Détection d'anomalies à la volée dans des flux de données de grande dimension

Anastasios Bellas

UNIVERSITÉ PARIS 1 - PANTHÉON-SORBONNE

THÈSE

présentée par

Anastasios BELLAS

pour obtenir le grade de

DOCTEUR EN MATHÉMATIQUES APPLIQUÉES

de l'Université Paris 1 - Panthéon-Sorbonne

Détection d'anomalies dans des flux de données de grande dimension

---

réalisée sous la direction de Marie COTTRELL
et Charles BOUVEYRON

JURY

Michel	VERLEYSEN	Professeur	Rapporteur
Julien	JACQUES	Maître de Conférences - HDR	Rapporteur
Jean-Bernard	BAILLON	Professeur	Président
Julie	JOSSE	Maître de Conférences	Examinateur
Étienne	CÔME	Chargé de Recherche	Examinateur
Jérôme	LACAILLE	Expert Algorithmique	Examinateur
Marie	COTTRELL	Professeur	Directeur
Charles	BOUVEYRON	Professeur	Directeur

À mon père

Abstract

The subject of this Thesis is to study anomaly detection in high-dimensional data streams with a specific application to aircraft engine Health Monitoring. In this work, we consider the problem of anomaly detection as an unsupervised learning problem. Modern data, especially those issued from industrial systems, are often streams of high-dimensional data samples, since multiple measurements can be taken at a high frequency and at a possibly infinite time horizon. Moreover, data can contain anomalies (malfunctions, failures) of the system being monitored. Most existing unsupervised learning methods cannot handle data which possess these features. We first introduce an offline subspace clustering algorithm for high-dimensional data based on the expectation-maximization (EM) algorithm, which is also robust to anomalies through the use of the trimming technique. We then address the problem of online clustering of high-dimensional data streams by developing an online inference algorithm for the popular mixture of probabilistic principal component analyzers (MPPCA) model. We show the efficiency of both methods on synthetic and real datasets, including aircraft engine data with anomalies. Finally, we develop a comprehensive application for the aircraft engine Health Monitoring domain, which aims at detecting anomalies in aircraft engine data in a dynamic manner and introduces novel anomaly detection visualization techniques based on Self-Organizing Maps. Detection results are presented and anomaly identification is also discussed.

Keywords : Classification, anomaly detection, high-dimensional data, data streams, trimming, online clustering, online mixture of PPCA, Self-Organizing Maps, aircraft engine, Health Monitoring.

Résumé

Le thème principal de cette thèse est d'étudier la détection d'anomalies dans des flux de données de grande dimension avec une application spécifique au *Health Monitoring* des moteurs d'avion. Dans ce travail, on considère que le problème de la détection d'anomalies est un problème d'apprentissage non supervisée. Les données modernes, notamment celles issues de la surveillance des systèmes industriels sont souvent des flux d'observations de grande dimension, puisque plusieurs mesures sont prises à de hautes fréquences et à un horizon de temps qui peut être infini. De plus, les données peuvent contenir des anomalies (pannes) du système surveillé. La plupart des algorithmes existants ne peuvent pas traiter des données qui ont ces caractéristiques. Nous introduisons d'abord un algorithme de clustering probabiliste offline dans des sous-espaces pour des données de grande dimension qui repose sur l'algorithme d'espérance-maximisation (EM) et qui est, en plus, robuste aux anomalies grâce à la technique du trimming. Ensuite, nous nous intéressons à la question du clustering probabiliste online de flux de données de grande dimension en développant l'inférence online du modèle de mélange d'analyse en composantes principales probabiliste. Pour les deux méthodes proposées, nous montrons leur efficacité sur des données simulées et réelles, issues par exemple des moteurs d'avion. Enfin, nous développons une application intégrée pour le Health Monitoring des moteurs d'avion dans le but de détecter des anomalies de façon dynamique. Le système proposé introduit des techniques originales de détection et de visualisation d'anomalies reposant sur les cartes auto-organisatrices. Des résultats de détection sont présentés et la question de l'identification des anomalies est aussi discutée.

Mots-clefs : Classification, détection d'anomalies, données de grande dimension, flux de données, trimming, clustering online, mélange de PPCA online, cartes auto-organisatrices, moteurs d'avions, Health Monitoring.

Acknowledgements

Je voudrais tout d'abord remercier mes deux directeurs de thèse, Charles Bouveyron et Marie Cottrell pour leur persistance et leur sens de pédagogie. Sans eux, cette thèse n'aurait pas pu être achevée en 3 ans.

Par ailleurs, je voudrais remercier les professeurs Michel Verleysen et Julien Jacques qui ont accepté de rapporter mon manuscrit de thèse et qui ont fait des remarques très intéressantes sur mon travail.

Je tiens aussi à remercier les professeurs Jean-Bernard Baillon, Julie Josse et Étienne Côme pour avoir accepté d'être dans mon jury de thèse.

Je voudrais aussi remercier Jérôme Lacaille, l'expert algorithmique de Snecma pour sa contribution à mieux orienter notre travail, Serge Blanchard, chef du Pôle Monitoring de l'entreprise ainsi que toute l'équipe du Pôle pour leur aide précieuse.

Enfin, et ce n'est pas des moindres, je voudrais remercier tous les collègues du SAMM, les doctorants en particulier, pour avoir crée une ambiance superbe au sein du labo. Ma famille, mes amis grecs, français et autres ont largement contribué à m'amuser, me divertir et mener à bien tout ce travail volumineux ces 3 dernières années. De toute façon, quelques lignes écrites ne suffiraient pas pour les remercier et j'aurai l'occasion de le faire de près.

Contents

Chapter 1

Introduction

Statistical Learning, also known as Machine Learning, is a research domain situated on the frontier between Mathematics, in particular, Statistics, and Computer Science. The goal is to automate procedures that human beings perform in order to take complex decisions. For example, e-mail spam detection is currently carried out by Machine Learning algorithms which, informally, *learn* to distinguish between an ordinary and a spam message based on samples of both. An example of a more complex task is the monitoring of critical industrial systems or components, such as aircraft engine. Malfunction detection in modern aircraft engines is being carried out by Statistical Learning systems.

More of a theoretically posed problem in the 70's and the 80's, Statistical Learning has found a widespread use in practical applications in the last 20 years, driven by developments in the field but also in computational power and resources.

1.1 Anomaly detection in high-dimensional datastreams

The goal of this Thesis is to study anomaly detection in high-dimensional datastreams with a specific application to aircraft engine health monitoring data. In this work, we consider the problem of anomaly detection as an unsupervised learning problem.

Due to advances in measurement technology (for instance, networks of remote sensors), data issued from real-world domains often have a large number of attributes, *i.e.* they are high-dimensional. As we will see, this can cause well-established methods to fail due to a phenomenon known under the name of the *curse of dimensionality*. Moreover, when real-world phenomena or procedures are being monitored, data are arriving at a constant rate, at a possibly infinite time horizon. Unfortunately, most existing methods have been designed for use in a static context, where learning is carried out on a fixed dataset.

In addition, data can contain anomalies, *i.e.* atypical data samples. For instance, in the monitoring of industrial systems, such data samples can correspond to a malfunction or damage of the system. Most general unsupervised classification methods have been designed with the implicit assumption of data clean of anomalies. For many of these

methods, one single anomaly would be enough to arbitrarily corrupt their results.

1.2 Contributions

This Thesis makes the following contributions: the introduction of a novel robust unsupervised classifier for high-dimensional data, the proposal of a novel algorithm for clustering high-dimensional datastreams and an integrated system for processing aircraft engine health monitoring data and detecting anomalies therein.

Robust unsupervised classifier for high-dimensional data

We develop an offline unsupervised classifier for high-dimensional data which is robust to anomalies. We use a subspace clustering algorithm to cope with dimensionality issues. This type of methods cluster the data in lower-dimensional subspaces, based on the fact that high-dimensional data often live in lower-dimensional subspaces. In order to make the classifier robust to anomalies, we plug in the well-established trimming technique, which trims the less likely data samples from the dataset. The goal is to "clean" the dataset of any anomalous data samples so that the clustering will not be corrupted.

The online Mixture of Probabilistic PCA for high-dimensional online clustering

The well-known Mixture of Probabilistic PCA (MPPCA) model is a probabilistic subspace clustering method. This means that the model can cope with high-dimensional datasets but not datastreams, since it is a batch method. In this Thesis, we develop the online Mixture of Probabilistic PCA (MPPCA). A number of works has already proposed various methods to perform incremental Principal Component Analysis in a non-probabilistic setting. We extend an existing incremental PCA method to the probabilistic setting. This allows us to develop online MPPCA, a datastream subspace clustering algorithm, which can efficiently cluster high-dimensional datastreams.

Anomaly detection in aircraft engine Health Monitoring data

We develop an integrated anomaly detection system for aircraft engine data. Probabilistic subspace clustering is used to cluster data into groups following environmental conditions. A simple linear model per class is used to correct data from environmental influence. Self-Organizing Maps are then used to detect and visualize anomalies. Preliminary work on anomaly identification is also presented.

1.3 Applications

Anomaly detection methods can be applied to numerous fields including control of critical industrial systems, medical diagnosis, network intrusion detection and security. In this Thesis we apply the proposed methods to the following domains.

Aircraft engine Health Monitoring

Snecma, the french aircraft engine constructor, monitors the condition of its engines either in a test cell environment or by analyzing, post-flight, data collected during the flight. Data are collected using multiple sensors placed on the test cell, the aircraft or the engine itself. In addition, they are often composed of a large number of attributes. We show the capacity of the proposed methods to detect abnormal data in high-dimensional aircraft engine data and we also conduct experiments on visualization and identification of anomalies in this data.

Breast cancer diagnosis

We apply one of the proposed methods to the UCI breast cancer dataset, containing measurements from patients with tumors, both benign and malignant. The goal is to make a diagnosis, that is, classify correctly the patients based on whether they have benign or malignant tumors. This is essentially a classification task and supervised methods have mostly been employed to carry it out. We illustrate the use of the robust offline unsupervised classifier as an one-class robust classifier in a binary classification problem.

1.4 Organisation of the Thesis

In chapter 3, we present a large survey which is divided into two major parts: unsupervised approaches for Anomaly Detection and unsupervised online clustering. As a sort of introduction, we first give basic notions of classification and probabilistic clustering and we discuss such issues as theoretical principles, inference and model selection for probabilistic clustering.

Chapter 4 focuses on the problem of clustering high-dimensional data with anomalies. Existing methods for subspace clustering are presented and an introduction to robust clustering with trimming (clustering data with anomalies) is given by discussing of recent advances in the field. A novel method is then proposed to cluster high-dimensional data with anomalies. The proposed *High-Dimensional Robust Clustering* (HDRC) algorithm combines the subspace clustering approach and trimming to carry out this task. We also present an application of HDRC to the UCI breast cancer dataset.

In chapter 5, we propose an online inference algorithm for the mixture of probabilistic PCA model for clustering high-dimensional datastreams. The probabilistic PPCA model is a subspace clustering model and can, thus, handle high-dimensional data efficiently. The proposed algorithm relies on an EM-based procedure and on a probabilistic and incremental version of PCA. Model selection is also considered in the online setting through parallel computing. We run numerical experiments on simulated and real data and demonstrate the effectiveness of our approach by comparing it with state-of-the-art online EM-based algorithms.

In chapter 6, we develop an application of the proposed methods for the task of anomaly detection and visualization in aircraft engine Health Monitoring data issued

from real flights. Our system is composed of many components, including a classification component, an anomaly detection component and a visualization stage. The goal is to develop a comprehensive system that would be able to detect anomalies dynamically, providing the experts with tools to monitor them. As an introduction to the subject, we first give a review of related articles.

Finally, chapter 7 discusses the conclusions of our research and experimental work as well as directions for future work.

Chapter 2

Introduction

L'*apprentissage statistique*, ou Machine Learning, est un domaine de recherche qui se situe sur la frontière entre les Mathématiques, la Statistique en particulier, et l'Informatique. Le but est d'automatiser des procédures que l'humain réalise au quotidien pour prendre des décisions complexes. Par exemple, la détection de messages non sollicités (spam) se fait aujourd'hui par le biais d'algorithmes d'apprentissage statistique, qui *apprennent* à distinguer entre des messages ordinaires et non sollicités en utilisant des échantillons de deux sortes. Un exemple d'une tâche plus complexe est la surveillance des systèmes critiques industriels ou de leurs composantes, comme les moteurs d'avion. La détection d'anomalies dans les moteurs d'avion modernes se fait par le biais des algorithmes d'apprentissage statistique.

Dans les années '70 et '80, il s'agissait de travaux théoriques, mais dans les 20 dernières années, l'apprentissage statistique a trouvé de nombreux champs d'application, poussée par les avancées dans le domaine mais aussi le progrès réalisé en matière de ressources computationnelles et de puissance de calcul.

2.1 Détection d'anomalies dans des flux de données de grande dimension

Le thème principal de cette thèse est d'étudier la détection d'anomalies dans des flux de données de grande dimension avec une application spécifique au *Health Monitoring* des moteurs d'avion. Dans ce travail, on considère que le problème de la détection d'anomalies est un problème d'apprentissage non supervisée.

Grâce aux avancées dans la technologie des capteurs (par exemple, des réseaux de capteurs distants), les données issues d'applications réelles ont souvent un grand nombre de variables, *i.e.* elles sont de grande dimension. On verra que ceci peut faire échouer des méthodes bien établies à cause du phénomène connu sous le nom de *fléau de la dimension*.

De plus, quand des systèmes réels sont surveillés, les données arrivent de façon dynamique dans un horizon temporel qui peut être infini. Les données arrivent donc en

flux, alors que la plupart des méthodes existantes sont conçues pour être utilisées dans un contexte statique, où l'apprentissage se fait sur un ensemble de données qui est fixe et disponible tout au long de l'expérience.

Enfin, les données peuvent contenir des anomalies,*i.e.* des observations atypiques. Par exemple, dans la surveillance des systèmes industriels, ces observations peuvent indiquer l'existence d'une panne. La plupart des algorithmes d'apprentissage non supervisée génériques ont été conçus avec l'hypothèse implicite des données sans anomalies. Pour plusieurs d'entre eux, une seule anomalie suffirait pour affecter de façon négative leurs résultats.

2.2 Contributions

Cette thèse apporte les contributions suivants: l'introduction d'un nouveau classifieur robuste et non supervisé pour des données de grande dimension, la proposition d'un nouvel algorithme pour l'inférence online du modèle de mélanges de PCA probabiliste pour faire du clustering des flux de données de grande dimension et un système intégré pour le traitement des données de moteurs d'avion et la détection d'anomalies.

Classifieur non supervisé robuste pour des données de grande dimension

Nous développons un classifieur non supervisé et offline pour des données de grande dimension qui est, en plus, robuste aux anomalies. Nous utilisons un algorithme de clustering dans des sous-espaces pour faire face aux problèmes liés à la grande dimension. Ces méthodes font un clustering des données dans des sous-espaces de faible dimension, basés sur le fait que les données de grande dimension vivent souvent dans des sous-espaces de plus faible dimension. Pour rendre le classifieur robuste aux anomalies, nous rajoutons la technique populaire de *trimming* (rogner ou tronquer), qui tronque les données les moins vraisemblables. Le but est de "nettoyer" les données des observations atypiques avant le clustering, pour que ce dernier n'en soit pas corrompu.

Le mélange de PCA probabiliste online

Le modèle populaire de mélange de PCA probabiliste (MPPCA) est une méthode de clustering probabiliste dans des sous-espaces. Donc, il peut traiter des données de grande dimension mais pas des flux de données, puisqu'il s'agit d'une méthode offline. Dans cette thèse, nous développons le MPPCA online. Plusieurs travaux ont déjà proposé des méthodes pour faire du PCA incrémental dans un contexte non probabiliste. Nous adaptons une méthode de PCA existante dans le contexte probabiliste, ce qui nous permet de développer un algorithme qui peut faire du clustering de flux de données de grande dimension.

Détection d'anomalies dans des données de moteurs d'avion

Nous développons un système intégré pour détecter des anomalies dans des données de moteurs d'avion. Une technique de clustering probabiliste est utilisée pour classi-

fier les données dans des classes d'environnement. Un modèle linéaire par classe est utilisé pour corriger les données de l'influence de l'environnement. Enfin, des cartes auto-organisatrices permettent de détecter et de visualiser les anomalies. Des résultats préliminaires sur l'identification d'anomalies sont aussi présentés.

2.3 Applications

La détection d'anomalies trouve des applications dans de nombreux domaines comme la surveillance des systèmes industriels, le diagnostic médical, la sécurité des réseaux et des systèmes. Dans cette thèse, nous appliquons les méthodes proposées aux domaines suivants.

Données de moteurs d'avion

Snecma, le constructeur français de moteurs d'avion, surveille l'état de ses moteurs, soit dans un environnement de banc d'essai, soit en analysant, après un vol, des données qui sont recueillies pendant le vol. Les données sont collectées par le biais de plusieurs capteurs qui sont placés sur le banc, l'avion ou le moteur lui-même. De plus, ces données ont souvent autant de variables qu'il y a des capteurs et même plus. Nous montrons la capacité des méthodes proposées à détecter des anomalies dans des données de grande dimension provenant de moteurs d'avion et nous menons, de plus, des expériences sur la visualisation et l'identification d'anomalies dans ce genre de données.

Diagnostic du cancer du sein

Nous appliquons les méthodes proposées aux données UCI sur le cancer du sein, composées de diverses mesures sur des patients ayant des tumeurs, bénignes ou malignes. Le but est de faire un diagnostic, à savoir de classer correctement les patients suivant qu'ils ont des tumeurs bénignes ou malignes. A priori, il s'agit d'une tâche de classification et des méthodes supervisées ont été utilisées pour la traiter. Nous illustrons l'utilisation du classifieur non supervisé et robuste en tant que classifieur robuste 'one-class' dans un problème de classification binaire.

2.4 Organisation de la thèse

Dans le chapitre 3, nous présentons une recherche bibliographique détaillée en deux parties: la première partie fait référence à des méthodes non supervisées pour la détection d'anomalies et la seconde présente des approches de clustering online pour des flux de données. Comme introduction, nous donnons d'abord des notions de base sur la classification et le clustering probabiliste et discutons des questions telles que les principes théoriques, l'inférence et la sélection de modèles.

Dans le chapitre 4, nous considérons le problème de clustering de données de grande dimension avec des anomalies. Des méthodes existantes de clustering dans des sous-espaces sont présentées et une introduction au clustering robuste avec trimming est donnée avec

une discussion sur les avancées récentes dans le domaine. Ensuite, une nouvelle méthode est proposée pour le clustering des données de grande dimension avec des anomalies. La méthode proposée, appelée *High-Dimensional Robust Clustering* (HDRC) combine une approche de clustering probabiliste dans des sous-espaces avec le trimming pour faire du clustering de ce genre de données.

Dans le chapitre 5, nous proposons un algorithme d'inférence *online* pour le modèle de mélange de PCA probabiliste (MPPCA) dans le but de faire du clustering des flux de données de grande dimension. Le modèle de MPPCA est une approche de clustering dans des sous-espaces et il peut donc traiter des données de grande dimension. L'algorithme proposé repose sur l'algorithme EM et sur une version incrémentale et probabiliste de PCA. La sélection de modèles est considérée au moyen du calcul parallèle. Nous menons des expériences numériques sur des données simulées et réelles et nous illustrons l'efficacité de notre approche en la comparant à d'autres algorithmes basés sur EM.

Dans le chapitre 6, nous développons une application des méthodes proposées pour la tâche de détection et visualisation d'anomalies dans les données de moteurs d'avion issues de vols réels. Notre système est composé de plusieurs modules: classification, détection d'anomalies et visualisation. Le but a été de développer un système intégré pour détecter les anomalies de manière dynamique et fournir aux experts des outils pour surveiller ces données atypiques. Nous donnons d'abord une petite introduction aux travaux similaires.

Enfin, le chapitre 7 conclut cette thèse et discute de perspectives de recherche.

Chapter 3

Related work

In this chapter we give a broad overview of the topic of this Thesis. The goal is to introduce basic notions of the areas that the topic of this Thesis encompasses and present existing works related to the Thesis' topic. With this aim, the rest of the chapter is divided in three parts: the first part contains an introduction to clustering, the second presents existing works on Anomaly Detection, whereas the third refers to existing works on estimation issues and probabilistic clustering algorithms for the online setting, where streams of data arrive in a possible infinite horizon. The two latter parts reflect the two aspects of the problem that this Thesis tries to solve: adaptive, real-time anomaly detection.

3.1 Clustering

In this section, we introduce the task of clustering, in particular, probabilistic clustering. We discuss various aspects of the problem including inference and model selection for probabilistic clustering methods.

3.1.1 Probabilistic clustering

The task of classification consists in organizing data into classes, *i.e.* groups of data sharing common features in a certain sense. When we dispose of the data and the class information, then we speak of *supervised* classification. When the class information is missing, we refer to unsupervised classification or *clustering*. The methods proposed in this work are all clustering methods.

Clustering can be cast into a probabilistic framework with the aim of taking advantage of the fact that the probability theory is well-founded and in particular, the fact that probabilities can be used to express uncertainty over a decision in a natural way. We then refer to probabilistic or *model-based* clustering. We present this probabilistic framework in detail in the following paragraphs.

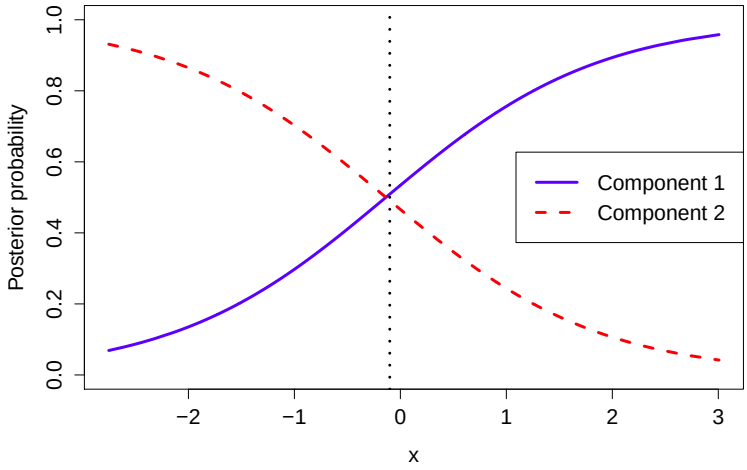


Figure 3.1: The *maximum a posteriori* rule for a one-dimensional mixture of two components. See the text for more details.

3.1.2 Model-based clustering and inference

We assume that data $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathbb{R}^p$ are independent realizations of a random vector $X \in \mathbb{R}^p$ whereas the class labels $\{z_1, ..., z_n\}$ are assumed to be independent realizations of an unobserved (latent) random variable Z with $Z(\Omega) = \{1, ..., K\}$. As we have already seen, clustering is trying to organize data into homogeneous groups. We can, however, deduce a partition of the dataset from the posterior probability of the classes using the following rule

$$\arg \max_{k=1,...K} P(Z = k \mid X = \mathbf{x}), \tag{3.1}$$

which is called MAP rule, for *maximum a posteriori*. Note that the posterior probability $P(Z = k \mid X = \mathbf{x})$ is not given and has to be calculated by the data. We illustrate it in Figure 3.1 for a one-dimensional mixture with 2 components. The blue and red curves correspond to the posterior probability that a data sample x belongs to class 1 and to class 2, respectively. As we can see, for $x < 0$, data samples will be affected to class 2, since the corresponding probability is higher and vice versa for $x > 0$.

Based on the above, we want to cluster $\mathcal{X}$ into K homogeneous groups, *i.e.* determine for each data sample $\mathbf{x}_i$ the value of its unobserved label z_i such that $z_i = k$ if $\mathbf{x}_i$ belongs to the k-th cluster. To this extent, *model-based clustering* (Fraley and Raftery, 2002; McLachlan and Peel, 2000) considers the overall population as a mixture of the groups and each component of this mixture is modeled through its conditional probability distribution. The set of pairs $\{(\mathbf{x}_i, z_i)\}_{i=1}^n$ is usually referred to as the complete data set. By denoting $f(\cdot)$ the probabilistic density function of X, the density of the finite

mixture model is:

$$f(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \phi_k(\mathbf{x}),$$

where π_k (such that $\pi_k \geq 0$ and $\sum_{k=1}^{K} \pi_k = 1$) and ϕ_k represent the mixture proportion and the conditional density function of the k-th mixture component, respectively. Furthermore, the clusters are often modeled by the same parametric density function (with different parameters). In this case, the density of the finite mixture model is:

$$f(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \phi(\mathbf{x}, \theta_k),$$

where θ_k is the parameter vector for the k-th mixture component. Among the possible probability distributions for the mixture components, the Gaussian distribution is certainly the one most frequently used for both theoretical and computational reasons. Moreover, it is a fairly "realistic" distribution, since it is has been found that it can model various real-world phenomena. In this case the density ϕ is supposed to be Gaussian $\mathcal{N}(\boldsymbol{\mu}_k, \Sigma_k)$ of mean $\boldsymbol{\mu}_k$ and covariance matrix Σ_k

$$\phi(\mathbf{x}, \theta_k) \quad = \quad \frac{1}{(2\pi)^{p/2}(\det \Sigma_k)^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \Sigma_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k)\right)$$

Here, the parameter vector θ_k contains the group mean vector $\boldsymbol{\mu}_k$ and the group covariance matrix Σ_k. This specific mixture model is usually referred to in the literature as the Gaussian mixture model (GMM). Figure 3.2 gives an example of a univariate Gaussian mixture model with two mixture components. The density of the components is given by the two dashed curves, while the density of the mixture is given by the solid curve.

Parametric mixture models and the MAP rule

In the context of parametric mixture models, Bayes' theorem allows us to write

$$P(Z = k \mid X = \mathbf{x}, \theta) = \frac{\pi_k \phi(\mathbf{x}, \theta_k)}{\sum_{k=1}^{K} \pi_k \phi(\mathbf{x}, \theta_k)}$$

We can then write the decision rule of (3.1) as

$$\arg \max_{k=1,\ldots,K} \{\pi_k \phi(\mathbf{x}, \theta_k)\}$$

As we can see, in order to get a partition of the dataset in the context of parametric mixture models, we simply have to estimate the parameters of the model.

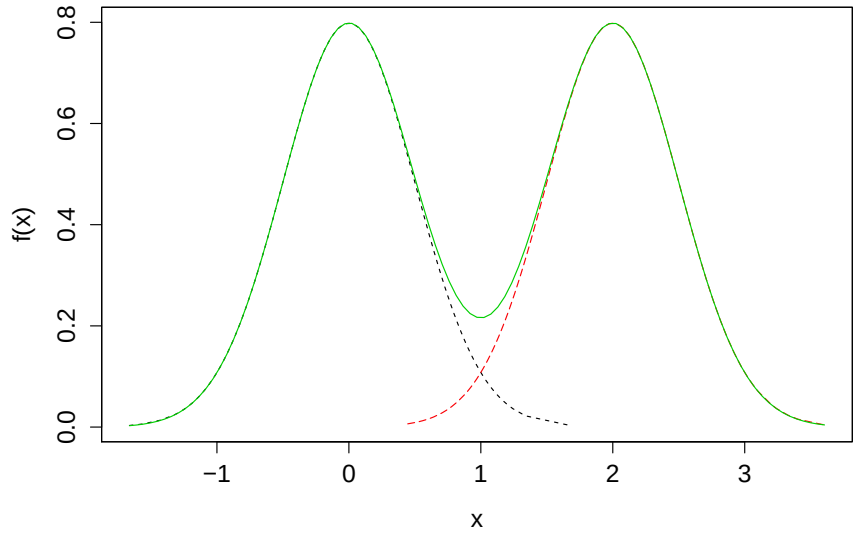


Figure 3.2: An example of a univariate Gaussian mixture model with two mixture components. The density of the components is given by the two dashed curves, while the density of the mixture is given by the solid curve.

Families of submodels derived from the most general model

In model-based clustering, one can derive different families of submodels from the most general model by imposing constraints on the covariance matrix and other parameters of the model. For example, we derive the submodel of common class covariance matrices by setting that they be all equal. Submodels can also be considered as *forms* of the (most general) model (see the paragraph on model selection in model-based clustering later on in this chapter for more on this).

3.1.3 Parameter estimation of a Gaussian mixture model via EM

We will now introduce the *maximum* likelihood principle for estimating parameters of statistical models. As we will see, this principle cannot be applied in the context of model-based clustering, since the class labels Z are unknown and are considered as *missing* data. In this context, we speak of *incomplete* data, since we only observe X. We present below the popular *expectation-maximization* (EM) algorithm (Dempster et al., 1977) that estimates parameters in the presence of incomplete data.

Estimation by the maximum likelihood principle

A standard way to estimate model parameters in parametric mixture models is to maximize the (observed) log-likelihood of the data

$$L(\mathbf{x};\theta) \quad = \quad \sum_{i=1}^{n} \log \sum_{k=1}^{K} \pi_k \phi(\mathbf{x}_i, \theta_k)$$

Note that we prefer the log-likelihood over the likelihood, as it is much more convenient to work with the former from a mathematical point of view. The maximum likelihood method then proposes to estimate the parameters of the model θ by $\hat{\theta}_{MV}$:

$$\hat{\theta}_{MV} = \arg\max_{\theta} L(\mathbf{x};\theta)$$

As we saw earlier, complete data $\{(\mathbf{x}_1, z_1), (\mathbf{x}_2, z_2), \ldots, (\mathbf{x}_n, z_n)\}$ are composed of pairs of data $\mathbf{x}$ and class information z. The *complete log-likelihood* $L_c(\mathbf{x}, z;\theta)$ is the log-likelihood calculated from the complete data:

$$L_c(\mathbf{x}, z;\theta) \quad = \quad \sum_{i=1}^{n} \sum_{k=1}^{K} s_{ik} \log\left(\pi_k \left(\phi(\mathbf{x}_i, \theta_k)\right)\right)$$

Here, we have defined s as the indicator variable of the classes, so that if $z_i = k$ for a data sample i, then $s_{ik} = 1$ and $s_{ij} = 0, \forall j \neq k$.

Unfortunately, inference cannot be done in a straightforward manner by maximizing the likelihood, since the class labels $\{z_1, ..., z_n\}$ are unknown. To overcome this problem, we can use the popular EM algorithm.

Estimation via Expectation-Maximization

The main idea behind the expectation-maximization (EM) algorithm is that it is easier to maximize the complete log-likelihood $L_c(\mathbf{x}, z;\theta)$, a conclusion based on the following relationship between the complete log-likelihood $L_c(\mathbf{x}, z;\theta)$ and the (observed) log-likelihood $L(\mathbf{x};\theta)$

$$L(\mathbf{x};\theta) = L_c(\mathbf{x}, z;\theta) - \log f(z;\mathbf{x},\theta)$$

However, as mentioned earlier, the class labels are unknown, so we cannot maximize directly the complete likelihood. However, we can maximize its conditional expectation $Q(\theta, \theta')$ given the data $\mathbf{x}$ and the current parameter estimate θ'

$$Q(\theta, \theta') = E\left[L_c(\mathbf{x}, z;\theta) \mid \mathbf{x}, \theta'\right]$$

By noting $t_{ik} = E[Z = k \mid X = \mathbf{x}_i, \theta']$, we can write

$$Q(\theta, \theta') = \sum_{i=1}^{n} \sum_{k=1}^{K} t_{ik} \log(\pi_k \phi(\mathbf{x}_i, \theta_k)) \tag{3.2}$$

The EM algorithm is maximizing iteratively $Q(\theta, \theta')$ in order to obtain the optimal parameter estimate.A detailed view of the EM algorithm for GMMs is given in Algorithm 1. From an initial solution $\theta^{(0)}$, the EM algorithm alternates two steps until convergence: the E-step and the M-step.

At iteration q, the expectation step (E-step) computes the expectation of the complete log-likelihood $E\left[L_c\left(\mathbf{x}, z; \theta\right) | \mathbf{x}, \hat{\theta}^{(q-1)}\right]$ conditionally to the current value of the parameter set $\theta' = \hat{\theta}^{(q-1)}$. In practice, one just has to calculate the posterior probability t_{ik} (where we omit the iteration superscript q when it is not needed).

$$t_{ik} \quad = \quad P\left(Z = k \mid X = \mathbf{x}_i, \hat{\theta}^{(q-1)}\right)$$

Then, the maximization step (M-step) consists in maximizing the same conditional expectation of the complete likelihood over θ to obtain a new parameter estimate

$$\hat{\theta}^{(q)} = \arg\max_{\theta} E\left[L_c\left(\mathbf{x}, z; \theta\right) \mid \mathbf{x}, \hat{\theta}^{(q-1)}\right]$$

Convergence is being determined by some stopping criterion, which can simply be $\| L\left(\mathbf{x}; \theta\right)^{(q)} - L\left(\mathbf{x}; \theta\right)^{(q-1)} \| < \varepsilon$, *i.e.* the absolute difference between two successive likelihood values. It would also be possible to use the Aitken's acceleration criterion (Lindsay, 1995) which estimates the asymptotic maximum of the likelihood. Alternatively, we can set a maximum number of iterations after which the algorithm stops if it has not yet converged.

The algorithm forms a sequence $\left(\theta^{(q)}\right)_q$ which is guaranteed to converge toward a local optimum of the likelihood (Wu, 1983). For further details on the EM algorithm, the reader may refer to (McLachlan and Krishnan, 1997).

Once the EM algorithm has converged, a partition of the data can be deduced from the posterior probabilities $t_{ik} = P(Z = k|X = \mathbf{x}_i, \hat{\theta})$ by using the *maximum a posteriori* (MAP) rule which assigns the observation $\mathbf{x}_i$ to the group with the highest posterior probability.

The EM algorithm has some known limitations. The first one is that the parameter estimate at convergence is highly dependent on the initialization of the algorithm. One of the solutions that that practitioners use for this issue is to initialize the algorithm with parameter estimates θ obtained by running multiple EM algorithms with different random initializations for a few iterations and then choosing the one which yields the maximum likelihood. Another solution is proposed by (McLachlan and Peel, 2000) in the Gaussian case. They initialize EM by setting all mixture proportions to be equal, initializing the means by simulating samples from a Gaussian distribution $\mathcal{N}(\mathbf{m}, S)$, where $\mathbf{m}$ and S are the mean and covariance matrix for the whole dataset, respectively. Finally, they set all class covariance matrices equal to S.

The second issue of EM is that it can be slow as far as convergence is concerned, since it can get stuck in local maxima and saddle points where the likelihood function is flat. In order to face this issue, (Celeux and Diebolt, 1985; Celeux et al., 1988) have proposed Stochastic EM (SEM) which is a stochastic variant of the standard EM algorithm.

Algorithm 1 Expectation-Maximization for GMM

Input: Data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathbb{R}^p$, number of groups K, threshold ε, maximum iteration number $maxit$

Output: Estimated parameters $\hat{\theta}$

1: Initialize to acquire an initial estimate $\hat{\theta}_0$.
2: Iteration index $q = 1$
3: **while** $\mid L^{(q)} - L^{(q-1)} \mid \geq \varepsilon$ OR $q < maxit$ **do**
4: **E-step:** Compute the expected log-likelihood of the complete data conditionally to the current value of the parameters

$$t_{ik} = P\left(Z = k \mid X = \mathbf{x}_i, \hat{\theta}^{(q-1)}\right)$$

5: **M-step:** Maximize $E\left[L_c\left(\mathbf{x}, z; \theta\right) \mid \hat{\theta}^{(q-1)}\right]$ over θ to obtain $\hat{\theta}^{(q)} = (\hat{\pi}_k, \hat{\boldsymbol{\mu}}_k, \hat{\Sigma}_k)$

$$\begin{aligned}
\hat{\pi}_k &= \frac{n_k}{n} \\
\hat{\boldsymbol{\mu}}_k &= \frac{1}{n_k} \sum_{i=1}^{n} t_{ik} \mathbf{x}_i \\
\hat{\Sigma}_k &= \frac{1}{n_k} \sum_{i=1}^{n} t_{ik} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T
\end{aligned}$$

where $n_k = \sum_{i=1}^{n} t_{ik}$ and $n = \sum_{k=1}^{K} n_k$.
6: Evaluate the likelihood function $L^{(q)}$.
7: $q = q + 1$
8: **end while**

3.1.4 Model Selection in model-based clustering

A known problem in model-based clustering is choosing the model form m as well as its complexity K, *i.e.* the number of groups of the mixture. The form of the model is determined by assumptions made on the structure of the covariance matrices. For some meaningful assumptions, the reader can refer to (Banfield and Raftery, 1993) and (Celeux et al., 1996). Unfortunately, neither of m or K can be estimated by the maximum likelihood principle since the likelihood of the mixture depends on both. Instead, there have been model selection criteria that were derived to tackle this issue. We briefly discuss their derivation below. Note that we followed the introduction given by (Biernacki et al., 2000), where a Bayesian treatment is used.

A solution to the aforementioned issue would be to maximize the integrated or marginal likelihood $L'(\mathbf{x}; m, K)$

$$(\hat{m}, \hat{K}) = \arg\max_{m,K} L'(\mathbf{x}; m, K)$$

where

$$L'(\mathbf{x}; m, K) = \int_{\Theta_{m,K}} L'(\mathbf{x}; m, K, \theta)\ \xi(\theta \mid m, K) \mathrm{d}\theta, \tag{3.3}$$

Here, $\theta = (\pi_k, \boldsymbol{\mu}_k, \Sigma_k), \forall k \in \{1, \ldots, K\}$ is the vector of parameters of the mixture and Θ is the parameter space. By $\xi(\theta \mid m, K)$ we denote the prior on the mixture parameters given the model.

The integral in (3.3) is intractable. The well-known Bayesian Information Criterion (BIC, see (Schwarz, 1978)) approximates (3.3) by

$$\log L'(\mathbf{x}; m, K) \approx \log L'(\mathbf{x}; m, K, \hat{\theta}) - \frac{v_{m,K}}{2} \log n \tag{3.4}$$

where $\hat{\theta}$ is the maximum likelihood estimate of the parameters and $v_{m,K}$ is the number of free parameters in the model with form m and K groups.

As (Biernacki et al., 2000) point out, regularity conditions necessary for applying the BIC criterion do not always hold in a classification/clustering setting. They propose instead to maximize the integrated likelihood of the complete data, also known as classification likelihood (CL)

$$L'_c(\mathbf{x}, \mathbf{z}; m, K, \theta) = \int_{\Theta_{m,K}} L'_c(\mathbf{x}, \mathbf{z}; m, K, \theta)\ \xi(\theta \mid m, K)\ d\theta \tag{3.5}$$

Given certain assumptions, (Biernacki et al., 2000) derive the Integrated Classification Criterion (ICL), which is better suited for a clustering task.

Note that (Akaike, 1973) has also proposed a criterion for model selection, known under the name of *Akaike Information Criterion* (AIC). However, it does not penalize sufficiently model complexity and therefore, tends to select models with a non optimal number of parameters.

3.1.5 The curse of dimensionality

We have already seen that modern data are often high-dimensional. This holds especially for data issued from real-world systems, since modern technology allows for a multitude of measurements to be taken simultaneously. We can cite, for instance, aircraft engine health monitoring data, collected using a large number of sensors placed on the engine or the test chamber.

Intuitively, one could expect that classification would be facilitated by adding more variables (measures), since more information would be available. Unfortunately, it has been shown that the number of data samples needed for a given task, *e.g.* classification grows exponentially with the dimension (number of variables). Bellman, who first used the term *curse of dimensionality* (Bellman, 1957, 1961), observed this phenomenon while trying to optimize exhaustively a function in a discrete space. Moreover, (Silverman, 1986) made the same observation in approximating an arbitrary Gaussian distribution with fixed Gaussian kernels.

