Faculté des Sciences Appliquées
Département d'Electricité
Laboratoire de Microélectronique - DICE
Machine Learning Group

Aspects of Probabilistic Modelling for Data Analysis

Nicolas Delannay

Thèse présentée en vue de l'obtention
du grade de Docteur en Sciences Appliquées

Membres du Jury:

Prof. Michel Verleysen (Université catholique de Louvain), Promoteur
Prof. Pierre-Antoine Absil (Université catholique de Louvain)
Dr. Gavin C. Cawley (University of East Anglia, UK)
Dr. Fabrice Rossi (INRIA Paris-Rocquencourt, France)
Prof. Marco Saerens (Université catholique de Louvain)
Prof. Luc Vandendorpe (Université catholique de Louvain), Président

Octobre 2007

Acknowledgements

It is a long way to get to a Ph.D. degree and I would not have succeeded in this project without the support of several people which I would like to thank here.

First, I would like to thank Professor Michel Verleysen, my supervisor, for his precious scientific guidance and the interest he manifested for my research all along the work. Thank you also Michel for setting this very friendly atmosphere between you and all of your researchers. I would like also to thank all the members of the committee. I am very grateful to Professors Fabrice Rossi and Marco Saerens for following my research, for the exciting discussions we had and the numerous advices given to me. I am also very grateful to Professors Gavin Cawley and Pierre-Antoine Absil for the time they have taken to read the thesis and for their helpful remarks. Further thanks to Gavin Cawley for correcting wording of the text.

I really appreciated working within the Machine Learning Group at UCL and I would like to express collectively my gratitude to all past and current members of the group. A particular thank goes to Cédric Archambeau with whom I could openly discuss all aspects of my work. Collaborating with you Cédric was a pleasure. Thanks also to all of you who shared my everyday life in the a.134 office: Nicolas Donckers, Simon Dablemont, Geoffroy Simon, Gaël de Lannoy, Catherine Krier and Arnaud de Decker. I was very lucky to have you, Frédéric Vrins, by my side all these years and I hope our friendship will last. Thanks to the people of the Microelectronics laboratory (DICE), administration team and researchers. You have been part of this great working environment.

My gratitude goes also to the members of the Empirical Inference Department of the Max Planck Institute for Biological Cybernetics in Tübingen during my stay at the Institute, in spring 2006. Thank you in particular to Dilan Görür, Olivier Chapelle and Carl Edward Rasmussen. I am also grateful to the CSML group in the Computer Department at University College London for the welcome I received there in July 2007.

My friends and family circle, I could count on your invaluable presence and support. Thank you for that. Dear Mum and Dad, this thesis is dedicated to you.

Abstract

Computer technologies have revolutionised the processing of information and the search for knowledge. With the ever increasing computational power, it is becoming possible to tackle new data analysis applications as diverse as mining the Internet resources, analysing drugs effects on the organism or assisting wardens with autonomous video detection techniques.

Fundamentally, the principle of any data analysis task is to fit a model which encodes well the dependencies (or patterns) present in the data. However, the difficulty is precisely to define such proper model when data are noisy, dependencies are highly stochastic and there is no simple physical rule to represent them.

The aim of this work is to discuss the principles, the advantages and weaknesses of the probabilistic modelling framework for data analysis. The main idea of the framework is to model dispersion of data as well as uncertainty about the model itself by probability distributions. Three data analysis tasks are presented and for each of them the discussion is based on experimental results from real datasets.

The first task considers the problem of linear subspaces identification. We show how one can replace a Gaussian noise model by a Student-t noise to make the identification more robust to atypical samples and still keep the learning procedure simple. The second task is about regression applied more specifically to near-infrared spectroscopy datasets. We show how spectra should be preprocessed before entering the regression model. We then analyse the validity of the Bayesian model selection principle for this application (and in particular within the Gaussian Process formulation) and compare this principle to the resampling selection scheme. The final task considered is Collaborative Filtering which is related to applications such as recommendation for e-commerce and text mining. This task is illustrative of the way how intuitive considerations can guide the design of the model and the choice of the probability distributions appearing in it. We compare the intuitive approach with a simpler matrix factorisation approach.

Contents

Introduction

Computer technologies have revolutionised the processing of information and the search for knowledge. Attempts to tackle problems that were unthinkable in the past are now becoming possible with the increasing computational power available. In parallel, it has never been so easy to collect huge amounts of information from very diverse applications (resources on the Internet, bio-informatics, video surveillance, population personal data, spatial observation...). These sources of information are generating large databases that are waiting to be exploited.

In fact, the gathering of data is at the heart of the scientific approach. Indeed, the first step of the scientific approach is always a careful observation of the system being analysed. Progressively, understanding can build upon observation, which in turn can be used to formulate an abstract representation of the system by means of a mathematical model.

This outlines the importance of designing efficient data analysis tools since data are the raw material of knowledge. Data analysis looks for structures (also called patterns) in the data. These patterns originate from the interactions (or rules) governing the system. Identifying these patterns is thus an integral part of the modelling procedure. Modelling is greatly simplified when one has already some knowledge of the rules and structures that should be present in the data. For example, the exploitation of meteorological measurements is made possible by the vast experience accumulated in the field and the complex physical models developped. Of course, it is still possible to improve these models. However, we will instead focus in this thesis on the analysis of data for which there is no such prior knowledge of the rules in the system, either because it is too complex to characterise these rules or because the system is significantly stochastic. This is typically the case for the examples given above about Internet resources or bio-informatics. The goal of data analysis is then to construct generic models which are capable of identifying patterns in such datasets. The construction of models is referred as the learning procedure.

Unfortunately, there is a fundamental limitation to the information that can be learned from data. This limitation has its roots in the fact that the observed data comprise a finite number of samples. Consequently the data only give partial information about the rules governing the system. Learning theory states that it is impossible to estimate the missing information with

perfect certainty using only the observed data. Moreover, randomness is also present in most datasets due to sources of noise or simply because the system is not fully deterministic. Information associated with randomness is not, by definition, predictable. One of the main difficulties of learning from data is to prevent the model from trying to identify deterministic patterns in the noise and in the non deterministic parts of the dataset. This is known as *over-fitting*.

There are essentially three aspects to research in the field of data analysis. The first aspect is known as *learning theory*. The goal of learning theory is to work out methods to estimate the adequacy of a model for the data coming from a system as well as the confidence that one can have in this estimate of adequacy. This theoretical work forms the core of statistics. The second aspect concerns the implementation of learning procedures and optimisation techniques for tackling practical data analysis tasks. Indeed, it is not sufficient to have good estimators from learning theory if their computational cost is excessive. Excessive computational costs might be due to the size of the dataset, but working with highly flexible models can also lead to excessive costs. Designing practical learning procedures is often viewed as the subject of the Machine Learning field. Finally, the third aspect of the work is the one closest to the field of the application. At this level, the motivation is to exploit information in the data and the goal is thus to make data analysis useful for a particular task. The usual framework to exploit data analysis is by asking the fitted model to return predictions for the missing parts of new observations. Classification and regression tasks belong to this scheme. The usefulness of the analysis is simple to assess for prediction tasks because it can be measured by quantitative criteria, e.g. a standard measure for classification is the proportion of correct classifications. On the other hand, when data analysis enters the preliminary stages of a study, it is rarely possible to express the goal by a quantitative measure. In this context, data analysis should rather be viewed as a tool for obtaining understanding about the system. But, understanding is fundamentally subjective and no quantitative measure would be completely adequate for assessing this kind of usefulness of the learning procedure.

Naturally, these different aspects of data analysis research are intimately linked and any learning procedure will have to concern itself at the same time with theoretical foundation, practical implementation and external evaluation of its usefulness.

Theme of the thesis The main focus of this thesis is not on one of these particular aspects but rather on a general framework for designing and training models, which is called the *probabilistic modelling* framework. As we will see in the introductory chapter, the main feature of this framework is to model randomness by probability distributions. Throughout the text, we try to show how the framework can be applied to different learning tasks, what advantages it can have over other approaches but also what are its shortcomings. The discussion is based on empirical studies and comparisons for a good deal. We try to identify the origin of the failures when they occur and propose solutions

to correct them.

The discussion will cover the three aspects of the data analysis task from the perspective of probabilistic modelling. We will see that the framework is complete from the theoretical point of view as it defines its own measure of adequacy and estimate of confidence. It will be interesting to verify that these measures are valid such that models can be learned on them. Concerning the implementation, probabilistic modelling mainly relies on the computation of posterior distributions representing the uncertainty about the model, and the computation of integrals, also known as marginalisation. Respecting some simple rules for the combination of distributions appearing in the model, it is possible to work with exact evaluation of the posterior and marginal. For more complex models, this is no longer possible and approximation techniques are required which can significantly increase the computational cost. In order to make the approach applicable to the datasets considered, we prefer to keep the modelling sufficiently simple and to be able to derive exact computations or simple approximations. Nevertheless, we will show that probabilistic modelling remains highly versatile as it is not tied to a particular learning task. This will be illustrated by the three tasks tackled in the thesis, namely robust exploratory analysis, regression and Collaborative Filtering. In addition, the framework proposes a natural way to encode prior knowledge or intuition about the system by means of the prior distributions and a hierarchical formulation. This can be exploited to improve the learning procedure but also to design models from which interpretation is "readily" available, at least if the intuition corresponds well to the system.

Contributions From a broad standpoint, the main contribution of the work is to expose the principles of the probabilistic modelling framework and to discuss its advantages and weaknesses by means of three data analysis tasks. In this sense, the thesis should help practitioners and researchers in the field to decide when and how the framework should be applied.

Moreover, three original models are described and put under test:

- The mixture of robust Probabilistic PCAs presented in the second chapter, for which an exact Expectation-Maximisation algorithm is derived.
- The smooth Automatic Relevance Determination scheme used within the Gaussian Process model for the regression task on NIR spectra in Chapter 3.
- The interlaced Generalized Linear Model (GLM) approach to tackle Collaborative Filtering which is described in Chapter 4.

An important part of the work lies in the experimental comparisons. From this angle the contributions are:

- The comparison of non robust against robust linear subspace mixture models.

- The comparison of regression methods for the NIR spectroscopy application showing among other results that the marginal likelihood framework within the Gaussian Process formulation is generally valid.

- The comparison of prediction performances of the mixture models for Collaborative Filtering against the simpler interlaced GLM model.

Outline The thesis is organised in four chapters. In Chapter 1, we introduce learning theory, formalise the different learning tasks and present the fundamental concept of *model capacity*. Also in this chapter, the general principles of probabilistic modelling are exposed and we derive two algorithms used in the following chapters.

Chapter 2 presents probabilistic models capable of identifying clusters in the data as well as local correlations. These models are appropriate for an exploratory analysis at the preliminary stages of a data analysis study. We show how the modularity of the probabilistic modelling framework can be used to improve an existing model such that its construction is less sensitive to erroneous measures or atypical samples which are frequent in real datasets.

In Chapter 3, the analysis concentrates on regression applied to data from near-infrared spectroscopy. Near-infrared spectroscopy produces particularly smooth spectra. We show how these spectra can be handled properly in the models and how one can make use of additional functional constraints on the parameters. Different regression methods are then compared. One of the comparisons tries to determine whether models relying on the Bayesian model selection principle (e.g. the Gaussian Process) are valid and how their performances compare with the non-probabilistic resampling scheme.

Finally, in Chapter 4 we look at Collaborative Filtering, a task arising in applications such as e-commerce or document mining. In the first part, two probabilistic mixture models designed with some intuitive arguments are presented. We show how the posterior analysis of these models can extract some interpretation from the patterns. We then present a simplified modelling procedure, called the interlaced GLM. We then compare the different models on the prediction of unobserved ratings.

Notation

We have tried to use consistent notation throughout the text and to have systematic rules to make the expressions more concise. Scalars are represented by lowercase italic letters like x. Vectors are denoted by lowercase bold letters, for example the symbol $\mathbf{x}$ could represent a vector observation. We always assume that vectors are column vectors unless specified in the definition. The dimensionalities and numbers of elements in a set are noted with uppercase italic letters. For example D might be the dimensionality of the vector observations such that $\mathbf{x} = [x_1, \ldots, x_D]^\top$ (or more simply $\mathbf{x} \in \mathbb{R}^D$) and N might be the number of observations in a set. The set of observations will generally be noted $\mathcal{D} \equiv \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. We have tried to use everywhere the convention of associating to a particular uppercase letter the same lowercase letter for indexes running over the dimensionality or the number of elements (i.e. d runs over D, n runs over N...). To simplify some expressions where summations or products appear, the set of index over these mathematical operators will be left implicit. For example, $\sum_n$ should be understood as $\sum_{n=1}^N$. Of course, if the convention does not apply, we will make explicit the set covered by the indexes. The same rule is used to make more concise the notation of sets; e.g. $\{\mathbf{x}_n\}$ should really be read as $\{\mathbf{x}_n\}_{n=1}^N$.

Matrices are noted with bold uppercase letters. For example, $\mathbf{X}$ could represent a $D \times N$ matrix obtained from the concatenation of all vector observations in $\mathcal{D}$ defined above, i.e. $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top$.

There are a few other recurrent notations. $P(\cdot)$ represents a probability distribution. We make no distinction in the notations between probability distributions over discrete and continuous random variables. The expectation operator is noted $\mathbb{E}_x\{\cdot\}$. Here, the subscript gives an indication of the distribution on which the expectation is computed. For this example, it would probably represent the true distribution $P^*(x)$ of a scalar random variable x. But this should be made clear in the text if there is an ambiguity (e.g. instead of the true distribution, it could also be the expectation over the prior distribution in the Bayesian framework).

To note the diagonal identity matrix of dimensionality D, we write $\mathbf{I}_D$. The notation $\mathrm{diag}\{[x_1, \ldots, x_D]\}$ stands for a diagonal matrix with the vector $\mathbf{x}$ on the diagonal entries.

Chapter 1

Introduction to statistical learning theory and probabilistic modelling

The goal of this chapter is to introduce the general concepts involved in learning a model from data and to give a general background to the problems tackled in the subsequent chapters. We will start by setting up the main mathematical notation used to formulate the learning task and present the supervised and unsupervised tasks that will be considered afterwards. The second section concentrates on the main challenge of learning: finding a good compromise between matching the data with a model and limiting the variability of the estimation of this model. This is called the model selection stage of learning and is related to the famous bias-variance trade-off of classical statistics. In the third section, we introduce the principles of probabilistic modelling and the Bayesian framework. As probabilistic modelling will be used recurrently through out the thesis, we try to give a feeling as to why and how this framework can be applied. The mathematical developments of this third section form the backbone of other algorithms appearing in the next chapters. Certainly, we are omitting to present some important learning tasks, such as reinforcement learning or time series prediction, because these tasks are not considered afterwards. Also, there exist many other strategies for model selection than those presented. We have made the decision to leave these other strategies out of the discussion. For a more complete overview of learning theory and probabilistic modelling, there are many good books. Standard references for statistical learning theory are [144, 142, 40]. Resampling methods are discussed in [42, 122]. We found the introductory part of the book "*kernel methods for pattern analysis*" [123] very clear and based some of the learning problem formulation on it. "*Pattern recognition and machine learning*" [17] is also very comprehensive for exposing the probabilistic modelling approach. The book [83] gives full support to the

Bayesian framework and sets links with information theory.

1.1 Preliminaries

In this section, we start by formulating the concepts of pattern recognition and generalisation. These concepts are then transformed into a mathematical formulation defining the learning task and the model configuration. We discuss shortly the adequacy of the mathematical formulation from the practitioner's point of view. Finally, the two standard learning tasks of supervised and unsupervised learning are described.

1.1.1 Pattern recognition and generalisation

We already know from the introductory chapter the importance of the data collected as they are supposed to contain some useful information one wants to learn about a system. The data at our disposal describing this useful information will be called the *observed* data. The difficulty of learning is that these observed data only give a partial and often noisy view of the system. At the heart of the learning task is the concept of *pattern recognition*. Pattern is here a generic term intended to express any type of rule or dependency structure present in the data. These patterns represent the information about the system that one is looking for. Note that the data are often noisy measurements and the system is not necessarily fully deterministic, such that the dependency expressed by patterns is rarely completely deterministic either. They are said to be statistical patterns.

The main challenge of learning is to identify *inherent* patterns, i.e. patterns that are exhibited by subsequent data collected from the system and not only by the observed data. A learning procedure that is able to identify inherent patterns *generalises* well. The generalisation property makes it possible to infer with some confidence the missing information of a partially observed state of the system. In contrast, if the learning procedure returns patterns that are present in the observed data but absent from other data collected from this system, it is said to be *over-fitting*. When over-fitting occurs, inference of missing information might be completely erroneous.

At this stage, it is not yet clear how a pattern recognition task is tackled in practice. This will become clearer as the concepts defined above are translated mathematically.

1.1.2 Mathematical formulation of learning

Let us start with the data. The observed dataset will be denoted $\mathcal{D} = \{\chi_n\}_{n=1}^{N}$. It contains N observations (or samples). For the moment, we do not need to know how these samples χ_n are represented. The theory exposed below is based on several assumptions about the data. First, it is assumed that the procedure used to collect the data can be recast as a generative process. The dataset

is supposed to have been generated from an unknown underlying distribution $P^*(\mathcal{D})$. In order to be able to perform inference on new data, this underlying distribution should be stationary in time. Another fundamental assumption is that there are no direct dependencies between the samples of this generative process; the samples are thus *independent and identically distributed* (*i.i.d.*), such that $P^*(\mathcal{D}) = \prod_n P^*(\chi_n)$. This assumption would not be adequate for many systems, in particular those generating times series.

