ for all $f \in \mathcal{H}$ is called a* ***loss function****.*

These functions play a key role in machine learning and measure the cost of applying the decision function f at the point $(\mathbf{x}, y)$. In general, we are interested in methods that output a function f with small average cost like the ERM introduced in Section 1.1.1.
It is required that the loss function be non-negative. This will mean that the efficiency of a decision function f will not be rewarded by doing a particularly good prediction while poorly performing in the rest of the samples.
It has to be noticed that the loss function is not always the objective function as it can be algorithmically infeasible.
There exists several loss functions tailored for the type of learning problem one intends to address. Most learning problems are classification and regression (in the former case, the output to be predicted is a discrete natural or relative value while in the latter case, the target spreads over $\mathbb{R}$). Hence, we will hereafter focus on the loss related to these problems. Most commonly used loss functions for a binary classification problem with samples $\{(\mathbf{x}_i, y_i)\}_{i=[\![n]\!]}$, $y_i \in \{-1, 1\}$ are listed below:

Misclassification error or (0-1)-loss. Consists on counting the misclassified examples, assigning 1 if an example is wrongly classified and 0 otherwise. It is a non-convex and a non-differentiable loss function:

$$\ell(f, \mathbf{x}, y) = \begin{cases} 0 & \text{if } y = f(\mathbf{x}) \\ 1 & \text{otherwise} \end{cases} .$$

Linex (Asymmetric) loss. One of these kinds of losses for which it is possible to solve analytically for the optimal predictor [Chri 97] is the *linex* loss:

$$\ell(f, \mathbf{x}, y) = b\Big(e^{a(yf(\mathbf{x}))} - a(yf(\mathbf{x})) - 1\Big),$$

with $a \in \mathbb{R} \setminus \{0\}, b \in \mathbb{R}^+$. For classification usage, it is named so because when $a > 0$, loss is approximately linear to the left of the origin and approximately exponential to the right, and conversely when $a < 0$, it will give a larger cost to the errors made. This loss is illustrated in Figure 1.3(a) and it is differentiable and convex.

Log-barrier loss. The log barrier loss assigns an infinite penalty for residuals $y - f(\mathbf{x})$ larger than c. The log barrier function is very close to the quadratic penalty for $|\frac{y-f(\mathbf{x})}{c}| \geq 0.25$. It arises in control theory and has several names [Boyd 91, Boyd 04]. With limit $c > 0$, it has the form:

$$\ell(f, \mathbf{x}, y) = \begin{cases} -c^2 \log\left(1 - \frac{(y-f(\mathbf{x}))^2}{c^2}\right) & \text{if } |y - f(\mathbf{x})| < c \\ 0 & \text{if } |y - f(\mathbf{x})| \geq c \end{cases} .$$

Input-dependent loss. This kind of loss functions depend on another function $\widetilde{\ell}$ which accounts for input-dependence.

$$\ell(f, \mathbf{x}, y) = \begin{cases} 0 & \text{if } y = f(\mathbf{x}) \\ \widetilde{\ell}(f, \mathbf{x}, y) & \text{otherwise} \end{cases} .$$

The (0-1)-loss is a special case of this kind of functions that can equivalently be expressed as:

$$\ell(f, \mathbf{x}, y) = \frac{|\text{sign}(f(\mathbf{x})) - y|}{2}.$$

Soft margin loss (Hinge loss). In this case, a confidence on the prediction can be included and depends on the product $yf(\mathbf{x})$ to assess the quality of the estimate. It is defined as:

$$\ell(f, \mathbf{x}, y) = (1 - yf(\mathbf{x}))_+ = \max\{0, 1 - yf(\mathbf{x})\}. \tag{1.5}$$

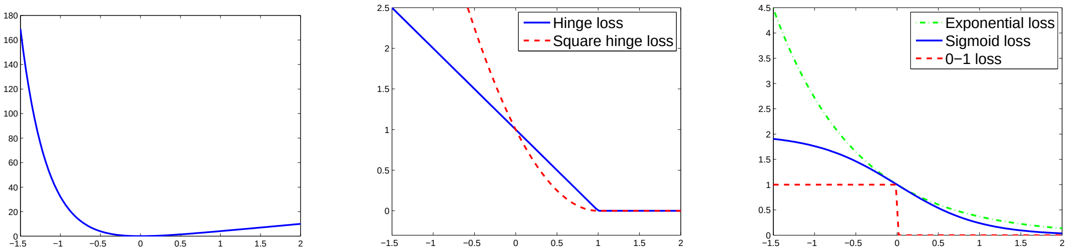
In some cases, to make this function easier to minimize, the square version is used:

$$\ell(f, \mathbf{x}, y) = \max\{0, 1 - yf(\mathbf{x})\}^2.$$

These loss functions are illustrated in Figure 1.3(b).

Logistic loss. This is a convex differentiable loss, which can be used to associate probabilities of belonging to a particular class: $\ell(f, \mathbf{x}, y) = \log\left(1 + \exp^{-yf(\mathbf{x})}\right)$.

Upper bounds for the (0-1)-loss. Several of the aforementioned loss functions are in fact what is called a **surrogate** loss function of the (0-1)-loss function, that is, a convex upper bound for the (0-1) loss. There exists other continuous differentiable examples like the ***exponential loss***: $\ell(f, \mathbf{x}, y) = \exp(-yf(\mathbf{x}))$, and also non-convex ones like the ***sigmoid loss*** function: $\ell(f, \mathbf{x}, y) = 1 - \tanh(yf(\mathbf{x}))$ which are an upper bound of the (0-1)-loss, see Figure 1.3(c).



(a) Linex loss with $a = -3, b = 2$ (b) Hinge loss and Square hinge loss (c) (0-1)-loss vs. sigmoid and exponential loss

Figure 1.3: Classification loss functions, the horizontal axis indicates the value of $yf(\mathbf{x})$ while the vertical axis is the output value for the loss function.

The following table makes a summary about the characteristics of the previous loss functions for classification

	Differentiable	Singular
Convex	Square hinge loss Linex loss Logistic loss Square hinge loss	Hinge loss
Non-convex	Sigmoid loss	0-1 loss Log-barrier loss

Table 1.1: Summary of classification losses properties.

In addition to the loss functions for classification, there are several ones for the regression case which is a learning problem defined with $\mathcal{Y} = \mathbb{R}$. The most common loss functions are:

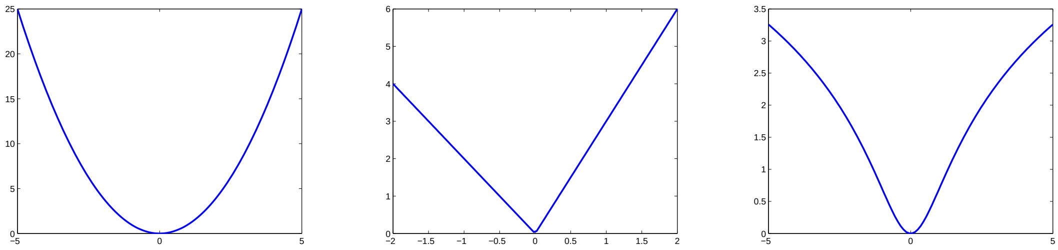
Square loss. Defined as: $\ell(f, \mathbf{x}, y) = (f(\mathbf{x}) - y)^2$, resulting in a convex and differentiable loss, illustrated in Figure 1.4(a).

Linlin (Asymmetric) loss. An asymmetric loss function for the regression case is the *linlin* [Chri 97] (Figure 1.4(b)):

$$\ell(f, \mathbf{x}, y) = \begin{cases} a|y - f(\mathbf{x})| & \text{if} \quad y - f(\mathbf{x}) > 0 \\ b|y - f(\mathbf{x})| & \text{otherwise} \end{cases}.$$

ε-insensitive. The ε-insensitive loss proposed by Vapnik allows errors in the approximating function without actual increase in loss. This was introduced in the support vector machines: $\ell(f, \mathbf{x}, y) = \max\big\{0, |f(\mathbf{x}) - y| - \varepsilon\big\}$, together with its quadratic analogous [Loog 04] which are both convex and the first one not differentiable.

Cauchy loss. Also useful for outliers detection : $\ell(f, \mathbf{x}, y) = \log(1 + (f(\mathbf{x}) - y)^2)$ [Stew 03], resulting in a non-convex differentiable regression loss, as can be seen in Figure 1.4(c).



(a) Square loss

(b) Linlin loss with $a = 3, b = 2$

(c) Cauchy loss

Figure 1.4: Regression loss functions, the horizontal axis indicates the value of $y - f(x)$ while the vertical axis is the output value for the loss function.

	Differentiable	Singular
Convex	Square loss	Linlin loss ϵ-insensitive loss
Non-convex	Cauchy loss	L_p square root

Table 1.2: Summary of regression loss functions properties.

All these functions can be plugged into the empirical risk minimization setup. The characteristic of the loss function will help to determine which resolution method is suitable. The advantage of the use of a convex function is that it will guarantee that an efficient learning algorithm will output a global minimum, while the interest of a non-differentiable loss lies on the fact that it opens the possibility to reach sparse solutions.

1.1.4 Complexity criteria: Regularization function

The objective function of Problem 1.1 is composed of a regularization function and a loss function. Particularly, the first one aims at measuring the complexity or regularity of the model.
There exists several ways of measuring the complexity of a function, again the convexity and differentiability plays an important role in the resolution algorithm choice.
In the case that $f \in \mathcal{H}$ with $\mathcal{H}$ a RKHS, owing to the representer theorem the decision function can be written as follows: $f(\mathbf{x}) = \sum\limits_{i=1}^{m} \alpha_i \mathbf{k}(\mathbf{x}, \mathbf{x}_i)$ and the complexity can be measured by a norm. Let matrix K with $K_{ij} = \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j), i, j = [\![m]\!]$ be the Gram matrix, the most used regularization functions Ω are listed below:

- $L^2_{\mathcal{H}}$ **norm**: $\Omega(f) = \|f\|^2_{\mathcal{H}} = \boldsymbol{\alpha}^\top K \boldsymbol{\alpha}$ is a differentiable convex function.

However, the decision function f can be viewed as an expansion over a basis $\{\mathbf{k}(\mathbf{x}_i, \cdot)\}$. Therefore, one can intend to address the complexity of f by considering some penalizations over the coefficients of the decomposition. This leads, for instance, to the following regularizers that are usually considered in signal processing community [Chen 98].

- L_2^2 **norm**: $\Omega(f) = \boldsymbol{\alpha}^\top \boldsymbol{\alpha}$ is the L_2-norm in the case that the decision function f is considered as a linear combination of elements of a dictionary of functions (see Figure 1.5 for illustration of this regularizer).
- L_1 **norm**: $\Omega(f) = \sum_{i=1}^{m} |\alpha_i|$, useful to measure the sparsity of a decision function. It is a convex but non-differentiable regularizer and is illustrated in Figure 1.5. This regularization function coupled with the square loss is well known in the signal processing community [Dono 03a] and in the statistical community as the lasso problem [Tibs 96].
- **Bridge penalty** (L_q^q) [Fu 00]: defined for a full-rank matrix M such that $\Omega(f) = \sum_{i=1}^{m} |\beta_i|^q$ with $\boldsymbol{\beta} = M^\top \boldsymbol{\alpha}$. This penalty is also called L_q ***norm*** with $\Omega(f) = \left(\sum_{i=1}^{m} |\beta_i|^q \right)^{1/q}$. This penalty corresponds to a general setting including the L_2^2 and L_1 regularizer. It is not convex and non differentiable if $q < 1$. This norm with $q = \frac{1}{2}$ and M the identity matrix is depicted in Figure 1.5.
- L_∞ **norm**: $\Omega(f) = \max_i\{|\alpha_i|\}$ (all vector in $\mathbb{R}^2$ with supremum norm equal to 1 can be seen in Figure 1.5).
- L_0 **norm**: this norm counts the number of active variables: $\Omega(f) = \sum_{\substack{i=[\![m]\!] \\ \alpha_i \neq 0}} 1.$
- **Sum of component wise penalizations**: Except the case of $L_{\mathcal{H}}^2$ a Bridge, all the previous regularization terms can be cast as the sum of component-wise penalties. If $\rho : \mathbb{R} \to \mathbb{R}^+$ is a penalty function, then, a global penalty can be built by letting: $\Omega(f) = \sum_{i=1}^{m} \rho(\alpha_i)$.
- **Sum of group-wise penalizations**: instead of being component-wise, the penalization is defined as:
$$\Omega(f) = \sum_j \rho(\|\boldsymbol{\alpha}_j\|)$$
where $\|\cdot\|$ represents any norm and the vectors $\boldsymbol{\alpha}_j$ are any (overlapping or not) subset of $\boldsymbol{\alpha}$.

Some properties of these regularizers can be mentioned:

- The L_2 norm tends to advantage nicely smoothed decision functions.
- The L_q norm with $q > 2$ exhibits also such kind of behavior. For $q = \infty$, it forces the parameters to have similar value.
- When the sparsity is concerned, the L_1 norm, the L_q ($q < 1$) pseudo norm are appealing. The extreme case of L_q penalization is the L_0 but the latter leads to a combinatorial problem to be solved.

An extended study of the univariate case can be consulted in [Anto 09].

1.1.5 Multi-criteria Optimization

The aim in this section is to review the different settings involving a loss function and a regularization function.

A multi-objective optimization problem is formalized as:

$$\min_{f \in \mathcal{H}} \begin{cases} \mathcal{L}(f, S) \\ \text{and} \\ \Omega(f) \end{cases} \qquad \text{also denoted} \qquad \min_{f \in \mathcal{H}} \{\mathcal{L}(f, S), \Omega(f)\} \tag{1.6}$$

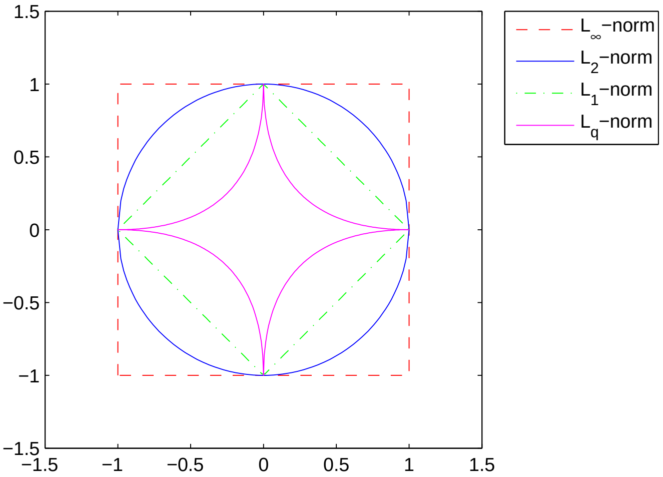


Figure 1.5: Different norms as regularization function. For $\mathcal{H} = \{f(\mathbf{x}) = a_1\mathbf{k}(\mathbf{x}, \mathbf{x}_1) + a_2\mathbf{k}(\mathbf{x}, \mathbf{x}_2)\}$, all vectors with norm equal to 1 are represented for each norm, in this example, $q = \frac{1}{2}$ and $M = I$ for the bridge penalty.

where S_X is the set of sample features with corresponding labels in S_Y. In this case, making some abuse of notation, we will denote, $\mathcal{L}(f, S)$ the empirical risk measure on sample (S_X, S_Y). This formulation sets a conflict in the goal because both objective functions cannot be at their minimum at the same point, which is the case in learning problems. In this situation, a certain trade-off has to be achieved. An elegant way to represent this trade-off is the Pareto frontier.

Pareto Frontier

As in the previous sections, the analysis relies on the relationship between hypothesis f and its associated objective functions $\mathcal{L}(f, S)$ and $\Omega(f)$ defined for a given sample. The ***feasible objective set*** is the set of couples $\big(\mathcal{L}(f, S), \Omega(f)\big)$ obtained for all $f \in \mathcal{H}$. Using this set, elements of $\mathcal{H}$ can partially be ordered according to Pareto dominance, a notion defined hereafter [Boyd 04].

Definition 1.11 (Pareto dominance)**.** *For a given sample S, a hypothesis $f \in \mathcal{H}$* ***dominates*** *a hypothesis $g \in \mathcal{H}$ in the sense of Pareto when $\mathcal{L}(f, S) < \mathcal{L}(g, S)$ and $\Omega(f) < \Omega(g)$.*

This definition can be generalized to more than two objective functions. From this definition, a notion of optimality can be defined: it is called the ***Pareto frontier***.
There are two ways of defining the Pareto optimal set depending on the optimality highlighted.

Definition 1.12 (Pareto optimal solution)**.** *A hypothesis $f \in \mathcal{H}$ is said to be a* ***Pareto optimal solution in*** *$\mathcal{L}$ if there exists a non-negative scalar C such that $\Omega(f) \leq C$ and $\mathcal{L}(f, S) \leq \mathcal{L}(g, S)$ $\forall g \in \mathcal{H}$ holding $\Omega(g) \leq C$.*
Analogously, a hypothesis $f \in \mathcal{H}$ is said to be a ***Pareto optimal solution in*** *Ω if there exists a non-negative scalar C' such that $\mathcal{L}(f, S) \leq C'$ and $\Omega(f) \leq \Omega(g)$ $\forall g \in \mathcal{H}$ holding $\mathcal{L}(g, S) \leq C'$.*

We will be interested in the set of solutions that are given when varying parameter C and the solutions cannot be compared between themselves in terms of joint minimization.

Definition 1.13 (Pareto frontier)**.** *The* ***Pareto frontier*** *(also called Pareto's optimal trade-off curve) of objective functions $\mathcal{L}$ and Ω is the set of optimal values $\big(\mathcal{L}(f, S), \Omega(f)\big)$ obtained for all Pareto optimal solutions f.*

The Pareto frontier is a part of the hull of the feasible objective set. An example of a Pareto frontier and Pareto dominance is given in Figure 1.6. Its is worth mentioning that Pareto frontier is a convex curve if $\mathcal{L}$ and Ω are both convex functions. It looks reasonable to assume the solution of the learning problem is Pareto optimal. The interest of this concept lies on the fact that the calculation of the regularization path will gives us exactly this frontier. Before analyzing under which conditions the equivalence between the regularization and Pareto frontier holds, let focus on the different formulations of the multi-objective minimization.

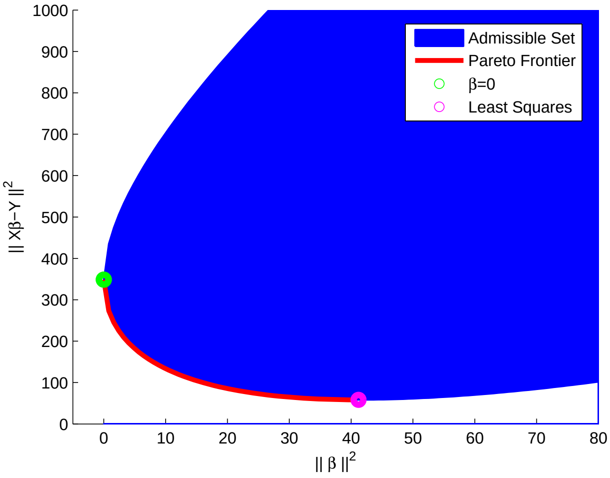


Figure 1.6: Pareto Dominance and Pareto frontier. In the case of a square loss function and a quadratic norm, any point in the blue area is less desirable than any of the points lying in the red curve.

Ivanov, Morozov and Tikhonov's regularization

If the Pareto frontier is to be calculated, there are two formulations of the multi-criteria problem to do it. Sometimes they are referred as ***Ivanov*** ($\mathcal{I}$) [Ivan 76] or ***Morozov*** [Moro 84] ($\mathcal{M}$) regularization, respectively:

$$(\mathcal{I}) \quad : \quad \left\{ \begin{array}{ll} \min\limits_{f\in\mathcal{H}} & \mathcal{L}(f,S) \\ \text{s.t.} & \Omega(f) \leq C \end{array} \right. \qquad\qquad (\mathcal{M}) \quad : \quad \left\{ \begin{array}{ll} \min\limits_{f\in\mathcal{H}} & \Omega(f) \\ \text{s.t.} & \mathcal{L}(f,S) \leq C'. \end{array} \right.$$

The Ivanov formulation corresponds to the Pareto optimal solution in $\mathcal{L}$. The Morozov one corresponds to the Pareto optimal solution in Ω.
When C varies, the solution of problem ($\mathcal{I}$) move along the Pareto frontier. Similarly, when C' varies, the solution of problem ($\mathcal{M}$) will also move along the frontier. This set of solutions is called the regularization path.

Definition 1.14 (Regularization path ($\mathcal{I}$ and $\mathcal{M}$))**.** *The **regularization path** of problem* ($\mathcal{I}$) *(respectively* ($\mathcal{M}$)*) is the set of obtained solutions when varying C (respectively C').*

There is another popular way of mixing objective functions known as ***Tikhonov*** regularization ($\mathcal{T}$) [Tikh 79] defined as follows:

$$(\mathcal{T}) \quad : \quad \min_{f\in\mathcal{H}} \quad \mathcal{L}(f,S) \; + \; \lambda\,\Omega(f) \tag{1.7}$$

for some $\lambda \in \mathbb{R}^+$. A global optimization function is then defined as: $J(f,S) = \mathcal{L}(f,S) + \lambda\Omega(f)$.

Definition 1.15 (Regularization path ($\mathcal{T}$))**.** *The* ***regularization path*** *of problem in Equation* (1.7) *is the set of all solutions obtained when varying* λ *over* $\mathbb{R}^+$ *i.e. Path* $= \{f_\lambda,\ \lambda \in [0, +\infty]\}$.

When both $\mathcal{L}$ and Ω are convex, the Pareto frontier is convex and therefore problems ($\mathcal{I}$), ($\mathcal{M}$) and ($\mathcal{T}$) are all equivalent [Miet 99, pages 12-13]. Indeed in this case, for any Pareto optimal point f^* there exists a $\lambda \geq 0$ such that f^* is the solution of the Tikhonov minimization problem. Nevertheless, in general, it is not always the case that the solution set of Problem ($\mathcal{T}$) (that is the regularization path) coincides with the Pareto frontier.

As optimization theory will be analyzed, we will see that there exists a large number of well supported methods to solve the Tikhonov formulation and it is therefore the usual choice in machine learning.

Note that the determination of the solution to the learning problem requires to choose the solution within Pareto's frontier or equivalently to find an optimal value for C, C' or λ depending on the formulation used. This is the ***model selection*** issue.

Notation will be simplified by letting $\mathcal{L}(f, S) = \mathcal{L}(f)$ for a fixed training set S. Given two convex functions $\mathcal{L}, \Omega : \mathcal{H} \to \mathbb{R}^+ \cup \{0\}$, we are interested in the behavior of the function

$$f^*(C) = \begin{cases} \underset{f \in \mathcal{H}}{\text{argmin}} & \mathcal{L}(f) \\ \text{s. t.} & \Omega(f) \leq C \end{cases} \tag{1.8}$$

with respect to C. To understand how the optimal function changes as C is continuously modified, it has to be noticed that in the Tikhonov's formulation the loss function ($\mathcal{L}(f)$) and the complexity function ($\Omega(f)$) vary in an inverse manner as C increases or decreases. This is shown in the following theorem:

Theorem 1.16 (Inverse Relation between the loss and the regularization function)**.** *If* $\mathcal{L}(f)$ *is strictly convex, for any* $C_0 < C$*, it holds that*

1. $\mathcal{L}(f^*(C_0)) \geq \mathcal{L}(f^*(C))$ *and*
2. $\Omega(f^*(C_0)) \leq \Omega(f^*(C))$.

Proof. Let $\Gamma(C) = \{f \in \mathcal{H} \mid \Omega(f) \leq C\}$ be the convex set of feasible solutions for C and

$$f^*(C) = \underset{f \in \Gamma(C)}{\text{argmin}}\, \mathcal{L}\big(f\big).$$

the optimal solution of this set. As $\mathcal{L}(f)$ is strictly convex, $f^*(C)$ is unique. By optimality of $f^*(C)$,

$$\mathcal{L}(f^*(C)) \leq \mathcal{L}(f(C)) \ \text{ for all } \ f \in \Gamma(C) \tag{1.9}$$

If $C_0 < C$, then

$$\Gamma(C_0) = \{f : \Omega(f) \leq C_0\} \subseteq \{f : \Omega(f) \leq C\} = \Gamma(C),$$

and since $f^*(C_0) \in \Gamma(C_0)$, it follows that $f^*(C_0) \in \Gamma(C)$.
So, statement (1.9) will be also valid for a particular $f = f^*(C_0) \in \Gamma(C)$
Therefore,

$$\mathcal{L}(f^*(C)) \leq \mathcal{L}(f^*(C_0))$$

which proves the first part of the proposition. For the second part, we observe that by definition of $\Gamma(C)$, $\Omega\big(f^*(C)\big) \leq C$. If:

i) $\Omega\big(f^*(C)\big) \geq C_0$, then $\Omega\big(f^*(C_0)\big) \leq C_0 \leq \Omega\big(f^*(C)\big)$

ii) $\Omega\big(f^*(C)\big) < C_0$, then $f^*(C) \in \Gamma(C_0)$ and therefore $f^*(C) = f^*(C_0)$, otherwise, it cannot be the minimum of problem in Equation (1.8), leading to $\Omega\big(f^*(C)\big) = \Omega\big(f^*(C_0)\big)$.

Therefore, we deduce that

$$\Omega(f^*(C)) \geq \Omega(f^*(C_0)).$$

□

A general proof with a set of minima will follow directly from the convexity of the function. This result emphasizes the fact that an additional set to the training set is necessary to make a judicious model selection and that sole use of the training error is not an accurate measure since the learning algorithm can produce a model that overfits the data. Some common strategies for model selection are: training-validation sets, cross validation, bootstrap, etc. Some of these strategies are devised in the next section.

1.2. Model Selection

The results of a decision function depend on the training samples and a set of training parameters. To ease the presentation and to make concrete the dissertation, we will consider the framework of Support Vector Machines that relies on the use of a RKHS induced by a kernel $\mathbf{k}$ as hypothesis space and the optimization of Thikonov formulation of the learning problem. Hence, the training parameters are the regularization term λ and other parameters like the hyperparameter(s) of the kernel, for instance the bandwidth of the Gaussian kernel or the degree in the polynomial kernel, to which we will refer in generic way as σ. Since these parameters determine the model, the task of finding the best parameter combination is called ***model selection***.
Another aspect of model selection includes the number of variables that will be defined and included in the model. This is as well an important issue since we will be interested in getting parsimonious models, that is, getting the best explaining model with the less variables.
There are two particular issues about model selection, the first one consists in deciding how to measure the performance of a model and the second one consists in knowing how to explore the space (λ, σ).
The first concern is how to measure a model. In an intuitive way, this question is easy to answer: a good model always gives a correct result according to the task to be performed even with samples that have never been seen, that is, the generalization ability. If the training error is driven to a very small value, but with unseen test data the test error is large, then the model suffers ***overfitting***. A theoretical approach to the generalization ability was introduced with the Vapnik-Chervonenkis dimension (VC dimension) [Vapn 79] as a way to bound the expected risk as a sum of the empirical risk and a term depending on the VC dimension and the size of training samples.
In a practical way, some methods to choose a model are the resampling methods [Efro 87] which consist in generating different sets to test the proposed models:

Training-Validation sets: in this case, a particular set for training is used while the generalization ability is tested on a different set called the validation set (strictly speaking, this is not really a resampling method).

Cross validation: consists in dividing the n samples in k sets called *folds*. Each fold of size $\frac{n}{k}$ is tested on a model trained with the remaining samples. In this way, the generalization ability of the model will be approximated. Having a set of models, they are tested on each fold and the model that gives the best average error is the one chosen. The set division can be seen as a sampling without replacement [Kurt 48].

Leave-one-out (LOO): this is a special case of cross validation setting where each test set is of size one [Elis 03, Evge 04]. This kind of sampling is used in the *jackknife* technique [Dwas 57] to estimate bias and variances of statistical estimators. The leave-one-out cross-validation procedure can be efficiently approximated in closed form for a wide variety of kernel learning methods, providing a convenient means for model selection using simplex methods or scaled conjugate gradient descent [Cawl 07].

Bootstrapping: Several training-validation sets are formed by making a random sampling of the population with replacement. This kind of sampling is used in the *bagging* technique to reduce the variance and helps to avoid overfitting [Efro 93].

The use of these methods will help us to have an idea of the generalization error that the chosen model will do, that is, we will be able to generalize better depending on the used method [Lend 03]. On the other hand, each method implies that the resolution of the learning problem will be repeated several times, the more problems have to be solved, the longest it will take to get an estimation of the generalization error (or a bound of it) but also the better this approximation will be (a narrower bound). This is summarized in Table 1.3.

Method	Estimator	Variance	Cost
Training-Validation sets	Biased	Small	Low
Cross validation	Unbiased	Large	Medium
LOO	Unbiased	Large	High†
Bootstrapping	Unbiased	Large	Medium

Table 1.3: Generalization error estimation characteristics vs. computational cost for each model selection method [Varm 06]. †The cost for the LOO method depends on the learning problem as in some cases, the decision function can be easily updated for a single exclusion of training sample.

Once a goodness measure has been chosen for a model, the next issue is to figure out how the parameter space (λ, σ) will be searched, since there is an infinite number of possibilities, it has to be done in a efficient way.
A common technique is the discretization of the search space:

Grid search: consists in selecting a finite set of values for λ and a set of values for σ, generally equally or logarithmically separated. All pairs generated by the combination of these points are then explored. Eventually this grid is refined and the process is repeated to obtain the parameter with a better precision. The built grid increases in dimension according to the number of parameters that have to be set and the parameter search on it can be very time consuming.

Random sampling: random pairs of (λ, σ) are tested to define a smaller search zone and the process is repeated in a smaller zone.

Genetic Algorithms: a set of models is encoded with a predefined precision. This will define the population, then a genetic algorithm is run and the model selection criterion is incorporated into a fitness function that uses empirical measures or theoretical bounds [Less 06] [Zhou 05].

The possible reached accuracy that is obtained in average by these methods is summarized in Table 1.4. In general, if the same number of points is to be explored, the cost of grid search and random sampling will be similar but the global optimum can be overlooked.

Method	Accuracy	Cost
Grid search	Medium	Medium
Random sampling	Medium	Medium
Genetic Algorithms	High	High

Table 1.4: Accuracy vs. Cost of discretization methods.

Other methods use continuous techniques to explore the parameter space:

Gradient Descent: a continuous approximation of the validation function is done in order to use a gradient descent approach along it [Keer 07] to choose the model that will minimize the validation error. Other approaches use the gradient to find a direction to improve the estimated generalization error and then make a correction step [Allg 90, Zang 81, Park 07, Gang 07].

Multi-kernel Optimization: the principle is to define the decision function as a convex sum of elementary functions f_j, defined each one over a prefixed RKHS, $\mathcal{H}_j$, induced by different kernels $\mathbf{k}_j$ (and by the way, different subsets of features or variables). The optimization problem attempts to retrieve the most useful combination of kernels according to the training objective function and for fixed value of λ. This formulation can be seen as group-wise penalization [Rako 07b, Sonn 06].

Parametric Quadratic Programming (PQP): regarding the set of optimization problems defined by the use of a different λ parameter have a singular form that can be transformed into a PQP problem, that is, a quadratic objective function and linear equality and inequality constraints. The complete set of optimal solutions turns out to be piecewise linear [Hast 04, Ross 07b, Wang 06b].

Bilevel Optimization: in the bilevel optimization problems, constraints in the principal optimization problem consist in sub-optimization problems with constraints, named hierarchical optimization. This kind of problems has already been studied in the operations research literature [Brac 73] and in the machine learning framework [Kuna 08].

Table 1.5 compares if in general the global optimum is reached by the use of the previous methods, the cost of each method usage and the problem universe they are applied to. The multi-kernel optimization was left out of this comparison because this method will output a combination of methods rather than a single one.

Method	Global Optimum	Cost	Problem Universe
Gradient Descent	No	Low	Derivable validation functions
PQP	Yes	Low to Medium	QP problems
Bilevel Optimization	No (non-convex)	Medium	Hierarchical optimization

Table 1.5: Global Optimality, computational cost and applicability for each model selection method. For PQP, parameter σ is considered fixed.

Model selection is a fundamental step to construct a model. In the statistical learning theory, a model selection framework aims at approximating the generalization error and the overfitting degree. The more accurately these are estimated, the better the chosen model will turn out to be. When dividing the dataset into subsets, model fitting must be done independently from the training data in order to avoid model selection bias.

1.3. Some SVM Learning Problem Examples

Once the necessary concepts of learning theory have been stated, in this section some of the most commonly used frameworks are described. We start with the classification framework revised in an interesting functional manner. The ranking problem is later illustrated and set as a learning problem that can again be written as an optimization problem. As explained previously, we will restrict our presentation to the SVM framework.

1.3.1 Classification Learning Problem

Classification is a particular kind of learning problems that belongs to the kind of ***Supervised Learning***. A set of samples or points $\{\mathbf{x}_i\}_{i=[\![n]\!]}$ is given, where $[\![n]\!] = \{1, ..., n\}$; it is assumed that these samples belong to a vector space, denoted by $\mathcal{X}$. Additionally, a corresponding set of outputs or labels $\{y_i\}_{i=[\![n]\!]}$ is also given and these samples belong to $\mathcal{Y}$.

We will initially deal with a two-class problem, where $|\mathcal{Y}| = 2$ and w.l.o.g. we can assume that $\mathcal{Y} = \{+1, -1\}$. An example of a two class problem is depicted in Figure 1.7. We assume that the magenta triangles represent the positive class and the blue circles the negative one. A decision

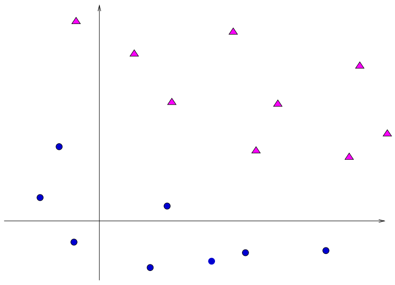


Figure 1.7: Example of a two-class problem

function $f(\mathbf{x})$ is sought so that it keeps all the positive samples on the subspace defined by $f(\mathbf{x}_i) > 0$, $\forall i, y_i > 0$ and the negative samples on the subspace defined by $f(\mathbf{x}_i) < 0 \quad \forall i, y_i < 0$. In a classification problem, we can start by having only two classes, but it is possible to extend a two-class classifier to a multi-class classifier. There exists several principles to do this extension, some of them consist in dividing the training samples into several subsets with only two classes to be able to train a binary classifier with each subset. Some of the alternatives assign points according to these classifiers and transform these points into probabilities of belonging to a particular set. A list of the most common methods to deal with more than two classes is given below.

One vs. One score If there are p classes, $\binom{p}{2}$ binary classifiers are pairwise trained for this problem [Furn 02]. To classify a new sample, this is tested in all binary models, resulting in $\binom{p}{2}$ proposed classes, which will be counted as a point for each winning class, points are summed by class and the sample will be labeled with the class having more points (majority vote). An example of this classifier with four classes is depicted in Figure 1.8, where there are $p = 4$ classes and 6 binary classifiers are trained.

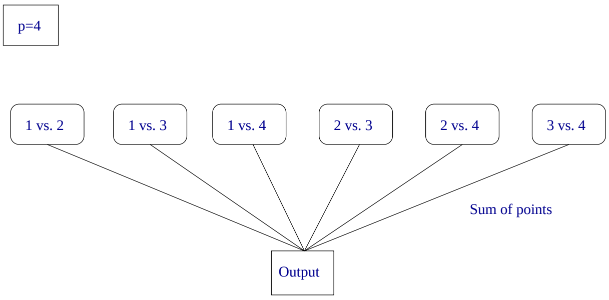


Figure 1.8: One versus one classifier for 4 classes

One vs. Rest score p different binary classifiers are trained, each one is trained to distinguish the examples in a single class from the examples in all remaining classes [Rifk 04]. When it is desired to classify a new example, the p classifiers are run, and the classifier which outputs the largest (most positive) value is chosen. Figure 1.9 depicts an example of a one-vs.-rest classifier with $p = 4$ classes, resulting in 4 binary classifiers.

DAG A directed acyclic graph is built where each node is a binary classifier that will lead to a choice of a smaller set of classes. Each leaf of the graph outputs exactly one class. A new point is then classified according to the decision path induced by the graph [Plat 00]. Figure 1.10 illustrates this method.

Probabilistic Outputs These methods assign to a new sample a probability of belonging to each class either by training a logit function with a maximum likelihood score [Hast 98] or

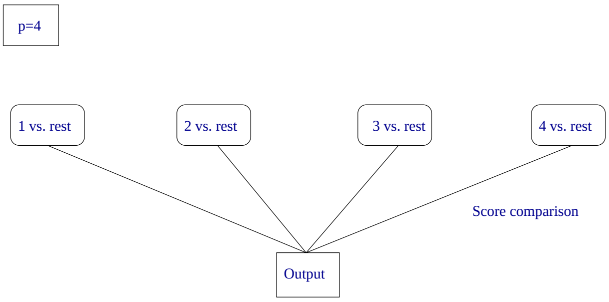


Figure 1.9: One versus rest classifier for 4 classes

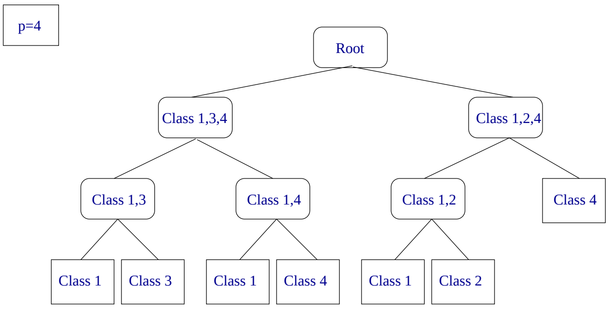


Figure 1.10: Example of a DAG for multi-class with 4 classes

by training an SVM that will be later map with a sigmoid function into probabilities. The assigned class is the one with the largest probability [Plat 99b].

Global A general problem is posed where all classes are trained together via an optimization problem with multiple criteria [Bred 99, Lee 04, West 98, Zhan 04]. Multi-class SVMs have been proposed [Guer 07] where a loss function over all classes is used, the decision function is a function with vectorial output (an entry for each class) and a hyperplane with maximum distance to all classes is searched .

There are many different methodologies which are widely documented, these have been compared, in terms of consistency and theoretical properties. Different works [Duan 05, Hsu 02, Tewa 05, Rifk 04] summarized done comparisons. According to Bach [Bach 08], the winner is still unclear. To solve the problem of binary classification, the SVM framework will be considered. It deals with binary problems by stating a problem issued from a trade-off multiple criteria, as next explained.

Functional formulation of the SVM classifier (SVC)

This learning problem considers a function f that belongs to a RKHS with kernel $\mathbf{k}$. The loss function is the hinge loss function and the used regularization term is the norm of the function in the RKHS. If $\mathcal{Y} = \{1, -1\}$ and we define $f(\mathbf{x}) = f_0(\mathbf{x}) + b$, it has to be noticed that the hinge loss function in Equation (1.5):

$$\ell(f, \mathbf{x}, y) = \max\{0, 1 - yf(\mathbf{x})\} \tag{1.10}$$

can be rewritten as follow:

$$\ell(f, \mathbf{x}, y) \quad = \quad \left\{ \begin{array}{ll} \underset{\xi}{\text{minimize}} & \xi \\ \text{subject to} & y(f(\mathbf{x})) \geq 1 - \xi \\ & \xi \geq 0 \end{array} \right. \tag{1.11}$$

The generalization ability will be controlled by adding a regularization term as the norm in the Hilbert space. We can therefore establish the following classification problem:

Problem 1.17 (Functional SVM Classifier Primal Problem). *Let $S = \{S_X, S_Y\}$ be a set of samples $S_X = \{\mathbf{x}_i\}_{i=[\![n]\!]}$ and $S_Y = \{y_i\}_{i=[\![n]\!]}$, $y_i \in \{+1, -1\}, i = [\![n]\!]$. Let $f_0 \in \mathcal{H}$ where $\mathcal{H}$ is a RKHS with kernel $\mathbf{k}$ and induced Gram matrix $K_{ij} = \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j), i, j = [\![n]\!]$. The primal optimization problem under a functional regularization framework with bias b is defined as follows:*

$$\begin{array}{rl}
\min\limits_{f_0 \in \mathcal{H}, b \in \mathbb{R}, \boldsymbol{\xi} \in \mathbb{R}^n} & \sum\limits_{i=1}^{n} \xi_i + \frac{\lambda}{2} \|f_0\|^2_{\mathcal{H}} \\
\text{s. t.} & y_i(f_0(\mathbf{x}_i) + b) \geq 1 - \xi_i, \; i = [\![n]\!] \\
& \xi_i \geq 0, \; i = [\![n]\!]
\end{array}$$

Remark It can be observed that, in fact, this problem is equivalent to the classical one expressed with C, where the objective function has weight C in the loss function and weight 1 in the regularization one. To obtain Problem 1.17 from Problem A.30, it is enough to do $C = \frac{1}{\lambda}$.

1.3.2 Ranking Learning Problem

Ranking algorithms address document retrieval applications where the main objective is to provide in the top list, the most relevant documents according to the given user query. The aim is that the more important a result is to the user, the closer to the top of the list it should be. For this sake, the learning to rank algorithms use stored information coming from the feedback of the user on previous queries to build a model able to achieve a satisfactory generalization rate.

To formally state the Ranking SVM algorithm, some precisions have to be done. Preference ordering is determined by user defined queries $\{q_i\}_{i=[\![N_q]\!]}$, where each one is expressed as a feature vector in $\mathbb{R}^Q$, and a list of possible associated documents $\{d_j\}_{j=[\![N_d]\!]}$, stated as a feature vector in $\mathbb{R}^D$, in this case, N_q and N_d indicates the number of available queries and documents, respectively. The searched decision function f will work as a scoring function, giving higher value to the most relevant documents according to the given query. An accuracy measure for ranking should emphasize errors at the beginning of the permuted list rather than at the end of the list. The proposed normalized discounted cumulative gain (NDCG) [Jarv 00] aims at that kind of scoring. A permutation π of documents $\{d_i\}$ is searched so that most important documents are placed at the beginning of the list [Jarv 00], if a function f is given, the reordering will be done according to the application of this function for each pair. The ranking problem can be stated as follows:

Problem Statement

Problem 1.18 (Ranking Problem). *Let (q_i, d_j, y_{ij}) be a triplet consisting of a vector $q_i \in \mathbb{R}^Q, i = [\![N_q]\!]$ containing query features, a vector $d_j \in \mathbb{R}^D, j = [\![N_d]\!]$ that includes document features and $y_{ij} \in \mathbb{R}$ which is its corresponding ranking with respect to query q_i. The* ***ranking problem*** *aims at finding a decision function $f : \mathbb{R}^Q \times \mathbb{R}^D \to \mathbb{R}$ that given query q_i and document d_j predicts the ranking or relevance y_{ij} of the document for the particular query q_i taking into account previous seen samples.*

For clarity and simplification sakes, let us consider an example of web pages search in ranking problems. To this purpose, a query-document sample feature $\mathbf{x} = (q, d) \in \mathcal{X}$ will be built, together with its corresponding label y that indicates the relevance of document d with respect to query q, so that if a document d' is less relevant than d with respect to q it will hold $y' < y$ with $\mathbf{x}' = (q, d')$. The previous relation can be restated as a directed graph in the sense that a relation of relevance can be seen as an edge e from $\mathbf{x}$ to $\mathbf{x}'$ where its direction indicates the sense where relevance decrease. Therefore, a directed acyclic graph (X, E) can be built for each query, with $X \subseteq \mathbb{R}^Q \times \mathbb{R}^D$ and $E \subseteq X^2$, $|E| = m$. The edges are of the form $e = ((q, d), (q, d')) = (\mathbf{x}, \mathbf{x}')$ with $d, d' \in \mathbb{R}^D$ and $q \in \mathbb{R}^Q$. It has to be noticed that the transitive property is respected, so that if $y > y'$ and $y' > y''$ then $y > y''$.

To illustrate this induced graph, we are going to consider an example with $N_q = 3$, $N_d = 12$. The documents are rated with three degrees of relevance as stated in Table 1.6. Using this example,

query	document	rank	coding	final representation
q_1	d_4	1	$\mathbf{x}_1 = (q_1, d_4), y_1 = 1$	$(\mathbf{x}_1, 1)$
q_1	d_5	2	$\mathbf{x}_2 = (q_1, d_5), y_1 = 2$	$(\mathbf{x}_2, 2)$
q_1	d_{12}	1	$\mathbf{x}_3 = (q_1, d_{12}), y_1 = 1$	$(\mathbf{x}_3, 1)$
q_2	d_1	1	$\mathbf{x}_4 = (q_2, d_1), y_1 = 1$	$(\mathbf{x}_4, 1)$
q_2	d_3	2	$\mathbf{x}_5 = (q_2, d_3), y_1 = 2$	$(\mathbf{x}_5, 2)$
q_2	d_5	1	$\mathbf{x}_6 = (q_2, d_5), y_1 = 1$	$(\mathbf{x}_6, 1)$
q_2	d_6	3	$\mathbf{x}_7 = (q_2, d_6), y_1 = 3$	$(\mathbf{x}_7, 3)$
q_2	d_7	2	$\mathbf{x}_8 = (q_2, d_7), y_1 = 2$	$(\mathbf{x}_8, 2)$
q_2	d_9	1	$\mathbf{x}_9 = (q_2, d_9), y_1 = 1$	$(\mathbf{x}_9, 1)$
q_2	d_{10}	3	$\mathbf{x}_{10} = (q_2, d_{10}), y_1 = 3$	$(\mathbf{x}_{10}, 3)$
q_2	d_{11}	3	$\mathbf{x}_{11} = (q_2, d_{11}), y_1 = 3$	$(\mathbf{x}_{11}, 3)$
q_3	d_2	2	$\mathbf{x}_{12} = (q_3, d_2), y_1 = 2$	$(\mathbf{x}_{12}, 2)$
q_3	d_6	1	$\mathbf{x}_{13} = (q_3, d_6), y_1 = 1$	$(\mathbf{x}_{13}, 1)$
q_3	d_8	1	$\mathbf{x}_{14} = (q_3, d_8), y_1 = 1$	$(\mathbf{x}_{14}, 1)$
q_3	d_{10}	2	$\mathbf{x}_{15} = (q_3, d_{10}), y_1 = 2$	$(\mathbf{x}_{15}, 2)$

Table 1.6: Example of a ranking problem with queries $\{q_i\}_{i=[\![3]\!]}$, documents$\{d_i\}_{i=[\![12]\!]}$ and ranks $\{y_i\} \in \{1,2,3\}$, $i = [\![15]\!]$. The third column assigns a new encoding for the query-document ranking that will be used to form the directed graph.

the induced directed graph will look as in Figure 1.11.

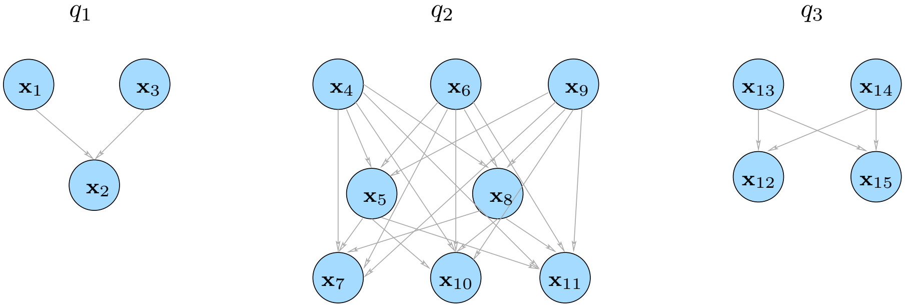


Figure 1.11: Induced directed graph from ranking problem stated in Table 1.6.

This kind of problems has gain major attention given the nowadays amount of available information. This is without doubt a challenging task in the medium and large scale context.
Several methods have been proposed to solve these problems. For the passive setting, the Rankboost algorithm ([Freu 03]) is an adaptation of the Adaboost algorithm to the ranking problem. This is a boosting algorithm which works by iteratively building a linear combination of several "weak" algorithms to form a more accurate algorithm. The Pranking algorithm ([Cram 02]) implements an online update of the ranking function similarly to the perceptron algorithm for classification. The RankSVM [Herb 00] and SVRank [Cort 07] algorithms are the adaptation for ranking of the classification and regression Support Vector machines, respectively, while the MPRank ([Cort 07]) is a magnitude-preserving algorithm, which searches not only to keep the relative position of each sample but also to preserve the given distance by the correct ordering. This last algorithm has as well the form of a regularization problem like the two previous ones with a different cost function.

Functional formulation of the RankSVM

The Ranking SVM (RankSVM) algorithm was proposed by [Herb 00] and [Joac 02] as an optimization problem with constraints given by the induced graph of the ordered queries' results. This

algorithm belongs to the family of kernel algorithms of the SVM type ([Bose 92, Schl 01]).
Similar to the classification problem, a decision function $f \in \mathcal{H}$ is searched so that if for query q, document d_{u_i} is more relevant than document d_{v_i}, that is, $y_{u_i} > y_{v_i}$ then, feature vectors $\mathbf{x}_{u_i} = (q, d_{u_i})$ and $\mathbf{x}_{v_i} = (q, d_{v_i})$ are formed and the searched decision function should hold:

$$f(\mathbf{x}_{u_i}) \geq f(\mathbf{x}_{v_i}), \qquad \forall i = [\![m]\!]$$

where m is the resulting number of composed feature vectors $\mathbf{x}_i$. Again, flexibility will be given by allowing some errors and introducing them in the objective function as follows:

$$\min_{f \in \mathcal{H}} \sum_{i=1}^{m} \xi_i, \quad \text{subject to } f(\mathbf{x}_{u_i}) - f(\mathbf{x}_{v_i}) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i = [\![m]\!]$$

A regularization term is included in the formulation to assure good generalization ability so that the Ranking SVM algorithm is stated with the following optimization primal problem:

$$\begin{array}{rll} \displaystyle\min_{f \in \mathcal{H}, \boldsymbol{\xi} \in \mathbb{R}^m} & \boldsymbol{\xi}^\top \mathbb{I} + \frac{\lambda}{2} \|f\|^2_{\mathcal{H}} & \\ \text{s. t.} & f(\mathbf{x}_{u_i}) - f(\mathbf{x}_{v_i}) \geq 1 - \xi_i & \forall i = [\![m]\!] \\ & \xi_i \geq 0 & \forall i = [\![m]\!]. \end{array}$$

with the same conventions as in Problem 1.17 As in SVM for classification, the slack variables ξ_i, $(u_i, v_i) \in E$ correspond to the cost of the constraints violation (u_i, v_i). The final document order is then obtained by sorting X according to f and resolving ties randomly.

1.4. Constrained Optimization Theory

In the previous section, a learning problem was stated with different formulations. The Tikhonov one is the general choice of problem setting, additional constraints are added to target the desired decision and all together form an ***optimization problem*** (OP) with ***constraints***. The advantage of using this OP is that the necessary conditions for a point to be optimal have already been very well studied. The other interest of this section relies on the fact that due to Equation (1.11), the widely used hinge loss function in SVM can be turned into a minimization problem under inequality constraints. Therefore, to derive the necessary optimality conditions, an overview of constrained optimization is potentially useful. Constrained optimization theory will be reviewed in this section, more details can be found in [Luen 69, Luen 05, Noce 99, Bonn 06].
The developed theory applies to minimization problems since a maximization problem can be trivially transformed into a minimization one by optimizing the negative of the objective function. We will focus our interest on constraint optimization problems with inequalities. In this section, $\mathbf{z}$ represents a vector in a generic way.

Problem 1.19 (Constrained Optimization Problem)**.** *A **constrained optimization problem** is defined as follows:*

$$\begin{array}{rl} \underset{\mathbf{z} \in \mathcal{X}}{\text{minimize}} & J(\mathbf{z}), \\ \text{subject to} & \mathbf{h}(\mathbf{z}) \geq \mathbf{0} \\ & \mathbf{g}(\mathbf{z}) = \mathbf{0} \end{array}$$

where $J : \mathcal{X} \to \mathbb{R}^+ \cup \{0\}$ is a function called the ***objective function***, with $\mathcal{X}$ a vector space; $\mathbf{h}(\mathbf{z})$ a vector of inequality constraint functions of the form $\mathbf{h}(\mathbf{z}) = (h_1(\mathbf{z}), h_2(\mathbf{z}), ...h_{|\mathcal{I}|}(\mathbf{z}))^\top$ with $h_i : \mathcal{X} \to \mathbb{R}, i \in \mathcal{I}$ and $\mathcal{I}$ a set of index; $\mathbf{g}(\mathbf{z})$ is a vector of equality constraint functions $\mathbf{g}(\mathbf{z}) = (g_1(\mathbf{z}), g_2(\mathbf{z}), ...g_{|\mathcal{E}|}(\mathbf{z}))^\top$ with $g_i : \mathcal{X} \to \mathbb{R}, i \in \mathcal{E}$ and $\mathcal{E}$ is again a set of index.
The special case with a convex objective function is largely treated in the work of Boyd [Boyd 04]. Problem 1.19 aims at minimizing function J under a reduced domain defined by functions $h_i, i \in \mathcal{I}$ and $g_i, i \in \mathcal{E}$ called the ***feasible set***. The solution of Problem 1.19 can be characterized under

certain conditions. We say that we are at an ***optimal*** point $\mathbf{z}^*$ if $\mathbf{h}(\mathbf{z}^*) \geq \mathbf{0}, \mathbf{g}(\mathbf{z}^*) = \mathbf{0}$ and $J(\mathbf{z}^*) \leq J(\mathbf{z})$ for all $\mathbf{z}$ such that $\mathbf{h}(\mathbf{z}) \geq \mathbf{0}$ and $\mathbf{g}(\mathbf{z}) = \mathbf{0}$.
An inequality h_i is said to be ***active*** at point $\mathbf{z}$ if its evaluation at this point reaches the equality, that is, if $h_i(\mathbf{z}) = 0$. The ***active set*** $\mathcal{A}(\mathbf{z})$ in a constrained problem at any feasible $\mathbf{z}$ is the union of all equality constraints and all active inequality constraints, that is,

$$\mathcal{A}(\mathbf{z}) = \{i \in \mathcal{I} | h_i(\mathbf{z}) = 0\} \cup \mathcal{E} \tag{1.12}$$

This set is relevant for the final solution since the constraints that are not active will not impose a condition in the solution and are not useful as will be seen later.

1.4.1 Derivation of the Optimality Conditions

Some assumptions about the optimal solution have to be done in order to find optimality conditions. These conditions exploit the notion of gradient that is next developed. Algorithms taking advantage of these conditions will be design to find the solutions.

Brief review of gradient notions

Definition 1.20 (Directional Derivative)**.** *It $\mathcal{X}$ is a vector space, the **directional derivative** of a scalar function $f(\mathbf{z}) : \mathcal{X} \to \mathbb{R}$ along a vector $\mathbf{d}$ is the function defined by the limit:*

$$\nabla_{\mathbf{d}} f(\mathbf{z}) = \lim_{\epsilon \to 0} \frac{f(\mathbf{z} + \epsilon \mathbf{d}) - f(\mathbf{z})}{\epsilon}.$$

Sometimes authors write $D_{\mathbf{d}}$ instead of $\nabla_{\mathbf{d}}$. If the function f is differentiable at $\mathbf{z}$, then the directional derivative exists along any vector $\mathbf{d}$, and one has

$$\nabla_{\mathbf{d}} f(\mathbf{z}) = \langle \nabla f(\mathbf{z}), \mathbf{d} \rangle$$

where the ∇ on the right denotes the gradient and $\langle \cdot, \cdot \rangle$ is the Euclidean inner product. At any point $\mathbf{z}$, the directional derivative of f intuitively represents the rate of change in f along $\mathbf{d}$ at point $\mathbf{z}$. Usually, the taken direction are normalized, so $\mathbf{d}$ is a unit vector, although the definition above works for arbitrary (even zero) vectors.

Definition 1.21 (Gateaux differential)**.** *Let $\mathbf{z} \in \Omega \subset \mathcal{X}$, $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$, with $\mathcal{X}$ a vector space and $\mathcal{Y}$ a normed space. The function $\mathcal{F}$ is Gateaux differentiable at $\mathbf{z}$ if the directional derivative*

$$\partial \mathcal{F}(\mathbf{z}; \mathbf{d}) = \mathop{lim}_{\epsilon \to 0} \frac{1}{\epsilon} [\mathcal{F}(\mathbf{z} + \epsilon \mathbf{d}) - \mathcal{F}(\mathbf{z})] = \langle \nabla \mathcal{F}(\mathbf{z}), \mathbf{d} \rangle \tag{1.13}$$

*exists for every direction $\mathbf{d} \in \mathcal{X}$ and is a linear and continuous function of $\mathbf{d}$. The linear operator $\nabla \mathcal{F}(\mathbf{z})$ represents the **Gateaux differential** of $\mathcal{F}$ at $\mathbf{z}$.*

The Gateaux differential generalizes the concept of directional derivative familiar in finite dimensional space. The existence of the Gateaux differential is a rather weak requirement since its definition requires no norm on $\mathcal{X}$; hence, properties of the Gateaux differential are not easily related to continuity. When $\mathcal{X}$ is normed, a more satisfactory definition is given by the Fréchet differential [Luen 69]. If $\mathcal{Y} = \mathbb{R}$, the Gateaux differential of $\mathcal{F}$, if it exists represents the gradient and is denoted sometimes as $\nabla_{\mathbf{z}} \mathcal{F}(\mathbf{z})$. If the vector space $\mathcal{X} = \mathbb{R}^{\mathcal{D}}$, then the gradient is a vector with real entries.

Definition 1.22 (Fréchet Differential)**.** *Let $\mathcal{F}$ be a function defined on an open domain Ω in a normed space $\mathcal{X}$ and having range in a normed space $\mathcal{Y}$. If for fixed $\mathbf{z} \in \Omega$ and each $\mathbf{d} \in \mathcal{X}$ there exists $\partial \mathcal{F}(\mathbf{z}; \mathbf{d}) \in \mathcal{Y}$ which is linear and continuous with respect to $\mathbf{d}$ such that*

$$\lim_{\|\mathbf{d}\| \to 0} \frac{\|\mathcal{F}(\mathbf{z} + \mathbf{d}) - \mathcal{F}(\mathbf{z}) - \partial \mathcal{F}(\mathbf{z}; \mathbf{d})\|}{\|\mathbf{d}\|} = 0$$

*then $\mathcal{F}$ is said to be **Fréchet differentiable** at $\mathbf{z}$ and $\partial \mathcal{F}(\mathbf{z}; \mathbf{d})$ is said to be the **Fréchet differential** of $\mathcal{F}$ at $\mathbf{z}$ with increment $\mathbf{d}$.*

In later chapters, the use of kernel operators will help to increase the classifier potential by taking the original samples $\mathbf{x} \in \mathcal{X}$ into a more complex space. We are in first place concerned with continuous linear bounded functions.

The linear functions on a vector space may be regarded as elements of a vector space by introducing definitions of addition and scalar multiplication. Given two linear function $\mathcal{F}_1, \mathcal{F}_2$ over a space $\mathcal{X}$, we define their sum $\mathcal{F}_1 + \mathcal{F}_2$ as the function over $\mathcal{X}$ given by $(\mathcal{F}_1 + \mathcal{F}_2)(\mathbf{z}) = \mathcal{F}_1(\mathbf{z}) + \mathcal{F}_2(\mathbf{z})$ for all $\mathbf{z} \in \mathcal{X}$. Similarly, given a linear function $\mathcal{F}$, we define $\alpha\mathcal{F}$ by $(\alpha\mathcal{F})(\mathbf{z}) = \alpha[\mathcal{F}(\mathbf{z})]$. The null element in the space of linear functions is the function that is identically zero on $\mathcal{X}$. The space of linear functions defined in this way is called the ***algebraic dual*** of $\mathcal{X}$.

Derivation of optimality conditions

Optimality conditions depend on the constraints given in the problem. These can be derived by analyzing the first order Taylor series.

In order to deduce the optimality conditions, we will follow [Noce 99].

No equality constraints (all constraints inactive). If the equality set is empty $\mathcal{E} = \{\}$ and if the optimal solution $\mathbf{z}^*$ belongs to the ***interior*** of the feasible set, that is if $h_i(\mathbf{z}^*) > 0$ for all $i \in \mathcal{I}$, then, the only condition that has to be verified would be

$$\nabla J(\mathbf{z}^*) = 0 \tag{1.14}$$

since we would be dealing with a problem without constraints.

At point $\mathbf{z}$, a direction $\mathbf{d}$ of improvement, that is, a direction that produces a decrease in J, must satisfy $0 > J(\mathbf{z} + \mathbf{d}) - J(\mathbf{z}) \approx \langle \nabla J(\mathbf{z}), \mathbf{d} \rangle$, or in first order,

$$\langle \nabla J(\mathbf{z}), \mathbf{d} \rangle < 0. \tag{1.15}$$

This condition is depicted in Figure 1.12.

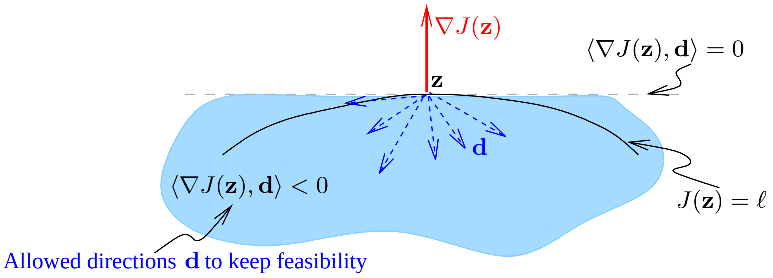

$\nabla J(\mathbf{z})$
$\langle \nabla J(\mathbf{z}), \mathbf{d} \rangle = 0$
$\mathbf{z}$
$\mathbf{d}$
$\langle \nabla J(\mathbf{z}), \mathbf{d} \rangle < 0$
$J(\mathbf{z}) = \ell$
Allowed directions $\mathbf{d}$ to keep feasibility

Figure 1.12: Admissible directions for a minimization problem without constraints. The black curved line is a level curve of objective function J at level ℓ. The blue arrows and blue area depict the allowed directions $\mathbf{d}$ at point $\mathbf{z}$. The dashed gray line contains the directions perpendicular to the gradient of the objective function (not included in the admissible set of directions).

Single equality constraint. If there is only one equality constraint, a direction $\mathbf{d}$ at point $\mathbf{z}$ satisfying the constraint $(g_i(\mathbf{z}) = 0)$, will keep feasibility if the following equation is satisfied: $0 = g_i(\mathbf{z} + \mathbf{d}) \approx g_i(\mathbf{z}) + \langle \nabla g_i(\mathbf{z}), \mathbf{d} \rangle = \langle \nabla g_i(\mathbf{z}), \mathbf{d} \rangle$, resulting in:

$$\langle \nabla g_i(\mathbf{z}), \mathbf{d} \rangle = 0, \quad i \in \mathcal{E}. \tag{1.16}$$

An example of a problem with a single equality constraint is illustrated in Figure 1.13.

If $\mathbf{z}^*$ is an optimal point and if the active set at it consists of one index $\mathcal{A} = \{i\}$, say g_i, then there should not be any direction satisfying both Equation (1.15) and (1.16) at the same time, that is when the admissible sets in Figure 1.12 and Figure 1.13 do not intersect. The

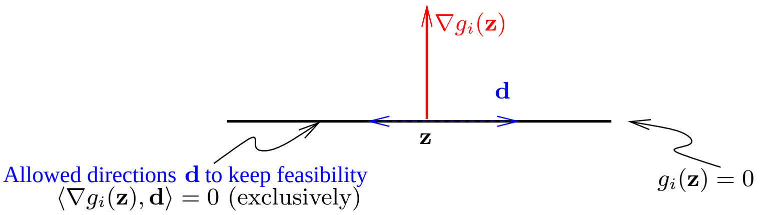

$\nabla g_i(\mathbf{z})$
$\mathbf{d}$
$\mathbf{z}$
Allowed directions $\mathbf{d}$ to keep feasibility
$\langle \nabla g_i(\mathbf{z}), \mathbf{d} \rangle = 0$ (exclusively)
$g_i(\mathbf{z}) = 0$

Figure 1.13: Admissible directions for a problem with equality constraints. The blue arrows depict the allowed directions $\mathbf{d}$ with respect to the equality constraint g_i at $\mathbf{z}$.

only way that such a direction cannot exist is if $\nabla J(\mathbf{z}^*)$ and $\nabla g_i(\mathbf{z}^*)$ are parallel [Noce 99], that is if at $\mathbf{z}^*$, it is hold

$$\nabla J(\mathbf{z}^*) = \beta \nabla g_i(\mathbf{z}^*), \qquad \beta \in \mathbb{R}. \tag{1.17}$$

If $\mathcal{X} = \mathbb{R}^{\mathcal{D}}$, then $\nabla g_i(\mathbf{z}) \in \mathbb{R}^{\mathcal{D}}$ and if condition given by Equation (1.17) is **not** satisfied, then there exists a direction of improvement. The following is an improving direction at $\mathbf{z}$:

$$\mathbf{d} = -\left(I - \frac{\nabla g_i(\mathbf{z}) \nabla g_i(\mathbf{z})^\top}{\|\nabla g_i(\mathbf{z})\|^2} \right) \nabla J(\mathbf{z})$$

where I is the identity matrix of size $\mathcal{D}$. It can be verified that this direction satisfies both conditions in Equations (1.15) and (1.16).

Inequalities constraints. If the inequalities are strictly satisfied at $\mathbf{z}$ (i.e. $0 \leq h_i(\mathbf{z})$), the only restriction for a direction $\mathbf{d}$ is to keep feasibility with respect to the inequality constraint by holding, $0 \leq h_i(\mathbf{z} + \mathbf{d}) \approx h_i(\mathbf{z}) + \langle \nabla h_i(\mathbf{z}), \mathbf{d} \rangle$ $i \in \mathcal{I}$ as shown in Figure 1.14, that is,

$$h_i(\mathbf{z}) + \langle \nabla h_i(\mathbf{z}), \mathbf{d} \rangle \geq 0 \tag{1.18}$$

To determine whether a direction $\mathbf{d}$ exists satisfying both Equations (1.15) and (1.18), two cases have to be considered:

Case I: z is inside the feasible set. If an inequality constraint i is inactive, that is, we have $h_i(\mathbf{z}) > 0$, and if $\nabla J(\mathbf{z}) \neq 0$, a direction $\mathbf{d}$ not violating this constraint can be found by setting:

$$\mathbf{d} = -h_i(\mathbf{z}) \frac{\nabla J(\mathbf{z})}{\|\nabla J(\mathbf{z})\| \|\nabla h_i(\mathbf{z})\|}$$

Case II: z is in the boundary of the feasible set. If an inequality constraint i is active, that is if $h_i(\mathbf{z}) = 0$, to retain feasibility with respect to it, Equation (1.15) has to be satisfied together with the condition given by Equation (1.18) which in this case becomes: $\langle \nabla h_i(\mathbf{z}), \mathbf{d} \rangle \geq 0$. The first condition defines an open subspace as illustrated in Figure 1.12, while the second condition defines a closed space as shown in Figure 1.14.

It is clear from these two figures that the two admissible regions fail to intersect only when ∇J and ∇h_i point in the same direction at the optimal point $\mathbf{z}^*$:

$$\nabla J(\mathbf{z}^*) = \alpha \nabla h_i(\mathbf{z}^*) \qquad \text{with} \quad \alpha \geq 0 \tag{1.19}$$

The ***Lagrange Function*** $\mathbf{L} : \mathcal{X} \times \mathbb{R}^{|\mathcal{I}|+|\mathcal{E}|} \to \mathbb{R}$ is introduced to help establish the necessary conditions for a feasible point $\mathbf{z}$ with vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ to be an optimal solution to Problem 1.19:

$$\mathbf{L}(\mathbf{z}, \boldsymbol{\beta}, \boldsymbol{\alpha}) = J(\mathbf{z}) - \langle \boldsymbol{\alpha}, \mathbf{h}(\mathbf{z}) \rangle - \langle \boldsymbol{\beta}, \mathbf{g}(\mathbf{z}) \rangle. \tag{1.20}$$

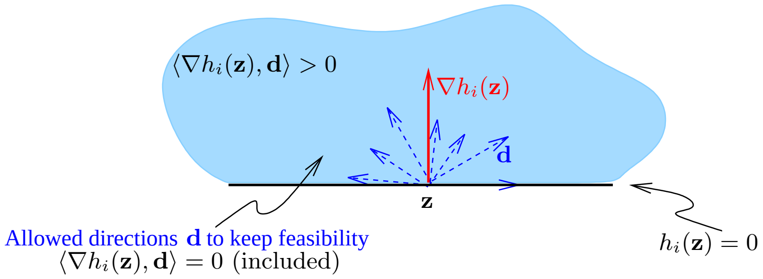

$\langle \nabla h_i(\mathbf{z}), \mathbf{d} \rangle > 0$
$\nabla h_i(\mathbf{z})$
$\mathbf{d}$
$\mathbf{z}$
Allowed directions $\mathbf{d}$ to keep feasibility
$\langle \nabla h_i(\mathbf{z}), \mathbf{d} \rangle = 0$ (included)
$h_i(\mathbf{z}) = 0$

Figure 1.14: Admissible directions for a problem with inequality constraints. The light blue zone depicts the allowed directions $\mathbf{d}$ with respect to h_i at $\mathbf{z}$.

The utility of this function lies on the characteristics of its derivative at the optimal point. It has to be noticed that $\nabla_{\mathbf{z}}\mathbf{L}(\mathbf{z}^*, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \nabla_{\mathbf{z}}J(\mathbf{z}^*) - \sum_{i\in\mathcal{I}} \alpha_i \nabla_{\mathbf{z}} h_i(\mathbf{z}^*) - \sum_{i\in\mathcal{E}} \beta_i \nabla_{\mathbf{z}} g_i(\mathbf{z}^*) = 0$, for all $\boldsymbol{\beta} \in \mathbb{R}^{\mathcal{E}}$ and $\alpha_i \geq 0, i \in \mathcal{I}$, which is a generalization of Conditions (1.14), (1.17) and (1.19) for more than one constraint function.

This observation suggests that we can search for solutions of the constrained Problem 1.19 by searching for stationary points of the Lagrangian function. The scalars α_i and β_j in Equation (1.20) are called ***Lagrange multipliers*** for the constraints $h_i(\mathbf{z}) \geq 0$ and $g_j(\mathbf{z}) = 0$, respectively. Attention has to be given to the properties of the constraint gradients. The vector $\nabla h_i(\mathbf{z})$ is often called the *normal* to the constraint h_i at the point $\mathbf{z}$, since it is usually a vector that is perpendicular to the contours of the constraint h_i at $\mathbf{z}$, and in the case of an equality constraint, it points toward the feasible side of this constraint. However, it can happen that ∇h_i vanishes due to the algebraic representation of h_i, so that the term $\alpha_i \nabla h_i(\mathbf{z})$ equals zero for all values of α_i and thus it plays no role in the Lagrangian gradient $\nabla_{\mathbf{z}}\mathbf{L}$. If the above condition holds, none of the active constraint gradients can be zero.

