Ecole Normale Supérieure de Cachan

Habilitation à Diriger des Recherches Thesis

Weakly Supervised Learning for Structured Output Prediction

Author:
M. Pawan Kumar

Guarantor:
Prof. Nicolas Vayatis

Referees:
Prof. Cristoph Schnoerr
Prof. Marc Schoenauer
Prof. Cristian Sminchisescu

Jury:
Dr. Francis Bach
Dr. Andrew Fitzgibbon
Prof. Nikos Paragios
Prof. Ramin Zabih

A thesis submitted in fulfilment of the requirements
for the degree of Habilitation à Diriger des Recherches

in the

Centre de Mathématiques et de Leurs Applications

February 2014

Declaration of Authorship

I, M. Pawan Kumar, declare that this thesis titled, 'Weakly Supervised Learning for Structured Output Prediction' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

Publications

Publications After PhD

P.-Y. Baudin, D. Goodman, K. Puneet, N. Azzabou, P. Carlier, N. Paragios and M. Pawan Kumar
"Discriminative Parameter Estimation for Random Walks Segmentation"
In Proceedings of International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI), 2013

W. Zaremba, M. Pawan Kumar, A. Gramfort and M. Blaschko
"Learning from M/EEG Data with Variable Brain Activation Delay"
In Proceedings of International Conference on Information Processing in Medical Imaging (IPMI), 2013

M. Pawan Kumar, B. Packer and D. Koller
"Modeling Latent Variable Uncertainty for Loss-based Learning"
In Proceedings of International Conference on Machine Learning (ICML), 2012

K. Miller, M. Pawan Kumar, B. Packer, D. Goodman and D. Koller
"Max-Margin Min-Entropy Models"
In Proceedings of Conference on Artificial Intelligence and Statistics (AISTATS), 2012

M. Pawan Kumar, H. Turki, D. Preston and D. Koller
"Learning Specific-Class Segmentation from Diverse Data"
In Proceedings of International Conference on Computer Vision (ICCV), 2011

M. Pawan Kumar, O. Veksler and P. Torr
"Improved Moves for Truncated Convex Models"
In Journal of Machine Learning Research (JMLR), 2011

M. Pawan Kumar, B. Packer and D. Koller
"Self-Paced Learning for Latent Variable Models"
In Proceedings of Advances in Neural Information Processing Systems (NIPS), 2010

M. Pawan Kumar and D. Koller
"Efficiently Selecting Regions for Scene Understanding"
In Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR), 2010

P. Kohli and M. Pawan Kumar
"Energy Minimization for Linear Envelope MRFs"

In Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR), 2010

M. Pawan Kumar and D. Koller
"Learning a Small Mixture of Trees"
In Proceedings of Advances in Neural Information Processing Systems (NIPS), 2009

M. Pawan Kumar, A. Zisserman and P. Torr
"Efficient Discriminative Learning of Parts-based Models"
In Proceedings of International Conference on Computer Vision (ICCV), 2009

M. Pawan Kumar and D. Koller
"MAP Estimation of Semi-Metric MRFs via Hierarchical Graph Cuts"
In Proceedings of Conference on Uncertainity in Artificial Intelligence (UAI), 2009

M. Pawan Kumar and P. Torr
"Improved Moves for Truncated Convex Models"
In Proceedings of Advances in Neural Information Processing Systems (NIPS), 2008

M. Pawan Kumar and P. Torr
"Efficiently Solving Convex Relaxations for MAP Estimation"
In Proceedings of International Conference on Machine Learning (ICML), 2008

Publications During PhD

M. Pawan Kumar, P. Torr and A. Zisserman
"OBJCUT: Efficient Segmentation using Top-Down and Bottom-Up Cues"
In Pattern Analysis and Machine Intelligence (PAMI), 2010

M. Pawan Kumar, V. Kolmogorov and P. Torr
"An Analysis of Convex Relaxations for MAP Estimation of Discrete MRFs"
In Journal of Machine Learning Research (JMLR), 2009

P. Kohli, M. Pawan Kumar and P. Torr
"P3 and Beyond: Move Making Algorithms for Solving Higher Order Functions"
In Pattern Analysis and Machine Intelligence (PAMI), 2009

M. Pawan Kumar, P.H.S. Torr and A. Zisserman
"Learning Layered Motion Segmentations of Video"
In International Journal of Computer Vision (IJCV), 2008

M. Pawan Kumar, V. Kolmogorov and P. Torr
"An Analysis of Convex Relaxations for MAP Estimation"
In Proceedings of Advances in Neural Information Processing Systems (NIPS), 2007

M. Pawan Kumar, P. Torr and A. Zisserman
"An Invariant Large Margin Nearest Neighbour Classifier"
In Proceedings of International Conference on Computer Vision (ICCV), 2007

P. Kohli, M. Pawan Kumar and P. Torr
"P3 and Beyond: Solving Energies with Higher Order Cliques"
In Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR), 2007

M. Pawan Kumar, P. Torr and A. Zisserman
"Solving Markov Random Fields using Second Order Cone Programming Relaxations"
In Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR), 2006

M. Pawan Kumar and P. Torr
"Fast Memory-Efficient Generalized Belief Propagation"
In Proceedings of European Conference on Computer Vision (ECCV), 2006

M. Pawan Kumar, P. Torr and A. Zisserman
"Learning Layered Motion Segmentations of Video"
In Proceedings of International Conference on Computer Vision (ICCV), 2005

M. Pawan Kumar, P. Torr and A. Zisserman
"OBJ CUT"
In Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR), 2005

M. Pawan Kumar, P. Torr and A. Zisserman
"Learning Layered Pictorial Structures from Video"
In Proceedings of Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP), 2004

M. Pawan Kumar, P. Torr and A. Zisserman
"Extending Pictorial Structures for Object Recognition"
In Proceedings of British Machine Vision Conference (BMVC), 2004

Abstract

We consider the problem of learning the parameters of a structured output prediction model, that is, learning to predict elements of a complex interdependent output space that correspond to a given input. Unlike many of the existing approaches, we focus on the weakly supervised setting, where most (or all) of the training samples have only been partially annotated. Given such a weakly supervised dataset, our goal is to estimate accurate parameters of the model by minimizing the regularized empirical risk, where the risk is measured by a user-specified loss function. This task has previously been addressed by the well-known latent support vector machine (latent SVM) framework. We argue that, while latent SVM offers a computational efficient solution to loss-based weakly supervised learning, it suffers from the following three drawbacks: (i) the optimization problem corresponding to latent SVM is a difference-of-convex program, which is non-convex, and hence susceptible to bad local minimum solutions; (ii) the prediction rule of latent SVM only relies on the most likely value of the latent variables, and not the uncertainty in the latent variable values; and (iii) the loss function used to measure the risk is restricted to be independent of true (unknown) value of the latent variables.

We address the the aforementioned drawbacks using three novel contributions. First, inspired by human learning, we design an automatic self-paced learning algorithm for latent SVM, which builds on the intuition that the learner should be presented in the training samples in a meaningful order that facilitates learning: starting frome easy samples and gradually moving to harder samples. Our algorithm simultaneously selects the easy samples and updates the parameters at each iteration by solving a biconvex optimization problem. Second, we propose a new family of LVMs called max-margin min-entropy (M3E) models, which includes latent SVM as a special case. Given an input, an M3E model predicts the output with the smallest corresponding *Rényi entropy of generalized distribution*, which relies not only on the probability of the output but also the uncertainty of the latent variable values. Third, we propose a novel learning framework for learning with general loss functions that may depend on the latent variables. Specifically, our framework simultaneously estimates two distributions: (i) a conditional distribution to model the uncertainty of the latent variables for a given input-output pair; and (ii) a delta distribution to predict the output and the latent variables for a given input. During learning, we encourage agreement between the two distributions by minimizing a loss-based dissimilarity coefficient. We demonstrate the efficacy of our contributions on standard machine learning applications using publicly available datasets.

Contents

Chapter 1

Introduction

1.1 Structured Output Prediction

Structured output prediction [53] refers to the task of predicting elements of a complex interdependent output space that correspond to a given input. Recent years have witnessed its increasing popularity in machine learning due to its ability to provide an elegant formulation for several applications in computer vision [9, 28, 40, 54], medical image analysis [2, 3, 68], natural language processing [39, 44] and many other areas of research.

A central problem in structured output prediction is to learn the parameters of the prediction model. Broadly speaking, there are two paradigms for parameter estimation: empirical risk minimization and probabilistic modeling. In empirical risk minimization, the parameters are learned by minimizing the regularized empirical risk, where the risk is measured by a user-specified loss function. In probabilistic modeling, a parameter estimate is made by means of the marginal likelihood function of the observed data; the model is typically represented by means of a parameter point estimate [8, 33] or an approximation to the posterior distribution over the model parameters [8, 10, 48, 49]. In this thesis, we will focus on the former paradigm, that is, empirical risk minimization.

Most of the research has focused on developing supervised learning techniques for structured prediction. Specifically, it is typically assumed that each sample of the training has a fully observed ground-truth annotation. A well-known example of such a learner is the structured output support vector machine (or SVM for short) [63, 65], whose parameters can be estimated efficiently by solving a convex quadratic program [32, 58, 63, 65]. Despite this progress and the practical relevance of the considered applications, fully supervised learning has made limited impact. This lack of success is due to the fact that

collecting annotated datasets is often prohibitively expensive, especially for complex models that require millions of training samples before their performance gets saturated. In fact, one might argue that for many applications it is impossible to obtain supervised annotations. For example, consider the task of region-based semantic segmentation [21, 37, 43], that is, grouping image pixels into coherent regions such that the semantic classes of the regions can be predicted accurately. The choice of regions will depend on the high-dimensional feature vector being employed, and a human annotator would be incapable of providing a reasonable guess for the optimal pixel-to-region assignment based on the description of the features. The lack of full supervision is even more common in applications related to medical image analysis and computational biology, where the ground-truth annotations cannot be provided by even the experts of the field.

In order to address the deficiencies of supervised learning, recent research has started focusing on learning from weakly supervised datasets, where a majority (or all) of the training samples are only partially annotated. Such datasets can be constructed at a reasonable cost using freely available data on the Internet, together with crowd sourcing solutions like Amazon's Mechanical Turk to obtain partial annotations. However, the convenience of using a large, inexpensive, weakly supervised dataset comes at the cost of solving a significantly more challenging machine learning problem. Specifically, the task of learning a structured output predictor from weakly supervised datasets can be interpreted as learning the parameters of a latent variable model (LVM), that is, a model that contains latent (or hidden) variables whose values are not observed during training. For example, consider the well-known LVM known as latent support vector machine (or latent SVM for short) [17, 60, 66], which is of particular interest to us in this thesis. Latent SVM extends the SVM formulation by modeling the missing information in the annotations of the training samples as latent variables. It modifies the prediction task to consider both the output space and the latent space, which allows it to formulate the problem of parameter estimation as minimizing a regularized upper bound on the empirical prediction risk. Similar to an SVM, the risk is measured by a user-specified loss function. While latent SVM offers a computational efficient solution to loss-based weakly supervised learning, it suffers from the following three drawbacks: (i) the optimization problem corresponding to latent SVM is a difference-of-convex program, which is non-convex, and hence susceptible to bad local minimum solutions; (ii) the prediction rule of latent SVM only relies on the most likely value of the latent variables, and not the uncertainty in the latent variable values; and (iii) the loss function used to measure the risk is restricted to be independent of true (unknown) value of the latent variables.

In this work, we overcome the aforementioned drawbacks of existing loss-based weakly supervised learning frameworks with the help of the following three contributions.

- We propose a novel iterative self-paced learning algorithm for solving the optimization problem corresponding to latent SVM, which builds on the intuition that the learner should be presented in the training samples in a meaningful order that facilitates learning: starting from easy samples and gradually moving to harder samples. Our algorithm simultaneously selects the easy samples and updates the parameters at each iteration by solving a biconvex optimization problem.

- We propose a new family of LVMs called max-margin min-entropy (M3E) models, which includes latent SVM as a special case. Given an input, an M3E model predicts the output with the smallest corresponding *Rényi entropy of generalized distribution* [56], which relies not only on the probability of the output but also the uncertainty of the latent variable values. Similar to latent SVM, the parameters of an M3E model are learned by minimizing a regularized upper bound on the empirical risk, which is measured using a loss function that is independent of the true value of the latent variables.

- We propose a novel learning framework that simultaneously estimates two distributions: (i) a conditional distribution to model the uncertainty of the latent variables for a given input-output pair; and (ii) a delta distribution to predict the output and the latent variables for a given input. During learning, we encourage agreement between the two distributions by minimizing a loss-based dissimilarity coefficient [55]. Our approach generalizes latent SVM in two important ways: (i) it models the uncertainty over latent variables instead of relying on a pointwise estimate; and (ii) it allows the use of loss functions that depend on latent variables, which greatly increases its applicability.

Before presenting our work in detail, we briefly describe the SVM and latent SVM frameworks in the remainder of this chapter. This will allow us to set up the notation that will be used throughout the thesis, as well as help put our contributions in context.

1.2 Structured Output Support Vector Machine

Given an input vector $\mathbf{x} \in \mathcal{X}$, a structured output support vector machine (henceforth referred to as simply SVM) provides a linear prediction rule for the structured output $\mathbf{y} \in \mathcal{Y}$. In more detail, we denote the joint feature vector of an input $\mathbf{x}$ and an output $\mathbf{y}$ by $\Psi(\mathbf{x}, \mathbf{y})$. The joint feature vector captures the relationship between the input and the output. For example, consider the application of detecting a car in an image. For simplicity, let us assume that an image can contain at most one car. In this case, the input $\mathbf{x}$ is the image, the output $\mathbf{y}$ is the bounding box of the car for a positive image

(that is, an image containing a car) and ϕ (null) for a negative image (that is, an image not containing a car). The joint feature vector $\Psi(\mathbf{x}, \mathbf{y})$ can be the HOG descriptor [13] of the bounding box specified by $\mathbf{y}$ for positive images, and $\mathbf{0}$ (that is, a vector of appropriate dimensions whose elements are all equal to 0) for negative images.

Given an SVM, parameterized by $\mathbf{w}$, the prediction rule can be formulated as the following optimization problem:

$$\mathbf{y}(\mathbf{w}) = \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmax}}\, \mathbf{w}^\top \Psi(\mathbf{x}, \mathbf{y}). \tag{1.1}$$

We refer to the quantity $\mathbf{w}^\top \Psi(\mathbf{x}, \mathbf{y})$ as the score of an output $\mathbf{y}$ for a given input $\mathbf{x}$. In the above car detection example, given an image $\mathbf{x}$, the SVM will either return a bounding box $\mathbf{y}$ if there exists a bounding box with a score greater than 0, or it will return ϕ.

1.2.1 Learning an SVM

The problem of learning an SVM is to estimate the parameter $\mathbf{w}$ from a training dataset. In this section, we will assume that the training dataset is fully supervised, that is, the complete ground-truth annotation of each training sample is known. Formally, we denote the training dataset by $\mathcal{D} = \{\mathbf{s}_i = (\mathbf{x}_i, \mathbf{y}_i), i = 1, \cdots, n\}$, that is, the training dataset consists of n samples, each of which is an input-output pair. Furthermore, we assume a user-specified loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$, which is used to measure the risk of prediction. Specifically, $\Delta(\mathbf{y}_1, \mathbf{y}_2)$ measures the difference between the outputs $\mathbf{y}_1$ and $\mathbf{y}_2$. The value of the loss is assumed to be non-negative and equal to 0 if and only if $\mathbf{y}_1 = \mathbf{y}_2$. For example, consider the task of learning an SVM for car detection. In this application, an appropriate loss function might be one that returns 0 if both the outputs belong to the same class (either both consider the image $\mathbf{x}$ to be positive, or both consider $\mathbf{x}$ to be negative), and 1 otherwise.

The parameters $\mathbf{w}$ are learned by minimizing a regularized upper bound on the empirical risk, where the empirical risk is defined as

$$\min_{\mathbf{w}} \sum_i \Delta(\mathbf{y}_i, \mathbf{y}_i(\mathbf{w})). \tag{1.2}$$

Here, $\mathbf{y}_i(\mathbf{w})$ refers to the prediction made using the parameters $\mathbf{w}$ as specified in equation (1.1). Formally, learning involves solving the following convex quadratic program:

$$\begin{aligned} &\min_{\mathbf{w}, \xi_i \geq 0} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{n} \sum_{i=1}^{n} \xi_i, \\ \text{s.t.} \quad &\mathbf{w}^\top \left(\Psi(\mathbf{x}_i, \mathbf{y}_i) - \Psi(\mathbf{x}_i, \hat{\mathbf{y}}_i)\right) \geq \Delta(\mathbf{y}_i, \hat{\mathbf{y}}_i) - \xi_i, \forall \hat{\mathbf{y}}_i \in \mathcal{Y}, i = 1, \cdots, n, \end{aligned} \tag{1.3}$$

where $C \geq 0$ is a user-specified constant, and ξ_i is the slack variable corresponding to the sample $\mathbf{s}_i$. Note that in this thesis we limit our discussion to the margin rescaling formulation of SVM (shown above). However, all of our methods can be trivially modified for the slack rescaling formulation [65]. Intuitively, the above problem introduces a margin between the score of the ground-truth output and the score of any other output. The desired margin is proportional to the loss value between the ground-truth and the corresponding output.

1.2.2 Optimization

Although the above problem appears to be a difficult optimization task due to a large number of constraints, there exist several polynomial-time algorithms that obtain a solution up to an arbitrary precision ϵ [32, 58, 63, 65]. In our work, we will use the 1-slack reformulation based cutting plane algorithm [32] for small datasets, and the stochastic subgradient algorithm [58] for large datasets. For completeness, we provide a brief description of the two methods in Algorithm 1 and Algorithm 2 respectively. We refer the interested reader to [32, 58] for details. Note that the key step in both the methods is the same, namely solving an optimization problem that maximizes the sum of the score of the output and the loss function (equation (1.4) and equation (1.6)). Henceforth, we will refer to this step as loss augmented inference. Intuitively, loss augmented inference requires us to find the most violated constraint of problem (1.3) for a given $\mathbf{w}$, that is, an output $\hat{\mathbf{y}}_i$ for each sample $\mathbf{s}_i$ that has a large score and a large loss value.

1.3 Latent Support Vector Machine

As mentioned earlier, in many applications it is onerous, or even impossible, to obtain a fully supervised training dataset. In order to estimate the parameters in a weakly supervised setting, it is common practice to use a latent variable model known as the latent support vector machine (henceforth referred to as simply latent SVM). Latent SVM extends the SVM framework by modeling the missing information in the annotations as latent (or hidden) variables $\mathbf{h} \in \mathcal{H}$. We denote the joint vector vector of an input $\mathbf{x}$, an output $\mathbf{y}$ and a latent variable $\mathbf{h}$ as $\Psi(\mathbf{x}, \mathbf{y}, \mathbf{h})$. For example, let us consider the car detection example, where the input $\mathbf{x}$ is an image. Unlike the supervised setting, the output $\mathbf{y}$ only specifies whether the image is positive ($\mathbf{y} = 1$) or negative ($\mathbf{y} = 0$). The missing information in the output, that is, the bounding box of the car is modeled as a latent variable $\mathbf{h}$. Specifically, $\mathbf{h}$ is the bounding box of the car when $\mathbf{y} = 1$ and ϕ (null)

Algorithm 1 *The 1-slack cutting plane algorithm for learning an* SVM.

input $\mathcal{D} = \{\mathbf{s}_i = (\mathbf{x}_i, \mathbf{y}_i), i = 1, \cdots, n\}$, C, ϵ.

1: Initialize $\mathbf{w} \leftarrow \mathbf{0}$, $\mathcal{W} \leftarrow \{\}$.
2: **repeat**
3: Update the solution as

$$\begin{aligned} (\mathbf{w}, \xi) \leftarrow \quad & \underset{\mathbf{w}, \xi \geq 0}{\operatorname{argmin}} \frac{1}{2}||\mathbf{w}||^2 + C\xi \\ \text{s.t.} \quad & \frac{1}{n}\mathbf{w}^\top \left(\sum_i \Psi(\mathbf{x}_i, \mathbf{y}_i) - \Psi(\mathbf{x}_i, \hat{\mathbf{y}}_i) \right) \geq \frac{1}{n} \sum_i \Delta(\mathbf{y}_i, \hat{\mathbf{y}}_i) - \xi, \\ \forall \quad & (\hat{y}_1, \cdots, \hat{y}_n) \in \mathcal{W}, \end{aligned}$$

4: **for** $i = 1$ to n **do**
5: Solve the following loss augmented inference problem

$$\hat{\mathbf{y}}_i = \underset{\overline{y} \in \mathcal{Y}}{\operatorname{argmax}} \Delta(\mathbf{y}_i, \overline{\mathbf{y}}) + \mathbf{w}^\top \Psi(\mathbf{x}_i, \overline{\mathbf{y}}). \tag{1.4}$$

6: **end for**
7: $\mathcal{W} \leftarrow \mathcal{W} \cup \{(\hat{\mathbf{y}}_1, \cdots, \hat{\mathbf{y}}_n)\}$.
8: **until** $\frac{1}{n}\sum_i \left(\Delta(\mathbf{y}_i, \hat{\mathbf{y}}_i) + \mathbf{w}^\top \Psi(\mathbf{x}_i, \hat{\mathbf{y}}_i) - \mathbf{w}^\top \Psi(\mathbf{x}_i, \mathbf{y}_i) \right) \leq \xi + \epsilon$.