Having collected this dataset, the goal of identifying patterns is specified by two elements. First, we need to specify a positive measure for assessing the adequacy of the patterns with the data. This measure will be called the *loss function* and it will be noted $\mathcal{L}$. Secondly, we need to specify a model which actually encodes the dependency rules in the patterns. The model will be noted by h. The adequacy of a model (and the patterns it encodes) with the observed dataset is given by

$$\mathcal{L}(\mathcal{D}, h) = \sum_n \mathcal{L}(\chi_n, h) \tag{1.1}$$

where the sum decomposition is a consequence of the *i.i.d.* assumption.

As stated above, the goal of learning is to find a model which has a good adequacy with not only observed data, but any subsequent samples. So, a model generalising well should have a low expected loss

$$\mathcal{R}(h) \equiv \mathbb{E}_\chi \left\{ \mathcal{L}(\chi, h) \right\} \ , \tag{1.2}$$

where the expectation is taken over the true underlying distribution of the data $P^*(\chi)$. This expected loss is the generalisation error of the model h. It is also called the *risk*. It is important to keep in mind that the risk depends on the loss function and the true underlying distribution.

Ideally, one would like to find the optimal model h^*, i.e. the model having the minimum risk

$$\mathcal{R}^* \equiv \mathcal{R}(h^*) = \min_h \mathcal{R}(h) \ . \tag{1.3}$$

But this optimisation cannot be carried out directly because we do not know the underlying distribution $P^*(\chi)$. Instead, it is necessary to set up a learning procedure; given a dataset $\mathcal{D}$ and a loss function $\mathcal{L}$, the learning procedure returns a model which hopefully has a risk close to the optimal risk $\mathcal{R}^*$. We will concentrate on learning procedures that are decomposed in two learning stages: the *training* stage and the *model selection* stage. These two stages are now described.

In the training stage, one specifies a *training configuration*, i.e. a model space $\mathcal{H}$ and a training criterion $\mathcal{E}$. The model space - sometimes called the hypothesis space, or the class of model - is the space from which the training procedure will select the model. The training criterion is a computable measure between the model and a set of data. Given a training configuration $(\mathcal{H}, \mathcal{E})$, the training stage amounts to selecting from $\mathcal{H}$ the model that minimises the

training criterion

$$\hat{h} = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \mathcal{E}(\mathcal{D}, h) \ . \tag{1.4}$$

One should also make sure that this optimisation can be done efficiently (at least, that it is not a NP-hard problem). Later in the thesis, we will see that training requires the inversion of a linear system for some of the methods and the derivation of a gradient descent algorithm for the others.

In practice, it is common to work with a parametric model space. Let us denote the set of parameters of a parametric model space by θ and the parameter space by Θ. So each model $h \in \mathcal{H}$ is dually expressed by its parameters $\theta \in \Theta$. Note that some learning procedures use a model space growing with the number of observed samples. These model spaces are said to be non-parametric in the sense that we cannot attribute to them a fixed size set of parameters.

The simplest training criterion one could be tempted to use is the *Empirical Risk*

$$\mathcal{R}_{\mathcal{D}}(h) \equiv \frac{1}{N} \mathcal{L}(\mathcal{D}, h) \ . \tag{1.5}$$

However, we will show in Section 1.2.1 that this criterion leads to over-fitting when the size of the model space gets too large. With large model space, it gets possible to select a model which fit the data very well, returning a small empirical risk. But the noise will be fitted at the same time. There must thus be some control on the capacity of the training configuration.

Choosing a good model space and training criterion $(\mathcal{H}, \mathcal{E})$ is the task of the *model selection* stage of the learning procedure. There exist different strategies to perform this model selection. We will present three of them in Section 1.2 and 1.3. Basically, the idea is to define a *selection criterion* $\mathcal{S}(\mathcal{H}, \mathcal{E})$ which should be well correlated with the risk of the trained model $\mathcal{R}(\hat{h})$ from Eq. (1.4). The selection looks thus for the training configuration minimising this selection criterion.

$$\left(\hat{\mathcal{H}}, \hat{\mathcal{E}}\right) = \underset{\mathcal{H}, \mathcal{E}}{\operatorname{argmin}} \mathcal{S}(\mathcal{H}, \mathcal{E}) \ . \tag{1.6}$$

However, in practice the selection criterion can be expensive to compute and the search is made over a finite set of combinations. Often, we will work within a parametric class of model spaces that are adjusted by a set of parameters $\eta_{\mathcal{H}}$ and equivalently within a class of training criteria adjusted by a set of parameters $\eta_{\mathcal{E}}$. These model selection parameters will be called the hyper-parameters since they are learned at the second stage of the learning procedure. To make the parametrisation explicit, the selection criterion could be rewritten $\mathcal{S}(\eta_{\mathcal{H}}, \eta_{\mathcal{E}})$. The parametric families of model spaces $(\mathcal{H}, \eta_{\mathcal{H}})$ and training criterion $(\mathcal{E}, \eta_{\mathcal{E}})$ on which the model selection takes place will be called the *model configuration*.

1.1.3 Interplay between theory and practice

At this stage we want to make some comments on the procedure defined above and see if it really meets the objective of the practitioner. Let us start with the

assumptions made about the data. For many applications, the practitioner has some knowledge of the system being analysed and his knowledge will dictate the way he collects the data. As such, it is imprecise to view the collecting procedure as a generative process, with independence between the events. Moreover, during the exploitation phase of the analysis, new samples arrive collected under other conditions. We cannot say that the underlying distribution of these new samples is the same as during the collecting stage. This is not so big an issue if data collected during the learning and the exploitation stages satisfy the same patterns. Simply, the risk will not be a good measure of the performance during exploitation. For many systems, the assumption of stationary process is also very simplistic as the system is slowly evolving over time. This is the case, for example, on the Collaborative Filtering data investigated in Chapter 4. We can be critical about these assumptions, but they are necessary to develop the theory. It is the role of experimental study to verify that the theory provides an adequate match with reality.

Maybe the most complicated task of the practitioner is to define clearly his objectives and what sort of patterns he is looking for. Trying several loss functions should help him get some understanding of the data in order to define his objectives. However, not all objectives can be expressed so simply as the minimisation of a scalar loss function. For example, it is often desirable to compress the information encoded in the data. One is thus looking for a trade-off between finding a good representation of the inherent patterns and using a simple model. There is no unique solution but a path of solutions following the level of compression required, or equivalently the loss of information allowed.

Related to the requirement of simplicity, the practitioner would like to be able to derive some interpretation of the model, in particular at the preliminary stage of the analysis. Interpretation is fundamentally a subjective concept. It is thus not possible to measure it in an objective quantity like the loss function. We will discuss for each application treated in the subsequent chapters the possibility of deriving some interpretation from the fitted models.

1.1.4 Supervised and unsupervised learning

There are essentially two types of learning task that will get our attention in this thesis. These tasks are referred to as *supervised* and *unsupervised learning* tasks. Let us start with supervised learning as it is easier to understand. The idea is that data samples χ are composed of an *input* and a *target* part $\chi = (\mathbf{x}, t)$. For standard problems, the target is just a scalar value. The inputs on the other hand are often described by vectors $\mathbf{x} \in \mathbb{R}^D$, but one could imagine working with more complex entities like sequences or trees. The goal of learning is to derive from the dataset the dependency of targets on inputs, such that the model is capable of returning target predictions for new inputs. Hence, the model is a prediction function $h \equiv f(\mathbf{x})$.

If the targets belong to an ordered set (i.e. for all pairs of targets (t, t') we have $t > t'$, $t < t'$ or $t = t'$), it is called a *regression* task. Depending on

the objectives, one will select a particular loss function for this regression task. The usual losses have the form $\mathcal{L}(t, y) = \mathcal{L}(|t - y|)$. In particular the *square error loss* $\mathcal{L}(t, y) = |t - y|^2$ is widely used mainly because training a model on this loss can often be done with a simple optimisation procedure. With more domain knowledge, one could also define a loss dependent on the input.

When the target does not belong to an ordered set (i.e. for each pair of target we can only say if $t = t'$ or $t \neq t'$), the problem is called a *classification* task. The different values the targets can take are the *labels*. For classification, the simplest loss function is the $0 - 1$ loss, $\mathcal{L}(t, y) = 0$ if $t = y$ and 1 otherwise. Note however that this loss is not continuous and thus difficult to optimise directly. Typically the cost of making an erroneous prediction for the different labels are not equivalent and one can take into account these different costs in the loss. Another interesting loss is the *hinge loss*. If the labels of a binary classification task are encoded with $t \in \{-1, 1\}$, the hinge loss is defined by

$$\mathcal{L}(t, y) = \begin{cases} 0 & \text{if } |y| > |t| \\ |t - y| & \text{otherwise} \end{cases} . \tag{1.7}$$

This is the training loss used to train the well-known *support vector machines* classifier.

The second type of learning problem is the unsupervised task. In contrast to supervised learning, we do not know at first what natural conditional dependency data encode. There are a few typical patterns that can be interesting to look for corresponding to different unsupervised tasks. The first task is called *clustering*. The goal is to identify inherent separations in the data. The idea is that there are different sources in the system generating samples and each source is responsible for a cluster. One could view the task as a classification problem without label information. Data belonging to a cluster should be close to one another while data from different clusters should be far appart. The concept of clustering is thus intimately linked to the definition of a similarity measure between samples. Different natural clusterings would be found just by changing the similarity measure. Moreover, the clusters could be overlapping such that the sources are not perfectly identifiable.

A task related to clustering is *quantisation*. The goal of quantisation is not to find natural separation anymore, but to represent the data with a small set of prototypes. Quantisation is thus a compression task. Each sample is associated to one prototype and the goal of learning is to maximise the similarity between the prototypes and their associated samples.

Another unsupervised learning problem that we will be looking at is the identification of subspaces in the distribution of vector data. The idea is that if there is some dependency between the data, the data should lie on a space of dimensionality smaller than their original dimensionality. In practice, useful information can already be extracted from a very low dimensional subspace of dimensionality 1 or 2. The term *manifold* is often used for nonlinear subspaces. However, identifying nonlinear manifolds is a very hard problem, in particular

when the original data are high dimensional. We will restrict our attention on linear manifolds. In Chapters 2 and 4, we present some methods for identifying linear subspaces in very different contexts.

A last unsupervised task that will be considered is the detection of atypical samples. In many real datasets, some samples are erroneous measures or just occurrences of very rare events. These points are problematic for the construction of the model and it is important to be able to spot them.

1.2 Model selection

In this section, we come back to the fundamental challenge of learning: deriving from the dataset a model which is able to generalise well. We have shown in the preliminaries that it corresponds to choosing an adequate training configuration $(\mathcal{H}, \mathcal{E})$. In the first part of this section, we illustrate the fundamental compromise related to the choice of the capacity of the training configuration. The second subsection sketches the basic concepts of modern learning theory for deriving the model selection criterion. This theory shows that one can evaluate an upper bound on the risk of the model trained even with a finite sample size. The last subsection presents the more pragmatic approach for model selection known as the resampling methods.

1.2.1 Capacity control

It is useful to illustrate the concepts of under-fitting and over-fitting on a simple 1D regression example. We will look at the influence of the size of the model space on the estimated model from the training stage. Let us suppose that the true dependency between targets and inputs can be expressed by $t(x) = f^*(x) + \epsilon$, where ϵ is an additive random noise term coming from a distribution $\epsilon \sim P(\epsilon)$ having its mean at zero. Suppose also that the loss function we want to optimise is the square loss. Consequently, as f^* is the mean of the true conditional distribution, it is also the optimal prediction function. The smallest risk we can hope to achieve is

$$\mathcal{R}^* \equiv \mathcal{R}(f^*) = \mathbb{E}_\chi \left\{ |t(x) - f^*(x)|^2 \right\} = \mathbb{E}\{\epsilon^2\} \,, \tag{1.8}$$

corresponding to the variance of the noise contribution. The goal is to estimate this prediction function from a set of samples. As discussed earlier, one needs to decide on the model space and the training criterion first. Let us take for the model space the set of polynomial functions of degree smaller or equal to M,

$$f \in \mathcal{H}_M : \quad f(x; \mathbf{w}) = \sum_{m=0}^{M} w_m x^m \,. \tag{1.9}$$

The model space is parameterised by the weights $\mathbf{w} = [w_0, \ldots, w_M]^\top \in \mathbb{R}^M$. Increasing the degree of the polynomial model space, we have a sequence of

nested model spaces $\mathcal{H}_0 \subset \mathcal{H}_1 \subset \mathcal{H}_2 \subset \ldots$. Additionally, let us take the empirical risk as training criterion, $\mathcal{E}(f) \equiv \mathcal{R}_{\mathcal{D}}(f)$. As shown in Chapter 3, the estimator of the optimal parameters $\hat{\mathbf{w}}$ is obtained with a linear solver.

The influence of the size of the model space is illustrated in Figure 1.1. Models on the upper row were estimated from an observed dataset containing 8 samples, while models on the lower row received 50 samples. The left column corresponds to polynomial functions of degree 2 and the right column to functions of degree 5. When the sample size is small, we clearly see the advantage of working in the smaller model space. The variability of the estimated model within this space is quite limited and the general dependency is easily spotted. On the other hand, with a small number of samples, we see that the larger model space significantly miss the inherent dependency of the function to estimate as the model fits part of the noise in the samples. However, when the number of samples increases, the advantage of working in a larger model space is clearly visible as the models estimated from this space can approximate almost perfectly the optimal function f^*. The smaller space does not contain good candidates for approximating f^* well.

To estimate the quality of the training configuration for a particular generative process $P(\chi)$, one should ideally measure the expected difference between the risk of the model obtained after training and the optimal risk:

$$\mathbb{E}_{\mathcal{D}}\{\mathcal{R}(\hat{h})\} - \mathcal{R}^* = \underbrace{\mathbb{E}_{\mathcal{D}}\{\mathcal{R}(\hat{h})\} - \mathcal{R}(h^\circ)}_{\text{Estimation}} + \underbrace{\mathcal{R}(h^\circ) - \mathcal{R}^*}_{\text{Approximation}} \tag{1.10}$$

where

$$h^\circ = \operatorname*{argmin}_{h \in \mathcal{H}} \mathcal{R}(h) \ .$$

The expectation is over the generation of datasets $\mathcal{D}$ (with fixed sample size). We have made explicit the two contributions to the discrepancy. The approximation term is present if the model space does not contain the optimal model h^*. In this case, there is an approximation bias between the best model in the model space h° and the optimal model h^*. The estimation discrepancy quantifies the tendency of the algorithm to miss the best model h° from the model space. This discrepancy can come from the tendency of fitting the noise instead of the inherent patterns (estimation variance), or the tendency of the training criterion to converge on average toward a model which is not h° (estimation bias).

As the space gets larger, the approximation bias decreases but the error from the estimation increases. There is therefore a compromise to be reached, which is called the *bias-variance trade-off* in standard statistical terms and which we call here the *control of the capacity* of the training configuration. Figure 1.2 shows this trade-off for the model presented above. We see that the optimal size depends on the number of samples. The optimal choice would be the polynomial functions of degree 2 when there are 8 samples and of degree 3 when there are 50. Of course, for real regression tasks, it is not possible to evaluate this risk and select the best size.

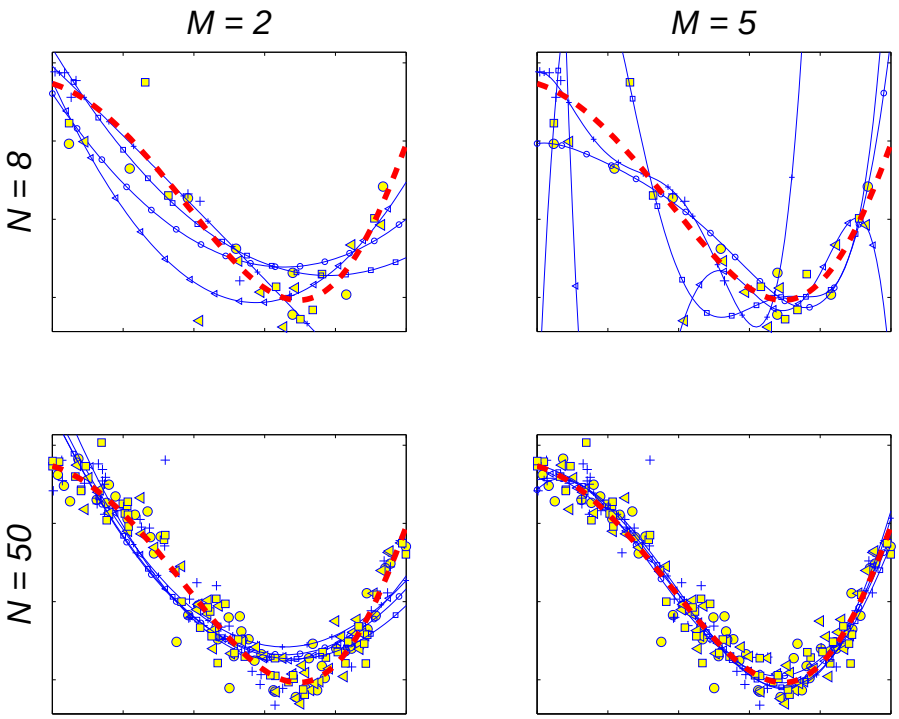


Figure 1.1: Illustration of the influence of the size of the model space on the model estimation. Left and right columns, polynomial functions of degree 2 and 5 respectively. Upper and lower rows, 8 and 50 training samples respectively. The optimal model is represented by the thick dashed line. Large markers stand for the learning samples.