1.4.2 Optimality Conditions

With the provided tools in the previous section, first order optimality conditions for a general nonlinear programming problem can be provided. Before that, let us expose a useful definition.

Definition 1.23 (Regular point)**.** *Let $\mathcal{X}$ be a vector space and let $\mathcal{Y}$ be a normed space with a positive cone P (see Appendix A.1.1) having nonempty interior. Let $h : \mathcal{X} \to \mathcal{Y}$ be a function which has a Gateaux differential that is linear in its increment. A point $\mathbf{z} \in \mathcal{X}$ is said to be a **regular point** of the inequality $h(\mathbf{z}) \leq \mathbf{0}$ if $h(\mathbf{z}) \leq \mathbf{0}$ and there is a $\mathbf{d} \in \mathcal{X}$ such that $h(\mathbf{z}) + \nabla_{\mathbf{d}} h(\mathbf{z}) < \mathbf{0}$.*

Optimality conditions are stated in the following theorem.

Theorem 1.24 (First-Order-Necessary Conditions (KKT) [Noce 99])**.** *Suppose that $\mathbf{z}^*$ is a local solution of Problem 1.19 and it is a regular point. Then there are Lagrange multipliers vectors $\boldsymbol{\alpha}^*$, $\boldsymbol{\beta}^*$ with components $\alpha_i, i \in \mathcal{I}$, such that the following conditions are satisfied at $(\mathbf{z}^*, \alpha^*, \boldsymbol{\beta}^*)$:*

$$\nabla_{\mathbf{x}}\mathbf{L}(\mathbf{z}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*) = 0, \tag{1.21a}$$
$$g_i(\mathbf{z}^*) = 0, \forall i \in \mathcal{E}, \tag{1.21b}$$
$$h_i(\mathbf{z}^*) \geq 0, \forall i \in \mathcal{I}, \tag{1.21c}$$
$$\boldsymbol{\alpha}_i^* h_i(\mathbf{z}^*) = 0, \forall i \in \mathcal{I}. \tag{1.21d}$$
$$\boldsymbol{\alpha}_i^* \geq 0, \forall i \in \mathcal{I}, \tag{1.21e}$$

The above conditions are often known as the ***Karush-Kuhn-Tucker conditions***, ***KKT*** conditions or system for short [Boyd 04], and they have to be satisfied in order to find an optimum for problem 1.19.

1.5. Dual of the SVM Learning Problem

The stated problems in Section 1.3 are ***well posed*** for a particular λ, in the sense that:

1. a solution exists,
2. the solution is unique,
3. the solution is stable.

There exists some numerical approaches to solve the proposed primal problems for classification and ranking [Bott 07, Teo 07]. However, in this work we focused in methods applied to the dual formulation that can be derived using the given optimality conditions. Taking the dual formulation, deeper analysis can be done in order to obtain efficiently the whole optimal solution set with respect to the parameter λ.

1.5.1 Classification Dual Problem

The dual of this problem can be obtained by considering the Lagrangian of Problem 1.17:

$$\mathbf{L}(f_0, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \gamma) = \sum_{i=1}^{n} \xi_i + \frac{\lambda}{2}\|f_0\|^2_{\mathcal{H}} - \sum_{i=1}^{n} \alpha_i \Big(y_i \big(f_0(\mathbf{x}_i) + b \big) - 1 + \xi_i \Big) - \sum_{i=1}^{n} \gamma_i \xi_i \tag{1.22}$$

with $\alpha_i \geq 0$, $\gamma_i \geq 0$ and $f_0 \in \mathcal{H}$, with $\mathcal{H}$ a RKHS with reproducing kernel $\mathbf{k}$ and $b \in \mathbb{R}$. The decision function will be then $f(\mathbf{x}) = f_0(\mathbf{x}) + b$.

Taking $\mathcal{H}$ as a RKHS is not a coincidence, this hypothesis space will give us several nice properties as it is endowed with an inner product under $\mathcal{H}$, $\langle f_0, g \rangle_{\mathcal{H}}$, which is a linear operator. Moreover, the norm is defined as:

$$\|f_0\|^2_{\mathcal{H}} = \langle f_0, f_0 \rangle_{\mathcal{H}}$$

with the reproducing property:

$$f(\mathbf{x}) = \langle f(\cdot), \mathbf{k}(\mathbf{x}, \cdot) \rangle_{\mathcal{H}}.$$

If $J(f) = \langle f, g \rangle_{\mathcal{H}}$, then the directional derivative $D_J(f, h)$ of $\langle f, g \rangle_{\mathcal{H}}$ at f in the direction h, with $f, g, h \in \mathcal{H}$, can be calculated as follow:

$$\begin{aligned} D_J(f,h) &= \lim_{\epsilon \to 0} \frac{\langle f + \epsilon h, g \rangle_{\mathcal{H}} - \langle f, g \rangle_{\mathcal{H}}}{\epsilon} \\ &= \lim_{\epsilon \to 0} \frac{\langle f, g \rangle_{\mathcal{H}} + \epsilon \langle h, g \rangle_{\mathcal{H}} - \langle f, g \rangle_{\mathcal{H}}}{\epsilon} \\ &= \langle h, g \rangle_{\mathcal{H}}. \end{aligned}$$

and therefore, $\nabla_f \langle f, g \rangle_{\mathcal{H}} = g(\cdot)$. Following a similar derivation, we can deduce,

$$\nabla_f \|f\|^2_{\mathcal{H}} = \nabla_f \langle f, f \rangle_{\mathcal{H}} = 2f(\cdot) \quad \text{and} \tag{1.23}$$

$$\nabla_f f(\mathbf{x}) = \nabla_f \langle f, \mathbf{k}(\mathbf{x}, \cdot) \rangle_{\mathcal{H}} = \mathbf{k}(\mathbf{x}, \cdot). \tag{1.24}$$

This results can help to deduce the KKT optimality conditions for Problem 1.17:

$$\begin{aligned}
\nabla_b \mathbf{L} = 0 \quad &\Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0 && (1.25a)\\
\nabla_{\xi_i} \mathbf{L} = 0 \quad &\Rightarrow \quad 1 - \alpha_i - \gamma_i = 0 && (1.25b)\\
&\Rightarrow \quad \gamma_i = 1 - \alpha_i &&\\
& \quad \text{but as } \gamma_i \geq 0 \quad \Rightarrow \quad 1 - \alpha_i \geq 0 &&\\
& \quad \text{and together with } \alpha_i \geq 0, \text{ we get} &&\\
& \quad 0 \leq \alpha_i \leq 1 &&\\
\nabla_{f_0} \mathbf{L} = 0 \quad &\Rightarrow \quad \lambda f_0(\cdot) - \sum_{i=1}^{n} \alpha_i y_i \mathbf{k}(\mathbf{x}_i, \cdot) = 0 &&\\
&\Rightarrow \quad f_0(\cdot) = \frac{1}{\lambda} \sum_{i=1}^{n} \alpha_i y_i \mathbf{k}(\mathbf{x}_i, \cdot) && (1.25c)\\
&\Rightarrow \quad \|f_0\|^2 = \frac{1}{\lambda^2} \sum_{j=1}^{n} \sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j). && (1.25d)
\end{aligned}$$

And since $f(\mathbf{x}) = f_0(\mathbf{x}) + b$, the decision function will have the form:

$$f(\mathbf{x}) = \frac{1}{\lambda} \sum_{i=1}^{n} \alpha_i y_i \mathbf{k}(\mathbf{x}_i, \cdot) + b. \tag{1.26}$$

Equation (1.25c) is commonly known as the ***representer theorem*** [Kime 71]. So plugging Equations (1.25c) and (1.25d) in (1.22), and using the relations (1.25b) and (1.25a) we get:

$$\mathbf{L} \quad = \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2\lambda} \sum_{j=1}^{n} \sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j).$$

Finally, a dual problem can be deduced putting all the conditions together:

Problem 1.25 (SVC Dual Problem)**.** *Let $S = \{(\mathbf{x}_i, y_i)\}_{i=[\![n]\!]}$ be a set of samples as before. The dual optimization problem is defined as follows:*

$$\begin{array}{rc}
\underset{\boldsymbol{\alpha} \in \mathbb{R}^n}{\text{maximize}} & \mathbb{I}^\top \boldsymbol{\alpha} - \frac{1}{2\lambda} \boldsymbol{\alpha}^\top Y K Y \boldsymbol{\alpha} \\
\text{subject to} & \mathbf{y}^\top \boldsymbol{\alpha} = 0 \\
& \mathbf{0} \leq \boldsymbol{\alpha} \leq \mathbb{I}
\end{array}$$

where $\mathbf{y} = (y_1, y_2, ..., y_n)^\top$, $Y = diag(\mathbf{y})$, $K_{ij} = \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j), i, j = [\![n]\!]$ is the induced Gram matrix from S and the inequalities are entry-wise considered. The vector $\mathbb{I} \in \mathbb{R}^n$ is a vector with all entrances equal to 1.

Remark

This problem turns out to be the same one derived from the maximal margin framework stated in the Appendix A.2, that is, a quadratic problem, which is convex if YKY is semi-definite positive. We should notice that in Problem 1.25, if we use a change of variable by defining $\tilde{\alpha}_i = \frac{\alpha_i}{\lambda} = C\alpha_i, \forall i = [\![n]\!]$, we retrieve the usual form of the SVM dual [Schl 01] (see Appendix A.2).

Bidual formulation

It is known in the case of convex optimization problems that the bidual is equivalent to the primal problem. This is the case for the SVM since a ***bidual problem*** can be derived using again the KKT optimality conditions for the dual Problem 1.25 expressed in its standard form:

$$\begin{array}{ll} \underset{\boldsymbol{\alpha}\in\mathbb{R}^n}{\text{minimize}} & \frac{1}{2\lambda}\boldsymbol{\alpha}^\top YKY\boldsymbol{\alpha} - \mathbb{1}^\top\boldsymbol{\alpha} \\ \text{subject to} & \mathbf{y}^\top\boldsymbol{\alpha} = 0 \\ & \mathbf{0} \leq \boldsymbol{\alpha} \leq \mathbb{1}. \end{array}$$

Its Lagrange function can be written as:

$$\begin{aligned} \mathbf{L}(\boldsymbol{\alpha}, b, \boldsymbol{\nu}, \boldsymbol{\xi}) &= \frac{1}{2\lambda}\boldsymbol{\alpha}^\top YKY\boldsymbol{\alpha} - \boldsymbol{\alpha}^\top\mathbb{1} + b\boldsymbol{\alpha}^\top\mathbf{y} - \boldsymbol{\alpha}^\top\boldsymbol{\nu} - (\mathbb{1}-\boldsymbol{\alpha})^\top\boldsymbol{\xi} \\ \mathbf{L}(\boldsymbol{\alpha}, b, \boldsymbol{\nu}, \boldsymbol{\xi}) &= \frac{1}{2\lambda}\boldsymbol{\alpha}^\top YKY\boldsymbol{\alpha} + \boldsymbol{\alpha}^\top\Big(-\mathbb{1} + b\mathbf{y} - \boldsymbol{\nu} + \boldsymbol{\xi}\Big) - \mathbb{1}^\top\boldsymbol{\xi} \end{aligned} \tag{1.27}$$

with $\boldsymbol{\xi} \geq \mathbf{0}$ and $\boldsymbol{\nu} \geq \mathbf{0}$. Optimality conditions imply:

$$\begin{aligned} \frac{\partial \mathbf{L}}{\partial \boldsymbol{\alpha}} = 0 \quad &\Rightarrow \quad \frac{1}{\lambda}YKY\boldsymbol{\alpha} - \mathbb{1} + b\mathbf{y} - \boldsymbol{\nu} + \boldsymbol{\xi} = 0 \\ &\Rightarrow \quad \boldsymbol{\alpha} = \lambda(YKY)^{-1}\left(\mathbb{1} - b\mathbf{y} + \boldsymbol{\nu} - \boldsymbol{\xi}\right) \end{aligned} \tag{1.28}$$

and plugging this last Equation (1.28) into the Lagrange Equation (1.27), together with the constraints of positiveness of the Lagrange multipliers we obtain:

$$\begin{aligned} \mathbf{L}(\boldsymbol{\alpha}, b, \boldsymbol{\nu}, \boldsymbol{\xi}) &= \frac{\lambda}{2}\left((YKY)^{-1}\left(\mathbb{1} - b\mathbf{y} + \boldsymbol{\nu} - \boldsymbol{\xi}\right)\right)^\top YKY(YKY)^{-1}\left(\mathbb{1} - b\mathbf{y} + \boldsymbol{\nu} - \boldsymbol{\xi}\right) \\ &\quad + \lambda\left((YKY)^{-1}\left(\mathbb{1} - b\mathbf{y} + \boldsymbol{\nu} - \boldsymbol{\xi}\right)\right)^\top\left(-\mathbb{1} + b\mathbf{y} - \boldsymbol{\nu} + \boldsymbol{\xi}\right) - \mathbb{1}^\top\boldsymbol{\xi} \\ &= -\frac{\lambda}{2}\left(\left(\mathbb{1} - b\mathbf{y} + \boldsymbol{\nu} - \boldsymbol{\xi}\right)\right)^\top (YKY)^{-1}\left(\mathbb{1} - b\mathbf{y} + \boldsymbol{\nu} - \boldsymbol{\xi}\right) - \mathbb{1}^\top\boldsymbol{\xi} \\ &= -\frac{\lambda}{2}\left(\left(\mathbb{1} - b\mathbf{y} + \boldsymbol{\nu} - \boldsymbol{\xi}\right)\right)^\top YK^{-1}Y\left(\mathbb{1} - b\mathbf{y} + \boldsymbol{\nu} - \boldsymbol{\xi}\right) - \mathbb{1}^\top\boldsymbol{\xi} \end{aligned}$$

together with the inequalities:

$$\boldsymbol{\xi} \geq 0, \quad \boldsymbol{\nu} \geq 0$$

Therefore, the new dual problem of the dual problem, the bidual problem, is stated as follows:

$$\begin{array}{ll} \underset{\boldsymbol{\nu},\boldsymbol{\xi}\in\mathbb{R}^n, b\in\mathbb{R}}{\text{minimize}} & \frac{\lambda}{2}\left(\left(\mathbb{1} - b\mathbf{y} + \boldsymbol{\nu} - \boldsymbol{\xi}\right)\right)^\top YK^{-1}Y\left(\mathbb{1} - b\mathbf{y} + \boldsymbol{\nu} - \boldsymbol{\xi}\right) + \mathbb{1}^\top\boldsymbol{\xi} \\ \text{subject to} & \boldsymbol{\nu} \geq 0, \\ & \boldsymbol{\xi} \geq 0 \end{array}$$

Next, a change of variable is proposed to obtain a standard form of the problem. We will let $\boldsymbol{\beta} = (YK)^{-1}[\mathbb{1} - b\mathbf{y} + \boldsymbol{\nu} - \boldsymbol{\xi}]$ so that

$$YK\boldsymbol{\beta} = \mathbb{1} - b\mathbf{y} + \boldsymbol{\nu} - \boldsymbol{\xi} \tag{1.29}$$

and

$$\begin{aligned} \boldsymbol{\nu} \geq \mathbf{0} \quad &\Rightarrow \quad YK\boldsymbol{\beta} - \mathbb{1} + b\mathbf{y} + \boldsymbol{\xi} \geq \mathbf{0}, \\ &\Rightarrow \quad YK\boldsymbol{\beta} + b\mathbf{y} \geq \mathbb{1} - \boldsymbol{\xi}. \end{aligned} \tag{1.30}$$

It has to be remarked that $Y = Y^\top$ and that $YY = I$, the identity matrix. The relation between the primal and dual variables can be obtained by deriving from Equation (1.29): $YK\boldsymbol{\beta}Y = (\mathbb{1} - b\mathbf{y} + \boldsymbol{\nu} - \boldsymbol{\xi})Y$, but $YK\boldsymbol{\beta}Y = YKY\boldsymbol{\beta}$ because Y is diagonal and $\boldsymbol{\beta}$ a vector, leading to $\boldsymbol{\beta} = (YKY)^{-1}(\mathbb{1} - b\mathbf{y} + \boldsymbol{\nu} - \boldsymbol{\xi})Y$. Using Equation (1.28), we get:

$$\boldsymbol{\alpha} \;=\; \lambda Y\boldsymbol{\beta}$$

With this result, the bidual problem can be derived:

Problem 1.26 (Bidual Problem)**.** *If $S = \{(\mathbf{x}_i, y_i)\}_{i=[\![n]\!]}$ is a set of samples, a **bidual problem** equivalent to the primal problem is expressed as follows:*

$$\begin{array}{cc} \underset{\boldsymbol{\beta}\in\mathbb{R}^n, b\in\mathbb{R}, \boldsymbol{\xi}\in\mathbb{R}^n}{\text{minimize}} & \mathbb{1}^\top\boldsymbol{\xi} + \frac{\lambda}{2}\boldsymbol{\beta}^\top K\boldsymbol{\beta} \\ \text{subject to} & Y(K\boldsymbol{\beta} + b\mathbb{1}) \geq \mathbb{1} - \boldsymbol{\xi} \\ & \boldsymbol{\xi} \geq 0. \end{array}$$

This problem turns out to be quite interesting because, as expected, the dual of the dual problem (the bidual) is equivalent to the original primal problem. This concise derivation does not depend anymore on the functional f but only in its expansion coefficients $\boldsymbol{\beta}$. The functional form can be retrieved as $f_0(\mathbf{x}) = \sum_{i=1}^{n} \beta_i \mathbf{k}(\mathbf{x}_i, \mathbf{x})$ (representer theorem!). Additionally, a clear relationship can be established between the Lagrange multiplier of the dual problem and the bias b. If the dual problem is solved together with the Lagrange multipliers, the **bias term b is the Lagrange multiplier associated to the equality constraint $\mathbf{y}^\top\boldsymbol{\alpha} = 0$.**
This way of calculating b contrasts with the one proposed by Keerthi [Keer 01] since if the dual is solved, no additional calculation is necessary to obtain b.

1.5.2 Ranking Dual Problem

This problem can easily be transformed into a PQP problem with exactly the same procedure as in the classification case. Therefore, all known methods can be used to improve the optimization problem resolution and the parameter search.
We recall the ranking SVM problem that we are interested to solve as stated before:

$$\begin{array}{lll} \underset{f\in\mathcal{H}, \boldsymbol{\xi}\in\mathbb{R}^m}{\text{argmin}} & \boldsymbol{\xi}^\top\mathbb{1} + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2 & \\ \text{s. t.} & f(\mathbf{x}_{u_i}) - f(\mathbf{x}_{v_i}) \geq 1 - \xi_i & \forall i = [\![m]\!] \\ & \xi_i \geq 0 & \forall i = [\![m]\!]. \end{array} \tag{1.31}$$

To ease the notation, we define $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_m)^\top$, let $\mathbf{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be the reproducing kernel of $\mathcal{H}$ and rename each vertex by $\mathbf{x}_i \in S_X = \{\mathbf{x}_i\}_{i=[\![n]\!]}$.
The Lagrange function will be defined as

$$\mathbf{L} = \boldsymbol{\xi}^\top\mathbb{1} + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2 - \sum_{i=1}^{m}\alpha_i(f(\mathbf{x}_{u_i}) - f(\mathbf{x}_{v_i}) - 1 + \xi_i) - \boldsymbol{\xi}^\top\boldsymbol{\gamma}$$

with $\alpha_i, \geq 0$, $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_m)^\top$ and $\gamma_i \geq 0, i = [\![m]\!]$. As f belongs to a RKHS, the gradient derivation of the dot product with respect to f again holds along with the derivative of the norm $\|f\|_{\mathcal{H}}^2$ and $f(\mathbf{x})$ with respect to f as stated by Equations (1.23) and (1.24). The KKT optimality conditions can be deduced as follows:

$$\frac{\partial \mathbf{L}}{\partial \boldsymbol{\xi}} = 0 \;\Rightarrow\; \mathbb{1} - \boldsymbol{\alpha} - \boldsymbol{\gamma} = \mathbf{0}$$

$$\Rightarrow\; \mathbf{0} \leq \boldsymbol{\alpha} \leq \mathbb{1} \tag{1.32}$$

$$\frac{\partial \mathbf{L}}{\partial f} = 0 \;\Rightarrow\; f(\cdot) = \frac{1}{\lambda}\sum_{i=1}^{m}\alpha_i(\mathbf{k}(\mathbf{x}_{u_i}, \cdot) - \mathbf{k}(\mathbf{x}_{v_i}, \cdot)) \tag{1.33}$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_m)^\top$. If $K \in \mathbb{R}^{n\times n}$ is the Gram matrix induced by kernel $\mathbf{k}$, so that $K_{ij} = \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j)$, a matrix $P \in \mathbb{R}^{m\times n}$ can be defined with entries

$$P_{ij} = \begin{cases} +1 & \text{if } j = u_i \\ -1 & \text{if } j = v_i \\ 0 & \text{otherwise} \end{cases} \quad \Longrightarrow \quad PK = \begin{pmatrix} \mathbf{k}(\mathbf{x}_{u_1})^\top - \mathbf{k}(\mathbf{x}_{v_1})^\top \\ \mathbf{k}(\mathbf{x}_{u_2})^\top - \mathbf{k}(\mathbf{x}_{v_2})^\top \\ \vdots \\ \mathbf{k}(\mathbf{x}_{u_m})^\top - \mathbf{k}(\mathbf{x}_{v_m})^\top \end{pmatrix} \tag{1.34}$$

with slight abuse of notation we wrote $\mathbf{k}(\mathbf{x}) = (\mathbf{k}(\mathbf{x}, \mathbf{x}_1), \mathbf{k}(\mathbf{x}, \mathbf{x}_2), ..., \mathbf{k}(\mathbf{x}, \mathbf{x}_n))^\top$. From the definitions of P and the vector $\mathbf{k}(\mathbf{x})$, we deduce that :

$$\begin{aligned} f(\mathbf{x}) &= \frac{1}{\lambda}\sum_{i=1}^{m} \alpha_i P_{i,.}\mathbf{k}(\mathbf{x}) \\ &= \frac{1}{\lambda}\boldsymbol{\alpha}^\top P\mathbf{k}(\mathbf{x}) \end{aligned} \tag{1.35}$$

where $P_{i,.}$ represents the i^{th} row of P. Hence if we define $\boldsymbol{\beta} = \frac{1}{\lambda}P^\top\boldsymbol{\alpha}$ with $\boldsymbol{\beta} = (\beta_1, \beta_2, \cdots, \beta_n)^\top$, the decision function will have the form suggested by the representer theorem

$$f(\cdot) = \sum_{i=1}^{n} \beta_i \mathbf{k}(\mathbf{x}_i, \cdot) \tag{1.36}$$

Plugging Equation (1.35) in the Lagrange function of the Ranking SVM together with the KKT condition associated to the primal variables ξ_i, we deduce after some algebra that the dual of a ranking Problem 1.31 with m preferences $E = \{(\mathbf{x}_{u_i}, \mathbf{x}_{v_i}) \mid i = [\![m]\!]\}$ can be written as:

Problem 1.27 (SVM Ranking Dual Problem)**.** *Let $S = \{(\mathbf{x}_i, y_i)\}_{i=[\![n]\!]}$ be a set of samples as before and E the set encoding the ranking constraints. The dual optimization problem for the ranking problem is defined as follows:*

$$\begin{aligned} \underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{argmax}} \quad & \boldsymbol{\alpha}^\top \mathbb{1} - \tfrac{1}{2\lambda}\boldsymbol{\alpha}^\top PKP^\top\boldsymbol{\alpha} \\ s.t. \quad & \mathbf{0} \leq \boldsymbol{\alpha} \leq \mathbb{1}. \end{aligned}$$

where the kernel matrix K is defined as $K_{ij} = \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j), i, j = [\![n]\!]$ and the matrix P defined according to E by Equation (1.34)*.*

This is a quadratic problem and there is a vast number of algorithms for its efficient resolution. As already shown, the parameters of the primal model can be retrieved as $\boldsymbol{\beta} = \frac{1}{\lambda}P^\top\boldsymbol{\alpha}$ from the solution of the QP problem. Nevertheless, this quadratic problem is of $O(m)$ which can be very large, as it can be of $O(n^2)$. An alternative problem is proposed in the next chapter with the aim of reducing the number of constraints.

As in the classification case, the bidual problem can be deduced in terms of $\boldsymbol{\beta}$. The canonical form of Problem 1.27 is:

$$\begin{aligned} \underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{argmin}} \quad & -\boldsymbol{\alpha}^\top \mathbb{1} + \tfrac{1}{2\lambda}\boldsymbol{\alpha}^\top PKP^\top\boldsymbol{\alpha} \\ \text{s.t.} \quad & \boldsymbol{\alpha} \geq \mathbf{0}, \\ & \mathbb{1} - \boldsymbol{\alpha} \geq \mathbf{0}. \end{aligned} \tag{1.37}$$

The Lagrangian function of the previous problem is then

$$\mathbf{L}(\boldsymbol{\alpha}, \boldsymbol{\nu}, \boldsymbol{\xi}) = \frac{1}{2\lambda}\boldsymbol{\alpha}^\top PKP^\top\boldsymbol{\alpha} - \boldsymbol{\alpha}^\top\mathbb{1} - \boldsymbol{\alpha}^\top\boldsymbol{\nu} - \boldsymbol{\xi}^\top(\mathbb{1} - \boldsymbol{\alpha}), \quad \text{with } \boldsymbol{\nu} \geq \mathbf{0},\ \boldsymbol{\xi} \geq \mathbf{0},$$

the optimality conditions can be derived as:

$$\begin{aligned} \frac{\partial \mathbf{L}}{\partial \boldsymbol{\alpha}} = 0 \quad &\Rightarrow \quad \frac{1}{\lambda} PKP^\top \boldsymbol{\alpha} - \mathbb{I} - \boldsymbol{\nu} + \boldsymbol{\xi} \\ &\Rightarrow \quad \boldsymbol{\nu} = \frac{1}{\lambda} PKP^\top \boldsymbol{\alpha} - \mathbb{I} + \boldsymbol{\xi} \geq \mathbf{0}. \end{aligned}$$

Plugging it in the Lagrangian function and taking conditions $\boldsymbol{\nu} \geq 0$, we obtain the problem:

$$\begin{array}{cl} \underset{\boldsymbol{\alpha} \in \mathbb{R}^m}{\operatorname{argmax}} & -\frac{1}{2\lambda} \boldsymbol{\alpha}^\top PKP^\top \boldsymbol{\alpha} - \boldsymbol{\xi}^\top \mathbb{I} \\ \text{s.t.} & \frac{1}{\lambda} PKP^\top \boldsymbol{\alpha} - \mathbb{I} + \boldsymbol{\xi} \geq \mathbf{0} \end{array} \tag{1.38}$$

If we define $\boldsymbol{\beta} = \frac{1}{\lambda} P^\top \boldsymbol{\alpha}$, the ***ranking bidual problem*** can be stated as:

$$\underset{\boldsymbol{\beta} \in \mathbb{R}^m, \boldsymbol{\xi} \in \mathbb{R}^m}{\operatorname{argmin}} \quad \boldsymbol{\xi}^\top \mathbb{I} + \frac{\lambda}{2} \boldsymbol{\beta}^\top K \boldsymbol{\beta} \tag{1.39}$$

$$\text{s.t.} \quad PK\boldsymbol{\beta} \geq \mathbb{I} - \boldsymbol{\xi} \tag{1.40}$$

$$\boldsymbol{\xi} \geq \mathbf{0}. \tag{1.41}$$

Again, the advantage of the bidual problem is that the functional form is not anymore involved in the problem formulation, but instead its explicit formulation in terms of $\boldsymbol{\beta}$.

1.6. Quadratic Programming Methods

Let the primal objective function $J(\mathbf{z})$ in Problem 1.19 have a quadratic form $J(\mathbf{z}) = \frac{1}{2}\mathbf{z}^\top Q \mathbf{z} - \mathbb{I}^\top \mathbf{z}$ and let the constraints be linear functions of the form $g(\mathbf{z}) = \mathbf{y}^\top \mathbf{z} - d$, (a linear constraint), $h_i(\mathbf{z}) = C_i - z_i$ and $h_{i'}(\mathbf{z}) = z_{i'}$, $i, i' \in \mathcal{I}$. Assuming $\mathbf{z} \in \mathbb{R}^{\mathcal{D}}$, it turns out that with these assumptions, Problem 1.19 takes the form of a quadratic problem:

$$\begin{array}{cll} \underset{\mathbf{z}}{\text{minimize}} & \frac{1}{2}\mathbf{z}^\top Q \mathbf{z} - \mathbb{I}^\top \mathbf{z} & \\ \text{subject to} & 0 \leq \mathbf{z}_i \leq C_i & \forall i = [\![\mathcal{D}]\!] \\ & \mathbf{y}^\top \mathbf{z} = d, & \end{array} \tag{1.42}$$

where $\mathbb{I}$ is a vector of ones; $0 < C_i$ is the upper bound for variable z_i; Q is an $\mathcal{D} \times \mathcal{D}$ positive semidefinite matrix, and $\mathbf{y}$ is a vector of size $\mathcal{D}$.

This is the dual of the SVM problem. The primal problem can also be efficiently solved in the primal [Chap 07a, Shal 07, Do 08, Bott 08] but is no further treated in this thesis.

The difficulty of solving the previous problem lies in the density of Q, because Q_{ij} is in general not sparse, so if the matrix Q is very large, it might not fit in the available memory, therefore, the choice of appropriate algorithms have to be done.

The methods proposed by [Osun 97], [Plat 99a] and [Joac 99a] are decomposition algorithms that modify only a subvector of $\mathbf{z}$ per iteration to approach this problem in a better way. This subset –denoted as the ***working set*** $\mathcal{B}$– leads to a small sub-problem to be minimized at each iteration. Several QP solver methods are listed and explained in the book of Nocedal and in the one of Luenberger [Noce 99, Luen 05].

In the case of the method proposed by Osuna [Osun 97] (usually called ***chunking***), a subset of the variables is taken and the resulting subproblem is solved. This breaks down the problem in smaller subproblems and will eventually lead to an optimal solution for the whole QP. At each iteration, the chosen variables are those that violate the most the KKT conditions. The variables with Lagrange multipliers α_i equal to zero, that is, all variables with $\alpha_i = 0$ will be left out of the problem resolution for this iteration. This technique has the advantage that the size of the temporal QP problems that have to be solved is a lot smaller than the original one causing less memory issues.

The ***decomposition technique*** [Osun 97] has the same principle as the previous one, but in this case, it is recommended to keep fixed the number of variables to be changed for each subproblem.

The ***Sequential Minimal Optimization*** (***SMO***) [Plat 99a] is a particular (extreme) case of the decomposition technique, where the subset is restricted to have size 2. Then, in each iteration a simple two-variable problem is solved. Chang and Lin in [Chan 05] developed an algorithm based in SMO but it relies more advanced working set selection scheme.

There exists several other methods for quadratic constrained optimization like the ***interior point method***, the ***gradient projection method*** and methods for general constrained optimization problems. A more detailed explanation of these methods can be found in optimization books [Noce 99, Boyd 04, Luen 05]. Because of their efficiency, we will focus in the SMO and the active set methods, explained in the next subsections. The interior points method is described in the Appendix A.3.

1.6.1 Sequential Minimal Optimization (SMO)

At each iteration, SMO identifies a pair $\{z_i, z_j\}$ to solve a subproblem with only two variables. This pair is chosen such that it leads to the maximum decrease in the objective function. The algorithm to select the working set is described in [Fan 05]. After selecting this subset, a new subproblem can be defined [Chan 05] as follows:

Problem 1.28 (Two-variable QP Subproblem). *For iteration t, if $\mathcal{B} = \{i, j\}$ and $Q_{ii} + Q_{jj} - 2Q_{ij} > 0$, then the following problem has to be solved*

$$\begin{aligned} \underset{\mathbf{z}_{\mathcal{B}}^t=(z_i,z_j)}{\text{minimize}} \quad & \tfrac{1}{2}[z_i \quad z_j] \begin{bmatrix} Q_{ii} & Q_{ij} \\ Q_{ij} & Q_{jj} \end{bmatrix} \begin{bmatrix} z_i \\ z_j \end{bmatrix} + (-\mathbb{1}_{\mathcal{B}} + Q_{\mathcal{B}\mathcal{N}}\mathbf{z}_{\mathcal{N}}^t)^\top \begin{bmatrix} z_i \\ z_j \end{bmatrix}, \\ \text{s. t.} \quad & 0 \le z_i \le C_i, \\ & 0 \le z_j \le C_j, \\ & y_i z_i + y_j z_j = d - \mathbf{y}_{\mathcal{N}}^\top \mathbf{z}_{\mathcal{N}}^t, \end{aligned}$$

where $\mathcal{N} = \{1, ..., \mathcal{D}\} \setminus \mathcal{B}$, $\mathbf{z}_{\mathcal{B}}^t$ and $\mathbf{z}_{\mathcal{N}}^t$ are the sub-vectors of $\mathbf{z}$ at iteration t corresponding to $\mathcal{B}$ and $\mathcal{N}$ respectively; $\mathbf{y}_{\mathcal{N}}$ is the sub-vector of $\mathbf{y}$ (vector containing the labels) corresponding to $\mathcal{N}$ and $\mathbb{1}_{\mathcal{B}}$ is the vector of ones of size $|\mathcal{B}|$.

In the particular case of SVM, we will see later that the quadratic term has the form $Q_{ij} = y_i y_j \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j)$, that is, Q is a matrix involving the labels, and the kernel evaluation.

To find the improving direction as $J(\mathbf{z})$ is a quadratic function, the decrease in the objective function is obtained via

$$\begin{aligned} J(\mathbf{z}^t + \mathbf{d}) - J(\mathbf{z}^t) &= \nabla J(\mathbf{z}^t)^\top \mathbf{d} + \frac{1}{2}\mathbf{d}^\top \nabla^2 J(\mathbf{z}^t)\mathbf{d} \qquad (1.43) \\ &= \nabla J(\mathbf{z}^t)_{\mathcal{B}}^\top \mathbf{d}_{\mathcal{B}} + \frac{1}{2}\mathbf{d}_{\mathcal{B}}^\top \nabla^2 J(\mathbf{z}^t)_{\mathcal{B},\mathcal{B}} \mathbf{d}_{\mathcal{B}} \qquad (1.44) \end{aligned}$$

where $\nabla^2 J(\mathbf{z}^t)_{\mathcal{B},\mathcal{B}}$ is the Hessian of J restricted to set $\mathcal{B}$. At the same time, the equality constraint is:

$$\mathbf{y}_{\mathcal{B}}^\top \left(\mathbf{z}_{\mathcal{B}}^t + \mathbf{d}_{\mathcal{B}}\right) = d \Rightarrow \mathbf{y}_{\mathcal{B}}^\top \mathbf{d}_{\mathcal{B}} = 0$$

In order to solve this subproblem, the following theorem in [Fan 05, Theorem 3] was used.

Theorem 1.29 (Working set selection). *The violating pair $\mathcal{B} = \{i, j\}$ with $Q_{ii} + Q_{jj} - 2Q_{ij} > 0$ which maximizes the error decrease 1.44 at point $\mathbf{z}^t$ is the pair yielding the minimum of*

$$-\frac{(-y_i \nabla J(\mathbf{z}^t)_i + y_j \nabla J(\mathbf{z}^t)_j)^2}{2(Q_{ii} + Q_{jj} - 2Q_{ij})}. \qquad (1.45)$$

The optimization problem that provides the working set, defines also the corresponding value of the parameter vector update $\mathbf{d}_{\mathcal{B}}$. If the induced new parameters $\mathbf{z}_{\mathcal{B}} = \mathbf{z}_{\mathcal{B}}^t + \mathbf{d}_{\mathcal{B}}$ do not satisfy the box constraints $0 \leq \mathbf{z}_i \leq C_i$, a necessary projection of $\mathbf{z}_{\mathcal{B}}$ on this box is carried over [Fan 05]. Further modifications were later proposed to this method in order to turn it more efficient since only few modifications are done at each iteration [Keer 01].

1.6.2 Active Set Method

This method is very efficient to solve the following kind of problems:

$$\begin{aligned} \underset{\mathbf{z}\in\mathbb{R}^{\mathcal{D}}}{\text{minimize}} \quad & \tfrac{1}{2}\mathbf{z}^\top Q\mathbf{z} - \mathbb{1}^\top \mathbf{z} && (1.46a)\\ \text{subject to} \quad & \mathbf{0} \leq \mathbf{z} \leq \mathbf{C} && (1.46b)\\ & \mathbf{y}^\top \mathbf{z} = 0 && (1.46c) \end{aligned}$$

where $\mathbf{z} = [z_1, z_2, ..., z_{\mathcal{D}}]^\top$, $\mathbf{0}$ is a vector of zeros and $\mathbf{C}$ is a vector containing only the value C and Q is again a positive definite matrix. A more general formulation with $a_i \leq z_i \leq C_i$ can be straightforwardly derived but we will focus on in the case $a_i = 0$ because this will allow us to handle datasets of huge size [Loos 07b].

The Lagrange equation of Problem 1.46 can be stated in an analogous way as follows:

$$\mathbf{L}(\mathbf{z}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \mu) = \frac{1}{2}\mathbf{z}^\top Q\mathbf{z} - \mathbb{1}^\top \mathbf{z} - \boldsymbol{\alpha}^\top \mathbf{z} - \boldsymbol{\beta}^\top (C\mathbb{1} - \mathbf{z}) + \mu \mathbf{y}^\top \mathbf{z} \tag{1.47}$$

with $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, ..., \alpha_{\mathcal{D}}]^\top$, $\boldsymbol{\beta} = [\beta_1, \beta_2, ..., \beta_{\mathcal{D}}]^\top$ and $\boldsymbol{\alpha} \geq \mathbf{0}$ and $\boldsymbol{\beta} \geq \mathbf{0}$.

Then, the optimality conditions at the optimum $(\mathbf{z}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \mu^*)$ of this problem would look like:

$$\begin{aligned} \nabla_{\mathbf{x}}\mathbf{L}(\mathbf{z}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \mu^*) &= 0, \\ \mathbf{y}^\top \mathbf{z}^* &= 0, \\ \mathbf{z}^* &\geq \mathbf{0}, \\ C\mathbb{1} - \mathbf{z}^* &\geq \mathbf{0}, \\ \alpha_i^* \mathbf{z}_i^* &= 0, && (1.48a)\\ \beta_i^* (C\mathbb{1} - \mathbf{z}^*)_i &= 0, && (1.48b)\\ \boldsymbol{\alpha}^* &\geq \mathbf{0}, && (1.48c)\\ \boldsymbol{\beta}^* &\geq \mathbf{0}, && (1.48d) \end{aligned}$$

This method uses the advantage provided by the box-constraint in Equation (1.46b), that is $0 \leq z_i \leq C$, so that if an optimal solution $\mathbf{z} = [z_1, z_2, ..., z_{\mathcal{D}}]^\top$ is given, variables z_i can be partitioned as follows:

- $\mathcal{I}_0 = \{i : z_i = 0\}$
- $\mathcal{I}_w = \{i : 0 < z_i < C\}$
- $\mathcal{I}_c = \{i : z_i = C\}$

With these sets, the previous Lagrange Equation 1.47 can be written as follows:

$$\begin{aligned} \mathbf{L}(\mathbf{z}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \mu) \;=\; & \frac{1}{2}\mathbf{z}_w^\top Q_{w,w}\mathbf{z}_w + \frac{1}{2}\mathbf{z}_c^\top Q_{c,c}\mathbf{z}_c + \frac{1}{2}\mathbf{z}_0^\top Q_{0,0}\mathbf{z}_0 \\ & +\mathbf{z}_c^\top Q_{c,w}\mathbf{z}_w + \mathbf{z}_c^\top Q_{c,0}\mathbf{z}_0 + \mathbf{z}_0^\top Q_{0,w}\mathbf{z}_w \\ & -\mathbb{1}_w^\top \mathbf{z}_w - \mathbb{1}_c^\top \mathbf{z}_c - \mathbb{1}_0^\top \mathbf{z}_0 \\ & -\boldsymbol{\alpha}_w^\top \mathbf{z}_w - \boldsymbol{\alpha}_c^\top \mathbf{z}_c - \boldsymbol{\alpha}_0^\top \mathbf{z}_0 \\ & -\boldsymbol{\beta}_w^\top (C\mathbb{1}_w - \mathbf{z}_w) - \boldsymbol{\beta}_c^\top (C\mathbb{1}_c - \mathbf{z}_c) - \boldsymbol{\beta}_0^\top (C\mathbb{1}_0 - \mathbf{z}_0) \\ & +\mu(\mathbf{y}_w^\top \mathbf{z}_w + \mathbf{y}_c^\top \mathbf{z}_c + \mathbf{y}_0^\top \mathbf{z}_0), \end{aligned} \tag{1.49}$$

where if u, v denote any of the sets $\{\mathcal{I}_0, \mathcal{I}_w, \mathcal{I}_c\}$, then $Q_{u,v}$ denotes the submatrix formed by the u rows and v columns of Q and $\mathbf{z}_u$ and $\mathbb{1}_v$ denote the sub-vectors containing the u and v entries of vector $\mathbf{z}$ or $\mathbb{1}$, respectively. It has to be noticed that this equation can be simplified as all $z_i \in \mathcal{I}_0$ are equal to zero and all $z_i, i \in \mathcal{I}_c$ are equal to C.

If we knew the repartition of the variables in these sets, we could reduce the size of the problem as we know that all z_i in $\mathcal{I}_0$ have zero value and all variables in $\mathcal{I}_c$ are equal to C. The remaining problem is the calculation of the exact value of variables in $\mathcal{I}_w$. Applying the same partition, the objective function can be rewritten in terms of these sets as follows:

$$\frac{1}{2}\mathbf{z}^\top Q\mathbf{z} - \mathbb{1}^\top \mathbf{z} = \frac{1}{2}\mathbf{z}_w^\top Q_{w,w}\mathbf{z}_w + C\mathbf{z}_w^\top Q_{w,c}\mathbb{1}_c - \mathbb{1}_w^\top \mathbf{z}_w + K$$

where $K = \frac{1}{2}\mathbf{z}_c^\top Q_{c,c}\mathbf{z}_c - \mathbb{1}_c^\top \mathbf{z}_c$ is a constant value.

Analogously, constraint $\mathbf{y}^\top \mathbf{z} = 0$ can be rewritten as:

$$\mathbf{y}_w^\top \mathbf{z}_w + C\mathbf{y}_c^\top \mathbb{1}_c = 0$$

Using this configuration, the optimization problem can be restated as follows:

$$\begin{array}{ll} \underset{\mathbf{z}_w}{\text{minimize}} & \frac{1}{2}\mathbf{z}_w^\top Q_{w,w}\mathbf{z}_w - (\mathbb{1}_w - CQ_{w,c}\mathbb{1}_c)^\top \mathbf{z}_w \\ \text{subject to} & \mathbf{y}_w^\top \mathbf{z}_w + C\mathbf{y}_c^\top \mathbb{1}_c = 0 \end{array} \tag{1.50}$$

It has to be noticed that in the case the sets are known, Constraint (1.46b) is not longer necessary as it is known that when optimality is reached, all variables in $\mathcal{I}_w$ would be between zero and C. The Lagrangian function of Problem 1.50 is therefore:

$$\mathbf{L}(\mathbf{z}_w, \mu) = \frac{1}{2}\mathbf{z}_w^\top Q_{w,w}\mathbf{z}_w - (\mathbb{1}_w - CQ_{w,c}\mathbb{1}_c)^\top \mathbf{z}_w + \mu(\mathbf{y}_w^\top \mathbf{z}_w + C\mathbf{y}_c^\top \mathbb{1}_c)$$

and the KKT optimality conditions for Problem 1.50 can be stated as:

$$\begin{gathered} Q_{w,w}\mathbf{z}_w - \mathbb{1}_w + CQ_{w,c}\mathbb{1}_c + \mu\mathbf{y}_w = 0 \\ \mathbf{y}_w^\top \mathbf{z}_w + C\mathbf{y}_c^\top \mathbb{1}_c = 0. \end{gathered}$$

It has to be noticed that these equations are derived from $|\mathcal{I}_w| + 1$ unknown variables ($\mathbf{z}_w$ and μ) and there are the same number of equations, therefore these two conditions can be stated as a linear system:

$$\begin{pmatrix} Q_{w,w} & \mathbf{y}_w \\ \mathbf{y}_w^\top & 0 \end{pmatrix} \begin{pmatrix} \mathbf{z}_w \\ \mu \end{pmatrix} = \begin{pmatrix} \mathbb{1}_w - CQ_{w,c}\mathbb{1}_c \\ -C\mathbf{y}_c^\top \mathbb{1}_c \end{pmatrix} \tag{1.51}$$

If after the resolution of this system, there are variables that violate the box constraints in Equations (1.46b), that is, if there are negative z_i or z_i is bigger than C, this shows that the initial supposed repartition of sets $\mathcal{I}_0, \mathcal{I}_w$ and $\mathcal{I}_c$ is incorrect, therefore, it will be necessary to readjust the sets by moving points from $\mathcal{I}_w$ to $\mathcal{I}_0$ or to $\mathcal{I}_c$.

In the case that all the variables satisfy the constraints, it is still not enough for the optimality since conditions in Equations (1.48c) and (1.48d) have to be verified for the original problem. To calculate the value of the Lagrange multipliers $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, some equivalences must be established.

It has to be noticed that for the variables that are in $\mathcal{I}_0$, Lagrange multipliers must hold $\boldsymbol{\beta}_0 = \mathbf{0}_0$ to satisfy Equation (1.48b), therefore, if the Lagrangian in Equation (1.49) goes to zero because a change in $\mathbf{z}_0$ implies that its gradient $\nabla_{\mathbf{z}_0}\mathbf{L} = Q_{0,0}\mathbf{z}_0 + Q_{0,c}\mathbf{z}_c + Q_{0,w}\mathbf{z}_w - \mathbb{1}_0 - \boldsymbol{\alpha}_0 + \mu\mathbf{y}_0 = 0$, and since the condition to be verified is $\boldsymbol{\alpha}_0 \geq 0$, the following inequality is to be verified:

$$Q_{0,\cdot}\mathbf{z} - \mathbb{1}_0 + \mu\mathbf{y}_0 \geq 0,$$

where $Q_{0,\cdot}$ denotes all columns of Q and rows in $\mathcal{I}_0$. Similarly, for the variables that are in $\mathcal{I}_c$, Lagrange multipliers $\boldsymbol{\alpha}_c = \mathbf{0}_c$ to satisfy Equation (1.48a), then, if the Lagrangian in Equation (1.49) goes to zero because a change in $\mathbf{z}_c$ implies that its gradient $\nabla_{\mathbf{z}_c}\mathbf{L} = Q_{c,c}\mathbf{z}_c + Q_{c,0}\mathbf{z}_0 +$

$Q_{c,w}\mathbf{z}_w - \mathbb{1}_c - \boldsymbol{\alpha}_c - \boldsymbol{\beta}_c + \mu\mathbf{y}_c = 0$, and the condition to be fulfilled is $\boldsymbol{\beta}_c \geq 0$, then the following inequality has to be verified:

$$- Q_{c,\cdot}\mathbf{z} + \mathbb{1}_c - \mu\mathbf{y}_c \geq 0.$$

If it not the case, the most violating point in $\mathcal{I}_0$ or $\mathcal{I}_c$ must change from set to go into $\mathcal{I}_w$.

So, the active set method consists in initializing sets $\mathcal{I}_w$,$\mathcal{I}_0$ and $\mathcal{I}_c$ and iteratively changing violating points from set until all optimality conditions are satisfied.

1.6.3 Complexity of Quadratic Solvers

In [Bott 07], an analysis of the complexity of SVM QP solvers is provided: there are given two intuitive lower bounds on the computational cost of any algorithm that solves the SVM problem for arbitrary kernel matrices K_{ij}. If a teacher or an oracle indicates the examples which are not support vectors, that is, those with $\mathbf{z}_i = 0$, and the examples which are bounded support vectors ($\mathbf{z}_i = C$), the coefficients of the R remaining free support vectors are determined by a system of R linear equations representing the derivatives of the objective function. The resolution of the system typically requires a number of operations proportional to R^3. Simply verifying that a vector $\boldsymbol{\alpha}$ is a solution of the SVM problem involves computing the gradient $\mathbf{g}$ of the dual and checking the optimality conditions. With n examples and S support vectors, this requires a number of operations proportional to $n \cdot S$. Few support vectors reach the upper bound C when it gets large. The cost is then dominated by the term $R^3 \approx S^3$. The final number of support vectors therefore is the critical component of the computational cost of solving the dual problem.

Since the asymptotic number of support vectors grows linearly with the number of examples, the computational cost of solving the SVM problem has both a quadratic and a cubic component. It grows at least like n^2 when C is small and n^3 when C gets large. Empirical evidence shows that modern SVM solvers come close to these scaling laws.

1.7. Conclusions

In this chapter, the framework for a learning problem was established in terms of a multi-criteria optimization (a balance to be set between the accuracy and the complexity of the model), these objectives are in general opposite to each other and their interaction is controlled through a regularization parameter. The problem of model selection is then exposed to clarify that a satisfactory learner depends not only in the proper problem statement but also in the right parameter choice. The idea of learning problem and model selection is illustrated with two particular cases: the classification and the ranking problem. A priori, the primal formulation can seem difficult to solve, but by means of the development and application of continuous optimization results, a primal problem can be transformed into another one like quadratic problems. The advantage of doing this, is that the resolution of this type of problems has already been largely studied and efficient methods to solve it have been developed. The disadvantage of this approach consists in the fact that in order to do model selection, several optimization problems with different parameters have to be solved for the given data partition. Unfortunately, the resolution of each one by continuous methods (methods that take advantage of the function continuity) implies a non-negligible computational cost as the number of data grows.

The reduction of this computational time is the target of this research. In later chapters, it is shown that the solution of a problem with quadratic regularization function and linear loss is piecewise linear with respect to the regularization parameter, leading to an efficient problem resolution not only for a particular parameter but for the complete parameter domain. This result will be proved and efficiently used in the following chapters.

2 Model Selection via Regularization Paths in Supervised Learning

As explained in the previous chapter, the SVM framework involves some user's choices (regularization parameter, kernel choice, features selection). In the sequel, we will focus our interest in the derivation of tools to tackle the automatic choice of the regularization parameter.

The first part of this chapter is focused on piecewise linear solutions [Ross 07b, Hast 04], which include the general form of the solution set and a complete development of the regularization path derived for the Support Vector machines for Classification (SVC). A ***result*** is proved here, showing that the ***subgradient*** coefficients and the ***Lagrange multipliers*** of this problem are ***equal*** at the optimality if the solution is unique.

The second part, Section 2.3, includes analysis of this path in terms of a validation set, called the ***Validation Path***, it is shown how an error path for the validation set can be easily built in order to make faster model selection. It is proved that it is enough to calculate the validation error in the regularization path breakpoints in order to get the proper regularization parameter.

In Section 2.4, the ranking problem is approached with a SVM-type framework. The problem is reformulated in order to obtain a ***significant smaller problem*** than in the original RankSVM framework with a low cost in accuracy. Model selection is proposed based on the ***regularization path for RankSVM***, which again uses the property that an SVM solution structure is mainly characterized by the choice of the more relevant points, namely the support vectors. The new framework will also improve the calculation of the regularization path as the problem to be solved has been considerably reduced, additionally, a large amount of singularities are removed.

As several classification datasets are used to be solved by a ranking algorithm, the equivalence of a classification and a raking framework, under certain assumptions, is proved in Section 2.4.1.

Finally, Section 2.5 considers a ranking function as an expansion over the basis formed by the functions $\{\mathbf{k}(\mathbf{x}_i,\cdot)\}_{i=[\![n]\!]}$. In this section, we explore the learning problem with the point of view of sparsity by considering a ℓ_2 loss function together with a ℓ_1 regularizer to promote a ranking solution with few points involved in.

2.1. Piecewise Linear Solutions

In this section, we are interested in multi-criteria problems like the one in Equation (1.7), page 24. The results established for classification could be extended to other SVM type algorithms like ranking [Zapi 08b], regression [Gunt 06, Gass 07a] and density estimation [Rako 07a]. The main advantage of this formulation is that the solution set has a linear form by intervals, that is, if we express the optimal solution $\boldsymbol{\alpha}$ as a function of λ in a particular interval, this will be linear. To realize the potential of this property, a non-exhaustive list of the derived algorithms and extensions based on it can be named: *Elastic net* (double penalization L_1 and L_2) [Zou 05], *Fused Lasso* (L_1 and total variation penalizations) [Tibs 05], *Grouped Lasso* [Yuan 06], *Least absolute deviation regression* (L_1 loss and L_1 penalization) [Wang 06c], *Non-negative garotte* [Yuan 07], *L_1 penalization in infinite dimension* [Ross 07a], *Graph data and Lasso* [Tsud 07], *L_1-norm SVM* (SVM with L_1 penalty) [Zhu 04], *Asymetric cost SVM* [Bach 06], *Doubly regularized SVM* [Wang 06b], *ν-SVM* [Loos 07a], *SVR extensions* [Gunt 06, Wang 06c, Gass 07b], *Laplacian Semi-supervised SVM* [Wang 06a, Gass 07c], *One-class SVM* [Rako 07a], *Ranking SVM* [Zapi 08b]. In all these problems, the cost is piecewise linear and the regularization is quadratic or vice versa. In this section, we will take advantage of this fact to characterize the set of optimal solutions with respect to the regularization parameter, as this turns out to be piecewise linear for the chosen multi-objective criteria.

This kind of framework has been studied in the past in parametric quadratic programming [Mark 59], least squares problems [Osbo 99] and more generally as piecewise-linear solutions [Ross 07b]. We are interested in the calculation of the solution for the Tikhonov formulation for all λ (see Definition 1.15 and Equation (1.7)).

We recall the type of problems our attention is focused on:

$$\hat{f}^{\lambda}(\cdot) = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \, \mathcal{L}(f, S) + \lambda \Omega(f) \tag{2.1}$$

For notation simplicity, we will denote $\hat{f}^{\lambda}(\cdot) = f^{\lambda}$ or simply $\hat{f}^{\lambda}(\cdot) = f$. As one can see, with this definition, the objective of getting all possible solutions can have a high computational cost since, in principle, it is infinite. However, for finite training sample size, it will be seen that the whole optimal solution set can be expressed with only a finite number of solutions as the rest of the solutions can be derived from a linear relation of the firsts. As the set of relevant parameters is finite, we can enumerate them by t and denote as λ^t the parameter λ at step t and the solution $\hat{f}^{\lambda^t}(\cdot)$ as f^t. In order to gain in efficiency, the family of piecewise linear solution paths is of particular interest. To highlight this fact, we consider the following definition.

Definition 2.1 (Piecewise Linear Solution Path). *The solution set is said to have a piecewise linear path when there exists a strictly decreasing (or increasing) sequence λ^t, $t = [\![N]\!]$ such that:*

$$\begin{aligned} f = f^t + (\lambda - \lambda^t)\, h^t \qquad & \forall \lambda \in [\lambda^{t+1}, \lambda^t] \;\; \textit{with} \;\; \lambda^{t+1} < \lambda^t \\ (\textit{resp.}\; f = f^t + (\lambda - \lambda^t)\, h^t \qquad & \forall \lambda \in [\lambda^{t}, \lambda^{t+1}] \;\; \textit{with} \;\; \lambda^{t} < \lambda^{t+1}) \end{aligned}$$

where h^t, $t = [\![N]\!]$ denotes a sequence of functions in $\mathcal{H}$.

With this property, it is easy to efficiently generate the whole path of solution. Indeed, in such a case, one only needs the sequence λ^t and the corresponding h^t. Any other functions in-between can be simply obtained by linear interpolation. This means that the whole regularization path can be completely defined by a finite set of regularization values $\{\lambda^t\}_{t=[\![N]\!]}$. This set together with its corresponding optimal decision functions set $\{f^t\}_{t=[\![N]\!]}$ are sufficient to easily reconstruct the rest of the solutions with low computational cost. Therefore, the QP does not need to be solved from scratch for each λ value. The advantage lies in the fact that it has been experimentally shown that a single QP resolution is only slightly less expensive than the computation of the whole regularization path [Hast 04].

Hence, owing to such property, the computational cost of obtaining the whole path of solutions may be of the order of a single solution computation.

The question induced by this remark is to find which kind of objective functions induce the solution path to be piecewise linear. Rosset stated the necessary conditions for the problem in Equation (2.1) to admit a linear solution path. The main result is summarized by the theorem below.

Theorem 2.2 (Piecewise Linear Solution Path Conditions [Ross 07b]). *Assume the loss $\mathcal{L}(f, S)$ and the regularizer $\Omega(f)$ are convex functions. If one objective function (either $\mathcal{L}(f, S)$ or $\Omega(f)$)*

is piecewise linear and the other one piecewise quadratic then the solution path of the Problem in Equation (2.1) *is piecewise linear.*

Proof. [Ross 07b] Assume $\mathcal{L}(f, S)$ and $\Omega(f)$ are twice differentiable in a neighborhood of f^t, the solution of Equation (2.1) corresponding to λ^t. Let also $\lambda = \lambda^t + \delta_\lambda$ and its corresponding solution f. Consider finally $J(f, \lambda) = \mathcal{L}(f, S) + \lambda\Omega(f)$ and $J(f^t, \lambda) = \mathcal{L}(f^t, S) + \lambda\, \Omega(f^t)$. The optimality conditions associated to f^t and f are, respectively:

$$\begin{aligned} \nabla_f\, J(f^t, \lambda^t) &= \nabla_f\, \mathcal{L}(f^t, S) + \lambda^t\, \nabla_f\, \Omega(f^t) = 0 \\ \nabla_f\, J(f^t, \lambda) &= \nabla_f\, \mathcal{L}(f^t, S) + \lambda\, \nabla_f\, \Omega(f^t) = 0 \end{aligned} \tag{2.2}$$

where ∇_f represents the functional gradient in $\mathcal{H}$. For small values of $\delta_\lambda = (\lambda - \lambda^t)$ we can approximate $\nabla_f\, J(f^t, \lambda)$ (viewed as a function of λ) by considering the following first order Taylor expansion of Equation (2.2) around λ^t:

$$\begin{aligned} \nabla_f J(f^t, \lambda) &= \nabla_f\, \mathcal{L}(f^t, S) + \lambda^t \nabla_f \Omega(f^t) \\ &\quad + \delta_\lambda \left[\nabla_f^2 \mathcal{L}(f^t, S)\, \frac{\partial f}{\partial \lambda}\bigg|_{\lambda^t} + \lambda^t \nabla_f^2 \Omega(f^t)\, \frac{\partial f}{\partial \lambda}\bigg|_{\lambda^t} + \frac{\partial \lambda}{\partial \lambda}\bigg|_{\lambda^t} \nabla_f \Omega(f^t) \right] + \epsilon(\delta_\lambda^2) \\ &= \nabla_f\, \mathcal{L}(f^t, S) + \delta_\lambda \nabla_f^2 \mathcal{L}(f^t, S)\, \frac{\partial f}{\partial \lambda}\bigg|_{\lambda^t} \\ &\quad + \lambda^t \left(\nabla_f \Omega(f^t) + \delta_\lambda \nabla_f^2 \Omega(f^t)\, \frac{\partial f}{\partial \lambda}\bigg|_{\lambda^t} \right) + \delta_\lambda \nabla_f \Omega(f^t) + \epsilon(\delta_\lambda^2) \end{aligned}$$

where the last equation is valid because $\frac{\partial \lambda}{\partial \lambda}\big|_{\lambda^t} = 1$. If the functions are quadratic or linear, this Taylor expansion will be exact as the derivative of third order or more will be zero: $\epsilon(\delta_\lambda^2) = 0$. This is not necessary for the following result. Using the previous approximation together with Equation (2.2), we have the following equivalent limit:

$$\lim_{\delta_\lambda \to 0} \frac{\nabla_f\, J(f^t, \lambda^t) - \nabla_f\, J(f^t, \lambda)}{\delta_\lambda} = \frac{\partial f}{\partial \lambda}\bigg|_{\lambda^t} \nabla_f^2\, \mathcal{L}(f^t, S) + \lambda^t\, \frac{\partial f}{\partial \lambda}\bigg|_{\lambda^t} \nabla_f^2 \Omega(f^t) + \nabla_f \Omega(f^t) = 0$$

that gives

$$\frac{\partial f}{\partial \lambda}\bigg|_{\lambda^t} = -\left(\nabla_f^2\, \mathcal{L}(f^t, S) + \lambda^t\, \nabla_f^2 \Omega\left(f^t\right) \right)^{-1} \nabla_f \Omega(f^t)$$

The piecewise behavior is possible if $\frac{\partial f}{\partial \lambda}\big|_{\lambda^t}$ is constant. To fulfill this condition, it is required $\nabla_f^2 \Omega(f^t)$ to be zero (independence with respect to λ) and $\nabla_f^2\, \mathcal{L}(f^t, S)$ be constant. The latter condition is satisfied as the loss is assumed to be quadratic and the regularizer linear, completing the proof. The same can be straightforwardly proved for quadratic regularizer and linear loss. □

If the constant derivative $\frac{\partial f}{\partial \lambda}\big|_{\lambda^t}$ is known in each interval, the solution for any λ can be straightforwardly given.

The piecewise variation of the learning function with respect to the regularization parameter λ was established for an optimization problem of the form $\min_f \mathcal{L}(f) + \lambda\Omega(f)$. The linear variation holds if $\nabla_f^2 \Omega(f) = 0$ and $\nabla_f^2 \mathcal{L}(f) =$ constant. One can remark that the same function f minimizes $C\mathcal{L}(f) + \Omega(f)$ with $C = \frac{1}{\lambda}$. Hence using the same arguments as in Theorem 2.2, we deduce that f is linear w.r.t. C if the loss function satisfies $\nabla^2 \mathcal{L}(f) = 0$ and the regularizer meets $\nabla^2 \Omega(f) =$ constant. The need to turn back to this formulation originates from the formulation of SVM where the regularizer $\Omega(f) = \|f\|_{\mathcal{H}}^2$ is quadratic and the loss is piecewise linear.

This piecewise linear property will help to find efficiently the complete set of solutions of the problem as a function of the regularization parameter if the rest of the parameters remain fixed [Efro 04, Hast 04]. It can also help on linear programming for feature selection [Yao 07] or for functional component pursuit [Yao 08]. The regularization path can be combined with the search of other parameters [Wang 06b] or to make an analogous search [Ross 08]. There exists other methods to obtain the complete solution set like the predictor-corrector algorithm [Bach 05] when the conditions of Theorem 2.2 are not met but those algorithms are out of the scope of this work. More references can be consulted in [Gass 08a].

2.2. SVM's Regularization Path for Classification

The SVM problem is formed by a linear cost function (the hinge-loss function) and a quadratic regularization function (norm in the Hilbert space), there exists a neighborhood of solutions (where the functions are differentiable) around an optimal solution where the conditions of piecewise linearity stated in Theorem 2.2 are satisfied. We will 2.1.1 be interested in explicitly formulating this linear relation which has been derived by Hastie [Hast 04]. We recall Problem 1.17:

$$\begin{aligned} \underset{f_0\in\mathcal{H},b\in\mathbb{R},\boldsymbol{\xi}\in\mathbb{R}^n}{\text{minimize}} \quad & \sum_{i=1}^{n}\xi_i + \frac{\lambda}{2}\|f_0\|^2_{\mathcal{H}} && \text{(2.3a)}\\ \text{s. t.} \quad & y_i(f_0(\mathbf{x}_i)+b) \geq 1-\xi_i,\ i=[\![n]\!] && \text{(2.3b)}\\ & \xi_i \geq 0,\ i=[\![n]\!]. && \text{(2.3c)} \end{aligned}$$

The decision function is defined as $f(\mathbf{x}) = f_0(\mathbf{x}) + b$. The piecewise linearity of the SVC problem can be deduced in two manners, the first approach gives a solution by using the subdifferential directly in the primal problem, while the second approach introduces a Lagrangian function to achieve the result as in [Hast 04].

For the resolution of the previous problem, the definition of subdifferential is needed. The complementary notions used throughout to review shortly the subdifferential and the optimality conditions in the convex case are explained in Appendix A.1.2.

Definition 2.3 (Subdifferential). *Let $g(\mathbf{z}) : \mathcal{X} \to \mathbb{R}$, with $\mathcal{X}$ a vector space, a proper and convex function with the domain of definition $Dom(g) \subseteq \mathcal{X}$. The **subdifferential** of g at $\mathbf{z}_0 \in Dom(g)$ is the non-empty convex set*

$$\partial_{\mathbf{z}} g(\mathbf{z}_0) = \{\boldsymbol{\nu} \in \mathcal{X} : g(\mathbf{z}) - g(\mathbf{z}_0) \geq \langle \boldsymbol{\nu}, \mathbf{z}-\mathbf{z}_0\rangle, \forall \mathbf{z} \in Dom(g)\}.$$

Each $\boldsymbol{\nu} \in \partial_{\mathbf{z}} g(\mathbf{z}_0)$ is called ***subgradient*** of g at $\mathbf{z}_0$.

If the function is differentiable at point $\mathbf{z}_0$, then, the subdifferential is a set containing a single element and it coincides with the gradient [Bonn 06].

The subdifferential is provided with similar properties as the gradient, like the sum and the chain rules [Schi 07] (necessary definitions are summarized in the Appendix A.1.2):

Definition 2.4 (Proximal subgradient). *A vector ν in $\mathbb{R}^n$ is said to be a proximal subgradient of f at $\mathbf{x}_0$ provided that there exist a neighborhood U of $\mathbf{x}_0$ and a number $\sigma > 0$ that*

$$\partial_P f(\mathbf{x}_0) = \{\nu \in U : f(\mathbf{x}) - f(\mathbf{x}_0) \geq +\langle \nu, \mathbf{x}-\mathbf{x}_0\rangle - \sigma\|\mathbf{x}-\mathbf{x}_0\|^2\ \forall \mathbf{x} \in U\}.$$

the set $\partial_P f(\mathbf{x}_0)$ is the proximal subdifferential.

Proposition 2.5 (Sum Rule (subdifferentials)). *Let $f_0, f_1, \ldots, f_n : \mathcal{X} \to \bar{\mathbb{R}}$ be proper and convex functions. Assume there exists $\mathbf{x}_0 \in Dom(f_0) \cap Int(Dom(f_1)) \cap \cdots \cap Int(Dom(f_n))$ such that f_i is continuous at $\mathbf{x}_0$ for $i = [\![n]\!]$. Then for each $\mathbf{x} \in Dom(f_0) \cap Dom(f_1) \cap \cdots \cap Dom(f_n)$, it holds*

$$\partial(f_0 + f_1 + \cdots + f_n)(\mathbf{x}) = \partial(f_0)(\mathbf{x}) + \partial(f_1)(\mathbf{x}) + \cdots + \partial(f_n)(\mathbf{x}).$$

This result can be proved by using the sandwich Theorem A.17 page 146 (see [Schi 07] for details). Additionally, [Clar 98] gives (Section 9, page 59) a chain rule theorem for $\partial(g \circ f)(x)$ when f is locally Lipschitz and g is Lipschitz near $f(x)$. A more general chain rule is here presented in

A.18, page 146. If the Clarke subdifferential (see [Schi 07]) and the subdifferential of the hinge loss function g coincide, this theorem could be applied with f is locally L-continuous.

The main purpose of the subdifferential is to detect minimum points. This can be first considered for unconstrain minimization [Schi 07]: if $f : \mathcal{X} \to \bar{\mathbb{R}}$ is convex and $\mathbf{x}^* \in \mathrm{Dom}(f)$, then we obtain by definition:

$$f(\mathbf{x}^*) = \min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \quad \Longleftrightarrow \quad 0 \leq f(\mathbf{x}) - f(\mathbf{x}^*)\ \forall \mathbf{x} \in \mathcal{X} \quad \Longleftrightarrow \quad 0 \in \partial f(\mathbf{x}^*). \tag{2.4}$$

This result is formalized in Theorem 4.36, page 94 in [Mord 03].

If there are constraints, local optimality conditions can be derived by using the subdifferential (see appendix A.1.2).

The importance of the obtained result is that **both the subdifferential and the Lagrange multipliers of the SVM optimization problem coincide if the problem has a unique solution**.

Theorem 2.6 (Equivalence of Subgradient and Lagrange Multipliers)**.** *If the SVM problem as stated in Equations* (2.3) *has a unique solution, the subgradient vector at optimality is equal to the derived Lagrange multipliers.*

Proof. The proof consists in deriving the form of the decision function in terms of the subdifferential and with the Lagrangian method. The decision function is in both cases a weighted linear combination where the weights are actually the corresponding subgradient and the corresponding Lagrange multipliers. If the solution is unique, the subgradient and the Lagrange multipliers must be the same.

The decision function is derived by the two approaches, resulting in the equivalent formulations (2.13) and (2.17).

2.2.1 Subdifferential Approach Derivation

Aiming at deriving directly the primal function, we can set $z = x \cdot y$ and $f(z) = z$, so that $\ell(f, x, y)$ can be expressed as $\ell(z)$. The hinge loss function can be rewritten as:

$$\ell(z) = \max\{0, 1 - z\} = \begin{cases} 0 & \text{if } z \geq 1 \\ 1 - z & \text{if } z < 1 \end{cases} \tag{2.5}$$

This implies that the primal problem in Equations (2.3) can be also expressed as the following learning problem:

$$\underset{f_0\in\mathcal{H}, b\in\mathbb{R}}{\text{minimize}} \qquad J(f_0, b) = \sum_{i=1}^{n} \max\{0, 1 - y_i(f_0(\mathbf{x}_i) + b)\} + \frac{\lambda}{2}\|f_0\|^2_{\mathcal{H}} \tag{2.6}$$

In order to check for optimality, the hinge-loss function has to be differentiated. This function is continuous and differentiable everywhere except at $z = 1$ (see Figure 1.3(b), page 20):

$$\ell'(z) = \begin{cases} -1 & \text{if } \ z < 1 \\ 0 & \text{if } \ z > 1. \end{cases}$$

At $z = 1$, the subdifferential will be considered.