Indeed, humans are used to the three-dimensional space of the real world, but there can be some unintuitive phenomena in a high-dimensional space. Perhaps the most illustrative example is the "empty space phenomenon" (Scott and Thompson, 1983), which is often presented using the example of the volume of a hypersphere. Let us

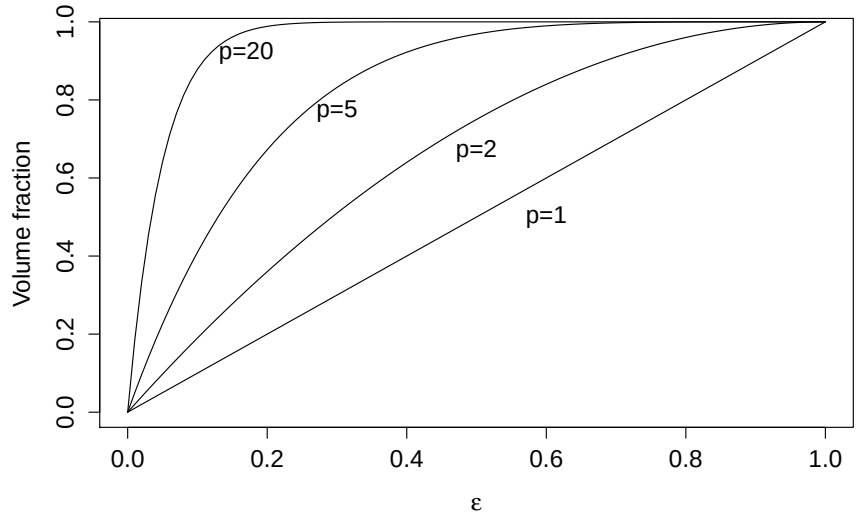


Figure 3.3: The fraction of the volume of a sphere in p-dimensional space that lies in the shell between $r = 1 - \varepsilon$ and $r = 1$ for various values of ε and p. Here, r is the radius of the sphere. As we can see, the fraction approaches 1 even for lower-dimensional spaces (small values of p).

consider a sphere of radius $r = 1$ in the p-dimensional space. What is the fraction of its volume that lies in the shell between radius $r = 1 - \varepsilon$ and $r = 1$? The volume of a sphere of radius r in the p-dimensional space is given by

$$V_p(r) = \frac{4\pi r^p}{p} = K_p r^p$$

where K_p depends only on p. Thus the fraction we are searching should be

$$\frac{V_p(1) - V_p(1-\varepsilon)}{V_p(1)} = 1 - (1-\varepsilon)^p \tag{3.6}$$

Figure 3.3 shows (3.6) as a function of ε, for different values of p. As we can see, this fraction is close to 1 even for small values of p. This would mean that in high-dimensional spaces, the entire volume of a sphere is concentrated near its surface. This is a rather surprising conclusion which goes on the contrary of human intuition and illustrates a type of unintuitive behavior one can have in high-dimensional spaces.

In the context of generative models, the curse of dimensionality manifests itself through the number of model parameters that have to be estimated. Generally speaking, in generative methods, we have to estimate a number of parameters that scales with the square of the dimension. In the Gaussian case this is primarily due to the estimation of the covariance matrix. Indeed, if n, the number of data samples, is smaller than

the dimension p, the estimation of the covariance matrix will be singular and it will be impossible to invert it (Bouveyron et al., 2007a).

3.2 Anomaly Detection

The field of Anomaly Detection has received a great deal of attention in the last decade. The interested reader can find some detailed survey papers covering many related topics in (Chandola et al., 2009; Markou, 2003a,b). As (Chandola et al., 2009) point out, we find in the literature different terms for the notion of anomalies depending on the context and the application: novelties, outliers, discordant observations, exceptions, aberrations, surprises, peculiarities or contaminants.

Some characteristics of the research conducted in the field, as pointed out by the survey papers above and based on our experience, are the following:

- Research in the field has very often been motivated by application-specific needs and thus, it has been conducted in a rather "unorganized" manner.
- An important problem is the lack of standard, established benchmarks, namely datasets on which proposed methods could be evaluated and compared to others.
- Although a definition encompassing any type of anomalies can be given, in practice, this depends on the application at hand.

In the literature, we can find a variety of type of techniques for the problem of Anomaly Detection: statistical (parametric and non-parametric) and clustering methods, neural-network based and classification (supervised) methods, as well as nearest-neighbor based, graph-based, information-theoretic and rule-based methods.

In this work, we focus on unsupervised learning methods for anomaly detection. and we will thus present the state of the art for these particular families of methods. We have built up this chapter based on existing surveys (Chandola et al., 2009; Markou, 2003a,b), updating them as necessary, since the most recent one dates from 2009.

(Chandola et al., 2009) have written a broad survey on anomaly detection, covering various techniques and applications found in the literature. (Markou, 2003a,b) have divided their survey in two distinct parts: the first part presents statistical approaches for Novelty Detection, while the second one is dedicated to Neural Network based techniques. In these works, there is a focus on the methods themselves rather than the applications.

3.2.1 Statistical approaches

We begin by presenting statistical parametric approaches for Anomaly Detection. This type of methods makes an assumption that the data follow a given distribution.

Parametric approaches

A simple method for determining if a data sample is anomalous, given the assumption that the underlying distribution of the population is Gaussian, is to flag as anomalous all data which are farther than a 3σ distance from the mean μ, where σ is the standard deviation. The range $[\mu - 3\sigma, \mu + 3\sigma]$ contains $99,7\%$ of the data.

Assuming that the data are generated from a Gaussian distribution, another option for detecting anomalies is the Grubb's test (Grubbs, 1969). For a data sample $\mathbf{x}$, we calculate its z-score:

$$z = \frac{|\mathbf{x} - \bar{\mathbf{x}}|}{s}$$

where $\bar{\mathbf{x}}$ is the sample mean and s the sample standard deviation. We can set a threshold by taking the value of a Student's t-distribution at a given confidence level α. If the z-score of a data $\mathbf{x}$ is above this threshold, then we declare $\mathbf{x}$ an anomaly. Thus, by controlling α, we control the threshold and the number of data that will be labeled as anomalies. Laurikalla (Laurikkala et al., 2000) apply the Grub's test to multivariate data by transforming them to a scalar using the Mahalanobis distance.

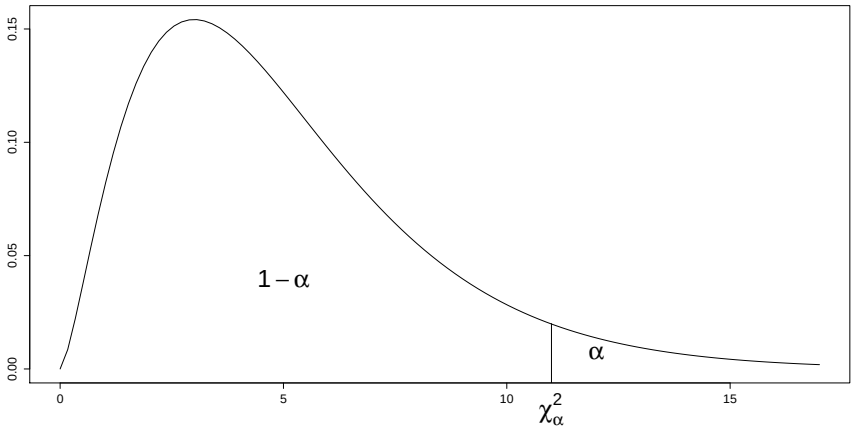
(Ye and Chen, 2001) use a χ^2 statistic to detect anomalies:

$$\chi^2 = \sum_{j=1}^{p} \frac{(X^{(j)} - E^{(j)})^2}{E^{(j)}}$$

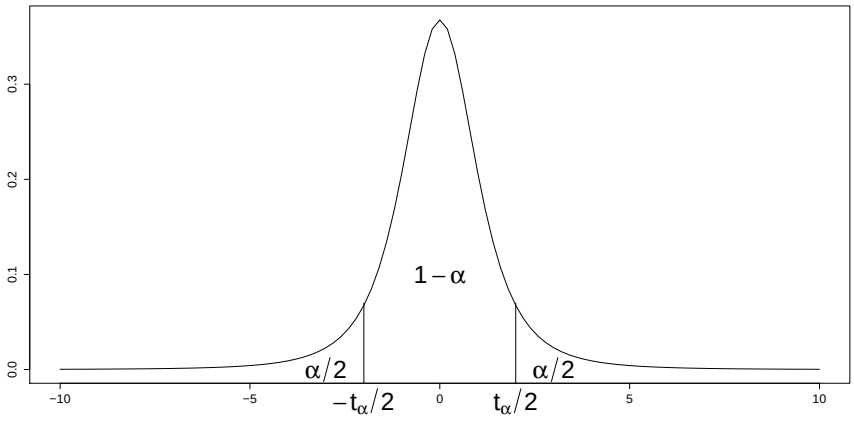
where p is the number of variables, $X^{(j)}$ is the j-th variable of a test data sample and $E^{(j)}$ is the expected value for the j-th variable given by its mean in the training data. Anomalies are characterized by large values of χ^2.

A more sophisticated (parametric) approach is to consider that the normal (anomaly-free) data has been generated by a Gaussian mixture model (GMM) and estimate the parameters using the EM algorithm. Once a GMM has been fitted to the training data, we assign each previously unseen test observation to its most probable class. Then, one can consider that if a novel test data sample falls in low-probability regions of the model, it is more likely to be abnormal than if it falls in high probability regions. A recurrent problem with this approach is that, at some point, a heuristic threshold must be set in order to distinguish between a normal and an abnormal observation. The quantity that one thresholds generally depends on the algorithm used, but common practices are to threshold the posterior class probability for a test sample, or its distance from the class mean.

(Roberts and Tarassenko, 1994) propose a method that gradually grows a GMM. The number of the classes is not fixed in advance but grows in the training phase based on an automatic criterion. In the training phase, distances of Mahalanobis are calculated between data and all mixture components. If, for a given training data sample, its Mahalanobis distance to its closest component is less than a given threshold ϵ_t, then a new Gaussian component, centered on the observation, is added to the mixture. The threshold ϵ_t reduces over time until it becomes smaller than a user-defined ϵ_{max} which



(a)



(b)

Figure 3.4: (a) The chi-squared distribution with 5 degrees of freedom. Here, α is the confidence level for the test and it holds $\alpha = P(X \geq \chi^2_\alpha)$. (b) The student-t distribution with 3 degrees of freedom. It holds $\alpha = P(|X| \geq \frac{t_\alpha}{2})$

corresponds to the desired precision. After the training phase, the mixture is no longer being grown. For a test observation, we calculate its Mahalanobis distance to all components and compare the smallest one to ϵ_{max}. If the distance is greater than the threshold, then the example is flagged as an anomaly.

(Tarassenko et al., 1995) estimate the density of normal data by the means of the marginal probability of the training data $P(X)$. They assume that anomalies are distributed uniformly outside the normal region. If a test observation falls in a region of the normal (anomaly-free) data space with a density lower than a certain threshold, then it is flagged as an anomaly. The authors set local thresholds (per class) by partitioning the normal data space using k-means and estimating the density independently in each partition using GMM.

(Roberts, 1999, 2000) advocate the use of Extreme Value Theory (EVT) as a principled approach to Anomaly Detection as opposed to (heuristic) thresholds widely used in the literature. EVT models the tails (extreme values) of distributions. Theoretical results from (Fisher and Tippett, 1928) show that one of the forms that can take a non-degenerate limit distribution for normalized maxima is the Gumbel distribution. The authors first estimate data density using GMM and then use the corresponding EVT probability to detect anomalies. In fact, they do use a threshold, but it is a rather "natural" (data-driven) one: they consider that any observation falling out of the $P > 0.95$ boundary of the EVT probability is abnormal.

(Lauer, 2001) tries to fit a two-component GMM to the data. One of the components aims at capturing the normal data and the other one the anomalies. Thus, contrary to other methods, this work does not suppose that the (training) dataset is anomaly-free. This type of methods has the benefit of not having to set a threshold. Assuming that data has been generated by a Gaussian distribution, the authors fit a GMM to normal data, with an extra fixed Gaussian component of large variance to capture anomalies. Then, an observation is declared to be abnormal if its probability of belonging to the anomaly distribution is greater than that of belonging to the distribution of normal data.

In a similar way, (Byers and Raftery, 1998) fit a mixture of two Poisson distributions to a dataset containing anomalies. One of the two Poisson components is used to capture anomalies. (Eskin, 2000) fits a GMM on the normal data with an extra uniform component to capture anomalies.

(Song et al., 2007) propose an approach in which anomalies are looked for in the indicators (endogenous variables) but the environment is considered as well. For example, in an aircraft engine context, environmental variables can be the ambient temperature and the air pressure, while the indicators are the engine rotation speed, the oil pressure inside the engine etc. In their approach, there are two GMMs: one to model the environment and another one to model the indicators. The key point is that the authors assume that a vector each component of the environmental GMM, generates or maps itself into a component of the indicator GMM with a certain probability. This probability can be seen as a *transition* probability from the environment GMM to the GMM of the indicators. It is not known in advance but it is estimated like the rest of the parameters of the two GMMs. Anomalies in the training set are detected by sorting the data according to their likelihood under the model in an increasing order and then

taking the first $c = \epsilon n$ data to be anomalies, where ϵ is the expected anomaly rate in the training data, specified by the user, and n the number of training data. For a test data sample, the method calculates its likelihood and compares it to the likelihood of the c-th (based on the sorting) training data sample that was deemed abnormal. In other words, if the new data sample is less probable under the model than the most probable ("less anomalous") among the anomalies, then it is tagged as an anomaly.

Non-parametric approaches

The above methods using GMM have one thing in common: they are parametric, that is they assume that data has been generated by a family of known distributions, for instance, the Gaussian distribution. This allows to estimate (model) the underlying density for the data. However, density estimation can also be performed in a non parametric manner, that is without assuming a specific distribution for the data.

One of the simplest parametric techniques is the box-plot rule, which is a statistical method to detect anomalies for univariate data. We can use boxplots for multivariate data, plotting a box for each of the variables.

Another simple non parametric technique is the histogram. For the anomaly detection task, we first build a histogram using normal data. Then, for a test observation, we check if it falls in one of the bins of the histogram, in which case it is normal, otherwise it is flagged as an anomaly. Moreover, the height of the bins can be used to obtain anomaly scores. Clearly, the choice of the size of the bins is critical, since small bins might result in high false alarm rates due to data falling in low-height bins. The histogram is essentially a technique for univariate data. To extend this to multivariate data, one can build a histogram per variable and then aggregate the different anomaly scores (Ho et al., 1999; Manson et al., 2000).

(Yeung and Chow, 2002) use Parzen windows (Parzen, 1962) with a Gaussian kernel to model data density. The choice of the Gaussian kernel is made for convenience and is not an assumption on the nature of the data. They then formulate the anomaly detection problem as a hypothesis testing problem, with the null hypothesis being that a data sample is normal. We can control the false alarm rate of the method by adjusting appropriately the confidence level of the test.

3.2.2 Clustering Approaches

(Ester et al., 1996) propose a density-based clustering algorithm for data with anomalies, based on the notion of the local density of a data sample, calculated by its neighbors. (Guha et al., 2000b) propose a robust, *i.e.* resistant to anomalies in the data, hierarchical clustering algorithm using links between data samples instead of distances. A user-defined threshold is used to detect anomalies. (Ertöz et al., 2003) define a similarity measure for two data samples based on the number of the nearest neighbors they share. Anomalies can be detected by setting an appropriate threshold.

(Eskin et al., 2002; Chan et al., 2003; He et al., 2003) used *fixed-width clustering*, where all clusters are of fixed width w and they can thus overlap. If a data instance falls

outside of the "region" of all existing clusters, then a new cluster is created. (He et al., 2003) define CBLOF (Cluster-Based Local Outlier Factor), an anomaly score which takes into consideration the size of the cluster as well as the distance to its closest cluster. The threshold on these quantities has to be set empirically.

3.2.3 Robust clustering

Another way to perform clustering on data contaminated with anomalies (robust clustering) is by trimming outlying (abnormal) data samples in a straightforward manner.

Early works include the least trimmed squares (LTS) and least median squares (LMS) for regression (Rousseeuw and Leroy, 2005), as well as the Minimum Covariance Determinant (MCD) and Minimum Volume Ellipsoid (Rousseeuw, 1985), which are both location-scatter estimators (single-class case) using the trimming principle.

(Cuesta-Albertos et al., 1997) developed a trimmed k-means algorithm. (Gallegos, 2002; Gallegos and Ritter, 2005) have developed a probabilistic clustering model for heterogeneous clustering called "spurious-outliers" model. In this method, the underlying optimization problem is ill-posed. Thus, (Gallegos, 2002) proposed a method based on unit group covariance matrices to constrain the optimization, while (Gallegos and Ritter, 2005) assume that groups share a common covariance structure.

In the same line of reasoning, (García-Escudero et al., 2008) propose TCLUST, a GMM-based trimming method imposing a constraint on the ratio of the largest to the smallest eigenvalue among eigendecomposition matrices of all group covariance matrices. (Fritz et al., 2012a) have recently proposed a faster variant of the TCLUST algorithm. A comprehensive review of the above works is given in (García-Escudero et al., 2010).

In a similar work, (Neykov et al., 2007) propose the use of the Weighted Trimming Likelihood Estimator (WTLE). Recently, (Punzo and McNicholas, 2013) proposed a method of robust GMM clustering that estimates the trimming parameter via the Expectation-Conditional Maximization algorithm (Meng and Rubin, 1993).

3.2.4 Nearest Neighbor Approaches

A broad category of approaches is based on the nearest neighbor schema. In its original form, the algorithm of k-nearest neighbors is a classification algorithm. Assuming that we have a training labeled set, the class of a previously unseen data sample is obtained by a majority vote of the classes of its k-nearest neighbors in the training set. Note here that no proper training step is being executed.

In the context of anomaly detection, the nearest neighbor technique is used in an unsupervised manner. We are not interested in the class of the test data sample, but rather in its distance to its k-nearest neighbors in the training set or the sum of these distances ((Guttormsson et al., 1999), (Eskin et al., 2002)). This distance can be used as an anomaly score. We can then set a threshold to detect anomalies. Intuitively, anomalies will tend to appear "far" from their nearest neighbors.

Another category of approaches is based on the fact that nearest neighbor approaches can function as data density estimators, since they calculate distances between data at

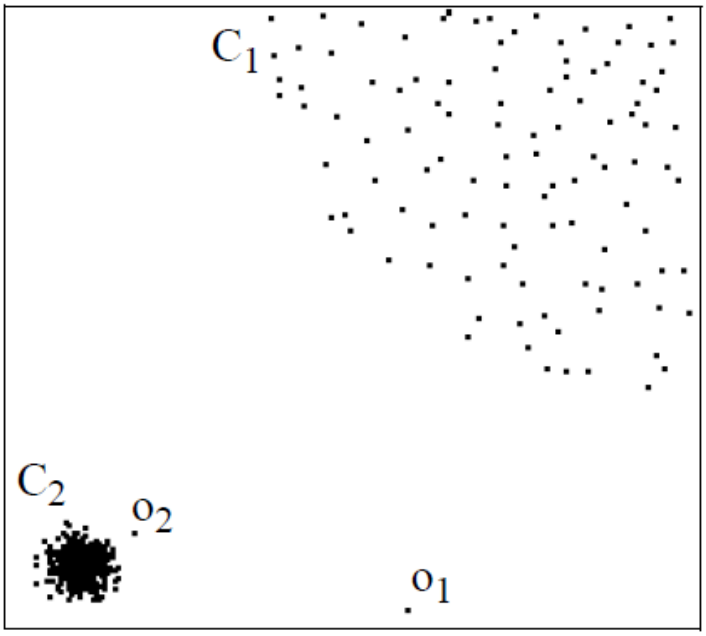


Figure 3.5: Traditional nearest neighbor methods for anomaly detection will only consider o_1 as an anomaly, but LOF (Breunig et al., 2000) will also flag o_2 as anomalous, since the nearest cluster C_2 is very dense and thus, o_2 is an anomaly with respect to its local neihborhood.

a given region of the space (Chandola et al., 2009).

(Breunig et al., 1999, 2000) introduce a measure called "Local Outlier Factor" (LOF). For a given data sample, its LOF score is equal to the ratio of the average local density of its k nearest neighbors and the local density of the sample itself. The local density of a data sample is simply the inverse of the average of the reachability distances in the sample's neighborhood, with the reachability distance being a measure which is relatively small for samples lying in a dense neighborhood and bigger for samples lying far away from this neighborhood. Thus, for normal data lying in dense regions, their local density will be similar to that of their neighbors, while for an anomalous sample, the local density will be lower than that of its nearest neighbors, and hence, its LOF score will be higher. There has been several improvements to LOF (Tang et al., 2002; Hautamaki et al., 2004; Papadimitriou et al., 2003).

(Guttormsson et al., 1999) are testing a variety of non-parametric techniques for an anomaly detection task. In particular, they are fitting surface boundaries "around" normal data and then compare the distance of each unseen data instance to this boundary to determine if a data sample is abnormal. They use surfaces such as a hypersphere, an ellipse and a box, making use of nearest neighbors.

(Tax and Duin, 1998, 2000) estimate the density of the normal data with a GMM. For a test observation, they calculate the quotient between the distance to its nearest

neighbor in the training set, as well as the distance of the latter to its nearest neighbor. If this quotient is much greater than 1, then the observation is flagged as anomalous. They also propose an anomaly detection method based on the instability of multiple classification, obtained by bootstrapping the training data. They then set appropriate thresholds on these measures.

3.2.5 SOM-based approaches

A self-organizing map (SOM) (Kohonen, 2001) is a network of inter-connected units that are positioned following a certain topology (for instance, a grid-like topology). For each of these units, a neighborhood of a given radius is defined by the set of units of the network being in a distance less or equal to the radius. Each unit is represented by a prototype vector. For each data sample, its closest prototype vector, say $\mathbf{m}$, is being found and the sample is affected to the corresponding class, i.e. to the set of samples whose closest prototype vector is $\mathbf{m}$. Thus, SOM defines a partitioning (clustering) of the data. Then, this prototype vector but also its neighbors are updated so that they come closer to the data sample. In this way, there is a an organisation of the units (the map) that respects the topology and neighborhoods of the original data space. Note that the method is unsupervised and does not require knowledge of the true labels of the data.

(Harris, 1993) train a SOM on normal data. Then, for each test data sample, he calculates its distance from its closest prototype vector. If it is over a certain threshold, then it is flagged as anomalous. Abnormal test data samples are generally expected to have larger distances from the map than normal ones. (Ypma and Duin, 1997) propose a similar approach .

(Emamian et al., 2000; Labib and Vemuri, 2002) train a SOM with normal data, noting their trajectories on the map, that is the indexes of the cells to which normal data has been affected. For a test data sample, we determine its closest prototype vector and we affect it to the corresponding class. It is generally expected that abnormal test samples will be affected to different classes than normal ones.

(Marsland et al., 2000) use a SOM complemented with a habituation mechanism which "forgets" normal data seen often and concentrates on abnormal ones.

3.2.6 Random projection anomaly detection

Recently, a series of works have used random projections to project a high-dimensional dataset onto a lower dimensional subspace. The theoretical foundation of the random projection framework lies on the Johnson Lindenstrauss Theorem (Johnson and Lindenstrauss, 1984), which claims that the pairwise distances are well preserved through random projection (see also results in (Achlioptas, 2001, 2003)).

(Ding and Kolaczyk, 2011) propose an anomaly detection method based on the well-known principal component analysis (PCA) and the random projection technique. Let $X = (\mathbf{x}_1, \ldots, \mathbf{x}_n), \quad \mathbf{x}_i \in \mathbb{R}^p \quad \forall i$ be a dataset. By doing PCA on X and using the first k principal vectors, we obtain $\hat{X}$, which is an approximation of X. Then the l_2-norm

of the residual of the projection is $Q = (X - \hat{X})^T(X - \hat{X})$. Assuming now that we have a random projection matrix Φ. Then let $Y = l^{-\frac{1}{2}}\Phi^T X$ be its projection to a subspace of dimension l, $l < p$. The authors assume that only the projection Y of the data is observed and not the data X itself. PCA on Y will give us $\hat{Y}$ with the squared l_2-norm of the residual being $Q^* = (Y - \hat{Y})^T(Y - \hat{Y})$. The residual can then be used to detect anomalies. The contribution of this work is that it shows that, under certain conditions on the covariance matrix of X, it is possible to apply PCA on Y yielding Q^* and nevertheless obtain anomaly detection comparable to that which would have been obtained by using Q, with the discrepancy between the two made precise. (Fowler and Du, 2012) also use random projections for anomaly detection.

3.3 Datastream clustering

We saw earlier that the parameters of a GMM are most of the times estimated using the EM algorithm. Modern data are often arriving sequentially in a possibly infinite time horizon; for instance, data coming from sensors installed on an object whose behavior we want to monitor. We usually refer to this data as *data streams* (or datastreams).

Mining datastreams poses new challenges for standard estimation/optimization algorithms:

- Datastreams are most of the times non-stationary, that is, the distribution that generates the data changes over time.
- Datastreams can be of infinite (time) horizon. Thus, it is infeasible to keep all of the past data in memory. This also poses the question of how 'forgetting' the contribution of past data should be implemented.
- Many applications require a short response time, as the rate of arrival of the data is high and critical decisions could be based on the output of the algorithm.
- There are real-world applications (e.g., sensor networks) where limited computational resources are available.

Thus, in the GMM context, an algorithm should incrementally estimate the parameters by integrating the contribution of each novel data sample in a time-efficient manner without refitting the mixture on the whole data. In its standard, '*batch*' version, the EM algorithm iterates through data multiple times until convergence to a local maximum of the likelihood function.

Online variants of the EM algorithm and methods for the incremental learning of GMMs have been developed to tackle these issues. The first category of methods focuses on online parameter estimation, developing online EM algorithms, which can be used in more general settings (not only Gaussian). In the second category of works, the focus is given on learning incrementally a GMM. The aim here is to develop methods that can update parameters as well as the model complexity (number of mixture components) in an online manner.

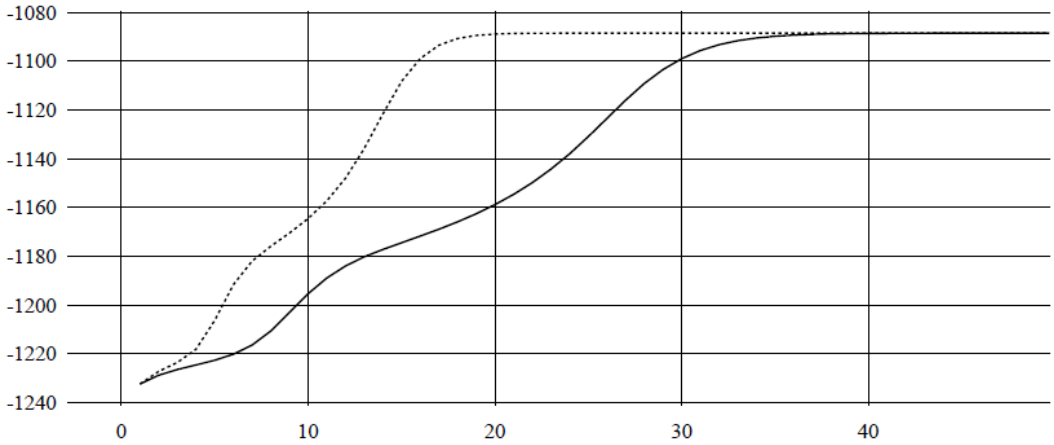


Figure 3.6: Convergence rates for the standard EM (solid line) and the incremental EM (dotted line) algorithms (Neal and Hinton, 1998). The horizontal axis shows the number of passes over the data, whereas the vertical axis shows the values of the log-likelihood.

3.3.1 Online EM

A pionering work is the one of (Titterington, 1984) who derived equations for online updating of the parameters of models with incomplete data using stochastic approximation. The method is proposed for the most general case, but it can be adapted to the standard EM setting. At some point, given posterior probabilities and estimates $\hat{\theta}^{(n)}$ of the parameters after having seen n data samples, a new sample $\mathbf{x}_{n+1}$ is arriving. Then, its posterior is calculated and the parameters of the model are updated through appropriate update equations. The latter were derived using a stochastic gradient descent technique on the Fisher Information Matrix.

Since this work from (Titterington, 1984), two different types of online expectation-maximization methods have appeared in the literature: the incremental EM of (Neal and Hinton, 1998) and the online EM (Sato and Ishii, 2000; Cappé and Moulines, 2009).

(Neal and Hinton, 1998) proposed a view of EM that justifies incremental variants of EM. They reformulate standard EM as a procedure in which we perform a maximization of a function at both steps. They show that we can think of the EM process as continually updating the sufficient statistics. In this view, when we process a data sample at each iteration, we remove the contribution of the previous data sample to the sufficient statistics and replace it with the contribution of the new sample. The algorithm makes *multiple passes* over the data. An inconvenience is that the matrix of sufficient statistics has to be kept in memory for all past data. In other words, the memory requirements grow linearly with the number of data. As for time requirements, multiple iterations of the algorithm will sometimes take only slightly more time than one iteration of the standard algorithm according to the authors.

(Sato and Ishii, 2000) proposed an online EM method, later generalized by (Cappé and Moulines,

2009), based on a convex combination of old and updated sufficient statistics. This means that new data is being gradually incorporated into the model through their sufficient statistics, while forgetting older contributions. The proposed method uses a time-decaying learning rate parameter to control this process.

(Samé et al., 2007) develop an online classification expectation-maximization (CEM) algorithm . The standard (batch) CEM (Celeux and Govaert, 1992) is simply a classification version of the standard EM, where a "hard" clustering step is added between the expectation and the maximization step. (Samé et al., 2007) reformulate CEM so that a stochastic version can be derived and they then adapt the update equations given by (Titterington, 1984) for the M-step to the CEM context.

3.3.2 Incremental Learning of GMMs

The works in this category focus on incrementally learning a GMM, that is, adapting the model parameters and the number of groups incrementally. Note that a number of works among those presented below, there are some which operate incrementally in building the GMM but implicitly assume that the dataset is fixed, that is, they are batch methods and cannot be applied in an online context. However, we present them as well because they have introduced criteria and techniques for updating the model complexity.

(Ueda et al., 2000) propose an EM which splits components in areas with high population and merges components in areas with low population, in order to avoid local *maxima* of standard EM algorithm. It is a batch method. They begin by performing standard EM until convergence to obtain an initial mixture. They then choose which mixture components to split and/or merge. For splitting, they use a Kullback-Leibner divergence based criterion, while for merging, a posterior probability dot product criterion. They iterate this procedure until they find a model of higher likelihood via the split-and-merge operations. It has to be noted that split and merge operations are performed simultaneously and thus, the total number of mixture components does not change.

(Vlassis and Likas, 2002; Verbeek et al., 2003) propose a scheme for greedy learning of GMMs. Initially, at iteration $t = 1$, they start with the optimal one-component mixture. At each iteration, the algorithm is searching for the optimal $(t + 1)$-component mixture by plugging in a new component with weight α. The overall new mixture is then a convex combination of the existing mixture (weighted by $1 - \alpha$) and the new component (weighted by α). Clearly, this is an optimization problem, which is solved by launching multiple partial EMs (for a few iterations) with different initializations, keeping the one which gives the largest log-likelihood. This procedure is being iterated until a stopping criterion is met.

(Hall et al., 2005) propose a method for learning GMMs incrementally. Initially, they merge two existing GMMs (that is, the existing GMM and the new data modeled by a GMM) by trivially combining their parameters. Next, they simplify their model by merging components based on weights calculated using links of groups between them by setting a threshold on the matrix of Chernov bounds.

(Yang and Zwolinski, 2001) start with a large number of components and gradually fit a mixture to the data based on mutual information theory. If a mixture component has a low mutual information, then it is assumed to be statistically independent of the other components of the system and cannot be removed. On the contrary, if its mutual information is large, then it is assumed that it is statistically dependent and can be removed without significant loss of information. Components are being removed iteratively until the mutual information of the system becomes non-positive. Then, the mixture constructed from this optimal set of components will accurately estimate the true mixture.

(Song and Wang, 2005) propose a method for online learning of GMMs. They suppose that data arrive in blocks and maintain only sufficient statistics. They perform an EM on the newly arrived data. They merge statistically equal components, based on W and Hotelling's T^2 statistics to check for covariance matrices and means equality, respectively. Performances are equivalent to those of standard EM.

(Dang et al., 2009) have developed an EM-based algorithm for data streams, *i.e.* data arriving sequentially. They suppose that data arrive in time slots and they model data points in each time slot by a GMM. They keep only data belonging the b last time slots and in particular, sufficient statistics for each of them, from which they derive update equations for the M-step. After each EM run, split and merge operations are performed. More specifically, two components are merged if they are small and are close enough, while a component is split across a specific dimension if it is large and has high variance. After these operations are performed, the algorithm 'forgets' the oldest time slot by subtracting its sufficient statistics from the model.

(Arandjelovic and Cipolla, 2006) make an assumption of temporal coherence for the data, *i.e.* that data points which are close (in a temporal sense) will be in general highly correlated. In practice, this leads to simple update equations for the mixture parameters. It also means that no 'old' data has to be kept in memory. Actually, at each time step, only two GMMs are kept, the so-called 'historical' GMM and the current GMM, with the latter being the former updated with the novel data point. It is a three-stage procedure: the at the first stage, the historical mixture is updated with the novel data sample. In the second stage, all components are being split in a particular manner, and in the third stage components are merged following a MDL-based criterion.

(Declercq and Piater, 2008) propose an approach for online learning of GMMs. Data arrive in blocks. On the first level, there is what they call a 'precise' mixture model which is essentially a GMM, aiming at modeling data with high precision. On the top of this, there is an 'uncertain' mixture model, which is a distribution with a weighted sum of Gaussian and uniform components. Two components of the uncertain model are merged following a statistical measure called *fidelity*, which measures how close is a distribution to being Gaussian. If at some point, data modeling is oversimplified (following the fidelity measure) due to successive merges, components of the uncertain model are split according to their underlying precise model.

(Engel and Heinen, 2011) propose an incremental method for learning GMMs. The method is operating on data streams, processing data samples one by one. It begins with one single component centered on the first data point with a user-specified variance. A

new mixture component is added if the likelihood of a new data point is smaller than a novelty threshold, which depends on another user-defined parameter. The criterion defines the minimun acceptable likelihood of a data point as a fraction of the maximum value of the likelihood function. If it is not the case, the mixture is refitted on the data including the new data point by EM. The authors derive update equations for the M-step.

Finally, several works have treated the problem of clustering datastreams in a non probabilistic setting (Babcock et al., 2003; Domingos et al., 2001; Guha et al., 2000a; O'callaghan et al., 2002). In a number of these works, some extensions of the popular k-means or k-median algorithms are developed in order to cluster data streams. A rather different approach is adopted in (Aggarwal et al., 2004), where an online k-means-like clustering component is combined with an offline one, in order to better capture the evolution of the data stream. (Yang, 2003) cluster streams and not raw data based on a weighted distance between streams. For a broad presentation of heuristic clustering algorithms for data streams, see (Gaber et al., 2005).

Other works have extended existing anomaly detection algorithms to the online setting. (Pokrajac et al., 2007) have extended the LOF algorithm of (Breunig et al., 2000) and (Kriegel et al., 2011) the PreDeCon algorithm of (Bohm et al., 2004).

3.4 Conclusion

The main core of the chapter was dedicated to a survey on anomaly detection and clustering of data streams. We have also presented notions of model-based clustering, where we addressed, among others, the questions of inference and model selection.

In this Thesis, our goal is to use model-based clustering methods in order to detect anomalies in data streams. In the next chapter, we start with the simpler problem of detecting anomalies in an offline batch mode, that is, on a fixed dataset and with no online estimation of the parameters being involved. To this extent, we present a robust model-based clustering method for high-dimensional data which makes use of trimming to detect anomalies.

Chapter 4

Robust model-based clustering of high-dimensional data

In Chapter 3 we introduced model-based clustering principles. We also saw that a characteristic of modern data is that they contain many variables, *i.e.* they are high-dimensional. If we apply model-based clustering algorithms to high-dimensional data in a straightforward manner, then we will see that they perform poorly due to problems related to the curse of dimensionality and in particular, the over-parametrization of the models.

Moreover, modern data, especially those composed of measurements issued from real-world systems, can also contain anomalies due to defects or failures of their mechanical components or measurement instruments. Unfortunately, most model-based clustering methods are not robust to anomalies. Indeed, their estimates tend to be corrupted by the presence of anomalous data samples.

In this Chapter, we introduce the *High-Dimensional Robust Clustering* (HDRC) algorithm, which was designed to cope with high-dimensional data containing anomalies. The proposed algorithm combines subspace clustering with robustness to anomalies, achieved by the use of the trimming technique. Mathematical details of the algorithm are given and experiments on simulated and real-world data are presented.

As a sort of introduction to high-dimensional clustering and robust clustering, we briefly review advancements in those fields in sections 4.1 and 4.2. Section 4.1 also contains details on a specific subspace clustering algorithm on which we were based to construct HDRC.

4.1 Subspace clustering

Since data often live in subspaces of a dimension lower than this of their original, high-dimensional space, dimension reduction methods are frequently used in practice to reduce the dimension of the data before the clustering step. Feature extraction methods, such as principal component analysis (PCA), or feature selection methods are very popular.

However, dimension reduction techniques usually provide a sub-optimal data representation for the clustering step since they imply an information loss which could have been discriminant.

To avoid the drawbacks of dimension reduction, several recent *subspace clustering* approaches have been proposed to allow model-based methods to cluster high-dimensional data efficiently. We present below some of the methods belonging to this category.

4.1.1 Related work

(Banfield and Raftery, 1993) proposed a spectral decomposition of the group covariance matrices as follows

$$\Sigma_k = \lambda_k D_k A_k {D_k}^T, \tag{4.1}$$

where D_k is the matrix of the eigenvectors of Σ_k, A_k is a diagonal matrix with the normalized eigenvalues of Σ_k, sorted in decreasing order and $\lambda_k = \det(\Sigma_k)^{1/p}$, with p being the dimension of the data. These terms control the volume (λ_k), the orientation (Q_k) and the form of the class density (A_k), respectively. This decomposition has been motivated by a need for *parsimony*, that is, having models with fewer parameters to estimate and thus, overcome problems related to the curse of dimensionality. This formulation allows for defining a family of models by imposing constraints on λ_k, D_k, A_k.

Subspace clustering methods are searching to model the data in subspaces of much lower dimension and, thereby, avoid numerical problems and boost clustering performance. To begin with, the standard factor analysis (FA) model (Basilevsky, 2009; Bartholomew et al., 2011) links linearly the p-dimensional random vector X to a d-dimensional latent vector Y:

$$X = UY + \boldsymbol{\mu} + \epsilon \tag{4.2}$$

The $p \times d$ factor matrix U relates the two random vectors and $\boldsymbol{\mu} \in \mathbb{R}^p$ is a fixed location parameter. It is also assumed that the error term is distributed according to $\epsilon \sim \mathcal{N}(\mathbf{0}, \Psi)$ where Ψ is a diagonal covariance matrix. When $d < p$, X provides us with a parsimonious representation of X. In this context, d is interpreted as the intrinsic dimension of X.

The mixture of factor analyzers (MFA) may be considered as the earliest and the most general subspace clustering method. MFA both clusters the data and locally reduces the dimensionality of each cluster.The MFA model differs from the FA model in that it allows for different local factor models, in different regions of the input space, unlike FA, which assumes a common factor model for the entire space. MFA is an extension of factor analysis to a mixture of K factor analyzers. This approach, introduced by (Ghahramani et al., 1996), was generalized a few years later by (McLachlan et al., 2003), which removed in particular the constraint on the variance of the noise. The MFA model was also extended by (Baek et al., 2010), which introduces the mixture of factor analyzers with common factor loadings (MCFA) by adding restrictions on the means and the covariance matrices.

A general framework for the MFA model, which includes the works of (Ghahramani et al., 1996) and (McLachlan et al., 2003), was also proposed by (McNicholas and Murphy,

2008). The authors propose a family of models known as the expanded parsimonious Gaussian mixture model (EPGMM) family. They derive 12 EPGMM models by constraining the terms of the covariance matrix to be equal or not, considering an isotropic variance for the noise term or re-parametrizing the factor analysis covariance structure.

Recently, (Bouveyron and Brunet, 2012) have proposed the *Discriminative Latent Model* (DLM), which is a family of mixture models which fit the data into a common discriminative subspace. DLM differs from the FA-based models in that, in the former, the latent subspace is common to all groups and is assumed to be the most discriminative subspace of dimension d. Moreover, the FA-based models choose the latent subspace(s) maximizing the projected variance whereas the DLM model chooses the latent subspace which maximizes the separation between the groups. Let us note that the inference of the DLM models is not possible with the EM algorithm and (Bouveyron and Brunet, 2012) have proposed an alternative inference algorithm, called the Fisher-EM algorithm.

4.1.2 High-Dimensional Data Clustering

(Bouveyron et al., 2007a,b) proposed a family of 28 parsimonious and flexible Gaussian models to deal with high-dimensional data. Their approach combines dimension reduction, parsimony and regularization in order to cluster efficiently high-dimensional data.