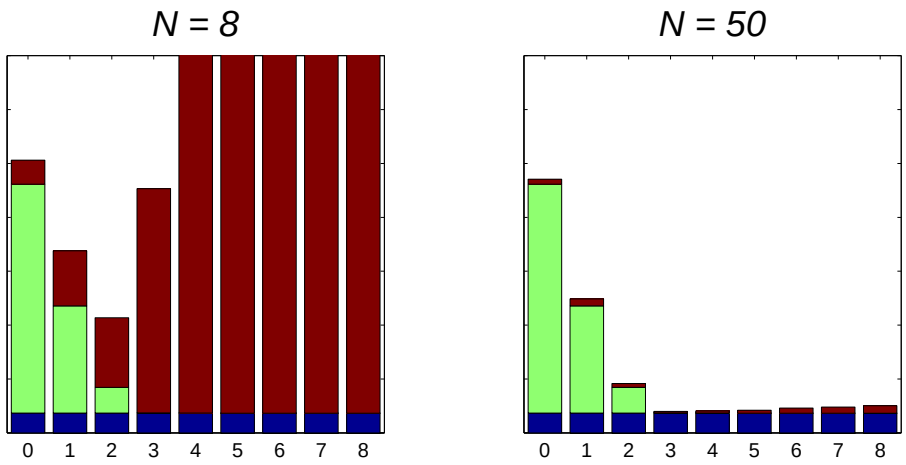


Figure 1.2: Average risk of trained model depending on the degree of the polynomial functions. Each bar is divided in the three contributions to the risk: noise (blue), approximation bias (green), estimation variance (red).

In fact, the size of the model space is not the only way to control the capacity of the training configuration. The second approach is to play on the training criterion. The majority of methods considered throughout the thesis are using a training criterion which has the following form

$$\mathcal{E}(h) = \sum_n \mathcal{L}_\mathcal{E}(\chi_n, h) + \mathcal{C}(h) \ . \tag{1.11}$$

Two terms appear in this expression. The term $\mathcal{C}(h)$ is called a *regulariser* [134]. It is a positive measure on the model space which associates with each model a prior cost to the training criterion. The regulariser is a way to bias the training estimate $\hat{h}$ toward models with low cost. Intuitively, one can understand that the cost of a model should be dependent on its tendency toward over-fitting, such that the additional estimation bias is compensated by a lower estimation variance. A classical regulariser for models like the example above would be to penalise the square norm of the weight vector $\mathbf{w}$,

$$\mathcal{C}(\mathbf{w}) = \lambda \|\mathbf{w}\|^2 \ , \tag{1.12}$$

where the hyper-parameter λ controls the trade-off between the two terms of the training criterion. This regulariser is known as the *ridge regulariser*. We will come back to this regulariser in Chapter 3.

The second element of the training criterion $\mathcal{L}_\mathcal{E}$ is simply a training loss function. One could wonder why the loss $\mathcal{L}$ associated with the learning task should not be used. First, because it could be too expensive to train the model on this particular loss $\mathcal{L}$. Secondly, choosing another loss can also be a way to

reduce the estimation variance (probably at the cost of an additional estimation bias). We will make use of this principle in Chapter 2.

1.2.2 Finite sample size bounds

Let us first define the property of the consistency of a learning procedure. A learning procedure is said to be consistent in a model space, if the risk of the estimated model converges to the optimal risk as the number of samples tend to infinity:

$$\lim_{N\to\infty} \mathcal{R}(\hat{h}) = \min_{h\in\mathcal{H}} \mathcal{R}(h) \ .$$

Moreover, if the model space grows with the number of samples such that at the limit $h^* \in \mathcal{H}$, then the procedure converges to the optimal model.

Showing that a learning procedure is consistent and deriving the rate of convergence was for a long time the best property that learning theory could derive. Unfortunately, this does not help much since the sample is always finite and it is difficult to tell how close to the limit we are. A major breakthrough in learning theory was made by Vapnik, V. N. and Chervonenkis [143] with the first general statement of a finite sample size bound on the risk of a model. For illustration, we give here one of the first bounds they derived. It applies to a classification task (with the 0-1 loss), when the classes might be overlapping. The statement tells us that with probability at least $1-\delta$

$$\forall h \in \mathcal{H}: \quad \mathcal{R}(h) \leq \mathcal{R}_{\mathcal{D}}(h) + \sqrt{\frac{v_{\mathcal{H}}(\log \frac{2N}{v_{\mathcal{H}}} + 1) - \log \frac{\delta}{4}}{N}} + \frac{1}{N} \ . \tag{1.13}$$

In other words, the risk can be bounded by the empirical risk plus some quantity, that we will call the *gap*. The gap tells us how over-optimistic we are in estimating the true risk by computing the empirical risk. As expected, the gap decreases with the number of samples. The dependency is the rate of convergence of the learning procedure. Also, we can verify that the gap increases with the required confidence level $1-\delta$ on the inequality. The important quantity for model selection is the quantity $v_{\mathcal{H}}$, called the *VC*-dimension of the model space (after the names of the authors). The VC-dimension measures the ability of models in this space to arbitrarily separate a set of points into two groups. Beneath this definition is the idea of fitting the arbitrary labels generated from the noise. Interestingly, this bound makes no particular assumption on the underlying data distribution, apart from the *i.i.d.* assumption. This type of bound is known as a *distribution free* bound. In fact, this property is rather a shortcoming. Indeed, by default a distribution free bound considers that the classification task is harder than it is in reality for most datasets. Such bounds are therefore generally quite loose.

Since the initial work, many other bounds have been obtained for different learning tasks. They are often expressed as *Probably Approximately Correct*

(PAC) statements [141], i.e.

$$P\left(\mathcal{R}(h) - \min_{h\in\mathcal{H}} \mathcal{R}(h) > \epsilon\right) \leq \delta \ , \tag{1.14}$$

and the statement makes explicit the relation between ϵ, δ, the model capacity and the data. The measure of the capacity of the model can vary. Some bounds use what is called the Rademacher complexity, others the concept of large margin, or sample compression. There is ongoing research in the field to formulate tighter bounds and bounds simpler to compute. Indeed, from a practical point of view, one wants to know if a bound is tight enough or at least well correlated with the risk (see for example [30]) and sufficiently simple to be used in the model selection procedure. For specific models like the support vector machine, some bounds are reducing the difference in performance that separates them from the cross-validation estimates discussed in the next subsection, e.g. [2]. But one must admit that selecting the best configuration from a theoretical bound is not yet the method of choice. Nonetheless, it can give a good intuition on the dependency between the configuration and the generalisation performances.

1.2.3 Resampling methods

Resampling methods follows a more pragmatic route for estimating the generalisation performances (or the risk) of a model. As the risk is by definition measured on data generated independently of the observed samples, the idea of resampling methods is to split the observed samples in two sets, one used for training the model, and the other for evaluating its generalisation performances. We have thus

$$\mathcal{D} = \mathcal{D}_L \cup \mathcal{D}_V \ , \tag{1.15}$$

where $\mathcal{D}_L$ is the learning set used to train the model and $\mathcal{D}_V$ is the held-out set. The risk is estimated on this held-out set by

$$\hat{\mathcal{R}}(h) = \mathcal{R}_{\mathcal{D}_V}(h) \ , \tag{1.16}$$

i.e. the empirical risk on the held-out data. Different training configuration $(\mathcal{H}, \mathcal{E})$ can be evaluated with this estimator, and one can thus select the best configuration. This procedure is called a *validation* procedure for model selection. Note that, because the held-out data are used to select the best configuration, the estimated risk at the selected configuration is expected to be an over-optimistic estimate of the risk.

Like any estimator computed on a finite sample size, $\hat{\mathcal{R}}(h)$ is subject to the fundamental bias-variance trade-off. When the size of the held-out set increases, the estimator has a lower variance but the estimated model on the other hand deteriorates as it is constructed with fewer learning samples. To reduce the variance of the estimator while keeping a small held-out set, one can repeat the splitting procedure and average over the estimates [128]. There are

many schemes for doing so. The simplest one is just to repeat a fixed proportion random split. However, one prefer in general to give *a priori* equal importance to all samples of the dataset, i.e. have each sample in the validation set the same number of times. For this purpose, on should perform a K-fold cross-validation: the complete dataset $\mathcal{D}$ is separated in K approximately equal size subsets $\{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$. Each of these subsets is in turn left out of the learning stage. The final estimator for the risk of a particular configuration is then

$$\hat{\mathcal{R}}(\mathcal{H}, \mathcal{E}) = \sum_k \mathcal{R}_{\mathcal{D}_k}(\hat{h}_k) \; . \tag{1.17}$$

At the limit, K can be set to N, such that learning proceeds with the complete dataset except one sample. This is called the *leave-one-out* scheme [128, 129]. For some simple models, linear regression models in particular (see Chapter 3), the leave-one-out estimated risk can be computed from a single learning run without actually splitting the dataset. It has been shown that the leave-one-out estimator has a small bias $\mathbb{E}_{\mathcal{D}}\{\hat{\mathcal{R}}_{loo}(h)\} \approx \mathcal{R}(h)$, but it has a higher variance than cross-validation with larger subsets [71]. The bootstrap [42, 122] is another resampling scheme that is very useful in assessing the accuracy of a statistical estimator as it makes no assumption on the underlying distribution (except the i.i.d. assumption). For this method, each bootstrap training set is composed of N samples picked randomly with replacement from the observed dataset. The validation set corresponding to a bootstrap set consists of the samples not picked for training. It was shown that the estimated risk computed by the bootstrap method can be further improved to account for the bias of this resampling scheme (see [54, 40]). In practice, for the models constructed in this thesis, the computational load makes it impossible to repeat the training procedure a sufficient number of times to benefit from the bootstrap method. We will thus only work with the simpler cross-validation scheme.

The big advantage of resampling methods are that they do not make any particular assumption regarding the dataset and they can be used to select any hyper-parameter appearing in the modelling. However, obtaining a good estimator of the risk requires repeating the learning a sufficient number of times. It is usual to start with a 5-fold cross-validation but of course this really depends on the dataset and the precision required for the estimation of the risk. For some models, it is possible to derive an numeric optimisation on the resampling criterion (e.g. a Nelder-Mead simplex optimisation [27]). Unfortunately, in general the cross-validation scheme must be carried out by a tedious search on the set of configurations $(\mathcal{H}, \mathcal{E})$ that could be adequate. When the configurations differ only by the value of a single hyper-parameter, it is possible to provide a good coverage of the range where we think this hyper-parameter could take its optimal value. For two hyper-parameters, we would probably have to make a grid search, which is already very costly for most models used in this thesis. And looking for more than two hyper-parameters is in general too time-consuming.

One should be conscious that while resampling methods are very robust to

over-fitting, they can still undergo some indirect over-fitting. Indeed, one can by chance come across a model space which is in the same time small (such that the estimation variance is small) and yet over-fits the data (such that the bias will seem to be small). This typically occurs when a large number of model configurations are tested and refined. Further insight is given in [152].

1.3 Probabilistic modelling

There are two ideas underpinning the probabilistic modelling approach. The first idea is to model the dispersion of data with a probability distribution and the second idea is to express by another probability distribution the prior uncertainty that one has about the true model. Let us begin with the first idea. Modelling the dispersion (or the noise, the randomness...) present in the data can be useful for several reasons. As described earlier in the chapter, ideally the learning problem should be specified by a loss function which summarises the goal of the analysis. However, in practice it is often impossible to decide initially which loss function should be preferred over the others and moreover the loss could change with the circumstances. For example, judging if a medicine should be administered to someone really depends on the probability to developing side effects and the harm it could have on this particular person. Ultimately, the decision is binary (take the medicine or do not take it) but assessing the probability of the side effects of the medicine is of great importance here. Another reason why modelling randomness can be useful is when one has intuition about the process generating the data. In Chapter 4, the analysis focuses on the modelling of user's behaviour. For that application, it is quite natural to represent behaviours by means of probability distributions over the possible decisions that the users can make.

Formally, the principle is thus to represent the underlying distribution of data $P^*(\chi)$ by means of a probabilistic model $h_\chi \equiv P_h(\chi)$ chosen from a model space of distributions $\mathcal{H}_\chi$. Note that here again we will concentrate on *i.i.d.* probabilistic models. Optimising h_χ based on a training criterion (e.g. maximum likelihood, see below), we obtain an estimator for the underlying distribution $\hat{P}(\chi) \equiv \hat{h}_\chi$. From this probabilistic model, one can derive an estimate of the risk associated with any loss function. Indeed, given the loss $\mathcal{L}$, the risk of a model $h_\mathcal{L}$ is

$$\hat{\mathcal{R}}(h_\mathcal{L}) = \hat{\mathbb{E}}_\chi\{\mathcal{L}(\chi, h_\mathcal{L})\} \,, \tag{1.18}$$

where $\hat{\mathbb{E}}_\chi$ stands for the expectation over $\hat{P}(\chi)$. The subscript is there to avoid confusion between $h_\mathcal{L}$ (related to $\mathcal{L}$) from h_χ. Selecting the best model $\hat{h}_\mathcal{L}$ once h_χ has been estimated is in general a simple problem. But arguably, estimating a probability distribution is a problem harder than directly minimising a particular loss function. It was shown for example that the minimisation of the 0-1 classification loss converge "infinitely" faster than the estimation of the true conditional probability distribution of belonging to each class [40]. In other

words, it is not interesting to follow the probabilistic approach if one does not intend to use the information about the dispersion of data.

As previously mentioned, there is a second idea often coming along with the probabilistic modelling approach. This idea is that not only dispersion of the data but any kind of uncertainty can be represented by probability distributions. This postulate is at the heart of the Bayesian framework. In this framework, the uncertainty about the model itself is expressed by a prior distribution. This construction leads to conditional hierarchical distributions. We will show how the Bayesian framework gives us a pragmatic scheme for adjusting the hyper-parameters of the training configuration.

1.3.1 Maximum likelihood fit

We will now show the standard approach to train probabilistic models. As described earlier, the supervised learning task is simpler because it is not necessary to know the distribution of inputs to evaluate an optimal target t. Indeed, for a given loss function, the optimal prediction for the target at a particular input location $\mathbf{x}$ is

$$f^*(\mathbf{x}) = \underset{y}{\operatorname{argmin}}\, \mathbb{E}_{t|\mathbf{x}}\{\mathcal{L}(t, y)\} \; . \tag{1.19}$$

Hence, only the conditional target distribution $P(t|\mathbf{x})$ is required.

Probabilistic models are also called *generative models* because one could generate pseudo samples from them. The natural criterion to measure the adequacy of the model with the data is the (density of) probability that the distribution model generated these data:

$$P(\{t_n\}|\{\mathbf{x}_n\}) = \prod_n P(t_n|\mathbf{x}_n) \; . \tag{1.20}$$

This measure is called the *likelihood* of the model. Taking the negative logarithm of this measure, one comes back to an empirical risk formulation

$$\mathcal{R}_{\mathcal{D}}(h_\chi) = -\sum_n \log P(t_n|\mathbf{x}_n) \; . \tag{1.21}$$

Note that for a continuous distribution, this quantity can be arbitrarily low when the distribution gets sharply peaked on the observations.

For regression, the classical assumption when one has little knowledge about the dispersion of targets is to assume an additive homoscedastic noise distribution

$$t(\mathbf{x}) = f(\mathbf{x}) + \epsilon \tag{1.22}$$

where $\epsilon \sim P_\epsilon(\epsilon)$ corresponds to the noise term. A standard assumption for the noise is a Gaussian distribution with mean at zero, $\epsilon \sim \mathcal{N}(0, \tau^{-1})$, where τ is the noise precision (i.e. the inverse noise variance). The empirical risk (i.e. the negative log-likelihood) is

$$\mathcal{R}_{\mathcal{D}}(h_\chi) = \frac{\tau}{2} \sum_n (t_n - y_n)^2 + \frac{N}{2} \log 2\pi\tau^{-1} \; , \tag{1.23}$$

where $y_n = f(\mathbf{x}_n)$. We observe that the square error loss appears in this criterion. In fact, estimating the function $f(\mathbf{x})$ with this criterion would return the same model as with the square error loss. But to be complete, the learning must also estimate the precision τ. Actually, formulating the learning problem by a Gaussian noise model instead of a square error minimisation problem really depends on the use of this precision parameter. It is frequent that one is interest in estimating error bars around the mean prediction and for this purpose the probabilistic approach is useful. But in this thesis, we will use the Gaussian noise formulation mainly in order to derive a Bayesian model selection procedure (see next subsection).

The Gaussian noise assumption is adequate when the conditional dispersion of targets is close to a Gaussian, i.e. the dispersion is essentially unimodal, with mean close to zero and concentrated. This last requirement is maybe the one that is most frequently erroneous in practice. Indeed, real data often exhibit samples quite far from the mean. The estimation of the mean function $f(\mathbf{x})$ might be strongly influenced by these samples because of the square contribution in the training criterion. In Chapter 2, we adopt a longer tail noise distribution model, namely the Student-t distribution, to restrain the influence of samples away from the mean function.