All the previous results of subdifferentials will be used to derive the solution of the optimization problem in Equation (2.6). The Hinge loss is not differentiable at $\mathbf{z}_0 = 1$, but its subdifferential at this point can be calculated as follows:

$$\begin{aligned}
\partial_{\mathbf{z}}\ell(1) &= \{\nu \in \mathbb{R}: \ \ell(\mathbf{z}) - \ell(1) \geq \langle \nu, \mathbf{z}-1\rangle \quad \text{for all } \mathbf{z} \in \mathbb{R} \ \} \\
&= \left\{ \nu \in \mathbb{R}: \left\{ \begin{array}{cc} 0-0 \geq \nu(\mathbf{z}-1) & \text{if } \mathbf{z} \geq 1 \\ \text{and} & \\ (1-\mathbf{z}) - 0 \geq \nu(\mathbf{z}-1) & \text{if } \mathbf{z} < 1 \end{array} \right\} \right\} \\
&= \left\{ \nu \in \mathbb{R}: \left\{ \begin{array}{cc} 0 \geq \nu(\mathbf{z}-1) & \text{if } \mathbf{z}-1 \geq 0 \\ \text{and} & \\ (\mathbf{z}-1)(-\nu-1) \geq 0 & \text{if } \mathbf{z}-1 < 0 \end{array} \right\} \right\} \\
&= \left\{ \nu \in \mathbb{R}: \left\{ \begin{array}{cc} 0 \geq \nu & \text{if } \mathbf{z}-1 \geq 0 \\ \text{and} & \\ \nu \geq -1 & \text{if } \mathbf{z}-1 < 0 \end{array} \right\} \right\} \\
\partial_{\mathbf{z}}\ell(1) &= \{-1 \leq \nu \leq 0\} \qquad (2.7)
\end{aligned}$$

where the last equation is the result of the intersection of both admissible sets of ν. For the rest of the domain, the subdifferential equals the gradient. Hence the subdifferential of the hinge loss function is:

$$\partial_{\mathbf{z}}\ell(\mathbf{z}) = \left\{ \begin{array}{lll} -\nu, & 0 \leq \nu \leq 1 & \text{if } \mathbf{z} = 1 \\ 0 & & \text{if } \mathbf{z} > 1 \\ -1 & & \text{if } \mathbf{z} < 1. \end{array} \right. \qquad (2.8)$$

If a subgradient of $\ell(f, \mathbf{x}, y)$ at $(\mathbf{x}_i, y_i)$ is denoted as $-\alpha_i$, the subdifferential of the hinge loss function $\ell(f, \mathbf{x}, y) = \max\{0, 1 - y_i f(\mathbf{x}_i)\}$ at $y_i f(\mathbf{x}_i)$ is:

$$\partial_{y_i f(\mathbf{x}_i)}\ell(f, \mathbf{x}_i, y_i) = \left\{ \begin{array}{lll} -\alpha_i, & 0 \leq \alpha_i \leq 1 & \text{if } \ y_i f(\mathbf{x}_i) = 1 \\ \alpha_i & \alpha_i = 0 & \text{if } \ y_i f(\mathbf{x}_i) > 1 \\ -\alpha_i & \alpha_i = 1 & \text{if } \ y_i f(\mathbf{x}_i) < 1, \end{array} \right. \qquad (2.9)$$

so that

$$0 \leq \alpha_i \leq 1 \qquad \forall\ i = [\![n]\!].$$

To minimize J in (2.6), optimality conditions will be analyzed and its calculation will be divided in three parts according to the hinge-loss function differentiability, letting $f(\mathbf{x}) = f_0(\mathbf{x}) + b$:

- $\mathcal{I}_\alpha = \{i : y_i f(\mathbf{x}_i) = 1\} \quad \Longrightarrow \quad 0 \leq \alpha_i \leq 1, \quad \forall \alpha_i \in \mathcal{I}_\alpha,$

- $\mathcal{I}_0 = \{i : y_i f(\mathbf{x}_i) > 1\} \quad \Longrightarrow \quad \alpha_i = 0, \quad \forall \alpha_i \in \mathcal{I}_0,$

- $\mathcal{I}_1 = \{i : y_i f(\mathbf{x}_i) < 1\} \quad \Longrightarrow \quad \alpha_i = 1, \quad \forall \alpha_i \in \mathcal{I}_1.$

These sets will be generally referred to as $\mathcal{I}_{(\cdot)}$. Objective function J can be rewritten using this partition as:

$$\begin{aligned}
J(f_0, b) \ &= \ \tfrac{\lambda}{2}\|f_0\|^2_{\mathcal{H}} + \sum_{i \in \mathcal{I}_\alpha} \max\{0, 1 - y_i(f_0(\mathbf{x}_i) + b)\} \\
&\quad + \sum_{i \in \mathcal{I}_0} \max\{0, 1 - y_i(f_0(\mathbf{x}_i) + b)\} + \sum_{i \in \mathcal{I}_1} \max\{0, 1 - y_i(f_0(\mathbf{x}_i) + b)\}.
\end{aligned} \qquad (2.10)$$

Since J is a function with no constraints, the equivalences in Equation (2.4) will be used to find the minimum of J. The subdifferential of J is:

$$\begin{aligned}
\partial_{f_0} J(f_0, b) &= \partial_{f_0}\Bigg(\frac{\lambda}{2}\|f_0\|^2_{\mathcal{H}} + \sum_{i\in\mathcal{I}_\alpha} \max\{0, 1 - y_i(f_0(\mathbf{x}_i) + b)\} \\
&\qquad + \sum_{i\in\mathcal{I}_0} \max\{0, 1 - y_i(f_0(\mathbf{x}_i) + b)\} + \sum_{i\in\mathcal{I}_1} \max\{0, 1 - y_i(f_0(\mathbf{x}_i) + b)\}\Bigg) \\
&= \partial_{f_0}\frac{\lambda}{2}\|f_0\|^2_{\mathcal{H}} + \partial_{f_0}\sum_{i\in\mathcal{I}_\alpha} \max\{0, 1 - y_i(f_0(\mathbf{x}_i) + b)\} \\
&\quad +\partial_{f_0}\sum_{i\in\mathcal{I}_0} \max\{0, 1 - y_i(f_0(\mathbf{x}_i) + b)\} + \partial_{f_0}\sum_{i\in\mathcal{I}_1} \max\{0, 1 - y_i(f_0(\mathbf{x}_i) + b)\} \\
&= \nabla_{f_0}\frac{\lambda}{2}\|f_0\|^2_{\mathcal{H}} + \partial_{f_0}\sum_{i\in\mathcal{I}_\alpha} \max\{0, 1 - y_i(f_0(\mathbf{x}_i) + b)\} \\
&\quad +\nabla_{f_0}\sum_{i\in\mathcal{I}_0} \max\{0, 1 - y_i(f_0(\mathbf{x}_i) + b)\} + \nabla_{f_0}\sum_{i\in\mathcal{I}_1} \max\{0, 1 - y_i(f_0(\mathbf{x}_i) + b)\} \\
&= \lambda f_0(\cdot) + \partial_{f_0}\sum_{i\in\mathcal{I}_\alpha} \max\{0, 1 - y_i(f_0(\mathbf{x}_i) + b)\} - \sum_{i\in\mathcal{I}_0}\alpha_i y_i \mathbf{k}(\mathbf{x}_i,\cdot) - \sum_{i\in\mathcal{I}_1}\alpha_i y_i \mathbf{k}(\mathbf{x}_i,\cdot) \\
&= \lambda f_0(\cdot) - \sum_{i\in\mathcal{I}_\alpha}\alpha_i y_i \mathbf{k}(\mathbf{x}_i,\cdot) - \sum_{i\in\mathcal{I}_0}\alpha_i y_i \mathbf{k}(\mathbf{x}_i,\cdot) - \sum_{i\in\mathcal{I}_1}\alpha_i y_i \mathbf{k}(\mathbf{x}_i,\cdot).
\end{aligned}$$

The second equality holds thanks to the sum rule of subdifferentials. The subdifferential is replaced by the gradient in the third equality as these functions are differentiable. We recalled results in Section 1.5.1 page 38 to obtain the fourth equality. It was stated that if f belongs to a RKHS, we have: $\nabla_f\|f\|^2_{\mathcal{H}} = \nabla_f\langle f, f\rangle_{\mathcal{H}} = 2f(\cdot)$ and $\nabla_f f(\mathbf{x}) = \nabla_f\langle f, \mathbf{k}(\mathbf{x},\cdot)\rangle_{\mathcal{H}} = \mathbf{k}(\mathbf{x},\cdot)$. Finally, it has to be notice that for the hinge loss function the proximal subdifferential and the subdifferential are the equal. Additionally, the hinge loss function, and f are Lipschitz. Therefore, the chain rule of Clarke [Clar 98] was used to obtain the last equality.
Analogously,

$$\partial_b J(f_0, b) \;=\; -\sum_{i\in\mathcal{I}_0}\alpha_i y_i - \sum_{i\in\mathcal{I}_\alpha}\alpha_i y_i - \sum_{i\in\mathcal{I}_1}\alpha_i y_i$$

To find a minimum of J, we will use the equivalences in Equation (2.4) or the Remark 4.21 in [Bonn 06]: *finding* $\mathbf{z}$ *that minimizes* $f : \mathcal{X} \to \mathbb{R}$ *is equivalent to find a point* $\mathbf{z}$ *that satisfies the relation* $0 \in \partial f(\mathbf{z})$.
If $0 \in \partial_{f_0} J(f_0, b)$ and $0 \in \partial_b J(f_0, b)$, then there exists a subgradient (equivalently a vector of coefficients α_i) so that ***the optimality conditions with the subdifferential*** is:

$$\partial_{f_0} J(f_0, b) : \; -\textstyle\sum_{i\in\mathcal{I}_0}\alpha_i y_i \mathbf{k}(\mathbf{x}_i,\cdot) - \sum_{i\in\mathcal{I}_\alpha}\alpha_i y_i \mathbf{k}(\mathbf{x}_i,\cdot) - \sum_{i\in\mathcal{I}_1}\alpha_i y_i \mathbf{k}(\mathbf{x}_i,\cdot) + \lambda f_0(\cdot) = 0 \tag{2.11}$$

and

$$\partial_b J(f_0, b) : \qquad -\textstyle\sum_{i\in\mathcal{I}_0}\alpha_i y_i - \sum_{i\in\mathcal{I}_\alpha}\alpha_i y_i - \sum_{i\in\mathcal{I}_1}\alpha_i y_i = 0 \quad \Rightarrow \quad \displaystyle\sum_{i=1}^{n}\alpha_i y_i = 0 \tag{2.12}$$

with α_i the subdifferential of the hinge-loss function with respect to $y_i f(\mathbf{x}_i)$ and defined according to (2.9).
A corresponding **representer theorem** can be deduced from Equation (2.11) resulting in a explicit formulation of function f_0:

$$f_0(\cdot) = \frac{1}{\lambda}\sum_{i=1}^{n}\alpha_i y_i \mathbf{k}(\mathbf{x}_i,\cdot) \tag{2.13}$$

and the decision function can be defined

$$f(\cdot) = \frac{1}{\lambda}\left(\sum_{i=1}^{n}\alpha_i y_i \mathbf{k}(\mathbf{x}_i,\cdot) + \alpha_0\right),$$

with $\alpha_0 = \lambda b$.

2.2.2 Lagrangian Approach Derivation

In this section, the same results as above are proved by using instead a Lagrange function as done by Hastie et al. [Hast 04]. The Lagrange function will be defined for problem in Equations (2.3) as:

$$\mathbf{L} = \sum_{i=1}^{n} \xi_i + \frac{\lambda}{2}\|f_0\|^2_{\mathcal{H}} - \sum_{i=1}^{n} \alpha_i\big(y_i(f_0(\mathbf{x}_i) + b) - 1 + \xi_i\big) - \sum_{i=1}^{n} \gamma_i \xi_i \tag{2.14}$$

with Lagrange multipliers $\alpha_i \geq 0$ and $\gamma_i \geq 0$. Hence, the regularization path is again based on the optimality conditions, that, following the development in Equations (1.25), are stated as follows:

$$\frac{\partial \mathbf{L}}{\partial b} = 0 \;\;\Rightarrow\;\; \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.15}$$

$$\frac{\partial \mathbf{L}}{\partial \xi_i} = 0 \;\;\Rightarrow\;\; 0 \leq \alpha_i \leq 1 \tag{2.16}$$

$$\frac{\partial \mathbf{L}}{\partial f_0} = 0 \;\;\Rightarrow\;\; \lambda f_0(\cdot) = \sum_{i=1}^{n} \alpha_i y_i \mathbf{k}(\mathbf{x}_i, \cdot). \tag{2.17}$$

Letting $f(\cdot) = f_0(\cdot) + b$, the KKT conditions can be stated as follows:

$$\begin{array}{rcll}
\alpha_i(1 - y_i f(\mathbf{x}_i) - \xi_i) & = & 0, & i = [\![n]\!], \\
\gamma_i \xi_i & = & 0, & i = [\![n]\!], \\
\gamma_i & \geq & 0, & i = [\![n]\!] \quad \text{and} \\
\xi_i & \geq & 0, & i = [\![n]\!].
\end{array}$$

Using all these constraints, the following implications can be deduced:

$$\begin{array}{rcl}
\alpha_i = 1 \;\text{ and }\; \xi_i > 0 & \Rightarrow & y_i f(\mathbf{x}_i) < 1 \\
y_i f(\mathbf{x}_i) > 1 & \Rightarrow & \alpha_i = 0 \;\text{ and letting }\; \xi_i = 0 \text{ to minimize the objective function (2.3a)} \\
y_i f(\mathbf{x}_i) = 1 & \Rightarrow & 0 \leq \alpha_i \leq 1
\end{array}$$

Once an optimal solution $\boldsymbol{\alpha}$ is obtained with a regularization parameter λ, three sets can be defined taking into consideration the distribution of the points according to the margin:

- $\mathcal{I}_\alpha = \{i : y_i f(\mathbf{x}_i) = 1,\; 0 \leq \alpha_i \leq 1\}$, points on the margin
- $\mathcal{I}_1 = \{i : y_i f(\mathbf{x}_i) < 1,\; \alpha_i = 1\}$, points inside the margin or badly classified
- $\mathcal{I}_0 = \{i : y_i f(\mathbf{x}_i) > 1,\; \alpha_i = 0\}$, points outside the margin and well classified.

This partition is exactly the same as the one given by the subgradient approach. □

2.2.3 Piecewise Linearity for SVM

We are interested in finding the regularization path for all $\lambda \geq 0$. The aim is to find an initial solution, for example, starting with λ very large and by decreasing its value, all events will be registered as it changes, until $\lambda = 0$. When λ decreases, the regularization term $\|f_0\|^2$ will be allowed to increase as it is less penalized in the objective function. The analogous to the linear soft case is translated as a decrease of the margin (see Appendix A.2), allowing more points to move from being inside the margin (holding $y_i f(\mathbf{x}_i) < 1$) to the outside ($y_i f(\mathbf{x}_i) > 1$) that by continuity pass through the margin ($y_i f(\mathbf{x}_i) = 1$) while their α_i goes down from 1 to 0.

If the repartition of our sets $\mathcal{I}_{(\cdot)}$ were known, the values for α_i, with $i \in \mathcal{I}_0 \cup \mathcal{I}_1$ would be zero and one, respectively. Then, the only unknown variables would be the subgradients for α_j, $j \in \mathcal{I}_\alpha \cup \{0\}$. To obtain these values, it has to be observed that

$$\begin{aligned} 1 \;=\; y_j f(\mathbf{x}_j) \;&=\; \frac{1}{\lambda}\left(\sum_{i\in\mathcal{I}_\alpha} \alpha_i y_i y_j \mathbf{k}(\mathbf{x}_i,\mathbf{x}_j) + \sum_{i\in\mathcal{I}_1} y_i y_j \mathbf{k}(\mathbf{x}_i,\mathbf{x}_j) + \alpha_0 y_j\right) \quad \forall\ \mathbf{x}_j \in \mathcal{I}_\alpha \\ \Rightarrow\quad \lambda \;&=\; \sum_{i\in\mathcal{I}_\alpha} \alpha_i y_i y_j \mathbf{k}(\mathbf{x}_i,\mathbf{x}_j) + \sum_{i\in\mathcal{I}_1} y_i y_j \mathbf{k}(\mathbf{x}_i,\mathbf{x}_j) + \alpha_0 y_j \quad \forall\ \mathbf{x}_j \in \mathcal{I}_\alpha. \end{aligned} \tag{2.18}$$

Additionally, constraint $\mathbf{y}^\top \boldsymbol{\alpha} = 0$ can be considered given by the optimality condition of Equation (2.12) or (2.15) when written in vectorial form. Recalling that $\alpha_i = 0\ \forall i \in \mathcal{I}_0$, $\alpha_i = 1\ \forall i \in \mathcal{I}_1$ and using a matrix notation for the previous Equation (2.18), a linear system is derived as:

$$\begin{aligned} 0 &= \mathbf{y}_{\mathcal{I}_\alpha}^\top \boldsymbol{\alpha}_{\mathcal{I}_\alpha} + \mathbf{y}_{\mathcal{I}_1}^\top \mathbb{1}_{\mathcal{I}_1} \\ \lambda \mathbb{1}_{I_\alpha} &= Y_{\mathcal{I}_\alpha} K_{\mathcal{I}_\alpha,\mathcal{I}_\alpha} Y_{\mathcal{I}_\alpha} \boldsymbol{\alpha}_{\mathcal{I}_\alpha} + Y_{\mathcal{I}_1} K_{\mathcal{I}_1,\mathcal{I}_\alpha} \mathbf{y}_{\mathcal{I}_\alpha} + \alpha_0 \mathbf{y}_{I_\alpha} \end{aligned}$$

where $\mathbb{1}_{I_\alpha}$ and $\mathbb{1}_{I_1}$ represent a vector of ones of size $|I_\alpha|$ and $|I_1|$, respectively, $Y_{\mathcal{I}_\alpha}$ and $Y_{\mathcal{I}_1}$ is a diagonal matrix containing elements y_i, $i \in \mathcal{I}_\alpha$ or $\mathcal{I}_1$ accordingly, $\mathbf{y}, \boldsymbol{\alpha} \in \mathbb{R}^n$ are vectors containing the values of y_i or α_i, $i = [\![n]\!]$, respectively. Finally, a subvector of such a vector is denoted with a subindex like $\mathbf{y}_{\mathcal{I}_{(\cdot)}}$. These equations will give the following system of $|\mathcal{I}_\alpha| + 1$ equations and the same number of variables:

$$\begin{aligned} \begin{bmatrix} 0 & \mathbf{y}_{\mathcal{I}_\alpha}^\top \\ \mathbf{y}_{I_\alpha} & Y_{\mathcal{I}_\alpha} K_{\mathcal{I}_\alpha,\mathcal{I}_\alpha} Y_{\mathcal{I}_\alpha} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \boldsymbol{\alpha}_{\mathcal{I}_\alpha} \end{bmatrix} &= \begin{bmatrix} -\mathbf{y}_{\mathcal{I}_1}^\top \mathbb{1}_{\mathcal{I}_1} \\ \lambda \mathbb{1}_{I_\alpha} - Y_{\mathcal{I}_1} K_{\mathcal{I}_1,\mathcal{I}_\alpha} \mathbf{y}_{\mathcal{I}_\alpha} \end{bmatrix} \\ \begin{bmatrix} \alpha_0 \\ \boldsymbol{\alpha}_{\mathcal{I}_\alpha} \end{bmatrix} &= M^{-1} \begin{bmatrix} -\mathbf{y}_{\mathcal{I}_1}^\top \mathbb{1}_{\mathcal{I}_1} \\ \lambda \mathbb{1}_{I_\alpha} - Y_{\mathcal{I}_1} K_{\mathcal{I}_1,\mathcal{I}_\alpha} \mathbf{y}_{\mathcal{I}_\alpha} \end{bmatrix} \end{aligned} \tag{2.19}$$

where $M = \begin{bmatrix} 0 & \mathbf{y}_{\mathcal{I}_\alpha}^\top \\ \mathbf{y}_{I_\alpha} & Y_{\mathcal{I}_\alpha} K_{\mathcal{I}_\alpha,\mathcal{I}_\alpha} Y_{\mathcal{I}_\alpha} \end{bmatrix}$.
For known sets $\mathcal{I}_{(\cdot)}$, the subgradient values $\alpha_i, i = [\![n]\!]$ at a fixed λ can be seen as unknow variables which can be deduced via the solution of a linear system.
Two important results are obtained by this last Equation (2.19). The **first** result is that if the three set $\mathcal{I}_{(\cdot)}$ are known, **the value for all the subgradient variables can be straightforwardly obtained** and therefore, the decision function.
The **second** one, is that the **subgradient variables** α_i for $i \in \mathcal{I}_\alpha \cup \{0\}$ **vary in a linear manner** as the derivative is constant and this result can be obtained from Equation (2.19):

$$\begin{bmatrix} \dfrac{\partial \alpha_0}{\partial \lambda} \\ \dfrac{\partial \boldsymbol{\alpha}_{\mathcal{I}_\alpha}}{\partial \lambda} \end{bmatrix} = \boldsymbol{\eta} = M^{-1} \begin{bmatrix} 0 \\ \mathbb{1}_{I_\alpha} \end{bmatrix}.$$

with $\boldsymbol{\eta} = (\eta_0, \eta_1, \ldots, \eta_{|I_\alpha|})^\top$, where η_0, corresponds to the derivative of α_0.
An initial solution for λ very large can be easily found, the details for this initialization can be found in the work of Hastie [Hast 04]. With this solution, the values for $\boldsymbol{\alpha}$ would be known and sets $\mathcal{I}_\alpha, \mathcal{I}_1$ and $\mathcal{I}_0$ can be set. The rest of the $\boldsymbol{\alpha}$ values can be exactly determined with the linear equation:

$$\begin{bmatrix} \alpha_0 \\ \boldsymbol{\alpha}_{\mathcal{I}_\alpha} \end{bmatrix} = \begin{bmatrix} \alpha_0^t \\ \boldsymbol{\alpha}_{\mathcal{I}_\alpha}^t \end{bmatrix} + \delta_\lambda \boldsymbol{\eta} = \begin{bmatrix} \alpha_0^t \\ \boldsymbol{\alpha}_{\mathcal{I}_\alpha}^t \end{bmatrix} - (\lambda^t - \lambda)\boldsymbol{\eta} \tag{2.20}$$

or

$$\alpha_i - \alpha_i^t = -(\lambda^t - \lambda)\eta_i \quad i \in \mathcal{I}_\alpha \cup \{0\}. \tag{2.21}$$

2.2.4 Event Detection

All previous results are valid as long as sets $\mathcal{I}_{(\cdot)}$ remain unchanged, therefore, the next interest is to be able to determine when this is no longer the case, which is called an ***event***. In order to be able to detect an event, the first aim of this section is to write the variation of the decision function f and variables $\alpha_i,\ i = [\![n]\!]$ as functions of λ. The variation of f with respect to λ will be first search.
The direct consequence of the remark after Theorem 2.2 is that the function $f(\mathbf{x})$ is linear w.r.t to C. To enforce this fact, let us write

$$f(\mathbf{x}) = \frac{1}{\lambda}\left(\sum_{i=1}^{n}\alpha_i y_i \mathbf{k}(\mathbf{x}_i,\mathbf{x}) + \alpha_0\right) \qquad \Rightarrow \qquad f(\mathbf{x}) = C\left(\sum_{i=1}^{n}\alpha_i y_i \mathbf{k}(\mathbf{x}_i,\mathbf{x}) + \alpha_0\right).$$

The piecewise variation stated that $f(\cdot) = f^t(\cdot) + (C - C^t)\left.\frac{\partial f}{\partial C}\right|_{C^t}$. However, we want to retrieve the formulation of f as a function of λ and can write

$$f(\cdot) = f^t(\cdot) + \frac{\lambda^t - \lambda}{\lambda^t\lambda}\left.\frac{\partial f}{\partial C}\right|_{C^t}$$

It just remains to drop down C in the derivative $\left.\frac{\partial f}{\partial C}\right|_{C^t}$. We can see that $\partial C = \partial\frac{1}{\lambda} = -\frac{\partial\lambda}{\lambda^2}$, leading to the equation

$$f(\cdot) = f^t(\cdot) + \frac{\lambda - \lambda^t}{\lambda}\lambda^t\left.\frac{\partial f}{\partial \lambda}\right|_{\lambda^t}. \tag{2.22}$$

The derivative of the decision function with respect to λ will be:

$$\begin{aligned}\frac{\partial f}{\partial\lambda} &= \frac{1}{\lambda}\left(\sum_{i=1}^{n} y_i\mathbf{k}(\mathbf{x}_i,\cdot)\frac{\partial\alpha_i}{\partial\lambda} + \frac{\partial\alpha_0}{\partial\lambda}\right) - \frac{1}{\lambda^2}\left(\sum_{i=1}^{n}\alpha_i y_i\mathbf{k}(\mathbf{x}_i,\cdot) + \alpha_0\right)\\ &= \frac{1}{\lambda}\left(\sum_{i=1}^{n} y_i\mathbf{k}(\mathbf{x}_i,\cdot)\frac{\partial\alpha_i}{\partial\lambda} + \frac{\partial\alpha_0}{\partial\lambda}\right) - \frac{1}{\lambda}f(\cdot)\\ &= \frac{1}{\lambda}\left(\sum_{i\in\mathcal{I}_\alpha} y_i\mathbf{k}(\mathbf{x}_i,\cdot)\frac{\partial\alpha_i}{\partial\lambda} + \frac{\partial\alpha_0}{\partial\lambda}\right) - \frac{1}{\lambda}f(\cdot).\end{aligned}$$

The last equation holds while sets $\mathcal{I}_0, \mathcal{I}_\alpha, \mathcal{I}_1$ remain fixed in a neighborhood of λ. If this derivative is evaluated at λ^t we get

$$\left.\frac{\partial f}{\partial\lambda}\right|_{\lambda^t} = \frac{1}{\lambda^t}\left(\sum_{i\in\mathcal{I}_\alpha} y_i\mathbf{k}(\mathbf{x}_i,\cdot)\left.\frac{\partial\alpha_i}{\partial\lambda}\right|_{\lambda^t} + \left.\frac{\partial\alpha_0}{\partial\lambda}\right|_{\lambda^t}\right) - \frac{1}{\lambda^t}f^t(\cdot)$$

Plugging this last derivative in our piecewise variation derivation, Equation (2.22) we get

$$\begin{aligned}f(\cdot) &= f^t(\cdot) + \frac{\lambda - \lambda^t}{\lambda}\left(\sum_{i\in\mathcal{I}_\alpha} y_i\mathbf{k}(\mathbf{x}_i,\cdot)\frac{\partial\alpha_i}{\partial\lambda} + \frac{\partial\alpha_0}{\partial\lambda}\right) - \frac{\lambda - \lambda^t}{\lambda}f^t\\ f(\cdot) &= \frac{\lambda^t}{\lambda}f^t + \frac{\lambda - \lambda^t}{\lambda}\left(\sum_{i\in\mathcal{I}_\alpha} y_i\mathbf{k}(\mathbf{x}_i,\cdot)\frac{\partial\alpha_i}{\partial\lambda} + \frac{\partial\alpha_0}{\partial\lambda}\right)\\ f(\cdot) &= \frac{\lambda^t}{\lambda}f^t + \frac{1}{\lambda}\sum_{i\in\mathcal{I}_\alpha} y_i\mathbf{k}(\mathbf{x}_i,\cdot)(\alpha_i - \alpha_i^t) + (\alpha_0 - \alpha_0^t)\end{aligned} \tag{2.23}$$

where the last line is true because it was proved that $\boldsymbol{\alpha}$ follows a linear trajectory:

$$\begin{aligned} \alpha_i &= \alpha_i^t - (\lambda^t - \lambda)\eta_i = \alpha_i^t - (\lambda^t - \lambda) \left.\frac{\partial \alpha_i}{\partial \lambda}\right|_{\lambda^t}, \quad i \in \{0\} \cup \mathcal{I}_\alpha & (2.24) \\ \alpha_i &= \alpha_i^t, \quad i \in \mathcal{I}_0 \cup \mathcal{I}_1 & (2.25) \end{aligned}$$

with η_i as given in Equation (2.21) and will be true, while sets $\mathcal{I}_0$,$\mathcal{I}_\alpha$ and $\mathcal{I}_1$ remain fixed.
The final objective will be to find the next λ causing an event, that is, the one involving an evolution of the sets $\mathcal{I}_\alpha$,$\mathcal{I}_0$ and $\mathcal{I}_1$. These changes can be detected by two principal events:

1. A point $\mathbf{x}_i, i \in \mathcal{I}_\alpha$ reaches $\mathcal{I}_0$ or $\mathcal{I}_1$, that is $\alpha_i, i \in \mathcal{I}_\alpha$ reaches its boundary value zero or one. These events can be detected using Equation (2.24) for all $\alpha_i, i \in \mathcal{I}_\alpha$, that is, solving for λ the following equations:

$$1 = \alpha_i^t - (\lambda^t - \lambda)\eta_i, \quad i \in \mathcal{I}_\alpha \qquad (2.26)$$

$$0 = \alpha_i^t - (\lambda^t - \lambda)\eta_i, \quad i \in \mathcal{I}_\alpha \qquad (2.27)$$

2. A point $\mathbf{x}_i,\ i \in \mathcal{I}_0 \cup \mathcal{I}_1$ reaches the margin (that is, $y_i f(\mathbf{x}_i) = 1$). The λ causing these events can be calculated by using the fact that $\alpha_i - \alpha_i^t = -(\lambda^t - \lambda)\eta_i$ and rewriting Equation (2.23) as:

$$f(\mathbf{x}) = \frac{\lambda^t}{\lambda} f^t(\mathbf{x}) + \frac{\lambda - \lambda^t}{\lambda} h^t(\mathbf{x}) \quad \text{with} \quad h^t(\mathbf{x}) = \sum_{i \in \mathcal{I}_\alpha} y_i \mathbf{k}(\mathbf{x}_i, \mathbf{x})\eta_i + \eta_0.$$

If $\mathbf{x}_i$, with $i \in \mathcal{I}_0 \cup \mathcal{I}_1$ reaches $\mathcal{I}_\alpha$, then $y_i f(\mathbf{x}_i) = 1$, obtaining $1 = \frac{\lambda^t}{\lambda} f^t(\mathbf{x}_i) y_i + \frac{\lambda - \lambda^t}{\lambda} h^t(\mathbf{x}_i) y_i$ and the λ that causes such an event can be calculated as:

$$\lambda = \lambda^t \left(\frac{f^t(\mathbf{x}_i) - h^t(\mathbf{x}_i)}{y_i - h^t(\mathbf{x}_i)} \right), \qquad (2.28)$$

2.2.5 Regularization Path Algorithm

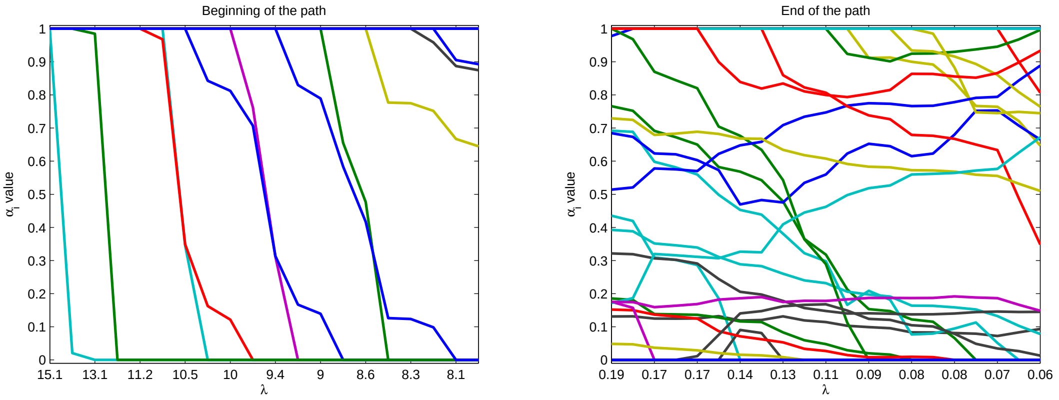
We recall the derived dual problem in Section 1.5.1:

$$\begin{aligned} \max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \quad & \mathbb{1}^\top \boldsymbol{\alpha} - \tfrac{1}{2\lambda} \boldsymbol{\alpha}^\top Y K Y \boldsymbol{\alpha} \\ \text{subject to} \quad & \mathbf{y}^\top \boldsymbol{\alpha} = 0 \qquad (2.29) \\ & \mathbf{0} \leq \boldsymbol{\alpha} \leq \mathbb{1} \end{aligned}$$

If the regularization path was started with λ^0 very large (that is, a solution $\boldsymbol{\alpha}^0$ of the QP (2.29) was obtained with λ^0), the path can be calculated by starting with this pair $(\boldsymbol{\alpha}^0, \lambda^0)$ and from it, sequentially deriving all λ's resulting from Equations (2.26), (2.27) and (2.28) as λ decreases and choosing as following step the largest $\lambda < \lambda^t$ to adjust slope $\boldsymbol{\eta}^t$.
Figure 2.1 shows an example of the trajectory that each α_i follows as λ decreases. The problem considered here is the mixture dataset [Hast 01], which is a case of binary classification that follows a distribution of Gaussian mixtures. We followed the values of the dual variables α_i as λ changed. The regularization path starts with $\lambda^0 \approx 15$ with all points in $\mathcal{I}_1$ and finalizes with $\lambda \approx .06$ when $\mathcal{I}_1$ turns empty. At each interval, α_i follows a linear way. Some instability can be observed at the end of the path.
The SVM regularization path algorithm is summarized in Algorithm 1.

Initialization issues

The algorithm can also be used starting with a very small λ and making it increase. However, the advantage of starting with a very large parameter value is that there exists techniques to efficiently obtain an initial solution : the principle is simple and consists in determining initial values λ_0 and b_0 so that all points belong at the beginning to $\mathcal{I}_1$. Then, at least one point from



(a) α's trajectories at the beginning of the path.

(b) α's trajectories at the end of the path.

Figure 2.1: Evolution of the $\boldsymbol{\alpha}$ values as λ changes.

Algorithm 1 Pseudo-code of the SVM regularization path computation

```
Input: Training set $\{(\mathbf{x}_i, y_i)\},\ i \in [\![n]\!]$.
Output: Complete solution set $\{(\boldsymbol{\alpha}_\lambda,\ \lambda)\}$.
  Set $t = 0$.
  Compute the initial value $\lambda^0$ and corresponding solution $\boldsymbol{\alpha}^0$.
  Deduce the corresponding sets $(\mathcal{I}_\alpha, \mathcal{I}_1, \mathcal{I}_0)$.
  repeat
    Compute the update direction $\boldsymbol{\eta}^t = \begin{pmatrix} 0 & \mathbf{y}^{t\top} \\ \mathbf{y}^t & K_*^t \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ \mathbb{1} \end{pmatrix}$, with $K_*^t = Y_{\mathcal{I}_\alpha^t} K_{\mathcal{I}_\alpha^t, \mathcal{I}_\alpha^t} Y_{\mathcal{I}_\alpha^t}$ .
    Use $\boldsymbol{\alpha}^t, \boldsymbol{\eta}^t$ and $f^t(\mathbf{x}_i)$ to detect all $\lambda$'s producing an event.
    Select the corresponding decreasing value $\lambda^{t+1}$.
    Save the corresponding event(s) type and the related point(s).
    Update parameter $\alpha$ using Equation (2.24).
    Update sets $(\mathcal{I}_\alpha, \mathcal{I}_1, \mathcal{I}_0)$ according to the retained event(s) and concerned point(s).
    Set $t = t + 1$.
  until $\lambda$ is small, $\mathcal{I}_1$ is empty or another termination criteria is reached.
```

each class is placed on the margin (i.e. in $\mathcal{I}_\alpha$) [Hast 04]. This procedure holds if the learning problem is perfectly balanced that is the numbers of samples in both classes are equal otherwise a QP problem is solved with user set value λ_0 to initialize the algorithm.

It is worth mentioning that for a training set, this initialization procedure provides an upper bound λ_{max} on the meaningful values of the regularization parameter. Indeed for $\lambda \in [\lambda_{max}, \infty)$, the margin is large and all the points could be positioned in $\mathcal{I}_1$ by adjusting b accordingly. Therefore in the path calculation, the focus will solely concerns values λ in the interval $[\lambda_{min}, \lambda_{max}]$, where λ_{min} has to be deduced. A guess of λ_{min}, if it is different from zero, can be derived using results in [Loos 07a] where a path for $\nu-$SVM algorithm is proposed along with an initialization that considers error-free solutions.

To conclude this part, it can be said that the beauty of the regularization path lies in the fact that the computational cost to calculate the whole path is experimentally about the same as the calculation of a single optimization problem with a fix λ [Hast 04]. After having calculated the path, not only a particular optimal solution will be obtained but the whole optimal solution set for all λ's will be in hand to apply any of the model selection methods to choose a final machine.

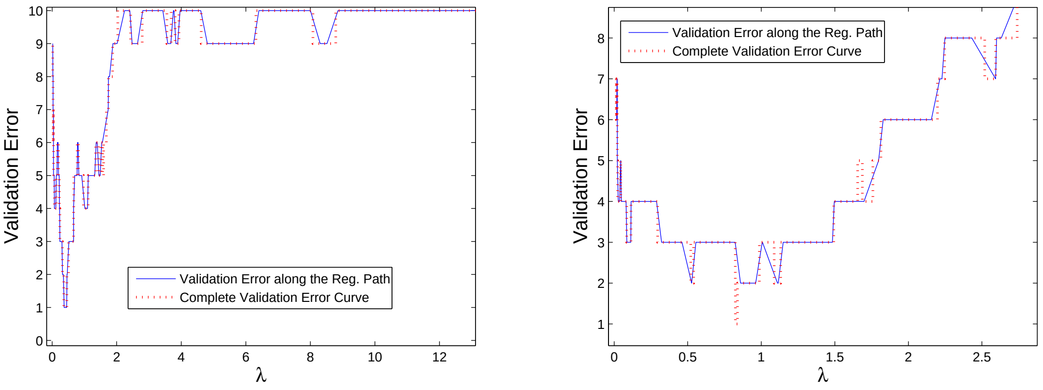
2.3. Complete Validation Error Curves

In Section 2.1, the conditions of a multi-objective function for its solution to have a piecewise linear path were established. If the conditions hold, the complete solution set can be efficiently calculated. The following question to be solved is: **How to estimate the generalization ability for each model?** The challenge in this issue lies in the fact that there is an infinite number of models to be tested. We will focus our first efforts to solve this question in the classification theory, particularly, in the *Support Vector Machines for Classification* (SVC) method, but the presented development can be straightforwardly extended to other frameworks holding the same piecewise characteristics.

As proved in the previous section, the entire regularization path will give as output all possible models, that is, the associated model to all regularization parameter λ. Some approaches have been done to automatically select the regularization parameter using the regularization path [Loos 07a] breakpoints that give a sampling of the regularization parameters.

Indeed, keeping trace of the validation error measure along the regularization path is a practical approach as relevant changes in the decision function is given at each regularization path breakpoint. Our matter in this section consists in knowing if that is a sufficient sampling of the regularization parameters to choose a model.

To illustrate this issue, consider Figure 2.2, the validation error for all λ is depicted with the dotted red line while the validation error on the regularization path breakpoints is depicted with a blue line. In Figure 2.2(a), the blue curve follows the red one even along the local minima while in Figure 2.2(b), the real minimum of the validation error curve between 0.5 and 1 is skipped by the regularization path curve.



(a) Validation error along the regularization and validation path.

(b) Validation error along the regularization and validation path. A value of λ with lower validation error is missed in the regularization path.

Figure 2.2: Validation path vs. regularization path examples. The blue line denotes the validation error at each of all breakpoints given by the regularization path. The dotted red line depicts the validation error for all regularization parameter λ, that is, the complete validation curve.

A justification of the fact that it is statistically enough to use the sampling given by the regularization path is developed in this section.

In this section, model selection for the SVC is analyzed under a training-validation-test sets framework. We take advantage of the piecewise-linear solution property in order to be able to calculate the complete validation error curve. As it will be noticed, this curve also has a piecewise form with respect to the previously defined sets $\mathcal{I}_\alpha$, $\mathcal{I}_0$ and $\mathcal{I}_1$ and new ones defined in a similar way for the validation set.

We will restrict ourself to the case when the data is partitioned in three sets for learning:

- Training Data (S_X, S_Y), with n points.
- Validation Data (S_{X_V}, S_{Y_V}), with n_V points.
- Test Data

The model choice is based on the validation data, that is, the model that outputs the minimum error on the validation data will be retained.
The regularization path reviewed in Section 2.2 outputs a set $\{\lambda^t\}$ and its corresponding solutions $\boldsymbol{\alpha}^t$ and slopes $\boldsymbol{\eta}^t$ that define the trajectory of the optimal solutions for each induced interval. Using the same convention as in Section 2.1, let us denote $f = \hat{f}^\lambda$ and $f^t = \hat{f}^{\lambda^t}$ as the corresponding solutions to λ and λ^t, respectively.
At each given model, f, with parameter λ, the validation error can be measured as:

$$E_V(\lambda, S_{X_V}, S_{Y_V}) = \sum_{\substack{i=1 \\ (\mathbf{x}_i, y_i) \in S_{X_V} \times S_{Y_V}}}^{n_V} \mathcal{L}(f, \mathbf{x}_i, y_i),$$

where $\mathcal{L}$ is a loss function and will generate in this way a ***validation error curve*** that depends on the value of λ. In order to generate the complete curve, the validation error should be measured for all regularization parameter, which turns out to be infinite. Fortunately, in this section, it is proved that this curve is piecewise constant with respect to the parameter λ.
It is later observed that, in average, observing the validation error at the regularization path breakpoints will be enough to choose a model.

2.3.1 Validation Error Curve Path

It was proved in Theorem 1.16 that as λ decreases, the training error decreases as well. Nevertheless, the validation error decreases until the model starts to overfit the training data, causing an increase in the validation error. In our case, we will consider the 0-1 loss to calculate the validation error for the model given by parameter λ:

$$E_V \quad = \quad E_V(\lambda, S_{X_V}, S_{Y_V}) \quad = \sum_{(\mathbf{x}_j, y_j) \in (S_{X_V}, S_{Y_V})} \Gamma_{y_j \cdot f(\mathbf{x}_j) < 0}$$

where Γ_u is the indicator function equals to 1 if the predicate u is true and 0 otherwise.
In a parallel approach, Rosset [Ross 08] follows the path of cross validated solutions to regularized kernel quantile regression, allowing him to efficiently solve the whole family of bi-level problems.
The validation error curve will change as λ varies. Let $\{\lambda^l\}$ be the set of breakpoints where the validation curve changes, that is, where a change in the error measure occurs. The set $\{\lambda^l\}$ will include all the breakpoints in the regularization path $\{\lambda^t\}$ additionally with breakpoints given by the change of side in the decision function of the validation points, which will be called ***validation events***. The curve given by all values $E_V^l = E_V(\lambda^l, S_{X_V}, S_{Y_V})$ for all λ^l will be called the ***validation path***.
Our first interest is to determine in an efficient manner the validation error for each λ. For this purpose, it has to be observed that the value of $f(\mathbf{x}_j)$ depends on λ and has an hyperbolic relation with respect to it see Equation (see Equation (2.22)), with $\lambda^{t+1} \leq \lambda \leq \lambda^t$ and sets $\mathcal{I}_\alpha, \mathcal{I}_0$ and $\mathcal{I}_1$ fixed in this interval.
Consider a sample $\mathbf{x}_j$ belonging to the validation set. The prediction output by the decision function for this sample is:

$$f(\mathbf{x}_j) \quad = \quad \frac{1}{\lambda}\left(\boldsymbol{\alpha}^\top Y \mathbf{k}(\mathbf{x}_j) + \alpha_0\right) \qquad \mathbf{x}_j \in S_{X_V}$$

where $\mathbf{k}(\mathbf{x}) = [\mathbf{k}(\mathbf{x}_1, \mathbf{x}) \ \cdots \ \mathbf{k}(\mathbf{x}_n, \mathbf{x})]^\top$ and n is the training set size. To simplify the notation and ease our development, here we let vector $\boldsymbol{\alpha}$ to include all the Lagrange multipliers and not only those related uniquely to support vectors. Using the piecewise linear variation $\boldsymbol{\alpha} = \boldsymbol{\alpha}^t + (\lambda - \lambda^t)\boldsymbol{\eta}^t$ on regularization path, the latter expression of $f(\mathbf{x}_j)$ takes the form

$$\begin{aligned}
f(\mathbf{x}_j) &= \frac{1}{\lambda}\left(\boldsymbol{\alpha}^\top Y K^{V}_{\cdot,j} + \alpha_0\right) \\
&= \frac{1}{\lambda}\left(\boldsymbol{\alpha}_{\mathcal{I}_\alpha}^\top Y_{\mathcal{I}_\alpha} K^{V}_{\mathcal{I}_\alpha,j} + \boldsymbol{\alpha}_{\mathcal{I}_1}^\top Y_{\mathcal{I}_1} K^{V}_{\mathcal{I}_1,j} + \alpha_0\right) \\
&= \frac{1}{\lambda}\left(\left(\boldsymbol{\alpha}^t_{\mathcal{I}_\alpha} + (\lambda - \lambda^t)\boldsymbol{\eta}^t_{\mathcal{I}_\alpha}\right)^\top Y_{\mathcal{I}_\alpha} K^{V}_{\mathcal{I}_\alpha,j} + \mathbb{1}^\top Y_{\mathcal{I}_1} K^{V}_{\mathcal{I}_1,j} + \alpha_0^t + (\lambda - \lambda^t)\boldsymbol{\eta}_0\right) \\
&= \frac{1}{\lambda}\left(\left(\boldsymbol{\alpha}^t_{\mathcal{I}_\alpha} - \lambda^t\boldsymbol{\eta}^t_{\mathcal{I}^t_\alpha}\right)^\top Y_{\mathcal{I}_\alpha} K^{V}_{\mathcal{I}_\alpha,j} + \mathbb{1}^\top Y_{\mathcal{I}_1} K^{V}_{\mathcal{I}_1,j} - \lambda^t\boldsymbol{\eta}_0 + \alpha_0^t\right) \qquad (2.30) \\
&\quad + \frac{1}{\lambda}\left(\lambda{\boldsymbol{\eta}^t_{\mathcal{I}_\alpha}}^\top Y_{\mathcal{I}_\alpha} K^{V}_{\mathcal{I}_\alpha,j} + \lambda\boldsymbol{\eta}_0\right) \\
&= \frac{1}{\lambda}\tau_j^t + \upsilon_j^t \qquad (2.31)
\end{aligned}$$

where K^{V} represents the $n \times n_V$ kernel matrix applied to the validation set, that is: $K^{V}_{i,j} = \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j), \mathbf{x}_i \in S_X, \mathbf{x}_j \in S_{X_V}$, and $K^{V}_{\cdot,j}$ is the submatrix of K^{V} containing all rows and column j. Finally, $Y_{\mathcal{I}_{(\cdot)}}$ is the diagonal matrix containing labels y_i with $i \in \mathcal{I}_{(\cdot)}$ and

$$\tau_j^t = \left(\boldsymbol{\alpha}^t_{\mathcal{I}_\alpha} - \lambda^t\boldsymbol{\eta}^t_{\mathcal{I}_\alpha}\right)^\top Y_{\mathcal{I}_\alpha} K^{V}_{\mathcal{I}_\alpha,j} + \mathbb{1}^\top Y_{\mathcal{I}_1} K^{V}_{\mathcal{I}_1,j} - \lambda^t\boldsymbol{\eta}_0 + \alpha_0^t,$$

$$\upsilon_j^t = {\boldsymbol{\eta}^t_{\mathcal{I}_\alpha}}^\top Y_{\mathcal{I}_\alpha} K^{V}_{\mathcal{I}_\alpha,j} + \boldsymbol{\eta}_0.$$

As $f(\mathbf{x}_j)$ is proportionally inverse to λ, the validation error count can be easily updated as λ moves. This is done by calculating when a validation sample is going through the zero of the decision function. As in the regularization path under the training set, our interest will be to detect the events that cause this relationship to change, therefore, three sets will be defined:

- $\mathcal{V}_0 = \{j : \mathbf{x}_j \in S_{X_V}, f(\mathbf{x}_j) = 0\}$,
- $\mathcal{V}_+ = \{j : \mathbf{x}_j \in S_{X_V}, y_j f(\mathbf{x}_j) > 0\}$,
- $\mathcal{V}_- = \{j : \mathbf{x}_j \in S_{X_V}, y_j f(\mathbf{x}_j) < 0\}$.

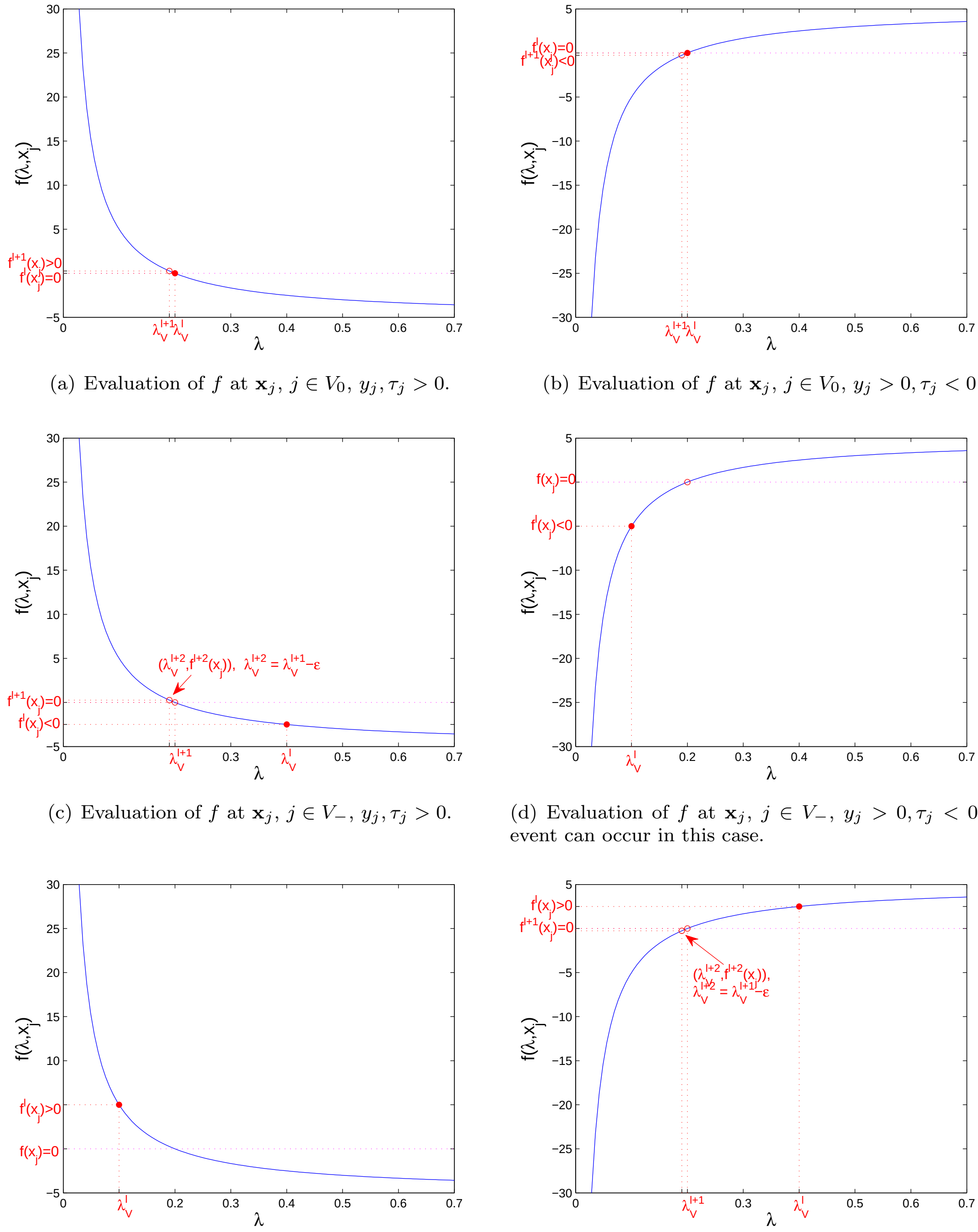
These sets contain points in the validation set that are, respectively, on the decision frontier, correctly or wrongly classified. Therefore, the λ values of interest can be determined by monitoring the ***validation events***, which are defined as the following set changes:

- $j \in \mathcal{V}_0 \longrightarrow \mathcal{V}_+ \cup \mathcal{V}_-$
- $j \in \mathcal{V}_+ \longrightarrow \mathcal{V}_0 \longrightarrow \mathcal{V}_-$
- $j \in \mathcal{V}_- \longrightarrow \mathcal{V}_0 \longrightarrow \mathcal{V}_+$

Where the first event denotes the points that are on the decision function and turn well or wrongly classified, the second one includes the points that are well classified and get exactly on the decision function. The third one is the incorrectly classified points that reach the decision function.

The previous list makes emphasis in the fact that if $\mathbf{x}_j, j \in \mathcal{V}_+$ (or $\mathcal{V}_-$) achieves the boundary of the two classes, that is, at λ^{l+1}, $f^{l+1}(\mathbf{x}_j) = 0$, event $j \longrightarrow \mathcal{V}_0$ will be produced, but, by continuity, an infinitesimal reduction of λ^{l+1}, say $\lambda^{l+2} = \lambda^{l+1} - \varepsilon$, where ε is an infinitesimal positive number, will again move $\mathbf{x}_j$ to $\mathcal{V}_-$ (or $\mathcal{V}_+$) for all $\lambda < \lambda^{l+1}$, producing a change in the validation error.

Figure 2.3 illustrates the change in the decision function for a point $\mathbf{x}_j$ in different cases for positive y_j. Figures 2.3(a), 2.3(c) and 2.3(e) consider the case when the coefficient τ_j of the evaluation of the decision function for $\mathbf{x}_j$ is positive. This would mean that if λ decreases along the regularization path, the evaluation of the decision function at $\mathbf{x}_j$ will monotonically increase. The case $\tau_j < 0$ is depicted in subfigures 2.3(b), 2.3(f) and 2.3(d).



(a) Evaluation of f at $\mathbf{x}_j$, $j \in V_0$, $y_j, \tau_j > 0$.

(b) Evaluation of f at $\mathbf{x}_j$, $j \in V_0$, $y_j > 0, \tau_j < 0$.

(c) Evaluation of f at $\mathbf{x}_j$, $j \in V_-$, $y_j, \tau_j > 0$.

(d) Evaluation of f at $\mathbf{x}_j$, $j \in V_-$, $y_j > 0, \tau_j < 0$, no event can occur in this case.

(e) Evaluation of f at $\mathbf{x}_j$, $j \in V_+$, $y_j > 0, \tau_j > 0$, no event can occur in this case.

(f) Evaluation of f at $\mathbf{x}_j$, $j \in V_+$, $y_j > 0, \tau_j < 0$.

Figure 2.3: Validation event detection for case $y_j = 1$. Evaluation of f for λ^l, λ^{l+1} and, if it applies, for λ^{l+2}. The filled red point indicates the position at λ^l and the empty one(s) illustrates the searched events (see Table 2.1).

For a specific range of λ, say an interval given by two consecutive breakpoints in the regularization path, $[\lambda^{t+1}, \lambda^t]$, the form of the evaluation of point $\mathbf{x}_j$ is known: $f(\mathbf{x}_j) = \frac{1}{\lambda}\tau_j^t + \upsilon_j^t$. We are interested in finding for $\mathbf{x}_j$ all possible validation events in this interval but only certain cases are matter of interest at each state. Let seek first, if it exists, the parameter value λ^{l+1} that causes

$$f^{l+1}(\mathbf{x}_j) \quad = \quad 0 \quad \text{with } \lambda^{t+1} < \lambda^{l+1} < \lambda^t. \tag{2.32}$$

Table 2.1 summarizes the conditions to detect validation events, which can be fast updated

Sets	Case	λ value	E_V	Sets update	Fig.
$j \in \mathcal{V}_0$ $f^l(\mathbf{x}_j) = 0$	$y_j \cdot \tau_j > 0$	$\lambda^{l+1} = \lambda^l - \varepsilon$	$E_V^{l+1} = E_V^l - 1$	$\mathcal{V}_+^{l+1} = \mathcal{V}_+^l \cup \{j\}$ $\mathcal{V}_0^{l+1} = \mathcal{V}_0^l \setminus \{j\}$	(a)
	$y_j \cdot \tau_j < 0$	$\lambda^{l+1} = \lambda^l - \varepsilon$	$E_V^{l+1} = E_V^l$	$\mathcal{V}_-^{l+1} = \mathcal{V}_-^l \cup \{j\}$ $\mathcal{V}_0^{l+1} = \mathcal{V}_0^l \setminus \{j\}$	(b)
$j \in \mathcal{V}_-$ $f^l(\mathbf{x}_j)y_j < 0$	$y_j \cdot \tau_j > 0$	λ^{l+1}, $f^{l+1}(\mathbf{x}_j) = 0$ $\lambda^{l+2} = \lambda^{l+1} - \varepsilon$	$E_V^{l+1} = E_V^l$ $E_V^{l+2} = E_V^{l+1} - 1$	$\mathcal{V}_0^{l+1} = \mathcal{V}_0^l \cup \{j\}$ $\mathcal{V}_-^{l+1} = \mathcal{V}_-^l \setminus \{j\}$ $\mathcal{V}_+^{l+2} = \mathcal{V}_+^{l+1} \cup \{j\}$ $\mathcal{V}_0^{l+2} = \mathcal{V}_0^{l+1} \setminus \{j\}$	(c)
	$y_j \cdot \tau_j < 0$	No event	$-$	$-$	(d)
$j \in \mathcal{V}_+$ $f^l(\mathbf{x}_j)y_j > 0$	$y_j \cdot \tau_j > 0$	No event	$-$	$-$	(e)
	$y_j \cdot \tau_j < 0$	λ^{l+1}, $f^{l+1}(\mathbf{x}_j) = 0$ $\lambda^{l+2} = \lambda^{l+1} - \varepsilon$	$E_V^{l+1} = E_V^l + 1$ $E_V^{l+2} = E_V^{l+1}$	$\mathcal{V}_0^{l+1} = \mathcal{V}_0^l \cup \{j\}$ $\mathcal{V}_+^{l+1} = \mathcal{V}_+^l \setminus \{j\}$ $\mathcal{V}_-^{l+2} = \mathcal{V}_-^{l+1} \cup \{j\}$ $\mathcal{V}_0^{l+2} = \mathcal{V}_0^{l+1} \setminus \{j\}$	(f)

Table 2.1: Validation set event detection. The first column denotes the set a point $\mathbf{x}_j$ belongs to at breakpoint λ^l. The second column divides the possible cases depending on the class a point belongs to and the form of the evaluation function. The third column enumerates the values of λ for the possible events for each particular case, followed by the corresponding validation error and sets update in the last two columns. The value ε denotes an infinitesimal value that will cause the point to move again into another set.

hereupon. As already mentioned, when a validation point reaches the decision function at λ^{l+1}, $f^{l+1}(\mathbf{x}_j) = 0$, by continuity, for an infinitesimal decrease of size $\varepsilon > 0$, on λ^{l+1}, a validation event will occur, for $\lambda^{l+2} = \lambda^{l+1} - \varepsilon$, leading the point on the other side of the decision function. This table was constructed by observing the form of the evaluation function for each point in the validation set. These functions are strictly increasing or decreasing according to λ. Observing Figures 2.3(e) and 2.3(d), it can be seen that some events cannot occur if λ decreases.

To better understand Table 2.1, the derivation of a case for a positive sample in the validation set, $\mathbf{x}_j$ with $y_j > 0$, will be developed. Let us take as example the case $j \in \mathcal{V}_-$ for λ^l, $\lambda^l \in [\lambda^{t+1}, \lambda^t]$, then $f^l(\mathbf{x}_j) < 0$ (a wrongly classified sample). In the case $\tau_j^t > 0$ (leading to the case $y_j \cdot \tau_j^t > 0$), the evolution of the decision function with respect to λ is an hyperbola increasing as λ decreases (see Figure 2.3(c)). If no validation event has been detected in interval $[\lambda^{t+1}, \lambda^t]$, the last validation event will coincide with an event in the regularization path, so that $\lambda^l = \lambda^t$. As point $\mathbf{x}_j$ is bad classified, we are looking for the event $f(\mathbf{x}_j) > 0$. To achieve this state, event $f^{l+1}(\mathbf{x}_j) = 0$ has to occur first. The next validation breakpoint, λ^{l+1}, will move point $\mathbf{x}_j$ from set $\mathcal{V}_-$ to set $\mathcal{V}_0$. At this stage, the validation sample will still be incorrectly classified, so that the validation error remains constant: $E_V^{l+1} = E_V^l$. It is important to notice that the evaluation function at $\mathbf{x}_j$ will continue to increase as λ decreases, so that if the next regularization breakpoint $\lambda^{t+1} < \lambda^{l+1}$, point $\mathbf{x}_j$ will instantaneously move from set $\mathcal{V}_0$ to set $\mathcal{V}_+$ for any infinitesimal reduction of λ^{l+1}, which can be numerically denoted as $\lambda^{l+2} = \lambda^{l+1} - \varepsilon > \lambda^{t+1}$, with any small $\varepsilon > 0$. Finally, this last event will indeed reduce the validation error so that $E_V^{l+2} = E_V^{l+1} - 1$.

On the other hand, if $\tau_j^t < 0$ (leading to the case $y_j \cdot \tau_j^t < 0$, see Figure 2.3(d)), no event can occur under the same previous conditions because $j \in \mathcal{V}_-$ implies that $f^t(\mathbf{x}_j) < 0$ and for $\tau_j^t < 0$ the evaluation of the decision function will continue to decrease (see Figure 2.3(e)). Therefore, it will never occur $f^{l+1}(\mathbf{x}_j) = 0$ or $f^{l+1}(\mathbf{x}_j) > 0$ for any $\lambda < \lambda^l$.

The rest of the cases in Table 2.1 can be also easily deduced depending on the shape of the decision function.

It is important to notice that the ***Validation path*** can be easily extended in a cross validation or bootstrap frameworks to assure a robust parameter choice. A Validation path can be built for each of the $\{training - validation\}$ partition sets and an approximation of the generalization error curve can be obtained by taking the average of the k-fold cross or bootstrap validation given paths.

Algorithm 2 describes the steps to follow in order to obtain the complete validation error curve.

Algorithm 2 Pseudo-code for the SVM validation error curve calculation

```
Input: training set $\{(\mathbf{x}_i, y_i)\},\ i = [\![n]\!]$,
  validation set $\{(\mathbf{x}_j, y_j)\},\ j = [\![n_V]\!]$.
Output: Complete validation error curve set $\{(E_V^l,\ \lambda^l)\}$
  Use Algorithm 1 to obtain the complete solution set $\{(\boldsymbol{\alpha}_\lambda^t,\ \lambda^t)\}$.
  Initialize the validation error for parameter $\lambda^0$ as $E_V^0$.
  Set $l = 1$ and $t = 1$.
  repeat
    At interval $[\lambda^{t+1}, \lambda^t]$, calculate the corresponding constants $\tau_\lambda^t$ and $\upsilon_\lambda^t$.
    repeat
      Use Table 2.1 to detect the next validation event, $\lambda^{l+1}$, in $[\lambda^{t+1}, \lambda^l]$. Add $\lambda^{l+1}$ to the
      validation path and update its corresponding validation error value and validation sets.
      Set $l = l + 1$.
    until no more events are found.
    Set $t = t + 1$.
  until the whole regularization path is covered.
```

2.3.2 Model Selection with the Validation Path

During the path calculation phase, a breakpoint λ^t is given when an event occurs. Each training event will modify the form of the function, and therefore, the validation error can increase or decrease between two consecutive decision functions. The validation error will substantially change if there is a large number of validation events between two solutions given by the regularization path.

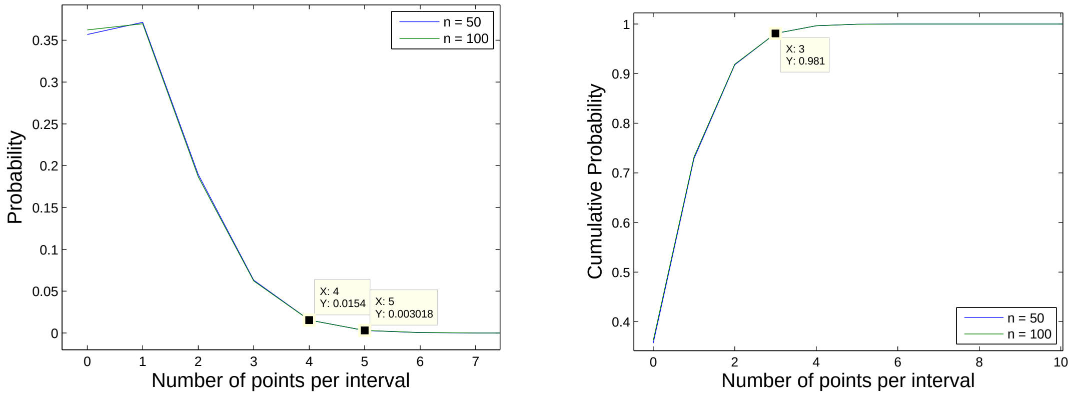
If the two sets were drawn independently from the same distribution, with high probability, dense and sparse regions all over the space should appear similar in both sets. Additionally, two optimal solutions from consecutive events in the regularization path will determine a subspace in the sample space where classification labels have not changed for the training points.

Each modification in the linear relationship of the decision function, that is, in the coefficients, implies an event in the training set. This suggests that an area with many events indicates a dense area of the samples distribution. Vice versa, a large area without training events will occur if the decision frontier is traversing a sparse area.

We can bound the space so that the whole training set is included in it, assigning zero probability measure to the rest of the space. Then, the space will be divided according to the given decision functions by the regularization path, if we assume that the areas are disjoint. This can be seen as if the decision frontiers were moving in a parallel manner encountering training points one after another. In order to approximate the distribution, it will be supposed that we have equal probability to encounter a point in each of these defined areas.

A rough approximation of the probability distribution can be done by giving an equal probability of finding a point in each of the defined areas by the sequence of decision function at each event.

If both sets (the training and validation one) are equally distributed, with high probability, we will find a maximum of three validation events between two regularization path breakpoints. So that the approximation of the validation error done at the breakpoints of the regularization path will have, with high probability, a maximum distance of three to the minimum possible validation error.
This can be better explained in an unidimensional space, as the bounded space is an interval and it will be divided by each of the events in sub-intervals. Suppose that $n = n_V$, then the admissible interval will be divided in $n+1$ subintervals. If we consider a particular subinterval, we can calculate the probability for each of the validations points to lie in this particular sub-interval if they are sampled one by one. That will give us the probability of encountering one, two, etc. validation points in this particular sub-interval. This turns out to be a Bernoulli distribution which density and cumulative distribution look as in Figure 2.4.



(a) Probability

(b) Cumulative Probability

Figure 2.4: Distribution of the number of validation points encountered between two decision frontiers given by two consecutive parameters in the regularization path. The figure illustrates the probability of having 0, 1, 2, etc. points out of n in a particular interval out of $n+1$ intervals. It is assumed that the probability of lying in any interval is equal. It was observed that for any n, the maximum probability is attained at one.

In Figure 2.4 it can be seen that the probability of encountering more than three points in the same interval is very low. This analysis was made considering that there is the same number of samples in the training and in the validation sets. So that, with high probability, the validation error at each of the regularization path breakpoints will not be larger than three as the probability of having more than three points in an interval is 0.98.
A deeper analysis of the maximum number of expected events can be found in [Ewen 06], where the probability for the maximum number of events in any interval is estimated.
The results coming out from the previous statements are summarized in the next theorem:

Conjecture 2.7 (Model choice using regularization path breakpoints[1])**.** *When choosing a model under the training-validation set framework, if the training and the validation sets possess the same distribution, choosing the best model given by the regularization path breakpoints will be, with high probability, the best possible chosen model among all models.*

The previous proposition states that if the validation error is measured over all possible models, which are infinite because λ is continuous, the best model will lie in average, on one of the breakpoints given by the regularization path obtained with the training set, therefore, statistically, it is not necessary to obtain the complete ***validation curve*** to be able to efficiently select a model.

[1] Special thanks to Gregory Mallet for his help in the formalization of this result.

After this proposition, model selection under the training-validation sets framework is reduced to finding all breakpoints in the regularization path and measuring in these points the validation error. This implies that the validation error can be in practice obtained for all λ, leading to an efficient model selection because the exact value at each point is exactly calculated, and the best possible model under a validation set criterion can be explicitly found.

2.3.3 Experiments

Proposition 2.7 was tested on several datasets from the UCI site[2] and in a toy dataset[3]. This last one is an example generated with several Gaussian mixtures. The datasets where normalized according to their standard deviation and mean.

The characteristics corresponding to the used dataset are summarized in Table 2.2, the dimension of each sample, the total number of samples and the number of existing classes.

Dataset	No. features	No. of samples	No. of classes
Mixture	2	200	2
Cancer	30	569	2
Diabetes	8	768	2
Heart	13	270	2
Schnitzel	3	280	2
Usps (1 vs. 7)	256	1858	2
Usps (2 vs. 5)	256	1657	2
Usps (4 vs. 9)	256	1200	2
Spam	57	4126	2

Table 2.2: Datasets characteristics.

The Gaussian kernel was used in all the experiments to generate the decision function. The used kernel bandwidth is the one proposed by a method that approximates the validation error using a gradient descent method [Chap 02] to later search the regularization parameter with the regularization path. For each run, the complete dataset was randomly partitioned in four parts, leaving three quarters of the data as learning set and the last quarter as test set. At each stage, a partition for the learning set was done to obtain a training set with about the half of the learning data and a validation set with the rest, so that both have an equal distribution according to the algorithm described in [Aupe 08]. The complete validation error curve is calculated in a bootstrap framework [Hast 01] with five bootstrap folds to smooth the validation error curve.

For the model selection task, the validation error was measured in both the regularization path breakpoints and in all the given validation path breakpoints. For both paths, the λ parameter incurring the minimum validation error was kept. The two kept parameters were used to derive two models that were trained using a data set combining the training and validation sets, that is, with the learning set. Finally, the test error is measured on the unseen test set. This process is repeated five times by randomly generating learning-test sets and the mean of the test error is taken to make an estimation of the generalization error.

The aim of the experiments is to show that the regularization path has enough breakpoints to be analyzed and also that it can help the gradient algorithm [Chap 02] to get out of local minimums while taking advantage of the proposed approximation given by the gradient method. Figure 2.5 shows the complete average validation error curve for the cancer dataset. It can be seen that with the regularization path, the local minimum found by the gradient method can be overcome.

After applying the explained method, the obtained results in terms of test error of the best selected model are reported in Table 2.3. Three options were considered: the regularization curve, the validation curve and the gradient method. The last column contains the average of the

[2] http://archive.ics.uci.edu/ml/

[3] taken from `http://www-stat.stanford.edu/~tibs/ElemStatLearn/`

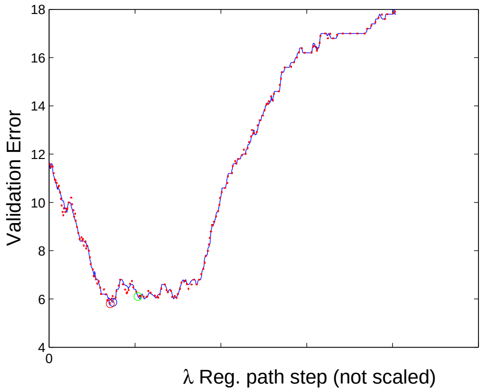


Figure 2.5: Usage of gradient method with the regularization and validation path in the cancer dataset. The green circle is the one founded by the gradient method, the kernel parameter is kept while the regularization parameter is searched with the validation path. The resulting λ using only the regularization path is drawn in blue and the one using the complete validation curve in dotted red. The validation error against the value of λ is plotted.

kernel parameter proposed by the gradient method. It can be seen that the performances of the algorithms are closely similar.