Algorithm 2 *The stochastic subgradient descent algorithm for learning an* SVM.

input $\mathcal{D} = \{\mathbf{s}_i = (\mathbf{x}_i, \mathbf{y}_i), i = 1, \cdots, n\}$, C, maximum number of iterations T.

1: Initialize $\mathbf{w} \leftarrow \mathbf{0}$, $t = 0$.
2: **repeat**
3: $t \leftarrow t + 1$.
4: Choose a random subset $\mathcal{S}^t \subseteq \mathcal{D}$.
5: Compute the stochastic subgradient $\mathbf{g}^t$ over $\mathcal{S}^t$ as

$$\mathbf{g}^t = \frac{1}{|\mathcal{S}^t|} \sum_{i \in \mathcal{S}^t} \left(\Psi(\mathbf{x}_i, \hat{\mathbf{y}}_i) - \Psi(\mathbf{x}_i, \mathbf{y}_i) \right). \tag{1.5}$$

Here $\hat{\mathbf{y}}_i$ is obtained by solving the following loss augmented inference problem:

$$\hat{\mathbf{y}}_i = \underset{\overline{y} \in \mathcal{Y}}{\operatorname{argmax}} \Delta(\mathbf{y}_i, \overline{\mathbf{y}}) + \mathbf{w}^\top \Psi(\mathbf{x}_i, \overline{\mathbf{y}}). \tag{1.6}$$

6: Update $\mathbf{w} \leftarrow (1 - 1/t)\mathbf{w} - C\mathbf{g}^t/t$.
7: **until** t = T.

otherwise. The joint feature vector $\Psi(\mathbf{x}, \mathbf{y}, \mathbf{h})$ is the HOG descriptor [13] of the bounding box $\mathbf{h}$ when $\mathbf{y} = 1$ and $\mathbf{0}$ otherwise.

Given a latent SVM, parameterized by $\mathbf{w}$, the prediction rule is extended to optimize over both the output space and the latent space as follows:

$$(\mathbf{y}^*, \mathbf{h}^*) = \underset{\mathbf{y} \in \mathcal{Y}, \mathbf{h} \in \mathcal{H}}{\operatorname{argmax}} \mathbf{w}^\top \Psi(\mathbf{x}, \mathbf{y}, \mathbf{h}). \tag{1.7}$$

In our car detection example, given an image $\mathbf{x}$, the latent SVM will either label the image as positive and return a bounding box $\mathbf{h}$ if there exists a bounding box with a score that is greater than 0, or it will label the image as negative.

1.3.1 Learning a Latent SVM

Given a training dataset $\mathcal{D} = \{\mathbf{s}_i = (\mathbf{x}_i, \mathbf{y}_i), i = 1, \cdots, n\}$, the parameters of a latent SVM are estimated by minimizing a regularized upper bound on the empirical risk, where the risk is measured using a loss function $\Delta(\mathbf{y}_1, \mathbf{y}_2)$. In our car detection example, the loss function can be defined similar to the supervised case, that is, it is 0 if both the outputs belong to the same class, and 1 otherwise.

Formally, learning a latent SVM involves solving the following optimization problem:

$$\begin{aligned} & \min_{\mathbf{w}, \xi_i \geq 0} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i, \\ \text{s.t.} \quad & \max_{h_i \in \mathcal{H}} \mathbf{w}^\top \Psi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i) - \mathbf{w}^\top \Psi(\mathbf{x}_i, \hat{\mathbf{y}}_i, \hat{\mathbf{h}}_i) \geq \Delta(\mathbf{y}_i, \hat{\mathbf{y}}_i) - \xi_i, \\ & \forall \hat{\mathbf{y}}_i \in \mathcal{Y}, \forall \hat{h}_i \in \mathcal{H}, i = 1, \cdots, n. \end{aligned} \tag{1.8}$$

Intuitively, the above problem introduces a margin between the score of the ground-truth output together with the best value of the latent variable and any other pair of output and latent variables. The desired margin is proportional to the loss between the ground-truth and the corresponding output. Note that in a latent SVM formulation, the loss function can also depend on the value of the latent variable $\hat{\mathbf{h}}_i$. However, in all our applications, the loss function will only depend on the values of the outputs. We refer the interested reader to [66] for details.

1.3.2 Optimization

Problem (1.8) can be viewed as minimizing the sum of a convex and a concave function. This observation leads to a concave-convex procedure (CCCP) [67] outlined in Algorithm 3, which has been shown to converge to a local minimum or saddle point solution [61]. The algorithm has two main steps. First, imputing the latent variables (step 3), that is, finding the best values of the latent variables that are consistent with the given annotation. Henceforth, we will refer to this step as annotation consistent inference. In the context of CCCP, annotation consistent inference corresponds to approximating the concave function by a linear upper bound. Second, updating the value of the parameter using the values of the hidden variables. Note that updating the

parameters requires us to solve an SVM learning problem where the output $\mathbf{y}_i$ is now concatenated with the hidden variable $\mathbf{h}_i^*$. See [17, 60, 66] for more details.

Algorithm 3 *The* CCCP *algorithm for learning latent* SVM*.*

input $\mathcal{D} = \{\mathbf{s}_i = (\mathbf{x}_i, \mathbf{y}_i), i = 1, \cdots, n\}$, $\mathbf{w}_0$, C, ϵ.

1: $t \leftarrow 0$
2: **repeat**
3: Update $h_i^* = \arg\max_{h_i \in \mathcal{H}} \mathbf{w}_t^\top \Psi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i)$.
4: Update $\mathbf{w}_{t+1}$ by fixing the hidden variables for output $\mathbf{y}_i$ to $\mathbf{h}_i^*$ and solving the corresponding SVM problem.
5: $t \leftarrow t + 1$.
6: **until** Objective function cannot be decreased below tolerance ϵ.

1.4 Outline of the Thesis

The rest of the thesis is organized as follows. In chapter 2, we present the self-paced learning algorithm for estimating the parameters of a latent SVM. In chapter 3, we describe the max-margin min-entropy family of latent variable models that generalizes latent SVM for the case where the loss function is independent of the true value of the latent variables. In chapter 4, we consider a general loss function (which can dependent on both the output and the latent variables) and present a dissimilarity coefficient based learning framework. Chapter 5 discusses interesting directions for future research.

1.5 Relevant Publications

The work described in this thesis has appeared in the following publications.

- M. Pawan Kumar, B. Packer and D. Koller. Self-Paced Learning for Latent Variable Models. In Proceedings of Advances in Neural Information Processing Systems (NIPS), 2010
- M. Pawan Kumar, H. Turki, D. Preston and D. Koller. Learning Specific-Class Segmentation from Diverse Data. In Proceedings of International Conference on Computer Vision (ICCV), 2011
- K. Miller, M. Pawan Kumar, B. Packer, D. Goodman and D. Koller. Max-Margin Min-Entropy Models. In Proceedings of Conference on Artificial Intelligence and Statistics (AISTATS), 2012
- M. Pawan Kumar, B. Packer and D. Koller. Modeling Latent Variable Uncertainty for Loss-based Learning. In Proceedings of International Conference on Machine Learning (ICML), 2012

Chapter 2

Self-Paced Learning for Latent Support Vector Machine

2.1 Introduction

As seen in the previous chapter, by extending SVM to handle latent variables, the latent SVM framework significantly increases the scope of loss-based learning for structured output prediction. For example, in computer vision, we may wish to learn a model of a car from many positive and negative samples, that is, images containing and not containing cars respectively. However, the exact location of the cars may be unknown and could be modeled as latent variables. In medical diagnosis, learning to diagnose a disease based on symptoms can be improved by treating unknown or unobserved diseases as latent variables (to deal with confounding factors). Unlike the original SVM, parameter learning in latent SVM requires solving a non-convex optimization problem. Although [17, 60, 66] provide a concave-convex procedure (CCCP) for solving this problem, it is prone to getting stuck in a bad local minimum (with high training and generalization error).

Machine learning literature is filled with scenarios in which one is required to solve a non-convex optimization task, for example learning the parameters of a probabilistic latent variable model using expectation-maximization [50, 57, 62], or (weakly) supervised learning for perceptrons or deep belief nets [25, 26, 42]. A common approach for avoiding a bad local minimum in these cases is to use multiple runs with random initializations and pick the best solution amongst them (as determined, for example, by testing on a validation set). However, this approach is adhoc and computationally expensive as one may be required to use several runs to obtain an accurate solution. Bengio *et al.* [5] recently proposed an alternative method for training with non-convex objectives, called curriculum learning. The idea is inspired by the way children are taught: start with

easier concepts (for example, recognizing objects in simple scenes where an object is clearly visible) and build up to more complex ones (for example, cluttered images with occlusions). Curriculum learning suggests using the easy samples first and gradually introducing the learning algorithm to more complex ones. The main challenge in using the curriculum learning strategy is that it requires the identification of easy and hard samples in a given training dataset. However, in many real-world applications, such a ranking of training samples may be onerous or conceptually difficult for a human to provide — even if this additional human supervision can be provided, what is intuitively "easy" for a human may not match what is easy for the algorithm in the feature and hypothesis space employed for the given application.

To alleviate this deficiency, we introduce *self-paced learning*. In the context of human education, self-paced learning refers to a system where the curriculum is determined by the pupil's abilities rather than being fixed by a teacher. We build on this intuition for learning a latent SVM by designing an iterative approach that simultaneously selects easy samples and updates the parameters at each iteration by solving a biconvex optimization problem. The number of samples selected at any stage is determined by a weight that is gradually annealed such that subsequent iterations consider more and more samples. The algorithm converges when all samples have been considered and the objective function cannot be decreased further (up to a certain tolerance). While we focus on the latent SVM framework, our method can be generalized to other learning settings (such as maximum likelihood estimation using EM) for which the learning objective decomposes into a linear sum of terms for each sample. It is worth noting that, in the above algorithm, the characterization of what is "easy" applies not to individual samples, but to sets of samples; *a set of samples is easy if it admits a good fit in the model space*.

We empirically demonstrate that our self-paced learning approach outperforms the state of the art algorithm on five standard machine learning applications using publicly available datasets.

2.2 Related Work

Self-paced learning is related to curriculum learning in that both regimes suggest processing the samples in a meaningful order. Bengio *et al.* [5] noted that curriculum learning can be seen as a type of continuation method [1]. However, in their work, they circumvented the challenge of obtaining such an ordering by using datasets where there is a clear distinction between easy and hard samples (for example, classifying equilateral

triangles vs. squares is easier than classifying general triangles vs. general quadrilaterals). Such datasets are rarely available in real world applications, so it is not surprising that the experiments in [5] were mostly restricted to small toy examples.

Our approach also has a similar flavor to active learning, which chooses a sample to learn from at each iteration. Different active learning approaches use different criteria for selecting the sample. For example, Tong and Koller [64] suggest choosing a sample that is close to the margin, that is, a hard sample, corresponding to *anti-curriculum learning*. In a similar flavor, Cohn *et al.* [12] advocate the use of the most uncertain sample (where uncertainty is measured using the variance in the prediction by the current classifier). However, unlike our setting, in active learning the labels of all the samples are not known when the samples are chosen.

Another related learning regime is co-training, which works by alternately training classifiers such that the most confidently labeled samples from one classifier are used to train the other [11, 52]. Our approach differs from co-training in that in our setting the latent variables are simply used to assist in predicting the target labels, which are always observed, whereas co-training deals with a semi-supervised setting in which some labels are missing.

2.3 Self-Paced Learning for Latent SVM

We now describe our self-paced learning strategy for learning a latent SVM, that is, for optimizing the following difference-of-convex problem:

$$\begin{aligned} & \min_{\mathbf{w},\xi_i \geq 0} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i, \\ \text{s.t.} \quad & \max_{h_i \in \mathcal{H}} \mathbf{w}^\top \Psi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i) - \mathbf{w}^\top \Psi(\mathbf{x}_i, \hat{\mathbf{y}}_i, \hat{\mathbf{h}}_i) \geq \Delta(\mathbf{y}_i, \hat{\mathbf{y}}_i) - \xi_i, \\ & \forall \hat{\mathbf{y}}_i \in \mathcal{Y}, \forall \hat{h}_i \in \mathcal{H}, i = 1, \cdots, n. \end{aligned} \tag{2.1}$$

Recall that the main difficulty that one faces is the lack of a readily computable measure of the *easiness* of a sample. In the context of latent SVM, for a given weight $\mathbf{w}$, this easiness can be defined in two ways: (i) a sample is easy if we are confident about the value of a latent variable; or (ii) a sample is easy if it is easy to predict its true output (that is, it lies far away from the margin). The two definitions are somewhat related: if we are more certain about the latent variable, we may be more certain about the prediction. They are different in that certainty does not imply correctness, and the latent variables may not be directly relevant to what makes the output of an example

easy to predict. We therefore focus on the second definition: easy samples are ones whose correct output can be predicted easily.

In the above argument, we have assumed a given $\mathbf{w}$. However, in order to operationalize self-paced learning, we need a strategy for simultaneously selecting the easy samples and learning the parameter $\mathbf{w}$ at each iteration. To this end, we introduce the binary variable v_i, which indicates whether the i^{th} sample is easy or not. If the sample is not easy, then its slack variable is not added to the value of the objective function. In other words, the problem minimizes the upper bound on the risk of only the easy samples. We formulate this as the following mixed-integer problem:

$$\begin{aligned} & \min_{\mathbf{w},\mathbf{v},\xi_i \geq 0} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{n}\sum_{i=1}^{n} v_i\xi_i - \frac{1}{K}\sum_{i=1}^{n} v_i, \\ \text{s.t.} \quad & \mathbf{v} \in \{0,1\}^n, \\ & \mathbf{w}^\top \Psi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i^*) - \mathbf{w}^\top \Psi(\mathbf{x}_i, \hat{\mathbf{y}}_i, \hat{\mathbf{h}}_i) \geq \Delta(\mathbf{y}_i, \hat{\mathbf{y}}_i) - \xi_i, \\ & \forall \hat{\mathbf{y}}_i \in \mathcal{Y}, \hat{\mathbf{h}}_i \in \mathcal{H}, i = 1, \cdots, n, \end{aligned} \tag{2.2}$$

where $\mathbf{h}_i^*$ is the current imputed latent value of the i^{th} sample (that is, the output of annotation consistent inference). The weight K determines the number of samples to be considered: if K is small, the above problem prefers to include more samples; if K is large, it only considers a few samples for which it can find small slack values — that is, those that are "easy." Importantly, however, the samples are tied together in the objective through the parameter $\mathbf{w}$. Therefore, no sample is considered independently easy; rather, a *set* of samples is easy if a $\mathbf{w}$ can be fit to it such that the corresponding slack values are small. We iteratively decrease the value of K in order to estimate the parameters of a latent SVM via self-paced learning. In other words, we begin with only a few easy examples, and gradually introduce more and more samples until the entire training dataset is used.

Problem (2.2) can be relaxed such that v_i take values between 0 and 1 (instead of just binary values), resulting in a biconvex optimization problem. Recall that a biconvex problem is one where the variables $\mathbf{z}$ can be divided into two sets $\mathbf{z}_1$ and $\mathbf{z}_2$ such that for a fixed value of each set, the optimal value of the other set can be obtained by solving a convex optimization problem. In our case, the two sets of variables are $(\mathbf{w}, \xi_i)$ and $\mathbf{v}$. It is worth noting that the biconvex relaxation does not change the optimal solution of problem (2.2); we still obtain an integer $\mathbf{v}$. This is easy to prove: for any given $(\mathbf{w}, \xi_i)$, v_i should be greater than 0 if and only if $C\xi_i/n \leq 1/K$ (otherwise it would add a positive term to the objective function). If this is the case, then the objective function would be minimized at $v_i = 1$ (which would add the negative value $C\xi_i/n - 1/K$ to the objective function).

The advantage of the biconvex reformulation is that it allows us to tap into the vast literature that exists on solving such problems [4, 19]. For this work, we choose the alternate convex search (ACS) [4] procedure, which alternately optimizes over each set of variables, keeping the other set fixed. In our setting, we optimize the v_i's given the weights $\mathbf{w}$ and vice versa. The ACS procedure is provably convergent, although it may not converge to the globally optimal solution. But it is efficient and easy to implement, and worked well in practice in our experiments.

Algorithm 4 *The self-paced learning algorithm for parameter estimation of latent* SVM.

input $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \cdots, (\mathbf{x}_n, \mathbf{y}_n)\}$, $\mathbf{w}_0$, K_0, ϵ.

1: $t \leftarrow 0$, $K \leftarrow K_0$.
2: **repeat**
3: Update $\mathbf{h}_i^* = \arg\max_{h_i \in \mathcal{H}} \mathbf{w}_t^\top \Psi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i)$.
4: Update $\mathbf{w}_{t+1}$ by solving the biconvex reformulation of problem (2.2)
5: $t \leftarrow t + 1$, $K \leftarrow K/\mu$.
6: **until** Objective function of problem (2.1) cannot be decreased below tolerance ϵ.

Algorithm 4 outlines the overall self-paced learning algorithm for latent SVM. Similar to the original CCCP method, our algorithm consists of two steps: (i) annotation consistent inference, where we impute the value of the latent variables using the current estimate of the parameters; and (ii) parameter update. In contrast to CCCP, the parameter update involves solving a biconvex optimization problem instead of a convex problem. At each iteration, the weight K is reduced by a factor of $\mu > 1$ thereby introducing more and more (difficult) samples from one iteration to the next. The algorithm is said to converge when it considers all samples but is unable to decrease the value of the objective function for latent SVM below a certain tolerance ϵ. We note that our algorithm provides the same guarantees as CCCP, that is:

Property: Algorithm 4 converges to a local minimum or saddle point solution of problem (2.1).

This follows from the fact that the last iteration of Algorithm 4 is simply the original CCCP algorithm.

Our algorithm requires an initial parameter $\mathbf{w}_0$ (similar to CCCP). In our experiments, we obtained an estimate of $\mathbf{w}_0$ by initially setting $v_i = 1$ for all samples and running the original CCCP algorithm for a fixed, small number of iterations T_0. As our results indicate, this simple strategy was sufficient to obtain an accurate set of parameters using self-paced learning.

2.4 Experiments

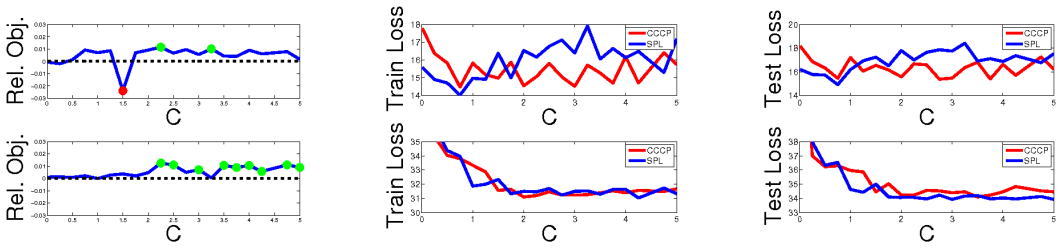
We now demonstrate the benefit of self-paced learning by comparing to CCCP on five standard machine learning applications. Unless otherwise stated, the initial weight K_0 is set such that the first iteration selects more than half the samples (as there are typically more easy samples than difficult ones), a factor of $\mu = 1.3$ is used to increase the number of samples over the iterations and the parameters are initialized using $T_0 = 2$ iterations of the original CCCP algorithm.

2.4.1 Noun Phrase Coreference

Problem Formulation. Given the occurrence of all the nouns in a document, the goal of noun phrase coreference is to provide a clustering of the nouns such that each cluster refers to a single object. This task was formulated within the SVM framework in [18] and later extended to include latent variables in [66]. Formally, the input vector $\mathbf{x}$ consists of the pairwise features $\mathbf{x}_{ij}$ suggested in [51] between all pairs of noun phrases i and j in the document. The output $\mathbf{y}$ represents a clustering of the nouns. A latent variable $\mathbf{h}$ specifies a forest over the nouns such that each tree in the forest consists of all the nouns of one cluster. Similar to [66], we employ two different loss functions: one corresponding to the pairwise score and one corresponding to the MITRE score. Annotation consistent inference involves finding the maximum spanning forest (which can be solved by, for example, Kruskal or Prims algorithm). In order to update the parameters during each iteration of CCCP and self-paced learning, we solve the corresponding SVM problem using the 1-slack reformulation based cutting plane algorithm. Similar to annotation consistent inference, loss augmented inference (required to compute the most violated constraint) can be formulated as a minimum spanning tree problem for both types of loss functions employed in our experiments.