The classification task is very similar, if not simpler. Instead of working with a conditional distribution on the ordered set of targets, one works with a conditional discrete distribution on the unordered set of labels. For binary classification, there is only one possibility that is the conditional Bernoulli distribution. It is usual to encode the labels with $t \in \{0, 1\}$ and specify the distribution by $P(t = 1|\mathbf{x}) = f(\mathbf{x})$. So the dependency function represents the conditional probability that the label is 1. Of course, the function must satisfy $0 \leq f(\mathbf{x}) \leq 1$. The empirical risk, or negative log-likelihood of this model is

$$\mathcal{R}_{\mathcal{D}}(h_\chi) = -\sum_n t_n \log y_n + (1 - t_n) \log(1 - y_n) \; . \tag{1.24}$$

where again we noted $y_n = f(\mathbf{x}_n)$.

For the unsupervised task, the problem is more complicated because without prior knowledge it corresponds to modelling the complete data distribution $P(\chi)$. However, as said above, one can be satisfied with a simple representation of the data. We have given some comments about the clustering and the subspace identification tasks in Section 1.1.4. Clustering is typically tackled in probabilistic modelling by mixture models ([139]):

$$P(\chi) = \sum_k P(\chi|\theta_k)\pi_k \; , \tag{1.25}$$

where $P(\chi|\theta_k)$ is the distribution of the kth source of the mixture, and π_k is the probability that a sample is generated from this same source. The parameters of this model are thus the component parameters θ_k and the repartition parameters π_k. To get to a (local) maximum of the likelihood of a mixture

model, it is common to use an Expectation-Maximisation (EM) algorithm [39]. The principle of the EM algorithm will be exposed in Section 1.3.3.

For the task of subspace identification, we will show in the next chapter that one approach is to represent the original data by means of a latent space of low dimensionality and attribute the discrepancy between the two representations (original and latent) to the noise.

Like any empirical risk minimisation procedure, maximising the likelihood or equivalently minimising the negative log-likelihood can lead to over-fitting. Here also, the capacity of the training configuration should be controlled to find a solution close to the optimum. Again, theoretical bounds and cross-validation methods are possibilities to estimate this optimum. We will see that the Bayesian approach can also be useful to select the best configuration.

1.3.2 Bayesian framework

In the Bayesian framework, the uncertainty about the model before seeing the data is expressed by means of a prior distribution $P(h)$. This distribution should reflect our confidence about the different models that could have generated the data. To simplify the derivation, we will consider a parametric model, such that the prior uncertainty is expressed directly on the parameters $P(\theta)$. With this formulation, the parameters are considered as random variables, like the data. The true model generating the data is assumed to belong to the space of models consistent with the prior distribution. We will discuss this assumption shortly.

One could be tempted to find the estimate of θ maximising the joint (density of) probability over both the data and the parameters

$$\hat{\theta} = \operatorname*{argmax}_{\theta \in \Theta} P(\mathcal{D}, \theta) = \operatorname*{argmax}_{\theta \in \Theta} P(\mathcal{D}|\theta) P(\theta) \ , \tag{1.26}$$

where we have just applied the product rule to make apparent the likelihood and the prior distribution. Equivalently, minimising the negative log-likelihood of this joint distribution, we would obtain a training criterion in the same formalism as (1.11)

$$\mathcal{E}(\theta) = -\sum_n \log P(\chi_n|\theta) - \log P(\theta) \ . \tag{1.27}$$

We see that the prior distribution defines the regulariser $\mathcal{C}(\theta) = -\log P(\theta)$. There is a constraint on such regulariser encoded in the integrability of the prior distribution.

One can do better than the point estimate proposed above. Applying again the product rules on the joint probability, we have

$$P(\mathcal{D}, \theta) = P(\mathcal{D}|\theta) \ P(\theta) = P(\theta|\mathcal{D}) \ P(\mathcal{D}) \ , \tag{1.28}$$

and isolating one of the factor distributions

$$P(\theta|\mathcal{D}) = \frac{P(\mathcal{D}|\theta) \cdot P(\theta)}{P(\mathcal{D})} \tag{1.29}$$

$$\text{Posterior} = \frac{\text{Likelihood} \cdot \text{Prior}}{\text{Marginal Likelihood}} .$$

This is known as the *Bayes formula*. The posterior over parameters tells us how uncertain we still are about the value of the parameters after seeing the data. Note that the idea to estimate the model from the joint distribution corresponds to a maximum a posteriori (MAP) estimate - the marginal likelihood being independent of the parameters. The role of the marginal likelihood will be described below. The strength of the Bayesian formulation is that the posterior uncertainty can be taken into account in making inference about the data distribution. This is obtained by integrating out the parameters, i.e.

$$\begin{aligned} P(\chi|\mathcal{D}) &= \int_\Theta P(\chi, \theta|\mathcal{D}) \; d\theta \\ &= \int_\Theta P(\chi|\theta) P(\theta|\mathcal{D}) d\theta \end{aligned} \tag{1.30}$$

which is called the *posterior predictive distribution*. We see that formally the Bayesian framework leads to a linear combination of models selected from a particular model space, where the weighting of models is given by their posterior distribution. From this predictive distribution, one can derive the optimal prediction for a particular loss function as discussed above.

The Bayesian framework can be misleading because the statement it derives about the posterior uncertainty are only (approximately) valid if the actual generative process is well represented by the model space. If the reality is very different, the inference will be completely erroneous. In practice, the posterior distribution tends to be over-confident because it does not take into account the uncertainty about all the models not belonging to the model space. Another source of confusion is the choice of the prior distribution. Contrarily to the spirit of the procedure described above, this choice of a prior distribution often seems arbitrarily or dictated by mathematical convenience. One might wonder if the Bayesian framework is still legitimate.

First, let us note that the consequences of choosing a bad prior distribution disappear as the number of samples increases because the likelihood becomes dominant. The prior distribution should really be thought as a way to bias the estimates toward some intuitively good model. With little intuition, it is usual to set a very vague prior on the parameters, such that the posterior distribution is almost completely determined by the likelihood. Second, we can be critical about the validity of the posterior uncertainty. Nevertheless, an analysis of this posterior distribution can help getting some understanding of the generative process. This idea will be followed in Chapters 2 and 4 with the mixture models. A good practice is always to check the "relative" validity of the model by means of a more robust approach like the resampling methods.

One thing that is maybe not so evident is that the training stage is the estimation of the posterior distribution. At the model selection stage on the other hand, one needs to evaluate the adequacy of the likelihood-prior configuration with the data. Actually, the Bayesian approach gives us a measure of this adequacy by means of the marginal likelihood (also called the *evidence*) appearing in (1.29). This quantity is formally the likelihood of the configuration. One way to compute it is by marginalising out the parameters from the joint distribution

$$P(\mathcal{D}) = \int_{\Theta} P(\mathcal{D}, \theta) d\theta = \int_{\Theta} P(\mathcal{D}|\theta)\, P(\theta)\, d\theta\ . \tag{1.31}$$

In practice, the likelihood and prior distribution can be adjusted by a set of hyper-parameters. Their respective hyper-parameters will be noted $\eta = (\eta_\ell, \eta_0)$. From this derivation, it should be clear that the marginal likelihood is nothing more than a higher-level training criterion. We will see in the next section that the marginalisation of the parameters has a damping effect on the tendency to over-fit.

1.3.3 Marginal Likelihood fit

Marginalisation of parameters in (1.31) can be done analytically only for simple models. For more complex ones, it is necessary to resort to approximations to evaluate and optimise this criterion. We will start by showing a simple procedure, called the Laplace approximation, which can be used when the parameters are taken from a continuous space, preferably not bounded. The procedure will be applied on some of the models in Chapter 3.

The second procedure is the Expectation-Maximisation algorithm. We will present the algorithm as an indirect manner to find the maximum likelihood estimate of some parameters of a model. Expressing the task by means of a marginal likelihood expression simplifies the optimisation.

There exist many other strategies to find approximate marginal likelihood estimates. The most versatile methods are sampling methods. The principle is to generate samples from the posterior distribution and base the estimation of the marginalisation on these samples. One of the procedures to generate the samples is the Markov Chain Monte Carlo (MCMC) method. The difficulty in practice is to generate samples sufficiently rapidly from every part of the posterior distribution. For complex nonlinear models, this is a true challenge. We will not use these methods in the thesis. Additional information can be found for example in [95].

Laplace Approximation The Laplace approximation is a simple procedure to evaluate the marginal-likelihood when the parameters are continuous and preferably not bounded. Let us write (1.31) again making explicit the presence of the hyper-parameters $\eta = (\eta_\ell, \eta_0)$:

$$P(\mathcal{D}|\eta) = \int_{\Theta} P(\mathcal{D}|\theta, \eta_\ell) P(\theta|\eta_0)\, d\theta \ . \tag{1.32}$$

First, we define the function $g_\ell(\theta)$ and $g_0(\theta)$ by rearranging the likelihood and the prior distribution in the following way

$$P(\mathcal{D}|\theta, \eta_\ell) = \frac{\exp -g_\ell(\theta)}{Z_\ell} \ , \tag{1.33}$$

$$P(\theta|\eta_0) = \frac{\exp -g_0(\theta)}{Z_0} \ , \tag{1.34}$$

where Z_ℓ and Z_0 are normalisation factors independent of θ. We define also $g(\theta) = g_\ell(\theta) + g_0(\theta)$.

Now, we search for the parameters that minimise the function $g(\theta)$. The optimal parameters are noted $\bar{\theta}$. One can verify easily that this estimate is simply the maximum a posteriori (MAP) estimate of (1.26).

The function $g(\theta)$ is then approximated by a Taylor expansion around the optimum

$$g(\theta) \approx g(\bar{\theta}) + \frac{1}{2} \Delta\theta^T \ \Sigma_\theta^{-1} \ \Delta\theta \tag{1.35}$$

where $\Delta\theta = \theta - \bar{\theta}$ and Σ_θ^{-1} is the Hessian of $g(\theta)$ with respect to the parameters. The first order term is absent as the gradient at $\bar{\theta}$ is null. Plugging the expansion in (1.29), one can understand the procedure as approximating the posterior distribution of the parameters by a multivariate normal distribution centred in the MAP,

$$P(\theta|\mathcal{D}, \eta) \approx \mathcal{N}(\bar{\theta}, \Sigma_\theta) \ . \tag{1.36}$$

The approximation is quite bad when the true posterior distribution is heavy tailed or when it is multimodal. On the other hand, if the posterior is unimodal and not too heavy tailed, we can get a good estimate of the marginal likelihood from

$$P(\mathcal{D}|\eta) \approx \int_{\Theta} \frac{\exp\left(-g(\bar{\theta}) - \frac{1}{2}\Delta\theta^T \ \Sigma_\theta^{-1} \ \Delta\theta\right)}{Z_\ell Z_0}\, d\theta \tag{1.37}$$

which is

$$P(\mathcal{D}|\eta) \approx (2\pi)^{\frac{M}{2}} \cdot \frac{\exp -g(\bar{\theta})}{Z_\ell} \cdot \frac{|\ \Sigma_\theta\ |^{\frac{1}{2}}}{Z_0} \ . \tag{1.38}$$

In this expression, the middle factor represents the quality of fit at the MAP estimate. The marginal likelihood increases with this fit, favouring large model spaces. But the last factor on the other hand counterbalances this tendency. This last factor is fundamentally the ratio of the posterior to the prior volumes accessible by the parameters. Large models space have a large prior accessible volume Z_0, while over-fitting is characterised by a small posterior volume |

$\Sigma_\theta\,|^{\frac{1}{2}}$. This ratio is sometimes referred to as the *Occam factor*[1] of the Bayesian framework since it penalises model spaces that are too large.

Now, the hyper-parameters η are adjusted in order to maximise the approximated marginal likelihood. Note that it is numerically more stable to work with the logarithm of the expression (1.38),

$$\log P(\mathcal{D}|\eta) \approx C^{te} - g(\bar{\theta}) - \log Z_\ell - \frac{1}{2}\log \mid \Sigma_\theta^{-1} \mid - \log Z_0 \; . \tag{1.39}$$

This expression can be differentiated with respect to the hyper-parameters η and be used in a gradient ascent algorithm. An entry of the gradient is computed by

$$\frac{\partial \log P(\mathcal{D}|\eta)}{\partial \eta_i} \approx -\frac{\partial g(\bar{\theta})}{\partial \eta_i} - \frac{\partial \log Z_\ell}{\partial \eta_i} - \frac{1}{2}\mathrm{tr}\left(\Sigma_\theta \frac{\partial \Sigma_\theta^{-1}}{\partial \eta_i}\right) - \frac{\partial \log Z_0}{\partial \eta_i} \; . \tag{1.40}$$

One must be conscious that adjusting the hyper-parameters can also lead to over-fitting in spite of the internal regularisation discussed above. This often happens when the likelihood hyper-parameters η_ℓ under estimate the noise in the data or more generally when the assumptions made by the probabilistic modelling do not encode the underlying distribution of data well. Adding a hierarchical level on the hyper-parameters (i.e. a prior distribution on the space of these hyper-parameters) is should make the learning more robust. Unfortunately, it is at the cost of simple tractable (or approximate) schemes.

Another strategy would be to combine the marginal likelihood with some resampling methods. Finally, note that the marginal likelihood often has multiple optima and one should thus try to verify that a better solution than the one reached cannot be found.

In the end, the marginal likelihood defines a pragmatic selection criterion using which several hyper-parameters of the configuration can be fitted. This is an important aspect of the Bayesian selection framework. To be fair though, we must say that it is not the only selection criterion that can be optimised with a gradient descent procedure on multiple hyper-parameters (e.g. the leave-one-out). The question is to know if the marginal likelihood can be useful on real datasets when generic probabilistic models are used, i.e. the models are not based on much expert or intuitive knowledge. This question will be discussed from the experimental comparison presented in Chapter 3.

Expectation-Maximisation algorithm For many simple models, and in particular mixture models, one can directly evaluate the likelihood of the parameters $P(\mathcal{D}|\theta)$. It is thus possible to derive the gradient and find an optimum solution from an iterative search. However, computing the gradient of such expression can be tricky and the gradient ascent algorithms are not always well adapted for handling large parameter sets common to these models.

[1] This name comes from the Occam's Razor principle (or *law of parsimony)* which makes the following statement: "All things equal, the simplest solution tends to be the best one".

The Expectation-Maximisation (EM) [39] algorithm is a simplified procedure to converge to an optimum of the likelihood of a model. The trick is to make explicit some *latent states* (also called *hidden states*) of the generative process and to marginalise over these states. We will represent the latent states by Z. The corresponding marginal likelihood is thus

$$P(\mathcal{D}|\theta) = \int_{\mathrm{Z}} P(\mathcal{D}|Z, \theta_\ell) P(Z|\theta_0) dZ \ . \tag{1.41}$$

Note that the procedure derived below could be applied on the parameter-hyper-parameter formulation used so far. For conceptual reasons that will get clearer later in the thesis, we have preferred to introduce the concept of latent states.

The algorithm proceeds by decoupling the marginalisation and the update of the parameters. Making use of the convexity of the logarithm function and applying the *Jensen inequality*, we can write

$$\begin{aligned} \log P(\mathcal{D}|\theta) &\geq \int_{\mathrm{Z}} Q(Z) \log \left(\frac{P(\mathcal{D}|Z, \theta_\ell) P(Z|\theta_0)}{Q(Z)} \right) dZ \\ &= \mathbb{E}_Q \left\{ \log P(\mathcal{D}, Z|\theta) \right\} - \mathbb{E}_Q \left\{ \log Q(Z) \right\} \end{aligned} \tag{1.42}$$

where $Q(Z)$ can be any distribution over the latent states Z. The right hand side is thus a lower bound on the marginal likelihood. It is sometimes referred to as the negative *free energy* [97]. The first term of the second line is called the *expected log complete likelihood*. The second term on this same line is independent of the parameters θ and it corresponds to the entropy of the distribution $Q(Z)$.

Instead of directly maximising the marginal likelihood, we maximise this lower bound by updating alternatively the distribution $Q(Z)$ and the parameters θ. Using the definition of the posterior distribution (1.29), we can re-write this inequality as

$$\log P(\mathcal{D}|\theta) \geq \log P(\mathcal{D}|\theta) - \mathcal{KL} \left(Q(Z) || P(Z|\mathcal{D}, \theta) \right) \tag{1.43}$$

where $\mathcal{KL}(\cdot||\cdot)$ represents the Kullback-Liebler (KL-) divergence between two distributions. This quantity is always positive, and null only if both distributions are identical. This equivalent expression for the bound tells us that the bound is maximised when the distribution $Q(Z)$ is set to the posterior distribution $P(Z|\mathcal{D}, \theta)$. When this is the case, the bound is tight and we have just derived another manner to compute the marginal likelihood. The advantage is that the maximisation of the lower bound with respect to the parameters is often a simple fitting problem, formulated as maximising the expected log-complete likelihood

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmax}} \, \mathbb{E}_Q \left\{ \log P(\mathcal{D}, Z|\theta) \right\} \tag{1.44}$$

which for many simple models can be separated in two analytical updates over θ_ℓ and θ_0.

Estimating the posterior distribution and computing the expectation with respect to this posterior is called the *E-step*. The marginalisation really takes place when the expectation is computed. The update for the hyper-parameters is called the *M-step*. The procedure must be iterated as the posterior depends on the value of the hyper-parameters. So, after their update the bound is no longer tight. Note that the lower bound can only increase during both steps. It was proved that this procedure converges to a local maximum of the likelihood. However, the convergence is often slow for large models.