Dataset	Test error reg. Path	Test error Val. Curve	Test error Grad. Method	kernel-σ
Mixture	0.2	0.2	0.2	0.466
Cancer	0.021	0.021	0.021	5.513
Diabetes	0.25	0.25	0.242	3.183
Heart	0.179	0.179	0.173	3.854
Schnitzel	0.037	0.037	0.043	0.677
Usps (1 vs.7)	0.008	0.008	0.007	12.200
Usps (2 vs.5)	0.003	0.003	0.002	10.897
Usps (4 vs.9)	0.009	0.009	0.01	14.510
Spam	0.064	0.064	0.057	5.757

Table 2.3: Mean test error with homogeneous partition. The errors are the ratio of the number of misclassified points over the size of the test set. The experiment was repeated ten times, the average test error is shown.

Table 2.4 compares the difference between the obtained regularization parameter according to the minimum validation error in the regularization path against the global minimum in the validation curve and the regularization parameter given by the gradient method. The calculation time of both methods is shown. The calculation of all possible models for all regularization parameters with fixed kernel parameter is larger than the needed time using the gradient method and it increases depending in the dataset complexity.

Finally, the last column in Table 2.4 measures the average number of validation events between two regularization path events. The remarkable issue in this result is the fact that in average there is one validation event every two regularization path events, confirming the fact that the given sampling by the regularization path is enough to choose a parameter as stated in Proposition 2.7.

In the case where there exists a local minimum in the validation curve, analyzing the complete curve can help to find the global minimum, still the usefulness of doing so is not clear as the test error was not systematically improved by doing so. However, having in hand a complete and entire validation path, even at a slightly expensive price, could help the user in the choice of the optimal

Dataset	λ diff. train-val	λ diff. train-grad	Reg. path time	Gradient time	No. Events
Mixture	-0.031	-0.179	0.51	0.18	0.40 (0.54)
Cancer	-0.020	-0.238	2.31	0.91	0.39 (0.39)
Diabetes	-0.009	0.145	7.26	1.57	0.39 (0.78)
Heart	-0.042	-3.605	0.56	0.24	0.39 (0.48)
Schnitzel	0.064	-0.765	0.76	0.28	0.39 (0.45)
Usps (1 vs.7)	-0.005	-1.283	50.08	12.48	0.45 (0.35)
Usps (2 vs.5)	0.745	-3.017	101.48	10.15	0.38 (0.87)
Usps (4 vs.9)	0.012	0.143	38.07	9.30	0.42 (0.39)
Spam	-0.002	-1.233	5710.48	108.74	0.42 (0.89)

Table 2.4: Difference between the proposed λ parameter per method, calculation time (in seconds) for each method and average number of events per regularization path breakpoint with its standard deviation in parenthesis.

solution.

Interestingly, results in Table 2.3 do not seem to be very affected if the sampling of the training-validation sets is not homogeneous. Test were done with training-validation partitions with less guarantee to have an homogeneous distribution, but after the bootstrap procedure, the non-homogeneity does not seem to substantially change the test error or the number of encountered validation events.

2.3.4 Conclusions

The Regularization path is a tool to obtain all optimal solutions for the SVM framework given a fixed kernel parameter. An algorithm to efficiently obtain the complete validation curve was developed so that along the regularization path, track can be kept of the corresponding validation error.

It was argued that given an homogeneous partition of the training and validation sets, it will be enough to choose one of the regularization parameters proposed by the regularization path to get, with high probability, the model with the lowest validation error. Experiments showed that there is, in average, one validation event between two regularization path events, that could eventually improve the validation error. This result suggests that the calculation of the complete validation error curve could be unnecessary as the breakpoints of the training regularization path contain almost enough informations for the evaluation of the generalization error.

The time to calculate the complete set of solutions with a fixed kernel parameter is about five to ten times the calculation of the optimal model with the gradient method. Still, it is not clear if after the solution given by the gradient method, fixing the kernel parameter and having at hand the regularization parameter that outcomes the minimum of the validation error will help to improve the test error. The exploration of the combination of a regularization path for kernel parameter determination simultaneously with the one of regularization parameter could provide insights and valuable answers to this question.

2.4. Ranking SVM

Supervised learning problems under the ranking framework is introduced in this section. As stated in Section 1.3.2, the flexibility of the kernel methods framework is used, so that the rankSVM algorithm is a regularization problem, controlling the generalization error by balancing the training error and the complexity of the decision function f by means of a regularization parameter λ . The complexity is measured as the norm (often the L_1 or L_2 norm in SVM setting) of f. The originally proposed rankSVM uses a L_2 norm for this purpose. We recall this optimization problem:

Problem 2.8 (SVM Ranking Dual Problem)**.**

$$\begin{array}{ll} \underset{f\in\mathcal{H},\boldsymbol{\xi}\in\mathbb{R}^m}{\min} & \boldsymbol{\xi}^\top \mathbb{1} + \frac{\lambda}{2}\left\|f\right\|^2_{\mathcal{H}} \\ \text{subject to} & f(\mathbf{x}_{u_i}) - f(\mathbf{x}_{v_i}) \geq 1 - \xi_i \quad \forall i = [\![m]\!] \\ & \xi_i \geq 0 \qquad\qquad\qquad\qquad\quad \forall i = [\![m]\!]. \end{array}$$

With its dual formulation:

Problem 2.9 (SVM Ranking Dual Problem)**.**

$$\begin{array}{ll} \underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\operatorname{argmax}} & \boldsymbol{\alpha}^\top\mathbb{1} - \frac{1}{2\lambda}\boldsymbol{\alpha}^\top PKP^\top\boldsymbol{\alpha} \\ s.t. & \mathbf{0} \leq \boldsymbol{\alpha} \leq \mathbb{1}. \end{array}$$

where P is a matrix encoding the ordering constraints (see Equations (1.34)). In this section, we concentrate on speeding up and automating the choice of λ by means of a regularization path [Zapi 08b] for the rankSVM and building on top of the results obtained in the last section, leading to an efficient problem resolution.

Additional issues on this framework include the fact that, if there are n samples, the number of constraints is of order $O(n^2)$ and the final model is a linear combination of all these. In general, ranking problems are large-scale problems and constrained in time.

2.4.1 Ranking vs. Classification

It is natural to expect that the classification problem and the ranking one are related under the SVM framework. A classification problem can be turned into a ranking one by artificially assigning preferences according to the class each sample belongs to. In this section, a link is made between the rankSVM and the SVC frameworks. The aim is to prove that the feasible set of both problems is the same under the same conditions and that the introduction of artificial preferences is useful in both cases.

Theorem 2.10 (Equivalence of the feasible sets for the ranking and classification problems (constraint to functions with no training error))**.** *If a ranking problem is composed of only two ranks (w.l.o.g. $y_i \in \{\pm 1\}$, $\mathscr{C}_1$ the positive class ($y_i = 1$) and $\mathscr{C}_2$ the negative class ($y_i = -1$)) and we define $\mathbf{F}_C = \{f : y_i(f(\mathbf{x}_i) + b) \geq 1\ \forall i = [\![n]\!]$, for a particular $b \in \mathbb{R}$ for a fixed $f\}$, the feasible set (of functions with no training error) of the classification problem, and $\mathbf{F}_R = \{f : f(\mathbf{x}_i) - f(\mathbf{x}_j) \geq 2, \forall i \in \mathscr{C}_1, j \in \mathscr{C}_2\}$ a slightly modified (but equivalent) feasible set (of functions with no training error) of the ranking problem, then, sets $\mathbf{F}_C$ and $\mathbf{F}_R$ are the same set.*

Proof. The proof is done in two parts:

- To prove: $F_C \subset F_R$.

  Let $f \in F_C$, then $y_i(f(\mathbf{x}_i) + b) \geq 1$ for all $i = [\![n]\!]$, so we have in particular:

  $$\begin{aligned} f(\mathbf{x}_i) + b \geq 1, &\qquad \forall \mathbf{x}_i \in \mathscr{C}_1 \\ -(f(\mathbf{x}_j) + b) \geq 1, &\qquad \forall \mathbf{x}_j \in \mathscr{C}_2, \end{aligned}$$

  for a particular $b \in \mathbb{R}$. Adding these inequalities we get:

  $$\begin{aligned} (f(\mathbf{x}_i) + b) - (f(\mathbf{x}_j) + b) \geq 1 + 1, &\qquad \forall \mathbf{x}_i \in \mathscr{C}_1, \mathbf{x}_j \in \mathscr{C}_2 \\ f(\mathbf{x}_i) - f(\mathbf{x}_j) \geq 2, &\qquad \forall \mathbf{x}_i \in \mathscr{C}_1, \mathbf{x}_j \in \mathscr{C}_2 \end{aligned}$$

  thus, f belongs to F_R

- To prove: $F_R \subset F_C$.

  If $f \in F_R$ then by hypothesis

$$f(\mathbf{x}_i) - f(\mathbf{x}_j) \geq 2 \text{ for all } \mathbf{x}_i \in \mathscr{C}_1, \mathbf{x}_j \in \mathscr{C}_2. \tag{2.33}$$

  Let

$$x_{min} = \min_{x_i, i \in \mathscr{C}_1} \{f(\mathbf{x}_i)\} \quad \text{and} \quad x_{max} = \max_{x_j \in \mathscr{C}_2} \{f(\mathbf{x}_j)\}$$

  and we will define

$$b = -\frac{f(x_{min}) + f(x_{max})}{2}$$

  then using (2.33) we get $f(x_{min}) - f(x_{max}) \geq 2$, so for $i \in \mathscr{C}_1$

$$\begin{aligned} f(\mathbf{x}_i) \geq f(x_{min}) &= f(x_{min}) - \frac{f(x_{min}) + f(x_{max})}{2} + \frac{f(x_{min}) + f(x_{max})}{2} \\ &= \frac{f(x_{min}) - f(x_{max})}{2} + \frac{f(x_{min}) + f(x_{max})}{2} \geq \frac{2}{2} - b \end{aligned}$$

  and therefore $f(\mathbf{x}_i) + b \geq 1$

  analogously, noting that $f(x_{max}) - f(x_{min}) \leq -2$ for $j \in \mathscr{C}_2$

$$\begin{aligned} f(\mathbf{x}_j) \leq f(x_{max}) &= f(x_{max}) - \frac{f(x_{min}) + f(x_{max})}{2} + \frac{f(x_{min}) + f(x_{max})}{2} \\ &= \frac{f(x_{max}) - f(x_{min})}{2} + \frac{f(x_{min}) + f(x_{max})}{2} \leq -\frac{2}{2} - b \end{aligned}$$

  and therefore $-(f(\mathbf{x}_j) + b) \geq 1$.

we can then conclude that both sets are the same. □

2.4.2 RankSVM Singularity and Graph Reduction

The rankSVM optimization problem induces a directed graph for each query (see Figure 1.11). Each edge corresponds to a relationship of relevance between samples that has to be satisfied, that is, each edge corresponds to a constraint. These constraints include as well all transitive relationships that could in fact be induced by other ones. It was experimentally observed that this redundancy causes the Hessian matrix $PKP^\top$ in Problem 2.9 to be singular with many zero eigenvalues. The singularity can be overcome by making a correction of the Hessian matrix, nevertheless, the number of variables remains large.

The fact that there exists several redundant constraints was already noticed in other RankSVM publication [Herb 00] and was used to derive uniform convergence bounds. In order to derive this bound in an analogous manner as for the classification case, samples must be drawn in a i.i.d. manner, which is not the case given the transitivity relations.

Here, we take advantage of the transitivity relations to reduce in a robust manner the number of constraints in the rankSVM problem by imposing a higher rank to one of the samples among each rank level.

This new graph is built so that for each rank level a particular sample is designed as the maximum among his ranking, so that edges from the chosen sample will be added to the other samples in the same level (intra-rank constraints), indicating that the chosen sample has more relevance than the rest at the same rank. For the immediate upper level, all samples in it will be only compared to the artificially designed maximum of the lower rank (inter-rank constraints) and repeating this for each rank level. We recall the ranking example seen in Section 1.3.2, with the following graph: The corresponding reduced graph for this problem would look as in Figure 2.7:

The procedure to create the new graph is described in Algorithm 3 page 73.

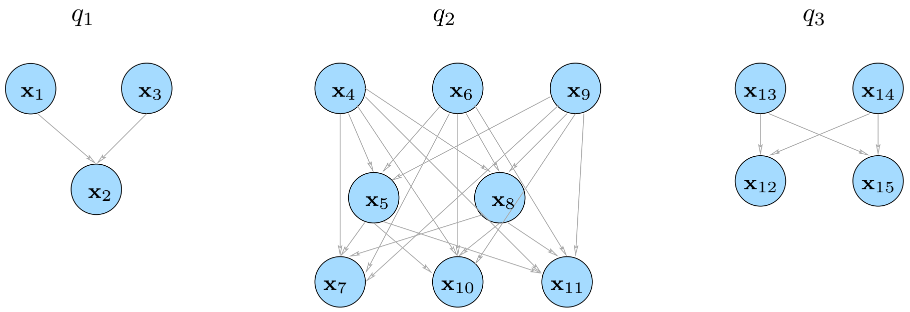


Figure 2.6: Induced graph for the ranking problem stated in Table 1.6.

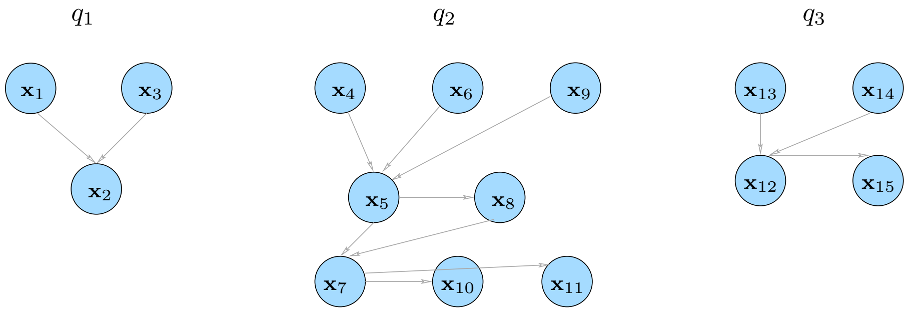


Figure 2.7: Reduced graph for the ranking problem stated in Table 1.6.

Algorithm 3 Reduced graph construction for the RankSVM problem.

```
Input: Training set $\{(\mathbf{x}_i, y_i, q_i)\},\ i = [\![n]\!]$ (sample, rank, query).
Output: adjacency matrix $P' \in \mathbb{R}^{m' \times n}$ corresponding to the reduced directed graph.
  Set $n_r$ = number of ranks, $n_q$ = number of queries, and $j = 1$.
  for $Q = 1$ to $n_q$ do
    for $R = n_r - 1$ to 1 do
      Choose at random a sample $k$, $\mathbf{x}_k$, with rank $y_k = R$ and $q_k = Q$.
      Let $V = \{i | y_i = R, q_i = Q, i \neq k\}$ samples with rank $R$ for the query $Q$, $n_s = |V|$,

      Intra-rank constraints generation
      for $l = 1$ to $n_s$ do
        Set $P'_{jk} = 1,\ P'_{jV_l} = -1,\ P'_{jm} = 0, \forall m \notin \{k \cup V_l\}$ with $V_l$ the $l^{th}$ element of set $V$.
        Set $j = j + 1$.
      end for

      Inter-rank constraints generation
      Let $V = \{i | y_i = R + 1, q_i = Q\}$ samples with rank $(R + 1)$ for the query $Q$, $n_s = |V|$,
      for $l = 1$ to $n_s$ do
        Set $P'_{jk} = -1,\ P'_{jV_l} = 1,\ P'_{jm} = 0, \forall m \notin \{k \cup V_l\}$.
        Set $j = j + 1$.
      end for
    end for
  end for
```

By transitivity reasons, Figure 2.7 includes the constraints of Figure 2.6. A solution to the ranking problem based upon Graph 2.7 and satisfying all constraints will be a solution of the same problem as Graph 1.11.

For a solution searching to violate as few as possible constraints, the best strategy would be to keep all samples having equal relevance within a particular band-value in the decision function evaluation. If this is hold, and the evaluation value increases as the ranking does, all inter-rank constraints will be satisfied, the unsatisfied constraints will be located between samples of the same rank (intra-rank constraints). Intra-rank violations will be preferred to inter-rank violations because the first ones are fewer. Notice that if one intends to enforce this fact, an asymmetric ranking problem can be formulated where different regularization parameters could be applied with regard to the type of ordering constraints (intra or inter-rank constraints).

The advantage of this new formulation is that the number of constraints is significantly smaller than in the original RankSVM algorithm. The first one can be of order $O(n^2)$, while the second one is of order $O(n)$. To see this, it has to be noticed that with the new formulation, for each rank level, if there are n_s samples at that level, there will be a maximum of $(2 \times n_s)$ edges: $n_s - 1$ edges to design the maximum of that level and n_s edges going to the lower level. This construction will lead to a smaller problem and, therefore, a faster training time with a consistent problem statement.

Theorem 2.11 (Relation between ordering constraint matrices P and P')**.** *Let matrix P be the adjacency matrix containing all constraints as proposed in the original rankSVM problem. Let P' be the resulting matrix built as described in Algorithm 3, then, matrix P' is the result of row operations on matrix P and if n is the number of rows (equal in both), the number of columns in P' will be less than $2n$.*

Proof. Let matrix P be of size $m \times n$ and matrix P' of size $m' \times n$ with n the sample size, m and m' the number of ordering constraints for respectively the complete and the reduced graph. Each edge in matrix P' is either an edge going from levels $(k+1)$ or k to level k.

i) a row in P' representing an edge going from level $(k+1)$ to level k will already exists in matrix P, say row i, therefore if a column vector $\mathbf{o}$ is defined with zeros everywhere except in the i-th row, multiplication $\mathbf{o}^\top P$ will output the desired row of matrix P'.

ii) a row in P' representing an edge going from $\mathbf{x}_{u_i}$ to $\mathbf{x}_{v_i}$ in the same level can be easily constructed by taking any sample $\mathbf{x}_i$ in an upper level and finding the corresponding rows e_u and e_v representing the edges from $\mathbf{x}_i$ to $\mathbf{x}_{u_i}$ or $\mathbf{x}_{v_i}$, respectively. Finally, let $\mathbf{o}^\top$ be the column vector with minus one in the e_u-th row and one in the e_v-th row. $\mathbf{o}^\top P$ will output the desired row of matrix P'.

For the size of P', it only has to be noticed that each of the n nodes will have at most two adjacent edges: one coming from the designed maximum of its own rank and the other going to the maximum of the lower rank. □

This new proposed graph construction will not only reduce the size of the optimization problem but will also help an important issue which is the fact that for a particular λ, there exists many optimal solutions as there are redundant constraints.

2.4.3 Regularization path for RankSVM

The number of ranking constraints m is of the order of the square of the number of labeled pairs $\mathbf{x} = (d, q)$ (with the full graph), and therefore, grid search on the regularization parameter will be very time consuming. The reduced graph proposes a problem of order n, the number of labeled pairs, but solving many QP is still costly. This section investigates the application of regularization path idea to help the tuning of λ.

Similar to SVM classification, it turns out that the derivation of the ranking function f involves a piecewise variation of its parameters with respect to λ and hence forms a **regularization path**. Indeed, in the RankSVM algorithm, the loss function $\mathcal{L}(f)$ is the hinge loss (which is a L_1 type-function) and the regularizer $\Omega(f)$ is chosen as a quadratic function. Following the works of [Ross 07b] and [Hast 04], the regularization path for the rankSVM framework was derived [Zapi 08b].

For a given λ, let $\boldsymbol{\alpha}$ and $f(\mathbf{x})$ be the optimal solution and the decision function for Problem 2.9, respectively. Then, the following sets derived from the KKT optimality conditions can be formed:

- $\mathcal{I}_\alpha \;=\; \{i \in [\![m]\!] \mid f(\mathbf{x}_{u_i}) - f(\mathbf{x}_{v_i}) = 1, 0 \leq \alpha_i \leq 1\}$,

- $\mathcal{I}_0 \;=\; \{i \in [\![m]\!] \mid f(\mathbf{x}_{u_i}) - f(\mathbf{x}_{v_i}) > 1, \alpha_i = 0\}$,
- $\mathcal{I}_1 \;=\; \{(i \in [\![m]\!] \mid f(\mathbf{x}_{u_i}) - f(\mathbf{x}_{v_i}) < 1\alpha_i = 1\}$.

$\mathcal{I}_\alpha$ is the set of pairs (or ordering constraints) that lie *on the margin* whereas $\mathcal{I}_0$ and $\mathcal{I}_1$ represent respectively the sets of satisfied and non-satisfied ordering constraints.

Similarly, we will denote by $\boldsymbol{\alpha}^t$ and $f^t(\mathbf{x})$ the optimal solution of the dual Problem 2.9 with regularization parameter λ^t. We assume the sets $(\mathcal{I}^t_\alpha, \mathcal{I}^t_1, \mathcal{I}^t_0)$ induced by the solution of the optimization problem with λ^t remain unchanged for all $\lambda \in (\lambda^{t+1}, \lambda^t)$, i.e., $(\mathcal{I}^t_\alpha, \mathcal{I}^t_1, \mathcal{I}^t_0) = (\mathcal{I}_\alpha, \mathcal{I}_1, \mathcal{I}_0)$. Hence, $\alpha_i, i \in \mathcal{I}_0 \cup \mathcal{I}_1$ remains with value zero or one respectively, while $\alpha_i \in \mathcal{I}_\alpha$ has a linear relation on λ.

This can be seen by writing $f(\mathbf{x})$ as follows:

$$\begin{aligned} f(\mathbf{x}) &= \left[f(\mathbf{x}) - \frac{\lambda^t}{\lambda} f^t(\mathbf{x})\right] + \frac{\lambda^t}{\lambda} f^t(\mathbf{x}) \\ &= \frac{1}{\lambda}\Big[(\boldsymbol{\alpha} - \boldsymbol{\alpha}^t)^\top P\mathbf{k}(\mathbf{x}) + \lambda^t f^t(\mathbf{x})\Big] \quad \text{using Equation (1.35)} \\ f(\mathbf{x}) &= \frac{1}{\lambda}\Big[(\boldsymbol{\alpha}_{\mathcal{I}_\alpha} - \boldsymbol{\alpha}^t_{\mathcal{I}_\alpha})^\top P_{\mathcal{I}_\alpha}\mathbf{k}(\mathbf{x}) + \lambda^t f^t(\mathbf{x})\Big] \end{aligned} \tag{2.34}$$

where the last line is true as $\alpha_i - \alpha^t_i = 0$ for all $i \notin \mathcal{I}_\alpha$. $P_{\mathcal{I}_\alpha}$ is the submatrix of P containing the rows corresponding to $\mathcal{I}_\alpha$ and all columns. For all $i \in \mathcal{I}_\alpha$ we have that $1 = f(\mathbf{x}_{u_i}) - f(\mathbf{x}_{v_i}) = f^t(\mathbf{x}_{u_i}) - f^t(\mathbf{x}_{v_i})$, leading to

$$1 = \frac{1}{\lambda}\Big[\big(\boldsymbol{\alpha}_{\mathcal{I}_\alpha} - \boldsymbol{\alpha}^t_{\mathcal{I}_\alpha}\big)^\top P_{\mathcal{I}_\alpha}\big(\mathbf{k}(\mathbf{x}_{u_i}) - \mathbf{k}(\mathbf{x}_{v_i})\big) + \lambda^t\Big].$$

Therefore

$$\lambda - \lambda^t = \big(\boldsymbol{\alpha}_{\mathcal{I}_\alpha} - \boldsymbol{\alpha}^t_{\mathcal{I}_\alpha}\big)^\top P_{\mathcal{I}_\alpha}\big(\mathbf{k}(\mathbf{x}_{u_i}) - \mathbf{k}(\mathbf{x}_{v_i})\big)\ . \tag{2.35}$$

This equation is valid for all pairs in $\mathcal{I}_\alpha$ for fixed sets $\mathcal{I}_\alpha, \mathcal{I}_0$, and $\mathcal{I}_1$. It can be simplified by transposing Equation (2.35) and using Equation (1.34) in it, getting:

$$(\lambda - \lambda^t)\mathbb{1}_{\mathcal{I}_\alpha} \;=\; P_{\mathcal{I}_\alpha} K P^\top_{\mathcal{I}_\alpha}\Big(\boldsymbol{\alpha}_{\mathcal{I}_\alpha} - \boldsymbol{\alpha}^t_{\mathcal{I}_\alpha}\Big) \tag{2.36}$$

If we define $\boldsymbol{\eta} = (P_{\mathcal{I}_\alpha} K P^\top_{\mathcal{I}_\alpha})^{-1}\mathbb{1}_{\mathcal{I}_\alpha}$, with $\mathbb{1}_{\mathcal{I}_\alpha}$ a vector of ones of size $|\mathcal{I}_\alpha|$, then it can finally be seen that $\alpha_i, i \in \mathcal{I}_\alpha$ changes piecewise linearly in λ as follows:

$$\alpha_i = \alpha^t_i - (\lambda^t - \lambda)\eta_i \qquad i \in \mathcal{I}_\alpha. \tag{2.37}$$

Therefore, for all λ, the optimal solution can easily be obtained using Equation (2.37) while the sets remain fixed, i.e., no event occurs. These events can be detected to update vector $\boldsymbol{\eta}$.

2.4.4 Initialization

An initial solution is necessary to define sets $\mathcal{I}_\alpha, \mathcal{I}_1$ and $\mathcal{I}_0$ and vector $\boldsymbol{\eta}$. Given λ, the quadratic Problem 2.9 can be solved but as mentioned before, it is computationally expensive. Instead, an initial solution can be easily obtained if λ is very large since $\boldsymbol{\beta} = \mathbf{0}$ minimizes the bidual problem expressed in Equation (1.39). This implies that $\xi_i = 1$ and because of the strict complementary and KKT conditions, the associated Lagrange parameter $\alpha_i = 1$. To have at least one element in $\mathcal{I}_\alpha$, we need to find a pair $(\mathbf{x}_{u_i}, \mathbf{x}_{v_i})$ such as $1 = f(\mathbf{x}_{u_i}) - f(\mathbf{x}_{v_i})$. Thus, the following equation

$$\lambda_{u_i,v_i} = \boldsymbol{\alpha}^\top P(\mathbf{k}(\mathbf{x}_{u_i}) - \mathbf{k}(\mathbf{x}_{v_i}))$$

has to be solved for all pairs and knowing that $\boldsymbol{\alpha} = \mathbb{1}$. Hence, initially all pairs will be in $\mathcal{I}_1$ and, as initial λ value, we take

$$\lambda^0 = \max\{\lambda_{u_i,v_i}\}.$$

The corresponding pair $(\mathbf{x}_{u_i}, \mathbf{x}_{v_i})$ is included as first element of $\mathcal{I}_\alpha$.

2.4.5 Event Detection

At step t the optimal solution $\boldsymbol{\alpha}^t$ defines a partition $\mathcal{I}_\alpha, \mathcal{I}_1, \mathcal{I}_0$. The linear variation of the parameters $\boldsymbol{\alpha}$ remains until a change in these sets happens. If an event occurs, then the linear equation has to be readjusted. Two types of events have to be determined: *a)* a pair in $\mathcal{I}_\alpha$ goes to $\mathcal{I}_1$ or $\mathcal{I}_0$ and *b)* a pair in $\mathcal{I}_1$ or $\mathcal{I}_0$ goes to $\mathcal{I}_\alpha$.

Pair in $\mathcal{I}_\alpha$ goes to $\mathcal{I}_1$ or $\mathcal{I}_0$

This event can be determined by analyzing at which value of λ the corresponding α_i turns zero or one. Equation (2.37) is used and the following systems are solved for λ_i:

$$1 = \alpha_i^t - (\lambda^t - \lambda_i)\eta_i \qquad i \in \mathcal{I}_\alpha \tag{2.38}$$

$$0 = \alpha_i^t - (\lambda^t - \lambda_i)\eta_i \qquad i \in \mathcal{I}_\alpha. \tag{2.39}$$

Using this last equation, the exact values for λ_i that produce an event on pairs in $\mathcal{I}_\alpha$ moving to $\mathcal{I}_0 \cup \mathcal{I}_1$ can be determined.

Pair in $\mathcal{I}_1$ or $\mathcal{I}_0$ goes to $\mathcal{I}_\alpha$

To detect this event, note that Equation (2.36) can also be written as follows:

$$\left(\boldsymbol{\alpha}_{\mathcal{I}_\alpha} - \boldsymbol{\alpha}^t_{\mathcal{I}_\alpha}\right) = (\lambda - \lambda^t)\left[\left(P_{\mathcal{I}_\alpha} K P_{\mathcal{I}_\alpha}^\top\right)^{-1} \mathbb{1}_{\mathcal{I}_\alpha}\right] = (\lambda - \lambda^t)\boldsymbol{\eta}. \tag{2.40}$$

Plugging Equation (2.40) in Equation (2.34), we can write $f(\mathbf{x})$ in a convenient manner and letting $h^t(\mathbf{x}) = \boldsymbol{\eta}^\top P_{\mathcal{I}_\alpha}\mathbf{k}(\mathbf{x})$, then

$$f(\mathbf{x}) = \frac{1}{\lambda}\left[\lambda^t f^t(\mathbf{x}) - \lambda^t h^t(\mathbf{x}) + \lambda h^t(\mathbf{x})\right] \tag{2.41}$$

An event on pair $(\mathbf{x}_{u_i}, \mathbf{x}_{v_i}) \in \mathcal{I}_0 \cup \mathcal{I}_1 \longrightarrow \mathcal{I}_\alpha$ means that $f(\mathbf{x}_{u_i}) - f(\mathbf{x}_{v_i}) = 1$ and can be detected by using Equation (2.41). The corresponding λ_i that generates this event is calculated as follows:

$$\lambda_i = \frac{\lambda^t\left[(f^t(\mathbf{x}_{u_i}) - f^t(\mathbf{x}_{v_i})) - (h^t(\mathbf{x}_{u_i}) - h^t(\mathbf{x}_{v_i}))\right]}{1 - (h^t(\mathbf{x}_{u_i}) - h^t(\mathbf{x}_{v_i}))} \tag{2.42}$$

λ^{t+1} will be the largest resulting $\lambda_i < \lambda^t$ from Equations (2.38), (2.39) and (2.42).

2.4.6 Remarks and comments

Here we briefly discuss some issues of the algorithm related to the piecewise variation, the numerical complexity and how to address the emptiness of the set $\mathcal{I}_\alpha$.

Implementation Details

It has to be noticed that the matrix multiplication $P_{\mathcal{I}_\alpha}.K_{\cdot,\cdot}$ can be reduced by detecting the points $A_\alpha = \{i | (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{I}_\alpha \text{ or } (\mathbf{x}_j, \mathbf{x}_i) \in \mathcal{I}_\alpha\}$ so that $P_{\mathcal{I}_\alpha}.K_{\cdot,\cdot} = P_{\mathcal{I}_\alpha A_\alpha} K_{A_\alpha,\cdot}$, which avoids quite a few multiplications by zero. An analogous reduction can be done when having $P_{\mathcal{I}_0}.K_{\cdot,\cdot}$ or $P_{\mathcal{I}_1}.K_{\cdot,\cdot}$. The path computation is summarized by the pseudo-code of Algorithm 4.

On the functional piecewise variation

Let function $g = \lambda f$ corresponds to the regularization parameter λ. In a similar manner, consider the function $g^t = \lambda^t f^t$ which corresponds to the solution for the value λ^t. From Equation (2.41), one derives easily the relation $g = g^t + (\lambda - \lambda^t)h^t$. Therefore, we recover the conditions for the piecewise linear variation stated in Theorem 2.2. This linear variation formally concerns the function g instead of f. However the parameters $\boldsymbol{\alpha}$ involved in f evolve linearly with λ.

Algorithm 4 Pseudo-code of the rankSVM regularization path computation

```
Input: The set of m preferences $E = \{(\mathbf{x}_{u_i}, \mathbf{x}_{v_i}) \mid i \in [\![m]\!]\}$
Output: All solutions $(f_\lambda,\ \lambda)$
  Set $t = 0$
  Compute the initial value $\lambda^0$ and estimation $\boldsymbol{\alpha}^0$.
  Deduce the corresponding sets $(\mathcal{I}_\alpha^0, \mathcal{I}_1^0, \mathcal{I}_0^0)$
  repeat
    Compute the update direction of the parameters $\boldsymbol{\eta}^t = (P_{\mathcal{I}_\alpha^t} K P_{\mathcal{I}_\alpha^t}^\top)^{-1} \mathbb{1}_{\mathcal{I}_\alpha^t}$
    Use $\boldsymbol{\eta}^t$ to detect all the necessary events and the corresponding regularization parameter values
    Select the appropriate boundary value $\lambda^{t+1}$, save the event type (pair in $\mathcal{I}_1 \cup \mathcal{I}_0$ goes to $\mathcal{I}_\alpha$ or pair in $\mathcal{I}_\alpha$ goes to $\mathcal{I}_1$ or $\mathcal{I}_0$) and the related pair(s) of constraint(s)
    Update the parameters according to Equation (2.37)
    Update the sets $(\mathcal{I}_\alpha^t, \mathcal{I}_1^t, \mathcal{I}_0^t)$ according to the retained event and the concerned pair(s)
    $t = t + 1$
  until $\lambda$ is small or another termination criteria is reached.
```

On the numerical complexity

The numerical complexity of the algorithm can be analyzed as follows. We assume the whole matrix $P\,K\,P^\top$ is available beforehand as it can be built and stored at the beginning of the algorithm and this computation requires $\mathcal{O}(mn^2)$ operations from the knowledge of the matrices P and K. At each iteration, solving the linear system in Equation (2.36) involves a cost of order $\mathcal{O}(|I_\alpha|^3)$. The calculation of all subsequent values λ^{t+1} (using Equation (2.38), (2.39) and (2.42)) has a numerical complexity of $\mathcal{O}(m|I_\alpha|)$ whereas the detection of the next event is of order $\mathcal{O}(m)$. According to Equation (2.37), the update of all α_i is $\mathcal{O}(m)$. We can note that the computational complexity is essentially related to the cardinality of $|I_\alpha|$. The cubic complexity of the linear system can be decreased to square complexity using a Sherman-Morrison rule to update the inverse of the matrix $P_{\mathcal{I}_\alpha}\,K\,P_{\mathcal{I}_\alpha}^\top$ or a Cholesky update procedure. The exact complexity of the algorithm is hard to predict since the total number of events needed for exploring entirely the regularization path is data-dependent and the mean size of $|\mathcal{I}_\alpha|$ is difficult to guess beforehand. However, the total complexity is few multiples of the cost for solving directly the dual Problem 2.9.

Applying this mechanism, the different solutions of the original rankSVM can be retrieved. Following the same arguments as [Hast 04], the theoretical numerical complexity of the path computation is a small multiple of the QP solving. However, in practice the computational cost can be slightly higher due to some numerical problems which requires sometimes to reinitialize the algorithm using the warm restart procedure [DeCo 00] of the SVM algorithms. Beyond these technical points, we want to emphasize that the singularity of the SVM hinge loss induces sparsity in ordering constraints but the number of parameters is the order of the number of the labeled pairs (d, q). To achieve sparsity in the parameters, we propose later in this chapter a L_1-norm penalization.

On the emptiness of $\mathcal{I}_\alpha$

It may happen during the algorithm that the set $\mathcal{I}_\alpha$ becomes empty. In such situation, a new initialization of the algorithm is needed. We apply the procedure developed in Subsection 2.4.4 except the fact we consider solely the pairs in $\mathcal{I}_1$ keeping unchanged the set $\mathcal{I}_0$.

2.4.7 Experimental Results

Several datasets where used to measure the accuracy and time to process the regularization path for the RankSVM algorithm. Firstly, a toy example generated from Gaussian distributions ([Hast 01]) was applied. Some investigations on real life datasets taken from the UCI repository[4] are further presented. Before delving into the details of the experimental results, one of the performance measure used to assess the quality of a ranking function is presented: the discounted cumulative gain.

The ndcg score

The discounted cumulative gain (**dcg**) [Jarv 02, Weim 08] is a score used to measure the accuracy of a ranking. Its particularity is that it assigns more weight to the first m results as in a search engine, where the user will typically pay attention only to the first m results. This score has a normalized version (ndcg), which runs between zero and one, the perfect ranking (π_s) will have a score of one while bad rankings will have a score close to zero. The **ndcg** at the m^{th} level is denoted as **ndcg@m**.

Let S be a set. A function $\pi : S \rightarrow S$ is called a ***permutation*** of S if it is injective (one-to-one) and surjective (onto). This function will map a vector to another vector with the same elements but in different order. If the vector $\mathbf{a}$ is permuted by π, $[\pi(\mathbf{a})]_i$ is the index of the element in $\mathbf{a}$ that appears at position i after the permutation. For instance if π_s is a permutation that sorts $\mathbf{a}$ decreasingly, then the following assertion is true:

$$[\pi_s(\mathbf{a})]_i \geq [\pi_s(\mathbf{a})]_{i+1} \quad \forall i \in \{1 \dots n-1\}$$

The **[n]dcg@m** of a sequence $\mathbf{y}$, permuted by π is:

$$\text{ndcg}(\mathbf{y}, m, \pi) = \frac{\text{dcg}(\mathbf{y}, m, \pi)}{\text{dcg}(\mathbf{y}, m, \pi_s)} \tag{2.43}$$

$$\text{dcg}(\mathbf{y}, m, \pi) = \sum_{i=1}^{m} \frac{2^{[\pi(\mathbf{y})]_i} - 1}{\log_2(i+1)} \tag{2.44}$$

where π_s is the permutation which sorts $\mathbf{y}$ decreasingly.

To illustrate this norm, we take a fictitious example for a particular query and a decision function f, its output is shown in Table 2.5.

i-th Sample ($\mathbf{x}_i$)	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$
Real Ranking (y_i)	1	1	3	2	1	3	4
Proposed Ranking by $f(\mathbf{x}_i)$	1	2	3	7	6	4	5

Table 2.5: Example of the use of the ndcg. This table presents the real ranking given by y_i and the proposed ranking given by f, both rankings induce a permutation assigning a new order.

The first row in Table 2.5 is the arbitrary label that a sample has (subindex $_i$), the second row, contains the real relevance (ranking) y_i associated to each of the samples $\mathbf{x}_i$. This ranking induces a perfect order given by permutation π_s. Finally, the third row contains the rank given by the decision function $f(\mathbf{x}_i)$ for each of the samples $\mathbf{x}_i$, inducing a permutation π.

In order to calculate the ndcg for f, it is necessary to calculate the dcg for the perfect permutation given by π_s and for the proposed permutation π. The given dcg at m with the perfect permutation π_s will be the normalization constant to calculate the ndcg at m. Samples $\mathbf{x}_i$ and real rankings $\mathbf{y}$ are ordered according to π_s in Table 2.6, where $\mathbf{y} = (y_1, y_2, ..., y_7)^\top$.

[4] http://archive.ics.uci.edu/ml/datasets.html

i-th Ordered Sample	1	2	3	4	5	6	7
$\pi_s(\mathbf{x})$ = Optimal Order	$\mathbf{x}_7$	$\mathbf{x}_3$	$\mathbf{x}_6$	$\mathbf{x}_4$	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_5$
$\pi_s(\mathbf{y})$ = Optimal Ranking order	4	3	3	2	1	1	1

Table 2.6: Ordered ranking values according to the perfect permutation π_s.

With this information, the dcg at different levels m can be calculated (in this example it is done for $m = 5$ and $m = 3$):

$$\begin{aligned}
\text{dcg}(\mathbf{y}, 5, \pi_s) &= \frac{2^4-1}{\log_2(1+1)} + \frac{2^3-1}{\log_2(2+1)} + \frac{2^3-1}{\log_2(3+1)} + \frac{2^2-1}{\log_2(4+1)} + \frac{2^1-1}{\log_2(5+1)} = 24.59 \\
\text{dcg}(\mathbf{y}, 3, \pi_s) &= \frac{2^4-1}{\log_2(1+1)} + \frac{2^3-1}{\log_2(2+1)} + \frac{2^3-1}{\log_2(3+1)} = 16.85
\end{aligned}$$

The same is done considering the proposed permutation π, as shown in Table 2.7. Again the formula for the dcg is applied at the same levels as previously done with π_s $m = 3$ and $m = 5$:

i-th Ordered Sample	1	2	3	4	5	6	7
$\pi(\mathbf{x})$ = Proposed Order	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_7$	$\mathbf{x}_6$	$\mathbf{x}_3$	$\mathbf{x}_2$	$\mathbf{x}_1$
$\pi(\mathbf{y})$ = Proposed Ranking order by f	2	1	4	3	3	1	1

Table 2.7: Ordered ranking values according to the permutation π fetched by the ranking function.

$$\begin{aligned}
\text{dcg}(\mathbf{y}, 5, \pi) &= \frac{2^2-1}{\log_2(1+1)} + \frac{2^1-1}{\log_2(2+1)} + \frac{2^4-1}{\log_2(3+1)} + \frac{2^3-1}{\log_2(4+1)} + \frac{2^3-1}{\log_2(5+1)} \approx 16.85 \\
\text{dcg}(\mathbf{y}, 3, \pi) &= \frac{2^2-1}{\log_2(1+1)} + \frac{2^1-1}{\log_2(2+1)} + \frac{2^4-1}{\log_2(3+1)} \approx 11.13
\end{aligned}$$

Then, the Normalized Discounted Cumulative Gain can be easily calculated:

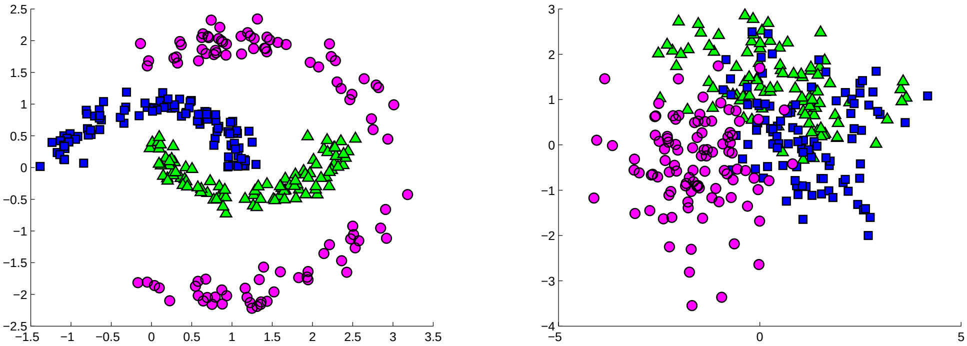
$$\begin{aligned}
\text{ndcg}(\mathbf{y}, 5, \pi) &= \frac{\text{dcg}(\mathbf{y}, 5, \pi)}{\text{dcg}(\mathbf{y}, 5, \pi_s)} = 0.68524 \\
\text{ndcg}(\mathbf{y}, 3, \pi) &= \frac{\text{dcg}(\mathbf{y}, 3, \pi)}{\text{dcg}(\mathbf{y}, 3, \pi_s)} = 0.48572
\end{aligned}$$

Consistency of the Reduced Graph

In this section we will illustrate the results given by the reduced graph proposed in the previous section. This section gives an intuitive idea of the form of the given solution using an induced constraint matrix with fewer constraints.

The reduced graph was analyzed with two artificial datasets, a generated separable three moons dataset and a combination of Gaussian mixtures. Both examples represent one query and have three rank levels. The samples distributions are shown in the Figure 2.8. Three relevances are represented, relevance one with green triangles, relevance two with blue squares and the most relevant, relevance three, with magenta circles.

In both cases, the regularization path was run and the solution was observed at a particular value of λ. For the three moons, the solution was registered for several graph designs changing at each time the chosen most representative sample of each rank. In the case of the three moons, the regularization path was run, and, as this is a separable case, it finished when all constraints were satisfied. This means that at the end of the regularization path (when $\lambda \approx 0$), the decision function had a margin of one between each class. Several runs with different graph constructions



(a) Three moons example

(b) Three Gaussian mixtures example

Figure 2.8: Examples for the reduced graph test. Magenta circles are the most relevant samples, blue squares the second more relevant and green triangles are the least relevant samples.

for the three moons example (where the maximum of each class was changed) are shown in Figure 2.9. The regularization path was observed at an arbitrary value of $\lambda \approx 1$ for each run, knowing that the solution with no error on the training set will occur when $\lambda \approx 0$. Some values of the decision function are illustrated as level curves and it can be seen that the algorithm tries to keep all samples with equal ranking on a band value in the decision function evaluation.
The same experiment was done with the three Gaussian mixtures example and the maximum of each rank level was changed at each run. This is a no-separable case, but it can also be seen in Figure 2.10 that samples in the same rank have similar values in the decision function.
The reduced graph for the three mixtures example gives consistent results as the previous one, samples with equal rank are kept in about the same value band on the decision function f.
The previous figures illustrate that the reduce graph forces the level curves of the decision function toward a segmentation of the points according to their rank, that is, the highest values of the decision function will be around the samples with the highest ranks and the value of the decision function will decrease along the space according to the repartition of the sample ranking.

Graphical illustration of the path.

The regularization path was run over several datasets. The mixtures dataset [Hast 04] was originally designed for binary classification with instances $\mathbf{x}_i$ and corresponding labels $y_i \in \{\pm 1\}$. However, it can be viewed as a ranking problem with $E = \{(\mathbf{x}_i, \mathbf{x}_j) \mid y_i > y_j\}$. It contains 100 positive and 100 negative points which would induce 10 000 constraints. The regularization path was run on this dataset, some level curves are depicted, showing that the most relevant class (red points) tends to stay in the part where the function has the largest values. Additionally, a decision function was taken on zero $f(\mathbf{x}) = 0$. This decision boundary can still be improved by observing the generated ROC curve at each level, that is, by counting at which level f_c of the decision function $f(\mathbf{x}) = f_c$ the training and validation error rate is minimal. Figure (2.11) illustrates the decision function for different breakpoints of the regularization path. The initial solution (a) is poor in terms of generalization but after some iterations the results are improved as shown in Subfigure (b). The most interesting solution is illustrated on sub-figure 2.11(b) where almost all constraints are satisfied. If the regularization path keeps going, as the parameter λ decreases the algorithm outputs solutions that overfit the data as in Figures 2.11(c) and 2.11(d).

Experiments with the Regularization Path

Dataset Description Classification and regression problems can also be viewed as ranking problems by constructing a set of constrains E as: $E = \{(\mathbf{x}_i, \mathbf{x}_j) \mid y_i > y_j\}$. The regression

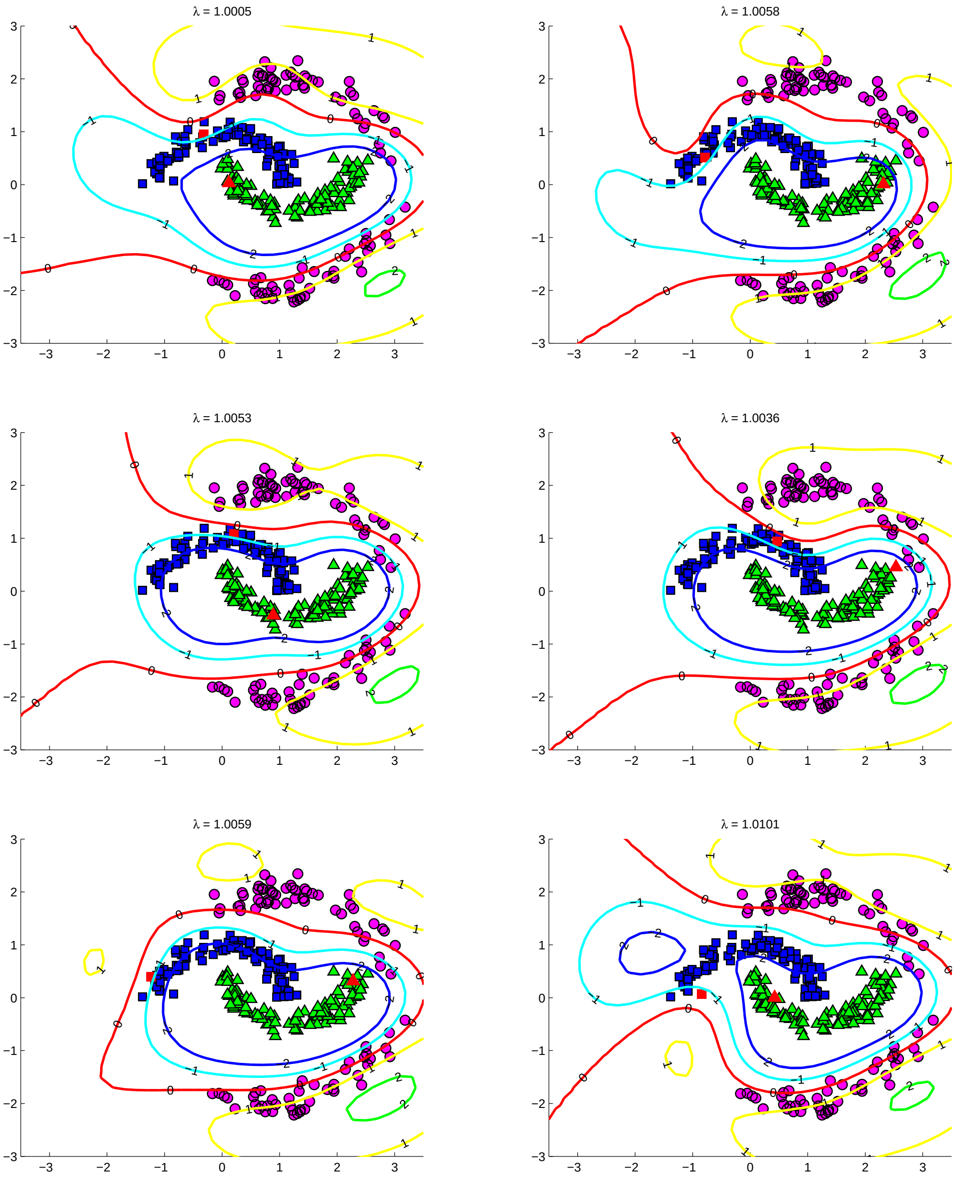
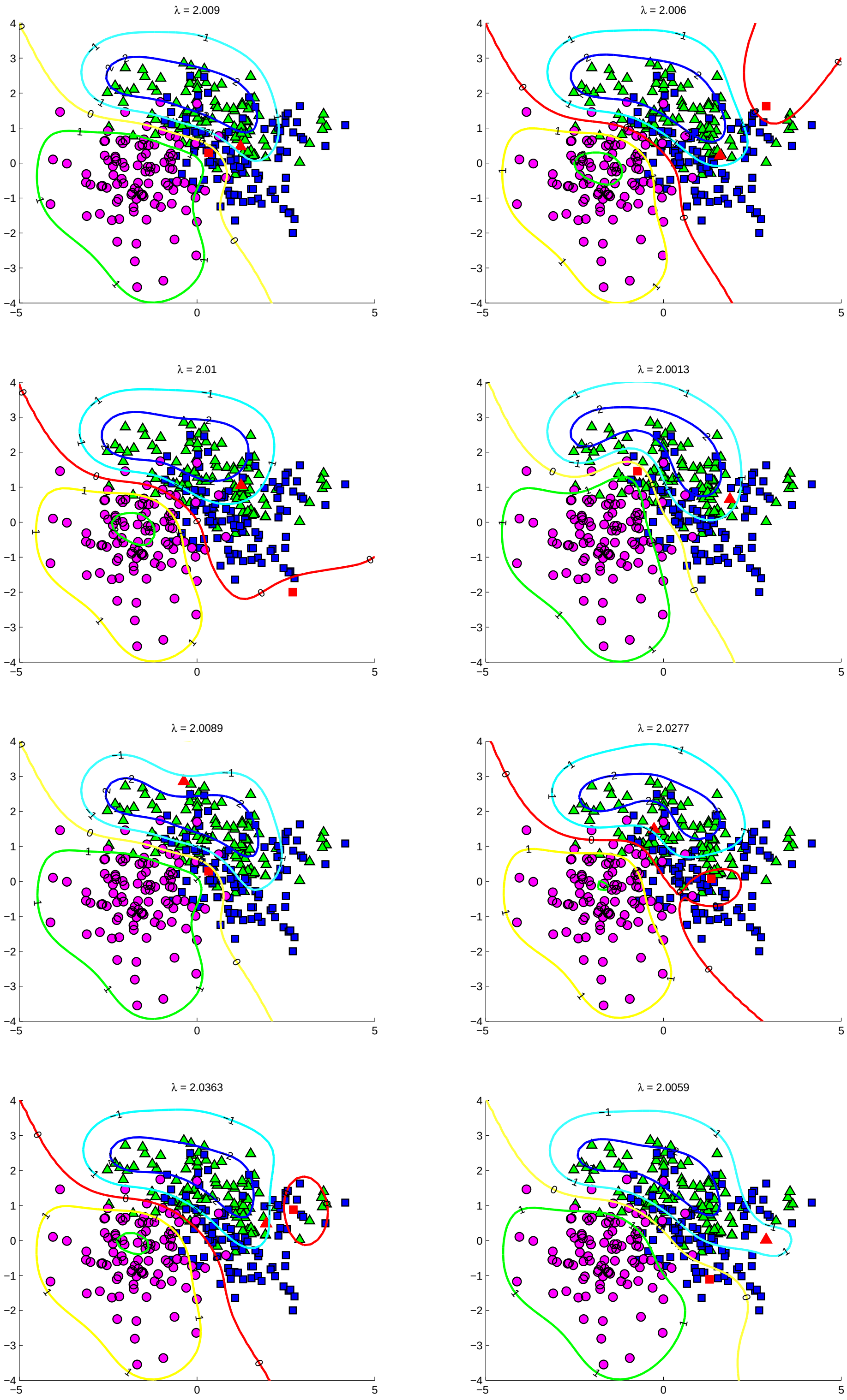


Figure 2.9: Reduced graphs used to solve a three moons problem. Green triangles have rank 1, blue squares rank 2 and magenta circles rank 3. Red points are the chosen maximum for each rank level. Level curves indicates decision function values. Each figure corresponds to a different trial where the most relevant point for each rank was changed. The value of decision function depends on the repartition of the point's ranking.



λ = 2.009

λ = 2.006

λ = 2.01

λ = 2.0013

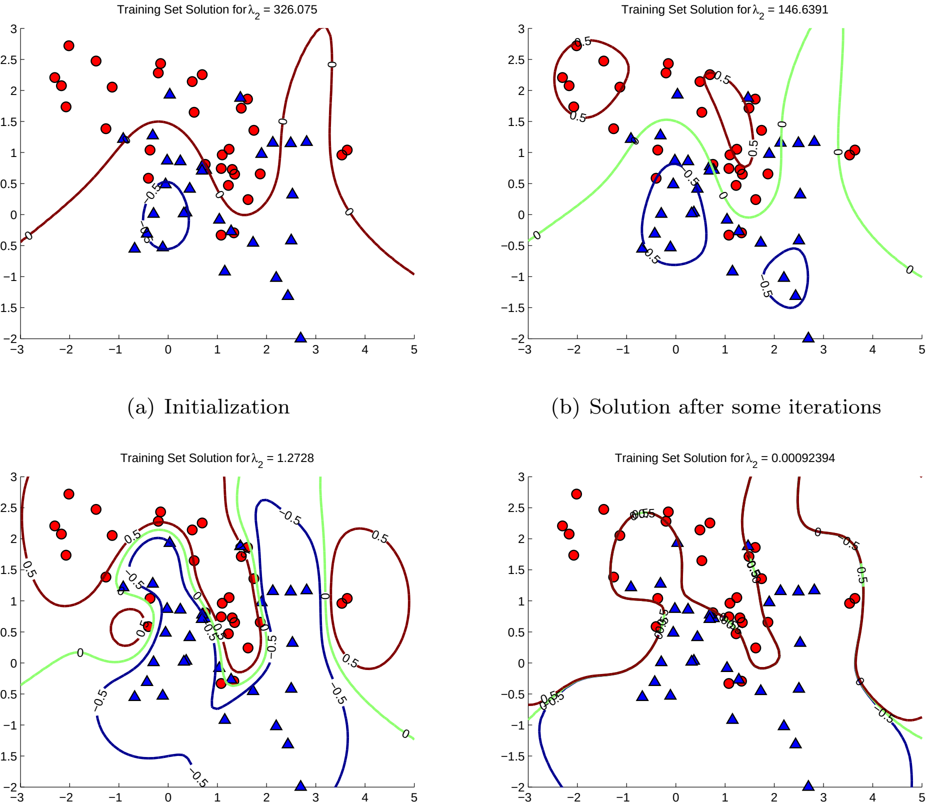
λ = 2.0089

λ = 2.0277

λ = 2.0363

λ = 2.0059

Figure 2.10: Reduced graph used to solve a problem of three Gaussian mixtures. Green triangles have a rank of 1, blue squares a rank of 2 and magenta circles, the highest rank of 3. Red points indicate the chosen maximum for each rank level. Illustration of the problem solution for $\lambda \approx 2$. Some values of the decision function are depicted in the level curves, showing that the solution tries to segment the space according to the sample ranking.



Training Set Solution for λ_2 = 326.075

Training Set Solution for λ_2 = 146.6391

(a) Initialization

(b) Solution after some iterations

Training Set Solution for λ_2 = 1.2728

Training Set Solution for λ_2 = 0.00092394

(c) Solution after more iterations

(d) Solution for the smallest λ

Figure 2.11: Illustration of the regularization path for the mixture dataset, all red points must be ranked higher than the blue points. As λ decreases, the margin gets smaller and the distance between pairs tends to be larger than one.

datasets (*auto* and *housing*) were modified for the ranking problem: both quantiles of $q_{\frac{1}{3}}$ and $q_{\frac{2}{3}}$ are found for the response variable y and the rank for each sample was assigned depending on the position with respect to each quantile, so that if a sample has a regression value lower than $q_{\frac{1}{3}}$, it will have rank 0, if it is higher than $q_{\frac{2}{3}}$, it will have rank 2 and rank 1 otherwise.

The number of induced constraints on the complete dataset and those obtained after following the graph design in Figure 2.7 are compared in Table 2.8. Attention was paid to the number of constraints (rows in P) involved in the multiple bootstrapping methodology to choose a parameter (with a subset of the training set) and in the number of rows of matrix $Ptrain$ used to build the final model with the chosen parameter using the complete training set. One can remark the important decrease in terms of problem complexity which depends on the number of given constraints.

Dataset				Bootstrapping	Final Train
				P rows	$Ptrain$ rows
	Features	Samples	Ranks	comp./red.	comp./red.
mixture	2	200	2	2491/99	5625/149
mixture3	2	300	3	3322/132	16872/297
3moons	2	300	3	3318/134	16866/301
cancer	30	569	2	2379/99	42612/426
diabetes	8	768	2	2155/99	75375/575
auto	7	392	3	3301/133	29007/390
housing	13	506	3	3310/131	48108/501
spam	57	4126	2	n.a./99	n.a./3094
ad	1558	2359	2	n.a./99	n.a./1769

Table 2.8: Datasets characteristics (complementary table). The number of features, samples and ranks is shown together with the average number of constraints used in the bootstrapping search given by both matrix P (number of rows) and by the reduced constraint matrix P' proposed in Section 2.4.2. The number of rows in $Ptrain$ is the number of constrains included in the training of the final model using the complete training set. For some datasets the size of the complete graph is not available (n.a.)

Experimental Protocol In order to measure the usefulness of the proposed methods (the reduced graph and the regularization path), the previously described problems were solved using a grid search and the regularization path. In both cases, bootstrapping was used to choose the parameters.

Five independent repetitions of the experiment were done. At each time, a training set and a test set were taken in order to estimate the generalization ability with the test error. These were averaged over the five experiments to have a more robust estimation.

Subsets of the training set were done to do bootstrapping, at each time, one hundred samples are taken to train a model and fifty samples to validate. This was repeated for ten bootstrap runs.

For the grid search, several models were built with different regularization parameters. Twenty values in the interval $[0.01, 50]$ were taken in a logarithmic scale. One is chosen and a finer search of ten values linearly separated around it are taken. The λ that gives the maximum average validation ndcg is used to build the final model.

For the regularization path, ten complete regularization paths are calculated for the sub-samples of one hundred points. The validation error is measured at thirty values distributed according to the given breakpoints of the regularization path, this means that the distribution of the thirty points depend on the distribution of the breakpoints, so that more samples will be taken around more dense areas of the breakpoint distribution. It has to be noticed that at each bootstrap trail the evaluated values for the regularization parameter will not be necessarily the same. A validation curve is obtained by averaging with all the bootstrap trials.

For both cases, the ndcg@30 was used as validation measure. The reason behind this choice is the fact that the ndcg applied to the first positions tends to be unstable.

Experimental results with the reduced graph. Grid Search vs. Regularization Path The main issue of the resolution of ranking problems lies in the fact that the use of the complete graph requires a calculation time that increases with the number of constraints to the point of making it unfeasible, either because the time turns too long, because of memory problems or because it turns numerically instable. Experiments were done with the above mentioned examples using the reduced graph.

Table 2.9 summarizes the obtained errors with the reduced graph. One measure is the percentage of pairs not respecting the ranking order (% error), this corresponds to the measure:

$$\frac{\sum_{i=1}^{m_{te}} I_{P_{i,\cdot}[f(\mathbf{x}_1),f(\mathbf{x}_2),\dots,f(\mathbf{x}_{n_{te}})]^\top \leq 0}}{m_{te}}$$

where n_t is the number of test samples and m_{te} the number of test constraints (in the complete graph). For both cases, it can be observed that quite good results can be obtained as the ndcg at different levels is close to 1 and the percentage error is very low.

Nevertheless, the model output by the regularization path search seems to be slightly better than the ones obtain by grid search in all cases.

Dataset	Grid Search				Reg. Path			
	Error (%)	ndcg			Error (%)	ndcg		
		@1	@5	@10		@1	@5	@10
mixture	15	1	1	0.96	13	1	1	0.98
mixture3	11	1	1	1	9	1	1	1
3moons	1	1	1	1	0	1	1	1
cancer	2	1	0.96	0.97	2	1	1	1
diabetes	22	0.73	0.81	0.81	18	0.87	0.94	0.9
auto	6	1	1	0.95	5	1	1	1
housing	16	1	0.97	0.97	8	1	0.97	0.98
spam	18	1	1	1	6	1	1	1
ad	7	0.87	0.9	0.93	3	0.87	0.94	0.94

Table 2.9: ndcg errors at different levels.

Table 2.10 compares several details in the resolution of the ranking problem with the reduced graph: the time for the grid search and the regularization path is measured on the bootstrapping setting in the two first columns, only a subset of the population of size one hundred was used on the bootstrapping setting. The time taken to solve a single model against a complete regularization path is shown in the last two columns. A search with the regularization path is faster than solving multiple times a single ranking problem in the dual formulation, that is, as a QP.

A Gaussian kernel was used in all cases with a bandwidth σ depending on the extent of the training sample domain. As Gaussian models result in a mixture of several Gaussian functions on certain samples, the chosen value should be able to cover the complete domain, but it also has to be fine enough to be able to define accurate models. If the domain belongs to a cube with all sides in the interval $[x_{min}, x_{max}]$, the kernel parameter taken is: $\frac{x_{min}, x_{max}}{10}$.

Table 2.11 gives an idea of the stability of the chosen parameter as the standard deviation of the given λ is shown together with the mean value. The used kernel bandwidth is given in the middle column. The size of the model, where $\mathcal{A}$ is the number of active variables, that is, the number of coefficients β_i involved in the model with value different from zero. The last column has the average number of breakpoints resulting on the resolution of a regularization path of the final model, this gives an idea of its complexity.

The advantage of the regularization path can be appreciated as a larger time is needed to do grid search (Table 2.10), additionally, the estimation of the proper regularization parameter to be used is more robust with the calculation of regularization paths (Table 2.11). For the final model, a single resolution of one problem remains faster than a complete regularization path.

Dataset	Bootstrapping		Final Train	
	Grid Time	Reg Path Time	Grid Time	Reg Path Time
mixture	18.7s	14s	0.1s	3.5s
mixture3	29.4s	22.5s	0.2s	19s
3moons	30.7s	20s	0.1s	12.4s
cancer	17.9s	9.6s	0.2s	22.3s
diabetes	16.6s	11.3s	1.5s	108.2s
auto	26s	16.5s	0.3s	18.9s
housing	27s	17.8s	0.8s	56.1s
spam	48.2s	12.4s	92.1s	9770.3s
ad	44.3s	39.3s	10.6s	2182.6s

Table 2.10: Search and training time in the case of grid search and regularization path. The first two columns indicates the time for the parameter search under a bootstrap framework. The last two columns have the average training time of the final model.

Dataset	Grid λ value	Reg Path λ value	used kernel	Grid Size $\mathcal{A}$	Reg Path Size $\mathcal{A}$	Number Breakpoints
mixture	4.37 (7.5)	0.4 (0.4)	0.65	97.6 (38.8)	83.6 (12.2)	243 (7)
mixture3	6.13 (7)	1.36 (0.7)	0.82	143.2 (6.1)	154.2 (6.4)	349 (16)
3moons	3.93 (7.5)	1.29 (1.2)	0.53	94.6 (56.2)	109.8 (32.4)	373 (13)
cancer	0.39 (0.5)	6.32 (11.5)	343.8	82.4 (16.5)	104.6 (70.9)	241 (7)
diabetes	5.49 (12)	0.45 (0.6)	75.16	364.4 (82.4)	317.6 (3.5)	223 (5)
auto	5.18 (9.2)	2.74 (6.1)	511.04	197.8 (24.3)	171.2 (36.7)	308 (5)
housing	7.75 (11)	0.07 (0.2)	71.1	270.8 (7.5)	232.4 (11.7)	318 (12)
spam	0.26 (0.3)	0.03 (0.1)	1006.2	2032.6 (71.5)	1521.2 (101.6)	98 (10)
ad	0.66 (1.4)	0.73 (1)	64	408 (64.4)	389 (32.6)	259 (13)

Table 2.11: General model Information. The average chosen regularization parameter λ is given with its standard deviation in parenthesis; the used kernel parameter and the number of variables involved in the final model for both parameter searches is given (Size of $\mathcal{A}$). Finally, the number of found breakpoints in the regularization path is indicated in the last column.

Experimental results, reduced graph vs. complete graph The reduced graph was also tested in the ranking algorithm proposed by Chapelle [Chap 07a], this is a ranking problem that uses a square hinge-loss and also a L_2 regularization term. A gradient based method is done to find the solution, this method is very efficient in terms of computational time. This algorithm was used to compare the output results using the complete graph as in the original paper [Herb 00] and the proposed reduced graph in this work.

The used dataset is the `ohsumed` dataset [Xu 07, Hers 94]. This is a collection of medical queries and documents ranked by experts. There are five folds in the ohsumed dataset, for each one, the ndcg@10 was observed on the test set.

In Table 2.12, the errors were measured using a fix regularization parameter $C = 10^{-4}$, the accuracy was decreased with respect to the use of the complete graph but with a significant gain in training time. Only the training set was used to train the final model that was used with the test set to measure an error.

Dataset	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Time
ohsumed (complete graph)	0.38	0.47	0.46	0.47	0.50	8.99s
ohsumed (reduced graph)	0.31	0.41	0.35	0.42	0.43	0.28s

Table 2.12: Results of the ndcg@10 with $C = 1e-4$ ($\lambda = 10000$) for each of the ohsumed dataset's folds with linear kernel. The last column contains the average time of training in seconds.

The following experiment consisted in using the given validation set at each fold to search the appropriate regularization parameter $C = \frac{1}{\lambda}$ for the final model of each fold.

Dataset	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Time
ohsumed (complete graph)	0.38	0.44	0.40	0.48	0.42	105s
used C	41.18	0.02	0.01	0.5	2.9	349.4s
ohsumed (reduced graph)	0.32	0.43	0.36	0.43	0.42	0.42s
used C	0.02	0.02	0.01	17.6	0.06	24.3s

Table 2.13: ndcg@10 for each of the ohsumed folds with linear kernel. Grid search was used to estimate the regularization parameter C for each fold. The average training time in seconds of the final model is given in the same row as the ndcg error while the average time of the grid search is given in the same row as the regularization parameter C.

For the experiments in Table 2.13, the regularization parameter C was searched with a grid using the training set to find a model and evaluating it in the given validation set at each fold. With the chosen parameter, the error was then measured in the test set for each fold with a model issued from the training and validation set together. For the reduced graph, the accuracy on the ndcg was improved in almost all folds, indicating that if the proper regularization parameter is used, the obtained results can be close to the ones obtained with the complete graph. The results got even closer to the one of the complete graph as for it the accuracy was strangely reduced in almost every fold with respect to the results obtained with an a priori chosen regularization parameter.

It is worth noticing that with a training-validation-test sets framework for the parameter choice, the chosen regularization parameter is less stable in the complete graph than in the reduced one. In the first case it runs in the $[0 - 42]$ interval while in the reduced case it belongs to a smaller interval: $[0 - 18]$.

With these results, it can be concluded that the reduced graph is a consistent and good approximation of the complete graph and gives reasonable results in toy and real problem. The next question to be solved is to be able to use the reduced graph as a warm start sub-solution for the complete one.

2.4.8 Conclusions

Regularization parameter search for the ranking SVM can be efficiently done by calculating the regularization path. This approach calculates efficiently the optimal solution for all possible regularization parameters by solving (in practice) small linear problems. This approach has the advantage of overcoming local minima of the validation error curve, in case there exists. Parameter selection is less time consuming and the obtained results are more robust, resulting in a slightly better generalization capacity with regularization paths than with grid search.

The numerical complexity of the algorithm depends on the number of iterations needed to explore the overall solution path and the mean size of $\mathcal{I}_\alpha$. At each iteration, a linear system is solved to get $\boldsymbol{\eta}$ which has complexity $O(|\mathcal{I}_\alpha|^2)$. Empirically we observed that the number of iterations for a regularization path resolution is typically about the same as the number of training pairs constraints m in the reduced graph.

For a ranking problem with quadratic cost and regularization term, the output given by the reduced graph are close to the ones obtained with the complete graph, improvements could probably be done by an even deeper parameter search.

In SVM methods, another key point is the determination of kernel hyper-parameter. This problem was not tackled here. However, one can seek to combine our regularization path with the kernel parameter path developed in [Gang 07].