Given a set of data samples $(\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathbb{R}^p$, HDDC assumes that the density of the data is given by a mixture of K Gaussian distributions

$$f(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \phi(\mathbf{x}; \boldsymbol{\mu}_k, \Sigma_k)$$

where $\boldsymbol{\mu}_k, \Sigma_k$ are the mean vector and the covariance matrix for the k-th group, respectively. The eigendecomposition of Σ_k gives

$$\Sigma_k = Q_k \Lambda_k {Q_k}^T$$

where Q_k is a $p \times p$ orthogonal matrix which contains the eigenvectors of Σ_k and Λ_k is a $p \times p$ diagonal matrix containing the associated eigenvalues (sorted in decreasing order). The key idea is to re-parametrize the matrix Λ_k, such as Σ_k has only $d_k + 1$ distinct eigenvalues, with $d_k < p$:

$$\Lambda_k = \begin{pmatrix} a_{k1} & & 0 & & \\ & \ddots & & \mathbf{0} & \\ 0 & & a_{kd_k} & & \\ & & & b_k & 0 \\ & \mathbf{0} & & & \ddots \\ & & & 0 & b_k \end{pmatrix} \begin{matrix} \left.\vphantom{\begin{matrix} a \\ a \\ a \end{matrix}}\right\} d_k \\ \left.\vphantom{\begin{matrix} a \\ a \\ a \end{matrix}}\right\} (p - d_k) \end{matrix} \tag{4.3}$$

where the d_k first values $a_{k1}, \ldots, a_{kd_k}$ parametrize the variance in the group-specific subspace and the $p - d_k$ last terms (b_k) model the variance of the noise. Therefore,

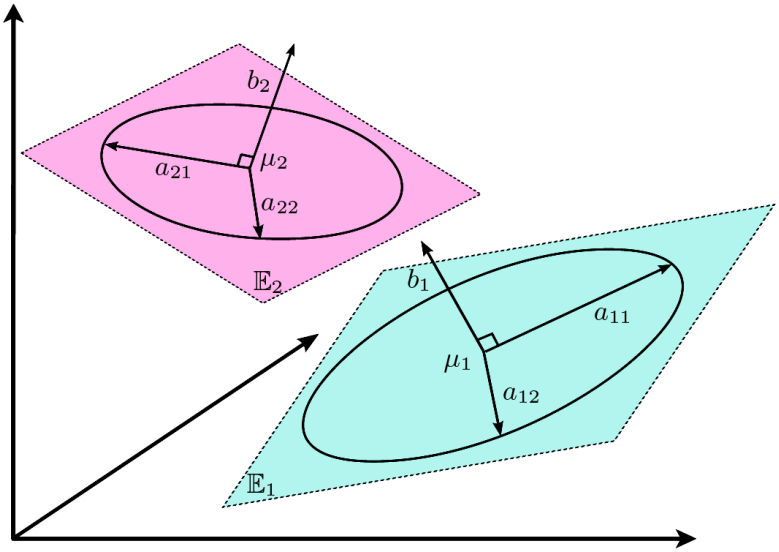


Figure 4.1: Two-dimensional subspaces, $\mathbb{E}_1$ and $\mathbb{E}_2$, for each of the two groups of a GMM of three-dimensional data. Variance in each group-specific subspace is given by the parameters a, while the noise variance is given by the parameters b. The noise variance for each group is contained in a subspace orthogonal to the subspace of the group (Bouveyron et al., 2007b).

d_k is the intrinsic dimension of the data for the group k. The matrices Q_k define the orientation of the k-th group. Figure 4.1 illustrates the role of the parameters a_{kj} and b_k graphically. With this parametrization, these parsimonious models assume that, conditionally to the groups, the noise variance of each cluster k is isotropic and is contained in a subspace which is orthogonal to the subspace of the kth group (see Figure 4.1).

The parameters of the model are $\boldsymbol{\mu}_k$, Σ_k, Q_k, a_{kj} and b_k, with $j = 1, \ldots, d_k$ and $k \in \{1, \ldots, K\}$. Parameter estimates can be obtained using the EM algorithm. In the E-step, we calculate the posterior probabilities of the classes given the current estimate of the parameter values and in the M-step we estimate the parameters given the posterior probabilities. The intrinsic dimension d_k is determined using the empirical scree test of Cattell (Cattell, 1966), which searches for a changepoint in the eigenvalues.

By imposing appropriate constraints on the model parameters Q_k, a_{kj}, b_k and the intrinsic dimension d_k, one can obtain a family of different models. For instance, if we set $d_k = d$, $\forall k = \{1, \ldots, K\}$, then we obtain the popular Mixture of Probabilistic Principal Component Analyzers (MPPCA) model of (Tipping and Bishop, 1999a,b). For a full list of models, the reader can refer to (Bouveyron et al., 2007a). Figure 4.2 shows the form of different models in an illustrative two-dimensional example. Note that the notation proposed in (Celeux et al., 1996) is used to distinguish between the models. Thus, the most general one is written as $[a_{kj}b_kQ_kd_k]$, which means that no constraints are imposed on the parameters. Similarly, the MPPCA model can be written as $[a_{kj}b_kQ_kd]$.

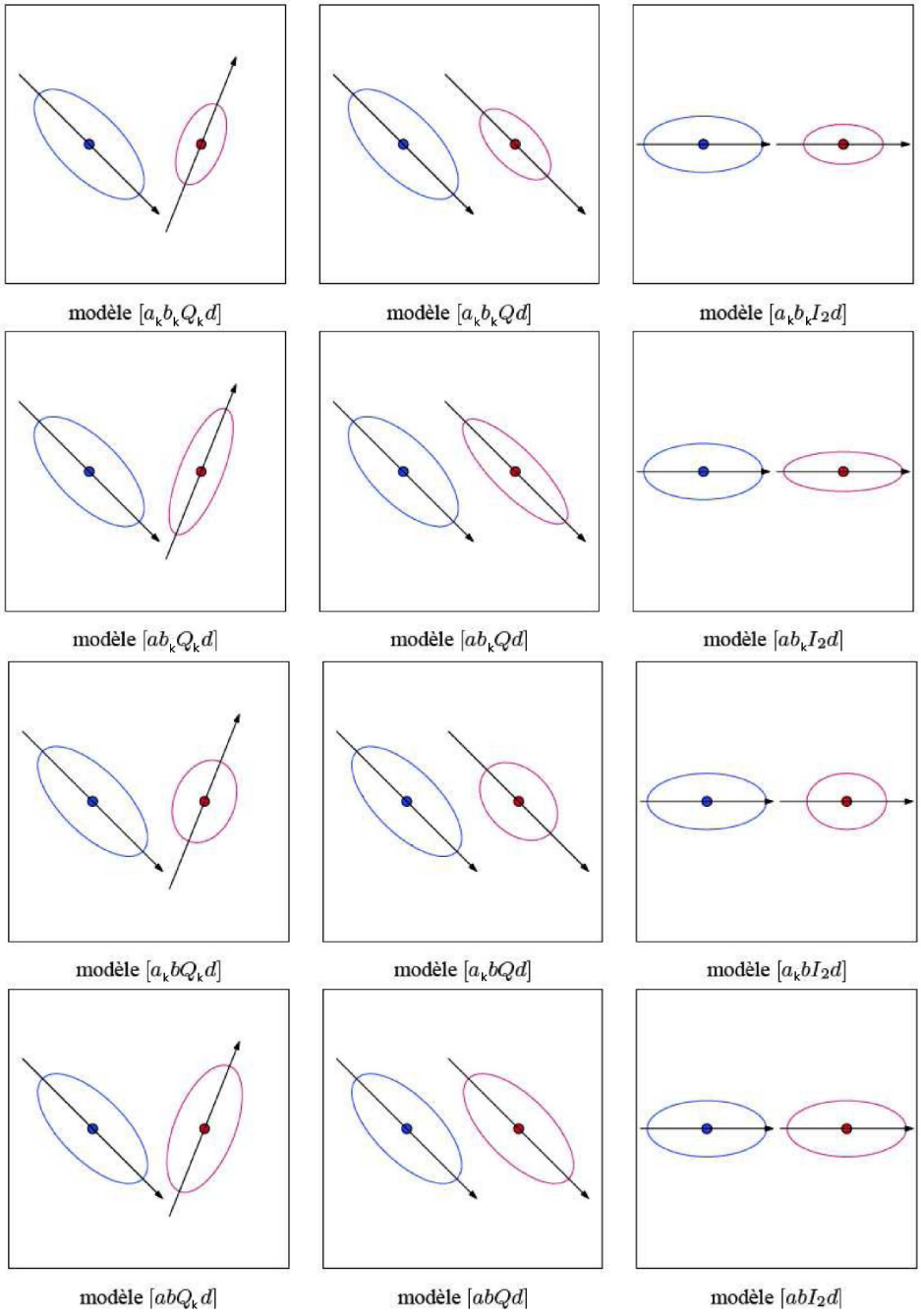


Figure 4.2: The influence of parameters on the class densities (Bouveyron, 2006).

We saw earlier that in order to cluster data in the model-based setting, one uses the MAP rule on the posterior probabilities. The aforementioned rule assigns each data sample to the class with the highest posterior probability. It can be shown (Bouveyron et al.,

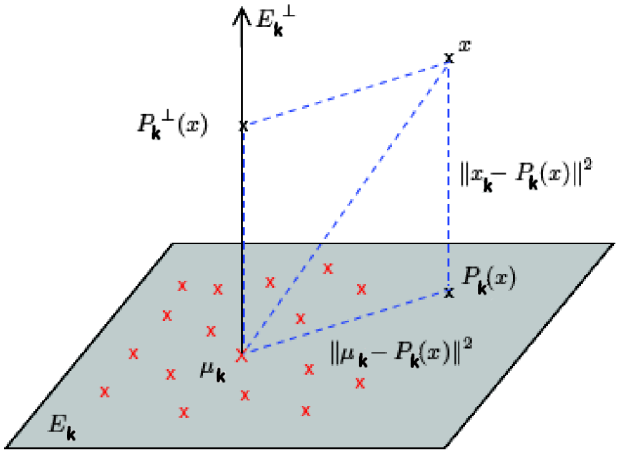


Figure 4.3: Subspaces $\mathbb{E}_k$ and $\mathbb{E}_k^{\perp}$ associated with the k-th group (Bouveyron et al., 2007b). Here, $P_k(\mathbf{x})$ is the projection of the data sample $\mathbf{x}$ on the subspace associated with group k.

2007a) that the classification step of the EM algorithm estimating the parameters of the model can be expressed through a classification cost function $\Gamma_k(\mathbf{x})$. Naturally, its exact expression depends on the form of the model. For instance, for the most general model $[a_k j b_k Q_k d_k]$, we have

$$\begin{aligned} \Gamma_k(\mathbf{x}) &= -2\log(f(\mathbf{x}, \boldsymbol{\theta}_k)) \\ &= \|\boldsymbol{\mu}_k - P_k(\mathbf{x})\|^2_{\mathcal{A}_k} + \frac{1}{b_k}\|\mathbf{x} - P_k(\mathbf{x})\|^2 + \sum_{j=1}^{d_k} \log(a_{kj}) + \\ &\quad (p - d_k)\log(b_k) - 2\log(\pi_k) \end{aligned} \tag{4.4}$$

with $\|\mathbf{x}\|^2_{\mathcal{A}_k} = \mathbf{x}^t \mathcal{A}_k \mathbf{x}$,$\mathcal{A}_k = Q_k \Lambda_k^{-1} Q_k^t$ and $P_k(\mathbf{x}) = Q_k Q_k^t(\mathbf{x} - \boldsymbol{\mu}_k) + \boldsymbol{\mu}_k$ is the projection of the data to the subspace of the k-th group. Cost functions for the other models are given in (Bouveyron et al., 2007a).

Figure 4.3 illustrates the utility of the cost function for the clustering task. As we see in Equation (4.4), its value depends on two distances: $\|\boldsymbol{\mu}_k - P_k(\mathbf{x})\|_{\mathcal{A}_k}$ which depends on $\mathcal{A}_k$ and $\frac{1}{b_k}\|\mathbf{x} - P_k(\mathbf{x})\|^2$, which is the Euclidean distance of $\mathbf{x}$ to its projection on the subspace of the group to which it was assigned. Thus, the classification function $\Gamma(\mathbf{x})$ (Equation 4.4) will assign a data sample to the class whose subspace is nearest to this sample and the projection of which is the nearest to the center of the class.

We remark that by using the above formulation, we do not have to explicitly inverse the group covariance matrices, a procedure which tends to be complicated when the matrix is ill-conditioned. Indeed, the variance of the classes enters the model through the parameters a and b and not the covariance matrix itself. Moreover, the projection of the data on the subspace orthogonal to the group specific subspace does not appear

Algorithm 3 Pseudo-code for the HDDC algorithm

Input: Data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathbb{R}^p$, number of groups K, threshold ε, maximum iteration number $maxit$

Output: Estimated parameters $\hat{\theta}$

1: Initialize to acquire an initial estimate $\hat{\theta}_0$.
2: $q = 1$
3: **while** $\mid L^{(q)} - L^{(q-1)} \mid \geq \varepsilon$ OR $q < maxit$ **do**
4: **E-step**

$$t_{ik} \quad = \quad P\left(Z = k \mid X = \mathbf{x}_i, \hat{\theta}^{(q-1)}\right)$$

5: **M-step** Update the estimates to obtain $\hat{\theta}^{(q)} = \left(\hat{\pi}_k, \hat{\boldsymbol{\mu}}_k, \hat{a}_{kj}, \hat{b}_k, \hat{Q}_k, d_k\right)$

$$\begin{aligned}
d_k &: \text{Cattell's test to find a changepoint in the eigenvalues} \\
\hat{\pi}_k &= \frac{n_k}{n} \\
\hat{\boldsymbol{\mu}}_k &= \frac{1}{n_k}\sum_{i=1}^{n} t_{ik}\mathbf{x}_i \\
\hat{a}_{kj} &= \lambda_{kj} \\
\hat{b}_k &= \frac{1}{p-d_k}\left(\mathrm{tr}(S_k) - \sum_{j=1}^{d_k}\lambda_{kj}\right) \\
\hat{Q}_k &: \text{the eigenvectors associated with the } d_k \text{ largest eigenvalues of } S_k
\end{aligned}$$

where $n_k = \sum_{i=1}^{n} t_{ik}$, $n = \sum_{k=1}^{K} n_k$ and $S_k = \frac{1}{n_k}\sum_{i=1}^{n} t_{ik}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^T$.
6: Evaluate the likelihood function $L^{(q)}$.
7: $q = q + 1$
8: **end while**

in the function. This means that we do not have to calculate the $p - d_k$ last columns of the group orientation matrices Q_k, $k = \{1, \ldots, K\}$.

The detailed EM algorithm for $[a_{kj}b_kQ_kd_k]$, the most general model, is given as an example (Algorithm 3). Naturally, different models have different update equations (M-step) which can be found in (Bouveyron et al., 2007a).

4.2 High-Dimensional Robust Clustering

In the previous section, we saw how subspace clustering methods, like HDDC, can cope with high dimensionality. However, if the data contain anomalies, parameters estimates of model-based clustering methods will be corrupted. Indeed, this is a problem that most traditional clustering algorithms face. For instance, the popular k-means algorithm (MacQueen et al., 1967), is very sensitive to anomalies and will, thus, behave

poorly when confronted to such data. A solution to this shortcoming is to robustify clustering by trimming outlying (abnormal) data samples in a straightforward manner. In this section, we first give a short introduction to robust clustering by presenting relative existing works. We then introduce the *High-Dimensional Robust Clustering* (HDRC) algorithm that can cope with high-dimensional data with anomalies. In the last part of the section, we present experiments using HDDC and relative results.

4.2.1 Introduction to robust clustering with trimming

The simplest trimming example is the trimmed mean which removes a proportion $\alpha/2$ of the largest and the smallest data samples before calculating the sample mean.

Early works include the least trimmed squares (LTS) and least median squares (LMS) for regression (Rousseeuw and Leroy, 2005), as well as the Minimum Covariance Determinant (MCD) and Minimum Volume Ellipsoid (Rousseeuw, 1985), which are both location-scatter estimators (single-class case) using the trimming principle.

(Cuesta-Albertos et al., 1997) developed a trimmed k-means. While standard k-means (MacQueen et al., 1967) optimizes the criterion

$$\arg \min_{m_1,\dots,m_K} \sum_{i}^{n} \min_{j=1,\dots,K} \|\mathbf{x}_i - m_j\|^2$$

where $(m_1, \dots, m_K)$ are the centers of the K clusters and n the number of data samples, the trimmed k-means of (Cuesta-Albertos et al., 1997) solves a double optimization problem

$$\arg Y \min_{m_1,\dots,m_K} \sum_{\mathbf{x}_i \in Y}^{n} \min_{j=1,\dots,K} \|\mathbf{x}_i - m_j\|^2$$

where Y ranges on the class of subsets of size $[n(1-\alpha)]$, *i.e.* of all possible normal datasets among the data and α is the proportion of the data to be trimmed. (García-Escudero et al., 1999) present results on robustness properties of the trimmed k-means compared to the standard k-means.

A well-known limitation of standard k-means is that it cannot handle heterogeneous clusters since it assumes that the covariance matrices for all clusters are equal to $\Sigma_1 = \ldots = \Sigma_k = \sigma^2 I$, where I is the identity matrix. (Gallegos, 2002; Gallegos and Ritter, 2005) have developed a probabilistic clustering model for heterogeneous clustering called "spurious-outliers" model, which maximizes the log-likelihood of the "normal" data whose indexes, for each cluster $j = 1, \dots, K$ are in R_j

$$\max \sum_{j=1}^{K} \sum_{i \in R_j} \log \phi(\mathbf{x}_i; \mu_j, \Sigma_j), \tag{4.5}$$

under the constraint $\# \cup_{j=1}^{K} R_j = [(1-\alpha)n]$. Here $\phi(\cdot)$ is the Gaussian density.

Unfortunately, the maximization in (4.5) is an ill-posed problem since the function is unbounded. If one of the determinants of the group covariance matrices goes to 0,

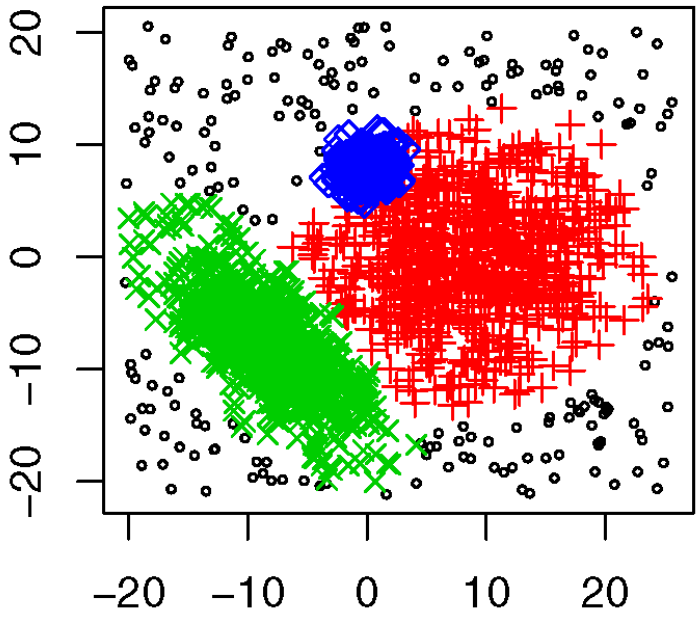


Figure 4.4: TCLUST (García-Escudero et al., 2008) on a simulated 2D dataset with $\alpha = 0.1$. Black circles are the data samples that TCLUST trimmed.

the function goes to ∞. Therefore, it is necessary to impose constraints on the group covariance matrices to eliminate this inconvenience.

In this direction, (Gallegos, 2002) proposed a method based on unit group covariance matrices, while (Gallegos and Ritter, 2005) assume that groups share a common covariance structure. (García-Escudero et al., 2008) propose TCLUST, a GMM-based trimming method imposing a constraint on the ratio of the largest to the smallest eigenvalue among eigendecomposition matrices of all group covariance matrices. (Fritz et al., 2012a) have recently proposed a faster variant of the TCLUST algorithm. A comprehensive review of the above works is given in (García-Escudero et al., 2010).

(Neykov et al., 2007) propose the use of the Weighted Trimming Likelihood Estimator (WTLE). The goal is to estimate parameters and the data samples to trim so that a weighted version of the negative log-likelihood of the model is minimized. This problem is infeasible for large data sets and a general method to approximate its solution is given in (Müller and Neykov, 2003).

A problem with trimming methods is that the proportion of data samples to trim, α, has to be defined by the user. In practice, this is never known in advance. Moreover, the data samples that will be actually trimmed depend on the clustering done. For TCLUST to trim the true outliers, an assumption of "well-clusterized" data is needed. The only free parameter related to clustering is K, the number of groups or classes. Thus, α and K have to be chosen simultaneously. For TCLUST, there is also the parameter c that determines the strength of the constraint on the eigenvalue ratio. However, it may be set following domain or expert knowledge.

The "forward search" (Atkinson et al., 2006) which is based on random starts is a robust criterion for choosing K. (Gallegos and Ritter, 2009) propose to use BIC (Schwarz, 1978) corrected by goodness-of-fit χ^2 tests on the mixture groups in order to determine the value of K. (Neykov et al., 2007) propose a graphical tool to select values of the trimming parameter α and number of groups K, by monitoring the value of the BIC criterion calculated using the trimmed likelihood of the model. (García-Escudero et al., 2008) propose an exploratory graphical method to choose K and α based on the trimmed log-likelihood of the model.

Recently, (Punzo and McNicholas, 2013) proposed a robust GMM clustering method that estimates α via the Expectation-Conditional Maximization algorithm (Meng and Rubin, 1993). They use mixtures of contaminated Gaussian distributions, which are themselves a mixture of two components: one that corresponds to normal data and another one for the anomalies. The latter has the same mean as the former but with an inflated variance. Thus, the authors assume a two-level mixture model. They use ECM to estimate the parameters. In the E-step, they calculate the posterior probabilities of the groups for each data sample, as well as the distribution of the binary variable v_{ki}, which is 0 if observation i in group k is "normal" and 0, otherwise. In the first conditional maximization step the means, covariance matrices, the parameter α and the mixture proportions for the second-level of the GMM are being estimated for *each* contaminated Gaussian. In the second conditional maximization step, the inflation parameter of the anomaly component in each contaminated Gaussian is estimated. Note that there is an α per class, that is, α_k, $k = \{1, \ldots, K\}$ instead of a global one as in all previous works.

4.2.2 High-Dimensional Robust Clustering

From the overview given in the previous section, it is clear that those methods cannot handle high-dimensional data. Indeed, the curse of dimensionality will result in estimates being corrupted.

We introduce a novel algorithm, called *High-Dimensional Robust Clustering* (HDRC), which combines HDDC and the trimming technique, therefore resulting in a robust clustering algorithm which is efficient for high-dimensional data.

In particular, we extend HDDC by adding an intermediate trimming step (T-step) between the E step and the M step of the EM part of the algorithm:

- At iteration q, the E step computes the posterior probabilities

$$t_{ik}^{(q)} = \mathbb{P}(Z = k | X = \mathbf{x}_i)$$

for $i = 1, \ldots, n$ and $k = 1, \ldots, K$ through the formula

$$t_{ik}^{(q)} = \frac{1}{\sum_{\ell=1}^{K} \exp\left(\frac{1}{2}(\Gamma_k^{(q)}(\mathbf{x}) - \Gamma_\ell^{(q)}(\mathbf{x}))\right)}$$

where the classification function $\Gamma_k^{(q)}$ has the following form when $a_{kj} = a_k$, $k = 1, ..., K$, $j = 1, ..., d_k$, that is, for the model $[a_k b_k Q_k d_k]$:

$$\begin{aligned}\Gamma_k^{(q)}(\mathbf{x}) &= \frac{1}{a_k}\|\mu_k - P_k(\mathbf{x})\|^2 + \frac{1}{b_k}\|\mathbf{x} - P_k(\mathbf{x})\|^2 \\ &\quad + d_k \log(a_k) + (p - d_k)\log(b_k) - 2\log(\pi_k)\end{aligned}$$

Here, P_k is the projection operator on the subspace of the k-th group and model parameters are estimated in the M-step at iteration $q - 1$.

- The T step trims a proportion α of the data samples with smallest values for

$$\max_{k=1,...,K} \pi_k^{(q)} \phi\left(\mathbf{x}; \boldsymbol{\mu}_k, \Sigma_k\right)$$

It can be shown that this is equivalent to trimming the data samples with the largest values for

$$\min_{k=1,...,K} \Gamma_k^{(q)}(\mathbf{x})$$

Let $R^{(q)}$ be the set of the trimmed data samples.

- The M step then updates the estimates of model parameters by maximizing the expectation of the *trimmed* complete likelihood conditionally to the posterior probabilities $t_{ik}^{(q)}$ for the data samples $x_i \notin R^{(q)}$. Update formulas for the parameters can be found in (Bouveyron et al., 2007a). As an example, we give the update equations for the $[a_k b_k Q_k d_k]$ model, where the variance in each subspace is the same for all its dimensions

$$\begin{aligned}\hat{\pi}_k &= \frac{n_k}{n(1-\alpha)} \\ \hat{\boldsymbol{\mu}}_k &= \frac{1}{n_k}\sum_{i=1}^{n(1-\alpha)} t_{ik}\mathbf{x}_i \\ \hat{a}_k &= \frac{1}{d_k}\sum_{j=1}^{d_k}\lambda_{kj} \\ \hat{b}_k &= \frac{1}{p-d_k}\left(\mathrm{tr}(S_k) - \sum_{j=1}^{d_k}\lambda_{kj}\right) \\ S_k &= \frac{1}{n_k}\sum_{i=1}^{n(1-\alpha)} t_{ik}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^T \\ Q_k &: \text{the eigenvectors of } S_k \text{ associated with the } d_k \text{ largest eigenvalues} \\ d_k &: \text{scree test of Cattell}\end{aligned}$$

where S_k is the empirical covariance matrix for group k, $n_k = \sum_{i=1}^{n(1-\alpha)} t_{ik}^{(q)}$.

4.2.3 Model selection and hyperparameter calibration

As we mentioned earlier, K, the number of mixture groups, and α, the anomaly proportion in the data, are interdependent. More precisely, the clustering given by a robust clustering algorithm and the data samples that it will trim depend one on another. A change in K can lead the trimming algorithm to consider that an up-to-here normal data sample is no longer well explained by the model and can be abnormal, or the other way around. Moreover, if we change α then this can immediately alter the clustering given by the algorithm, since it is possible that it trims some other data samples. Therefore, as we saw earlier, some authors (García-Escudero et al., 2008; Neykov et al., 2007) suggest that K and α be chosen simultaneously. Two empirical techniques have been proposed for this purpose in the aforementioned works. They are based on a trimmed BIC criterion (Neykov et al., 2007) and a trimmed likelihood criterion (García-Escudero et al., 2008).

Trimmed BIC

In order to present the technique proposed by (Neykov et al., 2007), we are going to use the example they use in their work. For each possible combination of $K = 1, \ldots, 5$ and $\alpha = 0, \ldots, 0.45$, they run their algorithm multiple times on the data. They use a dataset used by (McLachlan and Peel, 2000), composed of 3 bivariate Gaussian components. For each one of these executions, they calculate the value of the BIC_t criterion, *i.e.* BIC *after trimming*:

$$\text{BIC}_t = -2 \log L_t + v \log n_t. \tag{4.6}$$

where L_t is the maximized likelihood for the n_t *non-trimmed* data samples and v is the number of parameters of the model. For each combination of K and α, the authors keep the median BIC_t value over all the executions of the algorithm. These median values can be found in Figure 4.5a.

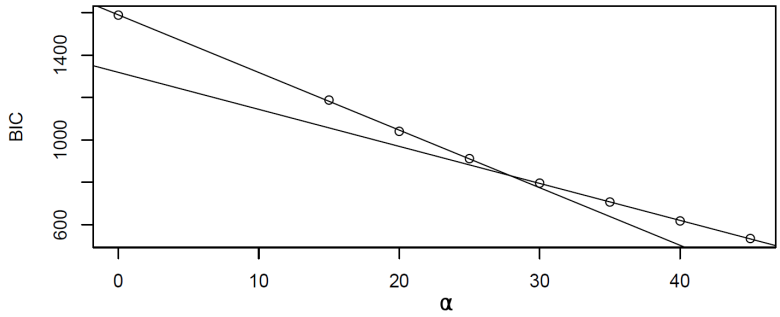
Then, they take the minimum of each column (each value of α), denoted in *italics* in the table and plot it versus its respective α value (Figure 4.5b). Thus, each point in Figure 4.5b is the minimum BIC_t value over all possible values of K for a given α. The dashed line shows the changepoint around $\alpha = 30\%$. The values of BIC_t seem to be stabilizing after $\alpha = 30\%$ and for $K = 3$. Therefore, according to the proposed technique, the authors choose $K = 3$ and $\alpha = 0.3$. As they state, the real anomaly proportion in this specific dataset is slightly higher but some of the anomalies fit the mixture.

Trimmed classification likelihood

A similar technique has been proposed in (Fritz et al., 2012b), where use of TCLUST algorithm is being made. The difference with (Neykov et al., 2007) is that on the one hand, (Fritz et al., 2012b) use an underlying clustering algorithm (TCLUST) which is more sophisticated and on the other hand, their technique is based on the trimmed log-likelihood rather than BIC.

	0%	15%	20%	25%	30%	35%	40%	45%
1	1664.3	1259.7	1130.0	1006.6	917.1	837.6	749.2	654.9
2	1650.9	1211.9	1056.6	927.8	822.8	738.2	646.6	560.5
3	1589.4	1197.6	*1040.5*	*911.9*	*796.4*	*707.3*	*618.0*	*535.3*
4	1598.3	*1188.0*	1051.7	912.3	807.5	718.9	630.6	545.6
5	*1588.5*	1193.0	1054.7	932.3	833.2	745.0	654.4	570.8

(a)



(b)

Figure 4.5: (a) Table of BIC_t for different values of K and α ((Neykov et al., 2007)). See the text for more details. (b) Minimum BIC_t values for each α. The dashed line shows the changepoint of the curve.

The authors generate a dataset of $K = 2$ classes with the true anomaly proportion at $\alpha = 5\%$. They run TCLUST with all possible combinations of values for K and α in a given, user-defined range. Note that they do not replicate the experiment multiple times for each combination, as with the trimmed BIC technique. They then plot the trimmed likelihood versus the values of α for each $K = 1, 2, 3, 4$ (Figure 4.6). Thus, there is as many curves as different values of K.

Figure 4.6 shows that right after $\alpha = 0.05$, the curves for $K = 2$, $K = 3$ and $K = 4$ "converge". Formally, for $\alpha \geq 0.05$, one observes that

$$L_t^{(3)} - L_t^{(2)} \approx 0 \approx L_t^{(4)} - L_t^{(3)} \tag{4.7}$$

where $L_t^{(2)}$ is the trimmed likelihood function value for $K = 2$ and so on. The authors choose the smallest value of α for which (4.7) holds. In this case, it is $\alpha = 0.05$ which is, indeed, the true proportion of anomalies among the data. As for K, they choose the smallest value among all values of K for which (4.7) holds for the given value of α. In this case, that would be $K = 2$, which is, indeed, the true number of groups.

Note, however, that for the experiments with this technique in the following sections we use HDRC instead of TCLUST.

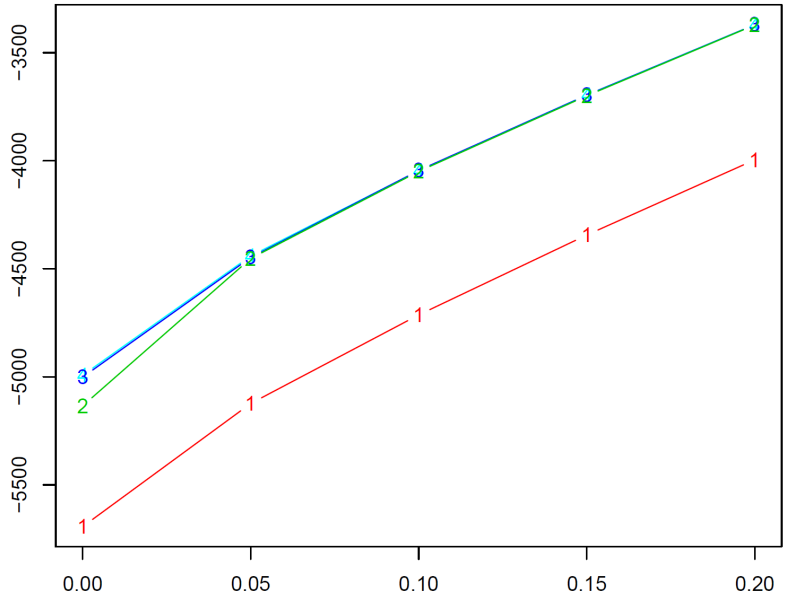


Figure 4.6: Example of the trimmed classification likelihood curves versus the values of the trimming parameter α, as proposed in (García-Escudero et al., 2008). Each curve corresponds to a different value of $K = 1, 2, 3, 4$. Curves for $K = 2, 3, 4$ become superposed starting from $\alpha = 0.05$, leading to the choice $K = 2$ and $\alpha = 0.05$, which are the true values of the parameters for this example.

4.3 Experiments

We run some experiments to test HDRC on simulated high-dimensional data and real data: breast cancer data from the UCI machine learning repository[1] and industrial data from the domain of aircraft engine health monitoring supplied by Snecma, the french aircraft engine constructor.

4.3.1 Evaluation measures

In our experiments, we are interested in measuring the performance of a classifier on high-dimensional data and in particular: clustering performance on "normal" data, as well as anomaly detection and false positive rates. Therefore, we calculate the clustering accuracy *acc* (up to label switching) as the ratio of correct cluster assignments to the size of the dataset

$$acc = \frac{\text{number of data samples correctly classified}}{\text{number of data samples}}$$

[1] Obtained by the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. Original paper using the data (Mangasarian et al., 1995).

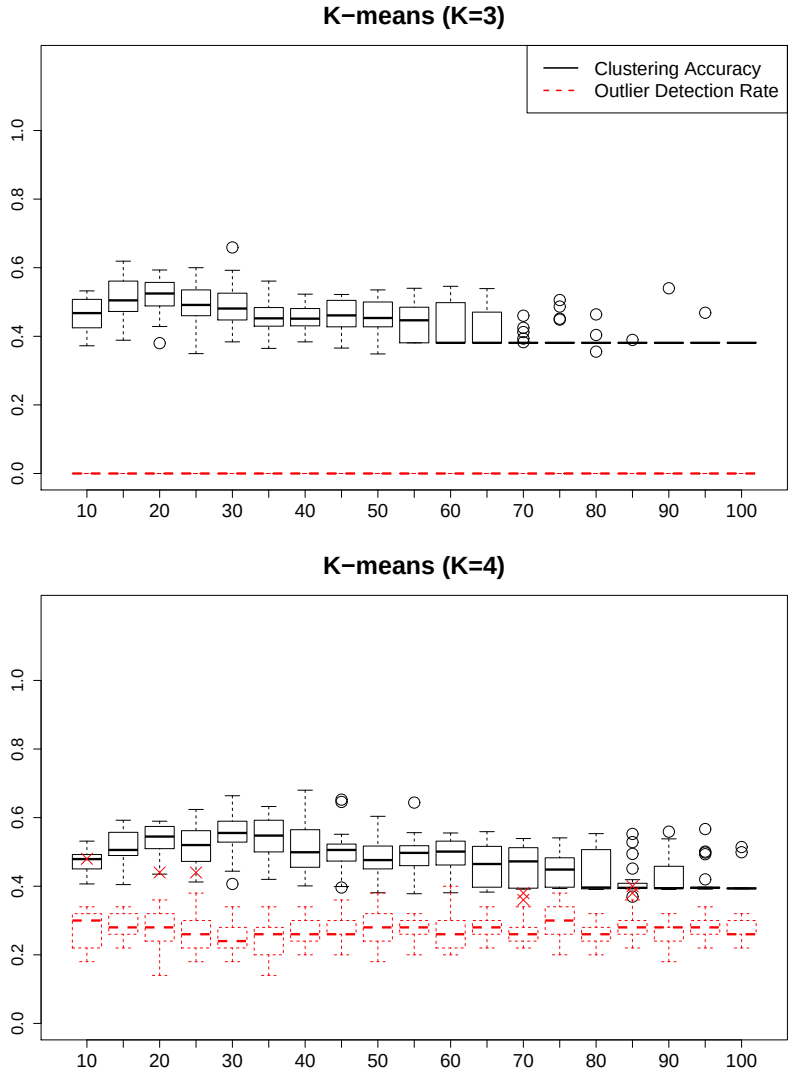


Figure 4.7: Clustering accuracy *acc* (black solid boxes) and *tpr* of the anomalies (red dashed boxes) for the simulated dataset, plotted versus the dimension of the data. Black circles correspond to extreme observations for the clustering accuracy, and red crosses to those for the *tpr* of the anomalies. For K-means with K=3 the *tpr* is artificially set to zero (see the text for details).

the true positive rate for the anomalies (*tpr*), *i.e.*, the anomaly detection rate

$$tpr \quad = \quad \frac{\text{number of detected anomalies}}{\text{number of anomalies}}$$

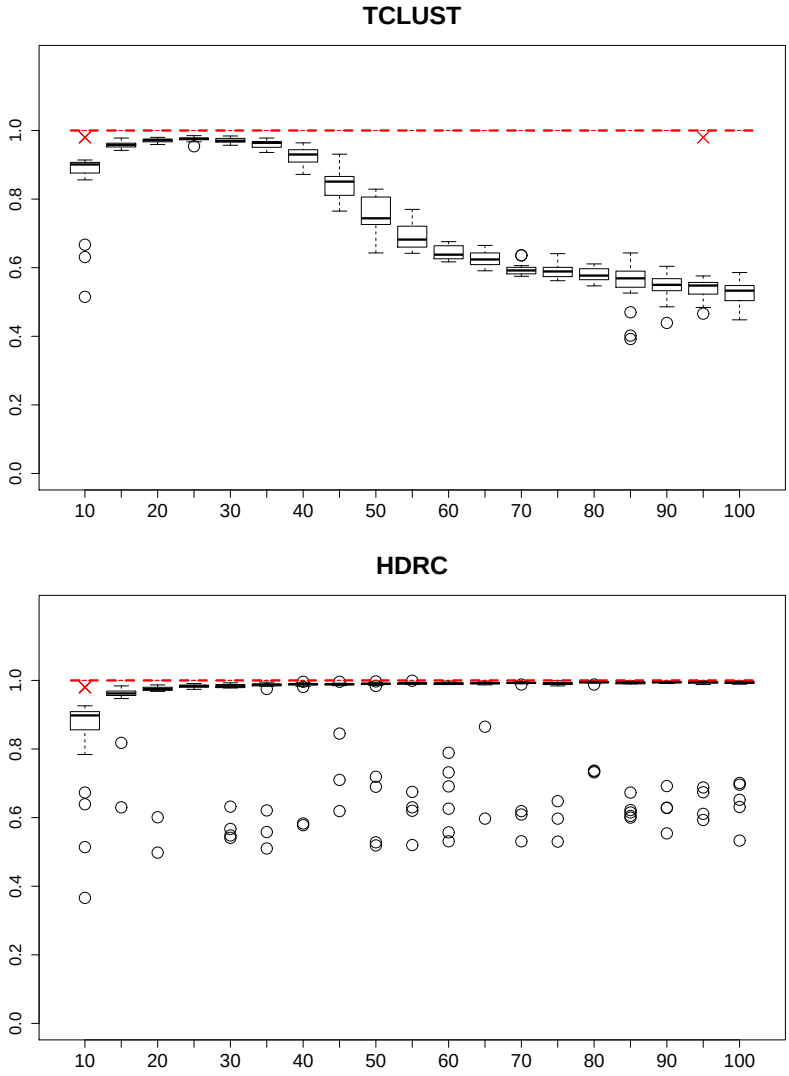


Figure 4.8: Clustering accuracy *acc* (black solid boxes) and *tpr* of the anomalies (red dashed boxes) for the simulated dataset, plotted versus the dimension of the data. Black circles correspond to extreme observations for the clustering accuracy, and red crosses to those for the *tpr* of the anomalies.

as the ratio of data points correctly detected as anomalies to the true number of anomalies, as well as the false positive rate (fpr)

$$fpr = \frac{\text{number of "normal" data samples flagged as anomalies}}{\text{number of "normal" data samples}}$$

as the ratio of "normal" (non anomalous) data flagged as anomalies by the classifier to the total number of normal data.