More generally, if the posterior distribution is intractable, one can resort to a variational approximation of the EM algorithm by constraining $Q(Z)$ to have a simple tractable form. A frequent scheme is the mean field approximation (e.g. [120]) for which $Q(Z)$ is factorised along each set of latent states.

Chapter 2

Robust probabilistic linear subspaces

In the preliminary stages of data analysis, the practitioner is often interested in having a general idea of the patterns present in the data. Identifying clusters and low-dimensional subspaces along which the data are lying provides precious information. A way to identify these dense subspace regions is to combine local low-dimensional linear models. This approach was proposed in the model called *mixture of probabilistic principal component analysers* [136]. An important issue with this model is its sensitivity to atypical data lying far from the main clusters. These points significantly bias the estimate of the linear subspaces. In this chapter, we show how the issue can be addressed by using a more robust training criterion which allows us to discard the contributions of atypical points in the estimation of the model parameters. More precisely, we build on the probabilistic formulation, replacing the standard assumption of a Gaussian noise distribution by a longer tail distribution, the Student-t. We derive an exact Expectation-Maximisation algorithm to train the new model. The resulting mixture model is an extension of the standard mixture of Gaussians, but is much more applicable as it gives at the same time a robust clustering and a concurrent dimensionality reduction for exploratory analysis. Its use is illustrated in the last section. The work was published in [4, 6].

2.1 Preliminaries

The discussion of the models in this chapter is mainly oriented toward the exploratory data analysis task. We start the section by sketching the role of this preliminary task. We then give a summary description of the Principal Component Analysis (PCA) that is the most well known linear subspace identification method. We will see that probabilistic models presented in Section 2.2 are highly related to this method. Finally, we discuss the origin of atypical data

and their influence on the model estimation.

2.1.1 Exploratory data analysis

The goal of the exploratory analysis is to help the practitioner identifying the main patterns present in the dataset and maybe help him to have some intuition on how to guide the design of a good model. For this purpose, methods allowing us to visualise the data with a low-dimensional representation are of precious help. A one-dimensional or two-dimensional representation is often preferred as higher dimensional representations require some training to be used properly.

Exploratory analysis generally occurs in the unsupervised framework, i.e. the learning task is not given particular conditional dependency to look for from the outset. When one has little knowledge about the types of patterns present in the data, looking for clusters is a good starting point. The concept of cluster was introduced in the previous chapter. Recall that the idea behind clustering is to identify distinct sources from which the data originate.

Besides clustering, identifying low-dimensional subspaces along which the data lie also provides interesting information. The dependencies expressed by the subspace are related to constraints in the system from which the data are collected. From linear subspaces, one can identify correlation between variables. Of course, the world is complex and systems are usually governed by nonlinear constraints. Unfortunately, the identification of nonlinear subspaces is much more complicated [76]. The most widely used method for identifying and visualising nonlinear subspaces is the *Self-Organising Map* (SOM) [72, 73]. Self-Organising Maps have been applied in many data analysis problems [101]. However, SOMs are not specifically designed to identify clusters, unlike the mixture models that we present in this chapter. These mixture models can only identify linear subspaces. It would be interesting to compare them with the SOMs and see if there is a qualitative difference between the approaches (in particular for moderately high-dimensional data).

In the previous chapter, we have seen that adjusting a model with a learning procedure requires the definition of an objective criterion, called the loss function. For clustering, the loss function should be a measure of the homogeneity of samples within each cluster. Different measures of homogeneity have been proposed in the literature; see [66] for example. Setting the number of clusters is also a delicate question as the optimal number might depend on the loss function used. In fact, it is already useful to identify the dominant clusters and simplify the preliminary analysis. On the other hand, subspace identification is more adequately formulated with a loss function measuring a reconstruction error (see next subsection). Both homogeneity and low reconstruction error requirements should be combined in a unique loss function.

As discussed in Section 1.3, probabilistic models try to estimate the underlying density of the data. They do not directly focus on the homogeneity criterion or the reconstruction error criterion of the clustering and subspace identification tasks. In this sense the strategy is suboptimal, but we have shown that

from the density estimation one can derive an estimation of other loss functions. Moreover, the exploratory analysis is for the most part subjective. The intuition gained from the analysis cannot be measured by a quantitative criterion. We will thus essentially try to show how the proposed models are useful in obtaining some intuition about the patterns in the data.

2.1.2 Principal Component Analysis

Principal Component Analysis (PCA) is a well-known statistical technique for linear dimensionality reduction [62, 68]. PCA is used as a pre-processing step in many applications involving data compression or data visualisation [69]. The goal is to find successive orthogonal directions in the original data space such that the variances along these directions are maximised. These directions are called the *principal components*.

Let the set of observations be noted $\mathcal{D} = \{\mathbf{y}_n\}_{n=1}^N$, with $\mathbf{y}_n \in \mathbb{R}^D$and the J first principal components $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_J$ with $\mathbf{v}_j \in \mathbb{R}^D$. The PCA is formulated as follows:

$$\begin{aligned} \text{For } j = 1 \ldots J, \qquad & \max_{\mathbf{v}_j} \mathbb{E}_{\mathbf{y}} \left\{ \mathbf{v}_j^\top (\mathbf{y} - \boldsymbol{\mu}) \cdot (\mathbf{y} - \boldsymbol{\mu})^\top \mathbf{v}_j \right\} \\ = \quad & \max_{\mathbf{v}_j} \mathbf{v}_j^\top \boldsymbol{\Sigma} \mathbf{v}_j \\ & \text{subject to} \quad \|\mathbf{v}_j\| = 1,\ \mathbf{v}_j^\top \mathbf{v}_{j'} = 0 \quad \forall j' < j \ . \end{aligned} \tag{2.1}$$

The expectation on the first line corresponds to the variance along the direction $\mathbf{v}_j$. The symbol $\boldsymbol{\mu}$ represents an offset such that the principal components (or the associated subspace) pass through the offset $\boldsymbol{\mu}$ instead of the origin of the space which is often arbitrary. In general, the mean is chosen $\boldsymbol{\mu} = \mathbb{E}_{\mathbf{y}}\{\mathbf{y}\}$, and in this case $\boldsymbol{\Sigma} = \mathbb{E}_{\mathbf{y}}\{(\mathbf{y} - \boldsymbol{\mu}) \cdot (\mathbf{y} - \boldsymbol{\mu})^\top\}$ is the $D \times D$ covariance matrix of the $\mathbf{y}$-distribution. The last line in (2.1) expresses the orthogonality constraint between the principal components. This formulation shows that the principal components are the successive dominant eigenvectors of $\boldsymbol{\Sigma}$. The number of components J is often specified by the proportion of the total variance that one is willing to discard (there is thus a compression trade-off, see the discussion in the previous chapter).

The principal components define a J-dimensional linear subspace with the orthogonal basis matrix $\mathbf{V}_J = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_J]$. The projected low-dimensional vectors $\mathbf{x} = \mathbf{V}_J^\top (\mathbf{y} - \boldsymbol{\mu})$ are called the *principal coordinates*. One can show that this subspace is optimal in the sense that it minimises the square reconstruction error

$$\begin{aligned} \mathbf{V}_J \quad = \quad & \underset{\mathbf{V} \in \mathbb{R}^{D \times J}}{\operatorname{argmin}} \ \mathbb{E}_{\mathbf{y}} \left\{ \|(\mathbf{y} - \boldsymbol{\mu}) - \mathbf{V}\mathbf{x}\|^2 \right\} \\ & \text{subject to} \quad \mathbf{V}^\top \mathbf{V} = \mathbf{I}_J, \qquad \mathbf{x} = \mathbf{V}^\top (\mathbf{y} - \boldsymbol{\mu}) \ . \end{aligned} \tag{2.2}$$

However, this second formulation leaves a rotational ambiguity as for any orthogonal matrix $\mathbf{R} \in \mathbb{R}^{J \times J}$, $\mathbf{V}_J\mathbf{R}$ returns the same reconstruction error.

In practice, the mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ are estimated by the empirical mean and empirical covariance

$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_n \mathbf{y}_n \ ,$$
$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N}\sum_n (\mathbf{y}_n - \hat{\boldsymbol{\mu}})(\mathbf{y}_n - \hat{\boldsymbol{\mu}})^\top \ .$$

Note also that when D is larger than N, it is computationally more stable to compute the eigenvectors of the empirical covariance matrix from the *singular value decomposition* of $\mathbf{Y}_c^\top = [(\mathbf{y}_1 - \hat{\boldsymbol{\mu}}), \ldots, (\mathbf{y}_N - \hat{\boldsymbol{\mu}})]$.

2.1.3 Robustness to atypical data

Formally speaking, atypical data (also called *outliers*) are samples lying apart from the main clusters of observed samples. There are several sources from which might originate atypical data. The first source is the erroneous measures, either due to a malfunctioning of the recording gear, or due to a human error in the preparation of the samples (e.g. a sample was attributed an incorrect label), or any other source of error in the process of collecting the data. A second source of atypical data are the rare events. These correspond to perfectly correct measures but still they are similar with none of the other measures. The dissimilarity might be due to a higher noise contribution, or because some region of the space was not sufficiently covered by the collecting procedure.

Certainly, one should try to withdraw erroneous measures from the dataset before starting the analysis. Keeping atypical but still valid samples is more debatable (see discussion in Section 3.1.2). But in general, one cannot tell if a sample is an erroneous measure or not and there is no optimal procedure to tell beyond what limit a sample should be considered as an outlier.

Erroneous measures and rare events are not the only sources of atypical samples. Samples may become atypical also from the model perspective as a consequence of the mismatch between the model assumptions and the inherent patterns in the data. For example, a linear subspace model is not able to adequately represent data lying on a nonlinear manifold; also looking for two clusters when data are generated from more than two sources leads to samples lying far from the model assumptions. These examples are treated in Section 2.4.

Like many standard learning algorithms, Principal Component Analysis is very sensitive to atypical samples because it is optimising a squared error criterion in (2.2). In practice, it is common to have atypical samples $\mathbf{y}_+$ standing at a distance to their projection $\mathbf{V}\mathbf{x}_+$ three or four time superior to the average distance to projection. It means that these atypical samples will have a weight in the square error criterion equivalent to about 9 to 16 standard samples. The

consequence is that for small datasets the estimated model is highly biased toward the atypical samples.

This issue has been known to the Statistics community for a long time and many robust approaches have already been proposed (see [65, 88]). They are referred to as Robust Statistics. The base feature of many robust approaches is to replace the square error criterion with another training criterion that gives less weight to points lying far from the denser (often central) region. The mean absolute error, the Huber loss or the biweight are such examples of more robust training criteria. Other classical robust approaches proceed by iteratively withdrawing the most extreme samples until none of the remaining samples can be considered to be outliers anymore. Of course, there are drawbacks to these robust estimators. From the learning perspective, the robust criteria will often be responsible for an estimation bias (see previous chapter). Also, the training problem might loose its convexity and the optimisation algorithm will generally be more complicated than fitting the model on the square error criterion.

As we will see in Subsection 2.2.2, the probabilistic procedure to handle outliers and deviations from model assumptions is to model the dispersion of samples by a heavy tailed distribution like the Student-t. The main advantage of the probabilistic framework is to formulate the learning problem in a simple principled way which generally requires fewer "heuristic" choices from the practitioner (like choosing the hyper-parameters of a robust training criterion). Also, we will see that the framework has some nice modularity properties.

2.2 Probabilistic PCA and extensions

In this section, the probabilistic approach to identify linear subspaces is presented. The advantage of the probabilistic formulation is that "building blocks" can be combined quite naturally to extend a simple model in a more complex hierarchical one. The original building block here is the *Probabilistic PCA* (PPCA) model presented in the first subsection. The robust and mixture of robust Probabilistic PCA extensions are then introduced. We finish the section with an overview of related extensions.

2.2.1 Probabilistic PCA

As introduced in the preliminaries, PCA can be formulated as the search for an optimal linear projection minimising a reconstruction error. The principal components are derived from the observations by projecting them on the principal components. In the probabilistic formulation, the view is inverted in the sense that now the observations $\{\mathbf{y}_n\}_{n=1}^N$, where $\mathbf{y}_n \in \mathbb{R}^D$, are assumed to be generated from a low dimension latent representation $\{\mathbf{x}_n\}_{n=1}^N$, where $\mathbf{x}_n \in \mathbb{R}^J$, with $J < D$. The link between the latent vectors and the observations is

$$\mathbf{y}_n = \mathbf{W}\mathbf{x}_n + \boldsymbol{\mu} + \boldsymbol{\epsilon}_n \tag{2.3}$$

where $\mathbf{W} \in \mathbb{R}^{D \times J}$ encodes the linear dependency between latent and observed spaces, $\boldsymbol{\mu}$ denotes an offset to the origin of the space and $\{\boldsymbol{\epsilon}_n\}_{n=1}^N$, $\boldsymbol{\epsilon}_n \in \mathbb{R}^D$ are additive noise vectors. Clearly, if we further assume that the noise is generated from an isotropic Gaussian distribution $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \tau^{-1}\mathbf{I}_D)$ where τ^{-1} is the noise variance, and constrain the matrix $\mathbf{W}$ to be orthogonal, we get the same learning criterion as (2.2). After training, $\mathbf{W}$ spans the same subspace as the principal components $\mathbf{V}_J$ (assuming the same offset $\boldsymbol{\mu}$ is used for both).

There is however another way to proceed which is the Bayesian framework. As explained in 1.3, the principle is to express uncertainty about (some of) the parameters of the model by prior distributions. In particular, one could attribute a centred isotropic Gaussian multivariate prior $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_J)$ to the latent vectors. The centre and variance of this Gaussian prior have no importance because they can be compensated by $\boldsymbol{\mu}$ and a multiplicative factor in $\mathbf{W}$ respectively. Choosing an isotropic prior is intuitively justified as there is no reason to prefer *a priori* some direction in the latent space over any other. On the other hand, working with a Gaussian distribution is essential for the mathematical convenience explained below. The hierarchical model is then

$$P(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{I}_J) \ , \tag{2.4}$$

$$P(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{W}\mathbf{x} + \boldsymbol{\mu}, \tau^{-1}\mathbf{I}_D) \ . \tag{2.5}$$

To generate samples from this model, one should first generate latent vectors from (2.4) and then pseudo-observations from (2.5). Note that to avoid cluttered notations, we do not write explicitly the presence of parameters ($\mathbf{W}, \boldsymbol{\mu}, \tau$ for this model) necessary to define the model distributions.

For this simple model, the prior distribution is the conjugate of the likelihood (which is the reason why this distribution was chosen). Hence, the marginal distribution is in closed form

$$P(\mathbf{y}) = \int_{\mathcal{X}} P(\mathbf{y}|\mathbf{x})\, P(\mathbf{x})\, d\mathbf{x} = \mathcal{N}(\mathbf{y}|\boldsymbol{\mu}, \Sigma) \tag{2.6}$$

where the marginal covariance is defined by $\boldsymbol{\Sigma} \equiv \mathbf{W}\mathbf{W}^\top + \tau^{-1}\mathbf{I}_D$. To train such a probabilistic model, the standard procedure is to maximise the marginal[1] likelihood (2.6) with respect to the parameters. Equivalently, to formulate the training problem by means of a training loss function $\mathcal{L}_\mathcal{E}$, one should minimise the negative log-likelihood,

$$\begin{aligned} \boldsymbol{\theta}_{\mathrm{ML}} &= \operatorname*{argmin}_\theta \mathcal{L}_\mathcal{E}(\mathcal{D}, \boldsymbol{\theta}) = \operatorname*{argmin}_\theta \ -\sum_n \log P(\mathbf{y}_n) \\ &= \operatorname*{argmin}_\theta \left\{ \sum_n \frac{\tau}{2} (\mathbf{y}_n - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{y}_n - \boldsymbol{\mu}) + \frac{N}{2} \log 2\pi\tau^{-1} \right\} \end{aligned} \tag{2.7}$$

[1] It is frequent to drop the "marginal" term and call is simply the likelihood of the parameters.

where we noted the set of parameters $\boldsymbol{\theta} \equiv \{\mathbf{W}, \boldsymbol{\mu}, \tau\}$. From this derivation, we see that the learning procedure is equivalent to fitting a multivariate Gaussian distribution to the observations with a constrained covariance matrix $\boldsymbol{\Sigma}$. It is interesting to work with the factorisation (2.4)-(2.5) in order to derive an Expectation-Maximisation (EM) algorithm for the estimation of the parameters (see Section 2.3).

This model is called *Probabilistic PCA* (PPCA) [117, 137]. Tipping and Bishop showed that maximising the criterion (2.7) leads to a solution $\mathbf{W}_{\text{ML}}$ that is equivalent to the principal components found by the PCA up to a scaling and rotation

$$\mathbf{W}_{\text{ML}} = \mathbf{V}_J(\mathbf{P}_J - \tau^{-1}\mathbf{I}_J)^{1/2}\mathbf{R}\,, \tag{2.8}$$

where $\mathbf{V}_J$ was defined above, $\mathbf{P}_J \in \mathbb{R}^{J\times J}$ is the diagonal matrix with the J dominant eigenvalues and $\mathbf{R} \in \mathbb{R}^{J\times J}$ is an orthogonal matrix. Consequently, $\mathbf{W}_{\text{ML}}$ spans the same subspace as $\mathbf{V}_J$. It was further shown that the noise variance τ^{-1} of this model is equal to the average variance lost from the discarded components. The rotational ambiguity expressed by $\mathbf{R}$ can be easily removed by a post-processing step (again see [137]). In any case, one does not always need to compute the principal components explicitly as the principal subspace is sufficient for many analyses.