2.5. Sparse Ranking SVM

In the previous section, the regularization parameter is efficiently sought via a regularization path. Still, the optimal solution is expressed using all training points. Indeed the presented L_2 norm Ranking SVM involves a number of ordering constraints. If the number of samples is n, then, the number of constraints is of order $O(n^2)$ or $O(n)$ in the reduced graph proposed in Section 2.4.2. As the used cost function is the hinge loss, its piecewise linear variation induces a sparsity in the number of constraints involved in the model expression. Basically, all constraint pairs that are in set $\mathcal{I}_0$ will not influence the model as the gradient of the loss function will be zero for those points. Unfortunately, the number of parameters remains in general of the order of the training set size. To earn more sparsity in the number of parameters, the use of the L_1-norm as regularization term is here investigated, replacing the L_2 regularization, and the L_2-norm will be used as penalization term. A good summary of different methods to solve a problem under L_1 norm constraint is given by Schmidt [Schm 07]. The objective is to help to control overfitting and to obtain a reduced model with similar accuracy.

It will be assumed that the searched function has the same form as in Section 1.5.2 [Rifk 07]: $f(\mathbf{x}_i) = \boldsymbol{\beta}^\top \mathbf{k}(\mathbf{x}_i)$ where $\mathbf{k}(\mathbf{x})$ is the vector containing as $i-th$ entry $\mathbf{k}(\mathbf{x}, \mathbf{x}_i)$ for, with $\mathbf{x}_i, i \in \mathcal{X}$ the learning set and $\mathbf{k}$ the reproducing kernel in a RKHS. We can define the L_1-norm of f as

$$\|\boldsymbol{\beta}\|_1 = \sum_{j=1}^{n} |\beta_j|.$$

In this formulation, the complete graph is used, but the reduced graph could also be applied, nevertheless, the complete graph was decided to be kept as the introduction of a L_1 norm will impose more constraints. A regularization path will be built, unfortunately, it was observed in the previous section that this approach turn out to be extremely slow because there are many events given by the L_2 regularization term (see Table 2.10), a solution will be to eliminate the L_2 regularization term. By removing this L_2 term, the number of events that have to be detected are considerably reduced, leading to a faster resolution. Additionally, for the ranking problem, there exists other successful approaches that have this form [Chap 07a].

Using the L_1-norm to regularize will help to control the number of terms in the final model. Instead of the hinge loss, we turn here to a least squares loss, so that the optimal coefficients $\boldsymbol{\beta}$

are the result of the following L_1-norm Ranking SVM optimization problem:

$$\begin{array}{rl} \underset{\boldsymbol{\beta}\in\mathbb{R}^n,\boldsymbol{\xi}\in\mathbb{R}^m}{\operatorname{argmin}} & \frac{1}{2}\boldsymbol{\xi}^\top\boldsymbol{\xi} \\ \text{s. t.} & \boldsymbol{\beta}^\top\big(\mathbf{k}(\mathbf{x}_{u_i}) - \mathbf{k}(\mathbf{x}_{v_i})\big) \geq 1 - \xi_i \quad \forall i = [\![m]\!] \\ & \|\boldsymbol{\beta}\|_1 \leq s \end{array} \tag{2.45}$$

Notice that in this case, $s \in \mathbb{R}^+$ is a hyperparameter which controls the sparsity of the solution. And therefore, the optimal solution depends on the choice of this hyperparameter. The final permutation π is then obtained by sorting V according to f and resolving ties randomly.

Using the aforementioned expression of f, this problem can be rewritten as a least squares ranking SVM with a L_1 penalty:

$$\begin{array}{rl} \underset{\boldsymbol{\beta}\in\mathbb{R}^n}{\operatorname{argmin}} & \frac{1}{2}(\mathbb{1} - PK\boldsymbol{\beta})^\top(\mathbb{1} - PK\boldsymbol{\beta}) \\ \text{s. t.} & \sum\limits_{j=1}^{n} |\beta_j| \leq s \end{array} \tag{2.46}$$

As it can be seen, the problem in Equations (2.46) is a L_2 cost function with a L_1-norm regularization consisting in estimating n parameters using m data (with $m = \mathcal{O}(n^2)$ or $m = \mathcal{O}(n)$) if the reduced graph is used).

There exist several efficient methods to find the optimal solution for this problem that consists on a L_2-loss function plus a L_1-regularization function with a fixed s-value, an example is the Path-wise Coordinate Optimization [Frie 07]. Since our interest is to choose s, that is, model selection, we will focus ourselves to a method that progressively solves the problem for all s.

This problem can also be related to the Lasso algorithm in regression [Tibs 96] and among the tools used for solving this problem, we can mention an efficient one which is the LARS (least angle regression and stepwise) [Efro 04]. Basically, the LARS algorithm relies on the derivation of a regularization path according to the variation of s. Hence, in this section, we adapt the LARS methodology to the least squares ranking SVM. There exists several methods to solve the LASSO problem [Schm 07, Figu 07, Duch 08], nevertheless, we are interested in solving this problem not only for a particular set of s values but for all the possible values. The LARS is an efficient approach that allows solving the LASSO problem for all value of s and is therefore an attractive tool.

The Lagrange function corresponding to problem in Equations 2.46 reads

$$\mathbf{L} = \mathcal{L}(\boldsymbol{\beta}) + \lambda\left(\sum_{j=1}^{n}|\beta_j| - s\right)$$

where $\mathcal{L}(\boldsymbol{\beta}) = \frac{1}{2}(\mathbb{1} - PK\boldsymbol{\beta})^\top(\mathbb{1} - PK\boldsymbol{\beta})$ stands for the loss function and $\lambda \geq 0$ is the Lagrange parameter associated to the L_1 penalty. The L_1 penalty term being non-differentiable, the derivation of the optimality condition relies on the use of the subdifferential. For a parameter β_j, this condition turns out to be

$$0 \in \nabla_{\beta_j}\mathcal{L}(\boldsymbol{\beta}) + \lambda g(\beta_j). \tag{2.47}$$

In this expression $g(\beta_j)$ represents the subdifferential of the absolute value function defined as

$$g(\beta_j) = \left\{ \begin{array}{lll} \operatorname{sign}(\beta_j) & \text{if} & \beta_j \neq 0 \\ \gamma, \quad \text{with } -1 \leq \gamma \leq 1 & \text{if} & \beta_j = 0 \end{array} \right. .$$

From this definition, if we split the variables into two sets:

- $\mathcal{A} = \{j = [\![n]\!] \mid \beta_j \neq 0\}$, set of active variables
- $\bar{\mathcal{A}} = \{j = [\![n]\!] \mid \beta_j = 0\}$ set of inactive variables

Equation (2.47) and the form of a subgradient can be used to establish the following properties:

- for $\beta_j \in \mathcal{A}$, we have $|\nabla_{\beta_j}\mathcal{L}(\boldsymbol{\beta})| = \lambda$ and
- for $\beta_j \in \bar{\mathcal{A}}$, we have $|\nabla_{\beta_j}\mathcal{L}(\boldsymbol{\beta})| \leq \lambda$.

where $\nabla_{\beta_j}\mathcal{L}(\boldsymbol{\beta})$ is the i-th element of this vector. In the LARS original paper by Efron et al. [Efro 04] these properties were termed as correlation conditions because the derivative $\nabla_{\boldsymbol{\beta}}\mathcal{L}(\boldsymbol{\beta}) = -KP^\top(\mathbb{1} - PK\boldsymbol{\beta})$ is simply a vector containing the correlations of the variables with the residuals (in case the covariates and the output are centered and normalized).
To better understand the derivation of the two previous properties, since $\lambda > 0$, Equation (2.47) can be then analyzed by cases:

- for $j \in \mathcal{A}$

$$\begin{array}{lll} \beta_j > 0, \quad \lambda > 0 & \Leftrightarrow & \nabla_{\beta_j}\mathcal{L}(\boldsymbol{\beta}) = -\lambda, \quad j \in_{\mathcal{A}} \\ \beta_j < 0, \quad \lambda > 0 & \Leftrightarrow & \nabla_{\beta_j}\mathcal{L}(\boldsymbol{\beta}) = \lambda \quad j \in_{\mathcal{A}} . \end{array}$$

leading to the condition $|\nabla_{\beta_j}\mathcal{L}(\beta)| = \lambda$ which under the specific problem formulation reads

$$| - (KP^\top\mathbb{1})_{\bar{\mathcal{A}}} + (KP^\top PK\boldsymbol{\beta})_{\bar{\mathcal{A}}}| = \lambda \tag{2.48}$$

- for $j \in \bar{\mathcal{A}}$, since $-1 \leq g(|\beta_j|) \leq 1$:

$$-\nabla_{\beta_j}\mathcal{L}(\beta) = \lambda g(|\beta_j|) \leq \lambda$$

and

$$-\nabla_{\beta_j}\mathcal{L}(\beta)) = \lambda g(|\beta_j|) \geq -\lambda$$

leading to the condition $|\nabla_{\beta_j}\mathcal{L}(\beta)| \leq \lambda$ which here is $| - (KP^\top\mathbb{1})_{\bar{\mathcal{A}}} + (KP^\top PK\boldsymbol{\beta})_{\bar{\mathcal{A}}}| \leq \lambda$.

To derive the regularization path, let us assume that $\boldsymbol{\beta}^t$ is the solution corresponding to s^t, $\mathcal{A}^t$ and $\bar{\mathcal{A}}^t$ is the associated sets. According to Equation (2.47) $\boldsymbol{\beta}^t$ will meet the general condition

$$- (KP^\top\mathbb{1})_{\mathcal{A}} + (KP^\top PK\boldsymbol{\beta})_{\mathcal{A}} + \lambda\,\mathrm{sign}(\boldsymbol{\beta}_{\mathcal{A}}) = 0 \tag{2.49}$$

along with the strict complementary one

$$\sum_{j\in\mathcal{A}} |\beta_j| = \mathrm{sign}(\boldsymbol{\beta}_{\mathcal{A}})^\top\boldsymbol{\beta}_{\mathcal{A}} = s. \tag{2.50}$$

Assume that it exists a value s^{t+1} such that if s increases to a smaller value than s^{t+1}, the sets $\mathcal{A}^t$ and $\bar{\mathcal{A}}^t$ remain unchanged. Hence, Equations (2.49) and (2.50) will still hold for $s < s^{t+1}$, but the active parameters must evolve. If $\Delta\theta$ denotes the change in parameter θ, the following system is obtained:

$$\begin{pmatrix} (KP^\top PK)_{\mathcal{A}\mathcal{A}} & \mathrm{sign}(\boldsymbol{\beta}_{\mathcal{A}}) \\ \mathrm{sign}(\boldsymbol{\beta}_{\mathcal{A}})^\top & 0 \end{pmatrix} \begin{pmatrix} \Delta\boldsymbol{\beta}_{\mathcal{A}} \\ \Delta\lambda \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \Delta s \end{pmatrix}$$

the last equation was derived by a simple differentiation of the two previous equations applied for s^t and any value s between s^t and s^{t+1}. From this relation, we are able to derive the overall regularization path. To begin, let focus on the initialization of the algorithm.

2.5.1 Initialization

A starting solution has to be found to define the sets and calculate the adjusting slope for parameters update. This initialization can be obtained via one off the shelf Lasso solver for a fixed value of λ providing hence an initial value of s, $\boldsymbol{\beta}$ and elements of set $\mathcal{A}$. One alternative is to consider the smallest meaningful value of s from which the entire path is run. For this sake, one can notice that if $s = 0$, the trivial solution is $\boldsymbol{\beta} = \mathbf{0}$. By increasing appropriately s, we allow at least one variable to be active. Therefore, the active set is initialized as $\mathcal{A} = \text{argmax}_j \, |\nabla_{\beta_j} \mathcal{L}(\boldsymbol{\beta})|$ as well as the initial value of $\lambda = |\nabla_{\beta_j} \mathcal{L}(\boldsymbol{\beta})| > 0$. The meaning of this initialization is simple: the first active parameters correspond to the variables in the model which will lead to the maximum decrease of the learning error.

From this solution, s can be increased up to a stopping criteria that could be when all variables are included in the model.

2.5.2 Event Detection

If we let H to be the matrix,

$$H = \begin{pmatrix} (KP^\top PK)_{\mathcal{A}\mathcal{A}} & \text{sign}(\boldsymbol{\beta}_{\mathcal{A}}) \\ \text{sign}(\boldsymbol{\beta}_{\mathcal{A}})^\top & 0 \end{pmatrix}$$

the following linear system hold as long as the active set $\mathcal{A}$ remains unchanged

$$\begin{pmatrix} \boldsymbol{\beta}_{\mathcal{A}} \\ \lambda \end{pmatrix} = \begin{pmatrix} \boldsymbol{\beta}^t_{\mathcal{A}} \\ \lambda^t \end{pmatrix} + \boldsymbol{\eta}(s - s^t) \tag{2.51}$$

with $\boldsymbol{\eta} = H^{-1} \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}$. From this point, two types of events have to be monitored.

- Check if a variable $\beta_j \in \mathcal{A}^t$ becomes inactive
  We need to solve (2.51) for s with $\beta_j = 0$. That is:

$$s_j = s^t + \frac{0 - \beta^t_j}{\eta_j} \quad \forall j \in \mathcal{A}^t \tag{2.52}$$

- Verify if a variable $\beta_j \in \bar{\mathcal{A}}^t$ will turn active
  For this situation to occur, Equation (2.48) must hold. If we define:

$$c_j = \nabla_{\beta_j} \mathcal{L}(\boldsymbol{\beta}) = -(KP^\top \mathbb{1})_j + (KP^\top PK\boldsymbol{\beta})_j \quad \forall j \in \bar{\mathcal{A}} \tag{2.53}$$

  then, using Eqs. (2.51) and (2.53), we can derive:

$$\begin{aligned} c_j - c^t_j &= (KP^\top PK\boldsymbol{\beta})_j - (KP^\top PK\boldsymbol{\beta}^t)_j \\ &= (KP^\top PK(\boldsymbol{\beta} - \boldsymbol{\beta}^t))_j \\ &= (KP^\top PK\boldsymbol{\eta}(s - s^t))_j \\ &= \boldsymbol{\eta}_{c_j}(s - s^t) \end{aligned}$$

  by letting $\boldsymbol{\eta}_{c_j} = (KP^\top PK\boldsymbol{\eta})_j$. And therefore, for a variable $\beta_j \in \bar{\mathcal{A}}^t$ that moves to $\mathcal{A}$, Equation (2.48) must be satisfied:

$$\text{If } \beta^{t+1}_j > 0 \Rightarrow s_j = s^t - \frac{\lambda^t + c^t_j}{\eta_{c_j} + \eta_\lambda} \quad \forall j \in \bar{\mathcal{A}}^t \tag{2.54}$$

$$\text{If } \beta^{t+1}_j < 0 \Rightarrow s_j = s^t + \frac{\lambda^t - c^t_j}{\eta_{c_j} - \eta_\lambda} \quad \forall j \in \bar{\mathcal{A}}^t. \tag{2.55}$$

Then, s^{t+1} will be the smallest s_j from Eqs. (2.52), (2.54) and (2.55) such that $s_j > s^t$.

The overall algorithm is closely similar in spirit to the one of L2 rankingSVM (except the types of events to be managed). For this reason, the sparse ranking SVM algorithm will not be developped here.

2.5.3 Experiments

Experiments were done on different datasets from the UCI repository. A 5-fold cross-validation was done to measure the error. For model selection, at each fold, a training and a validation set were randomly chosen. Table 2.14 summarizes the datasets characteristics, the number of features, the number of training instances and the number of induced constraints. Table 2.15 summarizes the achieved error in misclassification percentage, and the ndcg score for the 1,3,5,10 and 20 first ranked points. The needed time to choose additional model and kernel parameters, the chosen s and the final size of the active set A is shown in Table 2.16.

Dataset	Data size	Training size	Training Constr.
mixtures	100	80	1590
mixture3	150	120	4789
3moons	150	120	4788
diabetes	615	492	54958
cancer	285	228	12112

Table 2.14: Data characteristics, size of the dataset, number of training instances, number of training constraints.

Dataset	Error (%)	ndcg@1	ndcg@3	ndcg@5	ndcg@10	ndcg@20
mixtures	20	1	1	1	0.93	0.91
mixture3	12	1	1	1	1	1
3moons	2	1	1	1	1	1
diabetes	34	0.87	0.80	0.75	0.73	0.70
cancer	46	1	1	1	1	1

Table 2.15: Test error. Percentage of misclassified pairs, normalized discounted cumulative gain at the 1, 3, 5, 10 and 20 positions.

Comparison between Tables 2.9 and 2.15 shows that the accuracy was not substantially affected in this new framework with respect to the use of the reduced graph. Additionally, the number of variables involved is considerably smaller. The cost that has to be paid is the time of training (Tables 2.10 and 2.16) plus the additional time needed for the parameter search (not shown).

Dataset	Training Time	Kernel used	s value	$\|\mathcal{A}\|$
mixtures	2.65 (0.2)	5	40.6 (49.6)	4.5 (0.7)
mixture3	15 (0.9)	5	7975.9 (11259.2)	10 (7.1)
3moons	14.9 (0.5)	5	291530.6 (400600)	19 (12.7)
diabetes	988.1 (18.2)	5	4.5 (1.7)	26.6 (10.6)
cancer	11.63 (1)	5	0.42 (0.2)	7.5 (3.5)

Table 2.16: Required time in seconds, used Kernel, optimal s, number of active variables

These results show that a satisfactory model reduction can be obtained by the use of an L_1 norm in the model coefficients.

2.5.4 Conclusions

Using the L_1 norm instead of the L_2 norm as a regularizer, leads indeed to a sparser model, but as it can be expected, accuracy was reduced. On the other hand, the training time was substantially reduced. A good example is the ohsumed dataset, which would take a day with the L_1 norm, while the regularization path with the L_2 norm has been running for weeks. Nevertheless, the complete validation error curve could be analyzed in order to decide which local minimum is the optimal one for generalization.

2.6. Conclusions

In this chapter, the classification and ranking problems were analyzed under a SVM framework, their solution is related as classification problems can be turned into ranking ones, eventually, ranking problems could be transformed into multi-class problems. Theoretically, it was proved under several assumptions, that in the case of binary problems, the solution set of the SVM classification framework and the SVM ranking one are the same.

The common point of these algorithms is the fact that their optimal solution has a linear relation by pieces with respect to the regularization parameter. This piecewise linearity helps to efficiently find the optimal solutions corresponding to each regularization parameter via a regularization path. The regularization path derivation was extended for the ranking framework.

The developed linear relationship is valid in given intervals of the regularization parameter. These intervals are defined by breakpoints, where the corresponding slope changes. The stated problem of analyzing the optimal solution along the whole parameter space get significantly reduced by the fact that is was viewed that with high probability, the only worthy values are the ones given by the regularization path, which are a finite number of the order of the size of the dataset. Therefore, if a validation set has to be tested to choose a model, it is enough to test it at each breakpoint. Nevertheless, if we are willing to study the complete parameter space with respect to a validation set, this is possible as it is given by the validation path, where the validation error can be determined for all values and is piecewise constant; and again, the number of breakpoints of the validation error turns out to be finite.

The rankingSVM setting has a particular difficulty: the used datasets tend to be very large. This cause a very long calculation time for several resolution algorithms of the rankingSVM problem and their regularization path. This issue was tackled by two propositions: the first one was a robust reformulation of the preference graph, leading to a significantly smaller number of constraints; as a natural consequence, the accuracy is reduced. The other solution was to change the used loss and regularization functions, to get rid of the large number of encountered breakpoints caused by the hinge-loss function while only considering the breakpoints caused by the L_1-norm regularization that additionally control the sparsity of the model.

3 Model Selection in Semi-Supervised and Unsupervised Learning

Our everyday world is characterized by the development of digital new technologies resulting in the generation of overwhelming data. Labeling data implies in general a high cost or cannot be done for all individuals. Even though this information is not provided for all data points, implicit structure can be used to infer labeling in the whole dataset and improve the generalization ability of the learner. For instance, information like the statistical distribution of the points or the geometrical shape of the data can be exploited in order to compensate the lack of sufficient labeled data. Clustering approaches or graph based techniques can be applied to use these valuable information.

This chapter deals with two frameworks that use partially labeled and unlabeled data, respectively. The first problem consists in incorporating the information given by the topology of the unlabeled data into the final decision model while the second one consists on unraveling the internal structure of the data leading to the topology knowledge. Section 3.1 treats the case of semi-supervised learning where only a part of the data is labeled and the manifold assumption is used to help construct a decision function. Under this framework, efforts are done to obtain via a regularization path [Gass 07c] a model that will require fewer variables to be expressed. In the second part of this chapter, Section 3.4, a review of several methods that deal with unlabeled datasets is done. It is noticed that many of the algorithms are based on the construction of a neighborhood graph and their efficiency decreases if links between points that are not neighbors are included. A method to detect incorrect links in the neighborhood graph is proposed based also on the manifold assumption [Zapi 07].

3.1. Semi-Supervised Learning

Semi-supervised learning addresses the learning problem of a database containing a few number of labeled data (l) and a relative large amount of unlabeled data (u), that is, a total of $n = l + u$ data points is available. The goal of semi-supervised algorithms is to leverage the learning process by using unlabeled points to unravel the underlying structure of the dataset.

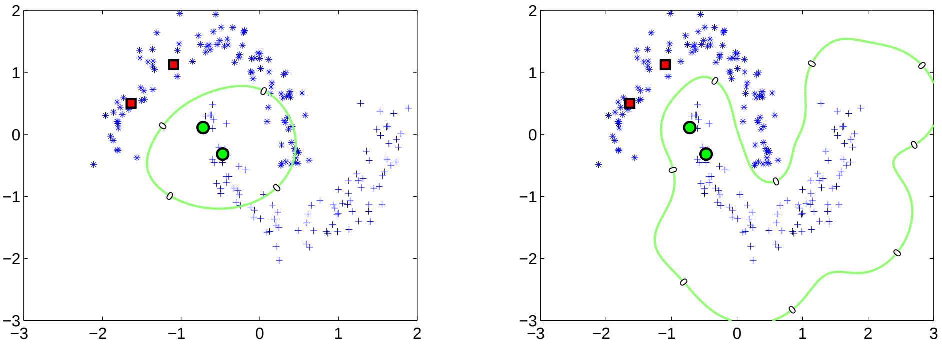
If we let $S_X = \{\mathbf{x}_i, i = [\![n]\!]\}$ be the set of all points, with $S_\ell = \{\mathbf{x}_i \in S_X, i = [\![l]\!]\}$ the set of labeled points with their corresponding labels $\mathcal{Y}_\ell = \{y_i \in \mathcal{Y} | i = [\![l]\!]\}$, a semi-supervised algorithm can be formalized as a problem with the following given sets [Chap 08]:

$$S_\ell \times \mathcal{Y}_\ell = \{(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}, \quad i = [\![l]\!]\} \qquad \text{(set of labeled points)}$$
$$S_u = \{\mathbf{x}_i \in \mathcal{X},\ i = l+1, \ldots, n,\ n = l+u\} \qquad \text{(set of unlabeled points)}$$

with $S_X = S_\ell \cup S_u$.

This situation can occur when the cost of giving a label to each point by an expert is high. An example is the problem of web pages or document classification. The arisen question is whether the knowledge of the labels of few points is enough to construct a decision function able to guess the correct class of the unlabeled data.

To show the interest of taking into account unlabeled points, example in Figure 3.1 will be considered. It can be seen that the obtained result by a usual SVM in Figure 3.1(a) fits perfectly according to the labeled data, nevertheless, the results are not satisfactory for the unlabeled points. Intuitively, if we exploit the form of the cloud of points around each label point, the form of the decision function could be improved. The interest of the semi-supervised machines consists in extracting such a kind of information and in integrating this information in the learning model. A decision function that would be closer to the one expected is depicted in Figure 3.1(b).



(a) Obtained solution by a SVM using only the labeled points.

(b) Obtained solution using additionally the unlabeled points.

Figure 3.1: Illustration of the utility of unlabeled points. There are only two labeled points per class (red squares and green circles) The points in blue represent unlabeled points. The decision function is depicted in light green.

Different approaches have been proposed and a good review can be found in [Chap 06a, Zhu 05a, Chap 08]. The overall spirit of these approaches is the smoothness assumption which states that if two points are close or similar in a certain sense, they should share the same label. Say=id in a different way, the labels of the data vary in a smooth way according to the similarity of these data. Two general frameworks to implement this smoothness hypothesis can be mentioned:

Cluster assumption: under this assumption, the aim is to avoid changes of labels in dense regions. An usual example to illustrate this assumption is to consider for instance that the data of each class (in a binary semi-supervised learning problem) are issued from different clusters. In this context, a reasonable hypothesis is to seek the decision function such that the frontier between the classes does not go across the clusters where typically we will encounter high data density. Some of the learning techniques that rely from this cluster assumption are presented hereafter.

Semi-Supervised Support Vector Machines (S^{3}VM): [Vapn 77, Chap 08] aims at optimizing *both* the decision function and the estimation of the unknown labels denoted

$\hat{\mathbf{y}}_u = (\hat{y}_{l+1}, \ldots, \hat{y}_n)^\top$:

$$\min_{f, \hat{\mathbf{y}}_u} \quad \Omega(f) + A \sum_{i=1}^{l} \mathcal{L}_1(f, \mathbf{x}_i, \mathbf{y}_i) + B \sum_{i=l+1}^{n} \mathcal{L}_2(f, \mathbf{x}_i, \hat{\mathbf{y}}_i) \tag{3.1}$$

the first two terms in the previous objective function define a standard SVM with Ω a regularization function and a loss function $\mathcal{L}_1$, while the third term incorporates unlabeled data, also used with a loss function $\mathcal{L}_2$. The loss over labeled and unlabeled examples is weighted by two fixed parameters A and B, which reflect confidence in labels and in the cluster assumption respectively. The most commonly used loss function $\mathcal{L}$ is the hinge loss but other losses can be considered as well as illustrated in [Chap 08]. The minimization problem in Equation (3.1) is solved under the class balancing constraint:

$$\frac{1}{u} \sum_{i=l+1}^{n} \max(y_i, 0) = r \qquad \text{or equivalently} \qquad \frac{1}{u} \sum_{i=l+1}^{n} y_i = 2r - 1,$$

where the fraction r is a user-defined parameter. The main interest of this constraint is to alleviate one drawback of the optimization problem in Equation (3.1) that consists in the arbitrary assignment of all unlabeled data to one class. The class balancing constraint permits to avoid this trivial solution by enforcing a fraction r of the unlabeled data to belong to the positive class.

A good overview of the two main categories of algorithms for solving problem in Equation (3.1) is presented in [Chap 08].

The first is the *Combinatorial Optimization*, consisting in iteratively solving an optimization problem for f with fixed $\hat{\mathbf{y}}_u$ (that will turn to be an usual SVM problem) and $\hat{f}$ being known, optimizing for $\mathbf{y}_u$ over a set of binary variables. One way to adress the optimization for $\mathbf{y}_u$ is the branch and bound algorithm [Benn 98, Chap 07b] able to find the global minimum but at a high computational cost. An alternative algorithm was proposed by Joachims [Joac 99b] with the S^{3}VM^{light}. The main idea consists in switching the labels of a pair of unlabeled data such that a decrease of the objective function occurs. With the new labelling, a SVM is trained for f and the procedure is repeated until convergence. Another approach considers the optimization problem in the dual space and converts it into a semi-definite programming [Bie 06] using convex relaxation. Finally, we should mention the deterministic annealing [Sind 06] procedure where to avoid the optimization over binary variables, the optimization is carried over the probability of an unlabeled data to belong to each. Doing so, analytic expression of these probabilies could be obtained or gradient based algorithm could be applied to the SVM semi-supervised learning.

The second category of S^{3}VM solvers is the *Continuous Optimization* approach and consists in eliminating the variables $\mathbf{y}_u$ in Problem 3.1 by assigning the optimal $\hat{\mathbf{y}}_u$ to be an application of a fixed f to the unlabeled data S_u ($\hat{y}_j = \operatorname{sign} f(\mathbf{x}_j), j \in S_u$). However, the resulting optimization problem is non-convex and has been solved directly in the primal or in the dual [Chap 05, Chap 06b, Coll 06, Fung 01].

Generative models: have been also proposed to do semi-supervised learning. The key point is to model the noise given by the unlabeled data [Lawr 05]. A generative model can be yielded for semi-supervised learning and the model includes a conditional density $\mathbb{P}(\mathbf{x}|y)$ part and a marginal distribution $\mathbb{P}(\mathbf{x})$ part to find the clusters [Fuji 05]. The parameters of the generative model are learned based on the EM algorithm (see Appendix A.1.6).

Entropy minimization: uses the prior which prefers minimal class overlap by using the label entropy on unlabeled data as regularizer [Gran 05, Zhu 05b].

Low density separation: as previously stated, a corollary of the cluster assumption is to force the decision function to go through low density regions. The S^{3}VM algorithm implements this idea by pushing unlabeled data apart from the decision function, but the low density desirable feature can also be improved by feeding the learning algorithm with a kernel enforcing this property. For instance, the authors in [Chap 05] have proposed a kernel based on low density sensitive distance and this kernel was further used for training (a SVM on only labeled data or for S^{3}VM) so that labeled and unlabeled data can be clustered and two points that are *close* share the same label. In the same spirit, Weston et al. [West 04] have proposed a cluster kernel based on the rescale of the original kernel by another crafted kernel representing the empirical probability of two points to belong to the same cluster.

Manifold assumption: under this assumption, the data are supposed to lie on a low dimensional manifold (this is the case for instance digits classification problem). Therefore a manifold based metric can be used to measure the similarity of the points. More precisely, a graph is used to measure the similarity between the points (labeled and unlabeled ones) [Bous 04, Chap 05, Belk 06]. The idea behind this is the same as the cluster assumption because the decision function should avoid to cut the manifolds where the points lie in majority. The basis of the related methods is the introduction of an additional regularization term that assumes label smoothness over the graph. Most of these algorithms are essentially discriminative and transductive (they aim to find directly the labels of the unlabeled points instead of the expression of the decision function). Some of them are summarized below.

Quadratic loss and Laplacian regularization for Transductive classification: a bunch of methods were proposed to minimize an objective function having the form [Wu 07]:

$$\min_{\mathbf{f}\in\mathbb{R}^n} \mathbf{f}^\top \mathbf{R}\mathbf{f} + (\mathbf{f}-\mathbf{y})^\top \mathbf{A}(\mathbf{f}-\mathbf{y}) \tag{3.2}$$

where $\mathbf{f} = [f(\mathbf{x}_1) \cdots, f(\mathbf{x}_l), f(\mathbf{x}_{l+1}) \cdots, f(\mathbf{x}_n)]$. The first term of Equation (3.2) refers to the graph regularization that enforces smoothness along the manifold. Matrix $\mathbf{R} \in \mathbb{R}^{n\times n}$ is generally named ***Laplacian*** $\boldsymbol{\mathcal{L}}$ or ***normalized Laplacian*** $\boldsymbol{\mathcal{L}}_n$ of the neighborhood graph defining the similarity between the labeled and unlabeled data, more details about these operators are given later. The normalized graph Laplacian is preferable to $\boldsymbol{\mathcal{L}}$, since the $\boldsymbol{\mathcal{L}}_n$ spectrum is contained in $[0, 2]$. The details of the Laplacian of a graph are given in Section 3.2. The second term of (3.2) is the loss function where $\mathbf{A}$ is a diagonal matrix with entries $\mathbf{A}_i = A$ for labeled points and $\mathbf{A}_i = B$ for unlabeled ones. The target output is $\mathbf{y} = [y_1 \cdots, y_l, 0 \cdots, 0]$ with y_i the label. From this unified view, different algorithms were derived according to the particular choice of $\mathbf{R}$ and $\mathbf{A}$. We can distinguish:

Gaussian Random Fields and harmonic function: it is a continuous relaxation of the difficult case discrete Markov random fields [Zhu 03]. This method corresponds to a choice $\mathbf{R} = \boldsymbol{\mathcal{L}}$, $A = \infty$ and $B = 0$. The values of $f(\mathbf{x}_i)$ are constrained to fit exactly y_i for the labeled data and are free for the unlabeled ones.

Tikhonov Regularization: this algorithm [Belk 04] uses $\mathbf{R} = \boldsymbol{\mathcal{L}}$ or $\boldsymbol{\mathcal{L}}^p$ for some integer p, $0 < C < \infty$ and $B = 0$. Compared to Gaussian Random Fields, this algorithm relaxes the constraint on $f(\mathbf{x}_i), i = [\![l]\!]$.

Label propagation: [Zhou 04] this algorithm is similar to the previous one except the fact that the normalized Laplacian is used and mainly $A = B$. Because of this last setting, all unlabeled and labeled points which are similar are forced to have the same label at the same time. Hence, there is a label-propagation effect from neighbors to neighbors.

Spectral Graph Transducer: [Joac 03] in this algorithm, the target output is differently encoded. Indeed, it considers $y_i = \sqrt{l_-/l_+}$ for positively labeled data and $\mathbf{y}_i = -\sqrt{l_+/l_-}$ for negatively labeled data, where l_+ is the number of positive data and l_- the number of negatively labeled data. For the unlabeled data, y_i is set to 0. The misclassification cost matrix $\mathbf{A}$ is freely set. Because of its origin from graph ratiocut optimization, additionnal constraints are enforced in the form $\mathbf{f}^\top \mathbb{I} = \mathbf{0}$ and $\mathbf{f}^\top \mathbf{f} = n$.

Graph Kernels from the Spectrum of the Laplacian: [Chap 03, Smol 03] it has been shown that the spectral transformation of a Laplacian $\boldsymbol{\mathcal{L}}$ results in kernels suitable for semi-supervised learning, this induces a semi-norm on f which penalizes the changes between adjacent vertices [Smol 03]. A class of regularization functionals on graphs can be defined as:

$$\langle f, r(\boldsymbol{\mathcal{L}}_n) f \rangle.$$

where $\boldsymbol{\mathcal{L}}_n$ is the normalized Laplacian, f a decision function and $r(\boldsymbol{\mathcal{L}}_n)$ is understood as applying a scalar valued function $r(\lambda)$ to the eigenvalues of $\boldsymbol{\mathcal{L}}_n$, that is,

$$r(\boldsymbol{\mathcal{L}}_n) = \sum_{i=1}^{m} r(\lambda_i) \boldsymbol{\nu}_i \boldsymbol{\nu}_i^\top,$$

where $\{\lambda_i, \boldsymbol{\nu}_i\}$ constitutes the eigensystem of $\boldsymbol{\mathcal{L}}_n$. There are some choices of $r(\lambda)$ of interest:

- The ***diffusion kernel*** corresponds to a spectrum transform of the Laplacian with $r(\lambda) = \exp\left(\frac{\sigma^2}{2}\lambda\right)$.
- The regularized Gaussian process kernel corresponds to $r(\lambda) = \frac{1}{\lambda + \sigma}$, where σ is a given parameter.
- Regularized Laplacian: $r(\lambda) = 1 + \sigma^2 \lambda$, where σ is a given parameter.
- One-Step Random Walk: $r(\lambda) = (AI - \lambda)^{-1}$, with $a \geq 2$.
- p-Step Random Walk: $r(\lambda) = (AI - \lambda)^{-p}$, with $a \geq 2$.
- Inverse Cosine: $r(\lambda) = (\cos(\lambda\pi/4)^{-1}$.

Local learning Regularization: a local linear classifier $o_i(\mathbf{x})$ is built from the nearest neighbors $\mathbf{x}_j$ of $\mathbf{x}_i$ with labels $f(\mathbf{x}_j)$. The Laplacian regularizer of Equation (3.2) is therefore replaced by the cost of the agreement between $f(\mathbf{x}_i)$ and the prediction value $o_i(\mathbf{x}_i)$ of $\mathbf{x}_i$ using this local classifier. The main idea is based in the fact that it can be shown [Wu 07] that the regularization term is also quadratic with matrix $\mathbf{R} = (I - A)^\top (I - A)$ where A is the transfer matrix that relates $\mathbf{f}$ to $\mathbf{O}$ (the vector formed by the output $o_i(\mathbf{x}_i)$ yielded by each local linear classifier) by $\mathbf{O} = A\mathbf{f}$.

Laplacian SVM: in this case [Belk 06], the loss function is the classical SVM hinge-loss defined over the labeled data. Beside it, two regularization terms are used, the first one being the classical quadratic regularizer of SVM and the second one, a aforementioned penalty preserving the smoothness of the decision function along the manifolds and based on the similarity graph.

In the remainder of this part of the chapter, our attention will be focused in the Laplacian SVM. Using the framework of the reproducing kernel Hilbert space, a corresponding representer theorem was derived expressing the decision function as a linear combination of the kernel function evaluated on all data points (labeled and unlabeled ones) [Belk 06].
The Laplacian SVM algorithm involves some difficulties: the construction of the similarity graph (problem common to all graph based method and the choice of the two regularization parameters (the parameters associated to the L_2-norm regularization and the manifold regularization). Wang et al. have addressed the later problem [Wang 06a] using a regularization path on one of the parameters. But the decision function still depends on all data, even though there might be several points that does not contribute to the information. To circumvent this drawback and to obtain a sparse solution to deal with large scale learning, Sindwhani et al. [Sind 05b] have introduced the linear Laplacian SVM. Aiming to induce sparsity in the obtained solution, it is proposed [Gass 07c]

to include in the semi-supervised classification problem a L_1-norm regularization following the idea pursued in [Wang 06b] as an extension to the original framework. The goal here is to express the classifier in function of few data points (labeled or not) rather than the solution with all points given by the original Laplacian SVM algorithm. To examine the evolution of the decision function in relation to this latest regularization, we compute a L_1-norm regularization path. According to [Hast 04, Wang 06b], it can be shown that the solution paths are piecewise linear and can be computed in a smart and efficient way.

In the sequel, the Laplacian SVM algorithm is described, then, the L_1-norm is introduced within the Laplacian SVM framework, followed by the description of the regularization path.

3.2. Laplacian SVM

The aim is to built a SVM classifier that exploits the information given by the unlabeled data. The framework of the manifold regularization considers that if two points are close in the intrinsic geometry of the marginal distribution $\mathbb{P}_X$, they share the same conditional density [Belk 06] i.e. if $\mathbf{x}_i \sim \mathbf{x}_j$ along the manifold $\mathcal{M}$, then $f(\mathbf{x}_i) \sim f(\mathbf{x}_j)$, where f is the decision function. This means that the labels vary smoothly along a submanifold the data is lying on, so that it can be seen as a traditional SVM classification problem with an additional penalization on the regular variation of the decision function f. The corresponding optimization problem is:

$$(\hat{f}_0, \hat{b}) = \underset{f_0 \in \mathcal{H}, b \in \mathbb{R}}{\operatorname{argmin}} \sum_{i=1}^{l} \mathcal{L}(f, \mathbf{x}_i, y_i) + \frac{\lambda}{2}\|f_0\|^2_{\mathcal{H}} + \frac{\mu}{2}\|f\|^2_{\mathscr{M}}$$

where $\mathcal{L}(\cdot,\cdot,\cdot)$ is a loss function, λ, μ are the regularization parameters for the regularization term and the manifold condition, respectively and $f(\mathbf{x}) = f_0(\mathbf{x}) + b$. The first penalty term is the classical L_2-norm regularizer in a Hilbert space whereas the second penalty is related to the smoothness along manifold $\mathscr{M}$. An approximation of this last term can be computed [Belk 06] as:

$$\|f\|^2_{\mathscr{M}} = \mathbf{f}^T \boldsymbol{\mathcal{L}} \mathbf{f}, \tag{3.3}$$

with $\mathbf{f} = [\, f(\mathbf{x}_1)\ f(\mathbf{x}_2)\ \ldots\ f(\mathbf{x}_{l+u})\,]$ and $\boldsymbol{\mathcal{L}}$ the Laplacian of the neighborhood graph which vertices are the $l+u$ points and edges obey the nearest neighbors rule ($\mathbf{x}_i$ is connected to its k-NN or $\mathbf{x}_i$ is connected to ε-close points, see Section 3.4). In general, the most commonly used distance to measure proximity is the Euclidean distance, but any other metric more adapted to the nature of data $\mathbf{x}_i$ can be used. A weight W_{ij} is associated to each edge between $\mathbf{x}_i$ and $\mathbf{x}_j$. Let matrix W be the corresponding weighted adjacency matrix with weights W_{ij} and D a diagonal matrix where all its diagonal elements are $D_{ii} = \sum_j W_{ij}$. The Laplacian is then defined as:

$$\boldsymbol{\mathcal{L}} = D - W \tag{3.4}$$

A normalized variant of the Laplacian can be computed by $\boldsymbol{\mathcal{L}} = (I - D^{-1}W)$.

To illustrate the effect of the Laplacian regularizer, let consider the following example. Suppose that nearest neighbors are connected in the neighborhood graph and the adjacency matrix W is defined as $W_{ij} = 1$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are nearest neighbors ($\mathbf{x}_i \sim \mathbf{x}_j$) and zero otherwise ($\mathbf{x}_i \nsim \mathbf{x}_j$). This is illustrated in Figure 3.2, where neighbors points are depicted with an edge ($W_{ij} = 1$), and the absence of edge means that the two points are not neighbors ($W_{ij} = 0$). Therefore, the regularity constraints take the form

$$\begin{aligned}
\sum_{\substack{i,j:\\ \mathbf{x}_i\sim\mathbf{x}_j}} (f(\mathbf{x}_i)-f(\mathbf{x}_j))^2 &= \sum_{\substack{i,j:\\ \mathbf{x}_i\sim\mathbf{x}_j}} f(\mathbf{x}_i)^2 - 2\sum_{\substack{i,j:\\ \mathbf{x}_i\sim\mathbf{x}_j}} f(\mathbf{x}_i)f(\mathbf{x}_j) + \sum_{\substack{i,j:\\ \mathbf{x}_i\sim\mathbf{x}_j}} f(\mathbf{x}_j)^2 \\
&= \sum_{i=1}^{n} f(\mathbf{x}_i)^2 D_{ii} - 2\sum_{\substack{i,j:\\ \mathbf{x}_i\sim\mathbf{x}_j}} f(\mathbf{x}_i)f(\mathbf{x}_j) + \sum_{j=1}^{n} f(\mathbf{x}_j)^2 D_{jj} \\
&= 2\sum_{i=1}^{n} f(\mathbf{x}_i)^2 D_{ii} - 2\sum_{\substack{i,j:\\ \mathbf{x}_i\sim\mathbf{x}_j}} f(\mathbf{x}_i)f(\mathbf{x}_j) \\
&= 2\sum_{i=1}^{n} f(\mathbf{x}_i)^2 D_{ii} - 2\sum_{i=1}^{n}\sum_{j=1}^{n} f(\mathbf{x}_i)f(\mathbf{x}_j)W_{ij} \\
&= 2\mathbf{f}^\top D\mathbf{f} - 2\mathbf{f}^\top W\mathbf{f} \\
&= 2\mathbf{f}^\top (D-W)\mathbf{f} \\
&= 2\mathbf{f}^\top \boldsymbol{\mathcal{L}}\mathbf{f}.
\end{aligned}$$

with $D_{ii} = \sum_j W_{ij}$ and $\mathbf{f} = [\, f(\mathbf{x}_1)\ f(\mathbf{x}_2)\ \ldots\ f(\mathbf{x}_{l+u})\,]$ as before, obtaining a direct relation with the graph Laplacian $\boldsymbol{\mathcal{L}}$.
As an example, the change along the graph in Figure 3.2 can be measured with

$$W = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \quad \text{and} \quad D = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

and

$$\begin{aligned}
\sum_{i,j:\mathbf{x}_i\sim\mathbf{x}_j} (f(\mathbf{x}_i)-f(\mathbf{x}_j))^2 &= (f(\mathbf{x}_1)-f(\mathbf{x}_2))^2 + (f(\mathbf{x}_1)-f(\mathbf{x}_4))^2 \\
&\quad + (f(\mathbf{x}_2)-f(\mathbf{x}_3))^2 + (f(\mathbf{x}_2)-f(\mathbf{x}_4))^2 + (f(\mathbf{x}_3)-f(\mathbf{x}_4))^2 \\
&= 2f(\mathbf{x}_1)^2 + 3f(\mathbf{x}_2)^2 + 2f(\mathbf{x}_3)^2 + 3f(\mathbf{x}_4)^2 - 2f(\mathbf{x}_1)f(\mathbf{x}_2) \\
&\quad -2f(\mathbf{x}_1)f(\mathbf{x}_4) - 2f(\mathbf{x}_2)f(\mathbf{x}_3) - 2f(\mathbf{x}_2)f(\mathbf{x}_4) - 2f(\mathbf{x}_3)f(\mathbf{x}_4) \\
&= [\, f(\mathbf{x}_1)\ f(\mathbf{x}_2)\ f(\mathbf{x}_3)\ f(\mathbf{x}_4)\,]\,(D-W)\,[\, f(\mathbf{x}_1)\ f(\mathbf{x}_2)\ f(\mathbf{x}_3)\ f(\mathbf{x}_4)\,]^\top
\end{aligned}$$

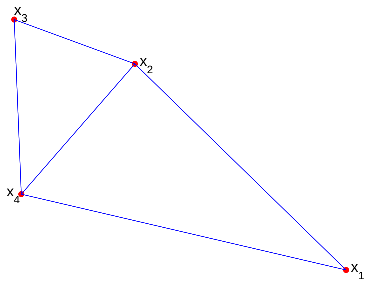


Figure 3.2: Induced graph by an adjacency matrix.

3.2.1 Formulation of the Laplacian SVM

It is considered that the decision function $f(\mathbf{x})$ belongs to a RKHS, and including the later definitions in the SVM framework, Belkin et al. [Belk 06] express the Laplacian SVM as:

$$\begin{array}{ll} \underset{f,\xi}{\text{minimize}} & \sum_{i=1}^{l} \xi_i + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2 + \frac{\mu}{2}\|f\|_{\mathscr{M}}^2 \\ \text{subject to} & y_i f(\mathbf{x}_i) \geq 1 - \xi_i, \quad i = [\![l]\!] \\ & \xi_i \geq 0, \qquad i = [\![l]\!]. \end{array} \tag{3.5}$$

Indeed, the extension of the Reproducing Kernel Hilbert Space (RKHS) theory to the manifold regularization brings them to propose a corresponding representer theorem which shows that the solution of problem in Equations (3.5) depends on all $(l+u)$ points as follows:

$$f(\mathbf{x}) = \sum_{j=1}^{l+u} \beta_j \mathbf{k}(\mathbf{x}, \mathbf{x}_j) + b \tag{3.6}$$

with $\mathbf{k}(\cdot,\cdot)$ the reproducing kernel. Using this result and letting $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)^\top$, the previous optimization problem can be rewritten in terms of the new parameters:

$$\underset{f,\xi}{\text{minimize}} \quad \sum_{i=1}^{l} \xi_i + \frac{\lambda}{2}\boldsymbol{\beta}^\top K \boldsymbol{\beta} + \frac{\mu}{2}\boldsymbol{\beta}^\top K \boldsymbol{\mathcal{L}} K \boldsymbol{\beta} \tag{3.7a}$$

$$\text{subject to} \quad y_i \left(\boldsymbol{\beta}^\top \mathbf{k}(\mathbf{x}_i) + b\right) \geq 1 - \xi_i, \quad i = [\![l]\!] \tag{3.7b}$$

$$\xi_i \geq 0, \qquad i = [\![l]\!] \tag{3.7c}$$

where $\mathbf{k}(\mathbf{x}_i) = (\mathbf{k}(\mathbf{x}_i, \mathbf{x}_1), \ldots, \mathbf{k}(\mathbf{x}_i, \mathbf{x}_{l+u}))^\top$ and $K \in \mathbb{R}^{(l+u)\times(l+u)}$ is the kernel matrix. It should be noticed that $\mathbf{f} = K\boldsymbol{\beta}$. If we note the Lagrange parameters for constraints (3.7b) and constraints (3.7c) as α_i and γ_i, $i = [\![l]\!]$ respectively, then, the Lagrangian $\mathbf{L}$ is defined as:

$$\mathbf{L} \;=\; \sum_{i=1}^{l} \xi_i + \frac{\lambda}{2}\boldsymbol{\beta}^\top K \boldsymbol{\beta} + \frac{\mu}{2}\boldsymbol{\beta}^\top K \boldsymbol{\mathcal{L}} K \boldsymbol{\beta} - \sum_{i=1}^{l} \alpha_i \left(y_i \boldsymbol{\beta}^\top \mathbf{k}(\mathbf{x}_i) + y_i b - 1 + \xi_i\right) - \sum_{i=1}^{l} \gamma_i \xi_i.$$

with $\alpha_i, \gamma_i \geq 0, i = [\![l]\!]$. The optimality conditions related to the primal variables yield:

$$\nabla_b \mathbf{L} = 0 \quad : \quad \boldsymbol{\alpha}^\top \mathbf{y} = 0 \tag{3.8}$$

$$\nabla_{\xi_i} \mathbf{L} = 0 \quad : \quad 1 - \alpha_i - \gamma_i = 0 \qquad \Rightarrow \qquad 0 \leq \alpha_i \leq 1 \tag{3.9}$$

$$\nabla_{\boldsymbol{\beta}} \mathbf{L} = 0 \quad : \quad \lambda K \boldsymbol{\beta} + \mu K \boldsymbol{\mathcal{L}} K \boldsymbol{\beta} - \sum_{i=1}^{l} \alpha_i y_i K_{i,\cdot}^\top = 0$$

where $\mathbf{y} = (y_1, y_2, \ldots, y_l)^\top$, $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_l)^\top$ and $K_{i,\cdot} = \mathbf{k}(\mathbf{x}_i)$ denotes the i-th row of the kernel matrix K. Defining $Y = \text{diag}(\mathbf{y}) \in \mathbb{R}^{l\times l}$, the parameters of the optimal decision function (the one that solves Equations (3.7)) are given by:

$$\hat{\boldsymbol{\beta}} = (\lambda I_{l+u} + \mu \boldsymbol{\mathcal{L}} K)^{-1} J^\top Y \hat{\boldsymbol{\alpha}} \tag{3.10}$$

with $J = \left[\begin{array}{cc} I_l & 0_{l\times u} \end{array}\right]$, I_l the identity matrix of size l and $0_{l\times u}$ a matrix full of zeros of size $l \times u$. Vector $\hat{\boldsymbol{\alpha}}$ is the solution of the dual problem involving only the labeled data:

$$\begin{array}{ll} \underset{\boldsymbol{\alpha}\in\mathbb{R}^l}{\text{maximize}} & -\frac{1}{2}\boldsymbol{\alpha}^\top Q \boldsymbol{\alpha} + \boldsymbol{\alpha}^\top \mathbf{1} \\ \text{subject to} & \boldsymbol{\alpha}^\top \mathbf{y} = 0, \quad i = [\![l]\!] \\ & 0 \leq \alpha_i \leq 1, \quad i = [\![l]\!] \end{array}$$

with $Q = YJK(\lambda I_{l+u} + \mu \boldsymbol{\mathcal{L}} K)^{-1} J^\top Y$.

Here two remarks can be made: first, the solution (3.10) requires to solve a linear system which has $\mathcal{O}((l+u)^3)$ complexity. This computation can be long especially for large scale database. The solution of the dual problem is sparse in the labeled points (that is, some elements of $\boldsymbol{\alpha}$ will be zero) but the parameter $\boldsymbol{\beta}$ still depends on all data inducing a non sparse solution in the variables of the model (if we consider the term $\mathbf{k}(\mathbf{x}, \mathbf{x}_j)$ in the expression of $f(\mathbf{x})$ as a variable).

We can remark also that due to the KKT conditions, we can make a partition of the labeled points in three sets as for a classical SVM:

$$\begin{aligned} \mathcal{I}_1 : &\quad y_i f(\mathbf{x}_i) < 1 \quad \text{and} \quad \alpha_i = 1, &&\text{bounded points} \quad (3.11)\\ \mathcal{I}_0 : &\quad y_i f(\mathbf{x}_i) > 1 \quad \text{and} \quad \alpha_i = 0, &&\text{useless points} \quad (3.12)\\ \mathcal{I}_\alpha : &\quad y_i f(\mathbf{x}_i) = 1 \quad \text{and} \quad 0 \le \alpha_i \le 1, &&\text{margin points} \quad (3.13) \end{aligned}$$

so that $S_\ell = \mathcal{I}_1 \cup \mathcal{I}_0 \cup \mathcal{I}_\alpha$. Support vectors will lie in sets $\mathcal{I}_1$ (misclassified points) and $\mathcal{I}_\alpha$ (points that are exactly on the margin). Points in $\mathcal{I}_0$ are well classified points that become useless for the construction of the decision function. The solution (3.10) is entirely determined by the knowledge of these sets which will be exploited in the derivation of the solution path in the next section.

3.3. L1-norm Laplacian SVM

In general vector $\boldsymbol{\beta}$ obtained in Equation (3.10) is not sparse. In this case the classification function relies on all the learning points (labeled or unlabeled). To obtain sparsity, we introduce an additional L_1-norm constraint to the problem. We formulate the L_1-norm Laplacian SVM as:

$$\begin{aligned} \underset{\boldsymbol{\beta}, b, \xi}{\text{minimize}} \quad & \sum_{i=1}^{l} \xi_i + \frac{\lambda}{2} \boldsymbol{\beta}^\top K \boldsymbol{\beta} + \frac{\mu}{2} \boldsymbol{\beta}^\top K \boldsymbol{\mathcal{L}} K \boldsymbol{\beta} \\ \text{subject to} \quad & y_i(\boldsymbol{\beta}^\top \mathbf{k}(\mathbf{x}_i) + b) \ge 1 - \xi_i, \quad i = [\![l]\!] \\ & \xi_i \ge 0, \quad i = [\![l]\!] \\ & \sum_{j=1}^{l+u} |\beta_j| \le s \end{aligned} \qquad (3.14)$$

this form satisfies the conditions of Rosset's [Ross 07b] formulations and it has therefore a piecewise linear solution with respect to s.

If we consider the form of the decision function in Equation (3.6), this formulation can be seen as the expansion of $f(\mathbf{x})$ over a dictionary composed of a set $\mathcal{D} = \{h_1(\mathbf{x}), h_2(\mathbf{x}), \dots, h_{l+u}(\mathbf{x})\}$ of basis functions. These basis functions $h_j(\mathbf{x})$ are here the kernel functions defined in the point $\mathbf{x}_j$ i.e. $h_j(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}_j)$. If $s = \infty$, the Laplacian SVM solution will be obtained, while making the parameter s small tends to cause many coefficients $\boldsymbol{\beta}_j$ to be null. The aim in this section is to analyze the degree of the sparsity when s varies.

The objective is to use an iterative algorithm that will give the set of all solutions for all $0 \le s \le \infty$ with the same computational cost as one single problem resolution: the regularization path. We are not trying to fix some values for s, solve the problem given by in Equations (3.14) for each value and then compare the degree of the sparsity obtained for each solution. It will be shown that if at step t, solution f^t is the optimal one, the solution for a *close* value s' to s will be deduced from the last one with a simple linear relation. As in previous sections, this linear relation will hold until the previously defined sets remain unchanged. The details of the regularization path is analyzed hereafter.

To deduce the regularization path, the Lagrangian associated to (3.14) is needed. It is given by:

$$\begin{aligned}
\mathbf{L} &= \sum_{i=1}^{l} \xi_i + \frac{\lambda}{2}\boldsymbol{\beta}^\top K \boldsymbol{\beta} + \frac{\mu}{2}\boldsymbol{\beta}^\top K \boldsymbol{\mathcal{L}} K \boldsymbol{\beta} \\
&\quad - \sum_{i=1}^{l} \alpha_i \left(y_i(\boldsymbol{\beta}^\top \mathbf{k}(\mathbf{x}_i) + b) - 1 + \xi_i \right) - \sum_{i=1}^{l} \gamma_i \xi_i \\
&\quad + \zeta \left(\sum_{j=1}^{l+u} |\beta_j| - s \right) \\
\text{with} \quad & \zeta \geq 0 \\
& \alpha_i \geq 0, \quad i = [\![l]\!] \\
& \gamma_i \geq 0, \quad i = [\![l]\!]
\end{aligned}$$

The optimality conditions related to the primal variables b and ξ_i of the previous section remain unchanged (see Equations (3.8) and (3.9)), but the derivation of the optimality condition for $\boldsymbol{\beta}$ is slightly different due to the non-differentiability of the ℓ_1-norm. Indeed, let $\mathcal{A} = \{j | \beta_j \neq 0\}$ be the set of ***active variables*** and $\bar{\mathcal{A}} = \{j | \beta_j = 0\}$ be its complement in $\mathcal{D}$. Remark that, due to the definition of $\mathcal{D}$, $\mathcal{A}$ coincides with the set of useful points (labeled or not). To facilitate notation, let

$$P = \lambda K + \mu K \boldsymbol{\mathcal{L}} K.$$

It can be noticed that $f(\mathbf{x}_j) = K_{j,\mathcal{A}} \boldsymbol{\beta}_{\mathcal{A}}$, for $j \in S_\ell \cup S_u$, with $K_{j,\mathcal{A}} = \mathbf{k}(\mathbf{x}_j)_{\mathcal{A}}$ being the vector formed with the j-th column of K and the corresponding rows to set $\mathcal{A}$, while $\boldsymbol{\beta}_{\mathcal{A}}$ is the sub-vector of $\boldsymbol{\beta}$ corresponding to the positions of set $\mathcal{A}$. The optimality conditions according to $\boldsymbol{\beta}$ could be derived directly using the notion of subdifferential, leading to $0 \in P\boldsymbol{\beta} - \sum_{i=1}^{l} \alpha_i y_i \mathbf{k}(\mathbf{x}_i) + \zeta \mathbf{g}$ where $\mathbf{g}$ is the vector of subdifferentials with entries $g_i = \operatorname{sign} \beta_i$ if $\beta_i \neq 0$ and $-1 \leq g_i \leq 1$ if $\beta_i = 0$. Using this optimality condition the path derivation analysis can be carried on as well. Therefore, the optimality condition with respect to $\boldsymbol{\beta}_{\mathcal{A}}$ yields

$$P_{\mathcal{A},\mathcal{A}} \boldsymbol{\beta}_{\mathcal{A}} - \sum_{i=1}^{l} \alpha_i y_i K_{i,\mathcal{A}}^\top + \zeta \operatorname{sign}(\boldsymbol{\beta}_{\mathcal{A}}) = 0$$

where $P_{\mathcal{A},\mathcal{A}}$ is the submatrix composed by the corresponding $\mathcal{A}$ columns and rows of P.
We have that this equation is equivalent to

$$P_{\mathcal{A},\mathcal{A}} \boldsymbol{\beta}_{\mathcal{A}} - K_{\mathcal{A},S_\ell} Y_{S_\ell,S_\ell} \boldsymbol{\alpha} + \zeta \operatorname{sign}(\boldsymbol{\beta}_{\mathcal{A}}) = 0 \tag{3.15}$$

At an optimal point, the KKT conditions for the points in $\mathcal{I}_\alpha$ become equality conditions, resulting in:

$$\begin{aligned}
y_i \left(K_{i,\mathcal{A}} \boldsymbol{\beta}_{\mathcal{A}} + b \right) &= 1, \qquad \forall i \in \mathcal{I}_\alpha \qquad (3.16) \\
\zeta \left(\sum_{j \in \mathcal{A}} |\beta_j| - s \right) &= \zeta \left(\operatorname{sign}(\boldsymbol{\beta}_{\mathcal{A}})^\top \boldsymbol{\beta}_{\mathcal{A}} - s \right) = 0
\end{aligned}$$

From this formulation, the L_1-norm regularization path is deduced. If λ and μ are fixed, the s-path (or equivalently the ζ-path) will provide the evolution of the form of f (that is, of $\boldsymbol{\beta}$, b and $\boldsymbol{\alpha}$) according to s. Remark that, conversely, fixing ζ and μ, we can analyze the variation of f with relation to λ giving the λ-path.

3.3.1 The sparsity regularization path

The problem in Equations (3.14) is a convex one, ans as seen in Section 1.1.5, it can be shown that its solution coincides with the following equivalent problem:

$$\underset{\boldsymbol{\beta},b}{\text{minimize}} \quad \textstyle\sum_{i=1}^{\ell} \max(0, 1 - y_i(\mathbf{k}(\mathbf{x}_i)^\top \boldsymbol{\beta} + b)) + \frac{\lambda}{2}\boldsymbol{\beta}^\top (K + \frac{\mu}{\lambda} K \boldsymbol{\mathcal{L}} K)\boldsymbol{\beta} + \zeta \|\boldsymbol{\beta}\|_1$$

where $\|\boldsymbol{\beta}\|_1$ is the L_1-norm of vector $\boldsymbol{\beta}$. This last problem involves a hinge-loss function, a quadratic penalty and a piecewise linear penalty. As this problem satisfies the required conditions in Theorem 2.2, it also has a piecewise linear solution if λ is fixed and it has a regularization path along ζ. In a similar way, the regularization path for λ can be found if ζ is fixed.

Following the ideas developed by Wang et al. [Wang 06b], a sufficiently small s can be found at step t so that the L_1-norm constraint is active, that is,

$$\operatorname{sign}\left(\boldsymbol{\beta}^t_{\mathcal{A}}\right)^\top \boldsymbol{\beta}^t_{\mathcal{A}} = s^t, \tag{3.17}$$

corresponding to $\zeta^t = 0$, when we increase s^t by an infinitesimal quantity, the constraint is inactive and the solution remains the same unless a certain value s is reached. When s^t increases sets $\mathcal{A}$, $\mathcal{I}_\alpha$, $\mathcal{I}_1$ and $\mathcal{I}_0$ also remain unchanged. By continuity of the solutions (meaning that a wrongly classified point cannot be well classified unless it passes through the decision function border, or margin), the variations of $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, b and ζ can be expressed as a function of the variation of s via the relations (3.8, 3.15-3.17). If these are reformulated in matrix form, the following linear system is obtained:

$$\begin{aligned}
P_{\mathcal{A},\mathcal{A}} \Delta\boldsymbol{\beta}_{\mathcal{A}} - K_{\mathcal{A},\mathcal{I}_\alpha} Y_{\mathcal{I}_\alpha} \Delta\boldsymbol{\alpha}_{\mathcal{I}_\alpha} + \Delta\zeta \operatorname{sign}(\boldsymbol{\beta}_{\mathcal{A}}) &= 0 \\
\mathbf{y}^\top_{\mathcal{I}_\alpha} \Delta\boldsymbol{\alpha}_{\mathcal{I}_\alpha} &= 0 \\
Y_{\mathcal{I}_\alpha}\left(K_{\mathcal{I}_\alpha,\mathcal{A}} \Delta\boldsymbol{\beta}_{\mathcal{A}} + \mathbf{1}_{\mathcal{I}_\alpha} \Delta b\right) &= 0 \\
\operatorname{sign}(\boldsymbol{\beta}_{\mathcal{A}})^\top \Delta\boldsymbol{\beta}_{\mathcal{A}} &= (s - s^t)
\end{aligned}$$

where, in order to simplify the notation, $Y_{\mathcal{I}_\alpha}$ stands for $Y_{\mathcal{I}_\alpha,\mathcal{I}_\alpha}$, the diagonal matrix containing labels of points in $\mathcal{I}_\alpha$. Notice that in these equations, only the variations of the Lagrangian parameters in vector $\boldsymbol{\alpha}_{\mathcal{I}_\alpha}$ associated to the margin points $\mathcal{I}_\alpha$ are considered since the other parameters are fixed either to one or zero.

We have a linear system of $|\mathcal{A}| + |\mathcal{I}_\alpha| + 2$ equations and $|\mathcal{A}| + |\mathcal{I}_\alpha| + 2$ unknown parameters $\Delta\boldsymbol{\beta}_{\mathcal{A}} \in \mathbb{R}^{|\mathcal{A}|}$, $\Delta\boldsymbol{\alpha}_{\mathcal{I}_\alpha} \in \mathbb{R}^{|\mathcal{I}_\alpha|}$, Δb and $\Delta\zeta$. This system can be written in matrix form (after some algebra):

$$\begin{aligned}
H\Delta\boldsymbol{\theta} &= (s - s^t)\mathbf{z}, \quad \text{where} \\
\Delta\boldsymbol{\theta} &= \begin{bmatrix} \Delta\boldsymbol{\beta}_{\mathcal{A}} \\ \Delta\boldsymbol{\alpha}_{\mathcal{I}_\alpha} \\ \Delta b \\ \Delta\zeta \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} \mathbf{0}_{\mathcal{A}} \\ \mathbf{0}_{\mathcal{I}_\alpha} \\ 0 \\ 1 \end{bmatrix} \quad \text{and} \\
H &= \begin{bmatrix} P_{\mathcal{A},\mathcal{A}} & -K_{\mathcal{A},\mathcal{I}_\alpha} Y_{\mathcal{I}_\alpha} & \mathbf{0}_{\mathcal{A}} & \operatorname{sign}(\beta_{\mathcal{A}}) \\ -Y_{\mathcal{I}_\alpha} K_{\mathcal{I}_\alpha,\mathcal{A}} & \mathbf{0}_{\mathcal{I}_\alpha} & -\mathbf{y}_{\mathcal{I}_\alpha} & \mathbf{0}_{\mathcal{I}_\alpha} \\ \mathbf{0}^\top_{\mathcal{A}} & -\mathbf{y}^\top_{\mathcal{I}_\alpha} & 0 & 0 \\ \operatorname{sign}(\boldsymbol{\beta}_{\mathcal{A}})^\top & \mathbf{0}^\top_{\mathcal{I}_\alpha} & 0 & 0 \end{bmatrix}
\end{aligned} \tag{3.18}$$

letting $\boldsymbol{\eta} = H^{-1}\mathbf{z}$, it is straightforward to see that the evolution of the various parameters of the model are piecewise linear in s as we have:

$$\theta = \theta^t + (s - s^t)\boldsymbol{\eta}_\theta \tag{3.19}$$

where θ stands in a general way either for $\boldsymbol{\beta}_A$, $\boldsymbol{\alpha}_{\mathcal{I}_\alpha}$, b or ζ. This linear variation holds until one of the previous sets $\mathcal{A}$, $\mathcal{I}_\alpha$, $\mathcal{I}_1$ or $\mathcal{I}_0$ changes. The specific value of s at which this change occurs is determined by reviewing four types of events:

1. A labeled point in $\mathcal{I}_\alpha$ moves to $\mathcal{I}_0$ or $\mathcal{I}_1$.

2. A labeled point in $\mathcal{I}_0 \cup \mathcal{I}_1$ reaches $\mathcal{I}_\alpha$.
3. An active variable $\beta_j \neq 0$ in $\mathcal{A}$ becomes inactive.
4. An inactive variable $\beta_j = 0$ in $\bar{\mathcal{A}}$ becomes active and joins $\mathcal{A}$.
5. A termination criteria is satisfied.

The detection of these events is described in the next subsection.

3.3.2 The events and determination of next value s

1. A labeled point $\mathbf{x}_i \in \mathcal{I}_\alpha$ moves to $\mathcal{I}_1$ or $\mathcal{I}_0$.
   Looking at the definition of $\mathcal{I}_1$ and $\mathcal{I}_0$, this point movement implies that the Lagrange parameter associated to it achieves one of the limits values, that is $\alpha_i = 1$ or $\alpha_i = 0$. Thus, using the Equation (3.19) to update the parameters, the next value of s that will cause one of these events is:

   $$\text{if} \quad \mathbf{x}_i \in \mathcal{I}_\alpha \rightarrow \mathcal{I}_1 \quad \Rightarrow \quad s^{t+1} = s^t + \frac{1 - \alpha_i^t}{\boldsymbol{\eta}_{\alpha_i}}$$

   or

   $$\text{if} \quad \mathbf{x}_i \in \mathcal{I}_\alpha \rightarrow \mathcal{I}_0 \quad \Rightarrow \quad s^{t+1} = s^t + \frac{0 - \alpha_i^t}{\boldsymbol{\eta}_{\alpha_i}},$$

2. A labeled point $\mathbf{x}_i \in \mathcal{I}_1$ or $\mathbf{x}_i \in \mathcal{I}_0$ moves to $\mathcal{I}_\alpha$.
   This event happens when the residual $r_i = 1 - y_i f(x_i)$ turns zero. More explicitly, the residual for the non-margin points is $r_i = 1 - y_i \left(K_{i,\mathcal{A}} \boldsymbol{\beta}_\mathcal{A} + b\right)$, $\forall i \in \mathcal{I}_1 \cup \mathcal{I}_0$. Thus, we get the variation of the residual:

   $$\boldsymbol{\eta}_{r_i} = -y_i \left(K_{i,\mathcal{A}} \boldsymbol{\eta}_{\beta_\mathcal{A}} + \boldsymbol{\eta}_b\right), \; \forall i \in \mathcal{I}_1 \,\cup\, \mathcal{I}_0$$

   and deduce the value of s corresponding to this event

   $$s = s^t + \frac{0 - r_i^t}{\boldsymbol{\eta}_{r_i}}, \quad \forall i \in \mathcal{I}_1 \cup \mathcal{I}_0$$

3. An active variable $\beta_j \neq 0$, $j \in \mathcal{A}$ becomes inactive.
   This event means that the parameter associated to variable β_j turns zero and the following conditions are satisfied:

   $$s^{t+1} = s^t + \frac{0 - \beta_j^t}{\boldsymbol{\eta}_{\beta_j}}, \quad \forall j \in \mathcal{A}$$

4. An inactive variable $\beta_j = 0$, $j \in \bar{\mathcal{A}}$ becomes active.
   To track this event, we consider the optimality condition associated to β_m in Equation (3.15):

   $$P_{j,\cdot}\boldsymbol{\beta} - K_{j,S_\ell} Y \boldsymbol{\alpha} \quad = \quad -\zeta \operatorname{sign}(\beta_j) \quad \forall j \in \mathcal{A}$$

   This expression can be seen as the generalized correlation of the variable $k(\mathbf{x}, \mathbf{x}_j)$ and shows that for all active variables, the right term is equal to $-\zeta$ times the sign of the parameter (as an analogy to the LARS algorithm [Efro 04] and Chapter 2.5). Hence, for all variable (active or not), if the generalized correlation is defined as:

   $$c_j = P_{j,\cdot}\boldsymbol{\beta} - K_{j,S_\ell} Y \boldsymbol{\alpha}$$

which depends on the parameters of the model. Therefore, using the same arguments of the variation, it can be established that c_j varies linearly, so that at step t, knowing that only $\boldsymbol{\beta}_{\mathcal{A}}$ is active we have that for an inactive variable β_j, $c_j^t = P_{j,\mathcal{A}}\boldsymbol{\beta}_{\mathcal{A}} - K_{j,S_\ell}Y\boldsymbol{\alpha}$ and the variation of c_j is

$$\boldsymbol{\eta}_{c_j} = P_{j,\mathcal{A}}\ \boldsymbol{\eta}_{\mathcal{A}} - K_{j,\mathcal{I}_\alpha}Y_{\mathcal{I}_\alpha}\ \boldsymbol{\eta}_{\mathcal{I}_\alpha}$$

where for simplicity sake, the notations $\boldsymbol{\eta}_{\mathcal{A}}$ and $\boldsymbol{\eta}_{\mathcal{I}_\alpha}$ stand respectively for $\boldsymbol{\eta}_{\boldsymbol{\beta}_{\mathcal{A}}}$ and $\boldsymbol{\eta}_{\boldsymbol{\alpha}_{\mathcal{I}_\alpha}}$. If an inactive variable $\beta_j, j \in \mathcal{A}$ turns active, it must verify the generalized correlation constraint:

$$\begin{aligned} |c_j^{t+1}| &= |\zeta^{t+1}| \\ \text{where} \qquad c_j^{t+1} &= c_j^t + \boldsymbol{\eta}_{c_j}(s^{t+1} - s^t) \\ \text{and} \qquad \zeta^{t+1} &= \zeta^t + \boldsymbol{\eta}_\zeta(s^{t+1} - s^t). \end{aligned}$$

From the correlation constraint we deduce the next value of s^{t+1} that will possibly activate variable β_j:

$$s^{t+1} = s^t + \min\left(\frac{\zeta^t - c_j^t}{\boldsymbol{\eta}_{c_j} - \boldsymbol{\eta}_\zeta}, \frac{-\zeta^t - c_j^t}{\boldsymbol{\eta}_{c_j} + \boldsymbol{\eta}_\zeta}\right)_+ \qquad j \in \bar{\mathcal{A}}$$

with $(\mathbf{x})_+ = \max(0, \mathbf{x})$

5. The last event that can occur, which is one of the stopping criteria of the algorithm, is that the generalized correlation of active variables turns zero, that is, that parameter ζ turns zero. This will happen for the value:

$$s^{t+1} = s^t + \frac{0 - \zeta^k}{\boldsymbol{\eta}_\zeta}.$$

If the regularization parameter is swept by increasing s, given s^t, the next value of s will be the minimum of the previously enumerated s^{t+A} with value larger than s^t. The sets are modified according to the identified event.