Dataset. We use the publicly available MUC6 noun phrase coreference dataset, which consists of 60 documents. We use the same split of 30 training and 30 test documents as [66].

Results. We tested CCCP and our self-paced learning method on different values of C; the average training times over all 40 experiments (20 different values of C and two different loss functions) for the two methods were 1183 and 1080 seconds respectively. Fig. 2.1 compares the two methods in terms of the value of the objective function (which is the main focus of this work), the loss over the training data and the loss over the test



(a) (b) (c)

FIGURE 2.1: *Results for the noun phrase coreference experiment. Top:* MITRE *score. Bottom: Pairwise score. (a) The relative objective value computed as* $(obj_{cccp} - obj_{spl})/obj_{cccp}$*, where* obj_{cccp} *and* obj_{spl} *are the objective values of* CCCP *and self-paced learning respectively. A significant improvement (greater than tolerance* $C\epsilon$*) is indicated by a green circle, while a significant declination is shown by a red circle. The black dashed line corresponds to no difference in the objective values. (b) Loss over the training data. Red line corresponds to* CCCP*, blue corresponds to self-paced learning. Minimum* MITRE *loss:* 14.48 *and* 14.02 *for* CCCP *and self-paced learning respectively; Minimum pairwise loss:* 31.10 *and* 31.03*. (c) Loss over the test data. Minimum* MITRE *loss:* 15.38 *and* 14.91*; Minimum pairwise loss:* 34.10 *and* 33.93*.*

data. Note that self-paced learning significantly improves the objective function value in 11 of the 40 experiments (compared to only once when CCCP outperforms self-paced learning; see Fig. 2.1(a)). It also provides a better training and testing loss for both MITRE and pairwise scores when using the optimal value of C (see Fig. 2.1(b)-(c)).

2.4.2 Motif Finding

Problem Formulation. We consider the problem of binary classification of DNA sequences, which was cast as a latent SVM in [66]. Specifically, the input vector $\mathbf{x}$ consists of a DNA sequence of length l (where each element of the sequence is a nucleotide of type A, G, T or C) and the output space $\mathcal{Y} = \{+1, -1\}$. In our experiments, the classes correspond to two different types of genes: those that bind to a protein of interest with high affinity and those that do not. The positive sequences are assumed to contain particular patterns, called *motifs*, of length m that are believed to be useful for classification. However, the starting position of the motif within a gene sequence is often not known beforehand. Hence, this position is treated as the latent variable $\mathbf{h}$. For this problem, we use the joint feature vector suggested by [66]. Specifically,

$$\Psi(\mathbf{x}, \mathbf{y}, \mathbf{h}) = \begin{cases} (\Psi_M(\mathbf{x}_M(\mathbf{h})); \Psi_{BG}(\mathbf{x}_{BG}(\mathbf{h}))) & \text{if } \mathbf{y} = +1, \\ (\mathbf{0}; \Psi_{BG}(\mathbf{x})) & \text{if } \mathbf{y} = -1, \end{cases} \tag{2.3}$$

where the operator $(;)$ denotes vector concatenation and $\mathbf{0}$ is a vector of zeroes of the appropriate length. The term $\mathbf{x}_M(\mathbf{h})$ denotes the motif in the sequence $\mathbf{x}$ as specified by

$\mathbf{h}$ and $\mathbf{x}_{BG}(\mathbf{h})$ denotes the remaining *background* sequence. We extract different types of features for the motif (Ψ_M) and the background (Ψ_{BG}). The motif is represented using a binary vector of length $4m$. The $(4(i-1)+j)^{th}$ element of the vector denotes whether the i^{th} position of the motif contains the j^{th} type of nucleotide. The vector Ψ_{BG} is the feature count for the background model (a Markov model of order k). When $\mathbf{y} = -1$, the entire sequence is consider background. The loss function Δ is the standard 0-1 classification loss. Here, annotation consistent inference simply involves a linear search for the starting position of the motif. The SVM problem corresponding to the parameter update step at each iteration is solved using the 1-slack reformulation based cutting plane algorithm. The cutting planes are computed by solving the loss augmented inference using a brute-force search over all possible values of output and latent variables (which can be carried out efficiently in a time linear to the length of the sequence, similar to annotation consistent inference).

Dataset. We use the publicly available UniProbe dataset [6] that provides positive and negative DNA sequences for 177 proteins. For this work, we chose five proteins at random. The total number of sequences per protein is roughly $40,000$. For all the sequences, the motif length m is known (provided with the UniProbe dataset) and the background Markov model is assumed to be of order $k = 3$. In order to specify a classification task for a particular protein, we randomly split the sequences into roughly 50% for training and 50% for testing.

(a) Objective function value

CCCP	92.77 ± 0.99	$\mathbf{106.50 \pm 0.38}$	94.00 ± 0.53	116.63 ± 18.78	75.51 ± 1.97
SPL	$\mathbf{92.37 \pm 0.65}^*$	106.60 ± 0.30	$\mathbf{93.51 \pm 0.29}^*$	$\mathbf{107.18 \pm 1.48}$	$\mathbf{74.23 \pm 0.59}^*$

(b) Training error (%)

CCCP	27.10 ± 0.44	$\mathbf{32.03 \pm 0.31}$	26.90 ± 0.28	34.89 ± 8.53	20.09 ± 0.81
SPL	$\mathbf{26.94 \pm 0.26}^*$	32.04 ± 0.23	$\mathbf{26.81 \pm 0.19}$	$\mathbf{30.31 \pm 1.14}$	$\mathbf{19.52 \pm 0.34}^*$

(c) Test error (%)

CCCP	27.10 ± 0.36	$\mathbf{32.15 \pm 0.31}$	27.10 ± 0.37	35.42 ± 8.19	20.25 ± 0.65
SPL	$\mathbf{27.08 \pm 0.38}$	32.24 ± 0.25	$\mathbf{27.03 \pm 0.13}$	$\mathbf{30.84 \pm 1.38}$	$\mathbf{19.65 \pm 0.39}^*$

TABLE 2.1: *Mean and standard deviations for the motif finding experiments using the original* CCCP *algorithm (top row) and the proposed self-paced learning approach (bottom row). The better mean value is highlighted in bold font. Note that self-paced learning provides significant improvement for 3 out of 5 proteins in the objective value (which is the primary concern of this work). The improvement in objective value also translates to significant improvement in training (for 2 proteins) and test errors (for 1 protein).*

Results. We used five different folds for each protein, randomly initializing the motif positions for all training samples using four different seed values (fixed for both methods).

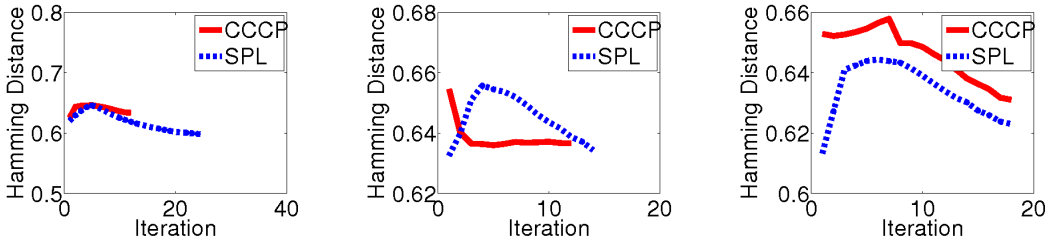


FIGURE 2.2: *Average Hamming distance between the motifs found in all selected samples at each iteration. Our approach starts with easy samples (small Hamming distance) and gradually introduces more difficult samples (large Hamming distance) until it starts to consider all samples of the training set. The figure shows results for three different protein-fold pairs. The average Hamming distance (over all proteins and folds) of the motifs obtained at convergence are* 0.6155 *for* CCCP *and* 0.6099 *for self-paced learning.*

We report results for each method using the best seed (chosen according to the value of the objective function). For all experiments we use $C = 150$ and $\epsilon = 0.001$ (the large size of the dataset made cross-validation highly time consuming). The average time over all 100 runs for CCCP and self-paced learning are 824 and 1287 seconds respectively. Although our approach is slower than CCCP for this application, as table 2.1 shows, it learns a better set of parameters. While improvements for most folds are small, for the fourth protein, CCCP gets stuck in a bad local minimum despite using multiple random initializations (this is indicated by the large mean and standard deviation values). This behavior is to be expected: in many cases, the objective function landscape is such that CCCP avoids local optima; but in some cases, CCCP gets stuck in poor local optima. Indeed, over all the 100 runs (5 proteins, 5 folds and 4 seed values) CCCP got stuck in a bad local minimum 18 times (where a bad local minimum is one that gave 50% test error) compared to 1 run where self-paced learning got stuck.

Fig. 2.2 shows the average Hamming distance between the motifs of the selected samples at each iteration of the self-paced learning algorithm. Note that initially the algorithm selects samples whose motifs have a low Hamming distance (which intuitively correspond to the easy samples for this application). It then gradually introduces more difficult samples (as indicated by the rise in the average Hamming distance). Finally, it considers all samples and attempts to find the most discriminative motif across the entire dataset. Note that the motifs found over the entire dataset using self-paced learning provide a smaller average Hamming distance than those found using the original CCCP algorithm, indicating a greater coherence for the resulting output.

2.4.3 Handwritten Digit Recognition

Problem Formulation. Handwritten digit recognition is a special case of multi-label classification, and hence can be formulated within the SVM framework. Specifically,

given an input vector $\mathbf{x}$, which consists of m grayscale values that represent an image of a handwritten digit, our aim is to predict the digit. In other words, $\mathcal{Y} = \{0, 1, \cdots, 9\}$. It is well-known that the accuracy of digit recognition can be greatly improved by explicitly modeling the deformations present in each image, for example see [59]. For simplicity, we assume that the deformations are restricted to an arbitrary rotation of the image, where the angle of rotation is not known beforehand. This angle (which takes a value from a finite discrete set) is modeled as the latent variable $\mathbf{h}$. We specify the joint feature vector as $\Psi(\mathbf{x}, \mathbf{y}, \mathbf{h}) = (\mathbf{0}_{\mathbf{y}(m+1)}; \theta_{\mathbf{h}}(\mathbf{x})\ 1; \mathbf{0}_{(9-\mathbf{y})(m+1)})$, where $\theta_{\mathbf{h}}(\mathbf{x})$ rotates the image represented by $\mathbf{x}$ by the angle corresponding to $\mathbf{h}$ and converts it to a vector. In words, the joint feature vector is the rotated image of the digit which is padded in the front and back with the appropriate number of zeroes. Similar to the motif finding experiment, we use the standard 0-1 classification loss. Annotation consistent inference simply involves a search over a discrete set of angles. The SVM problem for parameter update at each iteration is solved using the 1-slack reformulation based cutting plane algorithm, where loss augmented inference is performed by searching over all possible values of the output and latent variables.

Dataset. We use the standard MNIST dataset [41], which represents each handwritten digit as a vector of length 784 (that is, an image of size 28×28). For efficiency, we use PCA to reduce the dimensionality of each sample to 10. We perform binary classification on four difficult digit pairs (1 vs 7, 2 vs 7, 3 vs 8, and 8 vs 9), as in [69]. The training standard dataset size for each digit ranges from $5,851$ to $6,742$, and the test sets range from 974 to $1,135$ digits. The rotation modeled by the latent variable can take one of 11 discrete values, evenly spaced between -60 and 60 degrees.

Results. For each digit pair, we use C values ranging from 25 to 300, set $\epsilon = 0.001$, and set $K = \frac{10^4}{C}$. Modeling rotation as a latent variable significantly improves classification performance, allowing the images to be better aligned with each other. Across all experiments for both learning methods, using latent variables achieves better test error; the improvement over using no latent variables is 12%, 8%, 11%, and 22%, respectively, for the four digit pairs. CCCP learning took an average of 18 minutes across all runs, while self-paced learning took an average of 53 minutes.

Figure 2.3 compares the training and test errors and objective values between CCCP and self-paced learning. Those runs for which the objective value difference between self-paced learning and CCCP is greater than the convergence criteria range ($C\epsilon$) are highlighted in blue. Self-paced learning achieves significantly better values in 15 runs, and is worse in 4 runs, demonstrating that it helps find better solutions to the optimization problems. Though the training and test errors do not necessarily correlate to

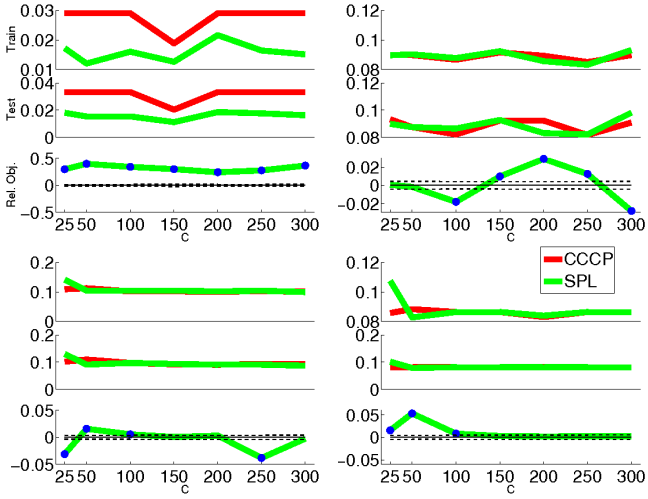


FIGURE 2.3: *Results for four digit pairs from the* MNIST *dataset. Relative primal objective is the difference between the self-paced learning and* CCCP *objective divided by the* CCCP *objective. Positive values indicate superior results for self-paced learning. The dotted black lines delineate where the difference is greater than the convergence criteria range ($C\epsilon$), and differences outside of this range are highlighted in blue.*

objective values, the best test error across C values is better for self-paced learning for one of the digit pairs (1 vs 7), and is the same for the other three.

2.4.4 Object Localization

Problem Formulation. Given a set of images along with labels that indicate the presence of a particular object category in the image (for example, a mammal), our goal is to learn discriminative object models for all object categories (that is, models that can distinguish between one object, say bison, from another, say elephant). In practice, although it is easy to mine such images from free photo-sharing websites such as Flickr, it is burdensome to obtain ground truth annotations of the exact location of the object in each image. To avoid requiring these human annotations, we model the location of objects as latent variables. Formally, for a given image $\mathbf{x}$, object class $\mathbf{y}$ and location $\mathbf{h}$, the score is modeled as $\mathbf{w}^\top \Psi(\mathbf{x}, \mathbf{y}, \mathbf{h}) = \mathbf{w}_\mathbf{y}^\top \Psi_\mathbf{h}(\mathbf{x})$, where $\mathbf{w}_\mathbf{y}$ are the parameters that corresponds to the class $\mathbf{y}$ and $\Psi_\mathbf{h}(\cdot)$ is the HOG [13, 17] feature extracted from the image at position $\mathbf{h}$ (the size of the object is assumed to be the same for all images — a reasonable assumption for our datasets). The loss function $\Delta(\mathbf{y}, \hat{\mathbf{y}})$ is again the standard 0-1 classification loss. For the above problem, annotation consistent inference involves a simple search over possible locations in a given image. In each iteration, the parameters are updated using the 1-slack reformulation based cutting plane algorithm. Loss augmented inference is performed by searching over all possible values of the output and the latent variables.

Dataset. We use images of 6 different mammals (approximately 45 images per mammal) that have been previously employed for object localization [24]. We split images of each category into approximately 90% for training and 10% for testing.

Results. We split the images into five different folds to compare our method with the state of the art CCCP algorithm. For each fold, we randomly initialized the initial location of the object in each image (the initialization was fixed for the two methods). We used a value of $C = 10$ and $\epsilon = 0.001$. The average training time over all folds were 362 seconds and 482 seconds for CCCP and self-paced learning respectively. Table 2.2 shows the mean and standard deviation of three terms: the objective value, the training loss and the testing loss. Self-paced learning provided a significantly lower (more than tolerance) objective value than CCCP for all folds. The better objective function value resulted in a substantial improvement in the training (for 4 folds) and testing loss (an improvement of approximately 4% for achieved for 2 folds). In these experiments, CCCP never outperformed self-paced learning for any of the three measures of performance.

Objective	Train Loss (%)	Test Loss (%)
4.70 ± 0.11	0.33 ± 0.18	16.92 ± 5.16

Objective	Train Loss (%)	Test Loss (%)
$\mathbf{4.53 \pm 0.15}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{15.38 \pm 3.85}$

TABLE 2.2: *Results for the object localization experiment. Left:* CCCP. *Right: Self-paced learning. Note that self-paced learning provides better results for all measures of performance.*

Fig. 2.4 shows the imputed bounding boxes for two images during various iterations of the two algorithms. The proposed self-paced learning algorithm does not use the hard image during the initial iterations (as indicated by the red bounding box). In contrast, CCCP considers all images at each iteration. Note that self-paced learning provides a more accurate bounding box for the hard image at convergence, thereby illustrating the importance of learning in a meaningful order. In our experience, this was a typical behavior of the two algorithms.

2.4.5 Semantic Segmentation

Problem Formulation. Given a set of partially annotated images, our goal is to learn a semantic segmentation model, that is, a model that labels each pixel of an image using its semantic class (such as road, car or person). Specifically, we are interested in learning the parameters of the region-based semantic segmentation model [21]. Briefly, the model groups the pixels of an image into coherent, contiguous regions and assigns each label to

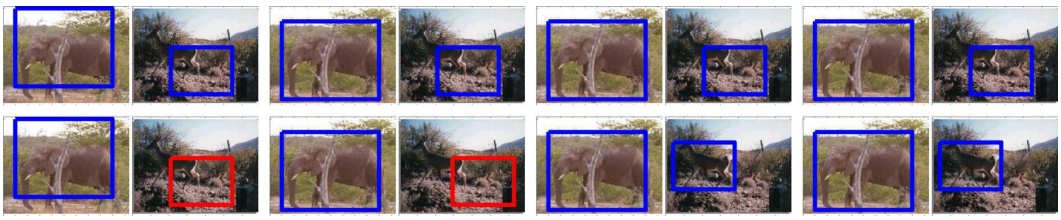


FIGURE 2.4: *The top row shows the imputed bounding boxes of an easy and a hard image using the* CCCP *algorithm over different iterations. Note that for the hard image (one with deer) the bounding box obtained at convergence does not localize the object accurately. In contrast, the self-paced learning approach (bottom row) does not use the hard image during initial iterations (indicated by the red color of the bounding box). In subsequent iterations, it is able to impute an accurate bounding box for both the easy and the hard image.*

a semantic class. The joint feature vector of the image and the segmentation depends on the features extracted from each region, and each pair of neighboring regions (where two regions are considered neighboring if they share at least one boundary pixel). We refer the interested reader to [21] for details regarding the region-based model. In our work, we are interested in three types of partial annotations: (i) where pixels can be labeled using a generic background or foreground class, instead of a specific semantic class; (ii) where the bounding box of an object is specified, without any pixelwise labeling; and (iii) where the presence of absence of an object class is specified, without any bounding box information or pixelwise labeling. The loss function is the standard overlap score [16] between two segmentations. Note that this implies that the loss function depends on the true (unknown) value of the hidden variables. In order to handle this, we use an iterative strategy, where the segmentations of all training images are estimated for the current set of parameters, and then the parameters are updated by solving a latent SVM problem that treats the estimated segmentations as ground-truth. The entire approach is repeated until convergence. We refer the interested reader to [38] for details. Annotation consistent inference and loss augmented inference (required for the parameter update step using stochastic subgradient descent) are performed by suitably modifying the linear programming relaxation based approach reported in our earlier work [37]. Both CCCP and self-paced learning are initialized using the parameters learned by a method specifically designed to estimate the parameters of the region-based model, known as closed loop learning (CLL) [21].