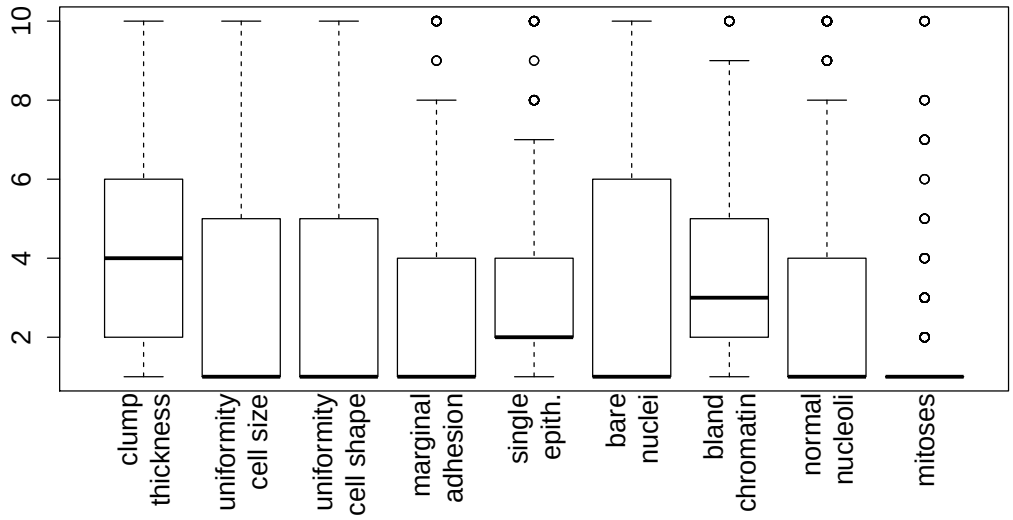


Figure 4.9: Boxplots of the variables of the breast cancer data.

4.3.2 Application to simulated data

For the simulated dataset, we generated 3 multivariate Gaussians with a total of 1000 data points according to the HDDC model in the p-dimensional space, where $p = 10, \ldots, 100$. The means and covariance matrices were chosen such that the task would not be too challenging for the algorithms tested. In particular, we set the parameters as follows: $a_{1j} = 150$ and $b_1 = 15$, $a_{2j} = 75$ and $b_2 = 10$, $a_{3j} = 50$ and $b_3 = 5$, respectively, for the three Gaussians, where $j = 1, \ldots, d_k$. The intrinsic dimensions d_k were $d_1 = 10$, $d_2 = 5$ and $d_3 = 2$ for the three groups. We added a quantity of anomalies equal to 5% ofthe dataset size, uniformly distributed on the interval $[-40, 40]$ for each of the p variables.

In the experiments, we tested K-means with $K = 3$ and $K = 4$, TCLUST and HDRC. K-means was used as a baseline; we wanted to examine its behavior when it has no knowledge of the existence of anomalies ($K = 3$) and when this knowledge is given explicitly by adding an extra group ($K = 4$). We proceed to this experiment since it could be argued that by adding an extra cluster, k-means would succeed in detecting the anomalies, in fact, that it would actually put all anomalies in this extra cluster.

For TCLUST, we restrict the *maximum* value for the ratio of the *maximum* to the *minimum* eigenvalues among all group covariance matrices to be $c = 50$ for the dimension $p > 10$. For HDRC we used a random initialization. For TCLUST and HDRC, we set the number of groups to $K = 3$ and gave both the true anomaly proportion, that is $\alpha = 0.05$. For all the algorithms, the number of initializations was set to 25 and the maximum number of iterations to 60. For each value of p, we replicated the experiment 25 times.

	tpr (%)	**fpr** %
hdrc	$92,6$	$3,3$
tclust	$92,2$	$3,5$

Table 4.1: Mean values for tpr and fpr over all replications of the experiment for HDRC and TCLUST.

Figures 4.7-4.8 present the clustering accuracy and the true positive rate for the anomalies for our experiments. We observe that k-means with $K = 3$ fails in clustering correctly the data. Indeed, the clustering is corrupted by the anomalies. Note that $K = 3$ indicates that k-means "naively" tries to cluster data with anomalies without being aware of their presence and that is why we did not evaluate the tpr (artificially set to zero). We also observe that even when an extra group is added with the aim of modeling anomalies ($K = 4$), k-means does not do much better. As expected, TCLUST succeeds in detecting the anomalies in all cases but appears to be sensitive in high dimension. The way we simulated data, the anomalies have a large variance and thus, it should have been easy to detect them correctly. This means that the mediocre performance of TCLUST in clustering is, to a great extent, due to the high dimensionality of the data. Finally, we see that HDRC successfully manages to cluster the data correctly and detect the anomalies even in high dimension.

4.3.3 Application to breast cancer diagnosis

The breast cancer data contain measurements taken from patients presenting some kind of breast tumor (benign or malignant). The dataset is composed of 699 data samples of 9 attributes each. The attributes are the following: clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, mitoses. All attributes are integer-valued. Data belong to two classes: benign (coded by 2) and malignant (coded by 4). There are 458 (65.5%) data samples belonging to the benign class and 241 (34.5%) to the malignant class. We have removed 17 data samples with missing values, obtaining a dataset of 682 data samples.

In this case, we used HDRC as a one-class classifier, since the algorithm was asked to cluster data into the 'benign' class and to trim data samples corresponding to the

'malignant' case. To achieve this, we ran HDRC with $K = 1$ and we set $\alpha = 34.5\%$, *i.e.*, to the true proportion of the 'malignant' class in the data.We initialized the algorithm 25 times.

Table 4.1 shows the anomaly detection rates (tpr) and false detection rates (fpr) over 30 replications for HDRC and TCLUST. We can see that HDRC performs well for a real dataset attaining a mean of $92,6\%$ of detection with a fairly low false positive rate. HDRC performs equally well to TCLUST, which is the actual reference algorithm in the field of robust model-based clustering. Figure 4.10 shows breast cancer data on the two first principal components with clustering given by HDRC (Figure 4.10a) and groudtruth clustering (Figure 4.10b).

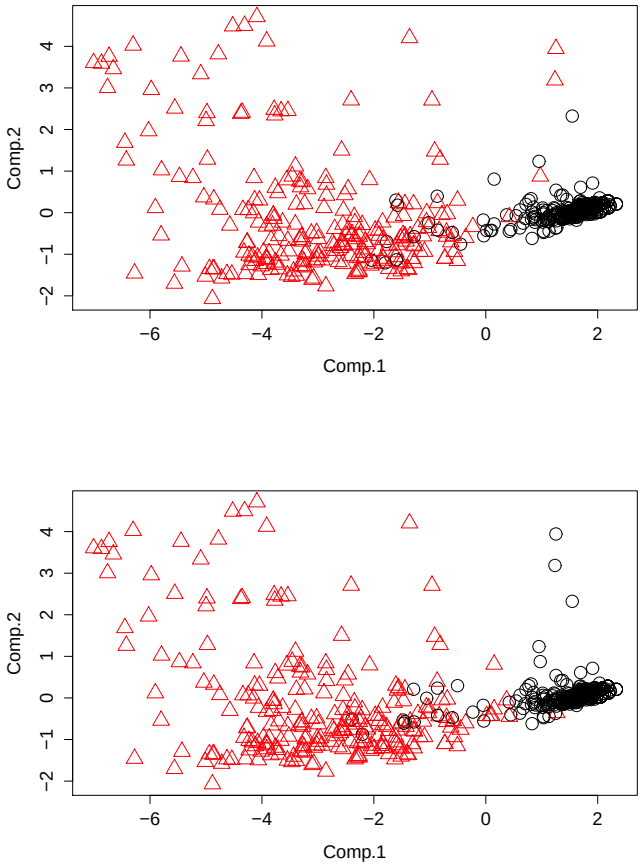


Figure 4.10: Breast cancer data on the first two principal components. Black circles are the 'benign' class and red triangles are the samples trimmed ('malignant') according to (top) HDRC (bottom) groundtruth class assignment. Proportion of explained variance: $65,6\%$ (Comp. 1) and $8,6\%$ (Comp. 2).

4.3.4 Application to aircraft engine health monitoring data

In this section, we present experiments on real-world data from the health monitoring of aircraft engines. We also discuss how to select hyperparameters K and α.

Introduction to the aircraft engine data

In the aircraft engine domain, the task of engine health monitoring is of crucial importance. Snecma, the French aircraft engine constructor, performs such tests in a test bench environment. During a bench test, the engine as well as the test cell itself are monitored using a wide set of sensors, which can attain even 1000 in numbers.

The measurements taken include performance measurements (pressures, temperatures,

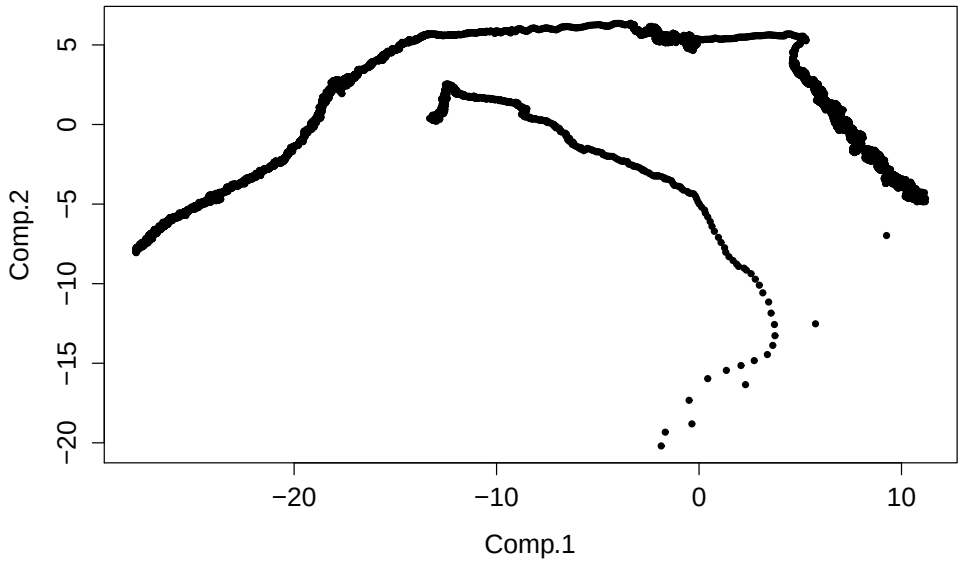


Figure 4.11: Aircraft engine data of dimension $p = 173$ from a test cell on the two first principal components. Proportion of explained variance: $81,5\%$ (Comp. 1) and $6,2\%$ (Comp. 2).

flows, gauges etc.) up to 100Hz, high frequency dynamic measurements (tip timing, accelerometers, microphones) up to 50kHz, and some context information that describes the test procedure. Snecma's test cells are monitored with a specific SPC (Statistic Process Control) tool that is able to register each sensor at different acquisition frequencies and output real time graphs with alert bounds. This SPC software also uploads all data to databases in real-time. A PHM (Prognostic and Health Monitoring) application is connected to the control system. Data is transfered from the control system to the monitoring system in real-time. The PHM system then makes use of the stored information in the database (Lacaille et al., 2010). We can see an illustration of this schema in Figure 4.12.

Aircraft engine anomaly detection

Some of the parameters measured are related to the environment of the test defined by external conditions and the test's pilot actions (aircraft speed), while others are internal parameters of the engine (inside temperature and pressure, rotation speed). The former are called exogenous or environmental variables, while the latter endogenous. For the anomaly detection task, we are typically interested in the endogenous variables. The goal of anomaly detection in the aircraft engine health monitoring domain is to be able to issue a warning whenever there is a malfunction (anomaly) of the engine or the test

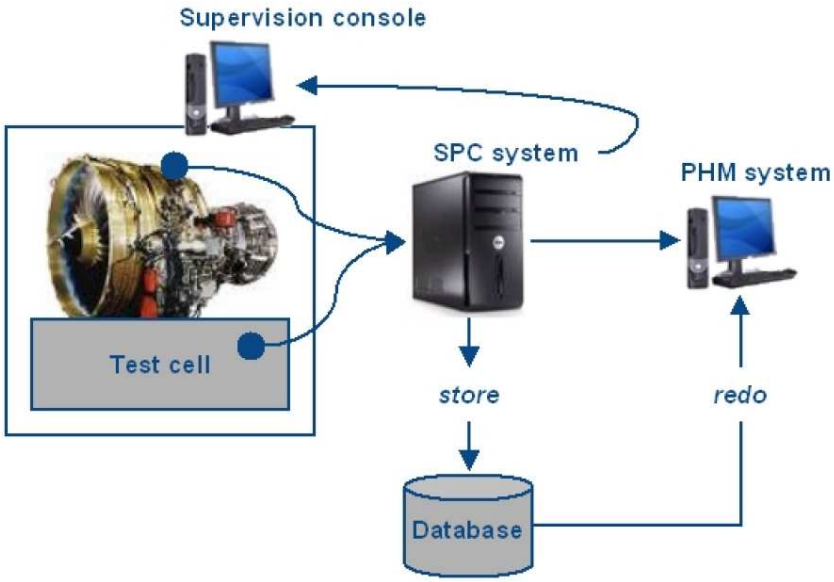


Figure 4.12: Snecma Test System Deployment (Lacaille et al., 2010)

cell, before significant damage occurs to any of the two.

The dataset we consider here consists of 46 830 observations and 173 variables of test cell data. Each test is a sequence of alternating stationary and non-stationary phases at different levels. Data on the first two principal components can be seen in Figure 4.11. The abrupt test stoppage at around $t = 45000$, designated by the red rectangle in Figure 4.13a, illustrates a case where the test pilot tried to change the flight phase but failed due to a malfunction, so he was forced to stop the test abrupty. A zoom-in on this area is given in Figure 4.13b.

We run HDRC to check if it will correctly detect the outlying data samples (anomalies). We empirically set $K = 6$ and gave HDRC $\alpha = 4 \times 10^{-5}$. The value of the trimming parameter α was set sufficiently small so that there would be no false alarms.

As we can see in Figure 4.13, HDRC succeeds in detecting the anomalies, which concentrate around the small "bump" at aproximately $t = 45\ 700$ in Figure 4.13b. Note that the clustering was performed in the p-dimensional space, where $p = 173$, and that we plotted only one of the variables in Figure 4.13 for visual clarity. This illustrates the ability of HDRC to cope with high-dimensional datasets.

4.3.5 Model selection and hyperparameter calibration on aircraft engine data

As we saw in Section 4.2, empirical methods of determining α and K have been proposed. We would like to test these methods to real data and in particular, the aforementioned aircraft engine Snecma data of original dimension $p = 173$, where we inject some artificial anomalies. However, Snecma experts do not just process all the 173 variables at once.

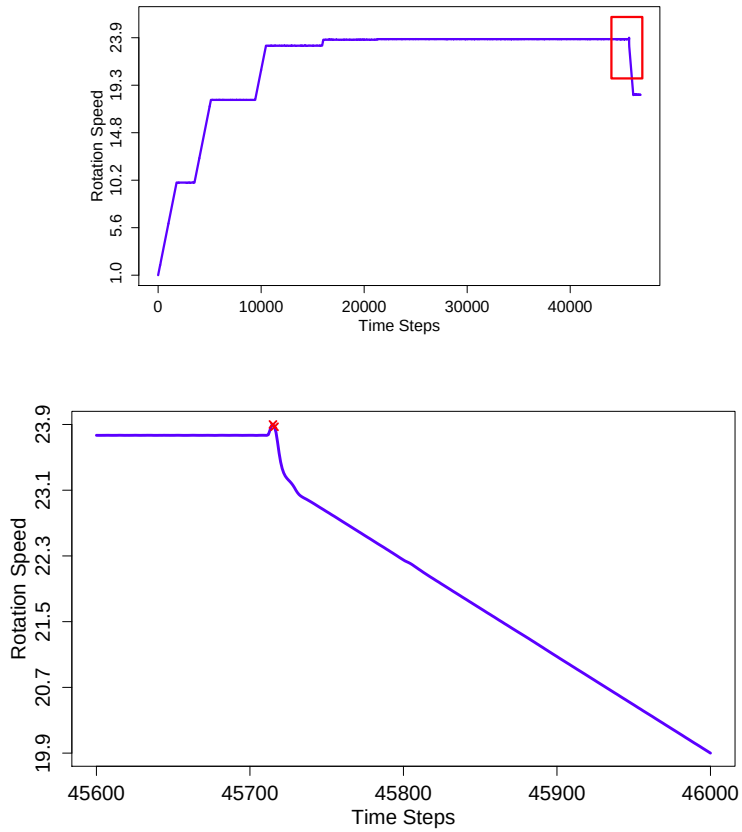
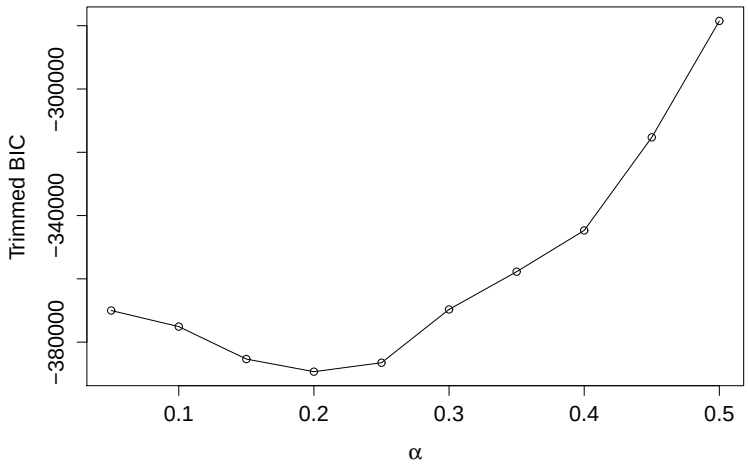


Figure 4.13: (top) Engine rotation speed. Red rectangle shows the area where the anomalies occured. (bottom) Zoom-in on the anomaly area: the litte "bump" corresponds to an engine malfunction. HDRC with $K = 6$ and $\alpha = 4 \times 10^{-5}$ detects successfully the anomalies (red crosses). Note that clustering was performed on the p-dimensional space with $p = 173$.

Instead, they focus on some variables of interest, where anomalies can appear, and which they first correct of the influence of factors that could deteriorate anomaly detection. Following this work methodology, we conducted this experiment in a slightly different manner.

In particular, we pick some variables of interest (endogenous) and we correct them from any environmental influence (due to context or environmental variables). To do this, we use a simple linear model, regressing the endogenous variables on the environmental ones. Then, the residuals are, in fact, the corrected endogenous variables. Variables of both categories were selected following advice from Snecma experts. Table 4.2 lists the variables used. Note that variables have been anonymized due to confidentiality reasons. In total, we used 10 variables, 6 of which are environmental and 4 endogenous. This means that our corrected dataset is of dimension $d = 4$.



	$\alpha = 5\%$	$\alpha = 10\%$	$\alpha = 15\%$	$\alpha = 20\%$	$\alpha = 25\%$	$\alpha = 30\%$
$K = 12$	-358442.5	-363482.7	-370652.5	-378900.0	-365138.9	-355207.3
$K = 13$	-364278.7	-370995.5	-378038.9	-378957.9	-375181.2	*-369667.2*
$K = 14$	*-370015.3*	*-375094.8*	*-385357.8*	*-389337.8*	*-386513.7*	-360908.8

Figure 4.14: Trimmed BIC method on Snecma data. Minimum BIC_t values plot (top) and corresponding table of median BIC_t values (bottom). The vertical red solid line in the plot shows the true anomaly proportion in the data. In the table, smaller values for each column are marked in *italics*. The experiment is non concluding since there is not a single, clear changepoint in the curve. This can also be verified at looking at the table.

Environmental variables	**Endogenous variables**
P1 (pressure)	T1 (temperature)
MOV1 (shift)	VIB2 (vibrations)
MOV2 (shift)	VIB3 (vibrations)
MOV3 (shift)	LOAD (load)
VIB1 (vibrations)	
N2 (rotation speed)	

Table 4.2: Environmental and endogenous variables used in the experiment.

Setting $X^{(1)} = \text{P1}$, $X^{(2)} = \text{MOV1}$, $X^{(3)} = \text{MOV2}$, $X^{(4)} = \text{MOV3}$, $X^{(5)} = \text{VIB1}$, $X^{(6)} = \text{N2}$ we can write the linear model, for each of the endogenous variables in $Y = [\text{T1, VIB2, VIB3, LOAD}]$ as

$$Y_i = \beta + \gamma_1 X_i^{(1)} + \gamma_2 X_i^{(2)} + \gamma_3 X_i^{(3)} + \gamma_4 X_i^{(4)} + \gamma_5 X_i^{(5)} + \gamma_6 X_i^{(6)} + \varepsilon_i$$

where $i = 1, \ldots, n$ is the index of the data samples and β is the intercept term.

We inject artificial anomalies to the variable T1, so that the proportion of anomalies in the data reaches 20%. We launch HDDC 25 times for all possible combinations of

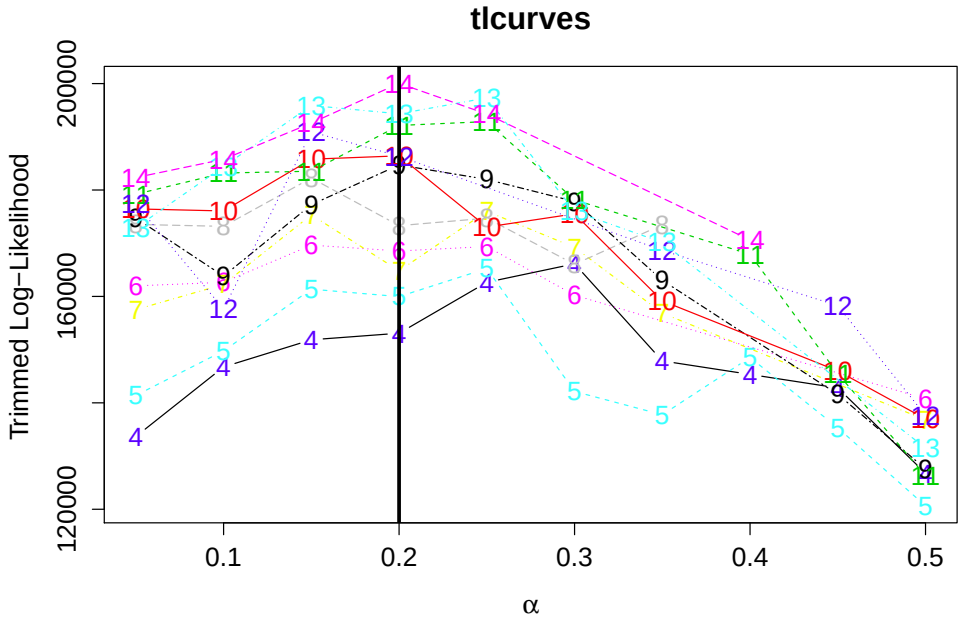


Figure 4.15: Trimmed log-likelihood method on Snecma data. Each curve corresponds to a particular value for K. The vertical solid line shows the true anomaly proportion in the data. The experiment is non concluding since there is no distinct and persistent curve superposition like for the synthetic data (see Figure 4.6).

$K = 4, 5, \ldots, 14$ and $\alpha = 0.05, 0.1, \ldots, 0.5$.

For the trimmed BIC technique, we can see the graphical result in Figure 4.14. Neither the form of the curve of minimum BIC_t values, nor the Table of median BIC_t values, allows us to reach to a safe conclusion regarding K and α. When we look at the median BIC_t values in the table of Figure 4.14, we can see a changepoint after $\alpha = 25\%$, for $K = 14$, but it is not the only one and it is not as clear as the one for the simulated data back in Figure 4.5). Note here that we ran tests for $K = 4, \ldots, 14$, but we only present results from $K = 12$ to $K = 14$, since they systematically contain the smallest BIC_t median values. The experiment is thus non concluding.

The result for the trimmed classification likelihood technique on the same dataset can be found in Figure 4.15. The experiment is also non concluding, since we cannot locate a distinct curve superposition as the authors did for simulated data back in Figure 4.6.

4.4 Conclusion and future work

In this chapter, we introduced the High-dimensional robust clustering (HDRC) algorithm, a robust subspace clustering algorithm which can cope with high-dimensional

data with anomalies. The proposed algorithm is based on the EM algorithm and robustness is added via the trimming technique, which simply trims, at each iteration, the less likely data samples, until convergence.

We have shown the effectiveness of HDRC on high-dimensional synthetic data of dimension up to $p = 100$. HDRC also succeeded in detecting abnormal behavior in real data of dimension $p = 173$ from the aircraft engine health monitoring domain. We also showed its use as a one-class classifier on the UCI breast cancer dataset.

Concerning the choice of the trimming parameter α, we experimented with two graphical methods designed for selecting the number of groups K and the trimming parameter α simultaneously. Unfortunately, both experiments were non-concluding, since it was not possible to safely choose values for the two parameters. We think that the solution of (Punzo and McNicholas, 2013) (see the introduction to robust clustering in Section 4.2) is able to solve the issue of choosing α in an elegant way. We are planning to run experiments to compare HDRC to this work in the near future.

Finally, HDRC is clearly an offline method, that is, it operates on a fixed dataset. The goal of this Thesis is to address the problem of online anomaly detection. In the next chapter, we take the first step in this direction, by developing an online (non robust) subspace clustering algorithm.

Chapter 5

Online mixture of PPCA

Recent advances in sensor technology, networks, databases and computational and storage power have made it possible to install and maintain complex systems for monitoring physical or industrial procedures. These systems emit measurements of interest at a high frequency for infinite (in practice, large) periods of time, forming long series of data, which are called *data streams*, or datastreams. In a typical example of such a system, there will be p sensors that emit simultaneously at time t a measurement, therefore giving us a vector of dimension p. We suppose that there are N such emissions yielding a datastream of length N (number of data samples) and dimension p.

In fact, a great deal of modern data are essentially datastreams issued from industrial process monitoring. Even if storing such long streams of data can be possible given modern storage capacities, it is rather impractical. Even if we suppose that an entire datastream could be stored in a database, mining algorithms would need large amounts of time to process such a great volume of data, a behavior which is, most of the times, not desirable in a real-time system where critical decisions have to be taken instantaneously. In addition, since the number of sensors can be large, datastreams are not only long but also high-dimensional.

Therefore, mining such data pose two challenges: develop tools for the online execution of mining algorithms that can efficiently process high-dimensional data.

For this purpose, we propose an online inference algorithm for the mixture of probabilistic PCA model, which is a subspace clustering model and can thus handle high-dimensional data efficiently. The proposed algorithm relies on an EM-based procedure and on a probabilistic and incremental version of PCA. Model selection is also considered in the online setting through parallel computing. We run numerical experiments on simulated and real data and demonstrate the effectiveness of our approach by comparing it with state-of-the-art online EM-based algorithms.

5.1 Online mixture of PPCA

In this section, we restrict ourselves to the mixture of probabilistic PCA (MPPCA) model and consider its online inference. Model selection and visualization of the data into low-dimensional subspaces are also discussed.

5.1.1 Mixture of probabilistic PCAs

The mixture of PPCA model (Tipping and Bishop, 1999b) is a constrained version of the mixture of factor analysis (MFA) model (Ghahramani et al., 1996). Once again, we consider that we dispose of a datastream $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \in \mathbb{R}^p$, where the $\mathbf{x}$ are independent realizations of a random vector $X \in \mathbb{R}^p$. In addition, $\{z_1, ..., z_n\}$ are assumed to be independent realizations of an unobserved (latent) random variable Z with $Z(\Omega) = \{1, ..., K\}$. The MPPCA model assumes that the observed random vector $X \in \mathbb{R}^p$ is, conditionally to Z, linked to a d-dimensional latent random vector $Y \in \mathbb{R}^d$ through a linear transformation of the form:

$$X_{|Z=k} = U_k Y + \boldsymbol{\mu}_k + \epsilon,$$

where U_k is the $p \times d$ orthogonal transformation matrix, $\boldsymbol{\mu}_k \in \mathbb{R}^p$ is the mean vector of the k-th factor analyzer and $\epsilon \in \mathbb{R}^p$ is a noise term. The dimension d of the latent vector is such that $d < p$ and assumed to be known (the choice of d is discussed in Section 3.3). Moreover, ϵ is assumed to be, conditionally to Z, a centered Gaussian noise term with a diagonal covariance matrix $\Psi_k = b_k I_p$:

$$\epsilon_{|Z=k} \sim \mathcal{N}(\mathbf{0}, b_k I_p).$$

Besides, the unobserved latent factor $Y \in \mathbb{R}^d$ is assumed to be, conditionally to Z, distributed according to a Gaussian density function such as:

$$Y_{|Z=k} \sim \mathcal{N}(\mathbf{0}, I_d).$$

This implies that the conditional distribution of X is also Gaussian:

$$X_{|Y,Z=k} \sim \mathcal{N}(U_k Y + \boldsymbol{\mu}_k, b_k I_p), \tag{5.1}$$

and its marginal distribution is therefore a mixture of Gaussians:

$$f(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \phi\left(\mathbf{x}; \boldsymbol{\mu}_k, \Sigma_k\right)$$

where π_k is the mixture proportion for the k-th component, ϕ is the multivariate Gaussian density function and $\Sigma_k = U_k^t U_k + b_k I_p$.

In order to facilitate the description of our online inference procedure, let us slightly re-parametrize the above model. Let us first introduce the orthonormal transformation matrix Q_k which is such that its j-th column $\mathbf{q}_{kj} = \mathbf{u}_{kj} / \parallel \mathbf{u}_{kj} \parallel$ where $\mathbf{u}_{kj}$ is the

corresponding column of U_k. If the transformation matrix Q_k is orthonormal, it is then necessary to report the variance of the latent factor within the distribution of the latent factor. We therefore now assume that:

$$Y_{|Z=k} \sim \mathcal{N}(0, \Delta_k),$$

where $\Delta_k = \text{diag}(\lambda_{k1}, \ldots, \lambda_{kd})$. The marginal distribution of X is then still a mixture of Gaussians but with covariance matrices $\Sigma_k = Q_k^t \Delta_k Q_k + b_k I_p$. By denoting by $W_k = [Q_k, \tilde{Q}_k]$ the $p \times p$ matrix made of Q_k and an orthonormal complementary $\tilde{Q}_k$, the projected covariance matrix $W_k \Sigma_k W_k^t$ has the following form:

$$W_k \Sigma_k W_k^t = \begin{pmatrix} a_{k1} & & 0 & & \\ & \ddots & & \mathbf{0} & \\ 0 & & a_{kd} & & \\ & & & b_k & 0 \\ & \mathbf{0} & & \ddots & \\ & & & 0 & b_k \end{pmatrix} \begin{matrix} \left.\vphantom{\begin{matrix} a \\ a \\ a \end{matrix}}\right\} d \\ \left.\vphantom{\begin{matrix} a \\ a \\ a \end{matrix}}\right\} (p-d) \end{matrix}$$

where $a_{kj} = \lambda_{kj} + b_k$ and $a_{kj} > b_k$, for $k = 1, \ldots, K$ and $j = 1, ..., d$. With these notations, the mixture of PPCA model is fully parametrized by the set of parameters $\theta = \{\pi_k, \boldsymbol{\mu}_k, Q_k, a_{kj}, b_k, d;\ k = 1, ..., K\}$.

It can be shown (Bouveyron et al., 2007a; Tipping and Bishop, 1999a) that, conversely to the MFA model, the MPPCA model is identifiable and its inference can be done using a simple EM algorithm. In particular, the update formula in the M step for the orientation matrices Q_k and the variance parameters a_{kj} and b_k are as follows:

- the d columns of Q_k are estimated by the eigenvectors associated with the d largest eigenvalues of the empirical covariance matrix S_k of the k-th group,
- the empirical covariance matrix of the k-th group is $S_k = \frac{1}{n_k} \sum_{i=1}^{n} t_{ik} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T$
- a_{kj} is estimated by the jth largest eigenvalues of S_k,
- b_k is estimated by:

$$\hat{b}_k = \frac{1}{p-d} \left(\text{tr}(S_k) - \sum_{j=1}^{d} \hat{a}_{kj} \right).$$

In addition, these update formulas illustrate the strong link between MPPCA and the principal component analysis (PCA) method, since they both consider eigenvectors corresponding to the largest eigenvalues of the covariance matrix eigendecomposition.

5.1.2 Online inference of mixture of PPCA

In order to extend MPPCA to the online setting, we develop hereafter an online EM-based algorithm which incorporates a probabilistic version of the incremental PCA (Hall et al.,

1998). We consider here a setting where data samples are arriving in an online manner and each data sample is being discarded after being processed.

Let us assume that we initially have observed a dataset of n_0 data samples$(\mathbf{x}_1, ..., \mathbf{x}_{n_0}) \in \mathbb{R}^p$ and that we have obtained an initial estimate $\hat{\theta}^{(n_0)}$ of these data. In practice, we obtain an initial estimation of the model parameters with a standard MPPCA on this initial dataset. Let us set $n = n_0$ and consider the arrival of a new data sample $\mathbf{x}_{n+1} \in \mathbb{R}^p$. The objective is therefore to update the estimate of θ from the sole knowledge of $\hat{\theta}^{(n)}$ and $\mathbf{x}_{n+1}$. This is a two-step procedure which involves an expectation step (E-step) and a maximization step (M-step).

The E-step

Before updating the estimate of θ, it is necessary to compute the expectation of the complete log-likelihood $E\left[L_c\left(\mathbf{x}, z; \theta\right)|\hat{\theta}^{(n)}\right]$ conditionally to the current estimate $\hat{\theta}^{(n)}$. This quantity will be maximized in the second step to obtain the new estimate $\hat{\theta}^{(n+1)}$ of θ. As with all mixture models, the computation of the conditional expectation of the complete log-likelihood reduces, in the context of the MPPCA model, to the computation of the probabilities $t_k^{(n+1)} = P(Z = k|X = \mathbf{x}_{n+1})$, $k = 1, ..., K$, that the new data sample belongs to the k-th mixture component. These probabilities can be computed as follows:

$$t_k^{(n+1)} = \frac{\pi_k \phi\left(\mathbf{x}_{n+1}; \hat{\theta}_k^{(n)}\right)}{\sum\limits_{\ell=1}^{K} \pi_\ell \phi\left(\mathbf{x}_{n+1}; \hat{\theta}_\ell^{(n)}\right)} = 1 \Big/ \sum_{\ell=1}^{K} \exp\left(\frac{1}{2}(\Gamma_k^{(n)}(\mathbf{x}_{n+1}) - \Gamma_\ell^{(n)}(\mathbf{x}_{n+1}))\right), \quad (5.2)$$

where the classification function Γ_k has the following form:

$$\Gamma_k(\mathbf{x}) = \|\boldsymbol{\mu}_k - P_k(\mathbf{x})\|^2_{\mathcal{A}_k} + \frac{1}{b_k}\|\mathbf{x} - P_k(\mathbf{x})\|^2 + \sum_{j=1}^{d} \log(a_{kj}) + (p-d)\log(b_k) - 2\log(\pi_k).$$

with $\|\mathbf{x}\|^2_{\mathcal{A}_k} = \mathbf{x}^t \mathcal{A}_k \mathbf{x}$,$\mathcal{A}_k = Q_k \Delta_k^{-1} Q_k^t$, $\mathcal{A}_k = Q_k \Lambda_k^{-1} Q_k^t$ and $P_k(\mathbf{x}) = Q_k Q_k^t(\mathbf{x} - \boldsymbol{\mu}_k) + \boldsymbol{\mu}_k$.

The M-step

Once the posterior probabilities $t_k^{(n+1)}$ have been computed, we update the model parameters so that they maximize $E\left[L_c\left(\mathbf{x}, z; \theta\right)|\theta^{(n)}\right]$. In order to derive an online inference strategy which does not keep all past data samples, it is necessary to make use of the following approximation:

$$E\left[L_c\left(\mathbf{x}, z; \theta\right)|\theta^{(n+1)}\right] \simeq E\left[L_c\left(\mathbf{x}, z; \theta\right)|\theta^{(n)}\right] + \sum_{k=1}^{K} t_k^{(n+1)} \log\left(\pi_k \phi\left(\mathbf{x}_{n+1}; \theta_k\right)\right).$$

Then, it is straightforward to show that the update formulas for the mixture proportions π_k and the component means $\boldsymbol{\mu}_k$, for every component $k = 1, ..., K$, are:

$$\hat{\pi}_k^{(n+1)} = \hat{\pi}_k^{(n)} + \frac{1}{n+1}\left(t_k^{(n+1)} - \hat{\pi}_k^{(n)}\right), \tag{5.3}$$

$$\hat{\boldsymbol{\mu}}_k^{(n+1)} = \frac{1}{n_k^{(n+1)}}\left(n_k^{(n)}\hat{\boldsymbol{\mu}}_k^{(n)} - t_k^{(n+1)}\mathbf{x}_{n+1}\right), \tag{5.4}$$

where $n_k^{(n+1)} = n_k^{(n)} + t_k^{(n+1)}$ and $n = \sum_{k=1}^{K} n_k^{(n)}$.

We then want to estimate the parameters Q_k, a_{kj} and b_k, for $k = 1, ..., K$ and $j = 1, ..., d$. We have already seen that the maximization of $E\left[L_c\left(\mathbf{x}, z; \theta\right)|\hat{\theta}^{(n)}\right]$ with respect to these parameters is equivalent to the eigendecomposition of the empirical covariance matrix S_k for each component $k = 1, ..., K$. The problem that we seek to solve can be therefore stated as follows: having already calculated eigenvectors $Q_k^{(n)}$ and eigenvalues $\Lambda_k^{(n)}$ from the n first data samples, we want to update those parameters on the arrival of a $(n+1)$-th data sample. In particular, on the arrival of the new data sample $\mathbf{x}_{n+1}$, the new eigenproblem that we need to solve is:

$$\Sigma_k^{(n+1)} Q_k^{(n+1)} = Q_k^{(n+1)} \Lambda_k^{(n+1)}, \tag{5.5}$$

where $\Lambda_k^{(n+1)} = \text{diag}\{\lambda_{k1}, ..., \lambda_{kp}\}$ and this for $k = 1, \ldots, K$.

To begin with, let us define:

$$g_k^{(n+1)} = \left(Q_k^{(n)}\right)^T \left(t_k^{(n+1)}\mathbf{x}_{n+1} - \boldsymbol{\mu}_k^{(n)}\right),$$

$$h_k^{(n+1)} = \left(t_k^{(n+1)}\mathbf{x}_{n+1} - \boldsymbol{\mu}_k^{(n)}\right) - Q_k^{(n)} g_k,$$

where $g_k^{(n+1)}$ is the projection of the data sample on the subspace defined by the eigenvectors and $h_k^{(n+1)}$ is the residue of the retro-projection on the original space. With these notations, the new eigenvectors $Q_k^{(n+1)}$ correspond to a rotation of the old ones plus the unit residue vector $\tilde{h}_k^{(n+1)}$:

$$\tilde{h}_k^{(n+1)} = \begin{cases} \frac{h_k^{(n+1)}}{\left\|h_k^{(n+1)}\right\|_2}, & \text{if } \left\|h_k^{(n+1)}\right\|_2 \neq 0, \\ 0, & \text{otherwise} \end{cases}$$

and thus we have:

$$\hat{Q}_k^{(n+1)} = \left[Q_k^{(n)}, \tilde{h}_k^{(n+1)}\right] R_k^{(n+1)} \tag{5.6}$$

where $R_k^{(n+1)}$ is a rotation matrix of size $(d+1) \times (d+1)$. Note that $Q_k^{(n)}$ is a $p \times d$ matrix, since we have discarded the $p-d$ less significant eigenvalues. The new covariance matrix $\Sigma_k^{(n+1)}$ for the class k is given by:

$$\Sigma_k^{(n+1)} = \frac{n_k^{(n)}}{n_k^{(n+1)}}\Sigma_k^{(n)} + \frac{n_k^{(n)}}{\left(n_k^{(n+1)}\right)^2}\bar{\mathbf{x}}\bar{\mathbf{x}}^T \tag{5.7}$$

where we have set $\bar{\mathbf{x}} = t_k^{(n+1)}\mathbf{x}_{n+1} - \boldsymbol{\mu}_k^{(n+1)}$. Then, by substituting Equations 5.6 and 5.7 into Equation 5.5, we get:

$$\left[Q_k^{(n)}, \tilde{h}_k^{(n+1)}\right]^T \left(\frac{n_k^{(n)}}{n_k^{(n+1)}} \Sigma_k^{(n)} + \frac{n_k^{(n)}}{\left(n_k^{(n+1)}\right)^2} \bar{\mathbf{x}}\bar{\mathbf{x}}^T \right) \left[Q_k^{(n)}, \tilde{h}_k^{(n+1)}\right] R_k^{(n+1)} = R_k^{(n+1)} \Lambda_k^{(n+1)}$$

The above problem can be written as:

$$\left(\frac{n_k^{(n)}}{n_k^{(n+1)}} \begin{bmatrix} \Lambda_k^{(n)} & 0 \\ 0 & 0 \end{bmatrix} + \frac{n_k^{(n)}}{\left(n_k^{(n+1)}\right)^2} \begin{bmatrix} g_k g_k^T & \gamma_k g_k \\ \gamma_k g_k^T & \gamma_k^2 \end{bmatrix} \right) R_k^{(n+1)} = R_k^{(n+1)} \Lambda_k^{(n+1)} \tag{5.8}$$

where we have set $\gamma_k^{(n+1)} = \left(\tilde{h}_k^{(n+1)}\right)^T \bar{\mathbf{x}}$. The solution to this new eigenproblem yields the rotation matrix $R_k^{(n+1)}$ and the new eigenvalues $\Lambda_k^{(n+1)}$ directly. Then, the new eigenvectors can be obtained using Equation 5.6. Note that both $R_k^{(n+1)}$ and $\Lambda_k^{(n+1)}$ are square matrices of dimension $(d+1)$, that is, we only need to solve an eigenproblem of dimension $(d+1)$ and not p. The update formulas for the variance parameters a_{kj} and b_k are then:

$$\begin{aligned} \hat{a}_{kj}^{(n+1)} &= \Lambda_{kj}^{(n+1)}, \\ \hat{b}_k^{(n+1)} &= \frac{1}{p-d} \left(\text{tr}(S_k) - \sum_{j=1}^{d} \hat{a}_{kj}^{(n+1)} \right), \end{aligned}$$

where $\text{tr}(S_k)$ is the trace of the empirical covariance matrix for group k.