Probabilistic PCA suffers from the fact that it is based on the Gaussian noise model. Like the standard PCA, the contribution to the likelihood (2.7) is quadratic with respect to the Mahalanobis[2] distance to the mean $\boldsymbol{\mu}$. As a result the estimation of the parameters is still very sensitive to atypical observations and data that are not well confined to a low-dimensional linear subspace.

2.2.2 Robust Probabilistic PCA

Robust Probabilistic PCA [5] is an extension of PPCA making it more applicable to datasets containing atypical samples. Instead of the Gaussian noise assumption, the randomness in the observations is modelled by a Student-t distribution (Appendix A). Compared to the Gaussian, the Student-t has an additional parameter ν called the *number of degrees of freedom*, which regulates the thickness of the distribution's tail. In Figure 2.1(a) a Gaussian and a Student-t distributions ($\nu = 2$) are compared. Both distributions have unit variance. We see that the Student-t is a heavy tailed generalisation of the Gaussian distribution. The Student-t becomes very similar to the Gaussian when $\nu > 50$. Here, we will not restrict ν to be an integer value.

Figure 2.1(b) shows the corresponding negative-log-likelihood which appears in the training criterion of probabilistic models. We see that when ν is small, the Student-t attributes a much smaller cost than the Gaussian to points lying far from the mean of the distribution. The sensitivity to atypical observations is therefore reduced.

[2]The Mahalanobis distance is the Euclidean distance weighted by the inverse covariance $\|\mathbf{y} - \boldsymbol{\mu}\|^2_{\boldsymbol{\Sigma}^{-1}} = (\mathbf{y}_n - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{y}_n - \boldsymbol{\mu})$

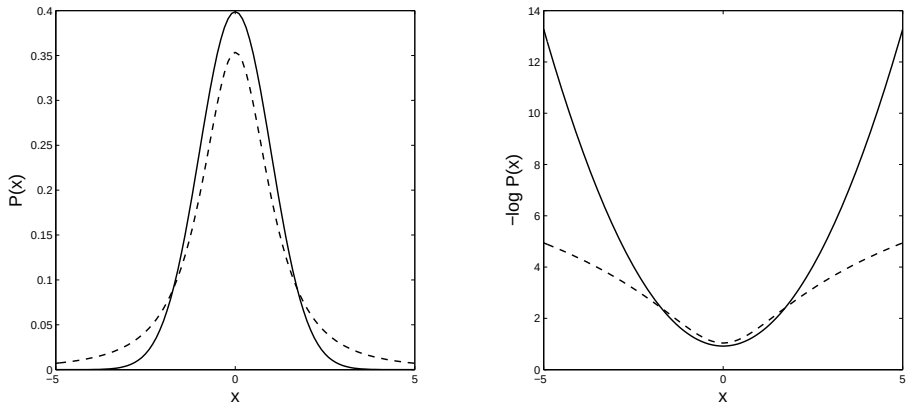


Figure 2.1: (a) Probability density functions of a Gaussian ($-$) and a Student-t with $\nu = 2$ ($--$). (b) Negative log-likelihood of these same distributions.

Let us turn to the model formulation. We make the assumption that atypical samples might come either from the generation of latent vectors $\mathbf{x}$ or from the noise contribution. This is expressed by Student-t distributions on the prior of the latent vectors and on the conditional distribution of observations

$$P(\mathbf{x}) = \mathcal{S}t(\mathbf{x}|\mathbf{0}, \mathbf{I}_J, \nu) \ , \tag{2.9}$$

$$P(\mathbf{y}|\mathbf{x}) = \mathcal{S}t(\mathbf{y}|\mathbf{W}\mathbf{x} + \boldsymbol{\mu}, \tau^{-1}\mathbf{I}_D, \nu) \ . \tag{2.10}$$

To simplify the parameterisation, both distributions are given the same degree of freedom ν. We will comment this choice later.

In order to work with the Student-t distribution in the probabilistic model and to be able to derive an EM algorithm for updating its parameters, it is necessary to work with a factorisation of the distribution [77]. The Student-t distribution can be reformulated as an infinite mixture of Gaussian distributions

$$\mathcal{S}t(\mathbf{y}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \nu) = \int_0^\infty \mathcal{N}(\mathbf{y}|\boldsymbol{\mu}, \frac{1}{u}\boldsymbol{\Sigma}) \ \mathcal{G}a(u|\tfrac{\nu}{2}, \tfrac{\nu}{2}) \ du, \quad \nu > 0 \ , \tag{2.11}$$

where $\mathcal{G}a(u|\cdot,\cdot)$ is a Gamma distribution (Appendix A) over the precision factor u.

Making use of this factorisation, the generative model can be represented with an additional level in the hierarchy where the latent precision u appears:

$$P(u) = \mathcal{G}a(u|\tfrac{\nu}{2}, \tfrac{\nu}{2}) \ , \tag{2.12}$$

$$P(\mathbf{x}|u) = \mathcal{N}(\mathbf{x}|\mathbf{0}, \frac{1}{u}\mathbf{I}_J) \ , \tag{2.13}$$

$$P(\mathbf{y}|\mathbf{x}, u) = \mathcal{N}(\mathbf{y}|\mathbf{W}\mathbf{x} + \boldsymbol{\mu}, \frac{1}{u\tau}\mathbf{I}_D) \ . \tag{2.14}$$

From this generative formulation, we see that the uncertainty about the observation (i.e. expressed by the variance in (2.14)) can be amplified by a small latent precision variable u. Note also that the latent precision u is shared by the conditional distributions of $\mathbf{x}$ and $\mathbf{y}$. The first reason to impose this constraint is to simplify the derivation of an EM algorithm. More fundamentally, this constraint implies that outliers in the observation space of $\mathbf{y}$ are also considered to be outliers in the latent space of $\mathbf{x}$ so their contributions to the identification of the latent space are down-weighted. The constraint is thus justified intuitively.

For robust PPCA, the marginal distribution of the observations is still tractable

$$\begin{aligned} P(\mathbf{y}) &= \int_0^\infty \int_{\mathcal{X}} P(\mathbf{y}|\mathbf{x}, u)\, P(\mathbf{x}|u)\, P(u)\, d\mathbf{x}\, du \\ &= \mathcal{S}t(\mathbf{y}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \nu) \end{aligned} \tag{2.15}$$

where again $\boldsymbol{\Sigma} \equiv \mathbf{W}\mathbf{W}^\top + \tau^{-1}\mathbf{I}_D$. Equivalently to (2.7) the training procedure consists in maximising this (marginal) likelihood with respect to the parameters $\boldsymbol{\theta} \equiv (\mathbf{W}, \boldsymbol{\mu}, \tau, \nu)$.

In contrast with previous robust approaches to the PCA (see for example [148] and [64], and the references therein), the probabilistic formalism has a number of important advantages. First, it only requires us to select the dimension of the projection space (see section 2.3.2), the other parameters being estimated using the maximum likelihood criterion. Previous attempts needed in general to select several additional parameters, from heuristics or with a model selection procedure. Second, the probabilistic approach provides a natural framework to extend the model to mixtures as shown in the next section. Third, a probabilistic model provides posterior estimates for the latent states. These posterior probabilities can be useful to analyse the generative model and get some intuition about the dependencies present in the data. Non-probabilistic approaches would need to set an ad hoc posterior analysis to get the same information.

2.2.3 Mixture of robust Probabilistic PCA

The robust PPCA model is not adequate for representing clusters of samples or nonlinear dependencies in the data. The mixture of PPCA [136] is a step in this direction. But again, the sensitivity of this model to atypical samples limits its use on many real world datasets. It is thus natural to look for a robust formulation of the mixture of PPCA [6].

The probability distribution of a sample generated from a mixture of K robust PPCA is defined as follows:

$$P(\mathbf{y}) = \textstyle\sum_k \pi_k P_k(\mathbf{y}) \tag{2.16}$$

where $\{\pi_k\}_{k=1}^K$ is the set of mixture proportions, with $\sum_k \pi_k = 1$ and $\pi_k \geq 0$ for all k; the $P_k(\mathbf{y})$ are defined as single robust PPCA components (2.15)

$$P_k(\mathbf{y}) = \mathcal{S}t(\mathbf{y}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \nu_k) \tag{2.17}$$

in which $\boldsymbol{\Sigma}_k \equiv \mathbf{W}_k\mathbf{W}_k^\top + \tau_k^{-1}\mathbf{I}_D$. The set of parameters of this model is $\boldsymbol{\theta} \equiv \{(\mathbf{W}_k, \boldsymbol{\mu}_k, \tau_k, \nu_k, \pi_k)\}_{k=1}^K$.

As previously mentioned, to simplify the derivation of an EM algorithm, it is necessary to factorise the probabilistic model as much as possible. Here, the additional level in the hierarchy brought by the mixture formulation can be factorised by means of a latent indicator variable $\mathbf{z} = [z_1, \ldots, z_K]$ with $z_k = 1$ if the kth component generated the observation $\mathbf{y}$, otherwise $z_k = 0$. The factorised mixture of robust PPCA is then

$$P(\mathbf{z}) = \prod_k \pi_k^{z_k}, \tag{2.18}$$

$$P(\mathbf{u}|\mathbf{z}) = \prod_k \mathcal{G}a(u_k|\tfrac{\nu_k}{2}, \tfrac{\nu_k}{2})^{z_k}, \tag{2.19}$$

$$P(\boldsymbol{\chi}|\mathbf{u}, \mathbf{z}) = \prod_k \mathcal{N}(\mathbf{x}_k|\mathbf{0}, \tfrac{1}{u_k}\mathbf{I}_J)^{z_k}, \tag{2.20}$$

$$P(\mathbf{y}|\boldsymbol{\chi}, \mathbf{u}, \mathbf{z}) = \prod_k \mathcal{N}(\mathbf{y}|\mathbf{W}_k\mathbf{x}_k + \boldsymbol{\mu}_k, \tfrac{1}{u_k\tau_k}\mathbf{I}_D)^{z_k}, \tag{2.21}$$

where $\mathbf{u} = [u_1, \ldots, u_K]$ and $\boldsymbol{\chi} = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$. We will see in Section 2.3.1 how to estimate the parameters of this model. Note that the different components could also have different latent dimensionalities $\{J_k\}_{k=1}^K$.

Increasing the robustness by replacing Gaussian densities with Student-t densities was also proposed in the context of finite mixture modelling [105, 3]. The main advantage over the finite mixtures of Gaussian or Student-t densities resides in the fact that the full-rank, possibly ill-conditioned covariance matrices are approximated by constrained covariance matrices $\boldsymbol{\Sigma}_k$. The number of free parameters per component depends on the dimension of the latent subspace. For each constrained covariance matrix this number of parameters is equal to $DJ_k + 1 - J_k(J_k - 1)/2$ (where the last term takes the rotational invariance (2.8) into account). In comparison, a full covariance matrix requires the estimation of $D(D+1)/2$ parameters. As this number increases rapidly with the space dimensionality, it is common to constrain the covariance to be diagonal, but this leads to axis aligned components which are generally suboptimal [3]. The latent space formulation of the (mixture of robust) PPCA take into account the dominant correlations in the data while keeping the number of free parameters relatively low.

2.2.4 Other extensions

Working with a probabilistic formulation, it is easy to think of natural extensions to a particular model. Our work has concentrated on robust extensions to Probabilistic PCA and mixture of Probabilistic PCA. Other extensions were proposed focusing on different aspects of the model. The linear dependency between a latent representation and the observations can be useful in contexts other than PCA. In [10], the principle is applied to derive a probabilistic *Canonical Correlation Analysis*. A robust extension of to this model was also proposed in [5]. PCA and probabilistic PCA are closely related to *Factor Analysis* [43] and maximum likelihood Factor Analysis [118], which can also be combined to form mixtures [57, 47]. The main difference with the PCA models is to account

for different variances of noise for each variable in the observations. For factor analysis, the noise vectors in (2.3) are assumed to come from $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Psi})$, where $\boldsymbol{\Psi} = \text{diag}\left\{[\tau_1^{-1}, \ldots, \tau_D^{-1}]\right\}$ is the diagonal matrix of variable noise variances. Another extension is to force the components of the mixture to respect some alignment constraints [7]. In a similar spirit, the *Generative Topographic Mapping* [18] makes use of a discrete latent space (in general by defining a one or two-dimensional grid) and a nonlinear basis expansion (see Chapter 3) to match the latent space with a nonlinear manifold in the observation space. The resulting model is very similar to the Self-Organising Maps with the difference of being formulated as a density estimation task. In [75], a dual approach is suggested where instead of fixing the latent representation and optimising the dependency to the observations, the dependency is set by a (nonlinear) Gaussian Process model (see Section 3.2.4) and the latent representation is optimised.

2.3 Learning procedure

A major difficulty in extending probabilistic models lies in the derivation of an efficient optimisation algorithm. Indeed, it is not possible in general to analytically compute the marginal likelihood of more complex models. One then has to resort to approximation methods, like variational EM algorithms, or the *Expectation Propagation* [91] procedure.

This is not an issue here as the marginal of the mixture of robust probabilistic PCA is still tractable (2.16). Moreover, the factorisation of the model (2.18)-(2.21) allows us to derive an exact Expectation-Maximisation algorithm, which is presented in the following subsection. Note that this algorithm encompasses the optimisation of the (mixture of) probabilistic PCA. One only needs to add the constraint $\nu_k = \infty$ (for all k) such that the Student-ts are in fact Gaussian distributions.

In the second part of the section, we give general comments on the model selection procedure for setting the size of the latent spaces and the number of components.

2.3.1 Expectation-Maximisation algorithm

We seek an optimum of the model's likelihood (2.16) to estimate the parameters $\boldsymbol{\theta} \equiv \{(\mathbf{W}_k, \boldsymbol{\mu}_k, \tau_k, \nu_k, \pi_k)\}_{k=1}^K$. The simplest way to proceed is by deriving an EM algorithm [39](Section 1.3.3) on the factorised distribution (2.18)-(2.21). The starting point of the algorithm is to bound the marginal likelihood (making use of the Jensen's inequality):

$$\log P(\{\mathbf{y}_n\}) \geq \mathbb{E}_Q \left\{\log P(\{\mathbf{y}_n\}, \{\boldsymbol{\chi}_n\}, \{\mathbf{u}_n\}, \{\mathbf{z}_n\})\right\} - \mathbb{E}_Q \left\{\log Q\left(\{\boldsymbol{\chi}_n\}, \{\mathbf{u}_n\}, \{\mathbf{z}_n\}\right)\right\} . \quad (2.22)$$

The first term on the right hand side is the expected log-complete likelihood (sometimes called the free energy). The expression of this log-complete likelihood for the model is given in Appendix C.1. From Section 1.3.3 again, we know that the bound is tight when the distribution over the latent variable $Q(\{\boldsymbol{\chi}_n\}, \{\mathbf{u}_n\}, \{\mathbf{z}_n\})$ coincides with the posterior distribution. Fortunately, the posterior distribution of the mixture of robust PPCA model is still tractable. Indeed, applying the Bayes formula, one can show that the posterior is

$$P(\{\boldsymbol{\chi}_n\}, \{\mathbf{u}_n\}, \{\mathbf{z}_n\}|\{\mathbf{y}_n\}) = \prod_n \prod_k P(\mathbf{x}_{nk}|u_{nk}, z_{nk}=1, \mathbf{y}_n) \cdot P(u_{nk}|z_{nk}=1, \mathbf{y}_n) P(z_{nk}=1|\mathbf{y}_n) \;, \quad (2.23)$$

where the factor distributions are

$$P(\mathbf{x}_{nk}|u_{nk}, z_{nk}=1, \mathbf{y}_n) = \mathcal{N}(\mathbf{x}_{nk}|\tau_k \mathbf{C}_k \mathbf{W}_k^\top (\mathbf{y}_n - \boldsymbol{\mu}_k), \frac{1}{u_{nk}} \mathbf{C}_k) \;, \quad (2.24)$$

$$P(u_{nk}|z_{nk}=1, \mathbf{y}_n) = \mathcal{G}a(u_{nk}|\alpha_k, \beta_{nk}) \;, \quad (2.25)$$

$$P(z_{nk}=1|\mathbf{y}_n) = \frac{\pi_k \mathcal{S}t(\mathbf{y}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \nu_k)}{\sum_k \pi_k \mathcal{S}t(\mathbf{y}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \nu_k)} \;, \quad (2.26)$$

and where we have defined

$$\mathbf{C}_k^{-1} = \tau_k \mathbf{W}_k^\top \mathbf{W}_k + \mathbf{I}_J \;, \quad (2.27)$$

$$\alpha_k = \frac{D + \nu_k}{2} \;, \quad (2.28)$$

$$\beta_{nk} = \frac{(\mathbf{y}_n - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} (\mathbf{y}_n - \boldsymbol{\mu}_k) + \nu_k}{2} \;. \quad (2.29)$$

Notice that there is only a single observation $\mathbf{y}_n$ appearing in each of these posterior factor distributions.