Dataset. We use four datasets: (i) the VOC2009 segmentation dataset [16], which provides us with annotations consisting of 20 specific foreground classes and a generic background for 2249 images; (ii) SBD [21], which provides us with annotations consisting of 7 specific background classes and a generic foreground for 715 images; (iii) a subset of the VOC2010 detection dataset [16], which provides us with bounding box annotations

for 1564 images; and (iv) a subset of the ImageNet dataset [15], which provides us with image-level labels for 1000 images. We use 225 VOC2009 and 53 SBD images for validation, and 750 VOC2009 and 90 SBD images for testing.

	avg	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	dtable	dog	horse	mbike	person	plant	sheep	sofa	train	tv	bg
CLL	**24.7**	32.1	16.3	09.5	05.9	25.0	43.7	40.2	**17.1**	04.6	26.6	16.8	10.3	32.7	31.3	35.7	**08.9**	27.7	11.1	29.7	23.6	69.6
LSVM	**26.9**	41.4	19.0	09.6	06.9	28.1	45.4	**41.4**	16.4	**05.2**	26.0	**18.7**	10.3	34.4	33.5	37.3	07.6	32.7	10.7	34.1	31.3	75.2
BOX	**28.3**	43.0	19.9	**10.9**	**08.0**	30.8	**47.4**	39.0	16.5	03.8	26.6	16.9	11.4	36.9	36.2	41.2	07.7	**37.4**	11.5	**35.8**	33.8	79.4
LABELS	**28.8**	**44.5**	**20.9**	10.7	07.8	**31.5**	45.6	40.0	15.7	03.5	**28.1**	16.0	**12.3**	**37.7**	**40.4**	**42.2**	08.3	35.6	**12.9**	**35.8**	34.6	**80.0**
CCCP	**24.7**	39.4	15.6	06.5	06.5	27.7	41.0	40.7	16.3	5.5	29.2	12.8	08.5	31.4	39.6	32.7	08.2	30.1	12.3	27.9	18.5	67.9

FIGURE 2.5: *Accuracies for the* VOC2009 *test set. First row shows the results obtained using* CLL *[21] with a combination of* VOC2009 *and* SBD *training images. The second row shows the results obtained using self-paced learning for latent* SVM *with the same training set of the training images. The third row shows the results obtained using an additional 1564 bounding box annotations. The fourth row shows the results obtained by further augmenting the training dataset with 1000 image-level annotations. The best accuracy for each class is underlined. The fifth row shows the results obtained when the latent* SVM *is learned using* CCCP *on the entire dataset.*

	avg	sky	tree	road	grass	water	bldg	mntn	fg
CLL	**53.1**	77.7	48.4	70.1	73.5	55.6	62.5	00.0	36.0
LSVM	**54.3**	79.1	48.2	75.5	76.0	55.1	61.4	00.0	39.1
BOX	**54.8**	78.3	48.6	75.4	76.0	59.9	60.8	00.0	39.6
LABELS	**55.3**	78.1	49.5	75.5	76.1	60.1	62.0	00.0	41.3
CCCP	**53.8**	75.4	48.7	70.0	74.0	59.9	62.5	00.0	39.9

FIGURE 2.6: *Accuracies for the* SBD *test set. See caption of Fig. 2.5 for an explanation of the various methods.*

Results. Figures 2.5 and 2.6 (rows 1 and 2) show the accuracies obtained for SBD and VOC2009 test images respectively, when the training images are restricted to the first two datasets (which provide generic class annotations). The accuracies are measured using the overlap score. Note that the latent SVM is trained using the self-paced learning algorithm. While both CLL and latent SVM produce specific-class segmentations of all the test images, we use generic classes while measuring the performance due to the lack of specific-class ground-truth annotations. Note that latent SVM provides better accuracies for nearly all the object classes in VOC2009 (17 of 21 classes). For SBD, latent SVM provides a significant boost in performance for 'sky', 'road', 'grass' and 'foreground'. With the exception of 'building', the accuracies for other classes is comparable. The reason for poor performance in the 'mountain' class is that several 'mountain' pixels

are labeled as 'tree' in SBD (which confuses both the learning algorithms). Our results convincingly demonstrate the advantage of using latent SVM.

Figures 2.5 and 2.6 (row 3) show the accuracies obtained by using the additional image provide by the third dataset (that is, the bounding box annotations), together with the self-paced learning algorithm. Once again, we observe an improvement in the accuracies for nearly all the VOC2009 classes (18 of 21 classes) compared to the latent SVM trained using only generic class annotations. For SBD, we obtain a significant boost for 'tree', 'water' and 'foreground', while the accuracies of 'road', 'grass' and 'mountain' remain (almost) unchanged.

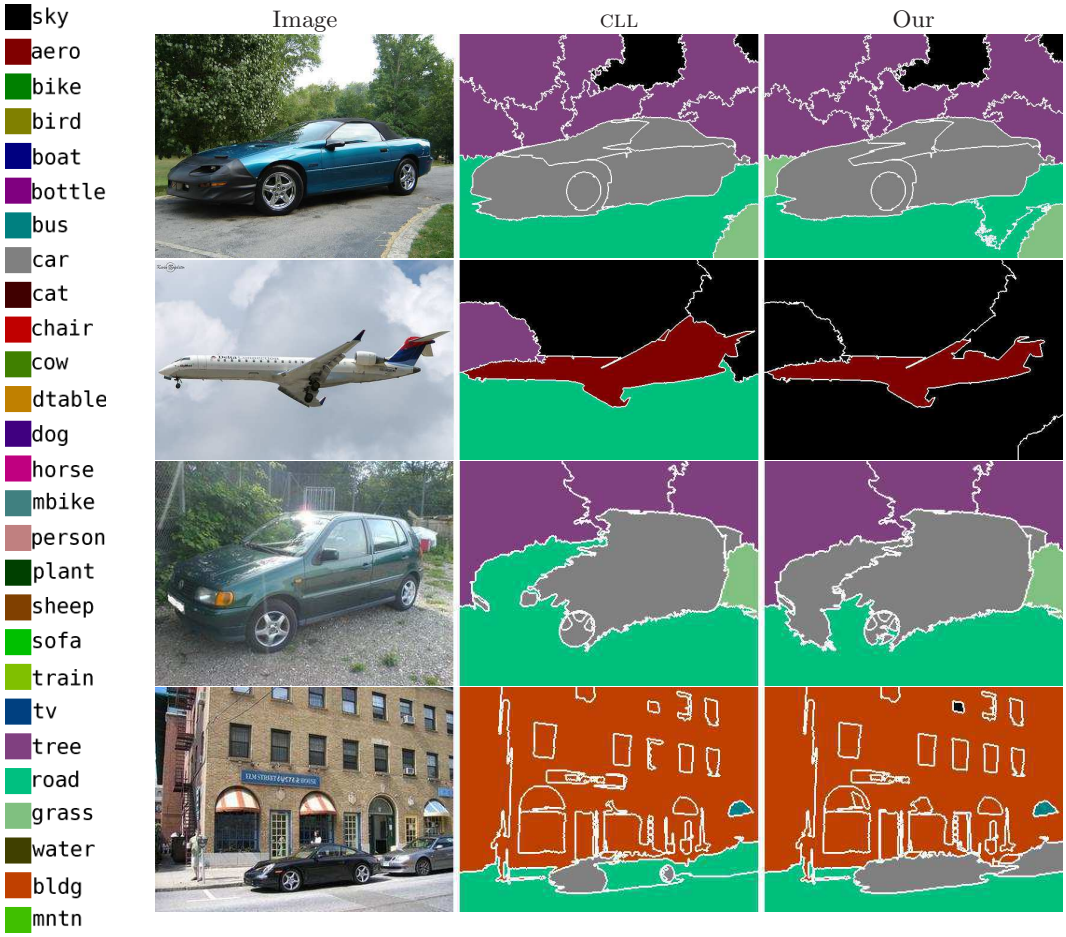


FIGURE 2.7: *The first two rows show the results obtained for images from the* VOC2009 *test set. Note that, unlike our approach that learns the parameters using latent* SVM *on a large dataset,* CLL *mislabels background pixels into the wrong specific classes. The last two rows show images from the* SBD *test set. While our approach is able to identify most of the foreground pixels correctly,* CLL *mislabels them as background.*

Figures 2.5 and 2.6 (row 4) show the accuracies obtained by using the additional images provide by the fourth dataset (that is, image-level annotations), together with the self-paced learning algorithm. For the VOC2009 segmentation test set, the final model learned from all the training images provides the best accuracy for 12 of the 21 classes.

Compared to the model learned using generic class labels and bounding boxes, we obtain a significant improvement for 13 classes by incorporating image-level annotations. Of the remaining 8 classes, the accuracies are comparable for 'bird', 'boat', 'chair' and 'train'. For the SBD test set, the model trained using all the data obtains the highest accuracy for 5 of the 8 classes. Figure 2.7 shows examples of the specific-class segmentation obtained using our method. Note that the parameters learned using our approach on a large dataset are able to correctly identify the specific classes of pixels.

Figures 2.5 and 2.6 (row 5) show the accuracies obtained using CCCP for VOC2009 and SBD respectively. Note that CCCP does not provide any improvement over CLL, which is trained using only the strongly supervised images, in terms the average overlap score for VOC2009 . While the overlap score improves for the SBD dataset, the improvement is significantly better when using self-paced learning (row 4). These results convincingly demonstrate that, unlike CCCP, self-paced learning is able to handle the noise inherent in the problem of learning with diverse data (for example, see Figure 2.8).

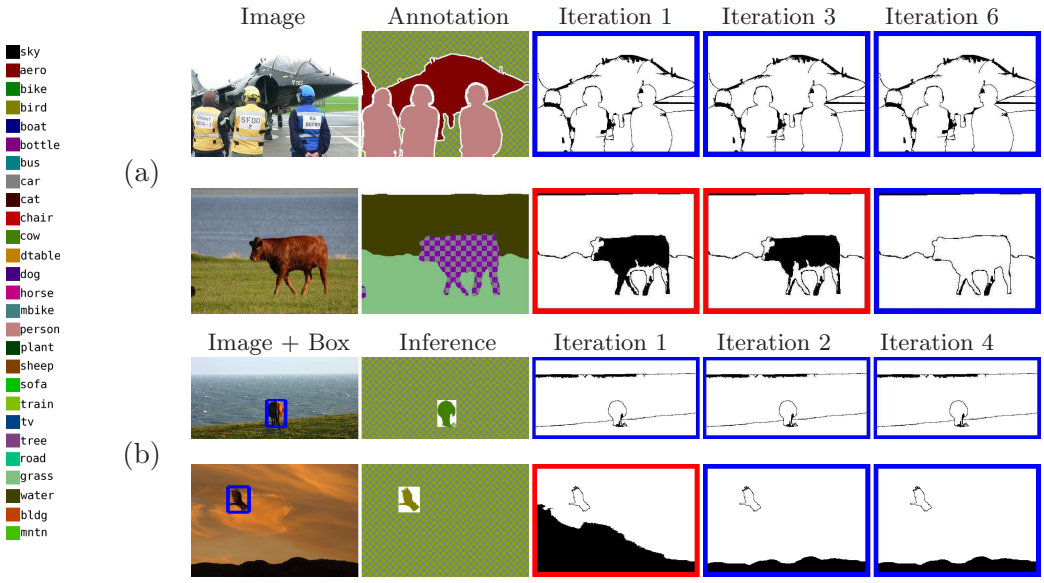


FIGURE 2.8: *Labelings obtained using annotation-consistent inference during different iterations of self-paced learning.* **(a)** *Images annotated with generic classes. Column 2 shows the annotation (where the checkered patterns indicate generic classes). In columns* $3-5$*, pixels labeled using the correct specific-class by annotation-consistent inference are shown in white, while pixel labeled using the wrong specific-class are shown in black (we labeled these images with specific-class annotations only for the purpose of illustration; these annotations were not used during training). A blue surrounding box on the labeling implies that the example was selected as easy by self-paced learning, while a red surrounding box indicates that is wasn't selected during the specified iteration. Note that self-paced learning discards the image where the cow (row 2) is incorrectly labeled.* **(b)** *Images annotated using bounding boxes. Column 2 shows the annotation obtained using bounding box inference. Note that the objects have been accurately segmented. Furthermore, self-paced learning discards the image where the sky (row 2) is incorrectly labeled.*

2.5 Discussion

We proposed an automatic self-paced learning regime in the context of parameter estimation for latent SVM. Our method works by iteratively solving a biconvex optimization problem that simultaneously selects easy samples and updates the parameters. Using four standard datasets from disparate domains (natural language processing, computational biology and computer vision) we showed that our method outperforms the state of the art approach.

In the current work, we solve the biconvex optimization problem using an alternate convex search strategy, which only provides us with a local minimum solution. Although our results indicate that such a strategy is more accurate than CCCP, it is worth noting that the biconvex problem can also be solved using a global optimization procedure, for example the one described in [19]. This is a valuable direction for future work.

We note that while we have defined our self-paced learning problem in the context of latent SVM, our approach generalizes to other learning settings in which the objective decomposes into a linear sum of terms for each sample. If this criterion is met, the variables v_i can multiply each such term, the resulting objective will be convex in $\mathbf{v}$ given the remaining parameters, and the convex relaxation will result in integer solutions for $\mathbf{v}$. For example, we can easily apply this to (hard or soft) EM for maximum likelihood estimation, using a simple change to the M-step. We believe that our approach provides an elegant mathematical formulation for self-paced learning, which allows it to be applied easily and effectively to a broad range of non-convex learning tasks.

Chapter 3

Max-Margin Min-Entropy Models

3.1 Introduction

As we have seen in the previous chapter, latent variable models (LVM) such as latent SVM provide an elegant formulation for weakly supervised learning. They are widely applicable to many areas of machine learning including computer vision, natural language processing and computational biology. Formally, an LVM consists of three types of variables: (i) the observed variables, or input, whose values are known during both training and testing; (ii) the unobserved variables, or output, whose values are known only during training; and (iii) the latent variables, whose values are unknown during both training and testing. An LVM can be interpreted as the distribution of the output and the latent variables conditioned on, or jointly with, the input. Modeling the conditional distribution results in discriminative LVMs, while modeling the joint distribution results in generative LVMs. Given an input, the output is typically predicted by either (i) computing the most probable assignment of the output and the latent variables according to the aforementioned distribution [17, 60, 66]; or (ii) computing the most probable assignment of the output by marginalizing out the latent variables [14]. Both these prediction criteria ignore an important factor: how certain are we about the values of the latent variables for the predicted output? Since the underlying assumption of LVM is that the latent variables provide useful cues for predicting the output, we argue that minimizing the confusion in their values will help improve the accuracy of the model. More importantly, in many cases, we would like to obtain an estimate of the latent variables with high certainty. For example, using an LVM for a 'car' we would like not only to classify an image as containing a car or not, but also predict the location of the car if present.

We propose a novel family of discriminative LVMs, called max-margin min-entropy (M3E) models, that predicts the output by minimizing the Rényi entropy [56] of the corresponding *generalized distribution* (that is, the unnormalized part of the distribution that models the output under consideration). This amounts to minimizing a score that consists of two terms: (i) the negative log-likelihood of the output obtained by marginalizing the latent variables; and (ii) the Rényi entropy of the normalized conditional probability of the latent variables given the input and the output. In other words, the predicted output not only has a high probability, but also minimizes the uncertainty in the values of the latent variables.

Given a training dataset, the parameters of an M3E model are learned by maximizing the margin between the Rényi entropies of the generalized distributions corresponding to the ground-truth output and all other outputs. Intuitively, this ensures that the output of a training sample is correctly predicted by the model. We show that the corresponding optimization problem amounts to minimizing an upper bound on a user-defined loss over the training dataset. Furthermore, we show that the M3E family includes, as a special case, the latent support vector machine (or latent SVM for short) formulation [17, 60, 66].

In order to use the M3E family of models in practice, we propose an efficient trust region style algorithm for learning their parameters. Our approach relies only on a solver for structured support vector machine (or structured SVM for short) problems [63, 65], of which there are several reported in the literature [32, 58]. Our algorithm is directly applicable for problems where the space of latent variables is tractable (for example, small number of latent variables with a small number of putative values, or when the underlying graphical model is a tree). When faced with an intractable latent space, similar to other LVMs, we can resort to approximate inference schemes in order to obtain an estimate of the Rényi entropy.

We demonstrate the efficacy of M3E models on two standard machine learning applications using publicly available datasets: discriminative motif finding and image classification.

3.2 Related Work

The most commonly used method for learning the parameters of an LVM is the expectation-maximization (EM) algorithm [14, 62], or its many variants [20, 50], including discriminative EM [57]. The EM algorithm attempts to maximize the expected likelihood of the training data, where the expectation is taken over a distribution of the latent variables. Once the parameters are learned, the output of the test sample is typically predicted by

marginalizing out the latent variables (corresponding to the objective optimized by *soft* EM) or by maximizing the joint probability of the output and the latent variables (corresponding to the objective optimized by *hard* EM, which approximates the expectation by a pointwise estimate). As argued earlier, predicting the output in this manner does not take into account any measure of uncertainty in the values of the latent variables.

Recently, Smola *et al.* [60], Felzenszwalb *et al.* [17] and Yu and Joachims [66] independently proposed the latent SVM framework, that extends the structured SVM [63, 65] to handle latent variables. The parameters of a latent SVM are learned by minimizing an upper bound on a user-defined loss, a process that is closely related to hard EM. The latent SVM formulation has steadily gained popularity, not least because its parameter learning problem only requires a *maximum a posteriori* inference algorithm—a well-studied problem with several accurate approximate (and in some cases, exact) methods. In section 3.5, we will show that latent SVM can be viewed as a special case of the M3E family.

Finally, we note that there have been several works reported in the literature based on the principle of maximum entropy [29], including classification [27] and feature selection [31]. Maximum entropy classification has also been extended to handle latent variables [30]. However, unlike M3E, maximum entropy methods measure the entropy of the input and the output, and not the entropy of the latent variables (which are, in fact, marginalized out).

3.3 Preliminaries

Notation. As usual, we denote the input by $\mathbf{x} \in \mathcal{X}$, the output by $\mathbf{y} \in \mathcal{Y}$ and the latent variables by $\mathbf{h} \in \mathcal{H}$. As mentioned earlier, the value of input $\mathbf{x}$ is known during both training and testing, the value of the output $\mathbf{y}$ is only known during training and the value of the latent variables $\mathbf{h}$ is not known during either training or testing. We denote the parameters of our model by $\mathbf{w}$. For simplicity, we assume a discrete setting. In this case, the conditional probability of the output and the latent variables, given the input, can be viewed as a set $\mathcal{P}_{\mathbf{x}} = \{\Pr(\mathbf{y}, \mathbf{h}|\mathbf{x}; \mathbf{w}), \forall (\mathbf{y}, \mathbf{h}) \in \mathcal{Y} \times \mathcal{H}\}$, whose elements are non-negative and sum to one. Furthermore, we denote the conditional probability of the latent variables, given the input and a particular output $\mathbf{y}$, as the set $\mathcal{P}_{\mathbf{x}}^{\mathbf{y}} = \{\Pr(\mathbf{h}|\mathbf{y}, \mathbf{x}; \mathbf{w}), \forall \mathbf{h} \in \mathcal{H}\}$. A generalized distribution refers to a subset of the distribution $\mathcal{P}_{\mathbf{x}}$ [56]. Of particular interest to us are those subsets that correspond to a particular output $\mathbf{y}$, that is, $\mathcal{Q}_{\mathbf{x}}^{\mathbf{y}} = \{\Pr(\mathbf{y}, \mathbf{h}|\mathbf{x}; \mathbf{w}), \forall \mathbf{h} \in \mathcal{H}\}$, where we use $\mathcal{Q}$ instead of $\mathcal{P}$ to indicate the fact that generalized distributions need not sum to one.

Rényi Entropy. Throughout the chapter, we will employ the concept of Rényi entropy [56], which is a family of measures for the uncertainty in a distribution. The entire family of Rényi entropy measures is parameterized by a single positive scalar α. Formally, the Rényi entropy of a generalized distribution $\mathcal{Q}_{\mathbf{x}}^{\mathbf{y}}$ is given by

$$H_\alpha(\mathcal{Q}_{\mathbf{x}}^{\mathbf{y}};\mathbf{w}) = \frac{1}{1-\alpha}\log\left(\frac{\sum_{\mathbf{h}}\Pr(\mathbf{y},\mathbf{h}|\mathbf{x};\mathbf{w})^\alpha}{\sum_{\mathbf{h}}\Pr(\mathbf{y},\mathbf{h}|\mathbf{x};\mathbf{w})}\right). \tag{3.1}$$

Some interesting special cases of Rényi entropy include the well-known Shannon entropy (corresponding to taking the limit $\alpha \to 1$) and the minimum entropy (corresponding to taking the limit $\alpha \to \infty$),

$$\begin{aligned} H_1(\mathcal{Q}_{\mathbf{x}}^{\mathbf{y}};\mathbf{w}) &= \frac{-\sum_{\mathbf{h}}\Pr(\mathbf{y},\mathbf{h}|\mathbf{x};\mathbf{w})\log\Pr(\mathbf{y},\mathbf{h}|\mathbf{x};\mathbf{w})}{\sum_{\mathbf{h}}\Pr(\mathbf{y},\mathbf{h}|\mathbf{x};\mathbf{w})}, \\ H_\infty(\mathcal{Q}_{\mathbf{x}}^{\mathbf{y}};\mathbf{w}) &= -\log\max_{\mathbf{h}}\Pr(\mathbf{y},\mathbf{h}|\mathbf{x};\mathbf{w}). \end{aligned} \tag{3.2}$$

The Rényi entropy family is *complete* in that no other function can satisfy all the postulates of an uncertainty measure. We refer the reader to [56] for details.