Algorithm and classification step

The online MPPCA algorithm that we proposed above is summarized in Algorithm 4. Even though, in the first place, the online MPPCA algorithm aims to infer the MPPCA model in the online setting, in this work, we are also interested in obtaining a partition of the data after having processed the last data sample. To do so, it is necessary to add a classification step at the end of the online MPPCA algorithm to provide the expected clustering. In the model-based clustering framework, data samples are usually assigned to a group using the maximum a posteriori (MAP) rule, which assigns a data sample $\mathbf{x} \in \mathbb{R}^p$ to the group for which it has the highest posterior probability $P(Z = k|X = \mathbf{x})$ at the end of the algorithm. Therefore, this final classification step simply consists in assigning the data sample $\mathbf{x}_i$ to the group with the highest $t_k^{(i)}$, for $k = 1, ..., K$ and $i = 1, ..., N$.

5.1.3 Model selection in the online framework

The online MPPCA algorithm, as presented above, performs an almost automatic inference of the MPPCA model, except for the hyper-parameters K and d. Indeed, those

Algorithm 4 The online MPPCA algorithm

1. Initialization: run a classical MPPCA on the n_0 first data samples to provide an initial set $\hat{\theta}^{(n_0)}$ of parameter estimates.
2. For each new data sample $\mathbf{x}_i$:
   - E-step: compute probabilities $t_k^{(i)}$, for $k = 1, ..., K$, using Equation (5.2),
   - M-step: update parameter estimates using Equations (5.3-5.4). Update $\hat{Q}_k$, $\hat{a}_{kj}$ and $\hat{b}_k$ for $k = 1, \ldots, K$ and $j = 1, \ldots, d$ by solving the incremental eigenproblem (5.8).
3. After the last data sample $\mathbf{x}_N$, return:
   - set $\hat{\theta}^{(N)}$ of model parameter estimates,
   - data partition which can be deduced from the probabilities $t_k^{(i)}$, $i = 1, ..., N$ and $k = 1, ..., K$ using the MAP rule.

parameters cannot be determined by maximizing the conditional expectation of the complete likelihood since they both control the model complexity. A popular and well-established way to determine the appropriate value for both K and d for the data at hand is to consider it as a model selection problem. Thus, the use of either the AIC (Akaike, 1981), BIC (Schwarz, 1978) or ICL (Biernacki et al., 2000) criteria allows to find the appropriate values for K and d. However, since in the online setting that we consider in this work, past data samples are not kept in memory, it is necessary to solve the model selection problem in an online manner as well. This is made possible nowadays by parallel computing. In our context, this consists in running several online MPPCA algorithms with different values for the hyper-parameters in parallel and selecting at the end the solution associated with the highest value of the model selection criterion.

5.1.4 Low-dimensional visualizations of the data

A final advantage of our online MPPCA algorithms is that it allows to provide low-dimensional visualizations of the whole data set, even though the high-dimensional data samples are not kept. The low-dimensional visualizations are the projections of the data into the K estimated subspaces of the groups. If d is small compared to p, it is reasonable to keep in memory these low-dimensional representations of the data since the necessary memory size after n data samples is $m_d = K \times n \times d$ instead of $m_p = n \times p$. However, this requires to be able to update the low-dimensional projections into the group subspaces at the arrival of each new data sample. At iteration $n + 1$, this can be done after the M-step as follows:

$$\boldsymbol{y}_i^{(n+1)} = \boldsymbol{y}_i^{(n)} R_k^{(n+1)},$$

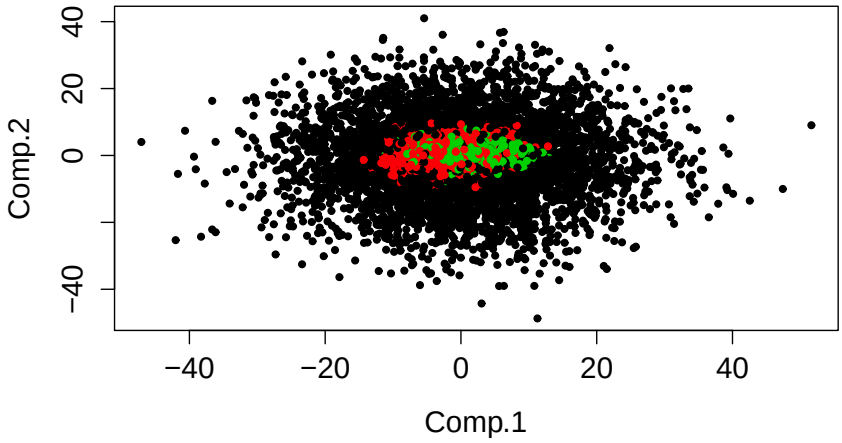


Figure 5.1: Simulated data from $K = 3$ classes (represented by the three colors) of original dimension $p = 30$, projected on the PCA axis. We can see that it is a challenging dataset for a clustering algorithm.

where $\boldsymbol{y} \in \mathbb{R}^d$ and $R_k^{(n+1)}$ contains the eigenvectors of the eigenproblem (5.8), for $k = 1, ..., K$ and $i = 1, ..., n$.

5.2 Experiments

In this section, we present and discuss the results of the experiments that we performed on simulated and real data, with the aim of validating the performance of online MPPCA and of comparing it with other online algorithms.

5.2.1 An introductory example

We begin by an introductory experiment on simulated data. We have generated a dataset of $n = 12\,000$ data samples $(\mathbf{x}_1, \dots, \mathbf{x}_n) \in \mathbb{R}^p$ based on the assumption that data live in low-dimensional subspaces, with $p = 30$ and $K = 3$. Hereafter, we refer to this dataset as X_{30}. The mixture proportions are $\pi_1 = 0.4$ and $\pi_2 = \pi_3 = 0.3$. For simplicity, we have considered that for each class, the variance is common across all dimensions, that is $a_{kj} = a_k$, for $k = 1, \dots, K$ and $j = 1, \dots, d$. We have set $a_1 = 150$, $a_2 = 75$, $a_3 = 50$, $b_1 = b_2 = b_3 = 5$ and $\boldsymbol{\mu}_1 = \mathbf{0}$, $\boldsymbol{\mu}_2 = \{0, \dots, 5, \dots, 0\}$ and $\boldsymbol{\mu}_3 = \{0, \dots, -5, \dots, 0\}$, with $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\mu}_3 \in \mathbb{R}^p$. We have set the intrinsic dimension (dimension of the subspaces) at $d = 2$. Figure 5.1 shows the projection of the simulated dataset of $p = 30$ on the PCA axis. We can see that it is a challenging dataset for a clustering algorithm.

Note that we have initialized online MPPCA with $n_0 = 100$ data samples. The algorithm was given the true values for K and d. In practice, one has to run it with

different values of K and d and keep the values giving the best model (according to a criterion, *i.e.* BIC (Schwarz, 1978)).

Figure 5.2 show the results obtained by online MPPCA for the dataset X_{30}. Figure 5.2a shows the evolution of the estimation of MPPCA parameters a_k for $k = 1, \ldots, K$ versus the number of the data samples. The horizontal line correspond to the true values of the parameters. We can see that as the number of data samples grows, online MPPCA converges towards the true value of the parameters.

Figure 5.2b shows the evolution of clustering accuracy versus the number of data samples for online MPPCA. We can see that clustering accuracy given by online MPPCA constantly increases as new data samples are arriving, converging to the accuracy given by a standard MPPCA model which passes over data multiple times.

Finally, let us note that an interesting feature of the proposed algorithm is its speed. It tooks online MPPCA only 65 seconds to process the dataset of $n = 12\ 000$ data samples.

5.2.2 Comparison with online EM and online CEM

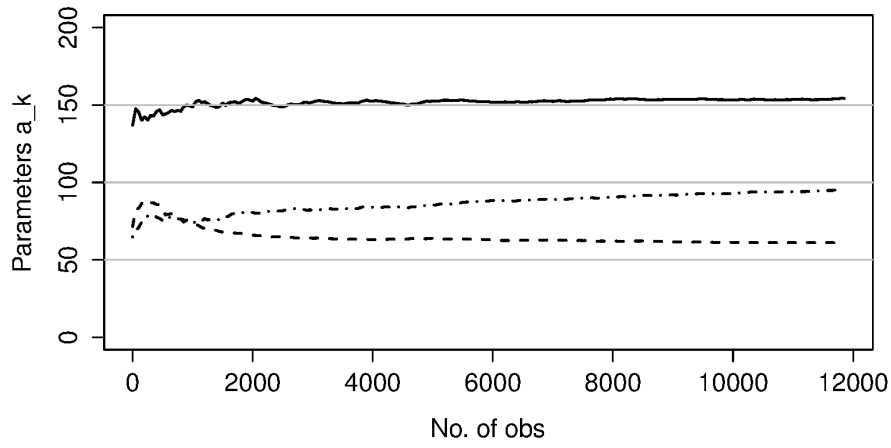
In this second experiment, we compare online MPPCA with two other online algorithms, online EM (Titterington, 1984) and online CEM (Samé et al., 2007). Note that these latter have not been designed to handle high-dimensional data. In this experiment, we have used X_{30}, the high-dimensional simulated dataset presented above, as well as a second simulated dataset of lower dimension ($p = 10$) , generated with the same parameters as the former. We will refer to this new dataset as X_{10}. Our goal was to study the behavior of the three algorithms in low dimension and then illustrate the capability of online MPPCA to cluster efficiently even in high dimension.

We have evaluated the three algorithms on the quality of their estimation of the class means and on the accuracy of the clustering produced. The quality of the estimation of the means was taken to be the square of the distance of the estimated means to the true ones, averaged over all $K = 3$ classes, a measure known as the *Mean Square Error* (MSE) in statistics

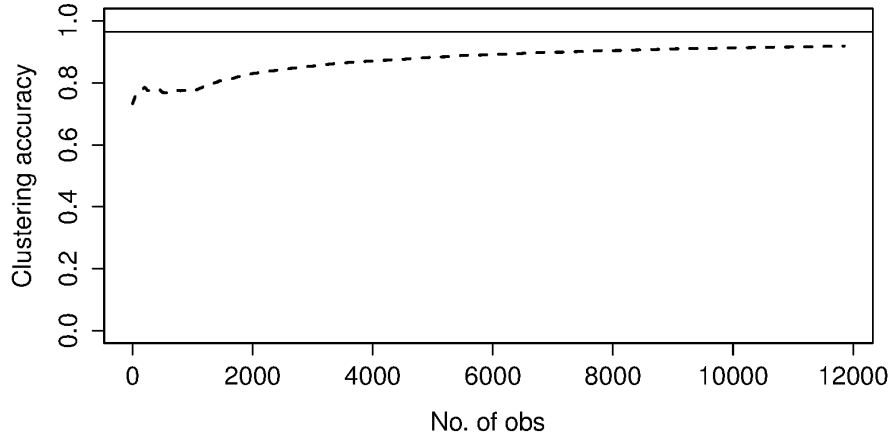
$$\mathrm{MSE}_\mu = \frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{p}\sum_{j=1}^{p}\left(\hat{\boldsymbol{\mu}}_{kj} - \boldsymbol{\mu}_{kj}\right)^2\right)$$

Online MPPCA, online EM and online CEM were initialized 30 times by a standard MPPCA, an EM and a CEM, respectively, of which the initialization giving the highest BIC value was kept. Figure 5.3 and Figure 5.4 show the comparative performance (MSE_μ and clustering accuracy) of online MPPCA (black), online EM (red) and online CEM (blue) for the datasets X_{10} and X_{30}, respectively.

For the dataset X_{10} it is clear, both from the clustering accuracy and the MSE_μ that online MPPCA converges faster than the other two algorithms. Online CEM converges faster than online EM, a result which is compatible with conclusions made in (Samé et al., 2007).

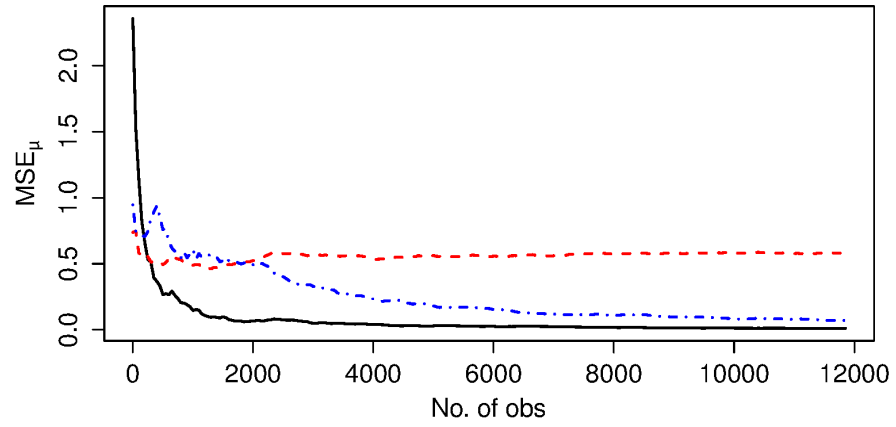


(a)

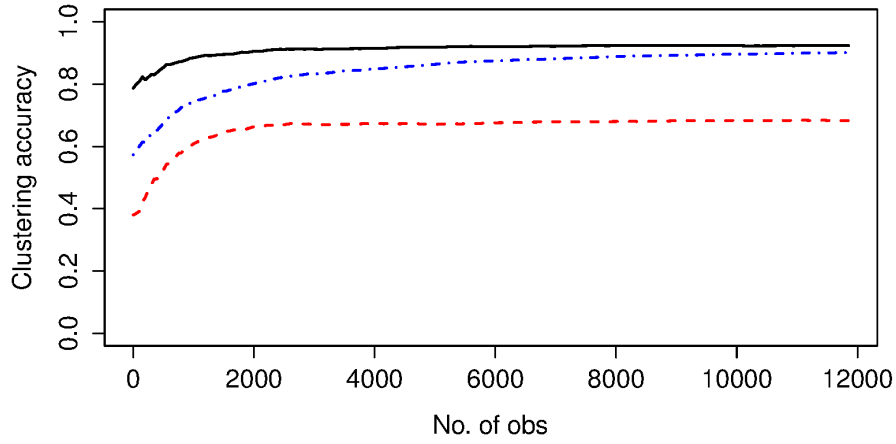


(b)

Figure 5.2: (a) Evolution of the estimated parameters a_k of the 3 groups ($k = 1, 2, 3$) for the dataset X_{30} versus the number of data samples for online MPPCA. Horizontal lines correspond to the true values of the parameters. (b) Clustering accuracy evolution for the dataset X_{30} versus the number of data samples for online MPPCA. The solid horizontal line corresponds to the clustering accuracy given by a standard MPPCA, which passes multiple times over data.

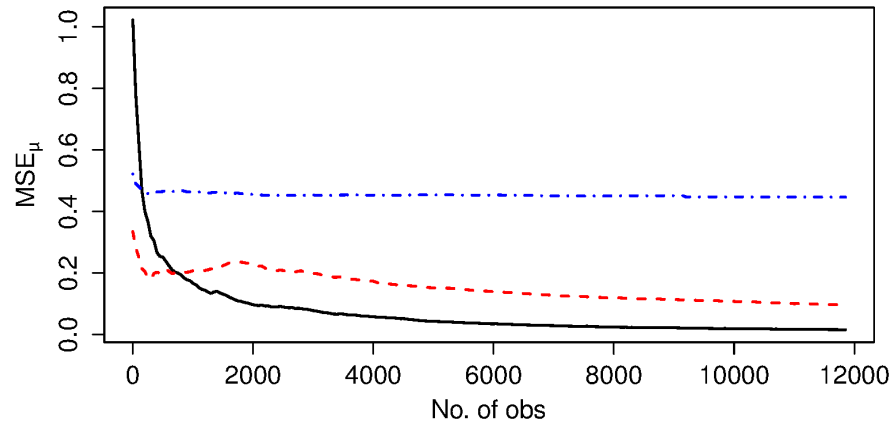


(a)

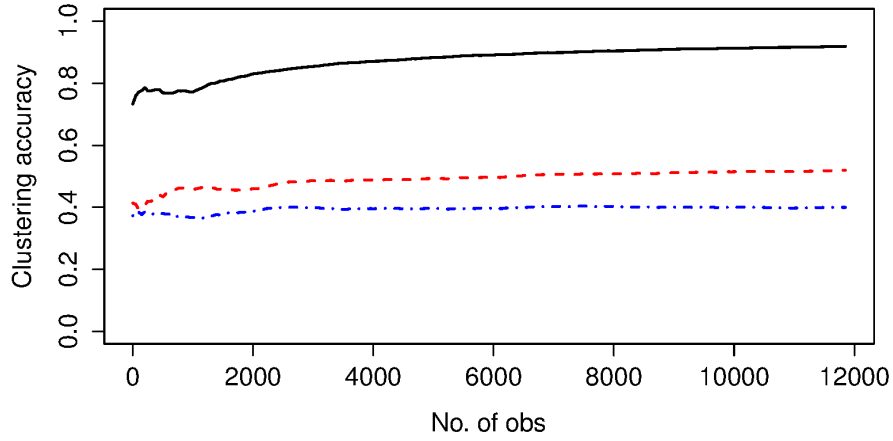


(b)

Figure 5.3: (a) Evolution of MSE_μ for the dataset X_{10} versus the number of data samples for online MPPCA (black solid), online EM (red dashed) and online CEM (blue dotted). (b) Clustering accuracy evolution for the dataset X_{10} versus the number of data samples for the three algorithms.



(a)



(b)

Figure 5.4: (a) Evolution of MSE$_\mu$ for the dataset X_{30} versus the number of data samples for online MPPCA (black solid), online EM (red dashed) and online CEM (blue dotted) (b) Clustering accuracy evolution for the dataset X_{30} versus the number of data samples for the three algorithms.

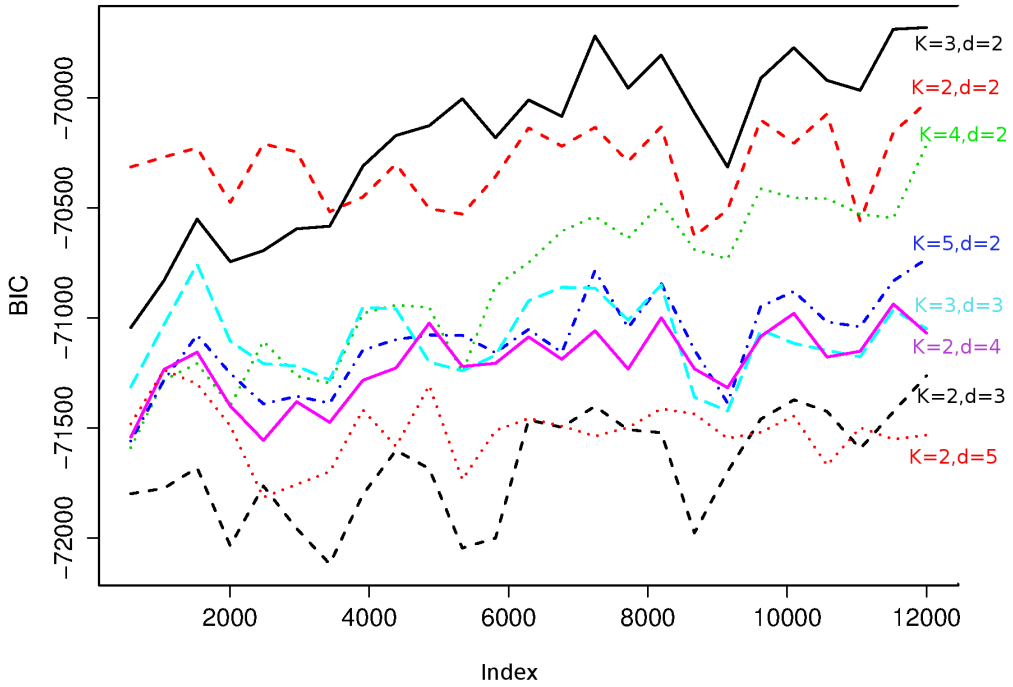


Figure 5.5: BIC values versus the data indexes for the 8 hyperparameter value combinations that gave the highest BIC in the last packet.

For the dataset X_{30}, we can see that online MPPCA clearly outperforms the other two algorithms, even in high dimension $p = 30$. As expected, high dimensionality affects the clustering performance of both online EM and online CEM. Note here that we have not compared the three algorithms in $p > 30$ because online CEM in particular cannot handle such a dimensionality due to numerical problems.

Model selection with parallel computation

As we have already seen, our model has two hyperparameters, the intrinsic dimension d and the number of groups of the mixture K. Selecting the optimal values for those hyperparameters is selecting the best model, since different values of K and d will give different models. In practice, this means that an instance of online MPPCA must be launched for each possible combination of the range of interest for K and d.

However, in an online setting, past data samples are not being stored and thus, traditional model selection methods, for instance, evaluation of BIC over the whole dataset, cannot be applied. Nonetheless, in real applications it can be acceptable to keep a small number of the most recent data samples.

Moreover, in an online setting, model selection has to be done in a timely manner.

Parameters	BIC
K=3, d=2	−69679.14
K=2, d=2	−70020.14
K=4, d=2	−70213.18
K=5, d=2	−70731.57
K=3, d=3	−71048.07
K=2, d=4	−71068.52
K=2, d=3	−71263.18
K=2, d=5	−71532.53

Table 5.1: Hyperparameter values and respective BIC for the last packet (data samples with indexes $11524 - 12000$) in decreasing order. We can see that the highest BIC value corresponds to the true value of the parameters $K = 3$ and $d = 2$.

Launching as many algorithms as there are possible combinations of K and d for each novel data sample is clearly not an option, since we would need a great amount of time. However, parallel computation can greatly reduce the necessary time for such a procedure.

Based on the above, we launch online MPPCA in parallel for all values K and d and for each one of them, we evaluate BIC in packets of data. We use the X_{30} dataset which is of dimension $p = 30$, composed of $n = 12\ 000$ data samples issued from $K = 3$ classes and where the intrinsic dimension is $d = 2$. We evaluate BIC

$$\text{BIC} = -2L_w + v \log n_w.$$

for packets of $n_w = 476$ data samples for all possible combinations of $K = 2, \ldots, 6$ and $d = 2, \ldots, 10$. Here, L_w is the log-likelihood of the maximum likelihood estimates for the data packet and v is the number of the model parameters.

Figure 5.5 shows the 8 combinations that gave the higher BIC value. We can see that after an initial period that the algorithm needs to converge, the BIC value of the true pair of parameters $(K = 3, d = 2)$ is higher than the rest. Table 5.1 shows the corresponding BIC values for the last packet of data.

Finally, let us note that the overall procedure, that is 45 parallel runs of online MPPCA in 20 Intel(R) Xeon(R) 3.07GHz processors, needed a total runtime of only 10 minutes. This experiment proves that by keeping a small amount of past data, one can select the proper model in a timely manner using the benefits of parallel computation.

5.2.3 Application to aircraft engine health monitoring

In this section, we test the proposed method to real data issued from the aircraft engine Health Monitoring domain. The data were obtained by Snecma, the french aircraft engine constructor.

Typically, there exists different phases during a flight, called flight modes: taking off, cruising, landing etc. Each test is actually a sequence of alternating stationary and

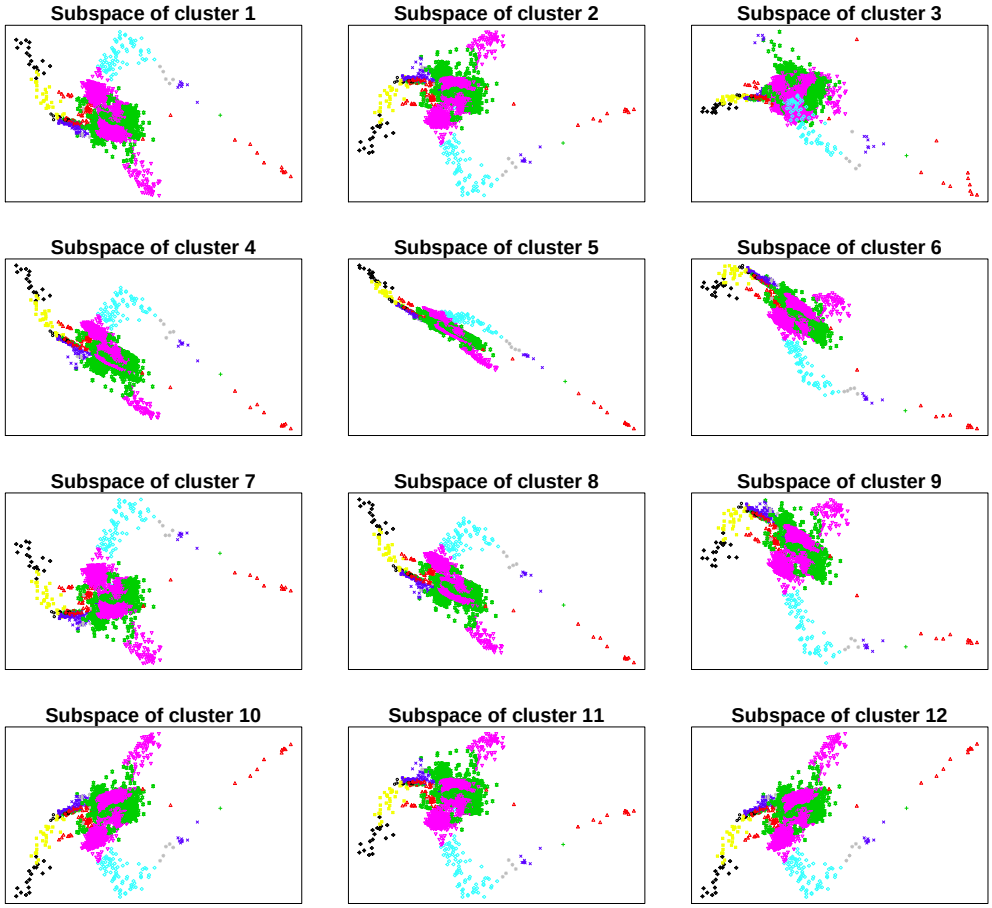


Figure 5.6: Projection of aircraft engine data on each of the class-specific subspaces of dimension $d = 2$. Colors correspond to different classes according to the clustering produced by online MPPCA. The projections give an interesting insight into the classes.

non-stationary phases at different levels. The stationary phases correspond in general to such flight modes, while the non-stationary ones reflect the transition between two such phases. Nevertheless, a flight mode can include multiple stationary phases, that is, a stationarity control on the data is not enough to detect the flight modes.

Aircraft engineers can identify these modes by looking at the data but this can be extremely time-consuming. Moreover, due to the high dimensionality of data, there can be relations that humans cannot perceive. Note that by knowing, at any given time, in which flight mode the engine currently is, tasks like anomaly detection can be performed much more reliably, since the 'local' context of the data is also taken into account.

Here, we initially consider a streaming dataset of $n = 4683$ data samples and $p = 173$ variables issued from an engine cell test. Expert advice provided us with a configuration

of 4 endogenous variables and 6 exogenous. Therefore, we consider only those $p = 10$ variables out of 173. We then treat them in order to remove the influence of the exogenous variables to the endogenous ones. In the end, we have a dataset of $p = 4$ endogenous variables, clean of exogenous influence.

We use this dataset to illustrate that online MPPCA can facilitate the detection of homogeneous groups of aircraft engine data. Such a group can coincide with a flight mode, subsume multiple flight modes or correspond to a part of a flight mode. Expert analysis is then needed to analyze these groups and relate them to the engine or to actual events (if any) that occurred during a test sequence.

We launched online MPPCA with $K = 12$ and $d = 2$. In fact, we tested different combinations of the values of these two parameters and we kept the one giving the greatest BIC value. The initial dataset size was set at $n_0 = 300$.

Figure 5.6 shows the projection of the data onto each one of the class-specific subspaces given by online MPPCA, after having processed all the data samples. Colors correspond to different classes according to the clustering produced by online MPPCA. We can see that the projections give an interesting insight into the clustering induced by online MPPCA. Clusterings in each subspace can provide aircraft engineers with a much richer information on a possible inherent substructure of the data.

5.3 Conclusion

We have proposed an online inference algorithm for the MPPCA model which relies on an EM-based procedure and a probabilistic and incremental version of PCA. The proposed strategy allows to incrementally update the estimates of the MPPCA parameters at the arrival of a new data sample. It allows also to provide low-dimensional visualizations of the data based on sufficient information. Model selection is also considered in the online setting through parallel computing. Numerical experiments on simulated and real data have shown that the online MPPCA algorithm performs better in high-dimensional spaces compared to existing online EM-based algorithms.

Among the possible extensions for this work, it could be interesting to consider the re-computation of the posterior probabilities for all data samples (including past data samples) in the E-step based on the kept projected data. In addition, a deeper analysis of initialization strategies should be considered for the algorithm, since not all the classes are represented in the initial part of a datastream used for initialization of the algorithm. A possible solution to this problem would be to adopt a dynamic model selection procedure, where classes could be added or removed. Finally, another extension of our work could be to make an online MPPCA which will be robust to anomalies, possibly using works on robust PCA, which could be adapted appropriately in order to develop an online robust MPPCA.

Chapter 6

Application to aircraft engine Health Monitoring

In this chapter, we develop an application of the proposed methods for the task of anomaly detection and visualization of aircraft engine anomalies. The data that we apply our method to are flight data supplied by Snecma, the French aircraft engine constructor. The objective is to develop a comprehensive system that would be able to detect anomalies online, providing the experts with tools to monitor them. As a small introduction to the subject, we first give a small review of related articles.

6.1 Related work

Health monitoring is a vast domain and giving a detailed survey of articles on the matter is out of the scope of this Thesis. Even if we constrain ourselves to Statistical Learning approaches, the volume of the articles that need to be reviewed remains considerable. Moreover, since the methodology we will follow to treat data and apply methods that we proposed in the previous chapters is, as expected, based on a well-established methodology that experts in Snecma have been using for years, such a large survey would prove impractical. Instead, we will only briefly present works on Health Monitoring carried out by Snecma but, also, works that have appeared during the last 5 years in the context of successive research projects between Snecma and SAMM[1], the author's affiliated research laboratory.

(Lacaille et al., 2010) propose *Continuous empirical score* (CES), an algorithm for Health Monitoring for a test cell environment based on three components: a clustering algorithm based on EM, a scoring component and a decision procedure.

The clustering component starts by replacing original exogenous variables by *indicators*. Indicators are generated from original data using various transformations. For instance, a possible transformation consists in calculating trends in the signal, another

[1] http://samm.univ-paris1.fr/

one in looking for changepoints or still, generic signal compressions based on signatures obtained by a Principal Component Analysis on the data. The transformed data are then used in detecting context classes using EM. The number of classes to look for is determined using the BIC criterion.

Data clustered in context classes are then used to calibrate the scoring component. To begin with, data in each context class are transformed *locally* in a two-phase procedure: first, data are normalized so that small, insignificant variations can be eliminated and second, a Gaussian score is being calculated. In particular, data normalization is being done using a L_1-penalized regression (LASSO criterion (Tibshirani, 1996)) that encourages sparsity and provides a natural way for choosing meaningful indicators. The sparsity hyperparameter is being chosen via cross-validation. Then, the residuals of the regression are used to calculate a score based on a Mahalanobis distance. This score should follow a χ^2 distribution under certain assumptions.

Finally, the decision component operates by comparing the scores of a neighborhood of points to a threshold α. A high-level decision voting scheme is adopted to issue or not a detection alert. Insertion of artificial anomalies into the data can lead to a reliable estimate for the threshold.

In (Lacaille and Gerez, 2011; Côme et al., 2010a; Lacaille et al., 2011), a similar methodology is being applied to detect changepoints in Aircraft communication, adressing and reporting System (ACARS) data, which are basically messages transmitted from the aircaft to the ground containing on-flight measurements of various quantities relative to the engine and the aircraft. Data are classified in context classes and then a piecewise linear L_1-penalized regression model is used to detect changepoints in an *online* fashion. By changepoints we usually mean abrupt changes of the signal. There is however a need for detecting slower variations of a signal as well. A recursive linear regression (RLS) model is used to detect such variations. Finally, engine state trajectories are being projected on a 2D Kohonen map (see also (Lacaille and Côme, 2011)), allowing for expert validation, analysis and identification of possible malfunctions based on known defects. This is made possible by monitoring the position of successive data samples of the same engine on the map. These successive positions define a *trajectory* on the map. Engines working properly will present similar trajectories, whereas engines with possible defects will tend to deviate from "normal" trajectories at a given time instant.

Further insight in normal states and/or further regrouping of similar states can be done by clustering the trajectories. Hierarchical clustering algorithms with an edit distance metric can be used to this extent (Côme et al., 2011).

An interesting point is made by (Côme et al., 2010b) on visualisation using a Kohonen map. In particular, the authors point out that Kohonen maps with classical architectures can be unintuitive for Health Monitoring purposes, since it is possible that the normal state of an engine is found near the edge of the map. Generally speaking, there is no Kohonen prototype (class) that takes a central position in the map. They propose a novel *star* architecture for Kohonen maps, composed of a star center and a predefined number of rays of predetermined size departing from this center. The idea here is that the center of the star will capture the normal state of an engine with some rays regrouping normal behaviours which have drifted away from the center state and other

rays capturing possible engine defects.

Finally, (Ricordeau and Lacaille, 2010) is an interesting article with a preliminary application of Random Forests (Breiman, 2001) to Health Monitoring. The authors examine ways to use the properties of Random Forests to perform Health Monitoring providing the experts with interactive and intuitive analysis tools.

6.2 Overview of the proposed methodology

Flight data consist of a series of measures acquired by sensors positioned on the engine or the body of the aircraft, as well as other information such as the name of the aircraft, the serial number of its engine(s) etc. Data may be issued from a single or multiple engines. We distinguish between measures related to the environment, such as ambient temperature and aircraft speed and those referring to the engine such as rotation speed and pressure measured inside the engine. We will refer to the former as exogenous or environmental and to the latter as endogenous measures or variables. Indeed, we expect engine-related anomalies to manifest themselves in the endogenous measures.

Typically, environmental measures influence endogenous ones. Therefore, data pre-processing is essential in order to erase this effect. In particular, we can first classify exogenous data and then remove, for each class, the effect of the environmental measures on the endogenous ones. For instance, we can do this with a linear regression model. In this way, we can work with the residuals, that is, with endogenous data corrected from the influence of the environment.

The entire procedure of pre-processing, detection and visualization of anomalies has two main phases. We give a graphical overview in Figure 6.1. The procedure can be described as follows:

1. The first phase is the training or learning phase. It is being assumed that we have access at some anomaly-free data, based on which the system can "learn" the normal, incident-free functioning of an engine. The first step is to classify data into classes of environmental conditions, by using only environmental variables. Afterwards, the influence of the environment on endogenous measures is being removed using a linear model, thus obtaining the corrected data (residuals). Next, a SOM is being learned based on the corrected data. The final step of the learning phase is to calibrate an anomaly detection component based on the corrected data.

2. The learning phase is followed by the test phase, where novel, unknown data are taken into account. Indeed, it is being assumed that novel data can arrive over time; data which has not been "seen" during the learning phase. A novel data sample is first being classified in one of the environment classes using the classification model learned in the training phase. Afterwards, it is being corrected of the environment influence using the linear model estimated earlier. The test sample is then projected to the Kohonen map constructed in the training phase and finally, the calibrated anomaly detection component determines if the sample is normal or not.

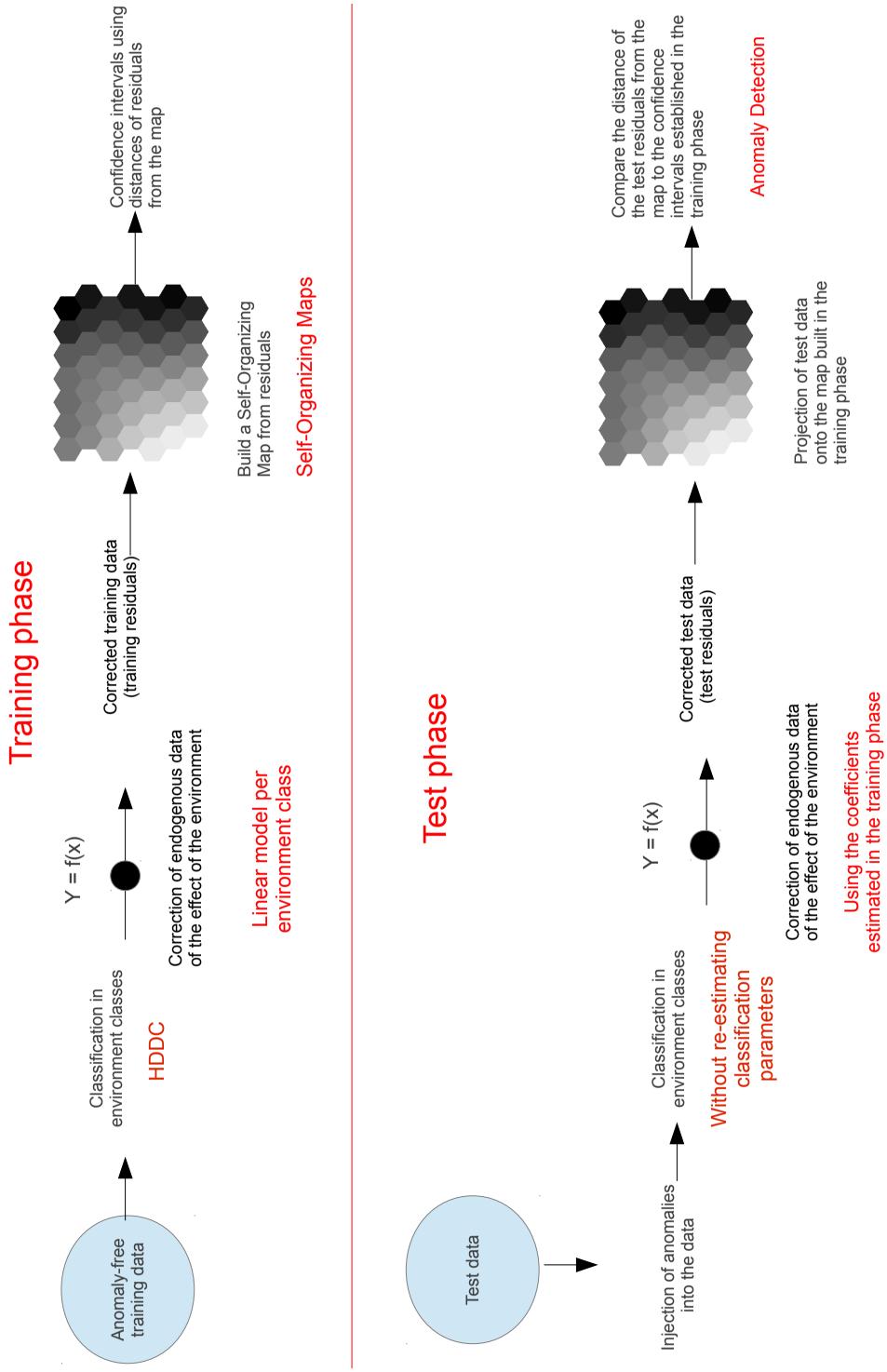


Figure 6.1: An overview of the proposed methodology.