3.3.3 Algorithm initialization

The initialization procedure is here briefly described. Two cases can be considered: the balanced case (where the number of points of the positive points and the number of negative points are equal, that is, $l_+ = l_-$) and the unbalanced case $l_+ \neq l_-$, letting l_+ and l_- be the number of positively and negatively labeled data points, respectively.

Balanced case ($l_+ = l_-$) When $s^t = 0$, all parameters β_j are zero ($\mathcal{A}$ is empty), $\boldsymbol{\beta} = 0$ then, the decision function is a straight line satisfying the equation $f(x) = b$. The solution b is not unique but it can be chosen so that all points belong to $\mathcal{I}_1$. This can be done by letting $\boldsymbol{\alpha} = \mathbf{1}_l$, a vector of ones of size l. The procedure consists in identifying the variables with maximum absolute generalized correlation $|c_j| = |K_{j,S_\ell}Y\boldsymbol{\alpha}|$ which will join $\mathcal{A}$. Since their sign is known, it can be solved for $\Delta\boldsymbol{\beta}_{\mathcal{A}}$ and $\Delta\zeta$ using Equation (3.18) from which we exclude the terms related to $\Delta\boldsymbol{\alpha}_{\mathcal{I}_\alpha}$ and Δb. Tiding up this information, we have

$$1 - (b + (s - s^t)K_{i,\mathcal{A}}\Delta\boldsymbol{\beta}_{\mathcal{A}}) \geq 0, \qquad \forall i, y_i > 0$$

and

$$1 + (b + (s - s^t)K_{i,\mathcal{A}}\Delta\boldsymbol{\beta}_{\mathcal{A}}) \geq 0, \qquad \forall i, y_i < 0.$$

Since all points are in $\mathcal{I}_1$ and $f(\mathbf{x}_i) = (s - s^t)K_{i,\mathcal{A}}\Delta\boldsymbol{\beta}_{\mathcal{A}}$, both inequalities involve b and s. These two last parameters are then determined so that at least one point in each class goes to $\mathcal{I}_\alpha$. It can be seen that the initial configuration is equivalent to set all points inside the margin or on the margin. Set $\mathcal{I}_0$ is empty at the initialization.

Unbalanced case ($l_+ \neq l_-$) To identify the different sets for this case, a Linear Programming problem is solved with a fixed value of s. We refer the reader to the paper of Wang et al. [Wang 06b] to get more insights.

3.3.4 The L_1 regularization path algorithm

The regularization path can be summarized by the pseudo-code of Algorithm 5.

Algorithm 5 Pseudo-code for the sparse Laplacian SVM regularization path

```
Input: training set $\{(\mathbf{x}_i, y_i)\},\ i = [\![l]\!]$, and $S_u = \{(\mathbf{x}_i)\},\ i = [\![u]\!]$
Output: Complete regularization path $\{(\alpha_s,\ s)\}$
  Find an initial solution for parameters $\boldsymbol{\beta}_\mathcal{A}$, $\boldsymbol{\alpha}$, $b$, $\zeta$ and its corresponding $s$.
  Initialize sets $\mathcal{A}$, $\mathcal{I}_\alpha$, $\mathcal{I}_1$ and $\mathcal{I}_0$ accordingly.
  Set $t = 1$.
  repeat
    Compute the direction of variation for the parameters: $\boldsymbol{\eta}_\mathcal{A}$, $\boldsymbol{\eta}_{\mathcal{I}_\alpha}$, $\boldsymbol{\eta}_b$ and $\boldsymbol{\eta}_\zeta$ using equation (3.18): $H\boldsymbol{\eta} = \mathbf{z}$.
    Deduce from these variations the residuals ones $\boldsymbol{\eta}_{r_j}$, $j \in \mathcal{I}_0 \cup \mathcal{I}_1$ and the ones of the generalized correlation $\boldsymbol{\eta}_{c_j}$, $j \in \bar{\mathcal{A}}$.
    Compute the next value of $s^{t+1}$ by detecting the following event to occur.
    Update the parameters of the model using (3.19).
    Update the sets $\mathcal{A}$, $\mathcal{I}_\alpha$, $\mathcal{I}_1$ and $\mathcal{I}_0$ according to the occurred event.
    Set $t = t + 1$.
  until no more events are found, $\zeta$ turns zero.
```

Other stopping criteria can be include like a minimum value for the parameters or a fix number of iterations.

3.3.5 Computational complexity

Beyond the computation cost of the neighborhood graph ($\mathcal{O}(l+u)^2$), the Laplacian $\boldsymbol{\mathcal{L}}$ (see Equation (3.4)) and the kernel matrix K, the sparse Laplacian solution path involves different costs.

The resolution of the linear system in Equation (3.18) requires a computational cost of $\mathcal{O}((|\mathcal{A}| + |\mathcal{I}_\alpha| + 2)^3)$ for the inversion of matrix H. This cost can be reduced to $\mathcal{O}((|\mathcal{A}| + |\mathcal{I}_\alpha| + 2)^2)$ using Sherman-Morrisson updating at a given step, the linear system differs from the previous by one row or column due to the update of the sets.

The computation of $\boldsymbol{\eta}_{r_i}$ and $\boldsymbol{\eta}_{c_j}$ costs respectively $\mathcal{O}(|\mathcal{A}|)$ and $\mathcal{O}(|\mathcal{A}| + |\mathcal{I}_\alpha|)$, therefore, the overall computation cost are respectively $\mathcal{O}(|\mathcal{A}| \times l)$ and $\mathcal{O}((l+u) \times (|\mathcal{A}| + |\mathcal{I}_\alpha|))$.

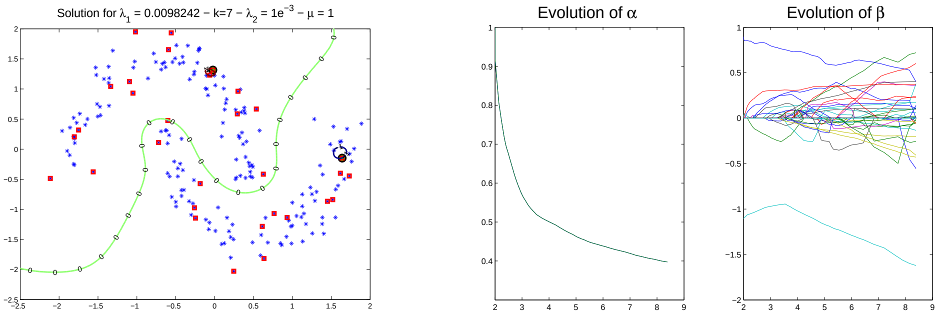
Finally, the detection of an event involves $\mathcal{O}(l + u)$ operations. From this analysis, the total complexity of a step of the algorithm is approximately $\mathcal{O}((|\mathcal{A}| + |\mathcal{I}_\alpha| + 2)^2 + (l+u) \times (|\mathcal{A}| + |\mathcal{I}_\alpha|))$.

It is very hard to predict the number of steps of the algorithm but it can be estimated like a small multiple of $l + u$ (it takes $l + u$ moves to examine all the $l + u$ variables). However, if a validation procedure is coupled with the algorithm, there is no need to explore entirely the regularization path. An early stop of the algorithm has to be performed in order to yield a sparse solution.

3.3.6 Applications of the algorithm

The algorithm is illustrated on a synthetic data: the classical two moons datasets (see Figure 3.3), and in real datasets from the UCI database.

For the toy dataset, each half-moon represents a class and is a non-linear separable case, and the dataset consists on 200 points. To evaluate the algorithm, the following procedure was followed: a given number l of labeled data was chosen and the algorithm was run over them and the other



(a) Resulting decision function at the end of the regularization path.

(b) Evolution of parameters α_i and β_j along the regularization path.

Figure 3.3: Illustration of the algorithm on the two moons dataset when only one labeled point per class is available. Blue points are unlabeled, red circles are labeled points and red squares are the active points in the optimal model with the used regularization parameter.

$200-\ell$ unlabeled points. A Gaussian kernel with σ as the bandwidth was used. The neighborhood graph is computed using k-NN technique with $k = 7$, which showed to give satisfactory results. At the beginning of the algorithm, there are 200 candidate variables $k(\mathbf{x}, \mathbf{x}_i)$. At the end of the algorithm we select the sparsest decision function (the function with the minimum number of variables) which gives a zero error classification performance. The experiment is repeated 10 times and is carried out for respectively $l = 4, 8, 16, 32, 64$ labeled data points. The obtained results are summarized in the Table 3.1 as well as the parameters used to realize the experiment. We can see that the number of selected variables increases when the number of labeled data is small. The decision frontier is supported by the variables $k(\mathbf{x}, \mathbf{x}_i)$ induced by the unlabeled data which tend to cover the manifold (see the illustrations of Figure 3.4). The relative high variability of the cardinality of $\mathcal{A}$ can be explained by the position in the manifold of the labeled data. If the labeled data cover well the manifold, a few number of variables is required to achieve a sparse solution.

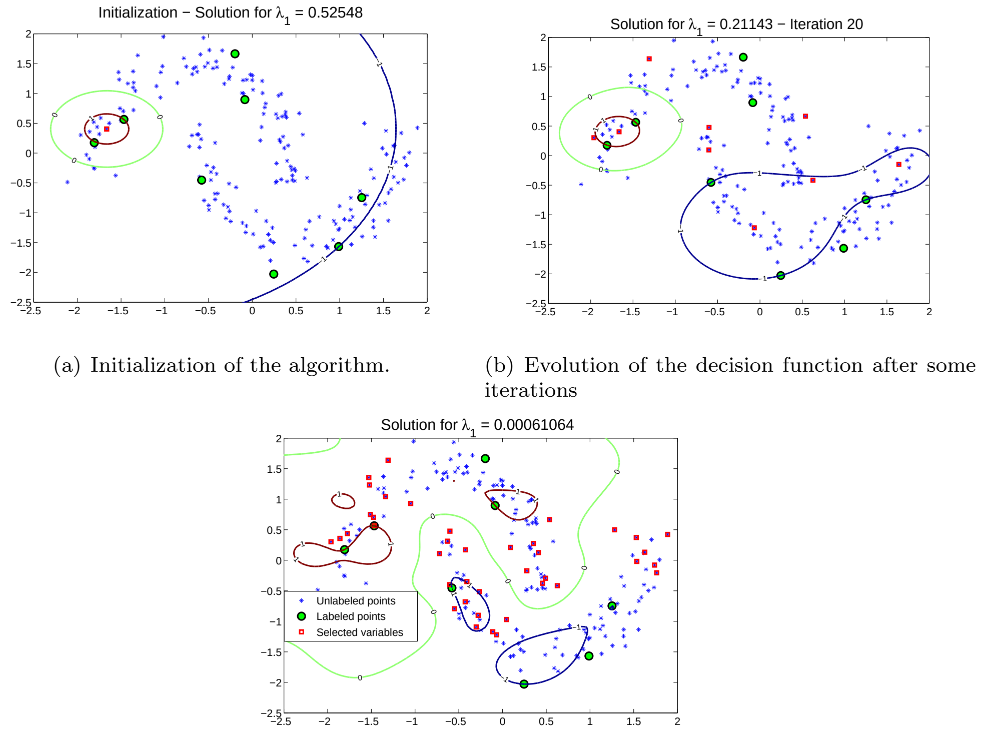
l	4	8	16	32	64
σ	0.5	0.5	0.5	0.5	0.5
λ	0.001	0.01	0.01	0.01	0.01
μ	1	1	1	1	1
$\lvert\mathcal{A}\rvert$	48.87 (12.58)	32.2 (11.69)	28.4 (12.81)	23.8 (6.54)	22.5 (2.27)

Table 3.1: Summary of the number of selected variables for different numbers of labeled points l for the two moons dataset. The mean number of selected variables over the 10 trials is given ($|\mathcal{A}|$) with its standard deviation in parenthesis.

It has to be remarked that for $l = 4$, more importance to the manifold assumption is given as $\lambda = 0.001$ in this case whereas $\lambda = 0.01$ is used in the other cases.

Finally, for this dataset, Figure 3.5 illustrates the evolution of the number of non-zero parameters as s varies. It can be noticed that when s is small, few variables are included in the model while it increases as s does.

The algorithm was also tested on real dataset. Firstly, it was tested on the USPS handwritten digit database from the UCI database. Initially, the algorithm was trained to separate class “2” against class “5”. The learning set comprises 847 points and the test set contains 1274 points. A Gaussian kernel with $\sigma = 10$ was used while the regularization parameters were $\lambda = 0.001$ and $\mu = 0.1$. The labeled points were randomly selected and, to limit the calculation time, the algorithm was stopped until the same results as the LapSVM were obtained (which involves all 847 variables). Table 3.2 summarizes the results which were averaged over 7 runs for each of



(a) Initialization of the algorithm.

(b) Evolution of the decision function after some iterations

(c) A final sparse solution.

Figure 3.4: Illustration of the two moons dataset. The sparse solution gives a 100% good classification with 40 variables whereas the 200 variables are needed in the original framework of the Laplacian SVM. labeled data points are marked with circles while the selected variables for the model correspond to points represented by a red square. These last ones tend to cover the manifold.

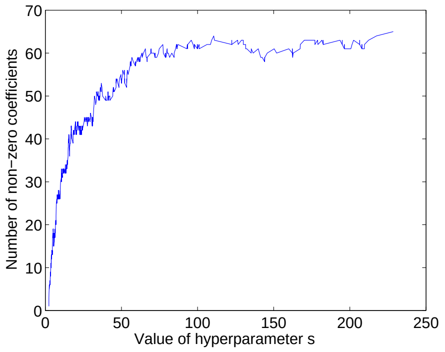


Figure 3.5: Evolution of the non-zero coefficients as s increases. Two moons dataset case with $\ell = 4$.

the values of ℓ. The standard deviations are indicated in parentheses. As we can see, a similar performance can be achieved with a reduced number of parameters.

The same experiment was done to classify “1” versus “7”. The results are summarized in Table 3.3. In this example, $\ell + u = 954$ learning points were considered. Again, it can be seen that the reduction of the number of variables is important.

These tests were then extended to two other datasets: the `Text` and the `coil3` set. `Text` is a docu-

l	16	32	64
$\mathcal{A}$	132 (104.17)	126 (154.18)	105.87 (114)
Test error (%)	19 (8.73)	13 (2.78)	9 (3.33)

Table 3.2: USPS dataset results of class "2" against class "5". The error is the average number of misclassified points (expressed in percentage), standard deviation of the average number of active points and the error is in parenthesis.

ℓ	16	32	64
$\|\mathcal{A}\|$	348 (118.7)	317 (37.42)	408 (190.40)
Test error (%)	36.8 (23.63)	23.25 (6.19)	13.75 (2.63)

Table 3.3: Sparse Laplacian SVM results on the USPS dataset "1" vs. "7". Average number of active variables and the test error (in percentage) is summarized together with its standard deviation in parenthesis.

ment classification problem while `coil3` is an image classification problem and their characteristics are explained in the Table 3.4.

	ℓ	$\ell + u$	Dimension	Number of Classes
`coil3`	6	216	1024	3
`Text`	50	1946	7511	2

Table 3.4: Characteristics of the `coil3` and `Text` datasets

A part of the set (25%) were used as test and the rest was used for the learning. Labeled datasets are again randomly selected and the accuracy is average along 4 tries. The obtained results can be seen in Table 3.5 for the `Text` dataset. It can be seen that the obtained accuracies are similar to the ones mentioned in [Sind 05a] or [Chap 08] but with a reduced number of variables. These accuracies are nevertheless lower than the ones given by the Transductive SVM algorithm that are of the 9% order [Chap 08].

	Laplacian SVM	Sparse Laplacian SVM
$\|\mathcal{A}\|$	1458	1027
Test error (in %)	10.38	10.38

Table 3.5: Obtained results on the `Text` dataset with the sparse Laplacian SVM algorithm.

For the `coil3` dataset the classification was done by pair. Given the low number of data, a cross validation was done. For each classifier, the dataset was divided in four, where one part was used as validation set and the other three as learning set. The results shown in Table 3.6 cannot be seen as a multi-class learner but as the performances of each classifier. It can be again appreciated that the use of a parsimonious model does not substantially decrease the generalization ability of the Laplacian SVM. Results with low accuracy are essentially linked to the difficulty to discriminate between the class "2" and "3". Nevertheless, the presented results are better to the ones given by a semi-supervised approach based on the cluster assumption [Chap 08].

3.3.7 L_2 regularization path for the sparse Laplacian SVM

To develop the previous regularization path, it was supposed that parameters λ and μ are known and fixed. If additionally, the interest is turned into the analysis of the influence of the parameter λ over the solution f, a regularization path can also be established, which will be called the L_2-path. Especially, if we fix μ and ζ (or s), the λ-path can be derived as briefly explained below.

		Laplacian SVM	Sparse Laplacian SVM
'1' vs. '2'	$\|\mathcal{A}\|$	108	83
	Test Error	5.55	4.86
'1' vs. '3'	$\|\mathcal{A}\|$	108	90
	Test Error	17.36	18.05
'2' vs. '3'	$\|\mathcal{A}\|$	108	88
	Test Error	29.16	29.16

Table 3.6: Obtained results on the `Coil3` datasets with the sparse Laplacian SVM algorithm. The error is expressed in percentage.

We suppose ζ fixed. Therefore, Conditions (3.8) and (3.15-3.17) hold. Recalling Relation (3.15) and making explicit the dependence of $P_{\mathcal{A}}$ on λ, we get

$$\lambda R_{\mathcal{A},\mathcal{A}}\boldsymbol{\beta}_{\mathcal{A}} - K_{\mathcal{A},S_\ell}Y\boldsymbol{\alpha} + \zeta\,\mathrm{sign}\,(\boldsymbol{\beta}_{\mathcal{A}}) \quad = \quad 0$$

with $R = K(I + \rho\boldsymbol{\mathcal{L}}K)$ and $\rho = \mu/\lambda$. If we let $\nu = \frac{1}{\lambda}$ and $\tilde{\alpha}_i = \nu\alpha_i$, the latest conditions become:

$$R_{\mathcal{A},\mathcal{A}}\boldsymbol{\beta}_{\mathcal{A}} - K_{\mathcal{A},S_\ell}Y\tilde{\boldsymbol{\alpha}} \quad = \quad -\nu\,\zeta\,\mathrm{sign}\,(\boldsymbol{\beta}_{\mathcal{A}}) \tag{3.20}$$

At this point, we proceed in a slightly different way to the L_1-path as some parameters will vary in a linear way with respect to λ while other do it inversely to λ. Given a value of λ^t at step t (corresponding to a ν^t) the solutions $\boldsymbol{\beta}^t$, $\boldsymbol{\alpha}^t$ b^t and the sets $\mathcal{I}^t_\alpha$, $\mathcal{I}^t_0$, $\mathcal{I}^t_1$, $\mathcal{A}^t$ and $\bar{\mathcal{A}}^t$ are known. Under the hypothesis that these sets remain unchanged when ν^t changes to ν, it can be established by differentiation of the Equations (3.8), (3.16) and (3.20) the following relations:

$$\mathbf{y}_{\mathcal{I}_\alpha}^\top \frac{\Delta\boldsymbol{\alpha}_{\mathcal{I}_\alpha}}{\Delta\lambda} \quad = \quad 0 \tag{3.21}$$

$$Y_{\mathcal{I}_\alpha}\left(K_{\mathcal{I}_\alpha,A}\frac{\Delta\boldsymbol{\beta}_{\mathcal{A}}}{\Delta\nu} + \mathbf{1}_{\mathcal{I}_\alpha}\frac{\Delta b}{\Delta\nu}\right) \quad = \quad 0 \tag{3.22}$$

$$R_{\mathcal{A},\mathcal{A}}\frac{\Delta\boldsymbol{\beta}_{\mathcal{A}}}{\Delta\nu} - K_{\mathcal{A},\mathcal{I}_\alpha}Y_{\mathcal{I}_\alpha}\frac{\Delta\tilde{\boldsymbol{\alpha}}}{\Delta\nu} \quad = \quad -\zeta\,\mathrm{sign}\,(\boldsymbol{\beta}_{\mathcal{A}}) \tag{3.23}$$

with $\Delta\nu = \nu - \nu^t$. It will be noticed that the variations of β_j, $\tilde{\alpha}_i$ and b are given with respect to ν instead of λ. The following relation can be deduced:

$$\frac{\Delta\tilde{\alpha}_i}{\Delta\nu} = \alpha_i^t - \lambda^t\frac{\Delta\alpha_i}{\Delta\lambda}$$

By the substitution of this relation in Equation (3.23), it can be finally established

$$R_{\mathcal{A},\mathcal{A}}\frac{\Delta\boldsymbol{\beta}_{\mathcal{A}}}{\Delta\nu} + \lambda^t K_{\mathcal{A},\mathcal{I}_\alpha}Y_{\mathcal{I}_\alpha}\frac{\Delta\boldsymbol{\alpha}_{\mathcal{I}_\alpha}}{\Delta\lambda} \quad = \quad \lambda R_{\mathcal{A},\mathcal{A}}\boldsymbol{\beta}^t_{\mathcal{A}}. \tag{3.24}$$

Combining the Equations (3.22), (3.21) and (3.24), it can be seen that the unitary variations of the parameters can be calculated from a system of linear equations with $|\mathcal{I}_\alpha| + |\mathcal{A}| + 1$ equations and $|\mathcal{I}_\alpha| + |\mathcal{A}| + 1$ variables: $\frac{\Delta\boldsymbol{\beta}_{\mathcal{A}}}{\Delta\nu}$, $\frac{\Delta\boldsymbol{\alpha}_{\mathcal{I}_\alpha}}{\Delta\lambda}$ and $\frac{\Delta b}{\Delta\nu}$.

By letting $\boldsymbol{\theta}$ be the vector containing the unitary variations, it can be deduced that as the parameter λ varies, parameters $\boldsymbol{\alpha}$ vary linearly with respect to λ while the parameters $\boldsymbol{\beta}$ and b vary linearly with respect to ν.

From this linear variation, it is then possible to use the same development line as the L_1 regularization path and to deduce the conditions for the different previous events to occur that will cause a change in the partition sets.

3.3.8 Conclusion

A L_1-norm regularization path was presented for a semi-supervised learning algorithm (the Laplacian SVM). In the original approach, the decision function relies on all data (labeled or unlabeled). By introducing a L_1-norm constraint, we show that similar classification performance can be achieved using a fewer number of variables. Since the choice of the L_1-norm regularization parameter can be difficult for a given problem, a regularization path is proposed. In the experiments, the parameters λ and μ are chosen ad hoc. They can be computed using also a regularization path.

3.4. Unsupervised Learning

In numerous machine learning application (as bioinformatics, signal processing, etc.) the used data possess a very large dimensionality as they are issued from the extraction of a large number of features. Nevertheless, it can turn out that only a reduced number of these characteristics are actually providing information or are matter of interest for classification or ranking tasks, for example. In this case, a multidimensional projection in a space with lower dimensionality might be useful. Other reasons for doing such a projection is to allow a 2D or 3D visualization of the data that will preserve the essential morphology of the dataset.

A task on a set of samples $S_X = \{\mathbf{x}_i\}_{i=[\![n]\!]}$ with n data points, $\mathbf{x}_i \in \mathbb{R}^{\mathcal{D}} i = [\![n]\!]$, with no label information form an ***unsupervised learning problem***. Even though no extra information is given, an intrinsic structure can still be learned. The aim of unsupervised learning is to discover this hidden structure without extra information apart from the data itself.

The underlying idea is to project the original data points in a space with reduced dimensionality so that the similarity (or dissimilarity) among points is mostly preserved in the projection. Most algorithms tend to minimize a criteria that measures the preservation of this similarity (or dissimilarity).

Unsupervised methods aim at ***clustering***, ***dimensionality reduction*** or ***manifold learning***. The first task looks for a partition of the dataset S_X that will keep in a cluster all samples sharing similar characteristics, while the last two tasks can be used as a preprocessing step for classification or regression, for feature extraction, improvement on calculation time or generation of computationally feasible problem. These techniques are useful to discover similarities and differences between data points or to establish relations between them. For instance, in semi-supervised learning setup, the informations conveyed by the clusters formed by the data or the informations about the geometrically shape of these data can help to improve the learning performances [Sind 05a, Chap 05] as shown in the previous section.

In the clustering task, the aim is to make a partition of S_X in k subsets, which will be denoted $\Delta = \{C_1, C_2, ..., C_k\}$. A label $y_i = f(\mathbf{x}_i)$ will be assigned to each element in S_X. The quality of the found partition will be measured by the cost function $\mathcal{L}(\Delta)$. It can be chosen to have either a hard clustering or a soft clustering by giving a degree of membership, γ_{ij}, of element $\mathbf{x}_i$ to cluster C_j, where $\sum_{j=1}^{k} \gamma_{ij} = 1$ for all $i = [\![n]\!]$.

Although clustering is clearly an unsupervised learning problem, we will not develop further the related algorithms. In the sequel, we will be more concerned by dimension reduction techniques for which we have developed some interesting extensions of existing algorithms. However, to be complete in our bibliography work, we would like to mention that there exists several techniques to realize clustering. Many of the most popular techniques are based on a statistical formulation and solved using the well-known ***Expectation Maximization*** technique (See Appendices A.1.5 and A.1.6). We can also cite algorithms like hierarchical clustering [John 67] and spectral clustering [Ng 01, Luxb 07].

The remainder is organized as follows: after a description of the dimensionality reduction, we present an overview of techniques presented in the literature to deal with such a problem. We describe the strengths and weaknesses of those methods as well. The details of the approach we developed to overcome one weakness common to projection techniques based on graph construction (namely shortcuts problem) is discussed afterward.

3.4.1 Brief state of the art of the dimensionality reduction problems

Formally, the ***dimensionality reduction*** task [Fodo 02] can be stated as follows: given the $\mathcal{D}$-dimensional random vector $\mathbf{x} = (x_1, \ldots, x_{\mathcal{D}})^\top$, find a lower dimensional representation of it, $\mathbf{y} = (y_1, \ldots, y_d)^\top$ with $d \leq \mathcal{D}$, that captures the content in the original data, according to some criterion, that is, the process of reducing the number of random variables under consideration. The components of $\mathbf{y}$ are sometimes called the hidden components. Different fields use different names for the d multivariate vectors: the term *variable* is mostly used in statistics, while *feature* and *attribute* are alternatives commonly used in the computer science and machine learning literature.

Several methods for dimensionality reduction uses the assumption that the data lies on or near a low dimensional manifold. ***Manifolds*** are spaces that are locally linear, but unlike Euclidean subspaces, they can be globally nonlinear. Curves and surfaces are familiar examples of one and two dimensional manifolds. These methods use the idea that a manifold is differentiable and therefore a tangent subspace to the original manifold can be estimated. This will correspond to a tangent hyperplane in a space of lower dimensionality. All nearest neighbors should lie close to this tangent subspace, where distance will be measured not only with the Euclidean distance but also with the deviation angle to the tangent subspace.

We are willing to find the embedding manifold $\mathcal{M} \subset \mathbb{R}^D$ of input data $X = \{\mathbf{x}_i\}$, $\mathbf{x}_i \in \mathcal{M}$, $i = 1, \ldots, n$. The pursued objective is the projection of X into a subspace $\mathbb{R}^d$ $(d < \mathcal{D})$ that preserves the topological characteristics of $\mathcal{M}$. This will yield to dataset $Y = \{\mathbf{y}_i\}$, $\mathbf{y}_i \in \mathbb{R}^d$. To achieve this goal, several techniques proceed by constructing the neighborhood graph related to the training points and then finding a transformation of this graph.

There are two common used techniques to determine nearest neighbors of each point in the dataset: the k-NN approach and the ϵ-ball technique, which are explained in Section 3.4.4.

To illustrate the use of manifolds, consider Figure 3.6, the original dataset is a toroidal helix in three dimensions (Figure 3.6(a)), the relevant information that is to be maintained among the data points is the immediate neighborhood, that is, close points along the manifold are to be kept close. The neighborhood of each point can be represented by a graph, where each vertex is a data point and each edge corresponds to close neighbors (Figure 3.6(b)), the dataset could be represented with an equivalent manifold in two dimensions by keeping the neighborhood relation, as in Figure 3.6(c), achieving the reduction in the dimensionality.

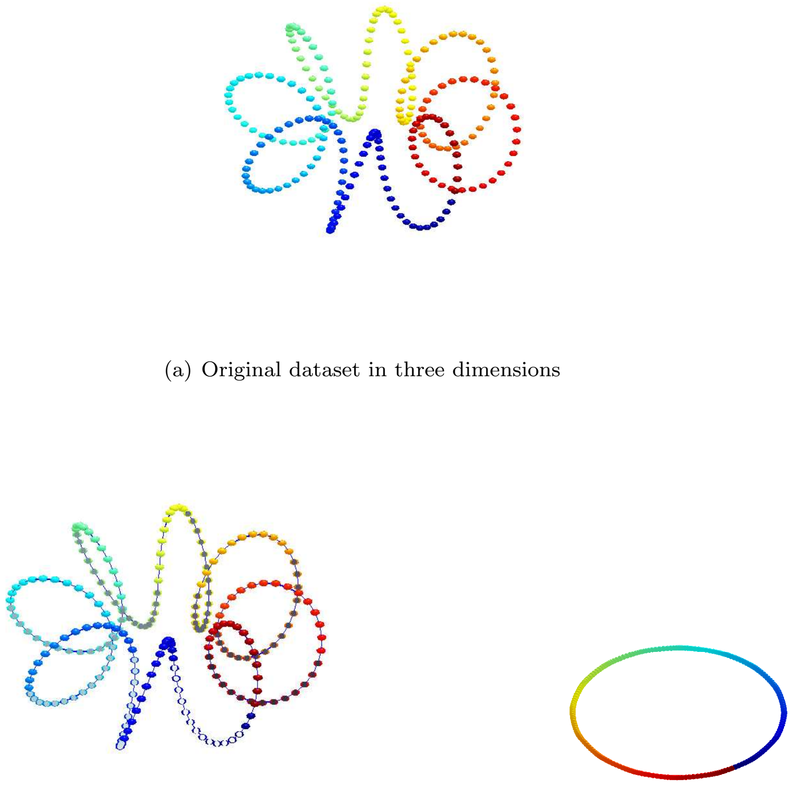
For the dimensionality reduction and manifold learning tasks, there is an equivalent goal consisting on the internal data structure unraveling with the aim at expressing data relations with less information. While learning the embedding manifold of a dataset, the learned manifold could be expressed as a compact function in a smaller space, that is, in a reduced dimensionality.

In order to obtain general information about a dataset, it is very usual to sample the data. If this subset of the whole population is taken under certain conditions, it will give a rough idea of the characteristics of the whole set.

There exists different methods for dimensionality reduction, good surveys can be found in several sources [Huo 07, Lawr 08, Lee 07, Verl 03, Gail 08]. These could be roughly divided in 3:

1. Statistically-based dimensionality reduction.
2. Graph-based dimensionality reduction.
3. Kernel extension, which are in general extensions of the previous one adapting the use of a kernel.

Several methods are based in the ***spectral decomposition*** technique, consisting in the factorization of a positive definite matrix A into $A = U\Lambda U^\top$ where Λ is a diagonal matrix containing the eigenvalues of A, matrix U contains its eigenvectors and $U^\top$ is the transpose of U. This



(a) Original dataset in three dimensions

(b) Information to be kept: neighborhood graph

(c) Data points embedding with only two dimensions

Figure 3.6: Manifold dimensionality reduction illustration. Figure 3.6(a) shows the distribution of the data points in the original space. In Figure 3.6(b), the corresponding nearest neighbors graph is depicted: two close points are linked by an edge. Finally, the desired projection in a space with lower dimensionality is shown in Figure 3.6(c). Colors are only used for visualization purposes.

.

decomposition can be written as a sum of outer products:

$$A = \sum_{i=1}^{n} \Lambda_i \mathbf{u}_i \mathbf{u}_i^\top,$$

where $\mathbf{u}_i$ is the i-th column of U. Given n points with $\mathcal{D}$-dimensional vector coordinates $\mathbf{u}_i$, let U be the $\mathcal{D} \times n$ matrix whose j-th column consists of the coordinates of the vector $\mathbf{u}_j$, with $j = [\![n]\!]$. Then define the $n \times n$ ***Gram matrix*** of dot products $a_{ij} = \mathbf{u}_i^\top \cdot \mathbf{u}_j$ as $A = U^\top U$. The Gram matrix determines the vectors $\mathbf{u}_i$ up to symmetry.

3.4.2 Dimensionality Reduction Background

3.4.3 Statistically-based dimensionality reduction

Methods which are statistically based, assume that the sampled points are issued from a probabilistic distribution, therefore, the variance among the points is observed or an estimation of the underlying density is done.

Principal Component Analysis (PCA). This method [Joll 86] consists in the projection of each data point $\mathbf{x}_i$ over the axis with maximum variance. Random variable $\mathbf{x}$ is assumed to have the following statistical properties:

- Variable is centered: $\mathbb{E}[\mathbf{x}] = 0$,
- Covariance matrix is $\Sigma = \mathbb{E}[\mathbf{x} \cdot \mathbf{x}^\top] \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$.

In order to realize the basis change, an orthonormal basis is considered $\{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_\mathcal{D}\}$ of S_X, with $\mathbf{w}_j \in \mathbb{R}^\mathcal{D}$. A matrix W will be built so that $W = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_\mathcal{D}]$. It will be noticed that given the orthonormality, we get that $W^\top W = I_\mathcal{D}$, the identity matrix in $\mathbb{R}^\mathcal{D}$. A projection on this new basis of a point $\mathbf{x}$ will be defined as follows:

$$\tilde{\mathbf{x}} = W^\top \mathbf{x} = \begin{bmatrix} \mathbf{w}_1^\top \mathbf{x} \\ \mathbf{w}_2^\top \mathbf{x} \\ \vdots \\ \mathbf{w}_\mathcal{D}^\top \mathbf{x} \end{bmatrix}.$$

Since W is orthogonal, $\mathbf{x}$ can be reconstructe by $\tilde{\mathbf{x}}$ by doing:

$$\mathbf{x} = W\tilde{\mathbf{x}} = \sum_{j=1}^{\mathcal{D}} (\mathbf{w}_j^\top \mathbf{x})\mathbf{w}_j.$$

Therefore, new variables $\mathbf{y}$ can be now considered to be issue of the projection of $\mathbf{x}$ on the vectorial sub-space spanned by vectors $\{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_d\}$ with $d < \mathcal{D}$, so that, in similar way, it can be written:

$$\mathbf{y} = V^\top \mathbf{x} \quad \text{with} \quad V = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_d] \in \mathbb{R}^{\mathcal{D}\times d}$$

The reconstruction of vector $\mathbf{x}$, denoted $\hat{\mathbf{x}}$ (which will be an approximation of $\mathbf{x}$) from $\mathbf{y}$ is written:

$$\hat{\mathbf{x}} = V\mathbf{y} = \sum_{j=1}^{d} (\mathbf{w}_j^\top \mathbf{x})\mathbf{w}_j.$$

In PCA, the minimization of the error between $\mathbf{x}$ and its reconstruction $\hat{\mathbf{x}}$ is searched:

$$J = \mathbb{E}\big[\, \|\mathbf{x} - \hat{\mathbf{x}}\|^2 \,\big] = \mathbb{E}\Big[\, \big\|(I - VV^\top)\mathbf{x}\big\|^2 \,\Big]$$

Using the properties $Var(A\mathbf{x}) = A{\cdot}Var(\mathbf{x}){\cdot}A^\top$, $tr(AB) = tr(BA)$, $tr(A{+}B) = tr(A){+}tr(B)$ and noticing that $V^\top V = I_d$, it can be established:

$$J = tr(\Sigma) - tr(V^\top \Sigma V)$$

see [Paie 03, Joll 86] for more details. Minimizing the reconstruction error is equivalent to look for vectors $\mathbf{w}_j, j = [\![d]\!]$ that maximize the trace of the variance-covariance matrix of the projection $\mathbf{y}$ because it has to be noticed that $\mathbb{E}[\mathbf{y}] = 0$ and $Var(\mathbf{y}) = Var(V^\top \mathbf{x}) = V^\top \Sigma V$.

Variance Maximization. Vectors $\mathbf{w}_j$ can be iteratively searched. For $\mathbf{w}_1$ it is searched:

$$\max_{\mathbf{w}_1} J \quad \text{subject to } \mathbf{w}_1^\top \mathbf{w}_1 = 1.$$

The solution is to take $\mathbf{w}_1$ as the associated eigenvector to the largest eigenvalue λ_1 of Σ. For $\mathbf{w}_2$:

$$\max_{\mathbf{w}_2} J \quad \text{subject to} \quad \mathbf{w}_2^\top \mathbf{w}_2 = 1 \quad \text{and} \quad \mathbf{w}_2^\top \mathbf{w}_1 = 0.$$

The solution will be to take $\mathbf{w}_2$ as the associated eigenvector to the second largest eigenvalue λ_2 of Σ. Proceeding like this, the d vector components of matrix V can be determined.

Practical approach. The PCA method can be summarized as follows:

- Center observations $\mathbf{x}_i, i = [\![n]\!]$.
- Calculate the empirical variance-covariance matrix $\Sigma = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^\top$.
- Diagonalize matrix $\Sigma = W \Lambda W^\top$ with $\Lambda = \text{diag}\,(\lambda_1, \cdots \lambda_{\mathcal{D}})$ and $\lambda_1 \geq \lambda_2 \ldots \geq \lambda_{\mathcal{D}} \geq 0$.
- Choose the d eigenvalues providing maximal variance.
- The searched matrix V is formed by the eigenvectors associated to these eigenvalues.

Properties. The projection done by PCA possesses the following characteristics:

1. Let y_k be the k^{th} component of variable $\mathbf{y}$: $Var(y_k) = \lambda_k$ which is the variance contribution by the k^{th} principal component.
2. $\mathbb{E}[\mathbf{y}_i{}^\top \mathbf{y}_j] = 0$: PCA linearly transforms dependent variables $\mathbf{x}_\ell, \ell = [\![\mathcal{D}]\!]$ in linearly independent variables $\mathbf{y}_k, k = [\![d]\!]$.
3. $\mathbf{y}_k = \sum_{\ell=1}^{\mathcal{D}} w_{\ell k} \mathbf{x}_\ell$ is a linear combination of the initial variables, where $w_{\ell k}$ is the k-th element of the ℓ-th vector $\mathbf{w}_\ell$.
4. For each observation $\mathbf{x}_i$, its projection is given by $\mathbf{y}_i = V^\top \mathbf{x}_i$, that is, it is possible to calculate the projection of each new point, knowing the PCA model, V.
5. It does not has tuning parameters.
6. It is an Eigenvector method and therefore, it does not require iterations.
7. There are no local optima.

Disadvantages. Some of the weaknesses of the PCA method are:

- It ignores correlations in the data that are higher than second order, as it is a linear transformation, and limited to second order statistics.
- It assumes that large variances corresponds to interesting directions.

Metric Multidimensional Scaling (MDS). Multidimensional scaling is a linear method for dimensionality reduction based in Euclidean distances. Classical (or metric) MDS finds an embedding that preserves the inter-point distances via spectral decomposition, which is in fact, equivalent to PCA when those distances are Euclidean.

The principle of the method can be summarized by the two following points:

- Similarity among samples $S_X = \{\mathbf{x}_i\}, i = [\![n]\!]$ is measured with the Euclidean distance. A distance matrix is then generated $\Delta = [\Delta_{ij}]$ with $\Delta_{ij} = d(\mathbf{x}_i, \mathbf{x}_j)$, the Euclidean distance for all $\mathbf{x}_i, \mathbf{x}_j \in S_X$.
- A configuration for points $\mathbf{y}_i, i = [\![n]\!]$ is searched in a space with reduced dimensionality so that $d(\mathbf{y}_i, \mathbf{y}_j) \approx \Delta_{ij}$, that is, the new space will try to preserve the distances between the initial points $\mathbf{x}_i$. This will be equivalent to optimize the criterion:

$$J = \sum_{i=1}^{n}\sum_{j=1}^{n}(\Delta_{ij} - d(\mathbf{y}_i, \mathbf{y}_j))^2.$$

Definition 3.1 (Euclidean Matrix)**.** *A matrix of distance $\Delta = [\Delta_{ij}] \in \mathbb{R}^{n\times n}$ is called an Euclidean matrix if for points $\mathbf{x}_i, i = [\![n]\!]$, we have $\Delta_{ij} = (\mathbf{x}_i - \mathbf{x}_j)^\top(\mathbf{x}_i - \mathbf{x}_j)$.*

Method. The multidimensional Scaling method is based in the following theorem that establishes if a matrix is Euclidean or not.

Theorem 3.2 (Euclidean Matrix [Paie 03])**.** *Let $\Delta = [\Delta_{ij}]$ be a matrix and $A = [a_{ij}]$ so that $a_{ij} = -\frac{1}{2}\Delta_{ij}$, finally, let $M = HAH$ where H is a centering matrix of the data defined by $H = I_N - \frac{1}{N}1_N 1_N^\top$. Matrix Δ is called Euclidean if and only if matrix M is positive semi-definite. In this case, matrix M coincides with the Gram matrix $G = HXX^\top H$ calculated with the centered variables.*

Remark: data matrix X is structured in the following way:

$$X = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} \in \mathbb{R}^{n\times\mathcal{D}}$$

Practical approach. The basis of the MDS method is derived following the next steps, assuming that the distance matrix Δ is given:

- Build from Δ matrix $M = HAH$ with $A = [a_{ij}]$, $a_{ij} = -\frac{1}{2}\Delta_{ij}$ and H the centering matrix.
- Calculate the spectral decomposition of M:

  $$M \quad = \quad W\Lambda W^\top \quad \text{with} \quad W = [\mathbf{w}_1 \cdots \mathbf{w}_n] \in \mathbb{R}^{n\times n} \quad \text{and} \quad \Lambda = \operatorname{diag}(\lambda_1, \dots, \lambda_n).$$

  Matrix M has $\mathcal{D}$ non-zero eigenvalues and $n - \mathcal{D}$ zero eigenvalues (assuming $n > \mathcal{D}$).
- Recalling Theorem 3.2, it can then be written $X_c = HX = W\Lambda^{1/2}$. To obtain a configuration of points in a space with a reduced dimensionality d, it is enough to consider:

  $$Y = W\Lambda_d^{1/2} \quad \text{with} \quad W = [\mathbf{w}_1 \cdots \mathbf{w}_d], \quad d < \mathcal{D} \quad \text{and} \quad \Lambda_d = \operatorname{diag}(\lambda_1, \dots, \lambda_d).$$

  The structure of matrix Y will be: $Y = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n]^\top \in \mathbb{R}^{n\times d}$. Finally, the resulting new variables will be written as: $\mathbf{y}_j = \sqrt{\lambda_j}\mathbf{w}_j$.

Disadvantages. Since the MDS scaling method is equivalent to PCA if the Euclidean distance is used, it has the same weaknesses as this last one.

Principal Curves. This method can also be seen as a continuous and unidimensional self-organizing map or as a non-linear generalization of PCA. In an intuitive manner, a principal curve is a smooth curve that passes in the middle of the observed data. More formally, they were defined by Hastie and Stuetzle [Hast 89] as a parametric curve that respects a consistency score:

$$f(\mathbf{w}) = \mathbb{E}[\mathbf{x}|g(\mathbf{x}) = \mathbf{w}] \tag{3.25}$$

where $\mathbf{x}$ is an out coming data point from a density probability $\mathbb{P}(\mathbf{x})$. $g(\mathbf{x})$ is the projection of $\mathbf{x}$ over the curve f, that is, is the value of $\mathbf{w}$ such that $f(\mathbf{w})$ is the closest one to $\mathbf{x}$ (or the largest value if several are possible):

$$g(\mathbf{x}) = \sup\{\lambda : \|\mathbf{x} - f(\lambda)\| = \inf_{\tau}\{\mathbf{x} - f(\tau)\}\}. \tag{3.26}$$

This means that each point of the curve is the mean (with respect to the distribution $\mathbb{P}(\mathbf{x})$) of all data points that projected over it.

In an analogous way to the PCA, the principal curve is the one that minimizes the error of distortion:

$$\mathcal{L} = \mathbb{E}_{\mathbb{P}(\mathbf{x})}[\|\mathbf{x} - g(\mathbf{x})\|^2]. \tag{3.27}$$

From the consistency Property (3.25), an algorithm was proposed to build a principal curve if the density $\mathbb{P}(\mathbf{x})$ is known. Given an initial curve (for example, the principal axis of the dataset), the algorithm iterates two steps that resembles the k-means algorithm:

- the projection of the dataset over the curve (3.26)
- the estimation of the mean of the data point that are projected to the same point (3.25).

In practice, the density $\mathbb{P}(\mathbf{x})$ is unknown and in this case, the expectation in Equation (3.25) cannot be calculated. When this density is characterized with a finite sampling, each point of the curve is then defined as a weighted sum of the data points that are locally projected around it.

Strengths.

- It can handle several connected components.
- It is a non-linear method.
- It can handle some noise.

Disadvantages.

- The intrinsic dimensionality of the model is one.

3.4.4 Graph-based Dimensionality Reduction Methods

The following methods are based in the construction of a neighborhood graph. The neighboring points are usually determined by three different rules:

k-Nearest Neighbors (k-NN). This method consists in choosing the closest k points to each sampled point and design them as the nearest neighbors.

ε-ball. In this case, a spherical neighborhood is chosen of fix diameter ε. Samples belonging to the sphere of size ε centered at a particular sampled point are set as nearest neighbors to it. In the context of a neighborhood graph with adjacency matrix W, the edges would be chosen as follows:

$$W_{ij} = \begin{cases} 1 & \text{if} \quad \|\mathbf{x}_i - \mathbf{x}_j\| \leq \varepsilon \\ 0 & \text{otherwise.} \end{cases}$$

Minimum Spanning Tree. If a graph has n vertex, a spanning tree of a graph is a subset of $n-1$ edges that form a tree. The minimum spanning tree of a weighted graph is a set of $n-1$ edges of minimum total weight which form a spanning tree of the graph [Weis]. Weights in a neighborhood graph is given by the distance. This technique [Zhao 06] consists in finding the pair nearest of nearest components. A component is each set of points that are connected between them, but disconnected from other components. The algorithm ends when there is left only one component.

k and ε are the parameters to set in order to rule the first two neighborhood searching algorithms. Some of the latest algorithms for dimensionality reduction based on the construction of the neighborhood graph are enumerated:

Isometric feature mapping (Isomap) Isomap [Tene 00] is a variant of MDS in which standard Euclidean distances are substituted by estimates of geodesic distances along the submanifold.

The key assumption made by Isomap is that the quantity of interest, when comparing two points is the distance along the curve between the two points. The geodesic distance is approximated by the shortest path in the net formed by neighboring points input.

This method preserves the pairwise distances between input patterns as *measured along the submanifolds from which they were sampled.*

Definition 3.3. ***Geodesic Distance***. *The geodesic distance between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined as the length of the shortest path from $\mathbf{x}_i$ to $\mathbf{x}_j$ along the manifold. This is different from the Euclidean distance in a graph, for example, as the distance between two points is measured along the shortest path over the connecting edges.*

Practical approach to Isomap. The following shows the three basic steps for the Isomap algorithm:

1. Construct the neighborhood graph
   - Determine the nearest neighbors of each sample using one of the previous algorithm.
   - Construct a graph whose vertex represent input patterns and whose (undirected) edges connect nearest neighbors.
   - Assign weights to the edges based on the Euclidean distance between nearest neighbors.
2. Compute the pairwise distance Δ_{ij} between all nodes $(\mathbf{x}_i, \mathbf{x}_j)$ along shortest paths through the graph.
   - Algorithms: *Djikstra's algorithm*, $O(n^2 \log n + n^2 k)$ operations; *Floyd's algorithm*, $O(n^3)$ operations.
   - Make matrix D such that $D_{ij} = \Delta_{ij}$.
3. Construct d-dimensional embedding by applying MDS to the shortest path distance matrix D.

As result we can obtain a low dimensional embedding ($d << \mathcal{D}$) of the original dataset in which the Euclidean distances between outputs match the geodesic distances between input patterns on the submanifold from which they were sampled.

True dimensionality of the data can be estimated from the decrease of the eigenvalues as the dimensionality of the embedding is increased.

Strengths

- Method that directly models the manifold [Burg 05] without mapping.
- It is capable of discovering the nonlinear transformations and recover the true dimensionality and geometric structure of a strictly larger class of nonlinear manifolds than MDS and PCA.
- It combines the major algorithmics features of PCA and MDS: reasonable cost, global optimality, and asymptotic convergence guarantees.
- Non-iterative (one pass through data).
- Polynomial-time optimizations.
- Only heuristic is the neighborhood size.

Weaknesses

- The scale-invariant k parameter is typically easier to set than ε, but this may yield to misleading results when the local dimensionality varies across the data set. When available, additionally constraints such as the temporal ordering of observations may also help to determine neighbors.
- It is vulnerable to noise because it considers the subset of point-to-point relationships that has the lowest signal-to-noise ratio; small changes to the trusted set can induce large changes in the set of constraints on the embedding, making solutions unstable. Topological stability could be achieve by tuning the parameter ε [Bala 02] or k. Thus, it is sensitive to the graph construction.
- It can fail if the data hull is non-convex [Bran 03].
- No immediate out-of-sample extension.

Maximum Variance Unfolding (MVU). Founded on the notion of *isometry*, MVU [Saul 05, Wein 04] considers the transformations that only preserve the geometric properties of local neighborhoods. Formally, two Riemannian manifolds are said to be isometric if there is a diffeomorphism such that the metric on one pulls back to the metric on the other. Informally, an isometry is a smooth invertible mapping that looks locally like a rotation plus translation, thus preserving distances along the manifolds.

Intuitively, for two dimensional surfaces, the class of isometries includes whatever physical transformation one can perform on a sheet of paper without introducing holes, tears, or self-intersections.

The aim is to translate inputs $\{\mathbf{x}_i\}_{i=1}^n, \mathbf{x}_i \in \mathbb{R}^{\mathcal{D}}$ into outputs $\{\mathbf{y}_i\}_{i=i}^n \in \mathbb{R}^d$ of lower dimensionality, $d << \mathcal{D}$. Neighborhoods of inputs and outputs will be related by translation and rotation if and only if all the distances and angles between points and their neighbors are preserved. Thus, whenever both $\mathbf{x}_j$ and $\mathbf{x}_k$ are neighbors of $\mathbf{x}_i$, for local isometry, we must have that:

$$\langle \mathbf{y}_i - \mathbf{y}_j, \mathbf{y}_i - \mathbf{y}_k \rangle = \langle \mathbf{x}_i - \mathbf{x}_j, \mathbf{x}_i - \mathbf{x}_k \rangle \tag{3.28}$$

or

$$||\mathbf{y}_i - \mathbf{y}_j||^2 = \|\mathbf{x}_i - \mathbf{x}_j||^2 \tag{3.29}$$

Equation (3.28) is sufficient for local isometry because the triangle formed by any point and its neighbors is determined up to a rotation by specifying the lengths of two sides and the angle between them. This can also be determined with Equation (3.29) by specifying the lengths of all its sides.

The outputs $\mathbf{y}_i$ will be also constrained to be centered on the origin:

$$\sum_{i=1}^{n} \mathbf{y}_i = 0. \tag{3.30}$$

Equation (3.30) removes a translational degree of freedom from the final solution. With this, a semidefinite optimization problem can be stated:

Problem 3.4 (MVU)**.**

$$\begin{array}{lll} \text{maximize} & \sum_{ij} ||\mathbf{y}_i - \mathbf{y}_j||^2, & (3.31) \\ \text{subject to} & \sum_i \mathbf{y}_i = \mathbf{0}, & (3.32) \\ & ||\mathbf{y}_i - \mathbf{y}_j||^2 = D_{ij}, \text{ for all } (i,j) \text{ with } \eta_{ij} = 1, & (3.33) \end{array}$$

where D is the distance matrix such that $D_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||^2$ and $\eta_{ij} = 1$ if node i is connected to node j and 0 otherwise.

The previous problem can be reformulated in terms of the elements of the inner product matrix $K_{ij} = \langle \mathbf{y}_i, \mathbf{y}_j \rangle$, this matrix is positive semi-definite, denoted by $K \succeq 0$. Using a change of variables, we have the following:

- Equation (3.32) can be rewritten as $\sum_{ij} K_{ij} = 0$ (*centering*).
- Equation (3.33) can be rewritten as $K_{ii} - 2K_{ij} + K_{jj} = D_{ij}$ (*local isometry*).
- Finally, if we expand the terms in the objective function, and using the property that the outputs are centered in the origin, Equation (3.31) can be rewritten as

$$\begin{aligned} \frac{1}{2n} \sum_{ij} ||\mathbf{y}_i - \mathbf{y}_j||^2 &= \frac{1}{2n} \sum_{ij} (||\mathbf{y}_i||^2 + ||\mathbf{y}_j||^2 - 2\mathbf{y}_i \cdot \mathbf{y}_j) \\ &= \sum_i ||\mathbf{y}_i||^2 = \sum_i K_{ii} = Trace(K). \end{aligned}$$

Additionally, the distance matrix K is constrained to be *positive semidefinite*, this is denoted by $K \succeq 0$, leading to a convex semi-define programming (SDP) problem.

Practical approach to MVU. The following steps describe the MVU algorithm:

1. Construct the neighborhood graph
   - Build a neighborhood graph for the n samples $\mathbf{x}_i$ using a k-NN algorithm.
   - Let $\eta_{ij} = 1$ if node i in the graph is connected to node j and 0 otherwise, D be the distance matrix for inputs $\mathbf{x}_i$.
2. Solve the following optimization problem.

   Problem 3.5 (Dual MVU)**.**

$$\begin{array}{ll} \text{maximize}_K & Trace(K), \\ \text{subject to} & K \succeq 0, \\ & \sum_{ij} K_{ij} = 0, \\ & K_{ii} - 2K_{ij} + K_{jj} = D_{ij}, \text{ for all } (i,j) \text{ with } \eta_{ij} = 1. \end{array} \tag{3.34}$$

3. Spectral decomposition
   - The outputs can be recovered by matrix diagonalization. For the matrix K, let $\boldsymbol{\nu}_{\alpha i}$ denote the i-th element of the α-th eigenvector, with eigenvalue λ_α.

- A d-dimensional embedding that is k-locally isometric to the inputs $\mathbf{x}_i$ is obtained by identifying the α-th element of the output $\mathbf{y}_i$ as:

$$\mathbf{y}_{\alpha i} = \sqrt{\lambda_\alpha} \boldsymbol{\nu}_{\alpha i} \tag{3.35}$$

- The eigenvalues of K are guaranteed to be nonnegative. Thus, from Equation (3.35), a large gap in the eigenvalue spectrum between the d-th and the $(d+1)$-th eigenvalues indicates that the outputs lie in or near a subspace of dimensionality d. In this case, a low dimensional embedding that is *approximately* locally isometric is given by truncating the elements of $\mathbf{y}_i$.

Problem 3.4 is a not convex problem, as it involves maximizing a quadratic form subject to quadratic equality constraints. As opposed to PCA and MDS, here the eigenvalue spectrum reflects the dimensionality of an underlying manifold and not merely a subspace.

Further refinements of the MVU algorithm was proposed by relaxing the equality constraints 3.34 and allowing slight violations of these constraints using positive slack variables similarly to SVM [Hou 08]. This relaxation may prove particularly usefulness in applications when the distances D_{ij} are not computed from Euclidean distances but are specified in some other way.

Strengths

- Eigenvalues reveal dimensionality.
- Constraints ensure local isometry.

Weakness

- Computationally expensive.
- Limited to $n \leq 2000$, $k \leq 6$ to keep computational time reasonable.
- Limited to isometric embeddings.

Locally Linear Embedding (LLE). The LLE algorithm [Rowe 00, Saul 00] is based on geometric intuitions. It is assumed that the sample data S_X is taken from some smooth underlying manifold. In the case where the manifold is well-sampled, there would be sufficient data so that each data point will lie with its neighbors close to a linear subspace of the manifold. The local geometry can be characterized by linear coefficients that reconstruct each data point from its neighbors, which are chosen either with the k-NN or the ε-ball techniques. Reconstruction errors are then measured by the cost function:

$$\mathcal{E}(W) = \sum_i \left\| \mathbf{x}_i - \sum_j W_{ij} \mathbf{x}_j \right\|^2, \tag{3.36}$$

which adds up the squared distances between all the data points and their reconstructions. The weights W_{ij} summarize the contribution of the j-th data point to the i-th reconstruction. To compute the weights W_{ij}, the cost function is minimized subject to two constraints:

- Each data point $\mathbf{x}_i$ is reconstructed only from its neighbors, enforcing $W_{ij} = 0$ if $\mathbf{x}_j$ does not belong to this set.
- The rows of the weight matrix sum to one: $\sum_j W_{ij} = 1$.

The constrained weights that minimize these reconstruction errors obey to an important symmetry: for any particular data point, they are invariant to rotations, rescalings, and translations of that data point and its neighbors, where the first invariance follows directly from Equation (3.36) and the last two are enforced by the sum-to-one constraint on the rows of the weight matrix. The enforced symmetry will lead to the reconstruction weights and characterization of the intrinsic geometric properties for each neighborhood, as opposed to properties that depend on a particular frame of reference.

As it is supposed that the samples are taken from a manifold of dimensionality $d < \mathcal{D}$, in the final step of LLE algorithm, each high dimensional observation $\mathbf{x}_i$ is mapped to a low dimensional vector $\mathbf{y}_i$ representing global internal coordinates on the manifold. This is done by choosing d-dimensional vector $\mathbf{y}_i$ representing global internal coordinates on the manifold, which will be done by minimizing the embedding cost function:

$$\Phi(\mathbf{y}) = \sum_i \left\| \mathbf{y}_i - \sum_j W_{ij} \mathbf{y}_j \right\|^2 . \tag{3.37}$$

As the previous function in Equation (3.36) is based on locally linear reconstruction errors, but here weights W_{ij} are fix while optimizing the coordinates $\mathbf{y}_i$. Additional constraints like centering of the outputs $\mathbf{y}_i$ and a unit covariance matrix for these outputs are also enforced in the learning problem.

Practical Approach of LLE

1. Compute the neighbors of each data point, $\mathbf{x}_i$.
2. Compute the weights W_{ij} that best reconstruct each data point $\mathbf{x}_i$ from its neighbors, minimizing the cost in Eq. (3.36) by constrained linear fits.
3. Compute the vectors $\mathbf{y}_i$ best reconstructed by the weights W_{ij}, minimizing the quadratic form in Eq. (3.37). These vectors are yielded by the d lower eigenvalues of the matrix $(I - W)(I - W)^\top$.

Strengths

- Polynomial-time optimizations
- No local minima
- Non-iterative (one pass through data)
- Only heuristic is the neighborhood size.

Weakness

- Sensitive to *shortcuts*
- No out-of-sample extension
- No estimate of dimensionality

Laplacian Eigenmaps. The core of this algorithm [Belk 01] is very simple: it consists in a few local computations and one sparse eigenvalue problem. The solution will reflect the intrinsic geometric structure of the manifold. The Laplacian operator is used to justify help providing an optimal embedding. The Laplacian of the graph obtained from the data points may be viewed as an approximation to the Laplace-Beltrami operator defined on the manifold (see the Appendix A.1 and [Belk 01, Hein 07] for additional definitions). The embedding maps for the data come from approximations to a natural map that is defined on the entire manifold. The connection of the Laplacian to the heat kernel enables the algorithm to choose the weights of the graph in a principled manner.

As the Laplacian Eigenmap algorithm tries to preserve locality, the algorithm is relatively insensitive to outliers and noise.

Practical Approach of Laplacian Eigenmaps.

1. Construct the neighborhood graph.
2. Compute the adjacency matrix. Here are two variations:
   - Heat kernel: if nodes i and j are connected, set
   $$W_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}}$$
   where it is necessary to choose parameter σ^2.
   - Simple-minded: if nodes i and j are connected, set
   $$W_{ij} = 1 \quad \text{and} \quad 0 \quad \text{otherwise.}$$
3. Eigenmaps. Assume the graph G, constructed above, is connected, otherwise proceed with this step for each connected component: compute eigenvalues and eigenvectors for the generalized eigenvector problem:
   $$\boldsymbol{\mathcal{L}}\mathbf{y} = \lambda D\mathbf{y} \tag{3.38}$$
   where D is a diagonal weight matrix, its entries are the sum of columns (or rows, since W is symmetric) of W, $D_{ii} = \sum_j W_{ji}$. $\boldsymbol{\mathcal{L}} = D - W$ is the Laplacian matrix. Laplacian is a symmetric, positive semidefinite matrix which can be thought of as an operator on functions defined on vertices of G.

   Let $\mathbf{y}_0, \ldots, \mathbf{y}_d$ be the solutions of Equation (3.38), ordered according to their eigenvalues with $\mathbf{y}_0$ having the smallest eigenvalue (in fact zero since the Laplacian has always a zero eigenvalue). The image of $\mathbf{x}_i$ under the embedding into the lower dimensional space $\mathbb{R}^d$ is given by $(\mathbf{y}_{1i}, \ldots, \mathbf{y}_{di})$.

Hessian Eigenmaps (HLLE). This method [Dono 03b] is derived from a conceptual framework of local isometry in which the manifold $\mathscr{M}$, viewed as a Riemannian submanifold in the Euclidean space $\mathbb{R}^{\mathcal{D}}$, is locally isometric to an open, connected subset $\mathbf{y}$ of Euclidean space $\mathbb{R}^d$. This algorithm is then an extension of the Isomap one, but in principle, for the HLLE algorithm, $\mathbf{y}$ does not have to be convex.

A quadratic form $\mathcal{H}(f) = \int_{\mathscr{M}} \|\nabla_f^2(\mathbf{x})\|_F^2 \partial\mathbf{x}$ defined on function $f : \mathscr{M} \to \mathbb{R}$, where, ∇_f^2 denotes the Hessian of f, and $\mathcal{H}(f)$ averages the Frobenius norm of the Hessian over $\mathscr{M}$ (see [Dono 03b] and Section A.1 for more details of these definitions). To define the Hessian, orthogonal coordinates are used on the tangent planes of $\mathscr{M}$.

The algorithm is based in the observation that if $\mathscr{M}$ is truly locally isometric to an open connected subset of $\mathbb{R}^d$, then, $\mathcal{H}(f)$ has a $(d+1)$-dimensional null space, consisting of the coordinates. Hence, the isometric coordinates can be recovered up to a linear isometry.

HLLE can be viewed as a modification of the LLE method, while the theoretical framework can be seen as a modification of the Laplacian Eigenmaps one, where a quadratic form based on the Hessian substitutes the Laplacian.

A space $\mathbf{y} \subset \mathbb{R}^d$ is assumed with a smooth mapping $\Phi : \mathbf{y} \to \mathbb{R}^{\mathcal{D}}$, with $d < \mathcal{D}$. The image $\mathscr{M} = \Phi(\mathbf{y})$ is called the manifold, although from the viewpoint of manifold theory it is actually a special case of a single coordinate patch.

The vector $\mathbf{y}$ can be thought of as some control points underlying a measuring device, and the manifold as the enumeration $\mathbf{x} = \Phi(\mathbf{y})$ of all possible measurements as the points $\mathbf{y}$ changes. Thus the mapping Φ associates points to measurements.

The aim is that, given points $\mathbf{x}_i$, we manage to recover the mapping Φ and points $\mathbf{y}_i$. As stated, this is an ill-posed problem, since Φ is one solution, and if $\Psi : \mathbb{R}^d \to \mathbb{R}^{\mathcal{D}}$ is a morphing of the Euclidean space $\mathbb{R}^d$, the combined mapping $\Phi \circ \Psi$ is another solution.

Two results are the basis of the HLLE algorithm:

Theorem 3.6. ***[Dono 03b]***. *Suppose $\mathscr{M} = \Phi(\mathbf{y})$ where $\mathbf{y}$ is an open, connected subset of $\mathbb{R}^d$, and Φ is a locally isometric embedding of $\mathbf{y}$ into $\mathbb{R}^{\mathcal{D}}$. Then, $\mathcal{H}(f)$ has a $(d+1)$-dimensional null space consisting of the constant function and a d-dimensional spaced of functions spanned by the original isometric coordinates.*

Corollary 3.7. ***[Dono 03b]*** *Under the same assumptions as the previous Theorem, the isometric coordinates $\mathbf{y}$ can be recovered, up to a rigid motion, by identifying a suitable basis for the null space of $\mathcal{H}(f)$.*

Practical approach of the HLLE. The setting where sampled data S_X is lying on $\mathscr{M}$, and the underlying parametrization Φ and parameter settings $\mathbf{y}_i$ is to be recover, at least up to a rigid motion. Theorem (3.6) and Corollary (3.7) suggest the following algorithm, which is based on the original LLE algorithm. The algorithm takes as input the set S_X and parameters d and k or ε are to be set:

- Let $\mathcal{N}_i$ denote the collection of those neighbors according to the chosen method. For each neighborhood $\mathcal{N}_i$, $i = [\![n]\!]$, a $k \times n$ matrix M^i is formed with rows consisting of the recentered points $\mathbf{x}_j - \bar{\mathbf{x}}_i$, $j \in \mathcal{N}_i$, where $\bar{\mathbf{x}}_j = \text{Ave}\{\mathbf{x}_j : j \in \mathcal{N}_i\}$.
- Obtain tangent coordinates: Perform a singular value decomposition of M^i, getting matrices U, D, and V; U is $k \times \min(k,n)$. The first columns of U give the tangent coordinates of points in N_i.
- Develop Hessian estimator: Infrastructure is derived for least-squares estimation of the Hessian. Basically, this is a matrix H^i with the property that if f is a smooth function $f : X \to \mathbb{R}$, and $f_i = (f(\mathbf{x}_i))$, then the vector $\mathbf{v}^i$ with entries that are obtained from f by extracting those entries corresponding to points in the neighborhood $\mathcal{N}_i$; then, the matrix vector product $H^i\mathbf{v}^i$ gives a $d(d+1)/2$ vector whose entries approximate the entries of the Hessian matrix, $\left(\frac{\partial f}{\partial U_i \partial U_j}\right)$.
- Develop quadratic form: Build a symmetric matrix $\mathcal{H}_{ij}$ so that:

$$\mathcal{H}_{ij} = \sum_l \sum_r (H^l)_{r,i}(H^l)_{r,j}.$$

  where H^l is the $d(d+1)/2 \times k$ matrix associated with estimating the Hessian over neighborhood $\mathcal{N}_l$, where rows r correspond to specific entries in the Hessian matrix and columns i correspond to specific points in the neighborhood.
- Find Approximate Null Space. Perform an eigenanalysis of $\mathcal{H}$, and identify the $(d+1)$-dimensional subspace corresponding to the $(d+1)$ smallest eigenvalues. There will be an eigenvalue zero associated with the subspace of constant functions, and the next d eigenvalues will correspond to eigenvectors spanning a d-dimensional space $\widehat{V_d}$ in which our embedding coordinates are to be found.
- Find Basis for Null Spaces. Select a basis for $\widehat{V_d}$ which has the property that its restriction to a specific fixed neighborhood $\mathcal{N}_0$ (the neighborhood may be chosen arbitrarily from those used in the algorithm) provides and orthonormal basis. The given basis has basis vectors $w_1, \ldots, w_d$; these are the embedding coordinates.

Strenghts

- Asymptotic convergence: for data sampled from a submanifold that is isometric to an open, connected subset of Euclidean space, HLLE will recover the subset up to rigid motion.
- No convexity assumption: convergence is obtained for a larger class of manifolds than Isomap, handling in this manifolds with *holes*.

There exists other graph based algorithms that do not belong directly to the machine learning field and will not be developed here, see [Gail 08].

Quality measures for dimensionality reduction methods

For any of these methods, there is still the remaining question of how to measure the quality of the proposed projection. Several works have been done in this direction where either Voronoi cells [Aupe 07] are used; measure of the conservation of local distances or topological characteristics; count of the number of intrusions and extrusions [Lee 09] and a priori knowledge can be also included.

3.5. Shortcuts problems and their influence in graph based methods

In the previous sections a brief description of the most usual algorithms for non-linear dimensionality reduction was done. Many of these methods are based on the construction of a neighborhood graph, but in the presence of ***shortcuts*** (union of two points whose distance measure along the submanifold is actually large), the resulting embedding will be unsatisfactory. Indeed, one important drawback is the fact that the underlying manifold cannot be estimated if the nearest neighborhood graph is not properly defined and contains ***shortcuts***. That is, two points are considered to be neighbors whereas they are actually far away from each other in the sense of measured distance along the manifold, as shortcuts will cause underestimation of real distances between points. This is explained by the fact that the main goal of most dimensionality reduction algorithms is to preserve the Euclidean distance between the nearest neighbors defined by the graph. The algorithm is constrained to keep these distances in the new embedding, but these can be constraints not achievable in a smaller space. This section proposes an algorithm to correct wrong graph connections based on the tangent subspace of the manifold at each point. This leads to the estimation of the proper and adaptive number of neighbors for each point in the dataset. Experiments show graph construction improvement.