3.4 M3E Models

We wish to develop an LVM such that, given an input $\mathbf{x}$, the best output $\mathbf{y}^*$ is predicted by optimizing an appropriate measure such that (i) $\mathbf{y}^*$ has a high probability; and (ii) $\mathbf{y}^*$ minimizes the confusion in the values of the latent variables. Using this LVM will not only allow us to accurately predict the output (for example, whether the image contains a 'car' or not) but also the latent variables (the location of the car in the image) with high certainty, which is important in many applications. The key observation of this work is that the readily available Rényi entropy of generalized distributions is just such a measure. Specifically, it can be verified that for any output $\mathbf{y}$, the following holds true:

$$H_\alpha(\mathcal{Q}_{\mathbf{x}}^{\mathbf{y}};\mathbf{w}) = -\log\Pr(\mathbf{y}|\mathbf{x};\mathbf{w}) + H_\alpha(\mathcal{P}_{\mathbf{x}}^{\mathbf{y}};\mathbf{w}). \tag{3.3}$$

In other words, the Rényi entropy of the generalized distribution of an output $\mathbf{y}$ is the sum of the negative log-likelihood of $\mathbf{y}$ (corresponding to point (i)) and the Rényi entropy of the normalized conditional probability of the latent variables given $\mathbf{y}$ (corresponding to point (ii)). We now provide a formal description of the family of LVMs, which we refer to as the max-margin min-entropy (M3E) models, that uses Rényi entropy for prediction.

Given an input $\mathbf{x} \in \mathcal{X}$, an M3E model defines a conditional distribution over all possible outputs $\mathbf{y} \in \mathcal{Y}$ and latent variables $\mathbf{h} \in \mathcal{H}$. For simplicity of the description, and computational tractability of the corresponding learning and inference algorithms, we

focus on log-linear models. Specifically, for a given set of parameters $\mathbf{w}$, the distribution is given by

$$\Pr(\mathbf{y}, \mathbf{h}|\mathbf{x}; \mathbf{w}) = \frac{1}{Z(\mathbf{x}; \mathbf{w})} \exp\left(\mathbf{w}^\top \Psi(\mathbf{x}, \mathbf{y}, \mathbf{h})\right), \tag{3.4}$$

where $\Psi(\mathbf{x}, \mathbf{y}, \mathbf{h})$ refers to the joint feature vector of the input, output and latent variables, and $Z(\mathbf{x}; \mathbf{w})$ is the partition function that normalizes the distribution to sum to one. Given an input $\mathbf{x}$, the corresponding output is predicted by minimizing the Rényi entropy of the corresponding generalized distribution, that is,

$$\mathbf{y}^* = \operatorname*{argmin}_{\mathbf{y}} H_\alpha(\mathcal{Q}^{\mathbf{y}}_{\mathbf{x}}; \mathbf{w}). \tag{3.5}$$

3.5 Learning M3E Models

Given a training dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \cdots, n\}$, we would like to learn the parameters $\mathbf{w}$ of an M3E model such that it predicts the correct output of a given instance. To this end, we propose a parameter estimation approach that tries to introduce a margin between the Rényi entropy of the ground-truth output and all other outputs. The desired margin is specified by a user-defined loss function $\Delta(\mathbf{y}_i, \overline{\mathbf{y}})$ that measures the *difference* between the two outputs $\overline{\mathbf{y}}$ and $\mathbf{y}_i$. Similar to previous max-margin formulations, we assume that $\Delta(\mathbf{y}, \mathbf{y}) = 0$ for all $\mathbf{y} \in \mathcal{Y}$.

Formally, our parameter estimation approach is specified by the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w}, \xi \geq 0} \quad & \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i \\ & H_\alpha(\mathcal{Q}^{\overline{\mathbf{y}}}_i; \mathbf{w}) - H_\alpha(\mathcal{Q}^{\mathbf{y}_i}_i; \mathbf{w}) \geq \Delta(\mathbf{y}_i, \overline{\mathbf{y}}) - \xi_i, \\ & \forall \overline{\mathbf{y}} \neq \mathbf{y}_i, \forall (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}, \end{aligned} \tag{3.6}$$

where we use $\mathcal{Q}^{\mathbf{y}}_i$ instead of $\mathcal{Q}^{\mathbf{y}}_{\mathbf{x}_i}$ for conciseness. The objective function of the above problem consists of two terms. The first term corresponds to regularizing the parameters by minimizing its ℓ_2 norm. The second term encourages the Rényi entropy for the ground-truth output to be smaller than the Rényi entropy of all other outputs by the desired margin. As can be seen from the constraints of the above problem, the greater the difference between $\mathbf{y}_i$ and $\overline{\mathbf{y}}$ (as specified by the loss function $\Delta(\cdot, \cdot)$), the more the desired margin. The fixed term $C > 0$ is the relative weight of these two terms.

Problem (3.6) can also be seen as minimizing a regularized upper bound on the user-defined loss $\Delta(\mathbf{y}_i, \mathbf{y}_i(\mathbf{w}))$ over the training dataset, where $\mathbf{y}_i(\mathbf{w})$ denotes the predicted

output of the i^{th} training sample using the parameters $\mathbf{w}$. More precisely, the following proposition holds true.

Proposition 1. $\Delta(\mathbf{y}_i, \mathbf{y}_i(\mathbf{w})) \leq \xi_i$, where ξ_i are as defined in problem (3.6).

Proof. Since $\mathbf{y}_i(\mathbf{w})$ is the predicted output using the M3E model, $\mathbf{y}_i(\mathbf{w}) = \text{argmin}_{\hat{\mathbf{y}}} \, H_\alpha(\mathcal{Q}_i^{\hat{\mathbf{y}}}; \mathbf{w})$. Using this observation, we obtain the following:

$$\begin{aligned} & \Delta(\mathbf{y}_i, \mathbf{y}_i(\mathbf{w})) - H_\alpha(\mathcal{Q}_i^{\mathbf{y}_i}; \mathbf{w}) \\ \leq \; & \Delta(\mathbf{y}_i, \mathbf{y}_i(\mathbf{w})) - H_\alpha(\mathcal{Q}_i^{\mathbf{y}_i(\mathbf{w})}; \mathbf{w}) \\ \leq \; & \max_{\hat{\mathbf{y}}} \Big(\Delta(\mathbf{y}_i, \hat{\mathbf{y}}) - H_\alpha(\mathcal{Q}_i^{\hat{\mathbf{y}}}; \mathbf{w}) \Big) \\ = \; & \xi_i - H_\alpha(\mathcal{Q}_i^{\mathbf{y}_i}; \mathbf{w}). \end{aligned} \tag{3.7}$$

Canceling the common term $H_\alpha(\mathcal{Q}_i^{\mathbf{y}_i}; \mathbf{w})$ in the first and last expressions of the above inequalities proves the proposition. □

The above proposition raises the question of the relationship between M3E models and the recently proposed latent SVM formulation [17, 60, 66], which was also shown to minimize an upper bound on the loss function [66]. Our next proposition provides an answer to this question by showing that the M3E model corresponding to the minimum entropy (that is, $\alpha \to \infty$) is equivalent to latent SVM.

Proposition 2. When $\alpha = \infty$, problem (3.6) is equivalent to latent SVM.

The proof is omitted since it follows simply by substituting the minimum entropy H_∞ (see equation (3.2)) in problem (3.6).

3.6 Optimization

While problem (3.6) is not convex, it has a *tractable* form that allows us to obtain an accurate set of parameters. Specifically, the following proposition holds true.

Proposition 3. Problem (3.6) is a difference-of-convex program for all values of $\alpha \neq 1$.

Proof Sketch. The objective function of problem (3.6) is clearly convex is $\mathbf{w}$ and slack variables ξ_i. The non-convexity arises due to the constraints. Specifically, the

constraints can be simplified as

$$\begin{aligned} & \frac{1}{1-\alpha}\log\sum_{\mathbf{h}}\exp(\alpha\mathbf{w}^\top\Psi(\mathbf{x}_i,\overline{\mathbf{y}},\mathbf{h})) \\ - & \frac{1}{1-\alpha}\log\sum_{\mathbf{h}}\exp(\mathbf{w}^\top\Psi(\mathbf{x}_i,\overline{\mathbf{y}},\mathbf{h})) \\ - & \frac{1}{1-\alpha}\log\sum_{\mathbf{h}}\exp(\alpha\mathbf{w}^\top\Psi(\mathbf{x}_i,\mathbf{y}_i,\mathbf{h})) \\ + & \frac{1}{1-\alpha}\log\sum_{\mathbf{h}}\exp(\mathbf{w}^\top\Psi(\mathbf{x}_i,\mathbf{y}_i,\mathbf{h})) \\ \geq & \Delta(\mathbf{y}_i,\overline{\mathbf{y}})-\xi_i. \end{aligned} \tag{3.8}$$

Since each term in the LHS of the above constraint has the so-called *log-sum-of-exponentials* form that is known to be convex, it follows that problem (3.6) is a difference-of-convex program. In other words, each of its constraints can be written in the form $f_i(\mathbf{w}) - g_i(\mathbf{w}) \leq 0$, where both $f_i(\mathbf{w})$ and $g_i(\mathbf{w})$ are convex. □

An approximate solution to difference-of-convex programs can be obtained using the concave-convex procedure (CCCP) [67]. Briefly, starting with an initial estimate $\mathbf{w}_0$, CCCP approximates the convex function $g_i(\mathbf{w})$ using a linear function $g_i'(\mathbf{w})$ whose slope is defined by the tangent of $g_i(\mathbf{w})$ at the current estimate $\mathbf{w}_t$. Replacing $g_i(\mathbf{w})$ by $g_i'(\mathbf{w})$ in the constraints results in a convex program, which is solved optimally to obtain a new estimate $\mathbf{w}_{t+1}$. The entire process is repeated until the objective function of the problem cannot be reduced below a user-specified tolerance.

The CCCP algorithm is guaranteed to provide a saddle point or local minimum solution to problem (3.6) [61]. However, it requires solving a series of optimization problems whose constraints are in the log-sum-of-exponentials form. While these constraints are convex, and the resulting problem can be solved in polynomial time, the typical runtime of the standard solvers is prohibitively large for real world applications. In § 3.6.2 we propose a novel trust region style algorithm that provides an approximate solution to problem (3.6) by solving a series of structured SVM problems. However, we begin by describing an important exception, corresponding to the minimum entropy (that is, $\alpha \to \infty$), where the CCCP algorithm itself reduces to a series of structured SVM problems.

3.6.1 Learning with the Minimum Entropy

While Proposition 2 demonstrates that the minimum entropy M3E model and the latent SVM are equivalent, there is a subtle but important difference in their respective optimization using CCCP. Consider the CCCP algorithm for the minimum entropy M3E model, described in Algorithm 5. The M3E model specifies a margin between the ground-truth

Algorithm 5 *The* CCCP *algorithm for parameter estimation of the minimum entropy* M3E *model.*

input $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \cdots, (\mathbf{x}_n, \mathbf{y}_n)\}$, $\mathbf{w}_0$, ϵ.

1: $t \leftarrow 0$

2: **repeat**

3: Update $\mathbf{h}_i^* = \text{argmax}_{h_i \in \mathcal{H}} \mathbf{w}_t^\top \Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i)$.

4: Update $\mathbf{w}_{t+1}$ by fixing the latent variables to $\mathbf{h}_i^*$ and solving the following convex problem:

$$\begin{aligned} \min_{\mathbf{w}, \xi_i \geq 0} \quad & \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{n}\sum_i \xi_i, \\ & \mathbf{w}^\top(\Psi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i^*) - \Psi(\mathbf{x}_i, \overline{\mathbf{y}}, \overline{\mathbf{h}})) \\ & \geq \Delta(\mathbf{y}_i, \overline{\mathbf{y}}) - \xi_i, \\ & \forall \overline{\mathbf{y}} \neq \mathbf{y}_i, \forall \mathbf{h} \in \mathcal{H}, \forall (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}. \end{aligned} \tag{3.9}$$

5: $t \leftarrow t + 1$.

6: **until** Objective function cannot be decreased below tolerance ϵ.

output $\mathbf{y}_i$ and all other *incorrect* outputs $\overline{\mathbf{y}} \neq \mathbf{y}_i$. In order to ensure that the problem defines a valid upper bound, it constrains the slack variables to be non-negative, that is, $\xi_i \geq 0$. During CCCP, this results in a succession of the convex optimization problems (3.9). In contrast, latent SVM simply specifies a margin between the ground-truth output and all outputs including the ground-truth (which ensures $\xi_i \geq 0$ since $\Delta(\mathbf{y}_i, \mathbf{y}_i) = 0$) [66]. During CCCP, this results in the following additional set of constraints:

$$\begin{aligned} & \mathbf{w}^\top(\Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i^*) - \Phi(\mathbf{x}_i, \mathbf{y}_i, \overline{\mathbf{h}})) \geq -\xi_i, \ \xi_i \geq 0, \\ & \forall \overline{\mathbf{h}} \in \mathcal{H}, \forall (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}. \end{aligned} \tag{3.10}$$

The additional constraints of latent SVM encourage the most likely estimates of the latent variables to remain unchanged during the parameter update step (step 4), since they try to maximize the margin between the log probability of $(\mathbf{y}_i, \mathbf{h}_i^*)$ and the log probabilities of $(\mathbf{y}_i, \overline{\mathbf{h}})$. Intuitively, this is a bad idea since it could make the algorithm converge earlier than desired. In our experiments we show that the minimum entropy M3E model provides better results than latent SVM.

3.6.2 Learning with General Entropies

As mentioned earlier, when $\alpha \neq \infty$, the CCCP algorithm requires us to solve a series of convex problem whose constraints contain terms in the log-sum-of-exponentials form. This limits the ability of CCCP for learning the parameters of a general M3E model using large datasets. To make M3E practically useful, we propose a novel optimization

Algorithm 6 *The algorithm for parameter estimation of the* M3E *model with general* α.

input $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \cdots, (\mathbf{x}_n, \mathbf{y}_n)\}$, $\mathbf{w}_0$, ϵ.

1: $t \leftarrow 0$

2: **repeat**

3: For each input $(\mathbf{x}_i)$ and output $\mathbf{y} \in \mathcal{Y}$, compute the following terms

$$\begin{aligned} G_\alpha(\mathcal{Q}_i^{\mathbf{y}}; \mathbf{w}_t) &= \nabla_{\mathbf{w}} H_\alpha(\mathbf{Q}_i^{\mathbf{y}}; \mathbf{w})|_{\mathbf{w}_t}, \\ C_\alpha(\mathcal{Q}_i^{\mathbf{y}}; \mathbf{w}_t) &= H_\alpha(\mathbf{Q}_i^{\mathbf{y}}; \mathbf{w}_t) - \mathbf{w}_t^\top G_\alpha(\mathbf{Q}_i^{\mathbf{y}}; \mathbf{w}_t). \end{aligned} \tag{3.11}$$

The above terms can be used to approximate the Rényi entropy $H_\alpha(\mathcal{Q}_i^{\mathbf{y}}; \mathbf{w})$ using the first-order Taylor's series approximation as

$$\begin{aligned} H_\alpha(\mathbf{Q}_i^{\mathbf{y}}; \mathbf{w}) \approx & \quad H'_\alpha(\mathbf{Q}_i^{\mathbf{y}}; \mathbf{w}) \\ = & \quad \mathbf{w}^\top G_\alpha(\mathbf{Q}_i^{\mathbf{y}}; \mathbf{w}_t) + C_\alpha(\mathbf{Q}_i^{\mathbf{y}}; \mathbf{w}_t) \end{aligned} \tag{3.12}$$

4: Update $\mathbf{w}_{t+1}$ by solving the following convex problem:

$$\begin{aligned} \min_{\mathbf{w}, \xi_i \geq 0} \quad & \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{n}\sum_i \xi_i, \\ & H'_\alpha(\mathcal{Q}_i^{\overline{\mathbf{y}}}; \mathbf{w}) - H'_\alpha(\mathcal{Q}_i^{\mathbf{y}_i}; \mathbf{w}) \\ & \geq \Delta(\mathbf{y}_i, \overline{\mathbf{y}}) - \xi_i, \\ & \forall \overline{\mathbf{y}} \neq \mathbf{y}_i, \forall (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}, \\ & ||\mathbf{w} - \mathbf{w}_t||^2 \leq \mu. \end{aligned} \tag{3.13}$$

The term μ specifies a trust region where $H'_\alpha(\cdot)$ accurately approximates $H'_\alpha(\cdot)$.

5: $t \leftarrow t + 1$.

6: **until** Objective function cannot be decreased below tolerance ϵ.

approach for problem (3.6), which is outlined in Algorithm 6. Our approach consists of two main steps: (i) linearization (step 3); and (ii) parameter update (step 4). During linearization, we obtain an approximation of the Rényi entropy for a general α using a first-order Taylor's series expansion around the current parameter estimate $\mathbf{w}_t$. This approximation, denoted by $H'_\alpha(\cdot; \mathbf{w})$, is a linear function in $\mathbf{w}$. Hence, the parameter update step reduces to solving the structured SVM problem (3.13). Since linearization provides a good approximation for the Rényi entropy near $\mathbf{w}_t$, but a poor approximation far from $\mathbf{w}_t$, we restrict the update step to search for new parameters only around $\mathbf{w}_t$ (analogous to defining a trust region for non-convex problems [7]) by specifying the constraint $||\mathbf{w} - \mathbf{w}_t||^2 \leq \mu$. It is worth noting that this constraint can be easily incorporated into any standard structured SVM solver [32, 58, 63, 65], which makes Algorithm 6 computationally tractable.

The parameter μ governs the size of the trust region, and therefore, influences the trade-off between the speed and the accuracy of our algorithm. Specifically, a large μ will allow us to search over a large space thereby increasing the speed, but may converge to an inaccurate solution due to the poor approximation provided by the linearization step over

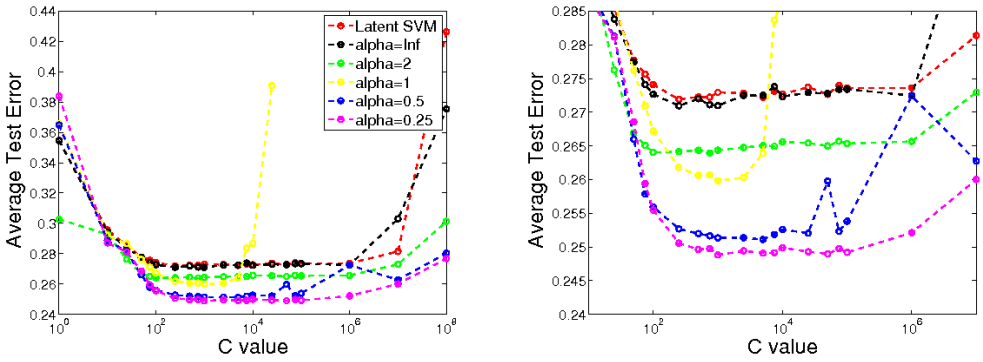


FIGURE 3.1: *The average (over all proteins and folds) test errors for the motif finding experiment across varying values of C and α. Left: All values of C and α that were used in our experiments. Right: Zoomed-in version to highlight the difference in performance among the various methods. For each (protein,fold) pair, the model with the best train error out of 4 random initializations was chosen. Further results are provided in Table 3.1. As can be seen, lower values of α achieve the best test errors, and larger values of α approach the performance of latent* SVM*, which solves the same problem as an* M3E *with $\alpha = \infty$ with a slightly different optimization procedure. Note that the results for $\alpha = 1$ become unstable for larger values of C due to numerical instability during parameter estimation. Best viewed in color.*

the entire trust region. A small μ will restrict us to a region where the approximation provided by the linearization is accurate, but will slow down the algorithm. In practice, we found that the following simple strategy provided a desirable trade-off. We start with a large value $\mu = \mu_{max}$, obtain the solution $\mathbf{w}'$ and compute the objective of problem (3.6). If the objective function has decreased above the tolerance ϵ since the previous iteration, then we set $\mathbf{w}_{t+1} = \mathbf{w}'$. Otherwise, we anneal $\mu \leftarrow \mu/\lambda$ and solve problem (3.13) to obtain a new $\mathbf{w}'$. Algorithm 6 is said to converge when the difference in the objective of problem (3.6) computed at $\mathbf{w}_{t+1}$ and $\mathbf{w}_t$ is below tolerance ϵ.

3.7 Experiments

We now demonstrate the efficacy of M3E models using two standard machine learning applications that were addressed using the latent SVM formulation in the previous chapter: motif finding and image classification. Specifically, we show how the more general M3E formulation can be used to significantly improve the results compared to latent SVM.

3.7.1 Motif Finding

Problem Formulation. We consider the problem of binary classification of DNA sequences. Specifically, the input vector $\mathbf{x}$ consists of a DNA sequence of length l (where

each element of the sequence is a nucleotide of type A, G, T or C) and the output space $\mathcal{Y} = \{0, 1\}$. In our experiments, the classes correspond to two different types of genes: those that bind to a protein of interest with high affinity and those that do not. The positive sequences are assumed to contain particular patterns, called *motifs*, of length m that are believed to be useful for classification. However, the starting position of the motif within a gene sequence is often not known. Hence, this position is treated as the latent variable $\mathbf{h}$. Given an input $\mathbf{x}$, an output $\mathbf{y}$ and a latent variable $\mathbf{h}$, we use the joint feature vector suggested by [66]. The loss function Δ is the standard 0-1 classification loss. The number of possible values of the latent variables is small (of the order of the size of the DNA sequence), which makes this problem tractable within the M3E formulation without having to resort to approximate inference schemes.