We have used HDDC (chapter 4) for the classification step and the SOM algorithm (Kohonen, 2001) for data visualization, utilizing its properties for the anomaly detection task.

The rest of the chapter is organized as follows: in Section 6.3, the different components of the system proposed are being described in detail, giving emphasis to algorithms and techniques used. Section 6.4 presents the data the we used in this application in detail and describes preliminary data analyses and exploration that we conducted aiming at understanding more deeply the nature of the data. Finally, section 6.5 presents the experiments that we carried out and their results.

6.3 Algorithms and methodologies

In this section we introduce algorithms and techniques that were used to build this application. We start with basic notions of classification. Then, a short introduction to Self-Organizing Maps (SOM) is given. We use SOMs to visualize data and calibrate the anomaly detection component. We also detail the method we used for adding anomalies to the data, since data with real anomalies. Finally, we propose a statistical method to detect anomalies.

6.3.1 Hierarchical clustering

We present below some basic notions of classification and we introduce the Hierarchical Clustering Algorithm (HAC) that we will use later on.

Classication

Assuming there are n data samples $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ distributed in K classes $C_1, C_2, \ldots, C_K$, obtained by an arbitrary classification technique. These K classes form a partition:

$$\Omega \quad = \quad C_1 \cap C_2 \cap \ldots \cap C_K$$

with

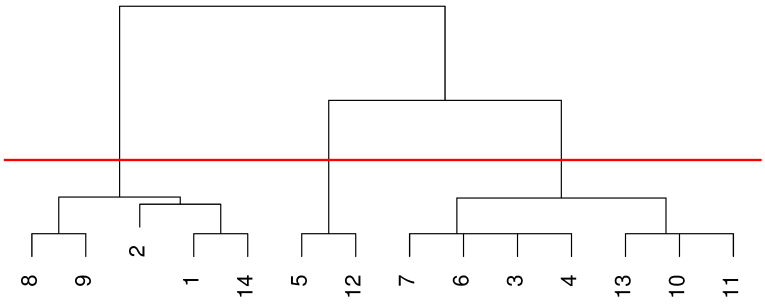
$$\forall i \neq j, C_i \cup C_j \neq \emptyset$$

We denote n_k the cardinality of class C_k and thus, we have $n = \sum_{k=1}^{K} n_k$. We denote $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$ the center of gravity of all data samples and $\bar{\mathbf{x}}_k = \frac{1}{n_k}\sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i$, $k = 1, \ldots, K$, the centers of gravities of all K classes.

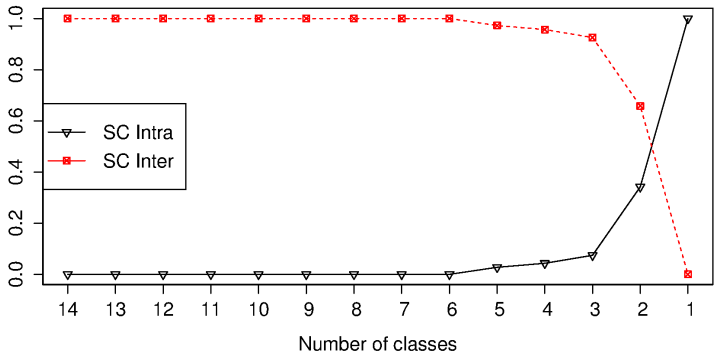
Once we have a classification of the data, we can define the following terms:

The total sum of squares

$$S_T(C_1, C_2, \ldots, C_K) \quad = \quad \sum_{k=1}^{K}\sum_{j=1}^{n_k} \|\boldsymbol{x}_{kj} - \bar{\mathbf{x}}\|_2^2 \tag{6.1}$$



(a) Dendrogram



(b) Sum of squares versus the number of classes

Figure 6.2: An example of a dendrogram obtained by a hierarchical classification. Indexes on the x-axis are those of the data samples. In the beginning, there are 14 classes containing one single data sample. Successive regroupings are then being performed (red lines joining two classes), until we are left up with a single class of n data samples (root of the dendrogram).The horizontal line marks the height at which we have "cut" the tree, in order to have the desired number of classes, which is here $K = 3$.

Note that it does not depend on the classes and that it characterizes the dispersion of the data. One can also define the within-class sum of squares:

$$S_W(C_1, C_2, \ldots, C_K) = \sum_{k=1}^{K} \sum_{j=1}^{n_k} \|\boldsymbol{x}_{kj} - \bar{\mathbf{x}}_k\|_2^2 \tag{6.2}$$

There is also the between-class sum-of-squares

$$S_B(C_1, C_2, \ldots, C_K) = \sum_{k=1}^{K} n_k \|\bar{\mathbf{x}}_k - \bar{\mathbf{x}}\|_2^2 \tag{6.3}$$

One can show that the following relation holds between the three terms

$$S_T = S_W + S_B \tag{6.4}$$

Hierarchical Ascending Classification

In a hierarchical ascending classification (HAC), data are being gradually aggregated until the desired number of classes is obtained. To do this, we group together at each step the two nearest classes.

Generally speaking, the number of classes is not fixed beforehand. In the beginning, we have n classes of a single element, ending with a single classes of n data samples. In fact, HAC gives a series of nested partitions of the dataset. Note that after being grouped together, two classes will remain in the same group until the end. For HAC, the choice of the distance to use to assess proximity is of essence and the result depends on the distance considered.

We begin by calculation pairwise distances between all data classes of 1 element and then grouping together the two nearest classes. We iterate this procedure until the granularity (number of classes) we want to have. There is a variety of distances to use. We often examine the increase of the within-class inertia SC_W, going from 0 (for K classes of a single element) up to SC_T (for one class containing all data samples), while simultaneously, between-class variance SC_B decreases from SC_T to 0 as the regroupings continue. We would like the variation of these two terms to be as smooth as possible. This leads us to choosing *Ward's distance*. For two classes C_a, C_b and $a \neq b$, Ward's distance is given by

$$\frac{n_a n_b}{n_a + n_b} \|\bar{\mathbf{x}}_a - \bar{\mathbf{x}}_b\|^2$$

where $\bar{\mathbf{x}}_a, \bar{\mathbf{x}}_b$ are the centers of gravity for the classes C_a and C_b, respectively. We can visualize the evolution of the classification with a dendrogram (Figure 6.2a). The initial classes are located on the dendrogram leaves and subsequent regroupings are given by the red vertical lines connecting two classes. An example is given in Figure 6.2a. We can "cut" the dendrogram at a given height to obtain the desired number of classes. The red horizontal line in Figure 6.2a marks the height at which we have cut the dendrogram to obtain 3 classes.

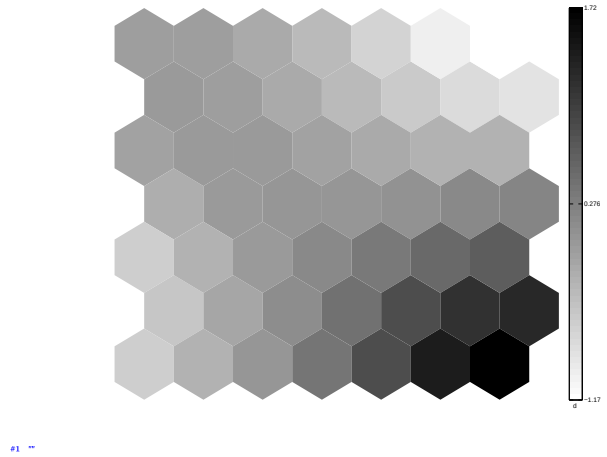


Figure 6.3: Example of an organized map, colored based on the distribution of one of the variables present in the data. Dark-coloured regions correspond to high-valued data samples, while the white ones correspond at low values. The smoothness in the coloring shows that the map has been well organized.

In Figure 6.2b, we can see the within- and between-classes normalized curves (Equations 6.2 and 6.3) versus the number of classes. For $K = n = 14$ classes, the within-class sum of squares is naturally $SC_W = 0$ and thus, the between-class sum of squares is equal to the total sum of squares, $SC_B = SC_T$. On the other hand, for $K = 1$, the between-class sum of squares is equal to $SC_B = 0$.

6.3.2 Self-Organizing Maps

In this section, we present the Self-Organizing Maps (Kohonen, 2001), following the introduction given in (Cottrell et al., 1998; Cottrell and Letrémy, 2005). Given a network of connected units, called Kohonen network or Self-Organizing Maps, with L ordered units according to a given topology. These L units are equipped with a homogeneous neighboring system in the space. For each unit l of the network, we define a neighborhood of radius r, which is, generally speaking, decreasing with time and which we denote $V_r(l)$. It is formed by the set of units which are located at a distance less or equal to r on the network.

Every unit l is represented in the space $\mathbb{R}^p$ by a vector $\mathbf{m}_l$, called prototype vector of unit l. Here, p is the data dimension. The state of the network at time t is given by $m(t) = (\mathbf{m}_1(t), \mathbf{m}_2(t), \ldots, \mathbf{m}_l(t))$.

Note that, to distinguish Kohonen classes from the classification environmental classes of the first step of the procedure, we use numbers $l = 1, \ldots, L$ to refer to the former.

For a given state m and a given data sample $\mathbf{x}$, the winning class $l^*(\mathbf{m}, \mathbf{x})$ is the one whose prototype vector $\mathbf{m}_{l^*}$ is the nearest to the data sample in the sense of a defined distance measure. The winning unit is called *Best Matching Unit* (BMU) for the data

sample $\mathbf{x}$. We thus have

$$BMU(\mathbf{x}) = l^*(\mathbf{m}, \mathbf{x}) = \arg_l \min \|\mathbf{x} - \mathbf{m}_l\|^2$$

Class l is composed of the data samples whose nearest prototype vector is $\mathbf{m}_l$. Every class is being represented by its corresponding prototype vector. Every data sample is being represented by its nearest prototype vector.

Thus, for a given state m, the network defines an application $\Phi_m : \mathbb{R}^p \to \{1, \ldots, L\}$ which associates each data sample $\mathbf{x}$ with its corresponding winning unit, that is, the Kohonen class number. The application Φ_m respects the topology of the input space after convergence of the Kohonen algorithm, meaning that neighboring data samples in the original R^p space are associated with the same or to neighboring units.

The SOM algorithm is an iterative algorithm. Here, we are presenting its stochastic version. The initialization part randomly associates each class with a p-dimensional prototype vector. Then, at each iteration, we choose a data sample randomly, we compare it to each prototype vector and determine its winning class, that is, the class whose prototype vector is closest to the data sample, given a selected distance measure. Having determined the winning class, we bring the prototype vectors of the latter *and* those of the neighboring classes closer to the data sample. This procedure is controlled by the parameter η, which is positive and can be constant or time-decaying.

The stochastic SOM algorithm is defined in an iterative manner as follows:

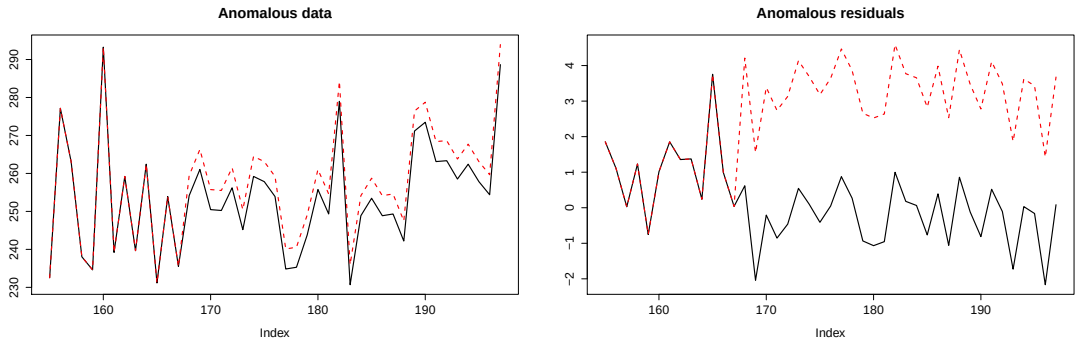
- At time 0, the L prototype vectors are initialized randomly (we can, for instance, draw randomly L data samples).
- At time t, the state of the network is $m(t)$ and we present a data sample $\mathbf{x}(t+1)$ following a probability law P:

$$\begin{cases} BMU(\mathbf{x}(\mathbf{t}+\mathbf{1})) = l^*\left(\mathbf{m}(t), \mathbf{x}(t+1)\right) = \arg\min\{\|\mathbf{x}(t+1) - \mathbf{m}_l\|\,, \quad 1 \leq l \leq L\} \\ \mathbf{m}_l(t+1) = \mathbf{m}_l(t) - \eta(t)\left(\mathbf{m}_l(t) - \mathbf{x}(t+1)\right), \quad \forall l \in V_{r(t)}(l^*) \\ \mathbf{m}_l(t+1) = \mathbf{m}_l(t), \quad \forall l \notin V_{r(t)}(l^*) \end{cases}$$

The output of the algorithm is a set of prototype vectors that define an "organized" map, that is, a map that respects the topology of the data in the input space. We can then color the map according to the distribution of the data for each variable. In this way, we can visually detect regions in the map where low or high values of a given variable are located. Figure 6.3 shows an organized map. High values are located in regions in black, while low values are located in regions in white. The smooth coloring of the map indicates that it has been well organized.

6.3.3 Corrupting data

In order to test the proposed system's capacity to detect anomalies, we need data with anomalies. However, it is difficult to fabricate such data in a test cell environment, since we would actually had to damage the engine or the cell, something prohibitive due to the



(a) (b)

Figure 6.4: An example of an anomaly of the FF variable of the cruise flight data (a) Superposition of the healthy data (solid black lines) and the data with anomalies (dashed red line) (b) Superposition of the corrected (from environmental influence) data obtained from the healthy data and corrected data obtained from corrupted data.

important cost of this equipment. Therefore, we create artificial anomalies by corrupting some of the data based on experts' specifications that have been established following well-known malfunctions of aircraft engines.

Corrupting the data with anomalies is carried out according to a *signature* that describes the defect. A signature is a vector $\mathbf{s} \in \mathbb{R}^p$ where p is the dimension of the data that we want to corrupt. For each variable j that we want to corrupt, s_j is strictly positive, while for the rest of the variables, s is null. Values of s can either be absolute values or percentages of the mean value of the variable. In any case, following s, a corruption term is added to the nominal value of the signal. For a given engine, we randomly pick a data sample with index t_0 between the 1st quartile and the median. We then corrupt every data sample of this engine after $\mathbf{x}_{t_0}$.

Figure 6.4a gives an example of the corruption of the FF variable for one of the engines. Figure 6.4b shows the corrupted variable of the corrected data, that is, after having removed the influence of the environmental variables. We can see that even a rather small anomaly on the original (before correction) data becomes large enough in the corrected data.

6.3.4 Anomaly detection

In this subsection, we present two anomaly detection methods that are based on confidence intervals. These intervals provide us with a "normality" interval of healthy data, which we can then use in the test phase to determine if a novel data sample is healthy or not.

We have already seen that Kohonen's algorithm associates each data sample with the nearest prototype vector, given a selected distance measure. Often, the Euclidean

distance is selected. We calculate, for each data sample $\mathbf{x}_i$, its distance to the map, namely the distance to its nearest prototype vector:

$$d(\mathbf{x}_i) \quad = \quad \min_l \|\mathbf{x}_i - \mathbf{m}_l\|^2 \tag{6.5}$$

where $i = 1, \ldots, n$.

The confidence intervals that we use here are calculated using distances of training data from the map. In the first method, we calculate a confidence interval based on the distances of the training data to their nearest prototype vector. We calculate, thus, $d(\mathbf{x}_i)$, $\forall i$. In the second method, we calculate a confidence interval for each Kohonen class l, where $l = 1, \ldots, L$, that is, $d(\mathbf{x}_i)$ with $\mathbf{x}_i$ in class l. Therefore, the first method has a global flavor whereas the second a local one. The main idea here is that the distance of a data sample from the data samples of a class, thus from its prototype vector will be "small". So, a "large" distance would be the indication of an anomaly. In the following sections, we compare the global and the local methods in terms of detection ability.

Global detection

During the training phase, we calculate the distance $d(\mathbf{x}_i)$, $\forall i$ (Equation 6.5). We can thus construct a confidence interval by taking the 99-th percentile of the distances, $P_{99}(\{d(\mathbf{x}_i),\ \forall i\})$, as the upper limit. The lower limit is equal to 0 since a distance is strictly positive. We define thus the confidence interval $\mathcal{I}$

$$\mathcal{I} \quad = \quad [0, P_{99}(\{d(\mathbf{x}_i),\ \forall i\})] \tag{6.6}$$

For a novel data sample $\mathbf{x}$, we establish the following decision rule:

$$\begin{cases} \text{The novel data sample is considered to be "normal", if } d(\mathbf{x}) \in \mathcal{I} \\ \text{The novel data sample is considered to be an anomaly, if } d(\mathbf{x}) \notin \mathcal{I} \end{cases} \tag{6.7}$$

Local Detection

In a similar manner, in the training phase, we can build a confidence interval for every class l. In this way, we obtain L confidence intervals $\mathcal{I}_l$, $l = 1, \ldots, L$ by taking the 99-th percentile of the *per* class distances as the upper limit

$$\mathcal{I}_l \quad = \quad [0, P_{99}\left(\{d(\mathbf{x}_i) : \mathbf{x}_i \text{in class} l\}\right)] \tag{6.8}$$

Therefore, to build the interval, we take the distance $d_l(\mathbf{x}_i)$ for the data $\mathbf{x}_i$ affected to the class l. For a novel data sample $\mathbf{x}$ (in the test phase), we establish the following decision rule:

$$\begin{cases} \text{The novel data sample,affected to class } l, \text{ is considered to be "normal", if } d(\mathbf{x}) \in \mathcal{I}_l \\ \text{The novel data sample, affected to class } l, \text{ is considered to be an anomaly if } d(\mathbf{x}) \notin \mathcal{I}_l \end{cases} \tag{6.9}$$

6.4 Aircraft flight cruise data

In this section, we present the data that we used for our experiments as well as the preliminary analyses that we carried out on it.

Name	Description
Endogenous variables	
EXH	Exhaustion gas temperature
N2	Core speed
Temp1	Temperature at the entrance of the fan
Pres	Static pressure before combustion
Temp2	Temperature before combustion
FF	Fuel flow
Environmental variables	
ALT	Altitude
Temp3	Ambient temperature
SP	Aircraft speed
N1	Fan speed
Other variables	
ENG	Engine index
AGE	Engine's age

Table 6.1: Description of the variables of the cruise phase data.

6.4.1 Data preprocessing

Data samples in this dataset are snapshots taken from the cruise phase of a flight. Each data sample is a vector of endogenous and environmental variables, as well as categorical variables. Data samples are issued from 16 distinct engines of the same type. For each time instant, there are two snapshots, one for the engine on the left and another one for the engine on the right. Thus, engines appear always in pairs. Note that snapshots in this dataset are rather disparate, *i.e.* they are not continuous. The reader can find a list of variables in Table 6.1. The dataset we used here contains $n = 2472$ data samples and 12 variables.

We have chosen to divide the dataset into a training set and a test set. This configuration is closer to a realistic scenario where an anomaly detection system is first calibrated on some data and then tested on novel data arriving with time.

In order to build the training set, we picked randomly 2000 data samples among the 2472 that we dispose of in total. The test set is composed of the 472 remaining data samples. We have verified that all engines are represented in both sets.

We have sorted data based on the engine ID (primary key of the sort) and for a given engine, based on the timestamp of the snapshot.

6.4.2 Preliminary data analysis

We present methodology and results of some necessary preliminary analysis of the cruise phase dataset. In particular, we examine the number of classes in the data, we perform variance analysis of the variables and we correct data from environmental influence.

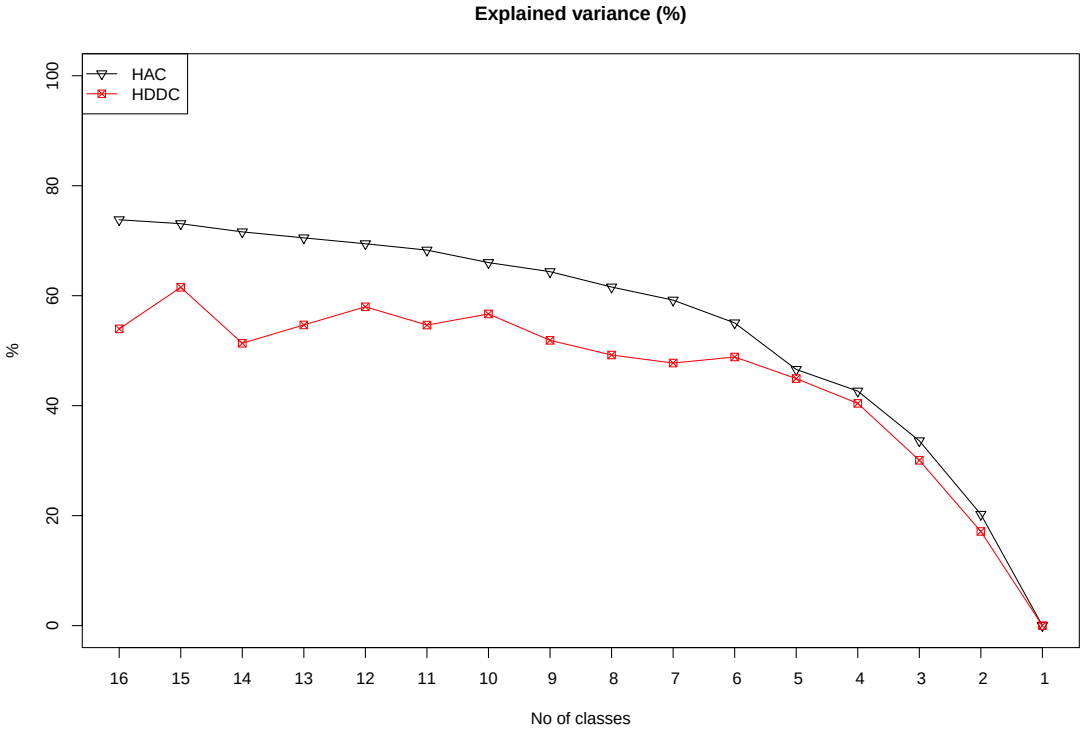


Figure 6.5: Explained variance ($(SS)_K$) for HAC (black triangles) and HDDC (red squeres) for various values of K, the number of classes in classification.

Preliminary remarks

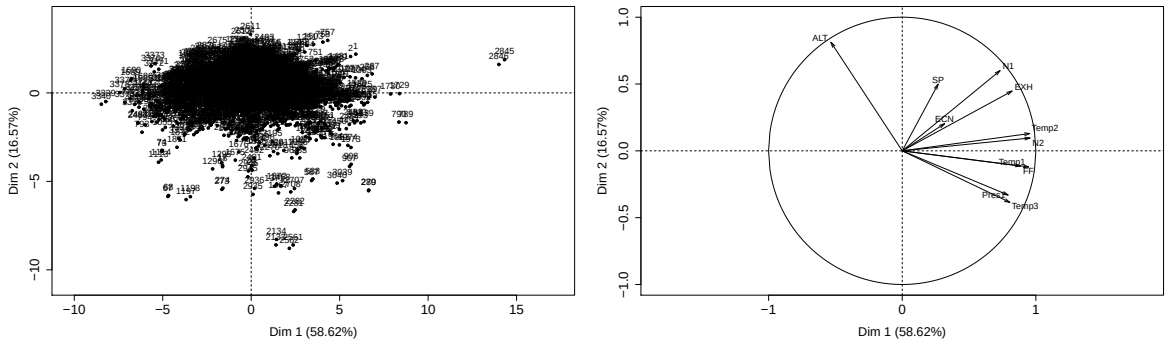
To begin with, we introduce some notation. We use n for the size of the training set, $n = 2000$. The matrix Y designates the endogenous variables, whether in training or in test phase. It is thus a matrix of size $n \times p$, where $p = 6$ is the number of endogenous variables. The matrix X designates the exogenous variables and is $n \times q$, where $q = 4$ is the number of exogenous variables. The matrix $Z = [X, Y, AGE]$ is used to designate all quantitative variables and is $n \times (p + q + 1)$. Variable AGE is the engine's age.

We have chosen to normalize the data (center and scale) because the scales of the variables were very different. We normalize the training sets, keeping the normalization coefficients. When a novel data sample arrives, we normalize it using these stored coefficients calculated in the training phase.

Let us note that preliminary analysis described in the following sections were carried out on the training set.

Selection of the number of classes in classification

We remind the reader that classification is carried out on X (exogenous variables). HDDC (Bouveyron et al., 2007a) is a clustering algorithm, efficient for high-dimensional



(a) (b)

Figure 6.6: (a) Training data (all variables) on the first two principal components of a Principal Component Analysis (b) The correlation circle for the variables.

data, but it is less relevant when the dimension is small. We have thus thought to compare HDDC to a classical classification algorithm. We have chosen the Hierarchical Ascending Classification (HAC) algorithm. An advantage of HAC is that by examining the tree that it builds, we can have a straightforward graphical representation of the explained variance, allowing us to choose the number of classes for the classification.

The first experiment we conduct is to represent graphically, for HDDC and for HAC, the proportion of the explained variance by varying the number of classes from $K = 1$ to $K = 16$

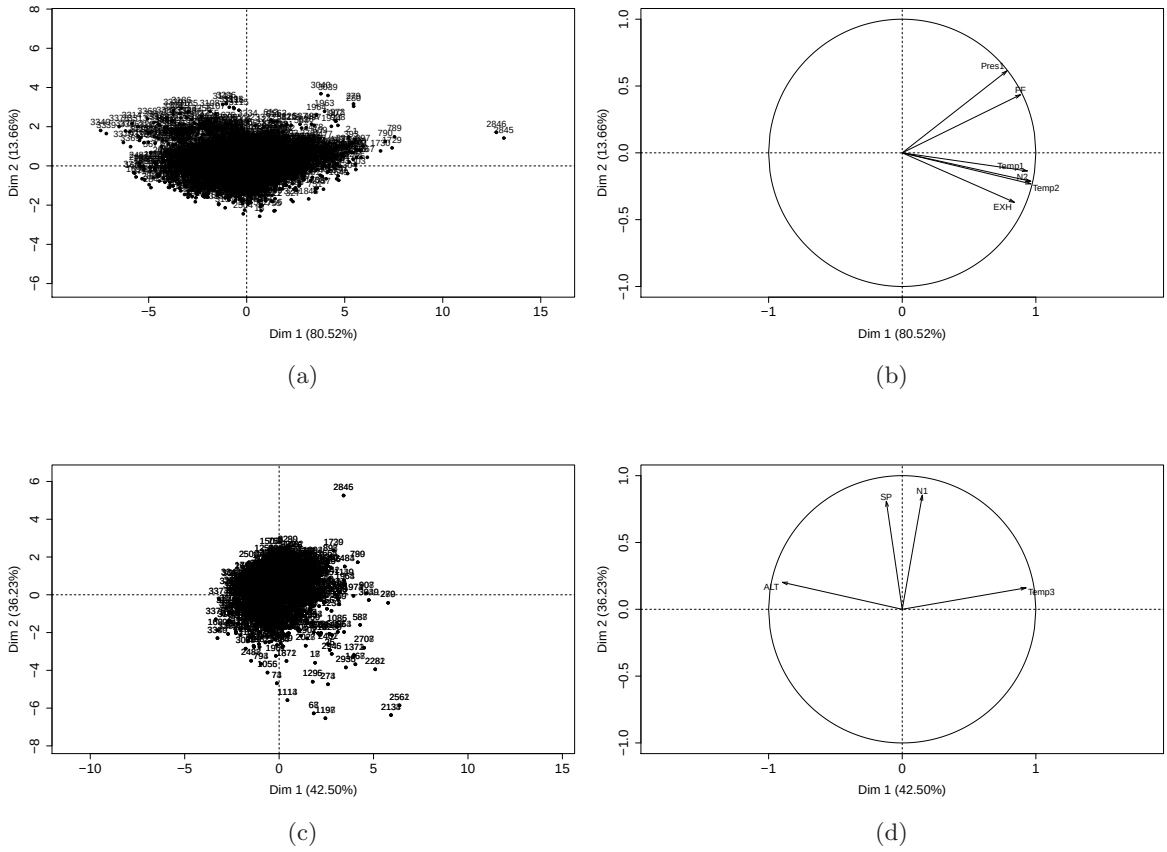
$$(SS_K) = \frac{S_B}{S_T} \quad ,\forall K \in \{1, \ldots, 16\}$$

where n_k is the size of class k, $\bar{\mathbf{x}}_k$ is the mean for class k and $\bar{\mathbf{x}}$ the global mean of the data. The goal is to compare HDDC and HAC in terms of explained variance and choose the number of the classes following the shape of the curve.

Figure 6.5 shows $(SS)_K$ for both algorithms and for different values of K.

This Figure should be read as follows: if the number of classes $K = n$, where n is the size of the dataset, ($n >> 16$, that is why we do not see it in the Figure), the curve is at 100%. On the other hand, if $K = 1$, the percentage of the explained variance falls to zero. Thus, reducing the number of classes K, we lose in explained variance but we gain in interpretation. Therefore, we are actually searching for a compromise between the two.

In Figure 6.5, we see that for $K = 5$, there is a changepoint for both curves. We can thus choose $K = 5$ (5 classes) for our application. Looking at the same Figure, it is clear that HAC manages to explain more variance than HDDC, something due to the small dimension which penalizes HDDC. We note, however, that for $K \leq 5$, the two curves are virtually equal. In this work, our goal was to develop a more general methodology that could process data of all kinds. Consequently, we are particularly



(a) (b)

(c) (d)

Figure 6.7: Training data on the first two principal components of a Principal Component Analysis (a) endogenous and (c) exogenous variables. The correlation circle for the (b) endogenous and (d) exogenous variables.

interested in methods based on subspaces such as HDDC, since they can provide us with a parsimonious representation of high-dimensional data. Thus, we will use HDDC for the environment classification, despite its less good performance for low-dimensional data.

Analysis of variables

After choosing the number of classes, we analyze the quantitative variables. To begin with, we project data on the first two principal axes in Figure 6.6. We can also see the correlation circle for the variables in the same Figure. The closer an arrow is to the perimeter of the circle, the better is the representation of the corresponding variable on the two-dimensional plane.

Looking at the Figure 6.6, well-known aspects of a functional aircraft engine can be found. Variables N2, SP, EXH (their corresponding arrows, more precisely) form a

group, something which is expected since an increase of the aircraft speed, SP, needs an increase of the engine's core speed, N2. This increase in speed increases the temperature of the exhaustion gas (EXH).

Moreover, variables ALT (altitude) and {Temp3 (ambient temperature), Pres (pressure before combustion)} vary in opposite directions. This can be explained by the fact that, in high altitudes, temperature and pressure are lower than in lower altitudes.

In Figures 6.7, we find the same type of graphic but only for the endogenous (a-b) and exogenous variables (c-d). In a similar manner, one can interpret the projection. For example, in Figure 6.7b, core speed, N2, varies in the same way as the fuel flow FF and the exhaustion gas temperature EXH. This behaviour can be explained as follows: to fly faster, more fuel is being burnt and this increases the temperature of exhaustion gases.

Analysis of variance

We also carry out an analysis of variance for each of the $K = 5$ classes to check if classifying the data into environment classes has some sense and to measure the relative discriminant weight for each environmental variable. To this extent, we consider the following model for each of the variables in X (environmental variables)

$$X_{kj} \quad = \quad a_k + \varepsilon_{kj}$$

with $k = 1, \ldots, K$, $j = 1, \ldots, n_k$, $\sum_k n_k = n$ et $\varepsilon_{kj} \sim \mathcal{N}(0, \sigma^2)$.

The test of Fisher allows to test for the hypothesis $H_0 : \alpha_k = \alpha$, $\forall k = 1, \ldots, K$, that is, if the means of all the classes are equal or not. This hypothesis is called the *null* hypothesis. If the null hypothesis holds, then we should reconsider classification of the data, since according to the test, all means are equal and thus, data belong to one single class. Otherwise, the choice to classify the data is founded.

In particular, we test the following hypotheses:

$$\begin{cases} H_0 : & \alpha_k = \alpha \quad \forall k = 1, \ldots, K \\ H_1 : & \exists (l, m) \text{ s.t. } \alpha_l \neq a_m \end{cases}$$

We then need to calculate the (observed) test statistic F_{obs}:

$$F_{obs} \quad = \quad \frac{\left(\sum_k^K (X_{k\cdot} - X_{\cdot\cdot})^2 \right) \Big/ (K-1)}{\left(\sum_k^K \sum_j^{n_k} (X_{kj} - X_{k\cdot})^2 \right) \Big/ (n-K)}$$

where $k = 1, \ldots, K$ is the class index, n_k is the size of class k and $j = 1, \ldots, n_k$ is the index of data affected to class k. The mean of class k is $X_{k\cdot}$ and the global mean of the data is $X_{\cdot\cdot}$. The test statistic follows the F-distribution with p_1 and p_2 degrees of freedom under the null hypothesis, where $p_1 = K - 1 = 4$ and $p_2 = n - p_1 - 1 = 1995$.

The values of the test statistic F_{obs} for the 4 exogenous variables are: $379,4$ for N1, $330,3$ for ALT, $306,6$ for SP and $291,4$ for Temp3. If F_{obs} falls is in the so-called critical region, then we can reject the null hypothesis H_0. The critical region depends on the significance level α we want the test to have, that is, a threshold on the probability that we reject the null hypothesis when it is, in fact, true. This threshold is set prior to the test and we set it at $\alpha = 0.01$.

We can then see, from a probability table for the Fisher distribution, that for 5 and 1995 degrees of freedom and $\alpha = 0.01$, the lower limit of the critical region is given by the critical value of the statistic $F_\alpha \approx 3.03$. The value of the test statistic F_{obs} for all variables are larger than F_α, and so we can reject the null hypothesis. This means that the choice of classifying exogenous data is justified. The larger the value F_{obs} is for a variable, more discriminant that variable is, meaning, informally, that is more "useful" for classification. Thus, we see that $N1$ and altitude ALT are the most significant ones, with the other two following not so far behind.

Correcting the endogenous data from environmental influence

In the flight data that we dispose of, anomalies will manifest themselves in the endogenous variables. At the same time, as expected, environmental conditions play an important role in the functioning of the engine and have an effect on endogenous measures.

Consequently, we are correcting the data using a two-stage procedure: first, we classify exogenous data into environment classes and then, for each of these classes, we have corrected data from environmental influence using a linear model of exogenous and some qualitative variables.

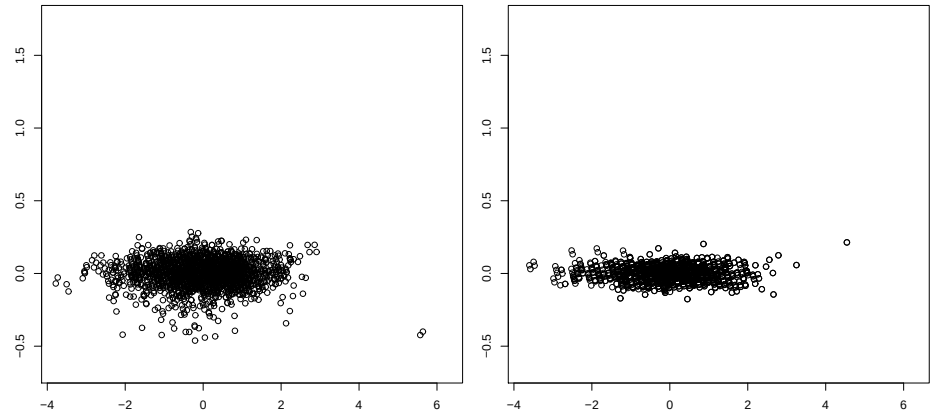
In the following equation, we give, for a given endogenous variable Y, the model used to correct the data. We first set $X^{(1)} = \text{N1}$, $X^{(2)} = \text{Temp3}$, $X^{(3)} = \text{SP}$, $X^{(4)} = \text{ALT}$ et $X^{(5)} = \text{AGE}$.

We obtain thus:

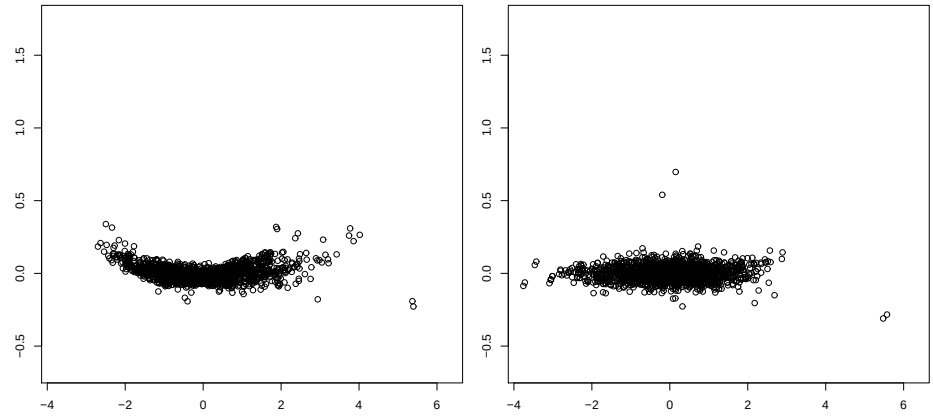
$$\begin{aligned} Y_{rkj} = \mu + \alpha_r + \beta_k + \gamma_{1k} X^{(1)}_{rkj} + \gamma_{2k} X^{(2)}_{rkj} + \gamma_{3k} X^{(3)}_{rkj} + \\ \gamma_{4k} X^{(4)}_{rkj} + \gamma_5 X^{(5)}_{rkj} + \varepsilon_{rkj} \end{aligned} \tag{6.10}$$

where $r \in \{1, \ldots, 16\}$ is the engine index, $k \in \{1, \ldots, 5\}$ is the class number, $j \in \{1, \ldots, n_{rk}\}$ is the observation index. Moreover, μ is the intercept, α_r is the effect of the engine and β_k the effect of the class.

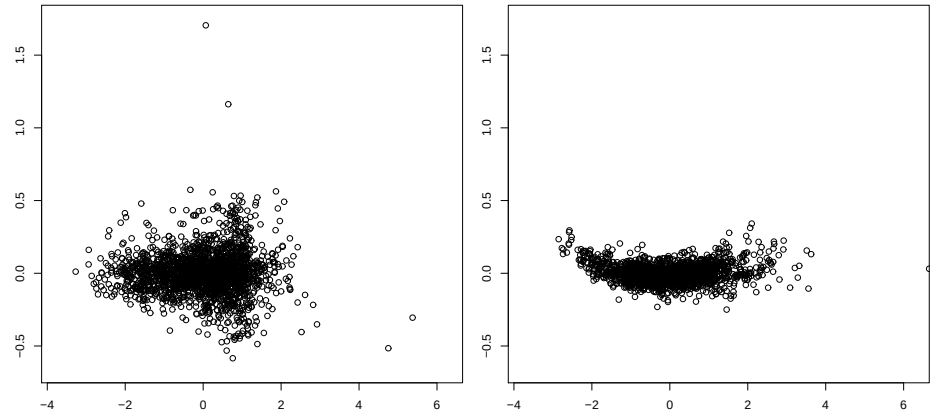
Finally, we note that we have decided to use this model taking into consideration both its precision (in terms of sum-of-squares of its residuals) and the fact that we want a relatively small number of parameters, since the number of the data at our disposal is rather limited.



(a) N2 (b) Temp1



(c) Pres (d) Temp2



(e) EXH (f) FF

Figure 6.8: Training phase residuals versus the predicted values $\hat{Y}$ for the $p = 6$ endogenous variables.