Local approximation techniques may have good results, but if the distribution along the submanifold is not uniform or dense enough, the adjacency graph will be either not connected or shortcuts will appear.

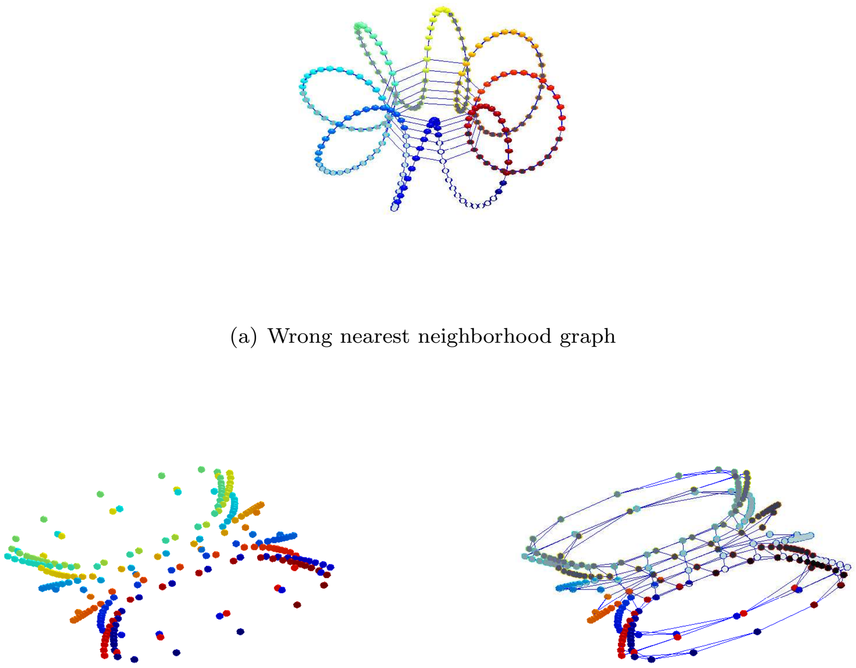
3.5.1 Problem formulation and illustration

Since many methods are graph-based, the matter of interest in this section is the correct graph construction, where the geodesic distance between every pair of points approximates properly its real distance along the manifold. If the graph is not properly built, the distances will not be correctly approximated and the results using any graph method will be unsatisfactory.

For the toroidal helix example in Figure 3.6(a), if a neighborhood is built with the k-nearest neighbors method with $k = 3$, the resulting neighborhood graph turns as depicted in Figure 3.7(a), where far points along the manifold are connected because the Euclidean distance is short. If this graph is directly used with a graph-based dimensionality reduction method, the resulting embedding will be as in Figure 3.7(b), where the imposed neighborings will be kept as shown in the Graph 3.7(c).

This problem will be denoted as ***shortcuts***, because the real distance between two points is no longer based on the manifold form but on some additional paths that will shorten its distance.

Problem 3.8 (Shortcuts Problem). *Let $S_X \subset \mathscr{M}$ be a subset of a manifold, and let $\mathcal{N}_i \subset S_X$ be a subset of nearest neighbors of $\mathbf{x}_i \in S_X$ defined according to the distance measured along the manifold. Let the longest distance between $\mathbf{x}_i$ and all points in $\mathcal{N}_i$ be Δ, that is, $\Delta = \max_{\mathbf{x}_l \in \mathcal{N}_i}\{\delta(\mathbf{x}_i, \mathbf{x}_l)\}$, with δ the distance along the manifold. If an approximation of $\mathcal{N}_i$ is done*



(a) Wrong nearest neighborhood graph

(b) Resulting projection in a space with reduced dimensionality

(c) Resulting projection in a space with reduced dimensionality. Neighborhood graph is shown in the new space.

Figure 3.7: Example of a wrong neighborhood assignation

by a subset $N_i \subset S_X$, then, there is a shortcut in $\mathbf{x}_i$ if there exists a point $\mathbf{x}_j \in N_i$ such that $\delta(\mathbf{x}_i, \mathbf{x}_j) > \Delta + \epsilon$, for a positive tolerance ϵ.

3.5.2 Approaches in the literature to solve the shortcuts problem

Choi [Choi 07] developed an algorithm to remove noisy points that produce shortcuts. For each point $\mathbf{x}_i$ the total flow is defined as the number of shortest paths passing through $\mathbf{x}_i$. This criterion for removing points might delete useful points and might miss several edges that should be removed. A similar approach was used by Yang [Yang 06]. Other approaches are based on the construction of minimal spanning trees [Zhao 06] which avoids having cycles, but will not necessarily respect intrinsic topological aspects. Finally, the work of Mekus [Meku 06] and Wang [Wang 05] are based on the estimation of the tangent subspace at each point independently.

We propose a method to correct neighborhood graphs using this last approach of tangent subspaces. One important difference is the fact that the estimation of the tangent space is done taking into consideration tangent spaces on neighboring points and therefore, respecting continuity in the manifold.

3.5.3 Our approach: Manifold and Tangent Subspace

By definition, a manifold is a differentiable surface, therefore, if we have the k nearest neighbors for each point, and the data lies on a connected submanifold $\mathcal{M} \in \mathbb{R}^d$, assuming that $d < k$, the tangent subspace to each point can be estimated from its k nearest neighbors. If additionally, the maximum curvature θ of the manifold is known, analysis of nearest neighbors can be done regarding the Euclidean distance with respect to a point $\mathbf{x}_i$ (which should be smaller than a value ϵ deduced from θ), and the angle deviation with respect to the tangent subspace $\mathcal{P}_i$ at $\mathbf{x}_i$ (which should be smaller than θ). The idea exploited in our solution is as follows: instead of considering a fix number of nearest neighbors using the k-NN algorithm or a fix radius ε of enclosing sphere for all samples, our strategy is to estimate an adaptive number of neighbors k_i for each sample $\mathbf{x}_i$ according to the density of points in its neighborhood and some mild assumptions on the manifold where these data lay.

Let $\mathcal{N}_i = \{\mathbf{x}_j | \mathbf{x}_j \text{ is nearest neighbor of } \mathbf{x}_i\}$ with $|\mathcal{N}_i| = k_i$ be the set containing the k_i nearest neighbors of $\mathbf{x}_i$, that is, for each sample $\mathbf{x}_i$, the number of nearest neighbors might be different.

For all neighborhood $\mathcal{N}_i$ $i = [\![n]\!]$, define $I_i = \{j | \mathbf{x}_j \in \mathcal{N}_i\}$, as the set of indexes of points belonging to neighborhood $\mathcal{N}_i$, which will be denoted as $I_i = \{i_1, i_2, ..., i_{k_i}\}$. Finally, define a matrix A_i of the form $A_i = [\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, ..., \mathbf{x}_{i_{k_i}}] \in \mathbb{R}^{D \times k_i}$ for all $i = [\![n]\!]$.

The implemented algorithm iterates between obtaining an estimation of the set $\mathcal{N}_i$ and the approximation of the tangent subspace $\mathcal{P}_i$ at $\mathbf{x}_i$ defined by $\mathcal{N}_i$.

3.5.4 Tangent subspace estimation and neighborhood estimation

If the Euclidean distance in the original space of a point $\mathbf{x}_i$ to its nearest neighbors $\mathcal{N}_i$ is a close approximation of the distance along the manifold, the the neighborhood is well defined. A plane $\mathcal{P}_i$ passing through $\mathbf{x}_i$ can be defined with an orthogonal vector $\mathbf{w}_i$ and a bias b_i:

$$\mathcal{P}_i(\mathcal{N}_i) = \{\mathbf{x} | \langle \mathbf{x}, \mathbf{w}_i \rangle + b_i = 0\}$$

Since the plane must pass through $\mathbf{x}_i$, the distance of $\mathbf{x}_i$ to the plane must be zero, that is

$$\langle \mathbf{x}_i, \mathbf{w}_i \rangle + b_i = 0.$$

Additionally, if the set of nearest neighbors is well defined, the orthogonal vector $\mathbf{w}_i$ must satisfy $A_i^\top \mathbf{w}_i + b_i \mathbb{1} = \mathbf{0}$, with $\mathbb{1}, \mathbf{0} \in \mathbb{R}^{k_i}$ vectors of ones and zeros, respectively. In order to ease the calculations to find $\mathbf{w}_i$, a similar condition will be searched: the projections of the nearest neighbors along the orthogonal vector should be at the same level, for example $A_i^\top \mathbf{w}_i + \mathbb{1} = \mathbf{0}$. The best approximation to this condition can be obtained via the pseudoinverse: $\mathbf{w}_i = \left(A_i^\top\right)^+ \mathbb{1}$.

Given a set of nearest neighbors resulting for each point, the approximation to the tangent subspace will be given by the following elements:

$$\mathbf{w}_i = \frac{(A_i^\top)^+ \mathbf{1}}{||(A_i^\top)^+ \mathbf{1}||_2} \text{ and } \; b_i = -\langle \mathbf{x}_i, \mathbf{w}_i \rangle \tag{3.39}$$

where $A_i^{\top +}$ represents the pseudo-inverse of matrix $A_i^\top$ and unicity of the orthogonal vector was added..

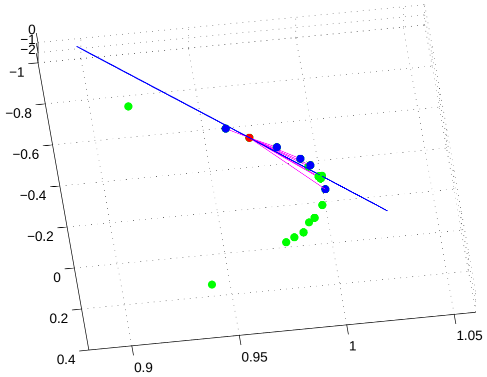


Figure 3.8: $\mathcal{P}_i$ (blue straight line) at $\mathbf{x}_i$ (red point) using $\mathcal{N}_i$ (blue points).

Figure 3.8 shows the estimated tangent plane, which in this case is a straight line (in blue) at the red point, estimated with the set of nearest neighbors, mark in blue. Vice versa, if we have an adjusted tangent subspace $\mathcal{P}_i$ at $\mathbf{x}_i$ and an initial neighborhood $\mathcal{N}_i^0$, we can measure the Euclidean distance ϵ_{ij} of $\mathbf{x}_j \in \mathcal{N}_i^0$ to $\mathbf{x}_i$ and the deviation angle θ_{ij} with respect to $\mathcal{P}_i$ as follows

$$\epsilon_{ij} = ||\mathbf{x}_j - \mathbf{x}_i||_2 \qquad \theta_{ij} = \arcsin \frac{|\,\langle \mathbf{w}_i, (\mathbf{x}_j - \mathbf{x}_i)\rangle|}{||\mathbf{x}_j - \mathbf{x}_i||_2}. \tag{3.40}$$

Then, neighborhood $\mathcal{N}_i$ can be updated as follows:

$$\mathbf{x}_j \in \mathcal{N}_i \quad \text{if} \quad \epsilon_{ij} < \epsilon \quad \text{and} \quad \theta_{ij} < \theta^0 \quad \forall\, \mathbf{x}_j \in \mathcal{N}_i^0. \tag{3.41}$$

where θ^0 is the maximum curvature input by the user and ϵ is estimated by the algorithm (see step (4) in Section (3.5.5)). Otherwise, $\mathbf{x}_j$ is removed from $\mathcal{N}_i^0$. An example of the form of the admissible area is depicted in Figure 3.9, the possible neighbors of the red points are the ones that lie in the light blue area. *Far* or not nearest neighbors points will lie in the light pink area.

Propagation of the tangent subspace estimation

Having a reliable estimation of $\mathcal{P}_i$ and $\mathcal{N}_i$ for point $\mathbf{x}_i$, this subspace is a good approximation of $\mathcal{P}_j$ for a $\mathbf{x}_j \in \mathcal{N}_i$. Then, $\mathcal{P}_i$ can be used to remove faulty nearest neighbors in $\mathcal{N}_j^0$ by using a relaxed parameter $\theta > \theta^0$ in Eqs. (3.41). The calculation of θ is explained in Algorithm 6. Remaining points will form set $\mathcal{N}_j$ which will allow us to calculate $\mathcal{P}_j(\mathcal{N}_j)$ using Eqs. (3.39).

Figure 3.10 shows how the tangent plane at each point can help to determine if there are faulty connections or not. Figure 3.10(a) depicts a correct neighborhood of the red point an Figure 3.10(b) the corresponding tangent plane of the red point (in blue) and the tangent planes of its nearest neighbors (in cyan), all of them are similar. Nevertheless, if there is a faulty set of neighbors like illustrated in Figure 3.10(c), the tangent planes of the red point (in blue) are not similar with the ones of its nearest neighbors (in cyan).

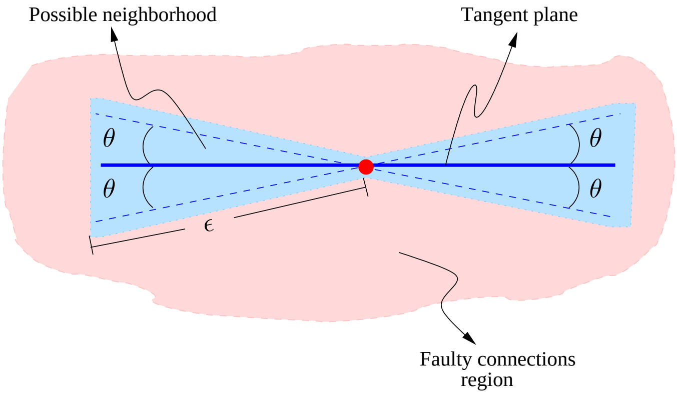
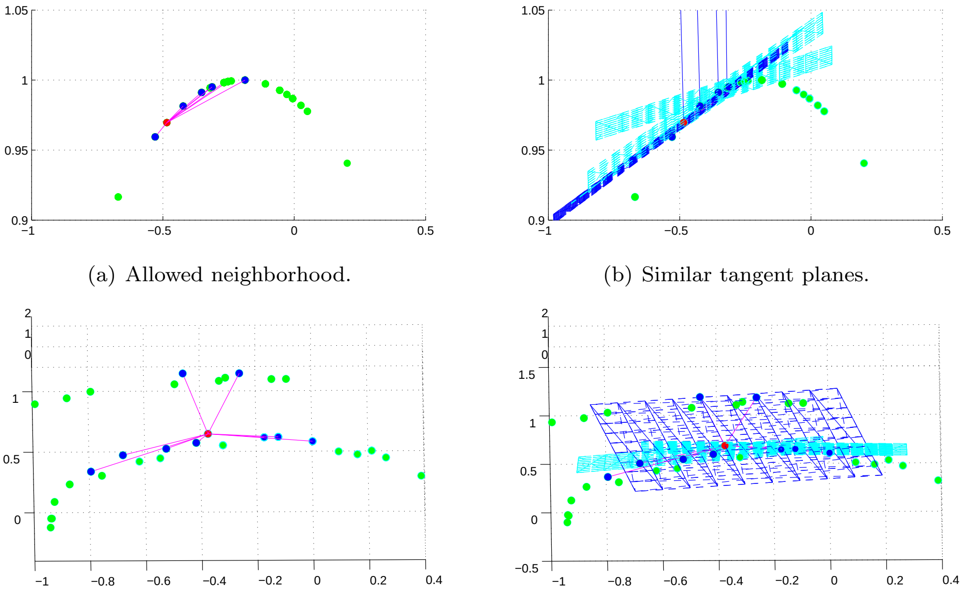


Figure 3.9: Admissible neighborhood area for red point. Graph connections leading to a point in the light pink area will be eliminated.



(a) Allowed neighborhood.

(b) Similar tangent planes.

(c) Faulty neighborhood.

(d) Different tangent planes.

Figure 3.10: Illustration of the use of tangent planes to detect faulty connections. Nearest neighbors (blue points) of red point. When there is no faulty connection, its tangent plane (in blue) is similar to the one of its neighbors (in cyan).

Finding a starting point

The calculation of the tangent subspace at each point $\mathbf{x}_i$ is done by propagation of the tangent subspace $\mathcal{P}_i$ at point $\mathbf{x}_i$ to its neighbors. This can be done only if we know that $\mathcal{P}_i$ is a reliable approximation. Therefore, the departure point is an important matter.

A first approximation to find $\mathcal{N}_i$, can be done by using the k nearest neighbors of each point, then each $\mathcal{P}_i$ will be estimated with these neighborhoods. If at a point $\mathbf{x}_i$ and at all its k-nearest neighbors in $\mathcal{N}_i$ a close approximation of the tangent subspace was found, the perpendicular vectors to their tangent subspaces should be all quite similar. Therefore, the chosen starting point $\mathbf{x}_{t_0}$ is the one whose orthogonal vector $\mathbf{w}_{t_0}$ maximizes the cost function

$$C(\mathbf{w}_i, I_i) = \frac{1}{k_i} \sum_{j \in I_i} |\langle \mathbf{w}_i, \mathbf{w}_j \rangle|. \tag{3.42}$$

If there is a shortcut at point $\mathbf{x}_i$, the estimated subspace at this point will be biased, and this subspace will not have the same orientation as its nearest neighbors, giving a smaller value in Equation (3.42).

3.5.5 Algorithm Description

Initialization phase

This phase consists in finding a point $\mathbf{x}_{t_0}$ located in a low curvature zone. So further tangent subspace approximations based on $\mathcal{P}_{t_0}$ are reliable. Let $\angle(\mathbf{x}, \mathbf{y})$ be the angle between vector $\mathbf{x}$ and vector $\mathbf{y}$, the algorithm can be initialized as follows:

Algorithm 6 Pseudo-code for the tangent plane approximation and initial neighborhood definition

```
Input: Samples set $S_X = \{\mathbf{x}_i\}_{i=[\![n]\!]}$, curvature $\theta_0$ and the maximum number of neighbors $k$, set $k_i = k$.
    Set $\mathcal{N}_i^0 \leftarrow \{\mathbf{x}_j | \mathbf{x}_j$ is a $k$-nearest neighbor of $\mathbf{x}_i\}$, $\forall i = [\![n]\!]$.
    Set $\theta_{ij} = \infty,\ i, j = [\![n]\!]$, which will be later updated with the deviation angle of $\mathbf{x}_j$ to stable tangent subspaces $\mathcal{P}_i$.
Output: initial $\mathcal{N}_i, \mathcal{P}_i$.
Estimate the initial tangent subspace $\mathcal{P}_i^0(\mathcal{N}_i^0) = \{\mathbf{w}_i, b_i\}$, $i = [\![n]\!]$ using Eq. (3.39).
Find $\mathbf{x}_t$ with the best tangent subspace approximation: the one that maximizes Eq. (3.42).
Set $\epsilon = ||\mathbf{x}_t - \mathbf{x}_{t_{k+1}}||$, where $\mathbf{x}_{t_{k+1}}$ is the $(k+1)$-nearest neighbor of $\mathbf{x}_t$. This will be the maximum radius of $\epsilon$-balls.
for all $\mathbf{x}_i$ reduce its neighborhood $\mathcal{N}_i$ using the $\epsilon$-ball.
Set $I_{done} \leftarrow \{t\}$ the index of points $\mathbf{x}_i \in S_X$ having a stable tangent space $\mathcal{P}_i$.
Select nearest neighbors $\mathcal{N}_t \in \mathcal{N}_t^0$ using tangent plane $\mathcal{P}_t^0$ and rule (3.41).
Update $\mathcal{P}_t(\mathcal{N}_t)$ using Eq. (3.39).
Set deviation angle $\theta_{tj} = \angle(\mathbf{w}_t, \mathbf{x}_j - \mathbf{x}_t)$,   for all  $j \notin I_{done}$.
```

Successive Approximation phase

This phase uses reliable estimations of tangent subspaces to make approximations at other points. Graph is covered considering deviation angles. The successive approximation scheme is described by Algorithm 7 which uses the outputs of the initialization algorithm 6. Again $\angle(\mathbf{x}, \mathbf{y})$ denotes the angle between vector $\mathbf{x}$ and vector $\mathbf{y}$ and $\angle(\mathbf{x}, P)$ denotes the angle between vector $\mathbf{x}$ and plane P.

In the algorithm, θ can be initialized with an approximation of a stable point, but through the experiments, it was observed that a constant $\theta = 20°$ turns to be appropriate.

Algorithm 7 Pseudo-code for the shortcut correction via tangent hyper-planes

```
Input: deviation angles $\theta_{ij}$, initial neighborhood $\mathcal{N}_i$, $I_{done}$
Output: Clean Neighborhoods $\{\mathcal{N}_i\}_{i=[\![n]\!]}$.
1: Choose $t = \text{argmin}_j\{\theta_{ij} |\ i \in I_{done},\ j \in \mathcal{N}_i\}$, so that for a point $\mathbf{x}_s,\ s \in I_{done}$, $\mathbf{x}_t \in \mathcal{N}_s$ has the
   minimum deviation to tangent subspace $\mathcal{P}_s$, let $s = \text{argmin}_i\{\theta_{it},\ i \in I_{done}\}$.
2: Select nearest neighbors $\mathcal{N}_t \in \mathcal{N}_t^0$ using Eqs. (3.41) with $\mathcal{P}_t = \mathcal{P}_s$ and $\theta = \theta + \angle(\mathbf{x}_t - \mathbf{x}_s, \mathcal{P}_s)$.
3: Update $\mathcal{P}_t(\mathcal{N}_t)$ with Eqs. (3.39).
4: Set $I_{done} = I_{done} \cup \{t\}$.
5: Set $\theta_{tj} = \angle(\mathbf{w}_t, \mathbf{x}_j - \mathbf{x}_t),\ j \notin I_{done}$. Set $\theta_{jt} = \infty,\ j \in I_{done}$ to avoid selecting again $\mathbf{x}_t$.
6: If $|I_{done}| < n$, go to step (1), else go to (7).
7: If the graph is not connected, make a single component based on $\frac{\theta_{ij}}{\epsilon_{ij}}$.
```

3.5.6 Experimental Results

To validate our algorithm, we applied it on the swissroll example (Figure 3.11) which real underlying dimensionality lies in $\mathbb{R}^2$. We generated a lightly sparse dataset with 300 points and compared our results to k-NN and the total flows algorithm [Choi 07]. Parameter k was set to values between 8 and 16, giving in our case similar results. Curvature θ was set to 20 degrees.

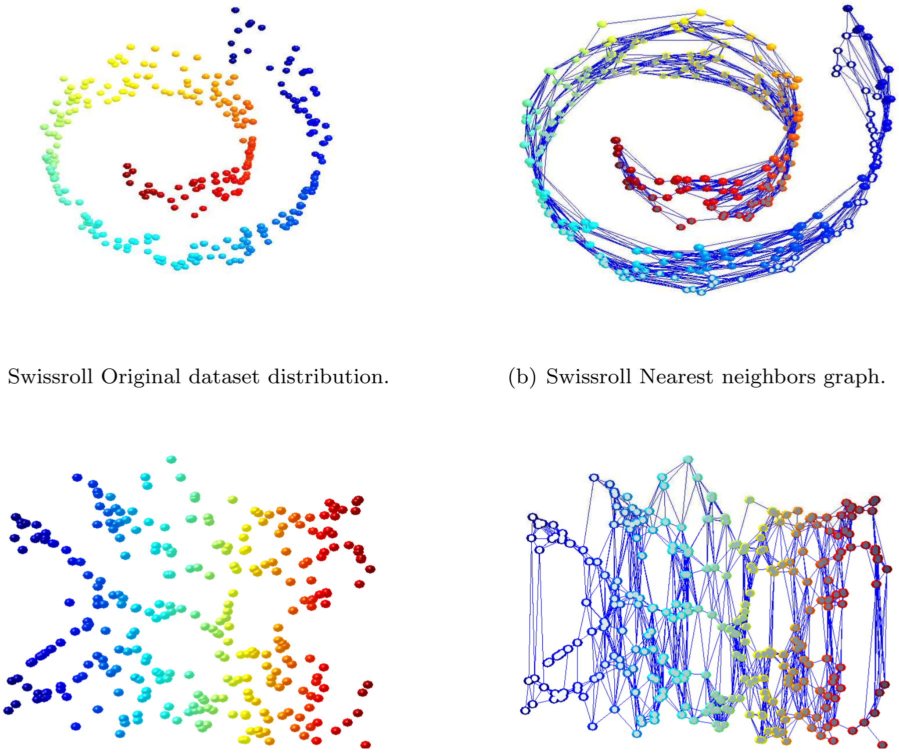
It has to be noticed that a swissroll generated with only 300 points is a very sparse example and there are therefore more difficulties to handle it. The expected resulting embedding would look like in Figure 3.11(c), which conserves local distances as set by the nearest neighborhood graph in Figures 3.11(b) and 3.11(d).

The tangent hyper-plane method was tested with the swiss-roll dataset. Three methods are shown in Figure 3.12, the first Figure 3.12(a) is the resulting neighborhood graph with the k-NN method, with $k = 8$, from here, shortcuts are detected using the flow method in Figure 3.12(b), it managed to clean the graph as the tangent plane method in Figure 3.12(c) but the embedding result are nicer with the last one.

The second line in Figure 3.12 shows the resulting embedding with the given graph by the k-NN method (Figure 3.12(d)), the flows method in Figure 3.12(e) and the tangent plane in Figure 3.12(f) using the Isomap method, while the third line shows the embedding results using the LLE method. In both cases, the embedding resulting with the tangent planes technique seems more appropriate.

An example were 10 nearest neighbors were used as the base is shown in Figure 3.13 the first line of figures shows as in the previous set of figures the resulting nearest neighbors graph, it can be seen that the tangent plane technique has successfully removed all shortcuts while the flows method did not. The second row, Figures 3.13(d), 3.13(e) and 3.13(f) shows the resulting embedding with the Isomap method while the last three figures, the resulting embedding with the LLE method. The number of nearest neighbors were increased up to 20, the tangent plane manage to successfully clean the graph.

Figure 3.14 illustrates the method with a dataset of rendered faces. In Figure 3.14(a), shortcuts are encountered, the two faces that in the original space are very similar and therefore impeding the dimensionality reduction are shown in Figure 3.14(b). It can be seen that these two faces are lacking of contrast and the only differentiations are on the left and right border. With the tangent plane techniques, faces that appear to be close in the original space but not in reality, are detected, resulting in the reduced representation in Figure 3.14(c). Using tangent planes, we manage to get rid of shortcuts (false neighbors) that might appear by using a large k. Figure 3.14(d) shows the kind of faces that can be found along the *unfolded* graph. If the tangent subspace at each point is estimated and the neighborhood graph is built using this subspace, we are able to reduce the shortcuts in the original graph construction with the k-NN method. Additionally experiments showed that if we add Gaussian noise with zero mean and variance equal to .1 to this dataset, our algorithm is robust.



(a) Swissroll Original dataset distribution.

(b) Swissroll Nearest neighbors graph.

(c) Swissroll Embedding.

(d) Swissroll Conservation of NN in the embedding.

Figure 3.11: Swiss roll example and its NN graphs.

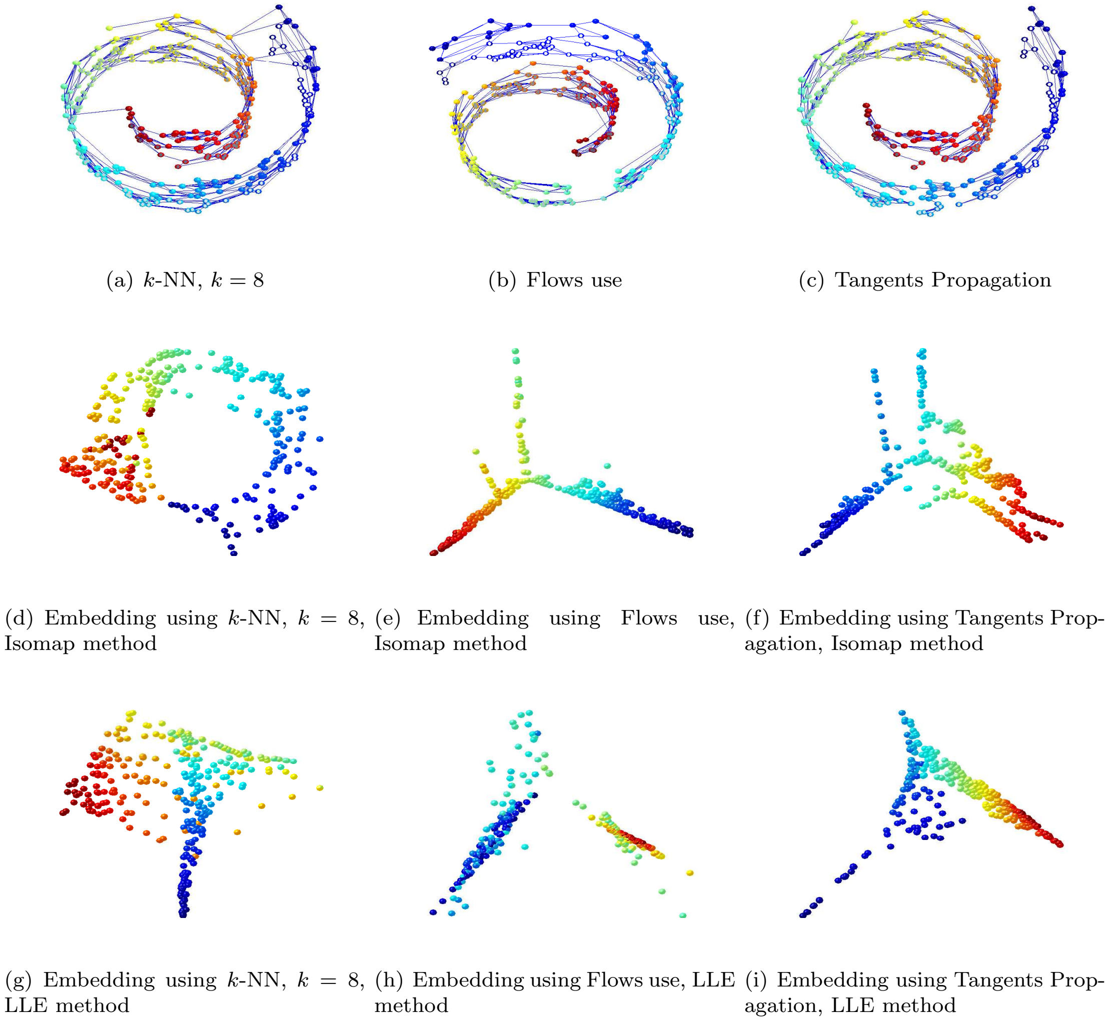
3.5.7 Dimensionality Reduction Conclusions

The estimation of tangent subspaces at each point can improve the structure of an initial graph given a maximum number of nearest neighbors. The use of neighborhood information helps to improve the estimation of tangent subspaces. We propose an algorithm that helps to detect the presence of shortcuts in a graph. Future work includes automatic estimation of the manifold curvature and use of this technique in real data.

3.6. Conclusions

Manifold theory has been successfully introduced in the learning theory, in this chapter, two application were shown: classification under a semi-supervised learning framework and dimensionality reduction under the manifold assumption.

For the semi-supervised learning, a manifold assumption is used in order to discover dense areas and nearest neighbors. A model can be successfully built with the use of this information despite the lack of labels in many data points. The arisen problem was the fact that the obtained decision function depends on the complete set of points, resulting in a slow model for new points classification. This problem was tackled by introducing a L_1-norm which controls the number of active variables in the model. Using this technique, according to the associated parameter s to the L_1-norm, the number of active variables in the model is adjusted, so that it can be controlled



(a) k-NN, $k = 8$

(b) Flows use

(c) Tangents Propagation

(d) Embedding using k-NN, $k = 8$, Isomap method

(e) Embedding using Flows use, Isomap method

(f) Embedding using Tangents Propagation, Isomap method

(g) Embedding using k-NN, $k = 8$, LLE method

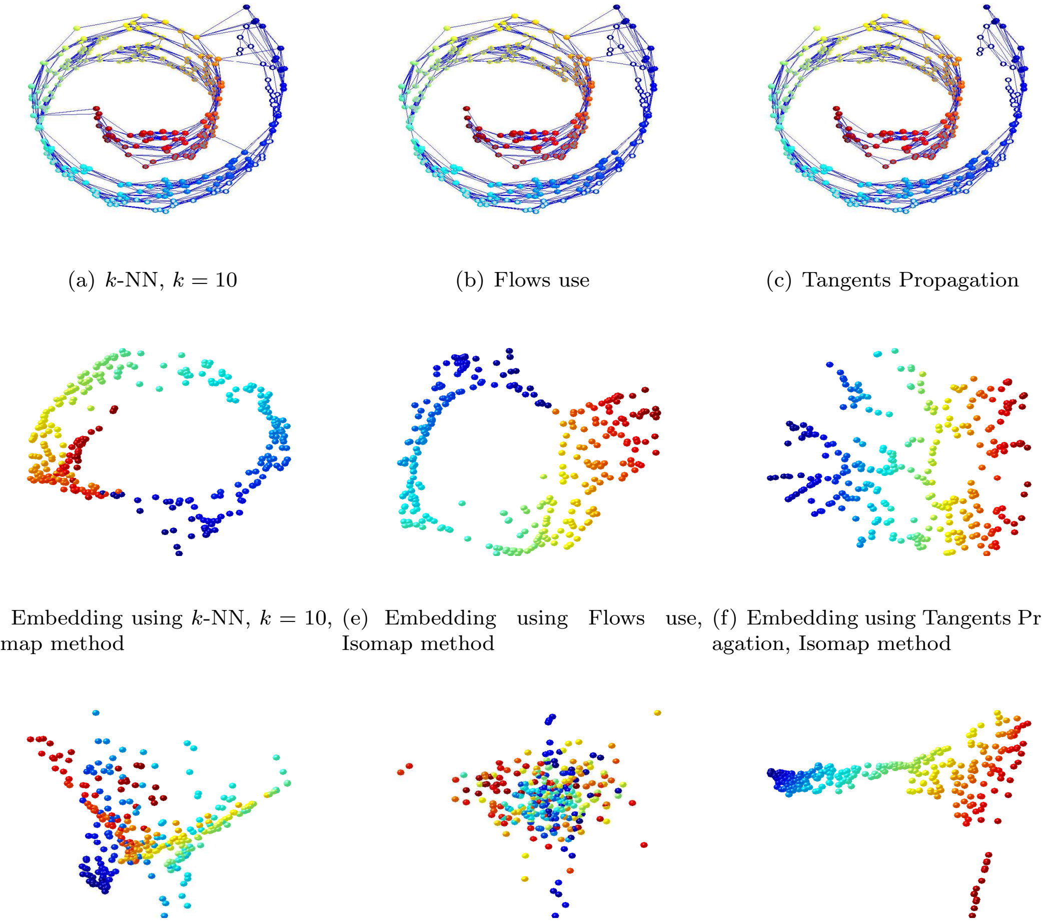
(h) Embedding using Flows use, LLE method

(i) Embedding using Tangents Propagation, LLE method

Figure 3.12: Resulting NN graph with different algorithms and its resulting embedding with the Isomap and LLE methods, $k = 8$.

by increasing or decreasing s. It was shown that a very similar generalization level can be reached as the original Laplacian SVM where $s = \infty$ but with significantly less data points.

The second part of the chapter analyzed dimensionality reduction methods based in neighborhood graphs, it was shown that if the neighborhood graph if not properly defined, these methods will fail to discover the real dimensionality. A solution to the problem of shortcuts was proposed taking advantage of the fact that the dataset lies in a manifold and that, locally, it can be considered like an Euclidean space and the tangent plane at each point can be approximated with the nearest neighbors as they can properly represent a plane given the manifold assumption. The use of this technique showed that, in many cases, most of the shortcuts can be eliminated, improving the results of the reduction methods.



(a) k-NN, $k = 10$

(b) Flows use

(c) Tangents Propagation

(d) Embedding using k-NN, $k = 10$, Isomap method

(e) Embedding using Flows use, Isomap method

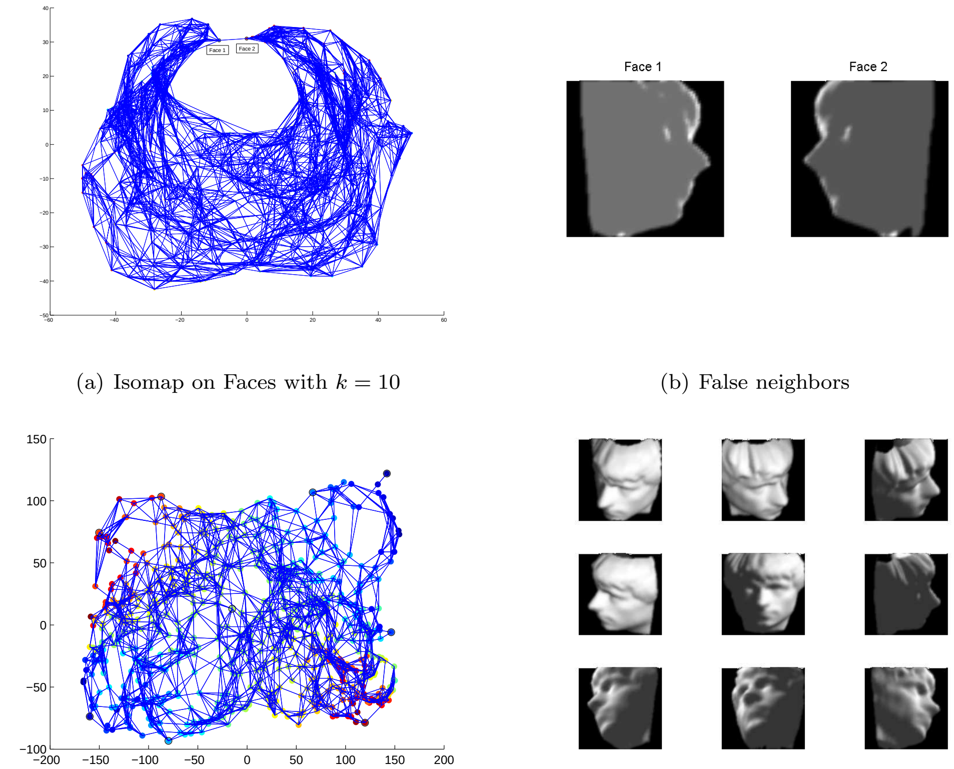
(f) Embedding using Tangents Propagation, Isomap method

(g) Embedding using k-NN, $k = 10$, LLE method

(h) Embedding using Flows use, LLE method

(i) Embedding using Tangents Propagation, LLE method

Figure 3.13: Resulting NN graph with different algorithms and its resulting embedding with the Isomap and LLE methods, $k = 10$.



(a) Isomap on Faces with $k = 10$

(b) False neighbors

(c) Isomap on Faces, Tangent planes $k_{max} = 10$

(d) Manifold Sample

Figure 3.14: Illustration of the algorithm use in the faces datasets.

Conclusions and Perspectives

Conclusions

Analysis of the Regularization Path

Along this work, several machine learning problems stated as multi-objective optimization problems were analyzed. Based on the optimality conditions, direct deduction of primal, dual and bi-dual problems can be done by the use of subdifferential operators on the Lagrangian function, their interest lies in the different resolution methods that can be applied to each of them and the form of the decision function. For example, the dual problem can be solved via quadratic solvers, while the bi-dual one explicitly contains the form of the decision function. The treated frameworks in supervised and semi-supervised learning for classification and ranking problems belong to the algorithms that hold the necessary conditions to have piecewise linear optimal solution as function of the regularization parameter. The extension to non-differentiable forms can be trivially deduced by using instead of the gradient, the more general subdifferential definition. It was shown that the piecewise linearity is helpful to efficiently calculate the complete regularization path. The advantage of this is the fact that the calculation time of the complete regularization path turns out to be in practice comparable to a few single resolutions of popular quadratic methods used to solve the dual problem for a particular regularization parameter value.

Extension of the Regularization Path

Under this setting, the regularization path was developed for the SVM ranking framework. The most important issue here encountered is the fact that the number of constraints is very large, leading to a very large regularization path, and a slow problem resolution. A proposition of constraints reduction was done which seems to be fairly satisfactory as it substantially reduces the calculation time with some accuracy lost. It was seen and analyzed that in general, the newly proposed constraint graph leads to coherent results as it can be compared to a weighted graph, where some relations have positive weights and other zero weights.

In a different approach aiming at calculation time reduction and parsimonious models for the ranking problem, the regularization function was changed from a L_2-norm to a L_1-norm, so that the number of involved components in the final chosen model can be controlled with an additional parameter, at the same time, the hinge-loss penalization function was substituted by a square one, decreasing the number of breakpoints in the regularization path, and leading therefore to a faster optimal set computation. The accuracy was slightly affected by this, but it has the advantage

that the solution is a lot sparser.
Finally, under the semi-supervised framework, sparsity was again included by the addition of the L_1-norm as a constraint of the Laplacian SVM framework. Equal accuracy rates can be achieved with considerably less variables, having the advantage that a regularization path can be also efficiently deduced.

Parameter selection

Even though the regularization path provides the complete solution set corresponding to each parameter value, the next natural issue is the model selection one. An analysis of the validation error curve was done, it was observed that for the problems belonging to the same family, the decision function has an hyperbolic relation with respect to the regularization parameter, so that the validation errors can be fast updated along the regularization path, that is, breakpoints for the change of the validation error (which is piecewise constant) can be given, this was named the validation path. Nevertheless, this is revealed not needed every time as a representative sampling of the parameter values can be done by using the resulting breakpoints from the regularization path calculation. It was shown that, indeed, the regularization path breakpoints induce a selection of the most interesting regularization parameter values which are to be explored and, with high probability, will contain the one that outputs the minimum of the validation error curve.
This analysis done under the training-validation-test sets setting can be straightforwardly extended to bootstrap and cross-validation frameworks by averaging the obtained validation error curves at each trial.
In the last section of this work, unsupervised learning was approached, particularly, graph methods for dimensionality reduction. The common issue of this methods is the construction of the graph, that is, the correct designation of neighbors for each point. In the case where there exists shortcuts, unsatisfactory results will be obtained when applying dimensionality reduction methods, where the degree of satisfactory projections can be measured as faulty distance, topological characteristics conservation or with a priori knowledge. If the manifold assumption is made in order to apply a method, then, we can also take advantage of the local linearity to iteratively estimate tangent hyperplanes at each point of the manifold, checking that these remain similar between neighbor points. Having an estimation of the tangent planes, most of the edges producing shortcuts can be detected by comparing the angular distance to the hyperplane and be removed. The used method to detect edges producing shortcuts manage to provide a sparser graph that will output better results than the initial one, usually calculated with the k-NN method.

Perspectives

As further research lines, double or multiple regularization paths could be considered. In the present work only the regularization parameter is moved. For instance, in the sparse Laplacian SVM, the parameter of the L_2-norm regularizer is fixed while the L_1 path is explored and vice versa. An extension to this work could be the efficient exploration of the surface or volume induced where different types of penalties are encoded in the learning problem. This problem boils down to an efficient exploration of a deformed *grid* when considering bi-dimensional parameter space. A way this grid exploration can be useful is to consider the exploration of the space of parameters formed by the regularization parameter and the kernel parameter in SVM framework. If exact calculation is not possible on bi-dimensional search spaces, approximations could be done if it is, for example, in the case of the kernel parameter.
With respect to the proposed reduced graph, the fact that it is the result of elementary operations and the encouraging obtained results with it give some clues in the sense that not all the information is needed to built a satisfactory model. The next question to be solved is to be able, for example, to efficiently use the reduced graph as a warm start sub-solution for the complete one for a faster convergence and for faster parameter search.

Given that the gradient-descent method leads to quite nice results, the stochastic gradient method for parameter search can be considered, this should normally overcome local optima in all directions and it has the advantage that it does not require differentiability of the objective function for its application because the subdifferential can be used instead of the derivative.

For the neighborhood setting, a better approximation of the curvature of the manifold can be considered with second order projections. This will be able to reduce the problem of over cutting, that is, removing to many edges from the original neighborhood graph. The technique of tangent planes could also be extended to the out-of-sample problem, but most of the dimensionality reduction methods, there is no direct out-of-sample extension.

A Appendix

A.1. Mathematical Background

A large number of definitions would be needed to exhaustively complete the mathematical used element in the present work. For a larger in deep into algebraic and analysis definitions there are different references to consult [Fral 89, Bart 76, Rudi 76]. In general, the samples are supposed to belong to a vector space $\mathcal{X}$ or that they can be transformed into such a space. The following definitions will allow us to search for an appropriate decision function.

A.1.1 Sets, Spaces and Sequences

Definition A.1 (Extended real numbers)**.** *The **extended real numbers** to be the set:* $\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$ *with* $0 \cdot (\pm\infty) = (\pm\infty) \cdot 0 = 0$ *but* $((+\infty) - (+\infty))$ *not defined.*

Definition A.2 (Vector space)**.** *Let F be a field with a sum operation denoted $a+b$ and a multiplication operation denoted ab for all $a, b \in F$. A **vector space** over F (or F-vector space) consists of a set $\mathcal{X}$ with a closed binary operation $\oplus$ between elements of $\mathcal{X}$ together with a multiplicative operation $\cdot$ of each element of $\mathcal{X}$ by each element of F on the left, such that for all $a, b \in F$ and $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathcal{X}$ the following conditions are satisfied:*

- *The binary operation $\oplus$ is associative:* $\mathbf{x} \oplus (\mathbf{y} \oplus \mathbf{z}) = (\mathbf{x} \oplus \mathbf{y}) \oplus \mathbf{z}$.
- *There is an element $\mathbf{e}$ in $\mathcal{X}$ such that $\mathbf{e} \oplus \mathbf{x} = \mathbf{x} \oplus \mathbf{e} = \mathbf{x}$ for all $\mathbf{x} \in \mathcal{X}$. This element $\mathbf{e}$ is an **identity element** for $\oplus$ on $\mathcal{X}$.*
- *For each $\mathbf{x} \in \mathcal{X}$, there is an element $\mathbf{x}'$ in $\mathcal{X}$ with the property that $\mathbf{x}' \oplus \mathbf{x} = \mathbf{x} \oplus \mathbf{x}' = \mathbf{e}$. The element $\mathbf{x}'$ is an **inverse** of $\mathbf{x}$ with respect to the operation $\oplus$.*
- *The operation $\oplus$ is commutative: $\mathbf{x} \oplus \mathbf{y} = \mathbf{y} \oplus \mathbf{x}$, a commutative group is called an **Abelian group**.*
- *The binary operation $\cdot$ is closed: $a \cdot \mathbf{x} \in \mathcal{X}$.*
- *Multiplicative operations are associative: $a \cdot (b \cdot \mathbf{x}) = (ab) \cdot \mathbf{x}$.*
- *The distribute property holds: $(a+b) \cdot \mathbf{x} = (a \cdot \mathbf{x}) \oplus (b \cdot \mathbf{x})$ and $a \cdot (\mathbf{x} \oplus \mathbf{y}) = (a \cdot \mathbf{x}) \oplus (a \cdot \mathbf{y})$.*
- *The identity 1 of F for its multiplicative operation satisfies: $1 \cdot \mathbf{x} = \mathbf{x}$.*

Definition A.3 (Algebra)**.** *An **algebra** consists of a vector space $\mathcal{X}$ over a field F, together with a binary operation of multiplication $\otimes$ on the set $\mathcal{X}$ of vectors, such that for all $a \in F$ and $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathcal{X}$, the following conditions are satisfied:*

- $(a \cdot \mathbf{x}) \otimes \mathbf{y} = a \cdot (\mathbf{x} \otimes \mathbf{y}) = \mathbf{x} \otimes (a \cdot \mathbf{y})$
- $(\mathbf{x} + \mathbf{y}) \otimes \mathbf{z} = \mathbf{x} \otimes \mathbf{z} + \mathbf{y} \otimes \mathbf{z}$
- $\mathbf{x} \otimes (\mathbf{y} + \mathbf{z}) = \mathbf{x} \otimes \mathbf{y} + \mathbf{x} \otimes \mathbf{z}$

*$\mathcal{X}$ is an **associative algebra over** F if additionally it holds:*

- $\mathbf{x} \otimes \mathbf{y} \otimes (\mathbf{z}) = \mathbf{x} \otimes (\mathbf{y} \otimes \mathbf{z}) \quad$ *for all* $\quad \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathcal{X}$.

Definition A.4 (Convex set)**.** *A set S in a linear vector space is said to be convex if, given $\mathbf{x}_1, \mathbf{x}_2 \in S$, all points of the form $\alpha \mathbf{x}_1 + (1 - \alpha)\mathbf{x}_2$ with $0 \leq \alpha \leq 1$ are in S.*

This definition merely says that given two points in a convex set, the line segment between them is also in the set.

Definition A.5 (Cone)**.** *A set S is said to be a cone if for all $x, z \in S$ and $\alpha > 0$, we have $\alpha x + z \in S$*

Definition A.6 (Cauchy sequence)**.** *A sequence $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ in $\mathbb{R}^{\mathcal{D}}$ is said to be a **Cauchy sequence** in case for every $\varepsilon > 0$ there is a natural number $M(\varepsilon)$ such that for all $m, n \geq M(\epsilon)$, then $\|\mathbf{x}_m - \mathbf{x}_n\| < \varepsilon$.*

Any convergent sequence in $\mathbb{R}^{\mathcal{D}}$ is as Cauchy sequence.

Definition A.7 (Metric space)**.** *A set $\mathcal{X}$, whose elements we shall call points, is said to be a **metric space** if with any two points $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ there is associated a real number $d(\mathbf{x}, \mathbf{y})$ called the **distance** from $\mathbf{x}$ to $\mathbf{y}$, such that*

- $d(\mathbf{x}, \mathbf{y}) > 0$ *if* $\mathbf{x} \neq \mathbf{y}$; $d(\mathbf{x}, \mathbf{x}) = 0$.
- $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$.
- $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y})$ *for any* $\mathbf{z} \in \mathcal{X}$.

Any function with these properties is called a **distance function**, or a **metric**. A metric space $\mathcal{X}$ is said to be complete (or Cauchy) if every Cauchy sequence of points in $\mathcal{X}$ has a limit that is also in $\mathcal{X}$ or alternatively if every Cauchy sequence in $\mathcal{X}$ converges in $\mathcal{X}$.

Definition A.8 (Banach space)**.** *A complete **normed vector space**, that is a vector space that has a norm defined over its elements (see A.20), is called as **Banach space**.*

This means that a Banach space is a vector space $\mathcal{X}$ over the real or complex numbers with a norm $\| \cdot \|$ such that every Cauchy sequence (with respect to the metric $d(x, y) = \|x - y\|$) in $\mathcal{X}$ has a limit in $\mathcal{X}$. Since the norm induces a topology on the vector space, a Banach space provides an example of a topological vector space.
A simple consequence of the first two axioms of Definition A.20, positive homogeneity and the triangle inequality, is $\|\mathbf{0}\| = 0$ and thus $\|\mathbf{x}\| \geq 0$ (positivity).
A **pre-Hilbert space** is a Vector space $\mathcal{X}$ endowed with a **dot product** $\langle \cdot, \cdot \rangle : \mathcal{X} \times \mathcal{X} \to F$, with $F = \mathbb{R}$ or $F = \mathbb{C}$ that for all $a \in F$ and $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathcal{X}$ holds:

- $\langle \mathbf{x}, \mathbf{y} \rangle = \overline{\langle \mathbf{y}, \mathbf{x} \rangle}$ (the hermitian-symmetric property)
- $\langle \mathbf{x} + \mathbf{y}, \mathbf{z} \rangle = \langle \mathbf{x}, \mathbf{z} \rangle + \langle \mathbf{y}, \mathbf{z} \rangle$ (additivity in first argument)
- $\langle \mathbf{x}, \mathbf{y} + \mathbf{z} \rangle = \langle \mathbf{x}, \mathbf{y} \rangle + \langle \mathbf{x}, \mathbf{z} \rangle$ (additivity in second argument)
- $\langle \mathbf{x}, \mathbf{x} \rangle \geq 0$ (and equality only for $\mathbf{x} = 0$: positivity)

- $\langle a\mathbf{x}, \mathbf{y}\rangle = a\langle \mathbf{x}, \mathbf{y}\rangle$ (linearity in first argument)
- $\langle \mathbf{x}, a\mathbf{y}\rangle = \bar{a}\langle \mathbf{x}, \mathbf{y}\rangle$ (conjugate-linearity in second argument)

Definition A.9 (Hilbert space)**.** *A **Hilbert space** is a Banach space with a norm induced by an inner product.*

Definition A.10 (Positive Cone)**.** *Let P be a convex cone in a vector space $\mathcal{X}$. For $\mathbf{z}, \mathbf{y} \in \mathcal{X}$, we write $\mathbf{z} \geq \mathbf{y}$ (with respect to P) if $\mathbf{z} - \mathbf{y} \in P$. The cone P defining this relation is called the **positive cone** in X. The cone $N = -P$ is called the **negative cone** in $\mathcal{X}$ and we write $\mathbf{y} \leq \mathbf{z}$ for $\mathbf{y} - \mathbf{z} \in N$.*

For example, in $\mathbb{R}^n$ the convex cone

$$P = \{\mathbf{z} \in \mathbb{R}^n : \mathbf{z} = (e_1, e_2, ..., e_n); e_i \geq 0 \text{ for all } i\}$$

defines the ordinary positive orthant.

A.1.2 Functionals

In this annex, we study continuity properties of convex functionals [Schi 07].

Definition A.11 (Indicator Functional)**.** *Let $M \subseteq \mathcal{X}$ be nonempty. The **indicator functional of M** $\delta_M : \mathcal{X} \to \bar{\mathbb{R}}$ is defined by*

$$\delta_M(x) := \begin{cases} 0 & \text{if } x \in M \\ +\infty & \text{if } x \in \mathcal{X} \setminus M \end{cases}$$

It can be seen that δ_M is proper and convex if and only if M is nonempty and convex.

Definition A.12 (Effective domain)**.** *If $f : M \to \bar{\mathbb{R}}$ with M an nonempty subset of a vector space, then we call the **effective domain of** f the set*

$$Dom(f) = \{\mathbf{x} \in M | f(\mathbf{x}) < \infty\}.$$

Definition A.13 (Proper function)**.** *$f : \mathcal{X} \to \bar{\mathbb{R}}$ is said to be **proper** if $Dom(f) \neq \emptyset$ and $f(\mathbf{x}) > -\infty$ for each $\mathbf{x} \in \mathcal{X}$.*

Definition A.14 (Convex function)**.** *Let $M \subset \mathcal{X}$ be nonempty and convex. The function $f : M \to \mathbb{R}$ is called **convex** if*

$$f\big((\lambda\mathbf{x}) + (1 - \lambda\mathbf{y})\big) \leq \lambda f(\mathbf{x}) + (1 - \lambda)f(\mathbf{y}) \tag{A.1}$$

holds for all $\mathbf{x}, \mathbf{y} \in M$ and all $\lambda \in (0, 1)$ for which the right-hand side is defined, i.e., is not of the form $(+\infty) + (-\infty)$ or $(-\infty) + (+\infty)$. If (A.1) *holds with $<$ instead of $\leq$ whenever $\mathbf{x} \neq \mathbf{y}$, then f is called **strictly convex**.*

Definition A.15 (Lipschitz continuous)**.** *If $\mathcal{X}$ is a normed vector spaced, then the proper functional $f : \mathcal{X} \to \bar{\mathbb{R}}$ is said to be **locally Lipschitz Continuous**, or briefly **locally L-continuous**, around $\mathbf{x}_0 \in Dom(f)$ if there exist $\epsilon > 0$ and $\lambda > 0$ such that*

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq \lambda\|\mathbf{x} - \mathbf{y}\|| \quad \forall\, \mathbf{x}, \mathbf{y} \in B(\mathbf{x}_0, \epsilon).$$

Moreover, f is called locally L-continuous on the open subset D of $\mathcal{X}$ if f is locally L-continuous around each $\mathbf{x}_0 \in D$.

Definition A.16 (Lipschitz condition and locally Lipschitz)**.** *A function $f : \mathcal{X} \to (-\infty, +\infty]$ is said to satisfy the **Lipschitz condition** of rank K on a given set S provided that f is finite on S and satisfies*

$$|f(x) - f(y)| \leq K\|x - y\|$$

*A function f is said to be **locally Lipschitz** on S if f is Lipschitz near x for every $x \in S$.*

All $f \in C^1(U)$, continuous functions, U open, are locally Lipschitz in U

Theorem A.17 (Sandwich Theorem)**.** *Let $\mathcal{X}$ be a topological vector space and let $p, q : \mathcal{X} \to \bar{\mathbb{R}}$ be proper, convex and such that $-q(\mathbf{x}) \leq p(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$. Suppose further that:*

1. *Int(Dom(p)) $\cap$ Dom(q) $\neq \emptyset$ and p is continuous at some point of Int(Dom(p)) or*
2. *Dom(p) $\cap$ Int(Dom(q)) $\neq \emptyset$ and q is continuous at some point of Int(Dom(q)).*

Then there exists $v \in \mathcal{X}^*$ and $c \in \mathbb{R}$ such that

$$-q(\mathbf{x}) \leq \langle v, \mathbf{x} \rangle + c \leq p(\mathbf{x}) \quad \forall\ \mathbf{x} \in \mathcal{X},$$

*with $\mathcal{X}^*$ is the **topological dual** of $\mathcal{X}$.*

Proof in [Schi 07].
A general chain rule for subdifferentials is next given:

Theorem A.18 (Chain Rule (subdifferentials))**.** *Let g and $\mathbf{h}$ be so that:*

$$\mathbf{h} : \mathcal{X} \to \mathbb{R}^n, \qquad g : \mathbb{R}^n \to \mathbb{R}, \qquad (g \circ \mathbf{h})(\mathbf{x}) = g(\mathbf{h}(\mathbf{x})), \mathbf{x} \in \mathcal{X},$$

$\mathbf{h} = (h_1, h_2, \ldots, h_n)^\top$, with $h_i : \mathcal{X} \to \mathbb{R},\ i = [\![n]\!]$, and define for any $\mathbf{a} \in \mathbb{R}^n$, $\mathbf{h}_\mathbf{a}(\mathbf{x}) := \langle \mathbf{a}, \mathbf{h}(\mathbf{x}) \rangle$, $x \in \mathcal{X}$. Let $\bar{co}^*(A)$ denotes the closed convex hull of the set $A \in \mathcal{X}$ according to $\mathcal{X}$ and its topological dual (see [Schi 07] for details). Assume that $\mathbf{h}$ is locally L-continuous around $\mathbf{x}_0 \in \mathcal{X}$ and g is locally L-continuous around $\mathbf{h}(\mathbf{x}_0)$. Let $\partial_\circ(g)$ be the Clarke generalized gradient (see [Schi 07] for details). Then:

(a) The composite function $g \circ \mathbf{h}$ is locally L-continuous around $\mathbf{x}_0$, and there holds

$$\partial_\circ(g \circ \mathbf{h})(\mathbf{x}_0) \subseteq \bar{co}^*\big(\{\partial_\circ \mathbf{h}_\mathbf{a}(\mathbf{x}_0) | \mathbf{a} \in \partial_\circ g(\mathbf{h}(\mathbf{x}_0))\}\big).$$

(b) If, in addition, g is regular at $\mathbf{h}(\mathbf{x}_0)$, any h_i is regular at $\mathbf{x}_0$, $i = [\![n]\!]$, and any $\mathbf{a} \in \partial_\circ(\mathbf{h}(\mathbf{x}_0))$ has nonnegative components, then the previous equation holds as equality and $g \circ \mathbf{h}$ is regular at $\mathbf{x}_0$.

The proof of the previous theorem can be found in [Schi 07].

Theorem A.19 (Local Necessary Conditions with subdifferentials (J-KKT) [Schi 07])**.** *Consider the following optimization problem:*

$$\min_{\mathbf{z} \in \mathcal{X}} \quad J(\mathbf{z}) \tag{A.2a}$$

$$\text{subject to} \quad \mathbf{h}(\mathbf{z}) \geq \mathbf{0} \tag{A.2b}$$

similarly defined as Problem 1.19 in page 33. Suppose that $h = (h_1, h_2, \ldots, h_n)$ and $J, h_1, \ldots, h_m : \mathcal{X} \to \bar{\mathbb{R}}$ are proper convex functionals and let A be a nonempty convex subset of D : Dom(J) $\cap$ Dom(h_1)$\cap \cdots \cap$Dom(h_m). The (Fritz) John conditions (J) and the Karush-Kuhn-Tucker conditions (KKT) are the following:

$$\begin{aligned}
\textit{(J):} \quad & \exists(\lambda^*, \alpha_2^*, \ldots, \alpha_m^*) \in \mathbb{R}_+^{m+1} \setminus \{0\} : \\
& 0 \in \lambda^* \partial f(\mathbf{z}^*) + \alpha_2^* \partial g_1(\mathbf{z}^*) + \cdots + \alpha_m^* \partial g_m(\mathbf{z}^*) + \partial \delta_\mathcal{X}(\mathbf{z}^*) \\
& \alpha_i h_i(\mathbf{z}^*) = 0,\ i = [\![m]\!]
\end{aligned}$$

$$\begin{aligned}
\textit{(KKT):} \quad & \exists(\alpha_2^*, \ldots, \alpha_m^*) \in \mathbb{R}_+^m : \\
& 0 \in \partial f(\mathbf{z}^*) + \alpha_2^* \partial g_1(\mathbf{z}^*) + \cdots + \alpha_m^* \partial g_m(\mathbf{z}^*) + \partial \delta_\mathcal{X}(\mathbf{z}^*) \\
& \alpha_i h_i(\mathbf{z}^*) = 0,\ i = [\![m]\!].
\end{aligned}$$

If the functionals $J, h_1, \ldots, h_m$ are continuous at some point of $A \cap Int(D)$ and the ***slater condition****:*

$$\exists \mathbf{z}_0 \in A : h_i(\mathbf{z}_0) < 0 \quad \textit{for } i = [\![m]\!],$$

is hold, then

$$(KKT) \iff \mathbf{z}^* \textit{is a global solution of Problem in Equations } (A.2) \iff (J).$$

The proof can be found in [Schi 07].

A.1.3 Norms

Definition A.20 (Norm)**.** *Given a vector space $\mathcal{X}$ over a subfield F such as the complex or real numbers, a* ***norm*** *on $\mathcal{X}$ is a function $\|\cdot\| : \mathcal{X} \to \mathbb{R}$ with the following properties: for all $a \in F$ and all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$,*

- $\|a\mathbf{x}\| = |a|\|\mathbf{x}\|$, *(positive homogeneity or positive scalability).*
- $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ *(triangle inequality or subadditivity).*
- $\|\mathbf{x}\| = 0$ *if and only if* $\mathbf{x}$ *is the zero vector (positive definiteness).*

Naturally, a **normed vector space** is a vector space that has a norm defined over its elements.

Definition A.21 (Norm of a Function)**.** *Let $\mathcal{F}$, a bounded functional from $\mathcal{X}$ onto $\mathbb{R}$ that is there exists a positive constant α such that $|\mathcal{F}(\mathbf{x})| \leq \alpha\|\mathbf{x}\|, \forall \mathbf{x} \in \mathcal{X}$. Therefore, the norm of $\mathcal{F}$ is the smallest positive constant M that holds $|\mathcal{F}(\mathbf{x})| \leq M\|\mathbf{x}\|$ for all $\mathbf{x} \in \mathcal{X}$ is called the* ***norm*** *of $\mathcal{F}$ and is denoted $\|\mathcal{F}\|$. Thus, $\|\mathcal{F}\| = \inf\{M : |\mathcal{F}(\mathbf{x}) \leq M\|\mathbf{x}\|$, for all $\mathbf{x} \in \mathcal{X}\}$.*

Is can be shown that this definition satisfies the usual requirements of a norm.
The space becomes a normed space by assigning the norm according to the last definition. The norm of a function $\mathcal{F}$ can be expressed in several ways. We have

$$\begin{aligned}
\|\mathcal{F}\| &= \inf\{M : |\mathcal{F}(\mathbf{x})| \leq M\|\mathbf{x}\|, \text{ for all } \mathbf{x} \in \mathcal{X}\} && \text{(A.3)}\\
\|\mathcal{F}\| &= \sup_{\mathbf{x} \neq 0} \frac{|\mathcal{F}(\mathbf{x})|}{\|\mathbf{x}\|} && \text{(A.4)}\\
\|\mathcal{F}\| &= \sup_{\|\mathbf{x}\| \leq 1} |\mathcal{F}(\mathbf{x})| && \text{(A.5)}\\
\|\mathcal{F}\| &= \sup_{\|\mathbf{x}\| = 1} |\mathcal{F}(\mathbf{x})|. && \text{(A.6)}
\end{aligned}$$

The norm defined in this way satisfies the usual requirements of a norm: $\|\mathcal{F}\| > 0$; $\|\mathcal{F}\| = 0$ if and only if $\mathcal{F} = 0$; $\|\alpha\mathcal{F}\| = |\alpha|\|\mathcal{F}\|$; $\|\mathcal{F}_1 + \mathcal{F}_2\| \leq \|\mathcal{F}_1\| + \|\mathcal{F}_2\|$.

Definition A.22 (Frobenius Norm)**.** *The Frobenius norm [Golu 96], sometimes also called the Euclidean norm (which may cause confusion with the vector $\mathcal{L}^2$-norm which also sometimes known as the Euclidean norm), is matrix norm of an $m \times n$ matrix A defined as the square root of the sum of the absolute squares of its elements:*

$$||A||_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2}$$

The Frobenius norm can also be considered as a vector norm. It is also equal to the square root of the matrix trace of AA^H, where A^H is the conjugate transpose, i.e.,

$$||A||_F = \sqrt{Trace(AA^H)}.$$

Definition A.23 (Consistent method)**.** *We say that the method of ERM is **consistent** for a set of functions f_n and for the probability distribution $\mathbb{P}$ if the following hold [Vapn 99]:*

$$\begin{array}{c} \lim_{n\to\infty} \mathbb{P}\left(\left(R(f_n) - \inf_{f\in\mathcal{H}}\left\{R(f)\right\}\right) > \epsilon\right) = 0 \quad \forall\epsilon > 0 \\ \lim_{n\to\infty} \mathbb{P}\left(\left(R_{emp}(f_n) - \inf_{f\in\mathcal{H}}\left\{R(f)\right\}\right) > \epsilon\right) = 0 \quad \forall\epsilon > 0 \end{array}$$

A.1.4 Laplace operator

These notions are used to manipulate objects related to manifolds.

Definition A.24 (Laplacian)**.** *The Laplacian is the sum of all the unmixed second partial derivatives:*

$$\Delta = \nabla^2 = \nabla \cdot \nabla = \sum_{i=1}^{n} \frac{\partial^2}{\partial \mathbf{x}_i^2} \tag{A.7}$$

Definition A.25 (Laplace equation)**.** *The n-th dimensional Laplace equation is:*

$$\Delta = \sum_{i=1}^{n} \frac{\partial^2}{\partial \mathbf{x}_i^2} = 0 \tag{A.8}$$

Definition A.26 (Metric Tensor)**.** *The metric tensor g^{ij} is a function which tells how to compute the distance between any two points $\mathbf{x}_i, \mathbf{x}_j$ in a given space. Its components can be viewed as multiplication factors which must be placed in front of the differential displacements $\partial\mathbf{x}_i$ in a generalized Pythagorean theorem*

$$\partial s^2 = g^{11}\partial\mathbf{x}_1^2 + g^{12}\mathbf{x}_1\partial\mathbf{x}_2 + g^{22}\partial\mathbf{x}_2^2 + ... \tag{A.9}$$

In Euclidean space, $g^{ij} = \delta_{ij}$ where δ is the Kronecker delta (which is 0 for $i \neq j$ and 1 for $i = j$), reproducing the usual form of the Pythagorean theorem:

$$\partial s^2 = \partial\mathbf{x}_1^2 + \partial\mathbf{x}_2^2 + ... \tag{A.10}$$

Definition A.27 (Laplace-Beltrami operator)**.** *The Laplace-Beltrami operator is defined in any local coordinates by:*

$$\Delta u(\mathbf{x}) = g^{-\frac{1}{2}}(\mathbf{x}) \sum_{i,j} \partial_{\mathbf{x}_i}\left(g(\mathbf{x})g^{ij}(\mathbf{x})\partial_{\mathbf{x}_j}\left(g^{-\frac{1}{2}}(\mathbf{x})u(\mathbf{x})\right)\right) \tag{A.11}$$

where g^{ij} is the metric tensor and $g \equiv |det\{g^{ij}\}|^{-\frac{1}{2}}$ is the canonical Riemannian density.

A.1.5 Mixture Models

Mixture models [Jain 88, McLa 00] represent a family of probabilistic approach used to represent the data in an unsupervised way.

For mixture models, observations $S_X = \{\mathbf{x}_i\}, i \in [\![n]\!], \mathbf{x}_i \in \mathbb{R}^{\mathcal{D}}$ are supposed to be realizations of a random variable denoted by $\mathcal{X}$ with probability density function $f(\mathbf{x})$ on $\mathbb{R}^{\mathcal{D}}$. In the framework of mixture models, this density function is defined as:

$$f(\mathbf{x}) = \sum_{i=1}^{k} \pi_i f_i(\mathbf{x}), \tag{A.12}$$

where the $f_i(\mathbf{x})$ are densities and the π_i are nonnegative quantities that sum up to one, that is, $0 \leq \pi_i \leq 1, i \in [\![k]\!]$ and $\sum_{i=1}^{k} \pi_i = 1$.

The quantities $\pi_i, \ldots, \pi_k$ are called the mixing proportions or weights and the corresponding densities $f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})$ are called the ***component densities*** of the mixture. Density (A.12) is referred as a k-component finite mixture distribution. Depending on the model, k be considered to be fixed or that it increases as the sample size n does, see [McLa 00] for references on these models.

One of the most known methods to estimate the mixing proportions along with the parameters of the components is the ***Expectation Maximization (EM)*** method [Demp 77]. In the identification of the mixture models based on EM, the hidden information are the membership of each sample to the components f_i. We would have known this information, the optimization problem will amount to a single maximum likelihood estimation step.

A.1.6 Maximum Likelihood Estimation (MLE)

Let X denote observable variables, and let Z denote latent variables. Often, X and Z decompose into sets of independent, identically-distributed (i.i.d.) pairs, in particular X can often be written as $X = (X_1, X_2, ..., X_n)$, where X_i are (i.i.d.) variables and the observed data, $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$, are the observed values of X. The aim is to estimate the probability model $\mathbb{P}(\mathbf{x}, \mathbf{z}|\theta)$ where θ represents the parameter vector to be determined.

If Z could be observed, then the MLE problem [Jord 02] would amount to maximizing the ***complete log likelihood***:

$$\log\big(\mathbb{P}(\mathbf{x}, \mathbf{z}|\theta)\big),$$

If the probability $\mathbb{P}(\mathbf{x}, \mathbf{z}|\theta)$ factors in some way, such that separate components of θ in separate factors, then the operation of the logarithm has the effect of separating the likelihood into terms that can be maximized independently, this is what is generally called ***decoupling*** the estimation problem.