Dataset. We use the publicly available UniProbe dataset [6] that provides positive and negative DNA sequences for 177 proteins. For this work, we chose five proteins at random. The total number of sequences per protein is roughly $40,000$. For all the sequences, the motif length m is known. In order to specify a classification task for a particular protein, we randomly split the sequences into roughly 50% for training and 50% for testing. We report results using 5 folds.

Results. Figure 3.1 shows the test errors for latent SVM and various M3E models across different values of C. The values are averaged over all 25 (protein, fold) pairs. For each protein and each fold, we initialize the methods using four different random seeds, and report the test error corresponding to the seed with the best training error (with ties broken by training objective value). As the results indicate, using high values of α provides similar results to latent SVM. Recall that while the objective for an M3E model with $\alpha = \infty$ is equivalent to that of latent SVM, the optimizations for each are different, thereby yielding different results. The M3E models with low values of α achieve significantly better performance than latent SVM, indicating that these values are more suitable for predicting whether a DNA sequence has a high affinity towards binding to a particular protein. Table 3.1 shows the average test error for the best C and α values. The best M3E model achieves 2.2% lower test error than the best latent SVM model.

3.7.2 Image Classification

Problem Formulation. Given a set of images along with labels that indicate the presence of a particular object category in the image (for example, a mammal), our goal is to learn discriminative object models. Specifically, we consider two types of problems:

	Latent SVM	M3E
Protein 052	$C = 5000$	$C = 7500, \alpha = 0.25$
Train Error	28.6%	**26.9%**
Test Error	29.2%	**27.4%**
Protein 074	$C = 5000$	$C = 10000, \alpha = 0.25$
Train Error	26.7%	**23.6%**
Test Error	27.6%	**24.2%**
Protein 108	$C = 500$	$C = 10000, \alpha = 0.25$
Train Error	26.8%	**25.0%**
Test Error	27.1%	**25.3%**
Protein 131	$C = 750$	$C = 750, \alpha = 0.25$
Train Error	28.8%	**27.3%**
Test Error	29.2%	**27.6%**
Protein 146	$C = 1000$	$C = 5000, \alpha = 0.25$
Train Error	22.2%	**19.9%**
Test Error	22.5%	**20.1%**
Average		
Train Error	26.6%	**24.5%**
Test Error	27.1%	**24.9%**

TABLE 3.1: *Average training and test errors for 5 randomly chosen proteins, split into 5 random folds. For each protein, the parameters that achieved the best mean training error across folds were chosen, and those parameters are shown. The* M3E *models outperform latent* SVM *on each protein, and overall yield an improvement of over* 2% *in terms of both the training error and the test error.*

(i) given an image containing an instance of an object category from a fixed set of c categories, predict the correct category (that is, a multi-class classification problem, where the set of outputs $\mathcal{Y} = \{0, 1, \cdots, c-1\}$); (ii) given an image, predict whether it contains an instance of an object category of interest or not (that is, a binary classification problem, where $\mathcal{Y} = \{0, 1\}$). In practice, although it is easy to mine such images from free photo-sharing websites such as Flickr, it is burdensome to obtain ground truth annotations of the exact location of the object in each image. To avoid requiring these human annotations, we model the location of objects as latent variables. Formally, for a given image $\mathbf{x}$, label $\mathbf{y} \in \mathcal{Y}$ and location $\mathbf{h}$, the score is modeled as $\mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{y}, \mathbf{h}) = \mathbf{w}_\mathbf{y}^\top \Phi_\mathbf{h}(\mathbf{x})$, where $\mathbf{w}_\mathbf{y}$ are the parameters that corresponds to the label $\mathbf{y}$ and $\Phi_\mathbf{h}(\cdot)$ is the HOG [13, 17] feature extracted from the image at position $\mathbf{h}$ (the size of the object is assumed to be the same for all images—a reasonable assumption for our datasets). The number of possible values of the latent variables is of the order of the number of pixels in an image, which makes M3E learning tractable without resorting to approximate inference. For both the settings (multi-class classification and binary classification), the loss function $\Delta(\mathbf{y}, \hat{\mathbf{y}})$ is the standard 0-1 classification loss.

Dataset. We use images of 6 different mammals (approximately 45 images per mammal) that have been previously employed for object localization [24]. We split the images

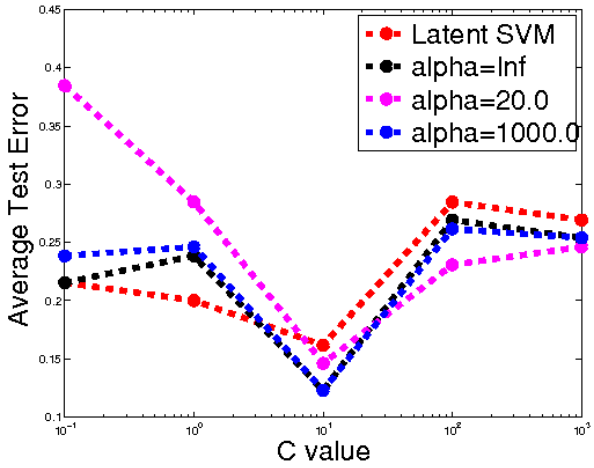


FIGURE 3.2: *Test errors for the multi-class classification setting. Latent* SVM *performs poorly compared to the* M3E *models, which attain the best test error of* 12.3% *for all* $\alpha \geq 1000.0$ *in our experiments.*

of each category into approximately 90% for training and 10% for testing. We report results for 5 such randomized folds.

Results. As in the motif finding application, we initialize each method using four different random seeds, and report the test error corresponding to the seed with the best training error (with ties broken by training objective value). Fig. 3.2 shows the results of the multi-class classification setting, averaged over all 5 folds. As can be seen, latent SVM performs poorly compared to the M3E models. All M3E models with $\alpha \geq 1000.0$ (including $\alpha = \infty$) provide the best test error of 12.3%. Fig. 3.3 shows the average (over 5 folds) test errors for all 6 binary classification problems. For the "deer" class, M3E models achieve the same performance as latent SVM. For the "bison" class, similar to the multi-class classification setting, $\alpha = \infty$ provides the best results. For the other four classes ("elephant", "giraffe", "llama" and "rhino"), the best performing M3E models use a smaller value of α (between 2.0 and 8.0). This illustrates the importance of selecting the *right* value of α for the problem at hand, instead of relying solely on the minimum entropy, as is the case with latent SVM. Overall, the average test classification errors across all six mammals are 5.7% for latent SVM, 5.4% for the minimum entropy M3E model, and 4.2% for the best M3E model.

3.8 Discussion

We presented a new family of LVMs, called M3E models, that predict the output of a given input as the one that results in the minimum Rényi entropy of the corresponding

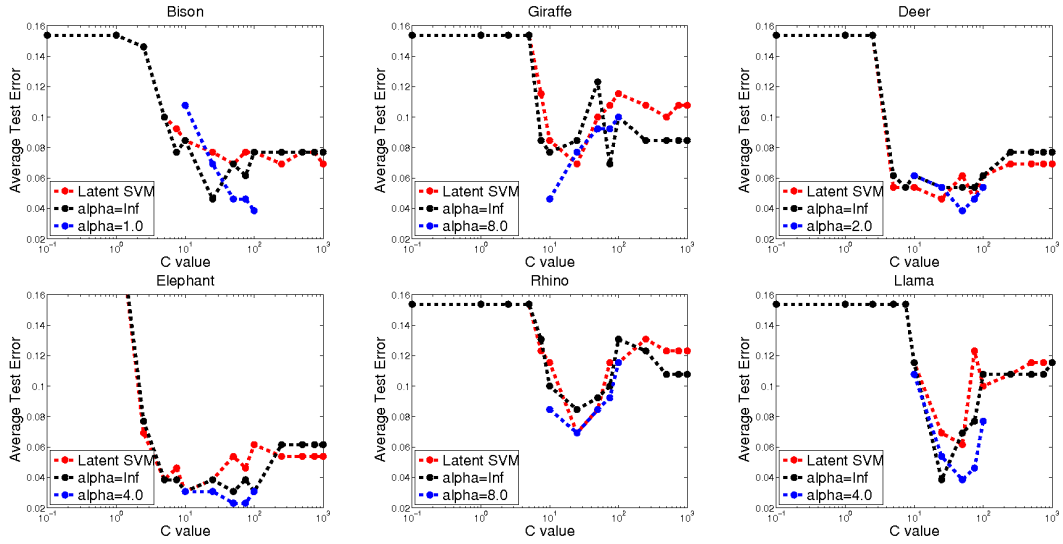


FIGURE 3.3: *Image classification test errors for all six mammal classes. Each number is averaged across 5 random folds; in each fold, the model with the best training error out of 4 random initializations was chosen. For each mammal, the* M3E *model that achieved the best test error is shown with the corresponding α value indicated, along with latent* SVM *and the minimum entropy* M3E *model ($\alpha = \infty$). The average test classification errors across all six mammals are* 5.7% *for latent* SVM*,* 5.4% *for the minimum entropy* M3E *model, and* 4.2% *for the best* M3E *model. Best viewed in color.*

generalized distribution. The upshot of this is that the predicted output (i) has a high probability; and (ii) minimizes the uncertainty in the latent variables. We showed how the parameters of an M3E model are learned using a max-margin formulation can that be viewed as minimizing an upper bound on a user-defined loss. Included as a special case in our family of models is the latent SVM framework. Empirically, we demonstrated that the more general M3E models can outperform latent SVM.

Similar to other LVMs, when the latent variable space is small, or when the underlying distribution is tractable (for example, a small tree-width distribution), the parameters of an M3E model can be learned accurately. Specifically, in this case, parameter learning is equivalent to solving a difference-of-convex optimization problem using CCCP [67] or the self-paced learning algorithm described in the previous chapter. When the latent variables lie in an exponentially large space, M3E can lend itself to approximate optimization. For example, we could design an appropriate variational inference procedure that best approximates a Rényi entropy of interest. This offers an interesting direction for future work.

The introduction of M3E models throws up several interesting questions. For example, is it possible to determine the *best* value of α for a type of latent variable? Given a problem that requires different types of latent variables (say, learning an image segmentation model using partially segmented images, or bounding box annotations, or image-level labels), should we employ different α values for them? Can these values themselves be

learned? Answers to these questions would not only be of great practical importance, but would also reveal interesting theoretical properties of the M3E family of models.

Finally, we note that while the method described in this chapter employs Rényi entropy, other forms of entropy, such as the generalized Rényi entropy [46], the Havrada-Charvat entropy [23] or Rao's quadratic entropy [55], are also readily applicable within our max-margin learning framework. In addition, the generalized Rényi entropy can be easily optimized using our trust region style algorithm. Designing efficient optimization techniques for learning M3E models with other entropies remains an open challenge.

Chapter 4

Dissimilarity Coefficient Learning with General Loss Functions

4.1 Introduction

In the previous chapters, we have consider the problem of learning latent variable models (LVM) when the loss function does not depend on the latent variables. In other words, we have looked at applications that do not require an accurate prediction of the latent variables at test time. As is typical for an LVM, we have employed a single distribution over the input, the output and the latent variables. In this setting, a natural framework would be to model the uncertainty in the value of the latent variables and learn an LVM by marginalizing them out (for example, in the case of the expectation-maximization, or EM, algorithm or M3E models). However, such an approach is unsuited for applications where the accuracy of the latent variable prediction is crucial during test time. For example, consider the problem of learning an object detector from images that have been labeled as either positive (contains an instance of the object category) or negative (does not contain an instance of the object category). In this case, the bounding box of the object in the positive images would have to be modeled as latent variables. However, at test time, the accuracy would rely not only on our ability to label a previously unseen image as positive or negative, but by accurately localizing the object instance in the positive image as well.

As an alternative to marginalizing the latent variables, we can use a delta distribution (which is 0 everywhere except for one value) that provides a pointwise estimate of the output and the latent variables (for example, in the case of the latent SVM, framework). However, discarding the uncertainty in latent variables can make such an approach

prone to error due to noise (for example, background clutter that can be confused with an object in feature space).

The above argument illustrates the deficiency of using a single joint distribution over the output and the latent variables to address two separate tasks: (i) modeling the uncertainty over latent variables during training; and (ii) making accurate predictions during testing. We address this deficiency by proposing a novel framework that consists of two distributions: (i) a conditional distribution to model the uncertainty of the latent variables for a given input-output pair; and (ii) a delta distribution to predict the output and the latent variables for a given input. In order to learn the distributions from a training dataset, we build on the intuition that they should *agree* with each other, that is, (i) the output predicted by the delta distribution should match the ground-truth output; and (ii) the latent variables predicted by the delta distribution should have a high probability according to the conditional distribution. Due to the limited representational power of any model we may not be able to achieve complete agreement (that is, all outputs are predicted correctly, and all predicted latent variables have probability 1). In order to make the two distributions as similar as possible, we minimize a regularized upper bound on a loss-based dissimilarity measure [55] between the distributions.

Unlike previous loss-based learning frameworks for LVMs, such as latent SVM, we consider a general loss function that not only depends on the output but also the latent variables. Such a loss function is essential when solving problems that require the accurate prediction of latent variables (for example, the aforementioned object detection problem). By not restricting the form of the loss function, our framework greatly enhances the applicability of loss-based learning with latent variables. In fact, our framework can be viewed as a strict generalization of latent SVM in the sense that, when the loss function is independent of the true (unknown) value of the latent variables, it reduces to an latent SVM.

Throughout this chapter, we will assume that the latent variables are helpful in predicting the correct output of a sample. For example, if we want to distinguish between images of deers and elephants, we would expect that the background clutter to have similar appearance for both categories, and an accurate object localization to be essential for correct prediction. There may be cases where this assumption does not hold. For example, images of deers and cars could be distinguished by detecting roads, or other objects that are more commonly found in urban environments. However, even in such cases, we may be able to learn to detect the object by providing fully supervised annotations for a small fraction of training images, which would help guide the learner towards the correct object locations in other weakly supervised training images.

4.2 Related Work

The most commonly used method for learning the parameters of an LVM is the EM algorithm [14, 62], or its many variants [20]. While the EM algorithm has an elegant probabilistic interpretation of maximizing the likelihood of the ground-truth output, it marginalizes out the latent variables, which makes it unsuited to problems that require the accurate prediction of latent variables. Furthermore, it does not employ a user-specified loss function, which captures the user's assessment of the quality of the solution.

The most related works to our approach are latent SVM [17, 60, 66] and its recently proposed generalization called max-margin min-entropy models (or M3E for short) [47]. The parameters of an latent SVM or an M3E are learned by minimizing a regularized upper bound of the training loss. However, the loss function is restricted to be independent of the true (unknown) value of the latent variables. While such loss functions are useful, and in fact have been successfully employed in practice [9, 17, 60, 66], they cannot model several important problems, including the two employed in our experiments—object detection and action detection. In contrast, our framework allows the use of a general loss function. In section 4.4 we will show that, for loss functions that are independent of the true value of the latent variable, our framework reduces to an latent SVM.

In our earlier work [38], we proposed an iterative latent SVM strategy (or ILSVM for short) with the aim of using a general loss function. In section 4.5, we show that ILSVM corresponds to using delta functions to model the conditional distribution of the latent variables given the input and the output. In our experiments, we show that using a non-delta conditional distribution significantly outperforms ILSVM.

4.3 Preliminaries

Notation. As usual, we denote the input by $\mathbf{x} \in \mathcal{X}$, the output by $\mathbf{y} \in \mathcal{Y}$ and the latent variables by $\mathbf{h} \in \mathcal{H}$. The training dataset $\mathcal{D} = \{\mathbf{s}_i = (\mathbf{x}_i, \mathbf{y}_i), i = 1, \cdots, n\}$ consists of n input-output pairs (or samples) $\mathbf{s}_i$.

We denote the parameters of the delta distribution, which predicts the output and the latent variables for a given input, as $\mathbf{w}$. The parameters of the conditional distribution of the latent variables given the input and the output are denoted by $\boldsymbol{\theta}$.

We assume that the user specifies a loss function $\Delta(\mathbf{y}_1, \mathbf{h}_1, \mathbf{y}_2, \mathbf{h}_2)$ that measures the difference between $(\mathbf{y}_1, \mathbf{h}_1)$ and $(\mathbf{y}_2, \mathbf{h}_2)$. Similar to previous approaches, we assume that $\Delta(\mathbf{y}_1, \mathbf{h}_1, \mathbf{y}_2, \mathbf{h}_2) = 0$ if $\mathbf{y}_1 = \mathbf{y}_2$ and $\mathbf{h}_1 = \mathbf{h}_2$. Otherwise, $\Delta(\mathbf{y}_1, \mathbf{h}_1, \mathbf{y}_2, \mathbf{h}_2) \geq 0$.

Rao's Dissimilarity Coefficient. We provide a brief description of the dissimilarity measure used in our framework, which was first introduced by Rao [55]. Given a loss function $\Delta(\mathbf{z}_1, \mathbf{z}_2)$, where $\mathbf{z}_1, \mathbf{z}_2 \in \mathcal{Z}$, the diversity coefficient of two distributions $\mathrm{P}_i(\mathbf{z})$ and $\mathrm{P}_j(\mathbf{z})$ is defined as the expected loss between two samples drawn randomly from the two distributions respectively, that is,

$$H(\mathrm{P}_i, \mathrm{P}_j) = \sum_{\mathbf{z}_1 \in \mathcal{Z}} \sum_{\mathbf{z}_2 \in \mathcal{Z}} \Delta(\mathbf{z}_1, \mathbf{z}_2)\, \mathrm{P}_i(\mathbf{z}_1)\, \mathrm{P}_j(\mathbf{z}_2). \tag{4.1}$$

Using the diversity coefficient, the dissimilarity coefficient between the two distributions can be defined as the following Jensen difference:

$$D(\mathrm{P}_i, \mathrm{P}_j) = H(\mathrm{P}_i, \mathrm{P}_j) - \beta H(\mathrm{P}_i, \mathrm{P}_i) - (1-\beta) H(\mathrm{P}_j, \mathrm{P}_j), \tag{4.2}$$

where $\beta \in (0, 1)$. Note that, in [55], the value of β was fixed to 0.5 in order to ensure that the dissimilarity coefficient is symmetric for P_i and P_j. However, dissimilarity coefficients do not necessarily have to be symmetric (for example, the well-known Kullback-Liebler divergence is non-symmetric); hence we use the more general version shown in equation (4.2). Rao [55] showed that the above formulation generalizes other commonly used dissimilarity coefficients such as the Mahalanobis distance and the Gini-Simpson index. We refer the reader to [55] for details.

4.4 Loss-based Learning Framework

Using the above notation and definitions, we now provide the details of our learning framework. We begin by describing the distributions represented by the LVM.

4.4.1 Distributions

We wish to address two separate tasks: (i) to accurately model the distribution of the latent variables for a given input-output pair; and (ii) to accurately predict the output and latent variables for a given input (where accuracy is measured by a user-defined loss). Instead of addressing these two tasks with a single distribution as in previous works, we define two separate distributions, each focused on a single task.

Given an input $\mathbf{x}$, we define a delta distribution parameterized by $\mathbf{w}$ that predicts the output and the latent variables according to the following rule:

$$(\mathbf{y}(\mathbf{w}), \mathbf{h}(\mathbf{w})) = \operatorname*{argmax}_{(\mathbf{y}, \mathbf{h})} \mathbf{w}^\top \Psi(\mathbf{x}, \mathbf{y}, \mathbf{h}). \tag{4.3}$$

Here, $\Psi(\mathbf{x}, \mathbf{y}, \mathbf{h})$ is a joint feature vector of the input $\mathbf{x}$, the output $\mathbf{y}$ and the latent variables $\mathbf{h}$. Note that, although for simplicity we defined a linear rule in $\mathbf{w}$, we can also employ a non-linear kernel within our framework. Formally, the delta distribution is given by

$$\mathrm{P}_{\mathbf{w}}(\mathbf{y}, \mathbf{h}|\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{y} = \mathbf{y}(\mathbf{w}), \mathbf{h} = \mathbf{h}(\mathbf{w}), \\ 0 & \text{otherwise.} \end{cases} \tag{4.4}$$

As mentioned earlier, since the true value of the latent variables is unknown, we would like to model the uncertainty in their values. To this end, we define a separate conditional distribution parameterized by $\boldsymbol{\theta}$ such that

$$\mathrm{P}_{\boldsymbol{\theta}}(\mathbf{h}_i|\mathbf{s}_i) = \frac{1}{Z(\mathbf{s}_i; \boldsymbol{\theta})} \exp\left(\boldsymbol{\theta}^\top \Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i)\right), \tag{4.5}$$

where $Z(\mathbf{s}_i; \boldsymbol{\theta})$ is the partition function, which ensures that the distribution sums to one and $\Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i)$ is a joint feature vector of the input $\mathbf{x}_i$, the output $\mathbf{y}_i$ and the latent variables $\mathbf{h}_i$. This feature vector can be different than the joint feature vector used to specify the delta distribution $\mathrm{P}_{\mathbf{w}}(\cdot)$. Once again, a log-linear distribution is used only to simplify the description. Our framework is valid for any general form of the distribution $\mathrm{P}_{\boldsymbol{\theta}}(\cdot)$. Using the above conditional distribution, we also specify a joint distribution as follows:

$$\mathrm{P}'_{\boldsymbol{\theta}}(\mathbf{y}, \mathbf{h}_i|\mathbf{x}_i) = \begin{cases} \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{h}_i|\mathbf{s}_i) & \text{if } \mathbf{y} = \mathbf{y}_i, \\ 0 & \text{otherwise.} \end{cases} \tag{4.6}$$

As will be seen shortly, this joint distribution would allow us to employ Rao's dissimilarity coefficient in our learning framework.