The *E-step* corresponds to computing the expectation of the log-complete likelihood in (2.22) (see Appendix) when $Q(\cdot)$ is set to the current posterior distribution (i.e. with the current values of the parameters $\boldsymbol{\theta}$). This computation boils down to a set of expectations

$$\bar{\rho}_{nk} \equiv \mathbb{E}_Q\{z_{nk}\} = \frac{\pi_k \mathcal{S}t(\mathbf{y}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \nu_k)}{\sum_k \pi_k \mathcal{S}t(\mathbf{y}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \nu_k)} \;, \quad (2.30)$$

$$\bar{u}_{nk} \equiv \mathbb{E}_Q\{u_{nk}\} = \frac{\alpha_k}{\beta_{nk}}, \;, \quad (2.31)$$

$$\log \tilde{u}_{nk} \equiv \mathbb{E}_Q\{\log u_{nk}\} = \psi(\alpha_k) - \log(\beta_{nk}) \;, \quad (2.32)$$

$$\bar{\mathbf{x}}_{nk} \equiv \mathbb{E}_Q\{\mathbf{x}_{nk}\} = \tau_k \mathbf{C}_k \mathbf{W}_k^\top (\mathbf{y}_n - \boldsymbol{\mu}_k) \;, \quad (2.33)$$

$$\bar{\mathbf{S}}_{nk} \equiv \mathbb{E}_Q\{\mathbf{x}_{nk}\mathbf{x}_{nk}^\top\} = \frac{1}{u_{nk}} \mathbf{C}_k + \bar{\mathbf{x}}_{nk} \bar{\mathbf{x}}_{nk}^\top \;, \quad (2.34)$$

where $\bar{\omega}_{nk} \equiv \bar{\rho}_{nk} \bar{u}_{nk}$ and $\psi(\cdot) \equiv \Gamma'(\cdot)/\Gamma(\cdot)$ is called the *digamma* function.

Maximising (2.22) with respect to the parameters (*M-step)* leads to the following updates

$$\pi_k \leftarrow \tfrac{1}{N}\sum_n \bar{\rho}_{nk} \;, \tag{2.35}$$

$$\boldsymbol{\mu}_k \leftarrow \tfrac{\sum_n \bar{\omega}_{nk}(\mathbf{y}_n - \mathbf{W}_k\bar{\mathbf{x}}_{nk})}{\sum_n \bar{\omega}_{nk}} \;, \tag{2.36}$$

$$\mathbf{W}_k \leftarrow \left(\sum_n \bar{\omega}_{nk}(\mathbf{y}_n - \boldsymbol{\mu}_k)\bar{\mathbf{x}}_{nk}^\top\right)\left(\sum_n \bar{\omega}_{nk}\bar{\mathbf{S}}_{nk}\right)^{-1} \;, \tag{2.37}$$

$$\tau_k^{-1} \leftarrow \tfrac{1}{DN\pi_k}\sum_n \bar{\omega}_{nk}\left(\|\mathbf{y}_n - \boldsymbol{\mu}_k\|^2 - 2(\mathbf{y}_n - \boldsymbol{\mu}_k)^\top \mathbf{W}_k\bar{\mathbf{x}}_{nk} + \mathrm{tr}\{\mathbf{W}_k\bar{\mathbf{S}}_{nk}\mathbf{W}_k^\top\}\right) \tag{2.38}$$

for all k and where $\mathrm{tr}\{\cdot\}$ is the trace operator. Observe how the contribution of each data point is weighted according to $\bar{\omega}_{nk}$, which accounts for both the effect of the responsibilities $\bar{\rho}_{nk}$ and the expected latent precision variables $\bar{u}_{nk}$. The latter ensures robustness as its value is small for $\mathbf{y}_n$ lying far from $\boldsymbol{\mu}_k$, such that the contribution in the M-step is small. For the non robust formulation ($\nu_k \to \infty$) we have $\bar{u}_{nk} = 1$ for all n and all k. Note also that these updates are coupled. One could cycle through these updates between each E-step until the M-step has converged.

There is no closed form update for $\{\nu_k\}_{k=1}^K$. Nevertheless, a solution can be computed at each EM iteration by solving the following expression by a line search algorithm (see for example [99]):

$$1 + \log\left(\tfrac{\nu_k}{2}\right) - \psi\left(\tfrac{\nu_k}{2}\right) + \tfrac{1}{N\pi_k}\sum_n \bar{\rho}_{nk}\{\log\tilde{u}_{nk} - \bar{u}_{nk}\} = 0 \tag{2.39}$$

for all k. Alternatively, a heuristic was proposed by Shoham [124] in the context of mixture modelling. This heuristic is also applicable here.

As the marginal likelihood of mixture models always exhibits local optima, it is recommended that the optimisation is repeated with different initialisations. This should avoid converging to a very bad local optimum for which the training criterion is still far from the global optimum. A good strategy to initialise the components is to set the centres $\boldsymbol{\mu}_k$ using a quantisation algorithm and then initialise the subspace orientation $\mathbf{W}_k$ from the first principal components of the samples in the Voronoi region of $\boldsymbol{\mu}_k$.

2.3.2 Model selection

At this stage, one might wonder how to choose the model configuration, i.e. how to select the number of components and the dimensionalities of the latent representations. As was discussed in the preliminaries of the chapter, the model is primarily an exploratory tool. In this sense, we should not focus solely on the selection procedure itself but rather find out how the information contained in the data can be usefully uncovered. Visualisation is the paramount aim of exploratory analysis and for this reason one and two-dimensional subspaces are the standard choices. Of course, not all datasets can be correctly represented by one or two-dimensional representations. Fixing the number of components

is also tricky. The number of components should ideally match the number of distinct sources, provided these distinct sources exist. In practice, it is not easy to avoid overlap between several components, due to the mismatch between the linear subspace representation and the data or due to local minima in the training criterion. Also, there is no guarantee that the patterns detected by visual inspection are inherent or just artefacts. However, if the goal at the initial stage is to identify the dominant sources, one will usually work with a small number of components which should limit the risk of over-fitting the data (there will rather be a under-fit of the data).

Certainly a more satisfactory approach is to combine the exploratory analysis with a quantitative model selection procedure. The advantage is that the model construction can be made more automatic and it should prevent the practitioner for exploring configurations which do not correspond well to the data. Obviously, resampling methods are applicable to select the configuration. The principle is always the same: a set of observations is held-out of the training, and the quality of the configuration is estimated after training on these validation observations. But this approach rapidly becomes too costly when there are several (hyper-)parameters to be selected.

In the literature, there are proposals to select the best configuration with a Bayesian approach. An additional hierarchy can be added to express the prior uncertainty over the number of components and the dimensionalities of the latent spaces. However, it is no longer possible to adjust the model with an exact EM algorithm. In [114], the marginalisation over models with different number of components is carried out with (MCMC) sampling techniques. The selection is treated with a (mean field) variational approximation in [145, 9]. Selecting the dimensionality of the latent subspaces was proposed in [16] by means of an Automatic Relevance Determination (ARD) scheme (see Section 3.1.4).

2.4 Experiments

In this section, we start by illustrating the construction of the probabilistic linear subspaces models on three toy examples. We then show on the USPS handwritten digit dataset that the procedure is still applicable and useful in higher dimensional spaces.

2.4.1 Toy examples

We start with a very simple example showing the sensitivity of the probabilistic PCA to outliers. In Figure 2.2, 25 samples are generated along a line and two outliers are added. A PPCA and robust PPCA with a one-dimensional latent space are constructed on these samples. We clearly see the influence of the two outliers on the principal component found by the PPCA model, while the robust PPCA component is almost unaffected by these outliers.

Figures 2.3(a)-(b) shows another illustration of the different behaviours.

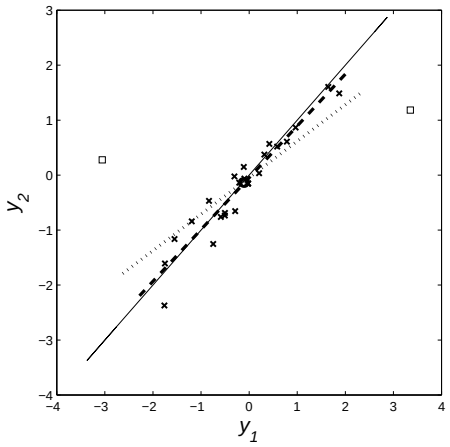


Figure 2.2: Samples generated along the solid line, except two outliers (squares). Principal components from probabilistic PCA ($\cdot\cdot$) and robust probabilistic PCA ($--$). Circles correspond to the estimated means $\boldsymbol{\mu}$.

The samples have been generated along a one-dimensional manifold, with higher density of samples in the right end and higher noise at the other end. PPCA is trying to account for all the observed data and thus estimate a global principal component. Note that the mean of the component lies in an empty region. It does not correspond to a typical sample of the set. On the other hand, robust PPCA discards many samples in order to concentrate on the higher density region of the manifold. The latent subspace identifies the local principal component.

Using three components in the model (Figures 2.3(c)-(d)), both the mixture of PPCAs and robust PPCAs estimate the local principal components of the manifold quite well. Note however that one of the components of the mixture of PPCAs tries to account for the noisy samples at the end, forcing its mean to move away from the manifold.

The last toy example is composed of 3 clusters (see Figure 2.4(a)). Each cluster corresponds to a 3-dimensional Gaussian distribution with a diagonal covariance matrix equal to $\text{diag}\{[5, 1, 0.2]\}$ before rotation around the second coordinate axis. Each component lies on an intrinsic two dimensional space as the variance in the third direction is significantly smaller. The two outer clusters make an angle of ± 30 degrees with the middle one and they are respectively shifted by ± 5 units along the axis of rotation.

For the first experiment, 30 points are generated for each cluster and we train mixtures of standard and robust PPCA models with a varying number of components K and latent space dimensions J. The generalisation performances are then evaluated based on the negative log likelihood criterion of the model on a large validation set (i.e. generated independently from the training

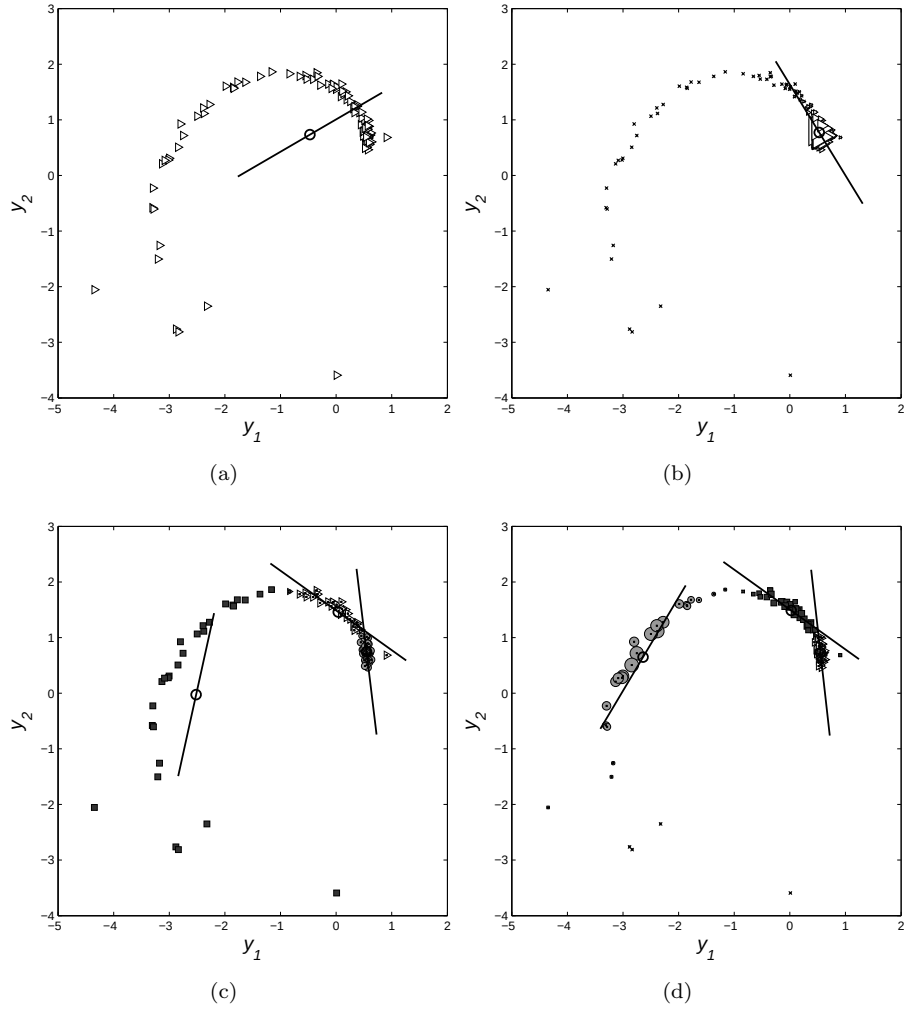
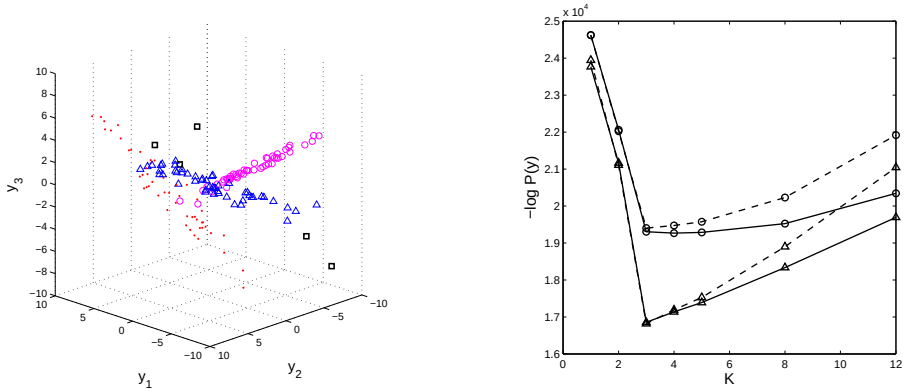


Figure 2.3: Samples generated along a 1-dimensional manifold with additional atypical points. (a) Probabilistic PCA, (b) robust PPCA, (c) 3 components mixture of PPCA, (d) 3 components mixture of robust PPCA. The sizes of the markers account for the contributions of the samples to the estimation of the component parameters.

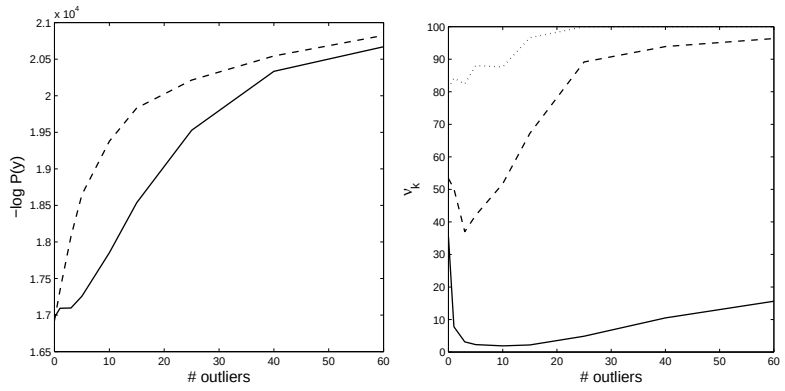


(a) (b)

Figure 2.4: (a) Synthetic example composed of three clusters which are generated from Gaussian distributions. See description in text. The squares represent outliers. (b) Dependency of the negative log likelihood on a validation set with respect to the number of components K and the dimensionality of the latent space ($\circ$: $J = 1$, $\triangle$: $J = 2$). Dashed line: standard. Plain line: robust.

samples). Generating the training data and training the model is repeated 50 times to reduce statistical variance. The generalisation performances are plotted in Figure 2.4(b) for $K \in \{1, \ldots, 12\}$ and $J \in \{1, 2\}$. As expected, the true model with $K = 3$ and $J = 2$ presents the best generalisation performances. Interestingly, we see that the standard and robust mixture models have comparable performances when the model under-fits the data (i.e. K is smaller than the optimal value) while the robust mixture has the edge on the standard mixture when K increases. The tendency of over-fitting is thus slightly reduced with the robust formulation.

For the second experiment, K and J are set to their optimal value (3 and 2 respectively) and we look at the sensitivity of the model construction to the number of outliers. The outliers are generated from a uniform distribution in the $[-10, 10]^3$ box which surrounds the three clusters. Again, 30 points are generated from each component and outliers (between 1 and 60) are added. The performances of the standard and robust mixture of PPCA models (measured on a validation set with no outliers from averaging 50 repetitions of generating the training samples and training the models) are shown on Figure 2.5(a). Again, we note the better behaviour of the robust formulation as the prediction performances are less sensitive to the number of outliers, especially when there are a small number of outliers. When the number of outliers increases to a significant proportion of the training samples (30 outliers correspond to 1/4 of the learning points) the robust formulation reduces its down-weighting of the outliers and consequently the gap between the performances of the standard



(a) (b)

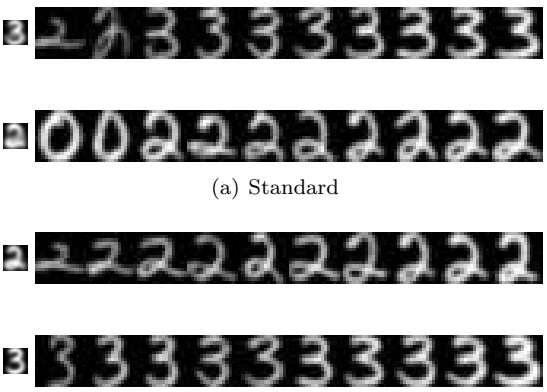
Figure 2.5: (a) Dependency of the generalisation performances (negative log likelihood criterion) with respect to the number of outliers added to the three main clusters. Dashed line: standard. Plain line: robust. (b) Dependency of the degree of freedom parameters for the three components in the robust mixture of PPCA model with respect to the number of outliers.

and robust models decreases. Figure 2.5(b) shows the average value of the degree of freedom parameters (ν_k for $k = 1 \dots 3$) for the same experiment. We note that the down-weighting of the outliers obtained with a small value of ν_k (see Subsection 2.3.1) comes mainly from a single component. The two other components have their degrees of freedom larger than 40 and are thus very close to Gaussian components.