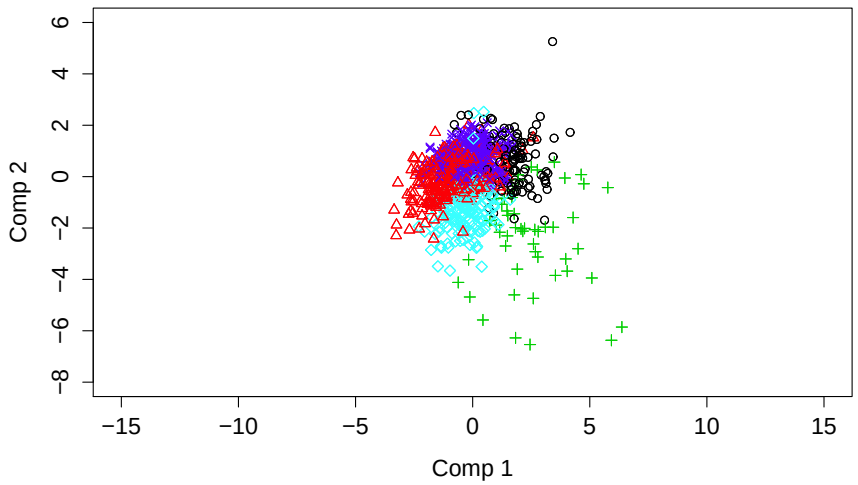


Figure 6.9: Environmental variables on the first two principal axes colored according to the classification of HDDC.

6.5 Experiments and results

In this section, we describe experiments on snapshot data and the results that we obtain. Both the training and the test phase will be described in detail, insisting on each component of the proposed methodology.

6.5.1 Training phase

In the training phase, the goal is to learn the normal state of an engine based on training data which we suppose are anomaly-free.

We begin by classifying data into $K = 5$ environmental classes. Figure 6.9 shows exogenous data on the first two principal axes with the classification given by HDDC annotated in color.

Once we have classified data into environment classes, we can correct data from the influence of the environment following the model of Equation 6.10.Figure 6.8 shows the regression residuals plotted versus the predicted values for each of the endogenous variables. It is clear that the model succeeds in capturing the influence of the environment on the endogenous measures, since the magnitude of the residuals is rather small. The residuals will therefore capture behaviors of the engine which are not due to environmental conditions. For variables Pres and FF, we see that there is a curve and not a sort of "homogeneous band" like for the rest of the variables. This means that we can indeed refine our model, possibly by adding interactions between certain variables or second or higher degree terms to the model. However, we have chosen to keep the model rather simple, judging that its power is satisfying for the purpose it is intended.

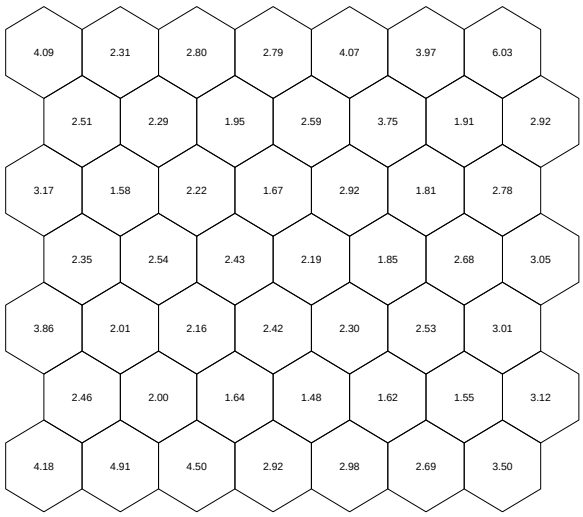


Figure 6.10: The upper limits of the local confidence intervals for the Kohonen classes. The lower limit is 0 for all intervals.

The residuals are expected to be centered, *i.e.* to have a mean equal to 0. However, they are not necessarily scaled, as they can have a variance larger than 1. We choose to rescale them, keeping the rescaling coefficients in order to rescale in a similar way the test phase residuals.

Moreover, we notice that data are characterized by small variations, *i.e.* they are not smooth. We thus choose to smoothen them with a moving average. We have used a smoothing window of width $w = 7$ (central element plus 3 elements on the left plus 3 elements on the right). We note that by smoothing, we lose $\lfloor \frac{w}{2} \rfloor$ data samples from the beginning and the end. Therefore, in this case, we end up with a set of 1994 residual samples instead of the 2000 that we had initially.

Next, we construct a Kohonen map from the training phase residuals (Figure 6.11). We have used existing MATLAB code of the SAMANTA software for the Kohonen maps, enhancing the code to suit our needs. Figure 6.17 shows the size of environment classes for each Kohonen class.

We can now calibrate the anomaly detection component by determining the global and local confidence intervals based on the distances of the data to the map.

For the global case, according to Equation 6.6, we have:

$$\mathcal{I} \quad = \quad [0, 4.1707]$$

For the local case, the upper limits of the local confidence intervals on the cell of the corresponding Kohonen class can be found in the Figure 6.10. We note that the lower limit of all intervals is set to 0, since they are all built from distances, which are strictly

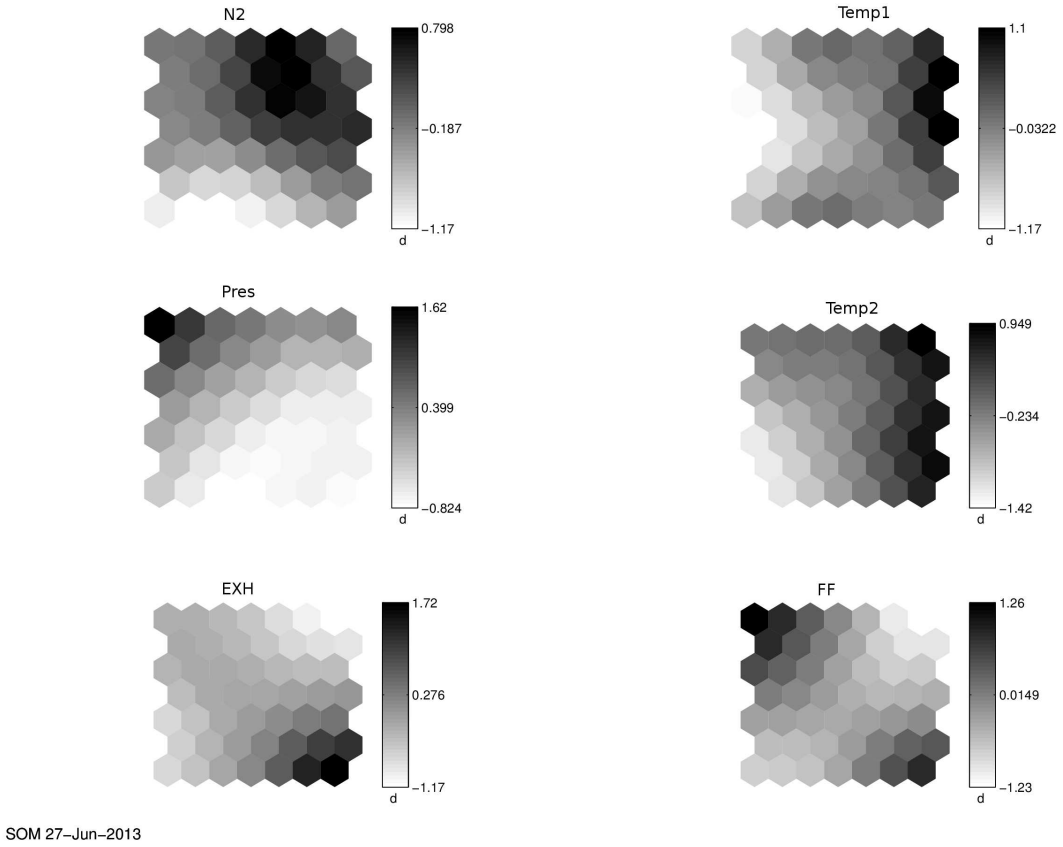


Figure 6.11: SOM built from the corrected training residuals for each of the $p = 6$ endogenous variables. Black cells contain high values of the variable while white ones contain low values.

positive.

6.5.2 Test phase

In the test phase, we assume that novel data samples are being made available. We first corrupt this data following the technique proposed in Section 6.3.3. Thus, some of the data is corrupted, but this is not known by our system.

The following step is to apply the same processing chain that we calibrated on using the training data. We start by normalizing test data with the coefficients we used to normalize training data earlier. We then classify data into environment classes using the model parameters we estimated on the training data earlier. For HDDC, this means that we just execute the classification step (E step) of the algorithm. Thus, we do not relearn the model. Next, we correct data from environmental influence using the model

we built on the training data. In this way, we obtain the test residuals, that we rescale with the same scaling coefficient used to rescale training residuals.

Given the fact that residuals are, in general, not smooth, we apply a smoothing transformation using a moving average, exactly like we did for training residuals. We use the same window size, *i.e.* $w = 7$. Smoothing causes some of the data to be lost, so we end up with 466 test residuals instead of the 472 we had initially.

Finally, we apply the decision rule, either the global decision rule of (6.7) or the local decision rule of (6.9) and we project data onto the Kohonen map that we built in the training phase.

Snecma experts provided us with descriptions of 12 known defects (anomalies), that we added to the data. For data confidentiality reasons, we are obliged to carry out an anonymization the defects and not to give any detail of their signatures (which variables they affect and to which extent). We will thus refer to these defects as "Defect 1", "Defect 2" etc.

Since the defects were added artificially, we know if a data sample is abnormal or not. In order to evaluate our system, we calculate the detection rate (tpr) and the false alarms rate (pfa). We have already defined these measures but we repeat their definition for convenience:

$$tpr = \frac{\text{number of detected anomalies}}{\text{total number of anomalies}}$$

$$pfa = \frac{\text{number of non-expected detections}}{\text{number of detections}}$$

6.5.3 Results

In Table 6.2, we can see detection results for all 12 defects and for both detection methods (global and local).It is clear that both methods succeed in detecting the defects, almost without a single miss. The global method has a lower false alarm rate than the local one. The high false alarm rate for the local method is mainly due to the small number of data that we dispose of for the specific map size. In other words, during training, each Kohonen class gets few data samples and thus, confidence intervals cannot be calculated reliably.

Figure 6.12 shows the distance d of each data sample (samples on the horizontal axis) to their nearest prototype vector (Equation 6.5). The light blue band shows the global confidence interval $\mathcal{I}$ that we calculated in the training phase.

Similarly, Figure 6.13 shows the distance d of the test data to their nearest prototype vectors (Equation 6.5). The light blue band corresponds to the local confidence intervals $\mathcal{I}_l$, with $l = 1, \ldots, L$ being the index of the Kohonen classes, that we aggregated on the same graphique, meaning that for each data sample, we plotted the upper limit of the confidence interval of the Kohonen class to which it was affected. This is what gives a variable band in the graphic.

	Global detection		Local detection	
Defect	**tpr**	**pfa**	**tpr**	**pfa**
Defect 1	100%	18,9%	100%	45,4%
Defect 2	100%	11,4%	100%	42,6%
Defect 3	100%	16,7%	100%	47,9%
Defect 4	100%	15,1%	100%	45,1%
Defect 5	96,7%	14,7%	100%	43,4%
Defect 6	100%	13,9%	100%	43,6%
Defect 7	96,7%	12,1%	96,7%	44,2%
Defect 8	100%	26,3%	100%	50%
Defect 9	100%	15,8%	100%	43,9%
Defect 10	100%	26,7%	100%	55,1%
Defect 11	100%	17,1%	100%	46,3%
Defect 12	100%	21%	100%	46,4%

Table 6.2: Detection rate (tpr) and false alarm rate (pfa) for different types of defects and for both anomaly detection methods (global and local).

In order to give a visual overview of the performance of the anomaly detection component, we annotate the detection on the above graphics. Thus, red crosses designate the false alarms and green stars designate the correct detections.

Moreover, we have developed a new visualization tool to monitor the trajectory of an engine with a defect on the map. The tool was built using SAMANTA, the proprietary Health Monitoring software of Snecma.

Using this tool, we visualize the distribution of the corrupted data in the cells of the map. In general, we want that anomalies appear to regions of the map where extreme data exist (black or white regions). Note that, for a given engine, data are corrupted after a certain data sample $\mathbf{x}_{t_0}$ and up to the last data sample of this engine. Therefore, this visualization has a green circle for anomaly-free data samples and a red circle for anomalies. At each time, it is only the most recent data sample that is being colored red or green. Here, "anomaly-free" refers to the groundtruth and not to the decision of our system.

Figures 6.18-6.23 show snapshots of the trajectory of an engine with Defect 1. The background of the maps is the exhaustion gas temperature (EXH). We can see that in the first graphics, anomaly-free data (green circles) appear in map regions with intermediate values for the given variable. Then, for defects (red circles), that is after $t = 15$, data appear abruptly in a region of the map with high values for the given variable and stay there until the end.

This experiment shows that our system makes a representative visualization of anomaly-free data, since anomalies tend to concentrate on high-valued regions.

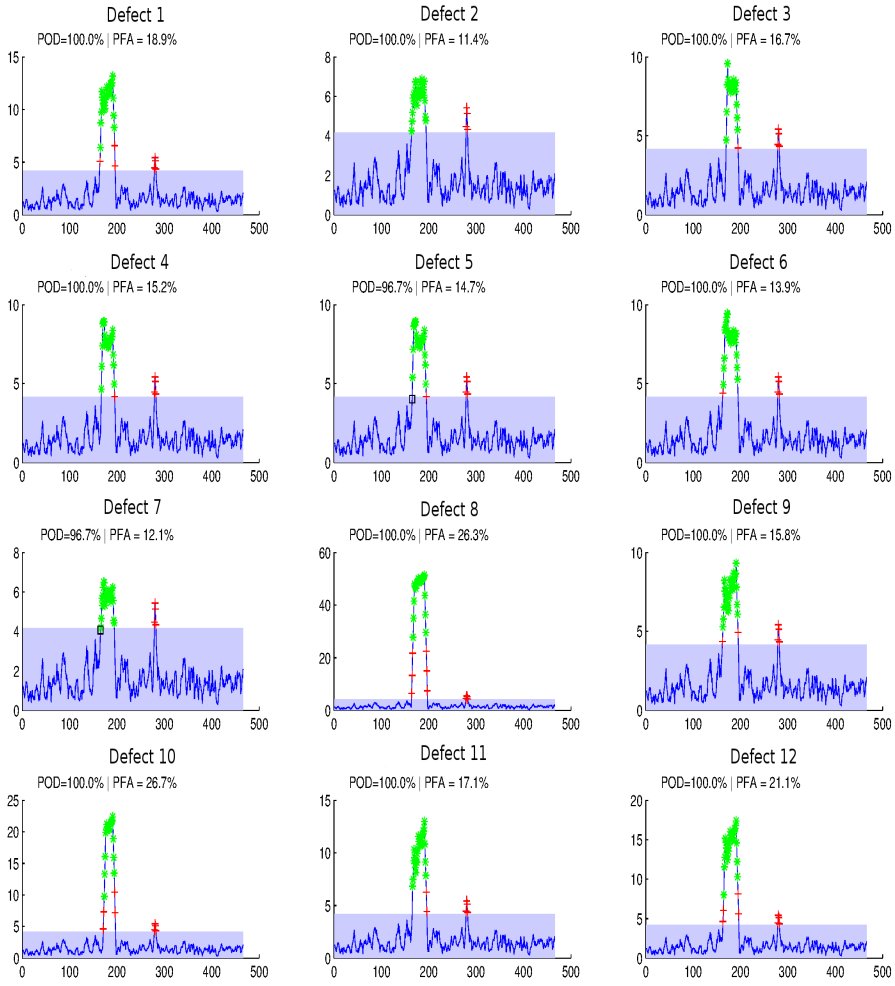


Figure 6.12: Distances of the test data to their nearest prototype vector and the global confidence interval (in light blue), calculated using anomaly-free data during the training phase. Red crosses show the false alarms and green stars show successful detections. We annotate to each graphic the tpr and the pfa.

6.5.4 Comparison of the SOM-based anomaly detector with HDRC

An interesting point is to examine if we have gained something by using the SOM-based anomaly detector which relies on the calculation of "normality" intervals. To this extent, we compare the performances of the SOM-based detector (Table 6.2) with the detection and false alarm rates that HDRC would give on the same test data.

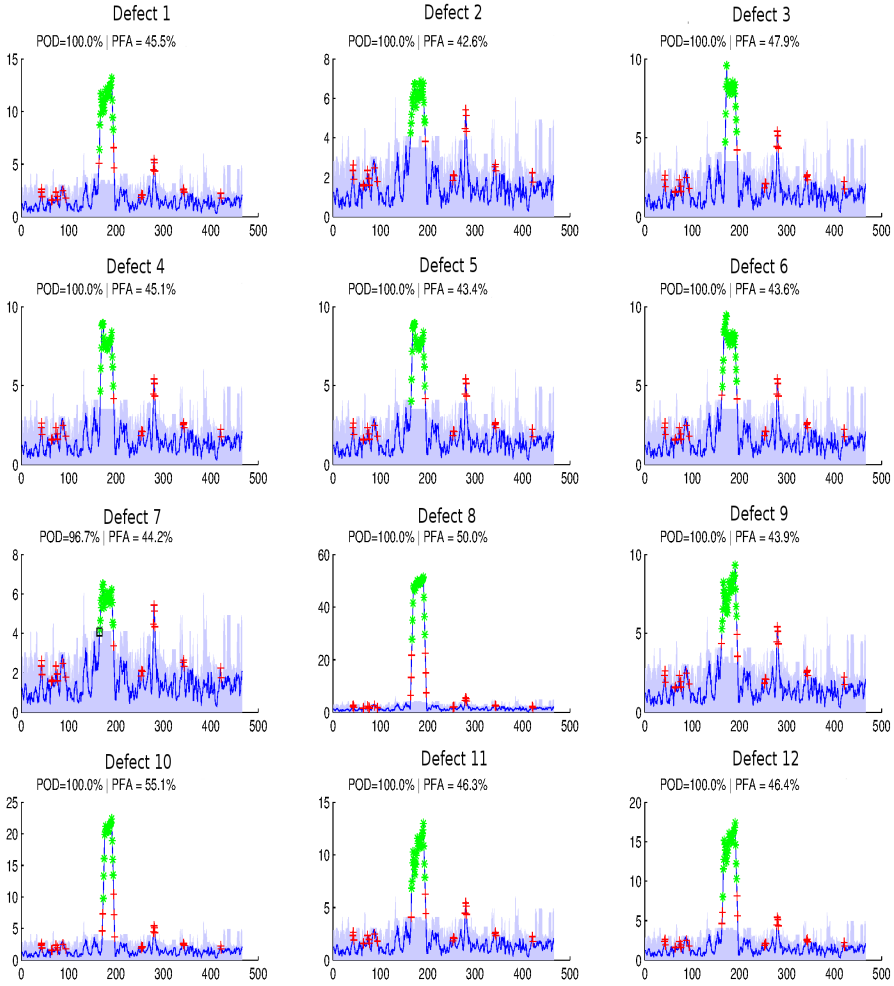


Figure 6.13: Distances of test residuals to their nearest prototype vector and local confidence intervals (for each Kohonen class) in light blue, aggregated in one single graphic. This Figure can be read as the one for the global interval above.

Therefore, for each defect type, we run HDRC 30 times with a random initialization and we calculate the mean tpr and mean pfa. We give HDRC the true value of the anomaly proportion in the data $\alpha = 0.064$. Remember that we correct endogenous data so that we eliminate the influence of the environment. This gives us (corrected) data where anomalies are detected due to their having far larger values than "normal" data. Therefore, it seems that, naturally, there are two kinds of data in the corrected data:

Defect	tpr	pfa
Defect 1	93,3%	6,67%
Defect 2	83,9%	16,1%
Defect 3	88%	12%
Defect 4	89,3%	10,7%
Defect 5	90%	10%
Defect 6	87,1%	12,9%
Defect 7	83,3%	16,7%
Defect 8	89,3%	10,7%
Defect 9	90,6%	9,4%
Defect 10	86,4%	13,6%
Defect 11	89,7%	10,3%
Defect 12	90%	10%

Table 6.3: Detection rate (tpr), false alarm rate (pfa) obtained by HDRC for different types of defects. Rates are mean values over 30 replications. Comparing these rates with those in Table 6.2, it is clear that the SOM-based anomaly detector clearly does better than a simple, straightforward application of HDRC.

healthy data and anomalies. This is why we set $K = 1$, so that the algorithm will concentrate on anomalies (given that it clusters correctly healthy data into the single cluster).

Detection rate (tpr) and false alarm rate (pfa) obtained by HDRC for different types of defects are shown in Table 6.3. Rates are mean values over 30 replications. Comparing these rates with those of the SOM-based detector in Table 6.2, it is clear that the SOM-based anomaly detector clearly does better than a simple application of HDRC. The false alarm rates of HDRC are slightly lower in some cases but not in a significant manner that could counterbalance the loss in detection.

6.5.5 Anomaly Identification

In this application, we have used Self-Organizing Maps as a means to visualize anomalies and calibrate the anomaly detection component of our system. An interesting point would be to see if SOMs could facilitate anomaly identification. Intuitively, we expect similar types of anomalies to be located at nearby regions on the map. Moreover, we would like that these regions are not the same regions where anomaly-free data are located. If this was true, we could have yet another method for detecting anomalies. In addition, if similar anomalies appear frequently in the same region of a SOM, then this would provide us with a means of identifying them. We proceeded to some experiments to see if these facts hold.

SOMs actually provide a natural way of monitoring changes by following the *trajectory* of successive data samples on the map. Their trajectory is defined by the index of the class to which each data sample was assigned, *i.e.* the cell corresponding to the closest

prototype vector. Since the data that we dispose of are aircraft engine data, we speak of engine trajectories. Therefore, we would like to see if similar anomalies on an engine would gives us similar trajectories. In addition, we would like to compare a trajectory of an engine without any incidents (anomalies) to that of an engine with anomalies. Moreover, an interesting point would be to see if different anomalies on different engines produce different trajectories (similar per anomaly category).

Trajectory of a single-engine anomaly of a single type

We begin by generating 100 engine anomaly signatures of type "Defect 1" for a given engine and plotting their trajectories on the SOM that we constructed earlier in Figure 6.14. Each trajectory is represented by a different color in Figure 6.14a. The symbols (triangles, circles etc.) mark the endpoint of each trajectory. We see that all trajectories are not identical but they tend to end in two distinct regions of the map (upper left and lower right corner). Moreover, these are regions where high values of tha examined variable are located. Figure 6.14b shows only the endpoints of each trajectory and corroborates the above conclusion. Thus, variations of a similar anomaly on the same engine will produce trajectories that are not identical but are similar in the sense that they concentrate over particular regions of the map.

Comparison of trajectories of healthy and non-healthy engines

The next experiment that we conduct is to compare the trajectory of a normal engine to that of an engine with an anomaly. We thus generate an anomaly signature of type Defect 8 and compare it to the trajectory of the same engine without any anomalies. Figure 6.15 illustrates this comparison. In green, there is the trajectory of the healthy engine. The trajectory of the engine with anomalies is in blue (healthy part) and red (part where the anomalies occur). We remind the reader that the blue part precedes the red part temporally speaking. We can see that the healthy engine traverses virtually the entire map, while the damaged engine is heading to a region of the map where high values of the examined variable are concentrated.

Trajectories of multiple-engine anomalies of various types

Finally, we would like to see if each type of anomaly has a specific region to which it is concentrated on the map. To this extent, we generate 400 signatures of 4 different types of anomalies (100 signatures of each) and we plot the ending points of their trajectories on the map. In particular, we generated anomalies of types "Defect 1", "Defect 7", "Defect 8" and "Defect 9". This time though, anomalies are not injected to a single engine but to multiple engines. The ending points can be found in Figure 6.16. Each color-symbol represents a different type of anomalies. In particular, green crosses were used for anomalies of type "Defect 1", magenta triangles for "Defect 7", yellow squares for "Defect 8" and blue diamonds for "Defect 9".

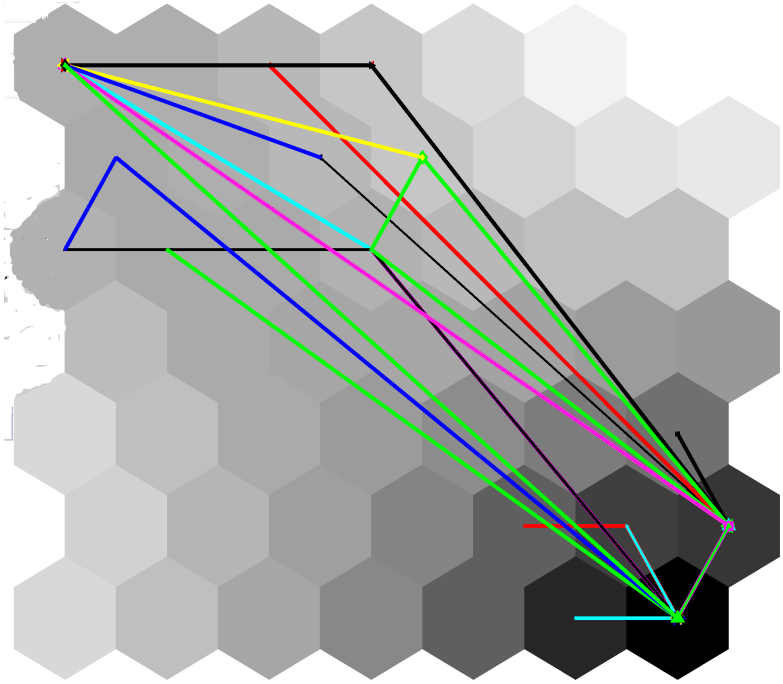
We can see that different anomalies tend to concentrate on different regions of the map, though the separation is not always clear. Note that types "Defect 1" and "Defect

9" describe similar anomaly signatures and that "Defect 7" describes an anomaly which is hard to detect. That being said, we can see that anomaly signatures of type "Defect 1" and "Defect 9" concentrate on the same regions of the map, while "Defect 8" is located in a distinct area of the map. Anomaly signatures of type "Defect 7" seem to be more dispersed due to the fact that it is a challenging type of aircraft engine anomaly.

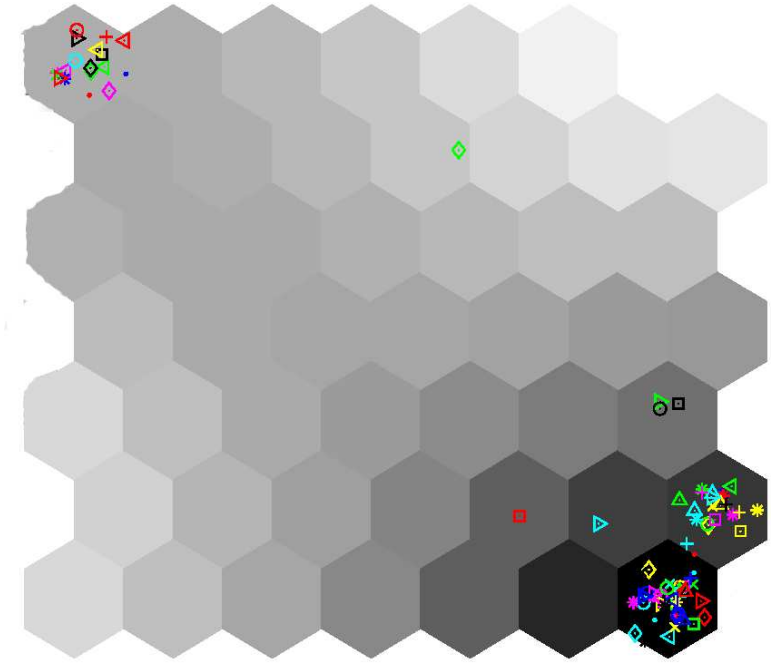
6.5.6 Dynamic Anomaly Detection

The Anomaly Detection system presented earlier is dynamic in the sense that once the training phase has been completed, each new data sample can be treated rapidly since the classification rule has been learned, the regression model has been estimated, the weights of the SOM have already been calculated and the anomaly intervals have been set. Thus, for a novel data sample we just perform five constant-time (O(1)) operations: apply the MAP rule for classification, calculate its residual from the regression model, find its closest prototype vector in the SOM, calculate its distance to the map and compare with the confidence interval.

To go further and have a fully online system, where there will be no training or test phase, one has to execute classification, regression and SOM learning incrementally. For the classification part, the online MPPCA algorithm that we introduced in this work can be used.



(a)



(b)

Figure 6.14: (a) Trajectories of a single engine with 100 different signatures of Defect 1. Each trajectory is represented by a different color. (b) Endpoints of trajectories. Each endpoint is represented by a different color and symbol. The background of the map is the exhaustion gas temperature (EXH).

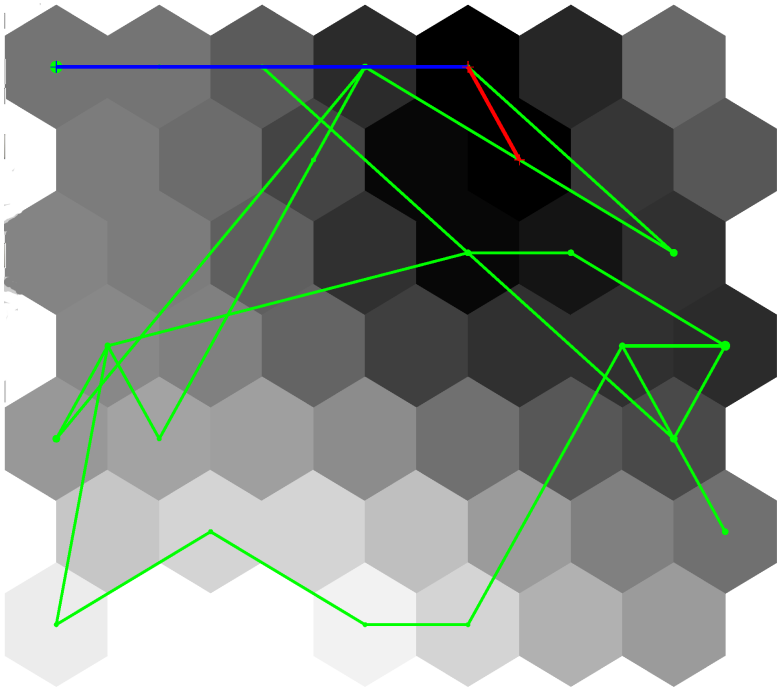


Figure 6.15: Trajectories of a healthy engine (green) and of the same engine with an anomaly of type Defect 8 (blue for the healthy part and red for the anomalous part). The background of the map corresponds to the N2 variable .

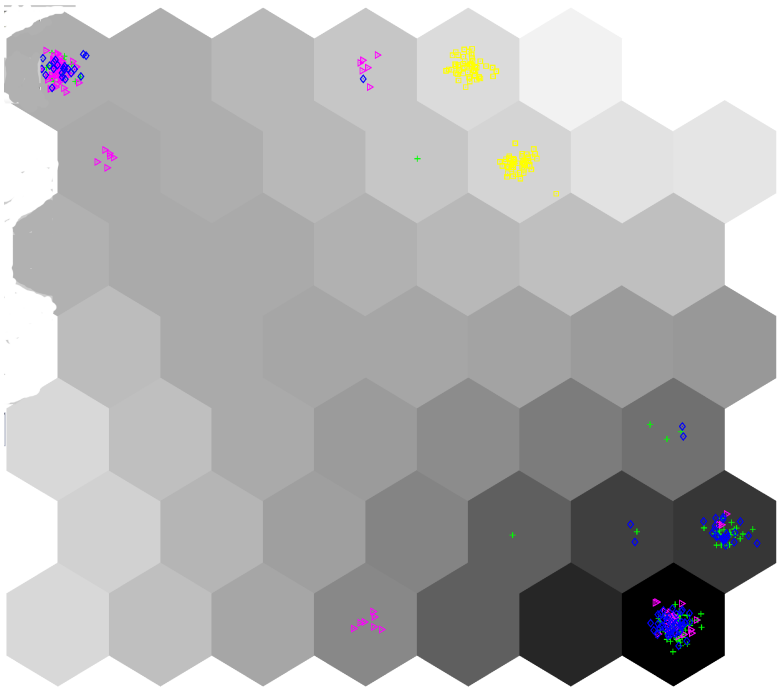


Figure 6.16: Ending points of trajectories of 400 anomaly signatures of 4 different anomaly types. Each type is represented by a different color-symbol: anomalies of type "Defect 1", magenta triangles for "Defect 7", yellow squares for "Defect 8" and blue diamonds for "Defect 9". Note that types Defect 1 and 9 describe similar anomaly signatures and that Defect 7 describes an anomaly which is hard to detect. We can see that different anomalies tend to concentrate on different regions of the map. The background of the map is the exhaustion gas temperature variable (EXH).

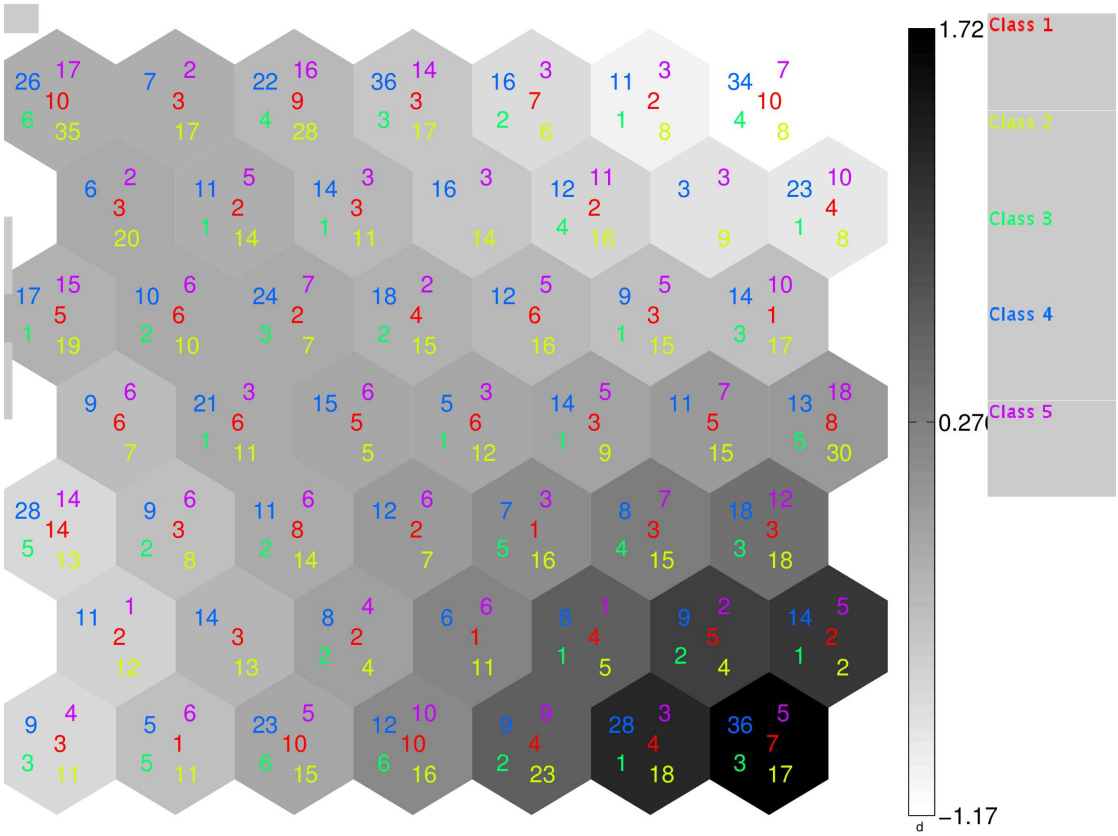
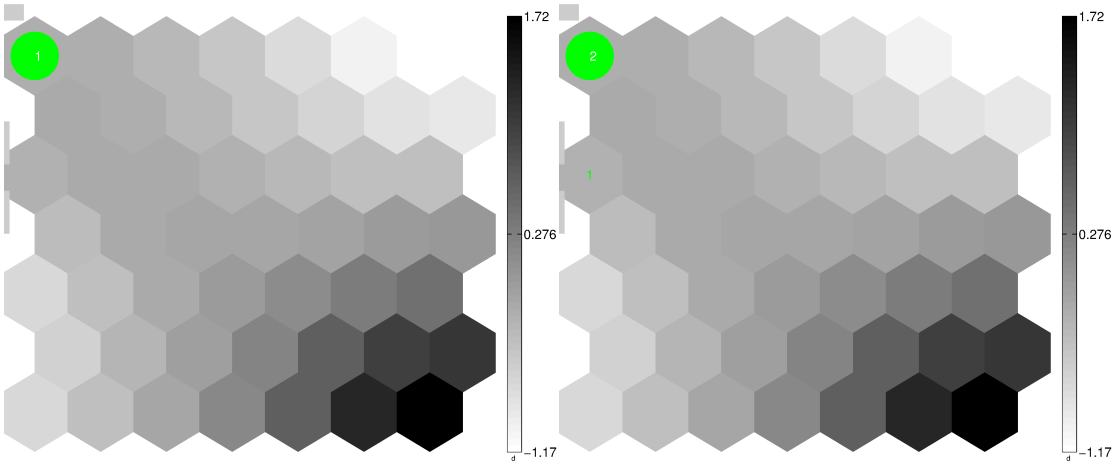
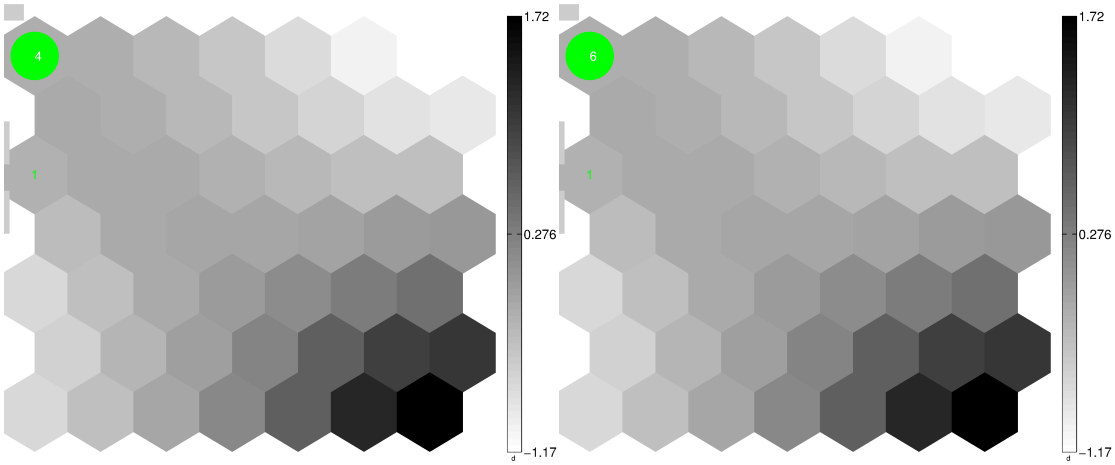


Figure 6.17: Distribution of training residuals to the $K = 5$ environment classes ($C_1, \ldots, C_K$, see Section 6.3), superposed on the map. The background of the map is the exhaustion gas temperature (EXH). For example, for the cell $(1, 1)$ (top left), we have 2 data samples of C_1, 6 of C_2, 1 of C_3, 17 of C_4 and 12 of C_5.