4.4.2 The Learning Objective

Given a dataset $\mathcal{D}$ and a loss function $\Delta(\cdot)$, we propose to learn the parameters $\mathbf{w}$ and $\boldsymbol{\theta}$ such that it minimizes the corresponding dissimilarity coefficient over all training samples. Before delving into the details, we give a broad overview of our objective function.

For a fixed $\mathbf{w}$, if the predicted output $\mathbf{y}_i(\mathbf{w})$ is similar to the ground-truth output $\mathbf{y}_i$, our objective encourages the probability of the corresponding latent variables, that is $\mathrm{P}_{\boldsymbol{\theta}}(\mathbf{h}_i(\mathbf{w})|\mathbf{s}_i)$, and other similar latent variables, to be high. If the predicted output $\mathbf{y}_i(\mathbf{w})$ is dissimilar to the ground-truth output $\mathbf{y}_i$, our objective encourages the diversity coefficient of the corresponding distribution, that is $\mathrm{P}_{\boldsymbol{\theta}}(\cdot|\mathbf{s}_i)$, to be high. In other words, for a correctly predicting sample, the conditional distribution is *peaky*, while for an incorrectly predicted sample, the conditional distribution is *flat*.

For a fixed $\boldsymbol{\theta}$, our objective minimizes the *expected loss* of the prediction $(\mathbf{y}_i(\mathbf{w}), \mathbf{h}_i(\mathbf{w})))$ over all the training samples $\mathbf{s}_i$. This is a key point of our formulation, as the expected loss incorporates the uncertainty of the latent variable values while learning the parameters $\mathbf{w}$. Formally, the expected loss of a pair of output and latent variables $(\mathbf{y}, \mathbf{h})$ for the sample $\mathbf{s}_i$, measured with respect to $\mathrm{P}_{\boldsymbol{\theta}}(\cdot|\mathbf{s}_i)$, is defined as

$$\Delta_i(\mathbf{y}, \mathbf{h}; \boldsymbol{\theta}) = \sum_{\mathbf{h}_i} \Delta(\mathbf{y}_i, \mathbf{h}_i, \mathbf{y}, \mathbf{h})\, \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{h}_i|\mathbf{s}_i), \tag{4.7}$$

that is, it is the expectation of the loss between $(\mathbf{y}, \mathbf{h})$ and $(\mathbf{y}_i, \mathbf{h}_i)$, where the expectation is taken over the distribution of the unknown latent variables $\mathbf{h}_i$.

We now provide a mathematical description of our learning framework. However, throughout this section and the next, we will reiterate the above intuition at the appropriate places. Our training objective is the sum of the dissimilarity coefficient between $\mathrm{P}_{\mathbf{w}}(\cdot)$ and $\mathrm{P}'_{\boldsymbol{\theta}}(\cdot)$ over all training samples. Using the definition of dissimilarity coefficient in equation (4.2), the objective can be written in terms of expected loss as

$$\begin{aligned} D(\mathbf{w}, \boldsymbol{\theta}) &= \frac{1}{n}\left(\sum_{i=1}^{n} H_i(\mathbf{w}, \boldsymbol{\theta}) - \beta H_i(\boldsymbol{\theta})\right), \\ H_i(\mathbf{w}, \boldsymbol{\theta}) &= \Delta_i(\mathbf{y}_i(\mathbf{w}), \mathbf{h}_i(\mathbf{w}); \boldsymbol{\theta}), \\ H_i(\boldsymbol{\theta}) &= \sum_{\mathbf{h}'_i} \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{h}'_i|\mathbf{s}_i)\Delta_i(\mathbf{y}_i, \mathbf{h}'_i; \boldsymbol{\theta}). \end{aligned} \tag{4.8}$$

Note that the diversity coefficient of $\mathrm{P}_{\mathbf{w}}(\cdot)$ is 0 since it is a delta distribution. Hence, the term $H_i(\mathbf{w})$ vanishes from the above objective.

Minimizing the objective (4.8) encourages two desirable properties: (i) the predicted output $\mathbf{y}_i(\mathbf{w})$ should be similar to the ground-truth output $\mathbf{y}_i$; and (ii) the predicted latent variable $\mathbf{h}_i(\mathbf{w})$ should be similar to the latent variables with high probabilities $\mathrm{P}_{\boldsymbol{\theta}}(\mathbf{h}_i|\mathbf{x}_i, \mathbf{y}_i)$. Importantly, the similarity (or, to be more precise, the dissimilarity) of the outputs and the latent variables is specified by the loss function $\Delta(\cdot)$. Hence, during learning, the parameters $\mathbf{w}$ and $\boldsymbol{\theta}$ are tuned according to the user's domain knowledge regarding the quality of a solution. This ability to learn loss-specific parameters is absent in traditional frameworks such as EM and its variants.

4.4.3 Upper Bound on the Learning Objective

While the objective (4.8) is smooth and differentiable in $\boldsymbol{\theta}$, for most commonly used choices of the loss function it is highly non-smooth in $\mathbf{w}$. The non-smoothness of the objective results in a difficult optimization problem, which makes the learner prone to

bad local minimum solutions. In order to overcome this deficiency, we minimize an upper bound on the objective, similar to the latent SVM formulation [60, 66].

Specifically, we upper bound the term $H_i(\mathbf{w}, \boldsymbol{\theta})$, which depends on $\mathbf{w}$, using $\xi_i(\mathbf{w}, \boldsymbol{\theta})$ defined as follows.

$$\begin{aligned} \xi_i(\mathbf{w}, \boldsymbol{\theta}) \leq \quad & \max_{\mathbf{y},\mathbf{h}} \left\{ \mathbf{w}^\top \Psi(\mathbf{x}_i, \mathbf{y}, \mathbf{h}) + \Delta_i(\mathbf{y}, \mathbf{h}; \boldsymbol{\theta}) \right\} \\ & - \max_{\mathbf{h}} \mathbf{w}^\top \Psi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}) \end{aligned} \tag{4.9}$$

Using the above inequalities, the objective $D(\mathbf{w}, \boldsymbol{\theta})$ can be upper bounded as

$$U(\mathbf{w}, \boldsymbol{\theta}) = \frac{1}{n} \left(\sum_{i=1}^{n} \xi_i(\mathbf{w}, \boldsymbol{\theta}) - \beta H_i(\boldsymbol{\theta}) \right). \tag{4.10}$$

However, if we learn the parameters $\mathbf{w}$ and $\boldsymbol{\theta}$ by minimizing the above upper bound (or indeed the original objective function), we run the risk of overfitting to the training data. In order to prevent this, we introduce regularization terms for the parameters. For this work, we use ℓ_2 norms, though other norms may also be employed. To summarize, the parameters are learned by solving the following optimization problem:

$$(\mathbf{w}^*, \boldsymbol{\theta}^*) = \underset{(\mathbf{w}, \boldsymbol{\theta})}{\operatorname{argmin}} \frac{1}{2}||\mathbf{w}||^2 + \frac{J}{2}||\boldsymbol{\theta}||^2 + CU(\mathbf{w}, \boldsymbol{\theta}), \tag{4.11}$$

where the hyperparameters J and C are the relative weights for the regularization of $\boldsymbol{\theta}$ and the upper bound of the dissimilarity coefficient respectively. Note that the upper bound derivation and the resulting optimization problem are similar to the latent SVM framework. In fact, the problem can be shown to be a strict generalization of latent SVM.

Observation 1. When the loss function does not depend on the value of the latent variables, problem (4.11) is equivalent to the problem of learning an latent SVM.

This observation follows from the fact that, when the loss function is independent of the latent variables, $H_i(\boldsymbol{\theta}) = \Delta_i(\mathbf{y}_i; \boldsymbol{\theta}) \sum_{\mathbf{h}'_i} \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{h}'_i | \mathbf{s}_i) = 0$. Hence, the optimization problem is equivalent to minimizing the sum of the regularization of $\mathbf{w}$ and $\xi_i(\mathbf{w}, \boldsymbol{\theta})$ (which are equivalent to the slack variables that model the upper bound of the loss function for the sample $\mathbf{s}_i$ in latent SVM). In fact, even if the loss function does depend on the predicted latent variable $\mathbf{h}_i(\mathbf{w})$, the optimization problem (4.11) still generalizes latent SVM. This follows from the fact that, in this case, the latent SVM problem is equivalent to using delta distributions to model $\mathrm{P}_{\boldsymbol{\theta}}(\cdot)$. Formal proofs are omitted.

4.5 Optimization

While the upper bound derived in the previous section still results in a non-smooth and non-convex optimization problem, we obtain an approximate solution using block coordinate descent. Specifically, starting with some initial estimate of parameters, we alternately fix one of the two sets of parameters (either $\mathbf{w}$ or $\boldsymbol{\theta}$) while optimizing problem (4.11) over the other set of parameters. The process is said to terminate when the decrease in the objective falls below $C\epsilon$, where C is the hyperparameter in problem (4.11) and ϵ is a user specified tolerance. The following subsections provide the details of the optimization over each set of parameters.

4.5.1 Optimization over w

For a fixed $\boldsymbol{\theta}$, problem (4.11) can be interpreted as minimizing a regularized upper bound on the expected loss induced by $\mathbf{w}$, that is,

$$\sum_i \Delta_i(\mathbf{y}_i(\mathbf{w}), \mathbf{h}_i(\mathbf{w}); \boldsymbol{\theta}), \tag{4.12}$$

since the term $H_i(\boldsymbol{\theta})$ is a constant for all samples $\mathbf{s}_i$. The expected loss is an intuitive objective: it gives more weight to the loss corresponding to the latent variables that have a high probability and less weight to those corresponding to the latent variables with low probability. Formally, for a fixed $\boldsymbol{\theta}$, the optimization problem (4.11) reduces to the following:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \tfrac{1}{2}||\mathbf{w}||^2 + \tfrac{C}{n}\textstyle\sum_i \xi_i \\ \text{s.t.} \quad & \xi_i = \max_{\mathbf{y},\mathbf{h}} \left\{\mathbf{w}^\top \Psi(\mathbf{x}_i, \mathbf{y}, \mathbf{h}) + \Delta_i(\mathbf{y}, \mathbf{h}; \boldsymbol{\theta})\right\} \\ & \qquad - \max_{\mathbf{h}} \mathbf{w}^\top \Psi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}). \end{aligned} \tag{4.13}$$

The following observation provides us with an efficient algorithm for the above optimization problem.

Observation 2. Problem (4.13) is a difference-of-convex program.

The regularization term $||\mathbf{w}||^2$ is convex. The term $\xi_i(\mathbf{w}, \boldsymbol{\theta})$ is a difference of two functions that are the pointwise maximum of a set of linear functions. Since the pointwise maximum of convex functions is convex, the observation follows. Similar to latent SVM, a local minimum or saddle point solution of problem (4.13) can be found using the concave-convex procedure (CCCP) [60, 66]. The main steps of CCCP are outlined in Algorithm 7.

It iteratively estimates the value of the latent variables using the current estimate of $\mathbf{w}$, and updates the parameters by solving a convex optimization problem (4.14). There are several efficient algorithms for problem (4.14), for example [32, 58]. In this chapter, we use the 1-slack reformulation method proposed in [32]. We can also solve problem (4.13) using the self-paced learning algorithm (see chapter 2), which can potentially improve the performance of our framework. However, in this chapter, we restrict ourselves to the simpler and more efficient CCCP algorithm.

Algorithm 7 *The* CCCP *algorithm for optimizing* $\mathbf{w}$.

input Dataset $\mathcal{D}$, initial estimate $\mathbf{w}_0$, tolerance ϵ.

1: $t \leftarrow 0$.
2: **repeat**
3: Update $\mathbf{h}_i^* = \text{argmax}_{\mathbf{h}_i} \mathbf{w}_t^\top \Psi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i)$.
4: Estimate the updated parameter $\mathbf{w}_{t+1}$ by solving the following convex optimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \tfrac{1}{2}||\mathbf{w}||^2 + \tfrac{C}{n}\textstyle\sum_i \xi_i \\ \text{s.t.} \quad & \xi_i \geq \mathbf{w}^\top \Psi(\mathbf{x}_i, \mathbf{y}, \mathbf{h}) + \Delta_i(\mathbf{y}, \mathbf{h}; \boldsymbol{\theta}) \\ & \quad -\mathbf{w}^\top \Psi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i^*), \forall \mathbf{y}, \mathbf{h}. \end{aligned} \tag{4.14}$$

5: $t \leftarrow t + 1$.
6: **until** Objective cannot be decreased below $C\epsilon$.

Problem (4.13) requires the computation of the expected loss $\Delta_i(\mathbf{y}, \mathbf{h}; \boldsymbol{\theta})$ as defined in equation (4.7), which can be found in $O(|\mathcal{H}|)$ time for each pair of $(\mathbf{y}, \mathbf{h})$ (where $\mathcal{H}$ is the space of all latent variables). For a sufficiently small $\mathcal{H}$ this operation is computationally feasible. For a large latent variable space $\mathcal{H}$, we have two options. First, we can choose the joint feature vector $\Phi(\mathbf{x}, \mathbf{y}, \mathbf{h})$ for the conditional distribution $\mathrm{P}_{\boldsymbol{\theta}}(\cdot)$ to be decomposable in such a manner as to facilitate efficient computation of sufficient statistics (for example, a low tree-width model). Note that this still allows us to use a more complex joint feature vector $\Psi(\mathbf{x}, \mathbf{y}, \mathbf{h})$ to make predictions for a given test sample. Second, if the problem requires a complex $\Phi(\mathbf{x}, \mathbf{y}, \mathbf{h})$ to encode the conditional distribution, then we can resort to using one of several inference techniques to compute the approximate sufficient statistics. However, we note that several important problems in machine learning can be formulated using latent variables whose space is sufficiently small to allow for exact computations of the expected loss, including motif finding [66], image classification (chapter 2), digit recognition (chapter 2), and the two problems used in our experiments, namely object detection and action detection.

4.5.2 Optimization over $\boldsymbol{\theta}$

For a fixed $\mathbf{w}$, problem (4.11) can be interpreted as a regularized upper bound on the following objective

$$\frac{1}{n}\left(\sum_{i=1}^{n} H_i(\mathbf{w},\boldsymbol{\theta}) - \beta H_i(\boldsymbol{\theta})\right), \tag{4.15}$$

where the divergence coefficients $H_i(\mathbf{w},\boldsymbol{\theta})$ and $H_i(\boldsymbol{\theta})$ are defined in equation (4.8). To gain an understanding of the above objective, let us consider a simple 0/1 loss (that is, the loss is 0 if both the outputs are equal and both the latent variables are equal, otherwise 1). If $\mathbf{y}_i(\mathbf{w}) = \mathbf{y}_i$, that is, $\mathbf{w}$ predicts the correct output for the sample $\mathbf{s}_i$, then the first term of the above objective dominates the second. In this case, the parameter $\boldsymbol{\theta}$ is encouraged to assign a high probability to the predicted latent variables $\mathbf{h}_i(\mathbf{w})$, and other similar latent variables, in order to minimize the objective. If $\mathbf{y}_i(\mathbf{w}) \neq \mathbf{y}_i$, the first term is a constant. Thus, the parameter $\boldsymbol{\theta}$ is encouraged to maximize the diversity of the conditional distribution $\mathrm{P}_{\boldsymbol{\theta}}(\cdot)$. In other words, for a correct prediction of output, we learn a *peaky* distribution and for an incorrect prediction of output, we learn a *flat* distribution. Formally, for a fixed $\mathbf{w}$, the optimization problem (4.11) reduces to the following:

$$\min_{\boldsymbol{\theta}} \frac{J}{2}||\boldsymbol{\theta}||^2 + CU(\mathbf{w},\boldsymbol{\theta}), \tag{4.16}$$

where $U(\mathbf{w},\boldsymbol{\theta})$ is defined in equation (4.10). We obtain an approximate solution to the above problem using stochastic subgradient descent (SSD). The main steps of SSD are outlined in Algorithm 8.

Algorithm 8 *The* SSD *algorithm for optimizing* $\boldsymbol{\theta}$.

input Dataset $\mathcal{D}$, initial estimate $\boldsymbol{\theta}_0$, $T > 0$.

1: $t \leftarrow 0$. $\lambda \leftarrow J/C$.
2: **repeat**
3: Choose a sample $\mathbf{s}_i$ randomly from $\mathcal{D}$.
4: Compute the stochastic subgradient $\mathbf{g}_t$ as

$$\mathbf{g}_t = \boldsymbol{\theta}_t + \nabla_{\boldsymbol{\theta}} H_i(\mathbf{w},\boldsymbol{\theta}) + \nabla_{\boldsymbol{\theta}} H_i(\boldsymbol{\theta}). \tag{4.17}$$

5: $t \leftarrow t + 1$.
6: Update $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \frac{1}{\lambda t}\mathbf{g}_t$.
7: **until** Number of iterations $t = T$.

Each iteration of SSD takes $O(|\mathcal{H}|^2)$ time (since the subgradient $\mathbf{g}_t$ requires a quadratic sum to compute $H_i(\boldsymbol{\theta})$). Similar to the expected loss, this can be performed exactly for a sufficiently small space of latent variables, or the appropriate choice of the joint feature vector $\Phi(\mathbf{x},\mathbf{y},\mathbf{h})$. For a large latent variable space and a complex joint feature vector, we would have to resort to approximate inference.

4.5.3 Comparison with ILSVM

Our overall approach is similar in flavor to the ILSVM algorithm [38], which iterates over the following two steps until convergence: (i) obtain the value of the latent variables for all training samples using the current estimate of the parameters; (ii) update the parameters by solving an latent SVM, where the loss function is measured using the latent variables estimated in the first step instead of the true latent variables. The following observation shows that ILSVM is a special case of our framework.

Observation 3. The first step of ILSVM minimizes the objective (4.15) when $\mathrm{P}_{\boldsymbol{\theta}}(\cdot)$ are restricted to be delta distributions. The second step of ILSVM solves an latent SVM problem similar to the one described in the previous subsection for optimizing over $\mathbf{w}$.

The observation regarding the second step is straightforward. For the first step, it follows from the fact that ILSVM minimizes $H_i(\mathbf{w}, \boldsymbol{\theta})$. As the second divergence coefficient $H_i(\boldsymbol{\theta})$ vanishes when using delta conditional distributions, ILSVM effectively minimizes objective (4.15) for a fixed $\mathbf{w}$. A formal proof is omitted.

4.6 Experiments

We now demonstrate the efficacy of our framework on two challenging machine learning applications: object detection and action detection. Specifically, we show how our approach, which models the uncertainty in the values of the latent variables during training, outperforms the previous loss-based learning frameworks, namely latent SVM and ILSVM, which only estimate the most likely assignment of the latent variables. All three methods used in our experiments share a common hyperparameter C (the relative weight for the upper bounds ξ_i), which we vary to take values from the set $\{10^{-4}, 10^{-3}, \cdots, 10^{2}\}$. In addition, our framework introduces two more hyperparameters: J (the relative weight for the regularization of $\boldsymbol{\theta}$) and β (the hyperparameter for Rao's dissimilarity coefficient). In all our experiments, we set $J = 0.1$ and $\beta = 0.1$. However, we may obtain better results by carefully tuning these hyperparameters. The tolerance value for all the methods was set to $\epsilon = 10^{-3}$.