Given that $\mathbf{z}$ is not in fact observed, the probability of the data $\mathbf{x}$ is a marginal probability, and the ***incomplete log likelihood*** takes the following form:

$$\log \mathbb{P}(\mathbf{x}|\theta) = \log \sum_{\mathbf{z}} \mathbb{P}(\mathbf{x}, \mathbf{z}|\theta),$$

where here the summation is used to stand for marginalization (the derivation goes through without change if it is integrated over a continuous $\mathbf{z}$). The logarithm on the right-hand side is separated from $\mathbb{P}(\mathbf{x}, \mathbf{z}|\theta)$ by the summation sign, and the problem does not decouple. At this point, it is not clear how to exploit the conditional independence structure that may be present in the probability model.

Given that Z is not observed, the complete log likelihood is a random quantity, and cannot be maximized directly. But suppose we average over $\mathbf{z}$ to remove the randomness, using an *averaging distribution* $q(\mathbf{z}|\mathbf{x})$. That is, let us define the ***expected complete log likelihood***:

$$l(\theta, \mathbf{x}, \mathbf{z})_q = \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}, \theta) \log \mathbb{P}(\mathbf{x}, \mathbf{z}|\theta),$$

a quantity that is a deterministic function of θ. Note that the expected complete log likelihood is linear in the complete log likelihood and thus should inherit its favorable computational properties. Moreover, if q is chosen well, then perhaps the expected complete log likelihood will not be too far from the log likelihood and can serve as an effective surrogate for the log likelihood. While we cannot hope that maximizing this surrogate will yield a value of θ that maximizes the likelihood, perhaps it will represent an improvement from an initial value of θ. If so, then we can iterate the process and hill-climb. This is the basic idea behind the EM algorithm. It will be shown that an averaging distribution $q(\mathbf{z}|\mathbf{x})$ can be used to provide a lower bound on the log likelihood. Consider the following line of argument:

$$\begin{aligned} l(\theta, \mathbf{x}) &= \log \mathbb{P}(\mathbf{x}|\theta) \\ &= \log \sum_{\mathbf{z}} \mathbb{P}(\mathbf{x}, \mathbf{z}, |\theta) \\ &= \log \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \frac{\mathbb{P}(\mathbf{x}, \mathbf{z}, |\theta)}{q(\mathbf{z}|\mathbf{x})} && \text{(A.13)} \\ &\geq \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log \frac{\mathbb{P}(\mathbf{x}, \mathbf{z}, |\theta)}{q(\mathbf{z}|\mathbf{x})} && \text{(A.14)} \\ &\equiv L(q, \theta, \mathbf{x}) \end{aligned}$$

where the last line defines $L(q, \theta, \mathbf{x})$, a function that will be referred to as an *auxiliary function.* In Equation (A.14), Jensen's inequality was used, a simple consequence of the concavity of the logarithm function. What has been shown is that –for an arbitrary distribution $q(\mathbf{z}, \mathbf{x})$– the auxiliary function $L(q, \theta, \mathbf{x})$ is a lower bound for the log likelihood.

The EM algorithm is a coordinate ascent algorithm on the function $L(q, \theta, \mathbf{x})$. At the $(t+1)$-st iteration, $L(q, \theta^t, \mathbf{x})$ is first maximized with respect to q to yield q^{t+1}, and then $L(q^{t+1}, \theta, \mathbf{x})$ is maximized with respect to θ, which yields the updated value θ^{t+1}. Giving these steps their traditional names, it is gotten:

$$\begin{aligned} &\textbf{E step:} && q^{t+1} = \operatorname{argmax}_q L(q, \theta^t, \mathbf{x}) \\ &\textbf{M step:} && \theta^{t+1} = \operatorname{argmax}_\theta L(q^{t+1}, \theta, \mathbf{x}) \end{aligned}$$

In order to understand the assigned names for each step, it has to be first noticed that the M step can be equivalently viewed as the maximization of the expected complete log likelihood. To see this, note that the lower bound $L(q, \theta, \mathbf{x})$ breaks into two terms:

$$\begin{aligned} L(q, \theta, \mathbf{x}) &= \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log \frac{\mathbb{P}(\mathbf{x}, \mathbf{z}|\theta)}{q(\mathbf{z}|\mathbf{x})} \\ &= \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log \mathbb{P}(\mathbf{x}, \mathbf{z}|\theta) - \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log q(\mathbf{z}|\mathbf{x}) \\ &= l(\theta, \mathbf{x}, \mathbf{z})_q - \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log q(\mathbf{z}|\mathbf{x}) \end{aligned}$$

and that the second term is independent of θ. Thus, maximizing $L(q, \theta, \mathbf{x})$ with respect to θ is equivalent to maximizing $l(\theta, \mathbf{x}, \mathbf{z})_q$ with respect to θ.

Let us now consider the E step. This maximization problem can be solved once and for all; indeed, we can verify that the choice $q^{t+1}(\mathbf{z}|\mathbf{x}) = \mathbb{P}(\mathbf{z}|\mathbf{x}, \theta^t)$ yields the maximum. To see this, evaluate $L(q, \theta, \mathbf{x})$ for this choice of q:

$$\begin{aligned} L(\mathbb{P}(\mathbf{z}|\mathbf{x}, \theta^t), \theta^t, \mathbf{x}) &= \sum_{\mathbf{z}} \mathbb{P}(\mathbf{z}|\mathbf{x}, \theta^t) \log \frac{\mathbb{P}(\mathbf{x}, \mathbf{z}|\theta^t)}{\mathbb{P}(\mathbf{z}|\mathbf{x}, \theta^t)} \\ &= \sum_{\mathbf{z}} \mathbb{P}(\mathbf{z}|\mathbf{x}, \theta^t) \log \mathbb{P}(\mathbf{x}|\theta^t) \\ &= \log \mathbb{P}(\mathbf{x}|\theta^t) \\ &= l(\theta^t, \mathbf{x}). && \text{(A.15)} \end{aligned}$$

Given that $l(\theta^t, \mathbf{x})$ is an upper bound for $L(\mathbb{P}(\mathbf{z}|\mathbf{x}, \theta^t), \theta^t, \mathbf{x})$, this shows hat $L(\mathbb{P}(\mathbf{z}|\mathbf{x}, \theta^t), \theta^t, \mathbf{x})$ is maximized by setting $q(\mathbf{z}, \mathbf{x})$ equal to $\mathbb{P}(\mathbf{z}|\mathbf{x}, \theta^t)$.

The conditional distribution $\mathbb{P}(\mathbf{z}|\mathbf{x}, \theta^t)$ is an intuitively appealing choice of averaging distribution. Given the model $\mathbb{P}(\mathbf{z}|\mathbf{x}, \theta^t)$ is our *best guess* as to the values of the latent variables, conditioned on the data $\mathbf{x}$. What the EM algorithm does is to use this *best guess* distribution to calculate an expectation of the complete log likelihood. The M step then maximizes this expected complete log likelihood with respect to the parameters to yield new values θ^{t+1}. We then presumably have an improved model, and we can now make a *better guess* $\mathbb{P}(\mathbf{z}|\mathbf{x}, \theta^{t+1})$, which is used as the averaging distribution in a subsequent EM iteration.

What is the effect of an EM iteration on the log likelihood $l(\theta, \mathbf{x})$? In the M step, the parameters are chosen so as to increase a lower bound on the likelihood. Increasing a lower bound on a function does not necessarily increase the function itself, if there is a gap between the function and the bound. In the E step, however, we have closed the gap by an appropriate choice of the q distribution. That is, we have:

$$l(\theta^t, \mathbf{x}) = L(q^{t+1}, \theta^t, \mathbf{x}),$$

by Equation (A.15), and thus an M-step increase in $L(q^{t+1}, \theta, \mathbf{x})$ will also increase $l(\theta^t, \mathbf{x})$.

In summary, it was shown that the EM algorithm is a hill-climbing algorithm in the log likelihood $l(\theta, \mathbf{x})$. The algorithm achieves his hill-climbing behavior indirectly, by coordinate ascent in the auxiliary function $L(q, \theta, \mathbf{x})$. The advantage of working with the latter function is that it involves maximization of the expected complete lo likelihood rather than the log likelihood itself, and it is ofter a substantial simplification.

A.2. Maximal Margin Classifier

A.2.1 Hard Margin linear Case

Having a set $\{(\mathbf{x}_i, y_i)\}_{i \in [\![n]\!]}$, $\mathbf{x}_i \in \mathcal{X} = \mathbb{R}^{\mathcal{D}}, y_i \in \{1, -1\}, i \in [\![n]\!]$, we say that the two classes are *linearly separable* if there is a linear function $f : \mathbb{R}^{\mathcal{D}} \to \mathbb{R}$, $b \in \mathbb{R}$, and the induced hyperplane $\mathcal{P} : \{\mathbf{x} \in \mathcal{X}, b \in \mathbb{R} | f(\mathbf{x}) + b = 0, \mathbf{x} \in \mathcal{X}\}$ can divide the space $\mathcal{X}$ in two subspaces holding, for example, $f(\mathbf{x}_i) - f(\mathbf{x}_j) > 0$ for all pairs $\{\mathbf{x}_i\}_{y_i=1}$ and $\{\mathbf{x}_j\}_{y_j=-1}$.

The hyperplane has the form $f(\mathbf{x})$ will give the length of $\mathbf{x}$ along direction $\mathbf{w}$ $f(\mathbf{x}) + b = \langle \mathbf{w}, \mathbf{x} \rangle + b$, so that $f(\mathbf{x})$ will give the length of $\mathbf{x}$ along direction $\mathbf{w}$.

Hyperplanes in Figure A.1 are all valid functions to divide the samples in Figure 1.7 and actually, there is an infinite number of hyperplanes that keep all samples of the same class on one subspace and the rest in the other one.

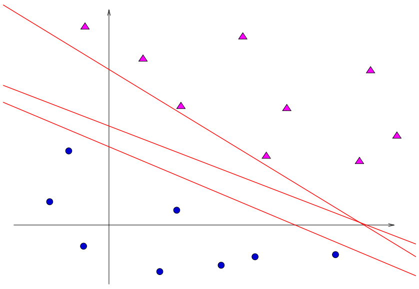


Figure A.1: Possible dividing hyperplanes for a two-class problem

Intuitively, the hyperplane that would like to be kept is the one that has the largest distance to both classes.

We will say that the two classes are perfectly separated by a hyperplane $\mathcal{P}$ if the following equations are fulfilled [Vapn 99]:

$$\begin{cases} f(\mathbf{x}_i) + b \geq 1 & \text{if } y_i = 1 \\ f(\mathbf{x}_i) + b \leq -1 & \text{if } y_i = -1 \end{cases} \quad \text{or equivalently} \quad \{y_i(f(\mathbf{x}_i) + b) \geq 1, \text{ for } i \in [\![n]\!] \tag{A.16}$$

To achieve a separation of the two classes with a maximum distance between them, we are interested in finding a particular hyperplane called *canonical hyperplane* for which, it holds

$$\min_{i \in [\![n]\!]} \|f(\mathbf{x}_i) + b\| = 1.$$

Figure A.2 shows a hyperplane with maximal margin for the classification problem in Figure A.1.

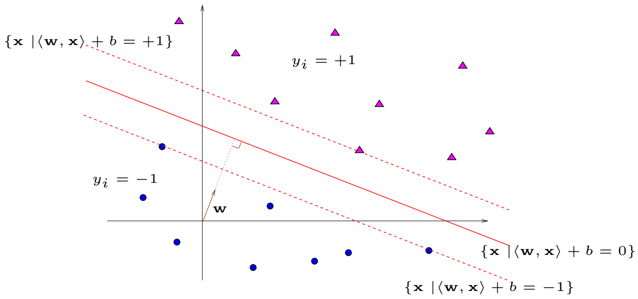

$\{\mathbf{x} \mid \langle \mathbf{w}, \mathbf{x} \rangle + b = +1\}$
$y_i = +1$
$y_i = -1$
$\mathbf{w}$
$\{\mathbf{x} \mid \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}$
$\{\mathbf{x} \mid \langle \mathbf{w}, \mathbf{x} \rangle + b = -1\}$

Figure A.2: Hyperplane with maximal margin for a two-class problem

In the example, the vector $\mathbf{w}$ is a vector orthogonal to the hyperplane $\mathcal{P}$ and geometrically, $\langle \mathbf{w}, \mathbf{x} \rangle$ can be interpreted as the length of $\mathbf{x}$ along direction $\mathbf{w}$. This distance can be standardized by scaling the inner product by the norm of $\mathbf{w}$, $\|\mathbf{w}\|$.

It has to be noticed that if a canonical hyperplane is used, the closest points of the different classes satisfy: $\mathbf{x}_i$ with $y_i = 1$ and a $\mathbf{x}_j$ with $y_j = -1$ holding $f(\mathbf{x}_i) + b = +1$ and $f(\mathbf{x}_j) + b = -1$. This leads to $\langle \mathbf{w}, \mathbf{x}_i \rangle - \langle \mathbf{w}, \mathbf{x}_j \rangle = 2$, which after normalization by the norm of $\mathbf{w}$, we get:

$$\frac{1}{\|\mathbf{w}\|}\big(\langle \mathbf{w}, \mathbf{x}_i \rangle - \langle \mathbf{w}, \mathbf{x}_j \rangle\big) = \frac{2}{\|\mathbf{w}\|} \tag{A.17}$$

With the previous equation, we can conclude that the distance of the closest vector to the hyperplane is $\frac{1}{\|\mathbf{w}\|}$, then finding the hyperplane with the maximum distance is equivalent to maximize the norm of the decision function f that corresponds to the hyperplane that will divide the two classes if Constraints A.16 are included in the setting.

Therefore, the regularization problem can be stated as follows:

Problem A.28 (SVM-Primal Optimization Problem)**.** *Consider a binary classification problem with class labels encoded as* $+1$ *and* -1*. The optimal margin hyperplane primal problem is defined as follows.*

$$\underset{\mathbf{w} \in \mathcal{H}, b \in \mathbb{R}}{\text{minimize}} \quad \tfrac{1}{2}\|\mathbf{w}\|^2, \tag{A.18}$$

$$\text{subject to} \quad y_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) \geq 1, \;\; i = 1, .., n, \tag{A.19}$$

The final decision function would look like

$$f'(x) = \operatorname{sign}(\langle \mathbf{w}, \mathbf{x} \rangle + b).$$

The samples lying on the margin (that is the patterns locate at a distance $\frac{1}{\|\mathbf{w}\|}$ from the hyperplane) are called **Support Vectors** (SVs) as the solution $\mathbf{w}$ will depend only on those points.

Using the mathematical tools developed in Section 1.4 of Chapter 1, on can easily derive the corresponding dual of Problem A.28 in the following form

Problem A.29 (SVM-Dual Optimization Problem)**.** *The dual problem corresponding to the optimal margin hyperplane Problem A.28 is given by*

$$\begin{aligned} \underset{\boldsymbol{\alpha} \in \mathbb{R}^n}{\text{maximize}} \quad & \textstyle\sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle, \\ \text{subject to} \quad & \alpha_i \geq 0, \;\; i = 1, ..., n, \\ & \textstyle\sum_{i=1}^n \alpha_i y_i = 0. \end{aligned}$$

knowing that the expression of $\mathbf{w}$ is given by $\mathbf{w} = \sum_{i=1}^{n} y_i \alpha_i \mathbf{x}_i$.

A.2.2 Soft Margin linear Case

Often, the problem can be an unfeasible problem because there does not exist any hyperplane that can separate perfectly both classes. For such problems, the C-SV classifier was introduced allowing some mistakes through slack variables with a penalization in the objective function, leading to the following problem.

Problem A.30 (SV Classifier (SVC) Primal Problem)**.** *For a two-class problem, the primal optimization problem with slack variables $\xi, i = 1, \cdots, n$ is defined as:*

$$\begin{array}{rc} \underset{\mathbf{w}\in\mathcal{H}, b\in\mathbb{R}, \boldsymbol{\xi}\in\mathbb{R}^m}{\text{minimize}} & \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} C\xi_i, \\ \text{subject to} & y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) \geq 1 - \xi_i,\ i = 1, .., n, \\ & \xi_i \geq 0,\ i = 1, .., n. \end{array}$$

Here again, the dual is straightforwardly given in the definition below.

Problem A.31 (C-SV Classifier (SVC) Dual Problem)**.** *In a two-class problem, the optimal margin hyperplane dual problem with slack variables is defined as follows*

$$\begin{array}{rc} \underset{\boldsymbol{\alpha}\in\mathbb{R}^n}{\text{maximize}} & \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle, \\ \text{subject to} & 0 \leq \alpha_i \leq C,\ i = 1, ..., n, \\ & \sum_{i=1}^{n} \alpha_i y_i = 0. \end{array} \tag{A.20}$$

Here, the support vectors include the samples in the margin as previously but also the points with slack variables strictly positive, namely the points lying inside the margin or those badly classified.

A.3. Quadratic Problems

Another commonly used method to solve nonlinear convex optimization is the **interior points method**. We are concerned in this method as it is the principle of some piecewise linear solutions for parametric quadratic programming [Hast 04, Ross 07b, Mark 59].

The original method was proposed for linear programming [Vand 01], there exist many extensions for nonlinear programming, some of them based in barrier methods using the logarithmic or quadratic function but in our case, we will be interested in the primal-dual implementation for the quadratic particular case.

The method will be derived for the following problem:

$$\begin{array}{rc} \underset{\mathbf{x}}{\text{minimize}} & \frac{1}{2}\mathbf{x}^\top Q\mathbf{x} - \mathbb{1}^\top \mathbf{x} \\ \text{subject to} & A\mathbf{x} \geq b \\ & \mathbf{y}^\top \mathbf{x} = d \end{array} \tag{A.21}$$

where Q is symmetric and positive semidefinite, and where A is a matrix and $\mathbf{y}$ a vector that define linear inequality and equality constraints respectively.

The optimal solution $(\mathbf{x}^*, \alpha^*, \beta^*)$ should satisfy the KKT conditions stated in Equations 1.21. This conditions are recalled under this framework:

$$\begin{array}{rcl} Q\mathbf{x} - \mathbb{1} - A^\top \boldsymbol{\alpha} + \beta \mathbf{y} & = & 0 \\ A\mathbf{x} - b & \geq & 0 \\ \mathbf{y}^\top \mathbf{x} - d & = & 0 \\ \alpha_i (A\mathbf{x} - b)_i & = & 0 \\ \boldsymbol{\alpha} & \geq & 0 \end{array}$$

where $(A\mathbf{x}-b)_i$ denotes the i-th element of vector $(A\mathbf{x}-b)$. A slack vector will be introduced in the inequality $\mathbf{s}=(A\mathbf{x}-b)$ equation so that it can be written as:

$$\begin{aligned} Q\mathbf{x}-\mathbb{1}-A^\top\boldsymbol{\alpha}+\beta\mathbf{y} &= 0 && \text{(A.23a)}\\ A\mathbf{x}-b-\mathbf{s} &= 0 && \text{(A.23b)}\\ \mathbf{y}^\top\mathbf{x}-d &= 0 && \text{(A.23c)}\\ \alpha_i s_i &= 0 && \text{(A.23d)}\\ \boldsymbol{\alpha},\mathbf{s} &\geq 0 && \text{(A.23e)} \end{aligned}$$

These KKT conditions are not only necessary but also sufficient as the objective function and feasible region are convex. Hence, the quadratic Problem A.21 can be solved if a solution can be found for Equations A.23. The previous system will be rewritten as a constrained system of nonlinear equations and derive a primal-dual interior points algorithms by applying modifications of Newton's method to this system by defining:

$$F(\mathbf{x},s,\boldsymbol{\alpha},\boldsymbol{\beta})=\begin{bmatrix} Q\mathbf{x}-\mathbb{1}-A^\top\boldsymbol{\alpha}+\beta\mathbf{y}\\ A\mathbf{x}-b-\mathbf{s}\\ \mathbf{y}^\top\mathbf{x}-d\\ S\Lambda\mathbb{1}\end{bmatrix}, \qquad \boldsymbol{\alpha},\mathbf{s}\geq 0 \tag{A.24}$$

with $S=\text{diag}(s_1,s_2,\ldots,s_m)$, $\Lambda=diag(\alpha_1,\alpha_2,\ldots,\alpha_m)$ and $\mathbb{1}$ a vector of ones.
Following [Noce 99], at solution $(\mathbf{x},s,\boldsymbol{\alpha},\boldsymbol{\beta})$ that satisfies $\boldsymbol{\alpha},\mathbf{s}\geq 0$, a duality measure μ can be defined by

$$\mu=\frac{1}{m}\sum_{i=1}^{m}s_i\alpha_i=\frac{\mathbf{s}^\top\boldsymbol{\alpha}}{m}$$

Then, the **central path C** is the set of points $(\mathbf{x}^t,s^t,\boldsymbol{\alpha}^t,\boldsymbol{\beta}^t)$, $t>0$ indicates in this case the iteration number, such that

$$F(\mathbf{x},s,\boldsymbol{\alpha},\boldsymbol{\beta})=\begin{bmatrix}0\\0\\0\\t\mathbb{1}\end{bmatrix},\boldsymbol{\alpha},\mathbf{s}\geq 0$$

In order to achieve this, we are interested in calculating the generic step $(\Delta\mathbf{x},\Delta s,\Delta\boldsymbol{\alpha},\Delta\boldsymbol{\beta})$ to make a Newton-like step from the current point $(\mathbf{x},s,\boldsymbol{\alpha},\boldsymbol{\beta})$ on the central path, where $\sigma\in[0,1]$ is a parameter chosen by the algorithm. By doing linearization, implying, for example that terms of second order like $\Delta S\Delta\Lambda$ will be neglected, this step satisfies the system:

$$\begin{bmatrix} Q & -A^\top & \mathbf{y} & 0\\ A & 0 & 0 & -I\\ \mathbf{y}^\top & 0 & 0 & 0\\ 0 & S & 0 & \Lambda\end{bmatrix}\begin{bmatrix}\Delta\mathbf{x}\\ \Delta\alpha\\ \Delta\beta\\ \Delta\mathbf{s}\end{bmatrix}=\begin{bmatrix}-r_{\mathbf{1}}\\ -r_b\\ -r_d\\ \Delta S+\sigma\mu\mathbb{1}\end{bmatrix} \tag{A.25}$$

where
$-r_{\mathbf{1}}=Q\mathbf{x}-\mathbb{1}-A^\top\boldsymbol{\alpha}+\mathbf{y}\beta$, $-r_b=A\mathbf{x}-b-\mathbf{s}$ and $-r_d=\mathbf{y}^\top\mathbf{x}-d$
and the following iterate is obtained by setting:

$$(\mathbf{x}^t,s^t,\boldsymbol{\alpha}^t,\boldsymbol{\beta}^t)=(\mathbf{x},s,\boldsymbol{\alpha},\boldsymbol{\beta})+\lambda(\Delta\mathbf{x},\Delta s,\Delta\boldsymbol{\alpha},\Delta\boldsymbol{\beta})$$

where λ is chosen to retain the inequality $(\mathbf{s}^t,\boldsymbol{\alpha}^t,\beta^t)>0$ and possibly to satisfy various other conditions.
The system A.25 can be differently restated in a more compact and symmetric form (see [Noce 99] for more details) which can be later used to form a normal equation –that has to be actualized at each iteration– and a modified Cholesky algorithm can be used to solve it.

References

[Abra 74] M. Abramowitz. *Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables.* Dover Publications, Incorporated, 1974.

[Allg 90] E. L. Allgower and K. Georg. *Numerical continuation methods: an introduction.* Springer-Verlag, New York, Inc., New York, USA, 1990.

[Anto 09] A. Antoniadis, I. Gijbels, and M. Nikolova. "Penalized Likelihood Regression for Generalized Linear Models with Nonquadratic Penalties". *Annals of the Institute of Mathematical Statistics*, 2009. To appear.

[Aron 50] N. Aronszajn. "Theory of Reproducing Kernels". *Transactions of the American Mathematical Society*, Vol. 68, No. 3, pp. 337–404, 1950.

[Aupe 07] M. Aupetit. "Visualizing distortions and recovering topology in continuous projection techniques". *Neurocomputing*, Vol. 70, No. 7-9, pp. 1304–1330, 2007.

[Aupe 08] M. Aupetit. "Homogeneous Bipartition Based on Multidimensional Ranking". In: *16th European Symposium on Artificial Neural Networks, Advances in Computational Intelligence and Learning*, pp. 259–264, Bruges, Belgique, April 2008.

[Bach 05] F. R. Bach, R. Thibaux, and M. I. Jordan. "Computing regularization paths for learning multiple kernels". In: L. K. Saul, Y. Weiss, and L. Bottou, Eds., *Advances in Neural Information Processing Systems 17*, pp. 73–80, The MIT Press, Cambridge, MA, 2005.

[Bach 06] F. R. Bach, D. Heckerman, and E. Horvitz. "Considering Cost Asymmetry in Learning Classifiers". *Journal of Machine Learning Research*, Vol. 7, pp. 1713–1741, 2006.

[Bach 08] F. Bach and J.-Y. Audibert. "Supervised learning for computer vision: Theory and algorithms - Part II". Tutorial Talk in 10th European Conference on Computer Vision (ECCV), 2008.

[Bala 02] M. Balasubramanian, E. L. Schwartz, J. B. Tenenbaum, V. de Silva, and J. C. Langford. "The Isomap Algorithm and Topological Stability". *Science*, Vol. 295, No. 7a, 2002.

[Bara 93] R. Baraniuk and D. Jones. "A signal-dependent time-frequency representation : optimal kernel design". *IEEE Transactions on signal processing*, Vol. 41, No. 4, pp. 1589–1602, 1993.

[Bart 06] P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. “Convexity, Classification, and Risk Bounds”. *Journal of the American Statistical Association*, Vol. 101, No. 473, pp. 138–156, 2006.

[Bart 76] R. G. Bartle. *The Elements of Real Analysis.* John Wiley & Sons, Inc., second Ed., 1976.

[Bele 99] A. D. Belegundu and T. R. Chandrupatla. *Optimization Concepts and Applications in Engineering.* Prentice Hall, 1999.

[Belk 01] M. Belkin and P. Niyogi. “Laplacian eigenmaps and spectral techniques for embedding and clustering”. In: T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds., *Advances in Neural Information Processing Systems 14*, pp. 585–591, The MIT Press, 2001.

[Belk 04] M. Belkin, I. Matveeva, and P. Niyogi. “Regularization and semi-supervised learning on large graphs”. In: *Lecture Notes in Computer Science (Proc. of 17th Conference on Learning Theory, COLT)*, pp. 624–638, Springer, 2004.

[Belk 06] M. Belkin, P. Niyogi, and V. Sindhwani. “Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples.”. *Journal of Machine Learning Research*, Vol. 7, pp. 2399–2434, 2006.

[Benn 98] K. P. Bennett and A. Demiriz. “Semi-Supervised Support Vector Machines”. In: M. S. Kearns, S. A. Solla, and D. A. Cohn, Eds., *Advances in Neural Information Processing Systems 11*, pp. 368–374, The MIT Press, Cambridge, MA, 1998.

[Bi 03a] J. Bi. “Multi-objective programming in SVMs”. In: *Machine Learning, Proceedings of the 20th International Conference (ICML 2003), AAAI*, pp. 35–42, Press, 2003.

[Bi 03b] J. Bi and V. N. Vapnik. “Learning with rigorous support vector machines”. In: *Proceedings of the 16th Annual Conference on Learning Theory*, pp. 35–42, AAAI Press, 2003.

[Bie 06] T. D. Bie and N. Cristianini. “Semi-supervised learning using semi-definite programming”. In: B. S. O. Chapelle and A. Zien, Eds., *Semi-supervised Learning*, Chap. 2, The MIT Press, 2006.

[Bonn 06] F. Bonnans. *Optimisation Continue.* Dunod, Paris, 2006.

[Bose 92] B. E. Boser, I. Guyon, and V. Vapnik. “A Training Algorithm for Optimal Margin Classifiers”. In: *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pp. 144–152, ACM Press, 1992.

[Bott 07] L. Bottou and C.-J. Lin. “Support Vector Machine Solvers”. In: L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, Eds., *Large Scale Kernel Machines*, pp. 301–320, MIT Press, Cambridge, MA., 2007.

[Bott 08] L. Bottou and O. Bousquet. “Learning Using Large Datasets”. In: *Mining Massive DataSets for Security*, IOS Press, Amsterdam, 2008. to appear.

[Bous 04] O. Bousquet, O. Chapelle, and M. Hein. “Measure based regularization”. In: S. Thrun, L. Saul, and B. Schölkopf, Eds., *Advances in Neural Information Processing Systems 16*, MIT Press, Cambridge, MA, 2004.

[Boyd 04] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, March 2004.

[Boyd 91] S. P. Boyd and C. H. Barratt. *Linear controller design: limits of performance.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1991.

[Brac 73] J. Bracken and J. McGill. “Mathematical programs with optimization problems in the constraints”. *Operations Research*, Vol. 21, pp. 37–44, 1973.

[Bran 03] M. Brand. “Charting a manifold”. In: S. T. S. Becker and K. Obermayer, Eds., *Advances in Neural Information Processing Systems 15*, pp. 961–968, MIT Press, Cambridge, MA, 2003.

[Bred 99] E. J. Bredensteiner and K. P. Bennett. “Multicategory classification by support vector machines”. *Computational Optimizations and Applications*, Vol. 12, No. 1-3, pp. 53–79, 1999.

[Burg 05] C. J. C. Burges. “Geometric Methods for Feature Extraction and Dimensional Reduction”. In: O. Maimon and L. Rokach, Eds., *Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers*, pp. 59–92, Springer, 2005.

[Cato 07] O. Catoni. “Pac-Bayesian Supervised Classification: The Thermodynamics of Statistical Learning”. *Lecture Notes–Monograph Series*, Vol. 56, 2007.

[Cawl 07] G. C. Cawley and N. L. C. Talbot. “Preventing Over-Fitting during Model Selection via Bayesian Regularisation of the Hyper-Parameters”. *J. Mach. Learn. Res.*, Vol. 8, pp. 841–861, 2007.

[Chan 05] C.-C. Chang and C.-J. Lin. “LIBSVM: a Library for Support Vector Machines”. www.csie.ntu.edu.tw/˜cjlin/papers/libsvm.pdf, 2005.

[Chap 02] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. “Choosing multiple parameters for support vector machines”. *Machine Learning*, Vol. 46, No. 1-3, pp. 131–159, 2002.

[Chap 03] O. Chapelle, J. Weston, and B. Schölkopf. “Cluster kernels for semi-supervised learning”. In: S. T. S. Becker and K. Obermayer, Eds., *Advances in Neural Information Processing Systems 15*, pp. 585–592, The MIT Press, Cambridge, MA, 2003.

[Chap 05] O. Chapelle and A. Zien. “Semi-supervised classification by low density separation”. In: *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.

[Chap 06a] O. Chapelle, B. Schölkopf, and A. Zien, Eds. *Semi-Supervised Learning.* The MIT Press, Cambridge, MA, 2006.

[Chap 06b] O. Chapelle, M. Chi, and A. Zien. “A continuation method for semi-supervised SVMs”. In: *ICML ’06: Proceedings of the 23rd international conference on Machine learning*, pp. 185–192, ACM, New York, NY, USA, 2006.

[Chap 07a] O. Chapelle. “Training a Support Vector Machine in the Primal”. *Neural Computation*, Vol. 19, No. 5, pp. 1155–1178, May 2007.

[Chap 07b] O. Chapelle, V. Sindhwani, and S. S. Keerthi. “Branch and Bound for Semi-Supervised Support Vector Machines”. In: B. Schölkopf, J. Platt, and T. Hoffman, Eds., *Advances in Neural Information Processing Systems 19*, pp. 217–224, The MIT Press, Cambridge, MA, 2007.

[Chap 08] O. Chapelle, V. Sindhwani, and S. Keerthi. “Optimization Techniques for Semi-supervised Support Vector Machines”. *Journal of Machine Learning Research*, Vol. 9, pp. 203–233, 2008.

[Chen 98] S. S. Chen, D. L. Donoho, Michael, and A. Saunders. “Atomic decomposition by basis pursuit”. *SIAM Journal on Scientific Computing*, Vol. 20, pp. 33–61, 1998.

[Choi 07] H. Choi and S. Choi. "Robust kernel Isomap". *Pattern Recognition*, Vol. 40, No. 3, pp. 853–862, 2007.

[Chri 97] F. Christoffersen and F. X. Diebold. "Optimal Prediction Under Asymmetric Loss". *Econometric Theory*, Vol. 13, pp. 808–817, 1997.

[Clar 98] F. H. Clarke, Y. S. Ledyaev, R. J. Stern, and P. R. Wolenski. *Nonsmooth analysis and control theory.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[Coll 06] R. Collobert, F. Sinz, J. Weston, and L. Bottou. "Large scale Transductive SVMs". *Journal of Machine Learning Research*, Vol. 7, pp. 1687–1712, 2006.

[Cort 07] C. Cortes, M. Mohri, and A. Rastogi. "An Alternative Ranking Problem for Search Engines.". In: C. Demetrescu, Ed., *WEA*, pp. 1–22, Springer, 2007.

[Cram 02] K. Crammer and Y. Singer. "Pranking with Ranking". In: T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds., *Advances in Neural Information Processing Systems 14*, The MIT Press, Cambridge, MA, 2002.

[DeCo 00] D. DeCoste and K. Wagstaff. "Alpha Seeding for Support Vector Machines.". In: *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 2000.

[Demp 77] A. Dempster, N. Laird, and D. Rubin. "Maximum likelihood from incomplete data via the EM algorithm". *Journal of the Royal Statistical Society*, Vol. 39, No. 1, pp. 1–38, 1977.

[Do 08] T.-M.-T. Do and T. Artières. "A Fast Method for Training Linear SVM in the Primal". In: *ECML PKDD '08: Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I*, pp. 272–287, Springer-Verlag, Berlin, Heidelberg, 2008.

[Dono 03a] D. L. Donoho and M. Elad. "Optimally sparse representation in general (non-orthogonal) dictionaries via l1 minimization". In: S. Falkow, Ed., *Proceedings of the National Academy of Sciences of the United States of America*, pp. 2197–2202, March 2003.

[Dono 03b] D. L. Donoho and C. Grimes. "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data". In: S. Falkow, Ed., *Proceedings of the National Academy of Sciences of the United States of America*, pp. 5591–5596, May 2003.

[Duan 05] K. bo Duan and S. S. Keerthi. "Which is the best multiclass SVM method? An empirical study". In: *Proceedings of the Sixth International Workshop on Multiple Classifier Systems*, pp. 278–285, 2005.

[Duch 08] J. Duchi, S. S. Shwartz, Y. Singer, and T. Chandra. "Efficient projections onto the L1-ball for learning in high dimensions". In: *ICML '08: Proceedings of the 25th international conference on Machine learning*, pp. 272–279, ACM, New York, NY, USA, 2008.

[Dwas 57] M. Dwass. "Modified randomization tests for nonparametric hypothesis". *Annals of Mathematical Statistics*, Vol. 28, No. 1, pp. 181–187, March 1957.

[Efro 04] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. "Least Angle Regression". *Annals of Statistics*, Vol. 32, No. 2, pp. 407–499, 2004.

[Efro 87] B. Efron. *The Jackknife, the Bootstrap, and Other Resampling Plans (CBMS-NSF Regional Conference Series in Applied Mathematics).* Society for Industrial & Applied Mathematics, January 1987.

[Efro 93] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap.* Chapman & Hall, New York, 1993.

[Elis 03] A. Elisseeff and M. Pontil. "Leave-one-out error and stability of learning algorithms with applications". In: J. Suykens, G. Horvath, S. Basu, C. Micchelli, and J. Vandewalle, Eds., *Advances in Learning Theory: Methods, Models and Applications, NATO Science Series III: Computer and Systems Sciences*, pp. 111–130, IOS Press, Amsterdam, 2003.

[Evge 04] T. Evgeniou, M. Pontil, and A. Elisseeff. "Leave One Out Error, Stability, and Generalization of Voting Combinations of Classifiers". *Machine Learning*, Vol. 55, No. 1, pp. 71–97, April 2004.

[Ewen 06] W. J. Ewens and H. S. Wilf. "Computing the distribution of the maximum in balls-and-boxes problems, with application to clusters of disease cases". *Proceedings of the National Academy of Sciences of the United States of America*, 2006.

[Fan 05] R.-E. Fan, P.-H. Chen, and C.-J. Lin. "Working Set Selection Using the Second Order Information for Training SVM". *J. Mach. Learn. Res.*, Vol. 6, pp. 1899–1918, 2005.

[Figu 07] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright. "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems". *IEEE Journal of Selected Topics in Signal Processing*, Vol. 1, No. 4, pp. 586–597, 2007.

[Fodo 02] I. Fodor. "A Survey of Dimension Reduction Techniques". Tech. Rep. UCRL-ID-148494, Lawrence Livermore National Laboratory, 2002.

[Fral 89] J. B. Fraleigh. *A first course in abstract algebra.* Addison Wesley, fourth Ed., 1989.

[Freu 03] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. "An efficient boosting algorithm for combining preferences". *J. Mach. Learn. Res.*, Vol. 4, pp. 933–969, 2003.

[Frie 07] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani. "Pathwise coordinate optimization". *Annals of Applied Statistics*, Vol. 1, No. 2, pp. 302–332, 2007.

[Fu 00] W. Fu and K. Knight. "Asymptotics for LASSO-type Estimators". *Annals of Statistics*, Vol. 28, No. 5, pp. 1356–1378, 2000.

[Fuji 05] A. Fujino, N. Ueda, and K. Saito. "A hybrid generative/discriminative approach to semi-supervised classifier design". In: *Proceedings of the Tenth National Conference on Artificial Intelligence*, 2005.

[Fung 01] G. Fung and O. L. Mangasarian. "Semi-supervised support vector machines for unlabeled data classification". *Optimization Methods and Software*, Vol. 15, pp. 29–44, 2001.

[Furn 02] J. Fürnkranz. "Pairwise Classification as an Ensemble Technique". In: *ECML '02: Proceedings of the 13th European Conference on Machine Learning*, pp. 97–110, Springer-Verlag, London, UK, 2002.

[Gail 08] P. Gaillard. *Apprentissage de la connexité d'un nuage de points par modèle génératif.* PhD thesis, Université de Technologie de Compiègne, Compiègne, France, November 2008.

[Gang 07] D.-Y. Y. Gang Wang and F. H. Lochovsky. "A kernel path algorithm for support vector machines". In: *Proceedings of ICML'2007*, 2007.

[Gass 07a] G. Gasso, K. Zapién, and S. Canu. "Computing and stopping the solution paths for ν-SVR". In: *Proceedings of ESANN*, 2007.

[Gass 07b] G. Gasso, A. Rakotomamonjy, M. Davy, G. Loosli, and S. Canu. "Regularization paths for nu-SVMs algorithms". Tech. Rep., LITIS, July 2007.

[Gass 07c] G. Gasso, K. Zapién, and S. Canu. "Sparsity Regularization Path for Semi-Supervised SVM". In: *Proceedings of 4th International Conference in Machine Learning and Applications*, pp. 25–30, IEEE Computer Society, Los Alamitos, CA, USA, 2007.

[Gass 08a] G. Gasso. "ECML PKDD Tutorial on Regularization frontier in machine learning". September 2008.

[Gass 08b] G. Gasso, K. Zapién, and S. Canu. "Apprentissage semi supervisé via un SVM parcimonieux : calcul du chemin de régularisation". *Revue I3 - Information Interaction Intelligence*, Vol. 8, No. 2, 2008.

[Gass 08c] G. Gasso, K. Zapién, and S. Canu. "Apprentissage semi supervisé via un SVM parcimonieux : calcul du chemin de régularisation". In: *RFIA 2008*, Amiens, France, 2008.

[Gent 02] M. G. Genton. "Classes of kernels for machine learning: a statistics perspective". *J. Mach. Learn. Res.*, Vol. 2, pp. 299–312, 2002.

[Golu 96] G. H. Golub and C. F. V. Loan. *Matrix computations (3rd ed.)*, p. 55. Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[Gran 05] Y. Grandvalet and Y. Bengio. "Semi-supervised learning by entropy minimization". In: Y. W. L. K. Saul and L. Bottou, Eds., *Advances in neural information processing systems*, The MIT Press, Cambridge, MA, 2005.

[Guer 07] Y. Guermeur. "HDR: SVM Multiclasses, Théorie et Applications". November 2007.

[Gunt 06] L. Gunter and J. Zhu. "Computing the Solution Path for the Regularized Support Vector Regression". In: Y. Weiss, B. Schölkopf, and J. Platt, Eds., *Advances in Neural Information Processing Systems 18*, pp. 483–490, The MIT Press, Cambridge, MA, 2006.

[Guo 05] Y. Guo and D. Schuurmans. "Support Vector Machines on General Confidence Functions". Tech. Rep., University of Alberta, Edmonton, Canada, 2005.

[Guyo 93] I. Guyon, B. Boser, and V. Vapnik. "Automatic Capacity Tuning of Very Large VC-Dimension Classifiers". In: S. J. Hanson, J. D. Cowan, and C. L. Giles, Eds., *Advances in Neural Information Processing Systems*, pp. 147–155, Morgan Kaufmann, San Mateo, CA, 1993.

[Haas 05] B. Haasdonk. "Feature Space Interpretation of SVMs with Indefinite Kernels". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 27, No. 4, pp. 482–492, 2005.

[Hast 01] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction.* Springer Verlag, New York, 2001.

[Hast 04] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. "The Entire Regularization Path for the Support Vector Machine". *Journal of Machine Learning Research*, Vol. 5, pp. 1391–1415, October 2004.

[Hast 89] T. Hastie and W. Stuetzle. "Principal Curves". *Journal of the American Statistical Association*, Vol. 84, No. 406, pp. 502–516, 1989.

[Hast 98] T. Hastie and R. Tibshirani. "Classification by pairwise coupling". In: A. S. M.I. Jordan, M.J. Kearns, Ed., *Advances in Neural Information Processing Systems 10*, The MIT Press, 1998.

[Hein 07] M. Hein, J.-Y. Audibert, and U. von Luxburg. "Graph Laplacians and their Convergence on Random Neighborhood Graphs". *J. Mach. Learn. Res.*, Vol. 8, pp. 1325–1370, 2007.

[Herb 00] R. Herbrich, T. Graepel, and K. Obermayer. "Large Margin Rank Boundaries for Ordinal Regression". In: A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, Eds., *Advances in Large Margin Classifiers*, pp. 115–132, The MIT Press, Cambridge, MA, 2000.

[Hers 94] W. R. Hersh, C. Buckley, T. J. Leone, and D. H. Hickam. "OHSUMED: An Interactive Retrieval Evaluation and New Large Test Collection for Research". In: *SIGIR*, pp. 192–201, 1994.

[Hou 08] C. Hou, Y. Jiao, Y. Wu, and D. Yi. "Relaxed maximum-variance unfolding". *Optical Engineering*, Vol. 47, No. 7, 2008.

[Hsu 02] C. Hsu and C. Lin. "A Comparison of Methods for Multi-Class Support Vector Machines". *IEEE Transactions on Neural Networks*, Vol. 13, No. 2, pp. 415–425, 2002.

[Huo 07] X. Huo, X. S. Ni, and A. K. Smith. "A Survey of Manifold-based Learning Methods. Emerging nonparametric methodology". In: T. W. Liao and E. Triantaphyllou, Eds., *Recent advances in data Mining of Enterprise Data: Algorithms and Applications*, Chap. 15, pp. 691–746, World Scientific, 2007.

[Ivan 76] V. V. Ivanov. *The theory of approximate methods and their application to the numerical solution of singular integral equations.* Noordhoff International Publishing, Schuttersveld 9, P. O. Box 26, Leyden, The Netherlands, 1976.

[Jain 88] A. K. Jain and R. C. Dubes. *Algorithms for clustering data.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[Jarv 00] K. Järvelin and J. Kekäläinen. "IR evaluation methods for retrieving highly relevant documents". In: *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 41–48, ACM, New York, NY, USA, 2000.

[Jarv 02] K. Järvelin and J. Kekäläinen. "Cumulated gain-based evaluation of IR techniques". *ACM Transactions on Information Systems (TOIS)*, Vol. 20, No. 4, pp. 422–446, October 2002.

[Joac 02] T. Joachims. "Optimizing Search Engines Using Clickthrough Data". In: *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 133–142, 2002.

[Joac 03] T. Joachims. "Transductive Learning via Spectral Graph Partitioning". In: *Proceedings of the Twentieth International Conference on Machine Learning, ICML*, 2003.

[Joac 99a] T. Joachims. "Making large-scale support vector machine learning practical". In: *Advances in kernel methods: support vector learning*, Chap. 2, pp. 169–184, MIT Press, Cambridge, MA, USA, 1999.

[Joac 99b] T. Joachims. "Transductive Inference for Text Classification using Support Vector Machines". In: I. Bratko and S. Dzeroski, Eds., *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pp. 200–209, Morgan Kaufmann Publishers, San Francisco, US, Bled, SL, 1999.

[John 67] S. Johnson. "Hierarchical clustering schemes". *Psychometrika*, Vol. 32, No. 3, pp. 241–254, 1967.

[Joll 86] I. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[Jone 95] D. Jones and R. Baraniuk. "An adaptive optimal-kernel time-frequency representation". *IEEE Transactions on Signal Processing*, Vol. 43, No. 10, pp. 2361–2371, 1995.

[Jord 02] M. I. Jordan and C. Bishop. "Introduction to Graphical Models". 2002.

[Keer 01] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. "Improvements to Platt's SMO Algorithm for SVM Classifier Design". *Neural Computation*, Vol. 13, No. 3, pp. 637–649, 2001.

[Keer 07] S. S. Keerthi, V. Sindhwani, and O. Chapelle. "An efficient method for gradient-based Adaption of hyperparameters in SVM models". In: B. Schölkopf, J. Platt, and T. Hoffman, Eds., *Advances in Neural Information Processing Systems 19*, pp. 673–680, MIT Press, Cambridge, MA, 2007.

[Kime 71] G. S. Kimeldorf and G. Wahba. "Some Results on Tchebycheffian Spline Functions". *Journal of Mathematical Analysis and Applications*, Vol. 33, No. 1, pp. 82–95, 1971.

[Kocs 04] A. Kocsor and L. Tóth. "Application of Kernel-Based Feature Space Transformations and Learning Methods to Phoneme Classification". *Applied Intelligence*, Vol. 21, No. 2, pp. 129–142, 2004.

[Kuna 08] G. Kunapuli, K. P. Bennett, J. Hu, and J.-S. Pang. "Classification model selection via bilevel programming". *Optimization Methods Software*, Vol. 23, No. 4, pp. 475–489, 2008.

[Kurt 48] A. K. Kurtz. "A research test of Rorschach test". *Personnel Psychology*, Vol. 1, pp. 41–53, 1948.

[Lawr 05] N. Lawrence and M. Jordan. "Semi-Supervised Learning via Gaussian Processes". In: Y. W. L.K. Saul and L. Bottou, Eds., *Advances in Neural Information Processing Systems 17*, pp. 753–760, The MIT Press, 2005.

[Lawr 08] N. Lawrence. "ICML tutorial on probabilistic dimensionality reduction". July 2008.

[Lee 04] Y. Lee, Y. Lin, and G. Wahba. "Multicategory Support Vector Machines: Theory and Application to the Classification of Microarray Data and Satellite Radiance Data". *Journal of the American Statistical Association*, Vol. 99, No. 465, pp. 67–81, 2004.

[Lee 07] J. A. Lee and M. Verleysen. *Nonlinear Dimensionality Reduction. Information Science and Statistics*, Springer, 2007.

[Lee 09] J. A. Lee and M. Verleysen. "Quality assessment of dimensionality reduction: Rank-based criteria". *Neurocomputing*, Vol. 72, No. 7-9, pp. 1431–1443, 2009.

[Lend 03] A. Lendasse, V. Wertz, and M. Verleysen. "Model Selection with Cross-Validations and Bootstraps - Application to Time Series Prediction with RBFN Models". In: *Proceedings of ICANN*, pp. 573–580, 2003.

[Less 06] S. Lessmann, R. Stahlbock, and S. F. Crone. "Genetic Algorithms for Support Vector Machine Model Selection". In: *International Joint Conference on Neural Networks*, pp. 3063 – 3069, July 2006.

[Lin 03] H.-T. Lin and C.-J. Lin. "A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods". Tech. Rep., National Taiwan University, March 2003.

[Liu 08] Q. Liu, Y. Zhang, and Z. Hu. “Extracting Decision Rules from Sigmoid Kernel”. In: *ADMA ’08: Proceedings of the 4th international conference on Advanced Data Mining and Applications*, pp. 294–304, Springer-Verlag, Berlin, Heidelberg, 2008.

[Loog 04] M. Loog. “Support Blob Machines. The sparsification of linear scale space”. In: T. Pajdla and J. Matas, Eds., *European Conference on Computer Vision*, pp. 14–24, Springer-Verlag, 2004.

[Loos 07a] G. Loosli, G. Gasso, and S. Canu. “Regularization Paths for *nu* -SVM and *nu* -SVR”. In: *Advances in Neural Networks - International Symposium on Neural Networks, ISNN*, pp. 486–496, 2007.

[Loos 07b] G. Loosli. *Méthodes à noyaux pour la détection de contexte.* PhD thesis, INSA de Rouen, Saint Etienne du Rouvray, October 2007.

[Luen 05] D. G. Luenberger. *Linear and Nonlinear Programming.* Springer, second Ed., 2005.

[Luen 69] D. G. Luenberger. *Optimization by Vector Space Methods.* John Wiley and Sons, Inc., 1969.

[Luss 08] R. Luss and A. D’Aspremont. “Support Vector Machine Classification with Indefinite Kernels”. In: J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds., *Advances in Neural Information Processing Systems 20*, pp. 953–960, MIT Press, Cambridge, MA, 2008.

[Luxb 07] U. von Luxburg. “A tutorial on spectral clustering”. *Statistics and Computing*, Vol. 17, No. 4, pp. 395–416, December 2007.

[Mark 59] H. M. Markowitz. *Portfolio selection: Efficient diversification of investments.* John Wiley and Sons, Inc., 1959.

[Mass 07] P. Massart. “Concentration Inequalities and Model Selection”. In: *Lecture Notes in Mathematics*, Springer, 2007.

[McLa 00] G. J. McLachlan and D. Peel. *Finite Mixture Models.* Wiley-IEEE, 2000.

[Meku 06] N. Mekuz and J. K. Tsotsos. “Parameterless Isomap with Adaptive Neighborhood Selection”. In: K. F. et al., Ed., *Proceedings of DAGM 2006*, Springer-Verlag Berlin Heidelberg, 2006.

[Merc 09] J. Mercer. “Functions of Positive and Negative Type, and their Connection with the Theory of Integral Equations”. *Royal Society of London Philosophical Transactions Series A*, Vol. 209, pp. 415–446, 1909.

[Miet 99] K. Miettinen. *Nonlinear Multiobjective Optimization.* Vol. 12 of *International Series in Operations Research and Management Science*, Kluwer Academic Publishers, Dordrecht, 1999.

[Moor 16] E. H. Moore. “On properly positive hermitian matrices”. *Bulletin of the American Mathematical Society*, Vol. 23, p. 59, 1916.

[Mord 03] A. Mordecai. *Nonlinear programming: analysis and methods.* Dover Publications, 2003.

[Moro 84] V. Morozov. *Methods for Solving Incorrectly Posed Problems.* Springer-Verlag, 1984.

[Ng 01] A. Y. Ng, M. I. Jordan, and Y. Weiss. “On spectral clustering: Analysis and an algorithm”. In: *Advances in Neural Information Processing Systems 14*, pp. 849–856, The MIT Press, 2001.

[Noce 99] J. Nocedal and S. Wright. *Numerical Optimization. Springer Series in Operations Research*, Springer-Verlag New York, Inc., 1999.

[Ong 04] C. S. Ong, S. Canu, and A. J. Smola. "Learning with non-positive kernels". In: *In Proc. of the 21st International Conference on Machine Learning (ICML)*, pp. 639–646, 2004.

[Osbo 99] M. R. Osborne, B. Presnell, and B. Turlach. "A New Approach to Variable Selection in Least Squares Problems". *IMA journal of numerical analysis*, Vol. 20, No. 3, pp. 389–403, 1999.

[Osun 97] E. Osuna, R. Freund, and F. Girosi. "Improved Training Algorithm for Support Vector Machines". In: *In Proc. of IEEE Workshop on Neural Networks and Signal Processing*, pp. 276–285, IEEE Press, 1997.

[Paie 03] J.-F. Paiement. *Généralisation des algorithmes de réduction de dimension.* PhD thesis, Université de Montréal, Montréal, November 2003.

[Pare 97] V. Pareto. *Cours d'Economie Politique.* Vol. 2, F. Rouge & Cie., Lausanne, Switzerland, 1897.

[Park 07] Park, M. Young, Hastie, and Trevor. "L1-regularization path algorithm for generalized linear models". *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, Vol. 69, No. 4, pp. 659–677, September 2007.

[Plat 00] J. C. Platt, N. Cristianini, and J. Shawe-taylor. "Large margin dags for multiclass classification". In: *Advances in Neural Information Processing Systems 12*, pp. 547–553, The MIT Press, 2000.

[Plat 99a] J. Platt. "Fast training of support vector machines using sequential minimal optimization". In: *Advances in Kernel Methods: Support Vector Learning*, pp. 185–208, MIT Press, Cambridge, MA, USA, 1999.

[Plat 99b] J. C. Platt. "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods". In: *Advances in Large Margin Classifiers*, pp. 61–74, The MIT Press, 1999.

[Pogg 89] T. Poggio and F. Girosi. "A theory of networks for approximation and learning". Tech. Rep., Artificial Intelligence Laboratory and Center for Biological Information Processing, Massachusetts Institute of Technology, 1989.

[Rako 07a] A. Rakotomamonjy and M. Davy. "One-class SVM regularization path and comparison with alpha seeding". In: *Proceedings of ESANN*, pp. 271–276, 2007.

[Rako 07b] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. "More efficiency in multiple kernel learning". In: *ICML '07: Proceedings of the 24th international conference on Machine learning*, pp. 775–782, ACM, New York, NY, USA, 2007.

[Rifk 04] R. Rifkin and A. Klautau. "In Defense of One-Vs-All Classification". *J. Mach. Learn. Res.*, Vol. 5, pp. 101–141, 2004.

[Rifk 07] R. M. Rifkin and R. A. Lippert. "Value Regularization and Fenchel Duality". *J. Mach. Learn. Res.*, Vol. 8, pp. 441–479, 2007.

[Rosa 04] L. Rosasco, E. D. Vito, A. Caponnetto, M. Piana, and A. Verri. "Are loss functions all the same?". *Neural Computation*, Vol. 16, No. 5, pp. 1063–1076, 2004.

[Ross 07a] S. Rosset, G. Swirszcz, N. Srebro, and J. Zhu. "L_1 Regularization in Infinite Dimensional Feature Spaces". In: *Proc. of Conference on Learning Theory*, pp. 544–558, 2007.

[Ross 07b] S. Rosset and J. Zhu. "Piecewise Linear Regularized Solution Paths". *Annals of Statistics*, Vol. 35, No. 3, pp. 1012–1030, 2007.

[Ross 08] S. Rosset. "Bi-level path following for cross validated solution of kernel quantile regression". In: *ICML '08: Proceedings of the 25th international conference on Machine learning*, pp. 840–847, ACM, New York, NY, USA, 2008.

[Rowe 00] S. T. Roweis and L. K. Saul. "Nonlinear Dimensionality Reduction by Locally Linear Embedding". *Science*, Vol. 290, No. 5500, pp. 2323 – 2326, December 2000. DOI: 10.1126/science.290.5500.2323.

[Rudi 76] W. Rudin. *Principles of Mathematical Analysis.* McGraw-Hill, third Ed., 1976.

[Saul 00] L. K. Saul and S. T. Roweis. "An Introduction to Locally Linear Embedding". Tech. Rep., ATT Labs-Research, 2000.

[Saul 05] L. K. Saul, K. Q. Weinberger, F. Sha, J. Ham, and D. D. Lee. "Spectral methods for dimensionality reduction". In: B. Schölkopf, O. Chapelle, and A. Zien, Eds., *Semisupervised Learning*, pp. 279–294, The MIT Press, : Cambridge, MA, 2005.

[Schi 07] W. Schirotzek. *Nonsmooth Analysis.* Springer-Verlag, Berlin, Heidelberg, New York, 2007.

[Schl 01] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* The MIT Press, Cambridge, MA, USA, 2001.

[Schl 97] B. Schölkopf. *Support Vector Learning.* PhD thesis, Technische Universität Berlin, Munich, Germany, 1997.

[Schm 07] M. Schmidt, G. Fung, and R. Rosales. "Fast Optimization Methods for L1 Regularization: A Comparative Study and Two New Approaches". In: *ECML '07: Proceedings of the 18th European conference on Machine Learning*, pp. 286–297, Springer-Verlag, Berlin, Heidelberg, 2007.

[Scho 01] B. Schölkopf, R. Herbrich, A.Smola, and R. Williamson. "A generalized representer theorem". In: *Proceedings of the Annual Conference on Computational Learning*, pp. 416–426, 2001.

[Sell 99] M. Sellathurai and S. Haykin. "The separability theory of hyperbolic tangent kernels and support vector machines for pattern classification". In: *ICASSP '99: Proceedings of the Acoustics, Speech, and Signal Processing, 1999. on 1999 IEEE International Conference*, pp. 1021–1024, IEEE Computer Society, Washington, DC, USA, 1999.

[Shal 07] S. Shalev-Shwartz, Y. Singer, and N. Srebro. "Pegasos: Primal Estimated sub-GrAdient SOlver for SVM". In: *ICML '07: Proceedings of the 24th international conference on Machine learning*, pp. 807–814, ACM, New York, NY, USA, 2007.

[Shaw 04] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, New York, NY, USA, 2004.

[Sind 05a] V. Sindhwani, P. Niyogi, and M. Belkin. "Beyond the point cloud: from transductive to semi-supervised learning.". In: *Proceedings of the International Conference on Machine Learning*, 2005.

[Sind 05b] V. Sindhwani, P. Niyogi, M. Belkin, and S. Keerthi. "Linear manifold regularization for large scale semi-supervised learning". In: *Proceedings of ICML Workshop on Learning with partially classified training data*, 2005.

[Sind 06] V. Sindhwani, S. Keerthi, and O. Chapelle. “Deterministic annealing for semi-supervised kernel machines”. In: *In Proc. International Conference on Machine Learning*, 2006.

[Smol 03] A. Smola and R. Kondor. “Kernels and Regularization on Graphs”. In: *Proc. Conf. on Learning Theory and Kernel Machines*, pp. 144–158, 2003.

[Sonn 06] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. “Large Scale Multiple Kernel Learning”. *Journal of Machine Learning Research*, Vol. 7, pp. 1531–1565, July 2006.

[Stei 07] I. Steinwart. “How to compare different loss functions and their risks”. Tech. Rep. LA-UR-05-7016, Los Alamos National Laboratory, 2007.

[Steu 86] R. E. Steuer. *Multiple Criteria Optimization (Probability & Mathematical Statistics).* John Wiley & Sons Inc, May 1986.

[Stew 03] C. V. Stewart, C. ling Tsai, and B. Roysam. “The Dual-Bootstrap Iterative Closest Point Algorithm with Application to Retinal Image Registration”. *IEEE Trans. Med. Img*, Vol. 22, pp. 1379–1394, 2003.

[Tene 00] J. B. Tenenbaum, V. de Silva, and J. C. Langford. “A Global Geometric Framework for Nonlinear Dimensionality Reduction”. *Science*, Vol. 290, pp. 2319–2323, 2000.

[Teo 07] C. H. Teo, A. Smola, S. V. Vishwanathan, and Q. V. Le. “A scalable modular convex solver for regularized risk minimization”. In: *KDD ’07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 727–736, ACM, New York, NY, USA, 2007.

[Tewa 05] A. Tewari and P. L. Bartlett. “On the consistency of multiclass classification methods”. In: *In Proceedings of the 18th Conference on Computational Learning Theory (COLT*, pp. 143–157, Springer, 2005.

[Tibs 05] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight. “Sparsity and smoothness via the fused lasso”. *Journal Of The Royal Statistical Society Series B*, Vol. 67, No. 1, pp. 91–108, 2005.

[Tibs 96] R. Tibshirani. “Regression shrinkage and selection via the lasso”. *Journal of the Royal Statistical Society*, Vol. 46, pp. 431–439, 1996.

[Tikh 79] A. N. Tikhonov and V. Y. Arsenin. “Solutions of Ill-Posed Problems”. *SIAM Review*, Vol. 21, pp. 266–267, April 1979.

[Tsud 07] K. Tsuda. “Entire regularization paths for graph data”. In: *ICML ’07: Proceedings of the 24th international conference on Machine learning*, pp. 919–926, ACM, New York, NY, USA, 2007.

[Vand 01] R. J. Vanderbei. *Linear Programming: Foundations and Extensions.* KAP, 2001.

[Vapn 77] V. Vapnik and A. Sterin. “On structural risk minimization or overall risk in a problem of pattern recognition”. *Automation and Remote Control*, Vol. 10, No. 3, pp. 1495–1503, 1977.

[Vapn 79] V. Vapnik and A. Cervonenkis. *Theorie der Zeichenerkennung.* Akademie Verlag, Berlin, 1979.

[Vapn 82] V. Vapnik. *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics (Springer Series in Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982.

[Vapn 99] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, NY, November 1999.

[Varm 06] S. Varma and R. Simon. "Bias in error estimation when using cross-validation for model selection.". *BMC Bioinformatics*, Vol. 7, No. 91, 2006.

[Verl 03] M. Verleysen. "Learning high-dimensional data". In: S. Ablameyko, L. Goras, M. Gori, and V. Piuri, Eds., *Limitations and future trends in neural computation*, Chap. 7, pp. 141–162, IOS Press, 2003.

[Wang 05] J. Wang, Z. Zhang, and H. Zha. "Adaptive Manifold Learning". In: Saul, Weiss, and Bottou, Eds., *Advances in Neural Information Processing Systems 17*, pp. 1473–1480, The MIT Press, Cambridge, MA, 2005.

[Wang 06a] G. Wang, T. Yeund, and F. Lochovsky. "Solution path of semi-supervised classification with manifold regularization". In: *Proceedings of 6th International Conference on Data Mining*, pp. 1124–1129, 2006.

[Wang 06b] L. Wang, J. Zhu, and H. Zou. "The doubly regularized support vector machine". *Statistica Sinica*, Vol. 16, pp. 589–615, 2006.

[Wang 06c] L. Wang, M. D. Gordon, and J. Zhu. "Regularized Least Absolute Deviations Regression and an Efficient Algorithm for Parameter Tuning". In: *ICDM '06: Proceedings of the Sixth International Conference on Data Mining*, pp. 690–700, IEEE Computer Society, Washington, DC, USA, 2006.

[Weim 08] M. Weimer, A. Karatzoglou, Q. Le, and A. Smola. "COFI RANK - Maximum Margin Matrix Factorization for Collaborative Ranking". In: J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds., *Advances in Neural Information Processing Systems 20*, pp. 1593–1600, The MIT Press, Cambridge, MA, 2008.

[Wein 04] K. Weinberger and L. Saul. "Unsupervised Learning of Image Manifolds by Semidefinite Programming". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-04) Washington D.C.*, 2004.

[Weis] E. W. Weisstein. "Minimum Spanning Tree". From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/MinimumSpanningTree.html.

[West 04] J. Weston, C. Leslie, D. Zhou, and W. S. Noble. "Semi-supervised protein classification using cluster kernels". In: *Advances in Neural Information Processing Systems 16*, pp. 595–602, The MIT Press, Cambridge, MA, USA,, 2004.

[West 98] J. Weston and C. Watkins. "Multi-class support vector machines". Tech. Rep. CSD-TR-98-04, Department of Computer Science, Royal Holloway College, 1998.

[Wu 07] M. Wu and B. Schölkopf. "Transductive Classification via Local Learning Regularization". In: X. S. Meila, M., Ed., *11th International Conference on Artificial Intelligence and Statistics*, pp. 628–635, Microtome, Brookline, MA, USA, 03 2007.

[Xu 07] J. Xu, T.-Y. Liu, and H. Li. "The OHSUMED Datasets in LETOR". Tech. Rep., Microsoft Research Asia, May 2007.

[Yang 06] L. Yang. "Building k-Connected Neighborhood Graphs for Isometric Data Embedding". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28, No. 5, pp. 827–831, 2006.

[Yao 07] Y. Yao and Y. Lee. "Another look at linear programming for feature selection via methods of regularization". Tech. Rep. 800, The Ohio State University, 2007.

[Yao 08] Y. Yao. "Efficient Linear Programming Algorithm for Functional Component Pursuit". 2008. Talk.

[Yuan 06] M. Yuan and Y. Lin. "Model selection and estimation in regression with grouped variables". *Journal of the Royal Statistical Society, Series B*, Vol. 68, pp. 49–67, 2006.

[Yuan 07] M. Yuan and Y. Lin. "On the non-negative garrote estimator". *Journal Of The Royal Statistical Society Series B*, Vol. 69, No. 2, pp. 143–161, 2007.

[Zang 81] W. I. Zangwill and C. B. Garcia. *Pathways to Solutions, Fixed Points, and Equilibria.* Prentice-Hall series in computational mathematics, 1981.

[Zapi 07] K. Zapién, G. Gasso, and S. Canu. "Estimation of tangent planes for neighborhood graph correction". In: *Proceedings of ESANN - European Symposium on Artificial Neural Networks*, pp. 397–402, 2007.

[Zapi 08a] K. Zapién, T. Gärtner, G. Gasso, and S. Canu. "Model Selection for Ranking SVM Using Regularization Path". In: *Proceedings of CAp, Conférence d'apprentissage*, Porquerolles, France, 2008.

[Zapi 08b] K. Zapién, T. Gärtner, G. Gasso, and S. Canu. "Regularisation Path for Ranking SVM". In: *Proceedings of ESANN 2008*, Bruges, Belgium, 2008.

[Zapi 09] K. Zapién, G. Gasso, T. Gärtner, and S. Canu. "Model Selection for Ranking SVM Using Regularization Path". 2009. To appear.

[Zhan 04] T. Zhang. "An Infinity-sample Theory For Multi-category Large Margin Classification". In: *Advances in Neural Information Processing Systems 16*, 2004.

[Zhao 06] D. Zhao and L. Yang. "Incremental Construction of Neighborhood Graphs for Nonlinear Dimensionality Reduction.". In: *Proceedings of ICPR (3)*, pp. 177–180, IEEE Computer Society, 2006.

[Zhou 04] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. "Learning with local and global consistency". In: L. S. S. Thrun and B. Schölkopf, Eds., *Advances in Neural Information Processing Systems 16*, The MIT Press, Cambridge, MA, 2004.

[Zhou 05] A. Zhou, Q. Zhang, Y. Jin, E. Tsang, and T. Okabe. "A model-based evolutionary algorithm for bi-objective optimization". In: *Congress on Evolutionary Computation*, pp. 2568–2575, IEEE Press, Edinburgh, 2005.

[Zhu 03] X. Zhu, Z. Ghahramani, and J. Lafferty. "Semi-supervised learning using Gaussian fields and harmonic functions". In: *20th International Conference on Machine Learning (ICML)*, 2003.

[Zhu 04] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. "L1-norm support vector machines". In: S. Thrun, L. Saul, and B. Schölkopf, Eds., *Advances in Neural Information Processing Systems 16*, MIT Press, Cambridge, MA, 2004.

[Zhu 05a] X. Zhu. "Semi-Supervised Learning Literature Survey". Tech. Rep. 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

[Zhu 05b] X. Zhu. *Semi-supervised learning with graphs.* PhD thesis, Carnegie Mellon University, 2005.

[Zou 05] H. Zou and T. Hastie. "Regularization and variable selection via the Elastic Net". *Journal of the Royal Statistical Society B*, Vol. 67, pp. 301–320, 2005.

List of Tables

List of Figures

List of Symbols and Acronyms

Acronyms

ERM	Empirical Risk Minimization
i.i.d.	Independent and identically distributed
OP	Optimization Problem
RKHS	Reproducing Kernel Hilbert Space
SVC	Support Vector Machine for Classification
SVM	Support Vector Machine
w.r.t.	with respect to

Statistics and Probability

$\mathbb{E}$	Expectation
L	Likelihood Function
$\mathbb{P}$	Probability function

Sets

2^S	power set of set S
$\mathcal{A}$	Active Set
$\mathrm{Dom}(f)$	Effective domain of function f
$\mathcal{I}_{(\cdot)}$	Any of the set partitions on the regularization path: $\mathcal{I}_\alpha$, $\mathcal{I}_1$ or $\mathcal{I}_0$
$\int(A)$	Interior of a set A
$\bar{\mathrm{c}}\mathrm{o}^*(A)$	closed convex hull of the set A ([Schi 07][])

Spaces

$B(\mathbf{x}_0, \epsilon)$	Ball of radius ϵ around $\mathbf{x}_0$
$\mathcal{H}$	Hilbert space
$\mathbb{R}$	real numbers
$\mathbb{R}^+$	positive real numbers
$\bar{\mathbb{R}}$	extended real numbers
$\mathcal{X}$	Vector Space

$\mathcal{Y}$	Normed Space

Functions and operators

$\angle(\mathbf{x}, \mathbf{y})$	Angle between vector $\mathbf{x}$ and vector $\mathbf{y}$
$\angle(\mathbf{x}, P)$	Perpendicular angle between vector $\mathbf{x}$ and plane P
Ω	Regularization function
δ_M	Indicator functional of M
f	Decision Function
f	$\hat{f}^{\lambda}$, resulting optimal decision function with regularization parameter λ
f^t	$\hat{f}^{\lambda^t}$, resulting optimal decision function with regularization parameter λ^t
h	Constraint Function
J	Primal objective function
$\mathbf{k}$	Kernel function
$\boldsymbol{\mathcal{L}}$	Laplacian
$\mathbf{L}$	Lagrange Function
$\mathcal{L}$	Empirical Risk
ℓ	Loss Function
W	Dual objective function

Data characteristics

d	Reduced dimensionality
$\mathcal{D}$	Original dimensionality
l	Number of labeled samples
n	Total number of samples
u	Number of unlabeled samples

Matrix and Vectors

$\mathbf{d}$	Vector direction
$D_{\mathcal{I}}$	Submatrix formed with the $\mathcal{I}$ rows and columns of D, with D a diagonal matrix
$M_{\mathcal{I}_1,\mathcal{I}_2}$	Submatrix formed with the $\mathcal{I}_1$ rows and $\mathcal{I}_2$ columns of M
$\mathbf{v}$	Vector
$\mathbf{v}_{\mathcal{I}}$	Subvector of $\mathbf{v}$ formed by the corresponding entries to the set $\mathcal{I}$
$\mathbf{x}$	Vector

Index