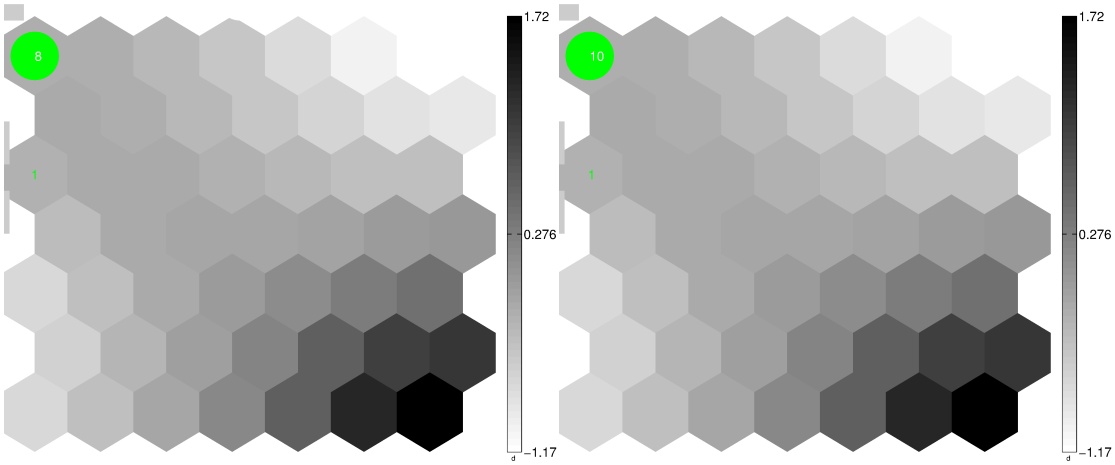


(a) t=1 (b) t=3

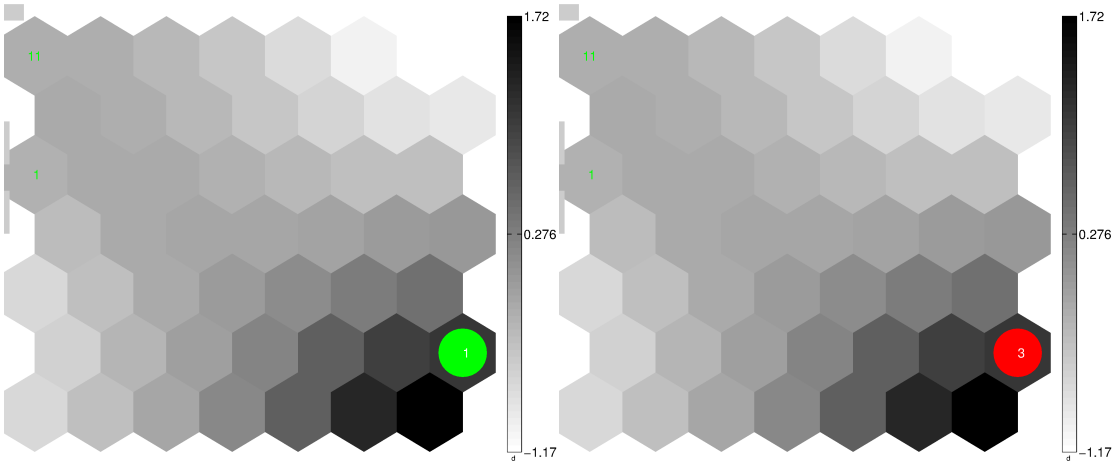


(c) t=5 (d) t=7

Figure 6.18: Trajectory of an engine with defects. The background of the map is the exhaustion gas temperature (EXH). See the text (p. 100) for more details.

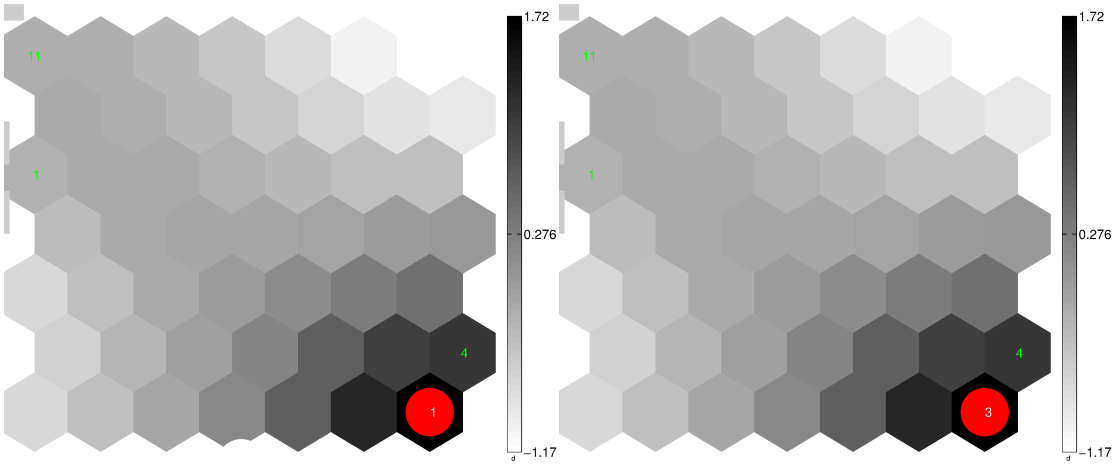


(a) t=9 (b) t=11



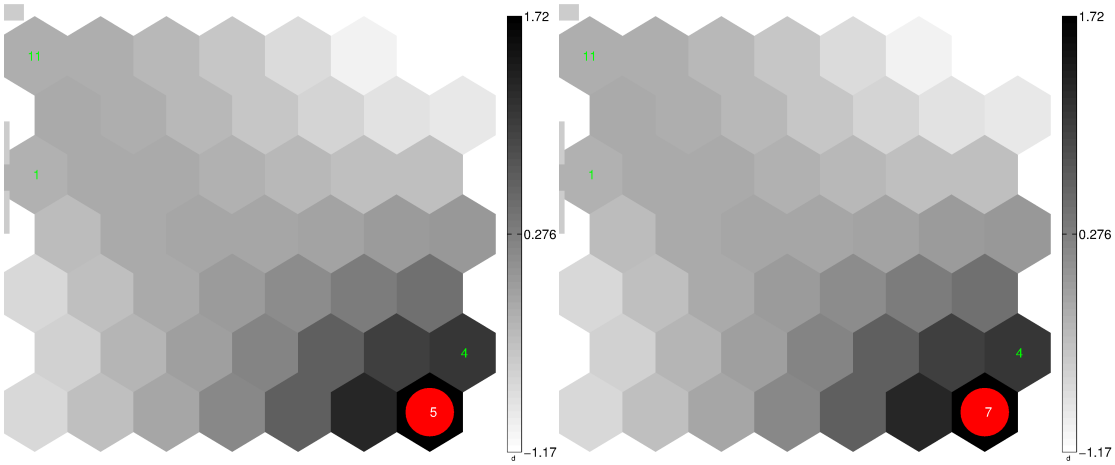
(c) t=13 (d) t=15

Figure 6.19



(a) t=17

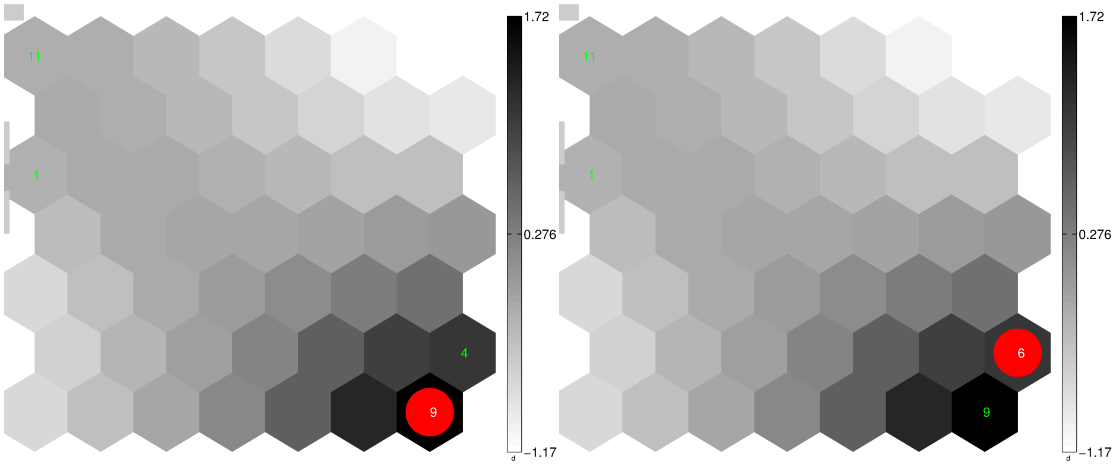
(b) t=19



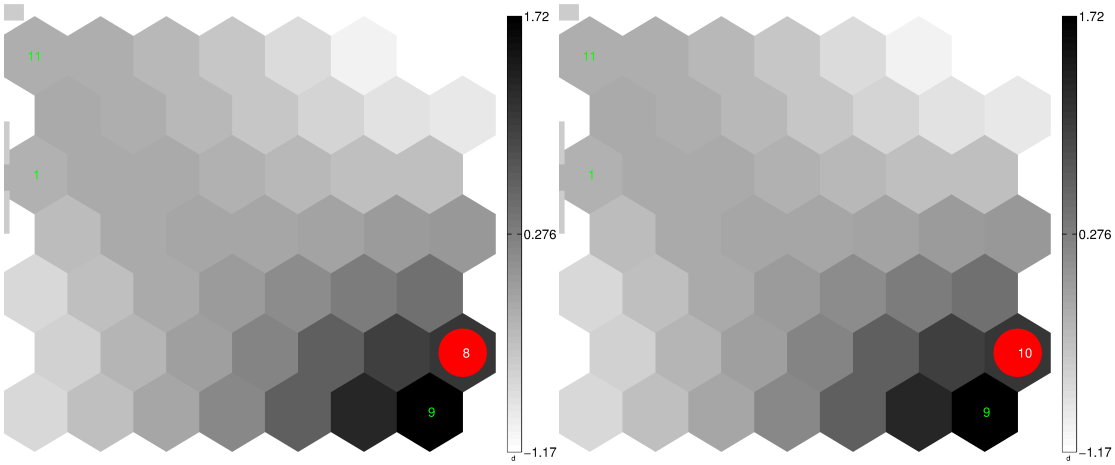
(c) t=21

(d) t=23

Figure 6.20

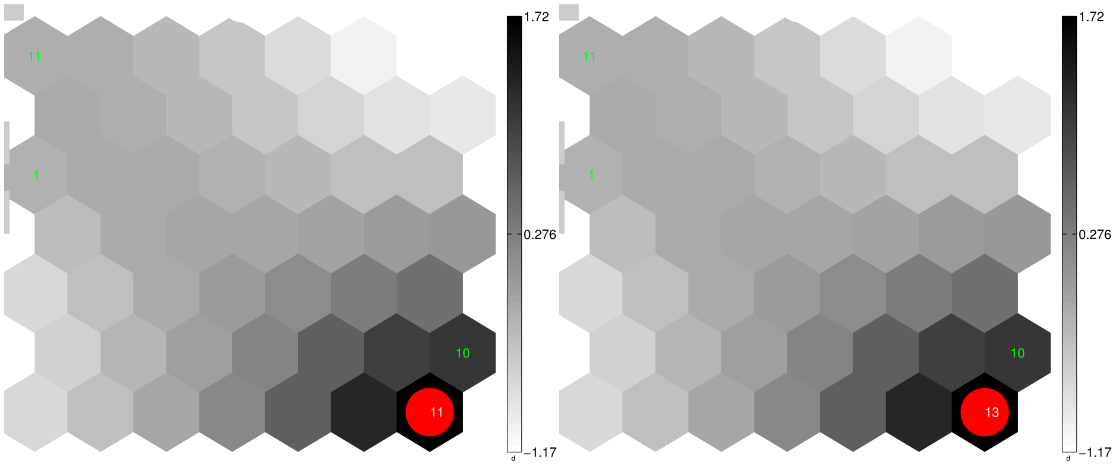


(a) t=25 (b) t=27



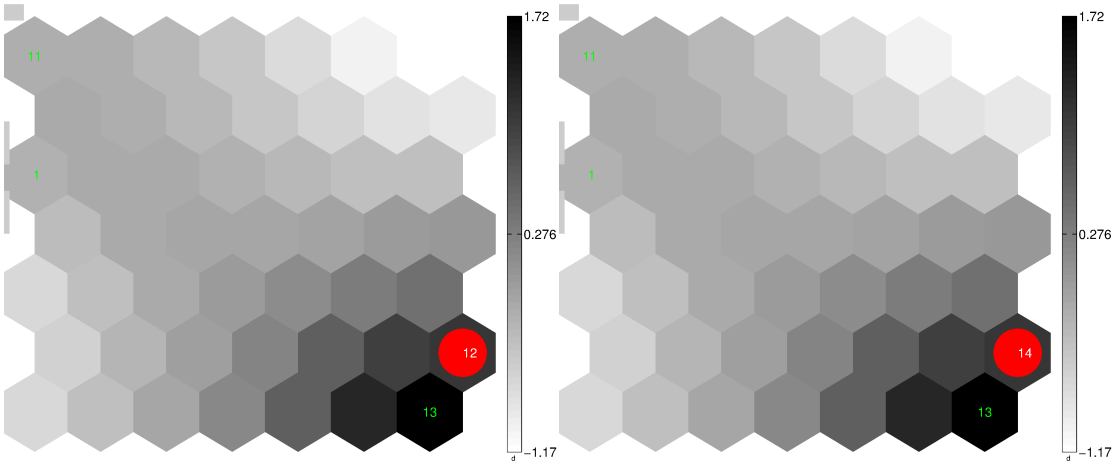
(c) t=29 (d) t=31

Figure 6.21



(a) t=33

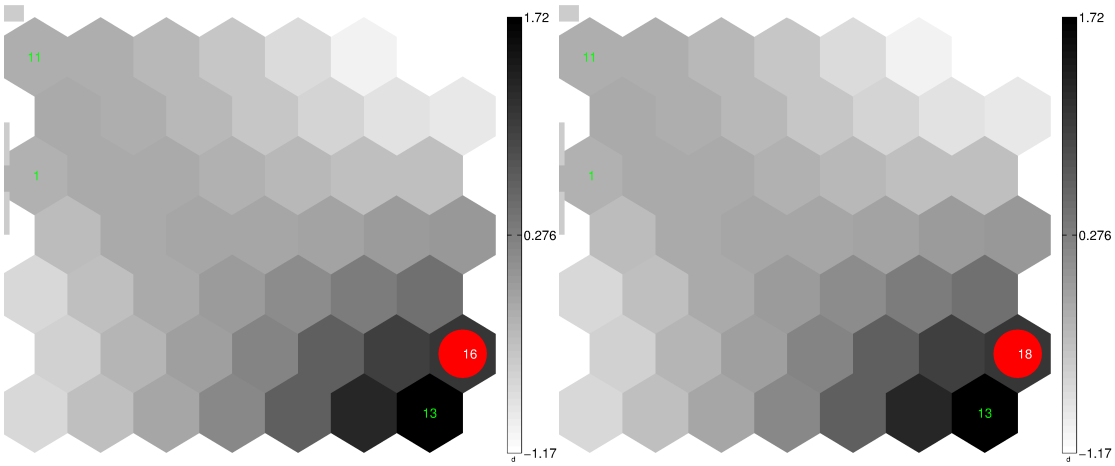
(b) t=35



(c) t=37

(d) t=39

Figure 6.22



(a) t=41 (b) t=43

Figure 6.23

Chapter 7

Conclusions and future work

In this chapter, we give a synthesis of the contributions of this Thesis, of the proposed methods and of the results of the experiments that we conducted (section 7.1). We also discuss perspectives for future work on the subject (section 7.2).

7.1 Synthesis of our work

The main subject of this Thesis is anomaly detection in high-dimensional datastreams with a specific application to aircraft engine health monitoring data. Below, we give a summary of the contributions of this Thesis.

Robust offline clustering with trimming for high-dimensional data

We have presented a robust offline clustering algorithm for high-dimensional data (HDRC), combining a subspace clustering algorithm (HDDC) and the trimming technique. We have shown its efficiency on noisy high-dimensional simulated datasets and on two real-world datasets with anomalies. In particular, HDRC manages to detect anomalies in high-dimensional (up to 173) real-world aircraft engine data, provided by Snecma, the french aircraft engine constructor. HDRC was also successful in breast cancer diagnosis using the breast cancer dataset of the UCI repository.

Online mixture of probabilistic PCA for clustering of high-dimensional datastreams

We have also proposed an online inference algorithm for the MPPCA model which relies on an EM-based procedure and a probabilistic, incremental version of PCA. The proposed strategy allows to incrementally update the estimates of the MPPCA parameters, at the arrival of a new data sample. It also provides low-dimensional visualizations of the data. Model selection is also considered in the online setting through parallel computing. Numerical experiments on simulated and real data from the aircraft engine Health Monitoring domain have shown that the online MPPCA algorithm performs better than existing online EM-based algorithms in high-dimensional spaces.

An integrated anomaly detection methodology with an application to Health Monitoring

In this Thesis, we have developed an integrated methodology for the analysis, detection and visualization of anomalies of aircraft engines, that includes the following stages: classification of the data in environment classes, data correction from the influence of the environmental conditions, construction of a self-organizing map from the corrected data and anomaly detection. There is a training phase in which the components of the overall system are calibrated using healthy data. Once training is completed, there is a test phase, where the system is fully operational and can process data that was not seen during training. In this phase, the same chain of processing as in the training phase is applied to each novel data sample. For the anomaly detection component, we have developed a statistical technique that builds intervals of "normal" functioning of an engine based on distances of healthy data from the map. Then, for a novel data sample, its distance from the map is calculated and compared with the interval in order to issue an alert or not.

We have tested this methodology on real data from the aircraft engine Health Monitoring Domain, supplied by Snecma. We have injected artificial anomalies of 12 different types, as they were established by Snecma experts. We have shown that the proposed system succeeds in detecting anomalies attaining even 100% of detection for most of the anomaly types. The false alarm rate varied from $11\% - 26\%$ for the global detection anomaly component. Note that this method does better in anomaly detection than a straightforward application of HDRC to corrected endogenous data.

7.2 Future work

HDRC, as trimming-based robust clustering methods in general, assume that the true anomaly proportion among the data is known. In practice, this is rarely true. Moreover, the data that are trimmed depend on the number of groups K of the underlying GMM and vice versa, since the final clustering depends on the trimming parameter α. We are planning to compare our work with that of (Punzo and McNicholas, 2013), which estimates the trimming parameter.

Regarding online MPPCA, it could be interesting to consider the re-computation of the posterior probabilities for all (past and present) observations in the E-step based on the low-dimensional projection that we keep for each data sample.

In addition, a deeper analysis of initialization strategies should be considered for the algorithm. In fact, initialization on some set of initial data implicitly assumes that all classes are represented therein. However, this may not always be the case, since datastreams are often issued from real-world procedures where classes are not all equally represented in the initial part of the stream. In this case, preliminary experiments show that online MPPCA tends to grow arbitrarily a single class, while the rest of the classes contain only a few data samples from the initial part of the stream.

A possible solution to this problem would be to adopt a dynamic model selection procedure, where classes could be added or removed. One way to do this, as we saw in

the anomaly detection survey in chapter 3, would be to use split and merge operations on the classes following some criteria. Nevertheless, these criteria should be easy to evaluate and not need all of the past data in order to operate.

Another extension of our work could be to make an online MPPCA which will be robust to anomalies. Trimming cannot be applied in a straightforward manner in the online context, since, by default, this technique needs all of the data to operate efficiently. There is a literature on robust PCA analysis (Chen et al., 1996; Wan and Karhunen, 1996; Croux et al., 2013; Skocaj et al., 2002; Hubert and Engelen, 2004; Croux et al., 2013), which could be adapted appropriately in order to develop an online robust MPPCA.

Finally, concerning the application we have given on aircraft engine data, an extension would be make it dynamic, as we have already discussed in the relative section. A naive solution would be to recalibrate the components of the system with each novel data sample, but it would be very time-consuming. Instead, one can try to make each component to operate in a dynamic manner. For the classification component, one could use online MPPCA.

Anomaly identification can also be further investigated through the use of supervised or unsupervised classification methods based on engine trajectories' ending points on the map. Data would then be composed of the indexes of the cells in which each trajectory ended. Powerful supervised classification techniques like boosting or even simpler clustering algorithms could then be applied to classify anomalies following their type.

Publications

Journals

Bellas, A., Bouveyron, C., Cottrell, M., and Lacaille, J. (2013). Model-based clustering of high-dimensional data streams with online mixture of probabilistic PCA. Advances in Data Analysis and Classification, 7:281-300.

International refereed conferences

Bellas, A., Bouveyron, C., Cottrell, M., and Lacaille, J. Anomaly detection based on confidence intervals using SOM with an application to Health Monitoring, submitted at the Workshop on Self-Organizing Maps (WSOM) 2014.

Bellas, A., Bouveyron, C., Cottrell, M., and Lacaille, J. (2012). Robust clustering of high-dimensional data. In Proceedings of the 20th European Symposium on Artificial Neural Networks.

Bellas, A., Bouveyron, C., Cottrell, M. (2012). Robust clustering of high-dimensional data. International Conference on Operation Research, Havana, Cuba.

French-speaking refereed conferences

Bellas, A., Bouveyron, C., Cottrell, M., and Lacaille, J. (2012). Clustering online de données en grande dimension par modélisation dans des sous-espaces. In Proceedings of the 19èmes Rencontres de la Société Francophone de Classification, Marseille, France.

List of Figures

List of Tables

Bibliography

Achlioptas, D. (2001). Database-friendly random projections. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 274–281. ACM.

Achlioptas, D. (2003). Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of computer and System Sciences*, 66(4):671–687.

Aggarwal, C. C., Han, J., Wang, J., and Yu, P. S. (2004). A framework for projected clustering of high dimensional data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 852–863. VLDB Endowment.

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In *Second international symposium on information theory*, pages 267–281. Akademinai Kiado.

Akaike, H. (1981). Likelihood of a model and information criteria. *Journal of econometrics*, 16(1):3–14.

Arandjelovic, O. and Cipolla, R. (2006). Incremental learning of temporally-coherent gaussian mixture models. *SME Technical Papers*, 2006.

Atkinson, A., Riani, M., and Cerioli, A. (2006). Random start forward searches with envelopes for detecting clusters in multivariate data. In *Data analysis, classification and the forward search*, pages 163–171. Springer.

Babcock, B., Datar, M., Motwani, R., and O'Callaghan, L. (2003). Maintaining variance and k-medians over data stream windows. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 234–243. ACM.

Baek, J., McLachlan, G., and Flack, L. (2010). Mixtures of factor analyzers with common factor loadings: Applications to the clustering and visualization of high-dimensional data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(7):1298–1309.

Banfield, J. D. and Raftery, A. E. (1993). Model-based gaussian and non-gaussian clustering. *Biometrics*, pages 803–821.

Bartholomew, D., Knott, M., and Moustaki, I. (2011). *Latent variable models and factor analysis: a unified approach*, volume 899. Wiley.

Basilevsky, A. (2009). *Statistical factor analysis and related methods: theory and applications*, volume 418. Wiley-Interscience.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.

Bellman, R. (1961). *Adaptive control processes: a guided tour*, volume 4. Princeton university press Princeton.

Biernacki, C., Celeux, G., and Govaert, G. (2000). Assessing a mixture model for clustering with the integrated completed likelihood. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(7):719–725.

Bohm, C., Railing, K., Kriegel, H.-P., and Kroger, P. (2004). Density connected clustering with local subspace preferences. In *Proceedings of the Fourth IEEE International Conference on Data Mining*, pages 27–34. IEEE.

Bouveyron, C. (2006). *Modélisation et classification des données de grande dimension: application à l'analyse d'images.* PhD thesis, Université Joseph-Fourier-Grenoble I.

Bouveyron, C. and Brunet, C. (2012). Simultaneous model-based clustering and visualization in the Fisher discriminative subspace. *Statistics and Computing*, 22(1):301–324.

Bouveyron, C., Girard, S., and Schmid, C. (2007a). High-dimensional data clustering. *Computational Statistics & Data Analysis*, 52(1):502–519.

Bouveyron, C., Girard, S., and Schmid, C. (2007b). High-dimensional discriminant analysis. *Communications in Statistics—Theory and Methods*, 36(14):2607–2623.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Breunig, M., Kriegel, H., Ng, R., and Sander, J. (1999). Optics-of: Identifying local outliers. *Principles of Data Mining and Knowledge Discovery*, pages 262–270.

Breunig, M., Kriegel, H., Ng, R., Sander, J., et al. (2000). Lof: identifying density-based local outliers. *Sigmod Record*, 29(2):93–104.

Byers, S. and Raftery, A. E. (1998). Nearest-neighbor clutter removal for estimating features in spatial point processes. *Journal of the American Statistical Association*, 93(442):577–584.

Cappé, O. and Moulines, E. (2009). On-line expectation–maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):593–613.

Cattell, R. (1966). The scree test for the number of factors. *Multivariate Behav. Res.*, 1(2):245–276.

Celeux, G., Chauveau, D., and Diebolt, J. (1996). Stochastic versions of the EM algorithm: an experimental study in the mixture case. *Journal of Statistical Computation and Simulation*, 55(4):287–314.

Celeux, G. and Diebolt, J. (1985). The SEM algorithm: a probabilistic teacher algorithm derived from the EM algorithm for the mixture problem. *Computational statistics quarterly*, 2(1):73–82.

Celeux, G., Diebolt, J., et al. (1988). A random imputation principle: the stochastic EM algorithm.

Celeux, G. and Govaert, G. (1992). A classification EM algorithm for clustering and two stochastic versions. *Computational statistics & Data analysis*, 14(3):315–332.

Chan, P., Mahoney, M., and Arshad, M. (2003). A machine learning approach to anomaly detection. *2003*.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly Detection : A Survey. *Computing*, (September):1–72.

Chen, J., Bandoni, J. A., and Romagnoli, J. A. (1996). Robust pca and normal region in multivariate statistical process monitoring. *AIChE journal*, 42(12):3563–3566.

Côme, E., Cottrell, M., Verleysen, M., and Lacaille, J. (2010a). Aircraft engine health monitoring using self-organizing maps. In *Advances in Data Mining. Applications and Theoretical Aspects*, pages 405–417. Springer.

Côme, E., Cottrell, M., Verleysen, M., and Lacaille, J. (2011). Aircraft engine fleet monitoring using self-organizing maps and edit distance. In *Advances in Self-Organizing Maps*, pages 298–307. Springer.

Côme, E., Cottrell, M., Verleysen, M., Lacaille, J., et al. (2010b). Self organizing star (sos) for health monitoring. In *Proceedings of the European conference on artificial neural networks*, pages 99–104.

Cottrell, M., Fort, J.-C., and Pagès, G. (1998). Theoretical aspects of the som algorithm. *Neurocomputing*, 21(1):119–138.

Cottrell, M. and Letrémy, P. (2005). How to use the kohonen algorithm to simultaneously analyze individuals and modalities in a survey. *Neurocomputing*, 63:193–207.

Croux, C., Filzmoser, P., and Fritz, H. (2013). Robust sparse principal component analysis. *Technometrics*, 55(2):202–214.

Cuesta-Albertos, J., Gordaliza, A., and Matrán, C. (1997). Trimmed k-means: An attempt to robustify quantizers. *The Annals of Statistics*, 25(2):553–576.

Dang, X., Lee, V., Ng, W., Ciptadi, A., and Ong, K. (2009). An EM-based algorithm for clustering data streams in sliding windows. In *Database Systems for Advanced Applications*, pages 230–235. Springer.

Declercq, A. and Piater, J. (2008). Online learning of gaussian mixture models-a two-level approach. In *Intl. l Conf. Comp. Vis., Imaging and Comp. Graph. Theory and Applications*, pages 605–611. Citeseer.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38.

Ding, Q. and Kolaczyk, E. D. (2011). A compressed pca subspace method for anomaly detection in high-dimensional data. *ArXiv e-prints*.

Domingos, P., Hulten, G., Edu, P. C. W., and Edu, C. H. G. W. (2001). A general method for scaling up machine learning algorithms and its application to clustering. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 106–113. Morgan Kaufmann.

Emamian, V., Kaveh, M., and Tewfik, A. H. (2000). Robust clustering of acoustic emission signals using the kohonen network. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, volume 6, pages 3891–3894. IEEE.

Engel, P. and Heinen, M. (2011). Incremental learning of multivariate gaussian mixture models. *Advances in Artificial Intelligence–SBIA 2010*, pages 82–91.

Ertöz, L., Steinbach, M., and Kumar, V. (2003). Finding topics in collections of documents: A shared nearest neighbor approach. *Clustering and Information Retrieval*, pages 83–104.

Eskin, E. (2000). Anomaly detection over noisy data using learned probability distributions. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 255–262.

Eskin, E., Arnold, A., Prerau, M., Portnoy, L., and Stolfo, S. (2002). A geometric framework for unsupervised anomaly detection. In *Applications of data mining in computer security*, pages 77–101. Springer.

Ester, M., Kriegel, H., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data mining*, volume 1996, pages 226–231. Portland: AAAI Press.

Fisher, R. and Tippett, L. (1928). Limiting forms of the frequency distribution of the largest or smallest member of a sample. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 24, pages 180–190. Cambridge Univ Press.

Fowler, J. E. and Du, Q. (2012). Anomaly detection and reconstruction from random projections. *Image Processing, IEEE Transactions on*, 21(1):184–195.

Fraley, C. and Raftery, A. (2002). Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97(458):611–631.

Fritz, H., García-Escudero, L. A., and Mayo-Iscar, A. (2012a). A fast algorithm for robust constrained clustering. *Computational Statistics & Data Analysis*.

Fritz, H., Garcıa-Escudero, L. A., and Mayo-Iscar, A. (2012b). tclust: An r package for a trimming approach to cluster analysis. *Journal of Statistical Software*, 47(12):1–26.

Gaber, M., Zaslavsky, A., and Krishnaswamy, S. (2005). Mining data streams: a review. *ACM Sigmod Record*, 34(2):18–26.

Gallegos, M. T. (2002). Maximum likelihood clustering with outliers. In *Classification, Clustering, and Data Analysis*, pages 247–255. Springer.

Gallegos, M. T. and Ritter, G. (2005). A robust method for cluster analysis. *The Annals of Statistics*, 33(1):347–380.

Gallegos, M. T. and Ritter, G. (2009). Trimming algorithms for clustering contaminated grouped data and their robustness. *Advances in Data Analysis and Classification*, 3(2):135–167.

García-Escudero, L. A., Gordaliza, A., and Matrán, C. (1999). A central limit theorem for multivariate generalized trimmed k-means. *Annals of statistics*, pages 1061–1079.

García-Escudero, L. A., Gordaliza, A., Matrán, C., and Mayo-Iscar, A. (2008). A general trimming approach to robust cluster analysis. *The Annals of Statistics*, pages 1324–1345.

García-Escudero, L. A., Gordaliza, A., Matrán, C., and Mayo-Iscar, A. (2010). A review of robust clustering methods. *Advances in Data Analysis and Classification*, 4(2-3):89–109.

Ghahramani, Z., Hinton, G., et al. (1996). The EM algorithm for mixtures of factor analyzers. Technical report, Technical Report CRG-TR-96-1, University of Toronto.

Grubbs, F. E. (1969). Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21.

Guha, S., Mishra, N., Motwani, R., and O'Callaghan, L. (2000a). Clustering data streams. In *Foundations of computer science, 2000. proceedings. 41st annual symposium on*, pages 359–366. IEEE.

Guha, S., Rastogi, R., and Shim, K. (2000b). Rock: A robust clustering algorithm for categorical attributes. *Information Systems*, 25(5):345–366.

Guttormsson, S., Marks, R., El-Sharkawi, M., and Kerszenbaum, I. (1999). Elliptical novelty grouping for on-line short-turn detection of excited running rotors. *Energy Conversion, IEEE Transactions on*, 14(1):16–22.

Hall, P., Hicks, Y., and Robinson, T. (2005). A method to add gaussian mixture models.

Hall, P., Marshall, D., and Martin, R. (1998). Incremental eigenanalysis for classification. In *British Machine Vision Conference*, volume 1, pages 286–295. Citeseer.

Harris, T. (1993). A kohonen som based, machine health monitoring system which enables diagnosis of faults not seen in the training set. In *Neural Networks, 1993. IJCNN'93-Nagoya. Proceedings of 1993 International Joint Conference on*, volume 1, pages 947–950. IEEE.

Hautamaki, V., Karkkainen, I., and Franti, P. (2004). Outlier detection using k-nearest neighbour graph. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, pages 430–433. IEEE.

He, Z., Xu, X., and Deng, S. (2003). Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10):1641–1650.

Ho, L. L., Macey, C. J., and Hiller, R. (1999). A distributed and reliable platform for adaptive anomaly detection in ip networks. In *Active Technologies for Network and Service Management*, pages 33–46. Springer.

Hubert, M. and Engelen, S. (2004). Robust pca and classification in biosciences. *Bioinformatics*, 20(11):1728–1736.

Johnson, W. B. and Lindenstrauss, J. (1984). Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1.

Kohonen, T. (2001). *Self-organizing maps*, volume 30. Springer.

Kriegel, H.-P., Kröger, P., Ntoutsi, I., and Zimek, A. (2011). Density based subspace clustering over dynamic data. In *Scientific and Statistical Database Management*, pages 387–404. Springer.

Labib, K. and Vemuri, R. (2002). Nsom: A real-time network-based intrusion detection system using self-organizing maps.

Lacaille, J. and Côme, E. (2011). Visual mining and statistics for a turbofan engine fleet. In *Aerospace Conference, 2011 IEEE*, pages 1–8. IEEE.

Lacaille, J., Côme, E., et al. (2011). Sudden change detection in turbofan engine behavior. In *Proceedings of the The Eighth International Conference on Condition Monitoring and Machinery Failure Prevention Technologies*, pages 542–548.

Lacaille, J. and Gerez, V. (2011). Online abnormality diagnosis for real-time implementation on turbofan engines and test cells. pages 579–587.

Lacaille, J., Gerez, V., and Zouari, R. (2010). An adaptive anomaly detector used in turbofan test cells. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society*.

Lauer, M. (2001). A mixture approach to novelty detection using training data with outliers. *Machine Learning: ECML 2001*, pages 300–311.

Laurikkala, J., Juhola, M., Kentala, E., Lavrac, N., Miksch, S., and Kavsek, B. (2000). Informal identification of outliers in medical data. In *Proceedings of the 5th International Workshop on Intelligent Data Analysis in Medicine and Pharmacology*, pages 20–24.

Lindsay, B. G. (1995). Mixture models: theory, geometry and applications. In *NSF-CBMS Regional Conference Series in Probability and Statistics*, volume 5, pages i–163. JSTOR.

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, page 14.

Mangasarian, O. L., Street, W. N., and Wolberg, W. H. (1995). Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577.

Manson, G., Pierce, S., Worden, K., Monnier, T., Guy, P., and Atherton, K. (2000). Long-term stability of normal condition data for novelty detection. In *Proceedings of SPIE*, volume 3985, page 323.

Markou, M. (2003a). Novelty detection: a review—part 1: statistical approaches. *Signal Processing*, 83(12):2481–2497.

Markou, M. (2003b). Novelty detection: a review—part 2: neural network based approaches. *Signal Processing*, 83(12):2499–2521.

Marsland, S., Nehmzow, U., and Shapiro, J. (2000). A real-time novelty detector for a mobile robot.

McLachlan, G. and Krishnan, T. (1997). *The EM Algorithm and Extensions*. Wiley, New York.

McLachlan, G. and Peel, D. (2000). *Finite mixture models*, volume 299. Wiley-Interscience.

McLachlan, G., Peel, D., and Bean, R. (2003). Modelling high-dimensional data by mixtures of factor analyzers. *Computational Statistics & Data Analysis*, 41(3):379–388.

McNicholas, P. and Murphy, B. (2008). Parsimonious Gaussian mixture models. *Statistics and Computing*, 18(3):285–296.

Meng, X.-L. and Rubin, D. B. (1993). Maximum likelihood estimation via the ecm algorithm: A general framework. *Biometrika*, 80(2):267–278.

Müller, C. H. and Neykov, N. (2003). Breakdown points of trimmed likelihood estimators and related estimators in generalized linear models. *Journal of Statistical Planning and inference*, 116(2):503–519.

Neal, R. and Hinton, G. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. *Learning in graphical models*, 89:355–368.

Neykov, N., Filzmoser, P., Dimova, R., and Neytchev, P. (2007). Robust fitting of mixtures using the trimmed likelihood estimator. *Computational Statistics & Data Analysis*, 52(1):299–308.

O'callaghan, L., Mishra, N., Meyerson, A., Guha, S., and Motwani, R. (2002). Streaming-data algorithms for high-quality clustering. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 685–694. IEEE.

Papadimitriou, S., Kitagawa, H., Gibbons, P. B., and Faloutsos, C. (2003). Loci: Fast outlier detection using the local correlation integral. In *Data Engineering, 2003. Proceedings. 19th International Conference on*, pages 315–326. IEEE.

Parzen, E. (1962). On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076.

Pokrajac, D., Lazarevic, A., and Latecki, L. J. (2007). Incremental local outlier detection for data streams. In *Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on*, pages 504–515. IEEE.

Punzo, A. and McNicholas, P. D. (2013). Outlier detection via parsimonious mixtures of contaminated gaussian distributions. *arXiv preprint arXiv:1305.4669*.

Ricordeau, J. and Lacaille, J. (2010). Application of random forests to engine health monitoring. In *Proceedings of the 27th International Congress of the Aeronautical Sciences*.

Roberts, S. (1999). Novelty detection using extreme value statistics. In *Vision, Image and Signal Processing, IEEE Proceedings-*, volume 146, pages 124–129. IET.

Roberts, S. (2000). Extreme value statistics for novelty detection in biomedical data processing. In *Science, Measurement and Technology, IEE Proceedings-*, volume 147, pages 363–367. IET.

Roberts, S. and Tarassenko, L. (1994). A probabilistic resource allocating network for novelty detection. *Neural Computation*, 6(2):270–284.

Rousseeuw, P. J. (1985). Multivariate estimation with high breakdown point. *Mathematical Statistics and Applications*, B:283–297.

Rousseeuw, P. J. and Leroy, A. M. (2005). *Robust regression and outlier detection*, volume 589. Wiley. com.

Samé, A., Ambroise, C., and Govaert, G. (2007). An online classification EM algorithm based on the mixture model. *Statistics and Computing*, 17(3):209–218.

Sato, M.-A. and Ishii, S. (2000). On-line EM algorithm for the normalized gaussian network. *Neural Computation*, 12(2):407–432.

Schwarz, G. (1978). Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464.

Scott, D. W. and Thompson, J. R. (1983). Probability density estimation in higher dimensions. In *Computer Science and Statistics: Proceedings of the Fifteenth Symposium on the Interface*, volume 528, pages 173–179. North-Holland, Amsterdam.

Silverman, B. W. (1986). *Density estimation for statistics and data analysis*, volume 26. CRC press.

Skocaj, D., Bischof, H., and Leonardis, A. (2002). A robust pca algorithm for building representations from panoramic images. In *Proceedings of the 7th European Conference on Computer Vision*, volume 4, pages 761–775. Springer.

Song, M. and Wang, H. (2005). Highly efficient incremental estimation of gaussian mixture models for online data stream clustering. In *Proc. SPIE*, volume 5803, pages 174–183.

Song, X., Wu, M., Jermaine, C., and Ranka, S. (2007). Conditional anomaly detection. *IEEE Transactions on knowledge and Data Engineering*, 19(5):631–645.

Tang, J., Chen, Z., Fu, A., and Cheung, D. (2002). Enhancing effectiveness of outlier detections for low density patterns. *Advances in Knowledge Discovery and Data Mining*, pages 535–548.

Tarassenko, L., Hayton, P., Cerneaz, N., and Brady, M. (1995). Novelty detection for the identification of masses in mammograms. In *Proceedings of the Fourth International Conference on Artificial Neural Networks, 1995.*, pages 442–447. IET.

Tax, D. and Duin, R. (1998). Outlier detection using classifier instability. *Advances in Pattern Recognition*, pages 593–601.

Tax, D. M. and Duin, R. P. (2000). Data description in subspaces. In *Proceedings of the 15th International Conference on Pattern Recognition*, volume 2, pages 672–675. IEEE.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

Tipping, M. and Bishop, C. (1999a). Mixtures of probabilistic principal component analyzers. *Neural computation*, 11(2):443–482.

Tipping, M. and Bishop, C. (1999b). Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622.

Titterington, D. (1984). Recursive parameter estimation using incomplete data. *Journal of the Royal Statistical Society. Series B (Methodological)*, 46(2):257–267.

Ueda, N., Nakano, R., Ghahramani, Z., and Hinton, G. (2000). Smem algorithm for mixture models. *Neural computation*, 12(9):2109–2128.

Verbeek, J., Vlassis, N., and Kröse, B. (2003). Efficient greedy learning of gaussian mixture models. *Neural computation*, 15(2):469–485.

Vlassis, N. and Likas, A. (2002). A greedy EM algorithm for gaussian mixture learning. *Neural processing letters*, 15(1):77–87.

Wan, L. and Karhunen, J. (1996). A unified neural bigradient algorithm for robust pca and mca. *International Journal of Neural Systems*, 7(01):53–67.

Wu, C. (1983). On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11(1):95–103.

Yang, J. (2003). Dynamic clustering of evolving streams with a single pass. In *Proceedings of the 19th International Conference on Data Engineering*, pages 695–697. IEEE.

Yang, Z. and Zwolinski, M. (2001). Mutual information theory for adaptive mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(4):396–403.

Ye, N. and Chen, Q. (2001). An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. *Quality and Reliability Engineering International*, 17(2):105–112.

Yeung, D. and Chow, C. (2002). Parzen-window network intrusion detectors. *Pattern Recognition*, 4:385–388.

Ypma, A. and Duin, R. P. (1997). Novelty detection using self-organizing maps. *Progress in connectionist-based information systems*, 2:1322–1325.