4.6.1 Object Detection

Problem Formulation. The aim of this application is to learn discriminative object models that predict the category (for example, 'deer' or 'elephant') and the location of the object present in an image. In a fully supervised setting, we would be required to

specify a tight bounding box around the object present in each of the training samples. As the collection of such annotations is onerous and expensive, we would like to learn the object models using image-level labels (that is, labels indicating the presence or absence of an object category in an image), which are considerably easier to obtain. Formally, for each sample, the input $\mathbf{x}$ is an image. The output $\mathbf{y} \in \{0, 1, \cdots, c-1\}$, where c is the number of object categories. The latent variable $\mathbf{h}$ models the tight bounding box around the object in the image. Similar to previous chapters, the joint feature vectors $\Psi(\mathbf{x}, \mathbf{y}, \mathbf{h})$ and $\Phi(\mathbf{x}, \mathbf{y}, \mathbf{h})$ are defined using the HOG descriptor [13, 17] extracted using the pixels of the bounding box. In our experiments, we consider non-overlapping putative bounding boxes that are 8 pixels apart, which results in a maximum of 350 bounding boxes for each image in our dataset. This allows us to compute the exact expected loss and the exact subgradients during learning. We employ two different loss functions, 0/1 loss and overlap loss, which are defined below.

$$\Delta_{0/1}(\mathbf{y}_1, \mathbf{h}_1, \mathbf{y}_2, \mathbf{h}_2) = \begin{cases} 0 & \text{if } \mathbf{y}_1 = \mathbf{y}_2, \mathbf{h}_1 = \mathbf{h}_2, \\ 1 & \text{otherwise,} \end{cases}$$

$$\Delta_{O}(\mathbf{y}_1, \mathbf{h}_1, \mathbf{y}_2, \mathbf{h}_2) = \begin{cases} 1 - \mathrm{O}(\mathbf{h}_1, \mathbf{h}_2) & \text{if } \mathbf{y}_1 = \mathbf{y}_2, \\ 1 & \text{otherwise,} \end{cases}$$

where $O(\mathbf{h}_1, \mathbf{h}_2) \in [0, 1]$ is the ratio of the area of the intersection and the area of the union of the two bounding boxes [16]. Both the loss functions not only encourage the models to predict the right category but also the right location of the object. We note that a similar experimental setup was also used in [9].

Dataset. We use images of 6 different mammals (approximately 45 images per mammal) that have been employed in the previous chapters. We split the images of each category into approximately 60% for training and 40% for testing. We report results using 5 folds.

Results. Figure 4.1 shows the test loss for latent SVM, ILSVM and our method using the 7 different C values. The test loss is computed using the ground-truth labels and bounding boxes for the test samples. Recall that, during training, only the ground-truth labels were assumed to be known, while the bounding boxes were modeled as latent variables.

While latent SVM was initially proposed for loss functions that do not depend on the value of the true latent variable, we adopted a similar approach to the CCCP algorithm for latent SVM to solve the object detection problem. Briefly, we iterate over two steps: estimating the value of the latent variables and solving a convex structured SVM problem

until the objective function could not be decreased below a user-specified tolerance. In our experiments, this approach provided similar results to the ILSVM method.

By incorporating the uncertainty in latent variables, our approach outperformed both latent SVM and ILSVM. Specifically, for the 0/1 loss, the best test loss (over all C values) for latent SVM, ILSVM and our method is 64.82 ± 4.96, 68.53 ± 5.52 and 47.76 ± 2.53 respectively (where the loss has been scaled to lie between 0 and 100). For the overlap loss, the best test loss is 44.93 ± 1.84, 47.26 ± 3.87 and 42.27 ± 3.64 respectively. While the improvement in the overlap loss is not statistically significant according to paired t-test, the improvement in the 0/1 loss is statistically significant with $p < 10^{-4}$.

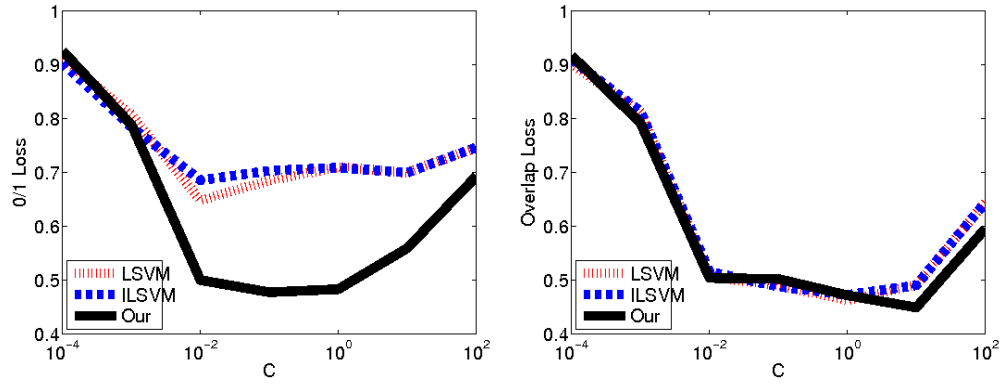


FIGURE 4.1: *The average test loss over five folds (y-axis) of the object detection application for different values of C (x-axis, shown in log-scale). Left: 0/1 loss; Right: Overlap loss. Our framework outperforms both latent* SVM *and* ILSVM *and provides statistically significant improvements for the 0/1 loss (see text for details).*

4.6.2 Action Detection

Problem Formulation. The aim of this application is to learn human action models that predict the action class (for example, 'running' or 'jumping') and the location of the person present in an image. Similar to object detection, a fully supervised dataset would require annotating each training image with the person bounding box. Instead, we use image-level labels that indicate which action is being performed by a person in the image. Formally, for each sample, the input $\mathbf{x}$ is an image. The output $\mathbf{y} \in \{0, 1, \cdots, c-1\}$, where c is the number of action classes. The latent variable $\mathbf{h}$ models the tight bounding box around the person in the image. The joint feature vectors are the Poselet descriptor [45] of the bounding box. We consider approximately 20 putative bounding boxes for each image, which are obtained automatically using a standard person detector [17]. The small search space for the latent variables avoids the need for approximate inference. Once again, we report results using both 0/1 loss and overlap loss.

Dataset. We use the PASCAL VOC 2011 'trainval' dataset [16], which consists of approximately 2500 images of 10 different action classes. We split the images of each class into approximately 60% for training and 40% for testing, and report results using 5 folds. In addition to the detected persons, we introduce the largest ground-truth bounding box into the latent variable space.

Results. Figure 4.2 shows the test loss for the three methods, computed using ground-truth labels and bounding boxes. For 0/1 loss, the best test loss over all C values for latent SVM, ILSVM and our method is 93.18 ± 1.95, 92.89 ± 3.70 and 76.10 ± 0.71 respectively. For overlap loss, the best test loss is 70.66 ± 0.76, 71.33 ± 1.14 and 67.16 ± 0.32 respectively. Our method significantly outperforms both latent SVM and ILSVM, as confirmed by the paired t-test with $p < 10^{-3}$.

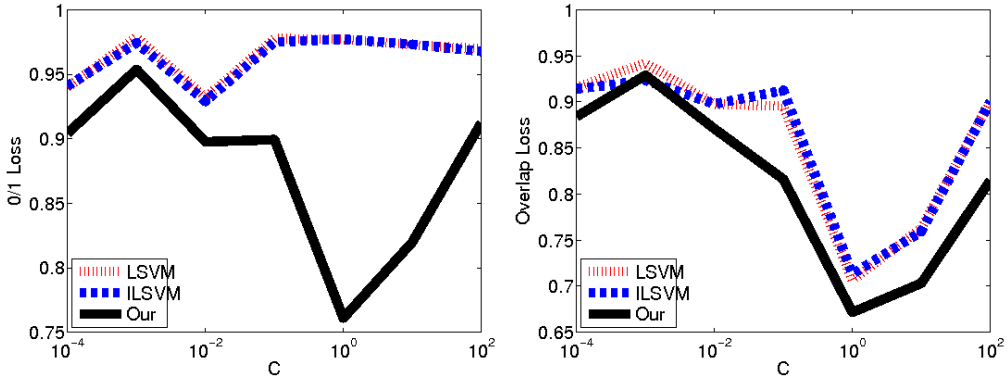


FIGURE 4.2: *The average test loss over five folds (y-axis) of the action detection application for different values of C (x-axis, shown in log-scale). Left: 0/1 loss; Right: Overlap loss. Our framework outperforms both latent* SVM *and* ILSVM *and provides statistically significant improvements for both types of loss (see text for details).*

4.7 Discussion

We proposed a novel framework for parameter estimation using weakly supervised datasets. Our framework consists of two distributions: a conditional distribution that captures the uncertainty in the latent variables, and a delta distribution that predicts the output and latent variable values. The parameters of the distributions are learned by minimizing a loss-based dissimilarity coefficient between the two distributions for all samples in the training dataset. We empirically demonstrate the benefit of our approach over previous loss-based learning frameworks using publicly available datasets of two challenging problems—object detection and action detection.

The proposed optimization requires the computation of the expected loss $\Delta_i(\mathbf{y}, \mathbf{h}|\boldsymbol{\theta})$ (shown in equation (4.7)) when learning the delta distribution and the loss-dependent

subgradient $\mathbf{g}_t$ (shown in equation (4.17)) when learning the conditional distribution. In special cases (for example, low tree-width models), these terms can be computed exactly. In general, we would have to resort to one of several existing approximate inference techniques or to design customized algorithms to compute the sufficient statistics. Note that, since the conditional distribution is not used during testing, an approximate estimate of its parameters, which is able to accurately model the uncertainty in the latent variables, would suffice in practice.

Chapter 5

Discussion

5.1 Conclusion

In this thesis, we considered the problem of learning the parameters of a structured output prediction model using weakly supervised datasets. The main advantage of weakly supervised learning is that it makes it possible to construct large datasets in a cost effective manner. However, it results in a far more challenging machine learning problem compared to fully supervised learning. In order to address these challenges in the context of empirical risk minimization, we presented the following three novel contributions.

- We proposed a novel iterative self-paced learning algorithm for solving the optimization problem corresponding to latent SVM, which builds on the intuition that the learner should be presented in the training samples in a meaningful order that facilitates learning: starting forme easy samples and gradually moving to harder samples. Our algorithm simultaneously selects the easy samples and updates the parameters at each iteration by solving a biconvex optimization problem.
- We proposed a new family of LVMs called max-margin min-entropy (M3E) models, which includes latent SVM as a special case. Given an input, an M3E model predicts the output with the smallest corresponding *Rényi entropy of generalized distribution* [56], which relies not only on the probability of the output but also the uncertainty of the latent variable values. Similar to latent SVM, the parameters of an M3E model are learned by minimizing a regularized upper bound on the empirical risk, which is measured using a loss function that is independent of the true value of the latent variables.

- We proposed a novel learning framework that simultaneously estimates two distributions: (i) a conditional distribution to model the uncertainty of the latent variables for a given input-output pair; and (ii) a delta distribution to predict the output and the latent variables for a given input. During learning, we encourage agreement between the two distributions by minimizing a loss-based dissimilarity coefficient [55]. Our approach generalizes latent SVM in two important ways: (i) it models the uncertainty over latent variables instead of relying on a pointwise estimate; and (ii) it allows the use of loss functions that depend on latent variables, which greatly increases its applicability.

We demonstrated the efficacy of our approaches on standard machine learning applications using publicly available datasets.

5.2 Future Work

We now discuss some interesting directions of future research.

Self-Paced Learning. In our past work, we have developed a simple version of self-paced learning for parameter estimation that gradually increases the complexity of the training samples. While it has obtained encouraging initial results, it fails to exploit the self-paced learning strategy to the fullest. We believe that the ability and applicability of self-paced learning can be greatly enhanced by introducing the idea of pacing the model complexity and the problem complexity within the learning framework. Specifically, in the initial iterations, the learner will solve simpler problems (for example, distinguish between a car and a deer) using a simple parameterization (such as those specified by linear kernels). In the later iterations, the learner will tackle more complex problems (for example, distinguish between a car and a van, or a deer and a dog) and solve them using more complex parameterizations (such as those specified by nonlinear kernels). The complexity of the samples, the model and the problem could be determined automatically by the learner such that it optimizes a suitable criterion for avoiding bad local minimum solutions (for example, the generalization bound of the learned parameters).

Active Learning. In order to reduce the ambiguity inherent in weakly supervised learning without substantially increasing the cost, it would be convenient to develop a unified framework for combining self-paced learning and active learning [12, 64], where a user can be queried to obtain the ground-truth value of latent variables. This would involve designing an appropriate criterion for selecting the samples to present to the

user in such a way that the number of queries required to obtain an accurate model is minimized. A successful unified framework should be able to capture the intuition that easy non-confusing samples should be used to update the learner (as specified by self-paced learning), while the hard confusing samples should be presented to the user for annotation.

Dynamic Dual Decomposition. The above extensions will result in a challenging, large-scale optimization problem. In order to make the learning framework practically useful, we would require efficient algorithms for solving the optimization problem. One potential solution to this would be to build on the decomposition techniques that allow us to make use of parallel computation in order to deal with large-scale problems [7, 22, 34–36]. In effect, decomposition methods can enable learning on large cloud-based architectures such as Azure.

A decomposition technique is an iterative algorithm, where each iteration consists of the following steps: (i) divide the original problem into a set of smaller subproblems; (ii) solve each subproblem independently; and (iii) pass messages among the subproblems in order to encourage them to agree on their optimal solution. One of the most important factors that determines the efficiency of dual decomposition is the specification of the subproblems. If the subproblems are defined in a way that they agree on the value of an optimization variable, then the entire procedure would converge quickly. In contrast, if two subproblems disagree, this will adversely affect convergence. Despite the importance of the decomposition, all current methods use the same subproblems through all iterations. Their performance could potentially be improved using a dynamic dual decomposition techniques that change the decomposition iteratively based on the disagreements between the subproblems of the current decomposition. Such an approach would need to take into account the efficiency of the individual subproblems, the inter-node communication cost for parallel computation, as well as the overhead cost of re-decomposition itself.

Bibliography

[1] E. Allgower and K. Georg. *Numerical continuation methods: An introduction*. Springer-Verlag, 1990.

[2] N. Bathmanghelich, B. Taskar, and C. Davatzikos. Generative-discriminative basis learning for medical imaging. *Transaction on Medical Imaging*, 2011.

[3] P.-Y. Baudin, D. Goodman, P. Kumar, N. Azzabou, P. G. Carlier, N. Paragios, and M. P. Kumar. Discriminative parameter estimation for random walks segmentation. In *MICCAI*, 2013.

[4] M. Bazaraa, H. Sherali, and C. Shetty. *Nonlinear Programming - Theory and Algorithms*. John Wiley and Sons, Inc., 1993.

[5] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *ICML*, 2009.

[6] M. Berger, G. Badis, A. Gehrke, and S. Talukder et al. Variation in homeodomain DNA binding revealed by high-resolution analysis of sequence preferences. *Cell*, 2008.

[7] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[8] C. Bishop. *Pattern recognition and machine learning*. Springer-Verlag New York, 2006.

[9] M. Blaschko, A. Vedaldi, and A. Zisserman. Simultaneous object detection and ranking with weak supervision. In *NIPS*, 2010.

[10] D. Blei and M. Jordan. Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 2006.

[11] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, 98.

[12] D. Cohn, Z. Ghahramani, and M. Jordan. Active learning with statistical models. *JAIR*, 4:129–145, 1996.

[13] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[14] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society*, 1977.

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarhical image database. In *CVPR*, 2009.

[16] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. *IJCV*, 2010.

[17] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.

[18] T. Finley and T. Joachims. Supervised clustering with support vector machines. In *ICML*, 2005.

[19] C. Floudas and V. Visweswaran. Primal-relaxed dual global optimization approach. *Journal of Optimization Theory and Applications*, 78(2):187–225, 1993.

[20] A. Gelman, J. Carlin, H. Stern, and D. Rubin. *Bayesian Data Analysis*. Chapman and Hall, 1995.

[21] S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. In *ICCV*, 2009.

[22] M. Guignard and S. Kim. Lagrangean decomposition: A mdoel yielding stronger lagrangean bounds. *Mathematical Programming*, 1987.

[23] J. Havrada and F. Charvat. Quantification method in classification processes: Concept of structural α-entropy. *Kybernetika*, 1967.

[24] G. Heitz, G. Elidan, B. Packer, and D. Koller. Shape-based object localization for descriptive classification. *IJCV*, 2009.

[25] G. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. In *NIPS*, 2006.

[26] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 2006.

[27] T. Jaakkola, M. Meila, and T. Jebara. Maximum entropy discrimination. In *NIPS*, 1999.

[28] J. Jancsary, S. Nowozin, T. Sharp, and C. Rother. Regression tree fields: An efficient, non-parameteric approach to image labeling problems. In *CVPR*, 2012.

[29] E. Jaynes. *Probability theory: The logic of science*. Cambridge University Press, 2003.

[30] T. Jebara. *Discriminative, generative and imitative learning*. PhD thesis, MIT, 2001.

[31] T. Jebara and T. Jaakkola. Feature selection and dualities in maximum entropy discrimination. In *UAI*, 2000.

[32] T. Joachims, T. Finley, and C.-N. Yu. Cutting-plane training for structural SVMs. *Machine Learning*, 2009.

[33] D. Koller and N. Friedman. *Probabilistic graphical models: Principles and techniques*. MIT Press, 2009.

[34] N. Komodakis. Efficient training for pairwise and higher order CRFs using dual decomposition. In *CVPR*, 2011.

[35] N. Komodakis and N. Paragios. Beyond pairwise energies: Efficient optimization of higher-order MRFs. In *CVPR*, 2009.

[36] N. Komodakis, N. Paragios, and G. Tziritas. MRF optimization via dual decomposition: Message-passing revisited. In *ICCV*, 2007.

[37] M. P. Kumar and D. Koller. Efficiently selecting regions for scene understanding. In *CVPR*, 2010.

[38] M. P. Kumar, H. Turki, D. Preston, and D. Koller. Learning specific-class segmentation from diverse data. In *ICCV*, 2011.

[39] G. Kundu, V. Srikumar, and D. Roth. Margin-based decomposed amortized inference. In *ACL*, 2013.

[40] L. Ladicky, C. Russell, P. Kohli, and P. Torr. Associative hierarchical CRFs for object class image segmentation. In *ICCV*, 2009.

[41] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[42] H. Lee, R. Grosse, R. Ranganath, and A. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, 2009.

[43] F. Li, J. Carreira, and C. Sminchisescu. Object recognition as ranking holistic figure-ground hypotheses. In *CVPR*, 2010.

[44] S. Li, J. Graca, and B. Taskar. Wiki-ly supervised part-of-speech tagging. In *EMNLP*, 2012.

[45] S. Maji, L. Bourdev, and J. Malik. Action recognition from a distributed representation of pose and appearance. In *CVPR*, 2011.

[46] A. Mathai and P. Rathie. *Basic Concepts in Information Theory and Statistics*. Wiley (Halsted Press), New York, 1974.

[47] K. Miller, M. P. Kumar, B. Packer, D. Goodman, and D. Koller. Max-margin min-entropy models. In *AISTATS*, 2012.

[48] T. Minka and Z. Ghahramani. Expectation propagation for infinite mixtures. In *NIPS Workshop on Nonparametric Bayesian Methods and Infinite Models*, 2003.

[49] R. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 2000.

[50] R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. Jordan, editor, *Learning in Graphical Models*. MIT Press, 1999.

[51] V. Ng and C. Cardie. Improving machine learning approaches to coreference resolution. In *ACL*, 2002.

[52] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *CIKM*, 2000.

[53] S. Nowozin and C. Lampert. Structured learning and prediction in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 2010.

[54] L. Pishchulin, A. Jain, M. Andriluka, and B. Thormaehlen, Schiele. Articulated people detection and pose estimation: Reshaping the future. In *CVPR*, 2012.

[55] C. Rao. Diversity and dissimilarity coefficients: A unified approach. *Theoretical Population Biology*, 1982.

[56] A. Renyi. On measures of information and entropy. In *Berkeley Symposium on Mathematics, Statistics and Probability*, 1961.

[57] J. Salojarvi, K. Puolamaki, and S. Kaski. Expectation maximization algorithms for conditional likelihoods. In *ICML*, 2005.

[58] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated subgradient solver for SVM. In *ICML*, 2009.

[59] P. Simard, B. Victorri, Y. LeCun, and J. Denker. Tangent Prop - a formalism for specifying selected invariances in adaptive network. In *NIPS*, 1991.

[60] A. Smola, S. .V. .N. Vishwanathan, and T. Hofmann. Kernel methods for missing variables. In *AISTATS*, 2005.

[61] B. Sriperumbudur and G. Lanckriet. On the convergence of concave-convex procedure. In *NIPS Workshop on Optimization for Machine Learning*, 2009.

[62] R. Sundberg. Maximum likelihood theory for incomplete data from an exponential family. *Scandinavian Journal of Statistics*, 1974.

[63] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *NIPS*, 2003.

[64] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *JMLR*, 2:45–66, 2001.

[65] I. Tsochantaridis, T. Hofmann, Y. Altun, and T. Joachims. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.

[66] C.-N. Yu and T. Joachims. Learning structural SVMs with latent variables. In *ICML*, 2009.

[67] A. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 2003.

[68] W. Zaremba, M. P. Kumar, A. Gramfort, and M. Blaschko. Learning from M/EEG data with variable brain activation delay. In *IPMI*, 2013.

[69] K. Zhang, I. Tsang, and J. Kwok. Maximum margin clustering made practical. In *ICML*, 2007.