2.4.2 USPS digits

We now illustrate the use of the models on the well-known USPS handwritten digit dataset[3]. Each sample is a greyscale 16×16 pixels image of a digit (0 to 9). To simplify the illustration, we kept only the images of digits 2 and 3 (respectively 731 and 658 images), as well as 100 (randomly chosen) images of digit 0. There are thus two dominant clusters of digits 2 and 3, and a smaller cluster of digit 0. We compared the mixture of PPCAs and the mixture of robust PPCAs in their ability to find the two main clusters and identify the main variability in these clusters with a one-dimensional latent space. The result is shown in Figure 2.6. Each row represents images close to the one-dimensional subspace. The mixture of robust PPCAs completely ignores the smaller cluster of digit 0. The first component concentrates on the digits 2 and the second on the digits 3. Interestingly, the model is able to discover that the main variability of digits 3 is along their width, while the main

[3]The USPS data were gathered at the Center of Excellence in Document Analysis and Recognition (CEDAR) at SUNY Buffalo during a project sponsored by the US Postal Service.



(a) Standard

(b) Robust

Figure 2.6: Mixture of 2 component PPCAs with 1-dimensional latent space to cluster USPS digit 2 and 3, and outliers digit 0. Top: standard; Bottom: robust.

variability of digits 2 is along their height. On the other hand, the mixture of PPCAs cannot down-weight the contribution of the digits 0 and we see that its second component makes the transition between digits 2 and digits 0. Its first component is also biased by atypical samples.

The behaviour of the (mixture of) PPCA model(s) is undesirable if we want each component to correspond to a single cluster. Of course, one could argue that three components would be a better choice in this case. But the problem is common, in particular within high-dimensional spaces: most of the density of points lies on a low-dimensional subspace (possibly different for each cluster of points) but not entirely as there are always samples diverting attention from the main densities.

Chapter 3

Regression applied to near-infrared spectroscopy

The work presented in this chapter originates from a preliminary study on the use of Radial Basis Function Networks [93, 107] to model functional data. The work was published in [36, 116]. For handling functional data, we followed the guidelines suggested in [108]. The experimental study of this work considered the Tecator near-infrared (NIR) spectroscopy dataset which was initially analysed with other neural networks models in [132]. The interesting feature of NIR spectroscopy data is that they are intrinsically smooth functions represented in practice by high dimensional vectors. This can be a problem for some models from a computational point of view when the cardinality of the set of parameters increases with the dimensionality of the representation. We will describe in section 3.3.1 how to handle the NIR spectra properly, and how regularisation constraints can be entered in the parameterisation.

From this preliminary work, our attention shifted to the general regression task and the different existing methods used to design regression models. In particular, the probabilistic approach appeared attractive as it allows consideration of flexible model configurations since the tendency to over-fit is automatically counteracted by the internal regularisation of the Bayesian selection scheme. Following the main theme of this thesis, the goal of this chapter is to show the usefulness of the probabilistic approach for tackling real data analysis tasks, and in particular for regression on the NIR spectroscopy application. In Section 3.4, we present a comparison of a set of regression models on 7 regression tasks. This makes the discussion of the performances in Section 3.5 more grounded than in many previous analyses.

Several aspects of the learning procedure are compared. Among these aspects, we are interested to know if the marginal likelihood criterion, which allows to adjust several hyper-parameters and avoids the necessity to repeat the learning procedure, is sufficiently robust to compete with the cross-validation

scheme on one or two hyper-parameters. A second aspect that is analysed is the processing of spectra by similarity (or distance) based computations. In particular, we compare the *Gaussian Process* model with a Gaussian covariance function (Section 3.2.4) with other approaches using a similar processing of the data. Always in this context, we sketch the lines in Section 3.2.2 of a procedure that we developped in [35] for adjusting the similarity measure in locally weighted models.

Finally, we derive in Section 3.3.3 a smoothed *Automatic Relevance Determination* scheme allowing us to identify the most relevant region of the spectra for the prediction task.

3.1 Preliminaries

In this section, we start with a reminder of the regression task and discuss the necessity of a good pre-processing of the data. Then, we give more details on the particular aspects of the model configurations and learning procedures that will be considered later in the comparison of the regression methods. Finally, we discuss the possibility of identifying some interpretation in parallel to the regression task. For this purpose, we introduce the concept of feature relevance.

3.1.1 Regression task

Let us review here the regression task introduced in Chapter 1. We are given a set of samples $\mathcal{D} = \{(\mathbf{x}_n, t_n)\}_{n=1}^N$, where $\mathbf{x}$ is an input sample and t is the corresponding target belonging to an ordered set (we will consider here only scalar targets, $t \in \mathbb{R}$). The input representation is discussed in the next subsection. The goal of the regression is to derive from these samples a prediction function $y = f(\mathbf{x})$ minimising a particular loss function $\mathcal{L}(t, y)$ in generalisation, i.e. on data independent from those used for learning. In practice, as we cannot collect new samples, the generalisation performances are evaluated by a resampling scheme, see Section 3.4 for details. In this chapter, we are interested in the comparison of regression methods (particularly on spectroscopy datasets) and have little knowledge about the conditions expected in the exploitation of the models. We have chosen to evaluate the performances of the different regression methods with the standard square error loss, $\mathcal{L}(t, y) = |t - y|^2$. One thing to bear in mind is that for a particular input $\mathbf{x}$ the best prediction $y^*(x)$ according to the squared error loss is the conditional mean of the target distribution $y^*(\mathbf{x}) = \mathbb{E}_{t|\mathbf{x}}\{t\}$. Hence, optimising a model on the squared error loss makes sense when the mean target is representative of the target distribution, i.e. this distribution is roughly unimodal and symmetric. From the analysis of results in the experiments below, the square error loss looks appropriate for the datasets.

Following the approach of the previous chapter, using a robust training loss $\mathcal{L}_\mathcal{E}$ (defined in Eq. (1.11)) like the Huber loss or a probabilistic model with larger

tail distribution than the Gaussian would make the training less sensitive to samples lying far from the mean target distribution. However, we have chosen to train the model on the empirical risk $\mathcal{L}_{\mathcal{E}} = \mathcal{L}_{\mathcal{D}}$ as the optimisation on this criterion is very simple for most of the methods described below. The identification of outliers is kept in the pre-processing stage. Note that for the square error loss, the empirical risk is also named the Mean Squared Error (MSE).

Note also that the different regression methods presented below could easily be applied to classification tasks. One just needs to work with loss functions ($\mathcal{L}$ and $\mathcal{L}_{\mathcal{E}}$) more appropriate for classification. We already gave the examples in Chapter 1 of the hinge loss and the conditional Bernoulli probabilistic model.

Note that a model configuration is characterised by a class of model spaces $(\mathcal{H}, \eta_{\mathcal{H}})$ and a class of training criteria $(\mathcal{E}, \eta_{\mathcal{E}})$. The model selection procedure tries to select the hyper-parameters $(\eta_{\mathcal{H}}, \eta_{\mathcal{E}})$ such that the model trained on the corresponding training configuration $(\mathcal{H}, \mathcal{E})$ generalises well. The different regression methods presented in Section 3.2 correspond to different *model configurations* and different model selection procedures, to use the same vocabulary. We describe in Section 3.1.3 the aspects of these model configurations on which our attention will concentrate for the comparison.

3.1.2 Data representation and pre-processing

We come back now to the question of the representation of input samples. The most common representation is to have D-dimensional vectors $\mathbf{x} \in \mathbb{R}^D$. Near Infrared spectra can be represented by vectors and for this reason we will almost exclusively discuss this representation in the models. The coordinates of input vectors are called the *features*. Actually, many other names are used in the literature depending on the research community; to cite just a few: *variables, predictors, regressors...*

We will see that regression methods require some standard mathematical operations be defined on the inputs; some methods are constructed based on the dot product in the space of the inputs, other are based on a distance measure, also the vector summation may be necessary. These standard operations are directly applicable to a vector representation. It happens however that some inputs are naturally represented with a non vectorial representation (strings, graphs, sets...). The strategy to apply a regression method to non-vector data is either to derive a vector representation during the pre-processing, or to define on these inputs the dot product or distance measure required by the regression method. This second approach is at the heart of the *kernel trick* presented in Section 3.2.4.

Before constructing a regression model and more generally before any data analysis task, it is a good habit to pre-process the data and make the following model construction less dependent on the original representation. We discuss the most common pre-processings here.

Standardised vector representation For vector data, the features often correspond to measurements expressed with different units. From a physical point of view, we should be careful in the way these features with different units are combined and avoid summing apples with pears! In addition, there is often no reason to prefer a particular unit's scale over another, e.g. the lengths are equally valid in meters and centimetres. Sometimes also the origin of the scale is completely arbitrary.

The usual way to reduce the arbitrary nature of the initial representation is to standardise the data. By this procedure, each feature is translated and scaled such that its empirical mean is set to 0 and its variance in the new representation is set to 1. Hence, the transformation for the dth feature of the new representation is

$$x_d^{(new)} = \frac{x_d - \hat{\mu}_d}{\hat{\sigma}_d} \tag{3.1}$$

where $\hat{\mu}_d$ is the empirical mean and $\hat{\sigma}_d$ is the empirical standard deviation in the original representation. It is also useful to apply the standardisation to the targets. This standardisation does not alleviate all criticisms in respect to the choice of units, neither is it based on strong arguments in favour of this particular choice of transformation rather than another one (scaling by the range is another common choice). Yet, in practice this simple pre-processing makes the model construction more systematic and *a priori* gives equal importance to all features.

For features constrained to be positive, converting them to a logarithm scale is sometimes useful. Often, it is advantageous to pre-process the features such that their empirical distribution is rather sub-Gaussian (i.e. with a short tail distribution) than super-Gaussian.

Principal Component Analysis We have reviewed Principal Component Analysis in the previous chapter. This method can also be applied as a pre-processing step in regression tasks. When the feature representation is high dimensional (more than 5 features is already moderately high), there are almost always strong approximate linear dependencies between the features, meaning that the intrinsic dimensionality of the inputs is lower than the original dimensionality D. Some of the regression methods described below have their set of parameters grow with the dimensionality of the inputs. For those methods, it is advantageous from a theoretical and a computational point of view to reduce the dimensionality of the inputs and work with fewer parameters. The usual strategy is to keep the dominant principal directions such that most of the variance from the original representation is still present in the transformed inputs.

Like standardisation, applying a PCA to the data and keeping only a few dominant principal directions is often favourable in subsequent learning. However, PCA is an unsupervised method, hence we have no guarantee that the new representation is better than the original one. Moreover, the new features

in the PCA representation are linear combinations of the original features. Getting some interpretation from these new features might be more difficult (see Subsection 3.1.4).

Outlier detection The issues regarding outliers have already been discussed in the previous chapter for unsupervised methods. In this chapter, we are more interested in the evaluation of performances; the presence of outliers in the datasets could substantially modify the analysis. It is thus necessary not only to spot the true outliers but also to leave them out of the dataset. By true outliers we mean those atypical samples probably coming from erroneous measures or bad preparations.

Handling the "gentle outliers", i.e. those samples lying just a bit apart from the main clusters, is more debatable. These samples might be perfectly correct measures. Leaving them in the datasets would put the methods under the test of extrapolating adequately the predictions. This can be considered as a desirable property of a method suggesting that we should keep these gentle outliers. However, we are not particularly focusing on the extrapolation performances of the different regression methods here. Besides, as there will probably be only just a small number of such samples, we will not be able to assess with any confidence the extrapolation ability of the model. This is an argument to say that even atypical but sane samples should not be considered for learning, nor for evaluation.

Anyway, we have in general very little knowledge of the data collecting process, giving us no support to identify systematically true and gentle outliers. It is very difficult to evaluate the limit beyond which a sample should be considered as an outlier. Besides, this is really dependent on the application and the dataset. Our purpose here is to apply a standard outlier detection technique. The technique is discussed in Section 3.4.3.

(Semi-)expert pre-processing A generally applicable piece of advice about data modelling and learning is to make as much use as possible of prior knowledge and intuition about the data to be analysed. At the pre-processing stage, one should try to represent the data such that their relevant aspects for the prediction task stand out; equally the irrelevant aspects should be de-emphasised by the pre-processing applied. For example, for an optical character recognition task, we know that the slew of handwritten characters varies a lot between people. The pre-processing should try to obtain a standard slew for all people's writing before applying the recognition algorithm. This would reduce a source of variability that is irrelevant for the prediction task.

In fact, expert knowledge could equivalently be inserted in the regulariser (or the prior distribution). We will make a slight distinction between the pre-processing and the regulariser approaches: the pre-processing is considered to be fixed during the learning while the regulariser can be adjusted with some hyper-parameters and a model selection procedure.

3.1.3 Various aspects of the comparison

The comparison of regression methods will focus on three aspects. We do not give them by order of importance here, but to make their description simpler.

The first aspect concerns models based on a similarity measure between inputs and in particular the Euclidean distance between input vectors:

$$\mathfrak{D}_e(\mathbf{x}, \mathbf{x}')^2 \equiv \|\mathbf{x} - \mathbf{x}'\|^2 = (\mathbf{x} - \mathbf{x}')^\top (\mathbf{x} - \mathbf{x}') \; . \tag{3.2}$$

We will see that the use of the Euclidean distance in the processing of the inputs appears in three of the regression methods discussed below. The processing is further completed by a nonlinear transformation of this distance. We have chosen to work with the widely used *radial squared exponential function*

$$\mathcal{G}(\mathbf{x}, \mathbf{x}'; \sigma_x) = \exp\left(-\frac{1}{2\sigma_x^2}\mathfrak{D}(\mathbf{x}, \mathbf{x}')^2\right) \; , \tag{3.3}$$

also called the (scaled) Gaussian function when the inputs are vectors and the similarity measure is the Euclidean distance. We will see that the scale parameter σ_x specifies the distance of dependency between inputs in the model. Depending on the method, the squared exponential function is referred to as a *basis function*, a *weighting function,* or a *covariance (*also *kernel) function*. While similar, the methods making use of this function are not equivalent. This is seen from the experimental results of Section 3.4. The choice of the squared exponential implicitly assumes that the dependency function $f(\mathbf{x})$ is very smooth (e.g. see [125]). This assumption is sensible for the NIR spectroscopy data because the physical relationship between inputs and targets is known to be smooth.

The second aspect of the comparison is to see if for the regression task on NIR spectroscopy datasets there is an advantage in using the similarity based methods rather than methods relying on a dot product between the (vector) inputs and the weight parameters

$$< \mathbf{x}, \mathbf{w} >= \mathbf{w}^\top \mathbf{x} \; . \tag{3.4}$$

The standard method used for regression on NIR spectroscopy is Partial Least Squares. This method is a special case of the standard linear model which is a dot product model. A second model using the dot product approach is the Multi-Layer Perceptron model (Section 3.2.3).

Finally, the third aspect compared is the model selection procedure. This aspect may be the most important as it follows the main concern of the thesis, namely the question about the potential advantages of the probabilistic approach for tackling real analysis. We will try to see if the marginal likelihood selection strategy derived from the probabilistic modelling approach is out performs the pragmatic cross-validation (CV) based methods. There are two advantages to the marginal likelihood. First, it is often less expensive to compute than cross-validation methods as the hyper-parameters are directly

tuned and training need not be repeated. Second, the procedure allows the optimisation of many hyper-parameters, giving more flexibility in the search for a good model. In contrast, we will systematically limit the search to one or exceptionally two hyper-parameter(s) to limit the computational cost of the resampling procedure. We will also see if the combination of a cross-validation selection with the marginal likelihood strategy is advantageous.

In fact, there are selection procedures other than the marginal likelihood which allow the derivation of an iterative optimisation scheme and thus to adjust several hyper-parameters; for example the leave-one-out estimator [1, 144, 27] for linear models and the radius margin bound for classification [30]. We are not comparing with these methods here; see [112, 28] for a discussion of the advantages and disadvantages these methods can have over the marginal likelihood.

As explained in Chapter 1, one can focus on different elements of the model configuration during the model selection stage. The marginal likelihood approach will be used to select the noise and prior distribution hyper-parameters but also to adjust the model space by means of the similarity measure. The simplest case is to work with the isotropic Euclidean (3.2) distance and only to adjust the global scale parameter σ_x. We will try to see if there is an advantage to adjusting instead a weighted Euclidean distance

$$\mathfrak{D}_{we}(\mathbf{x}, \mathbf{x}')^2 \equiv \|\mathbf{x} - \mathbf{x}'\|^2_{\mathbf{U}} = (\mathbf{x} - \mathbf{x}')^\top \mathbf{U} (\mathbf{x} - \mathbf{x}') \tag{3.5}$$

where $\mathbf{U}$ is a symmetric positive semi-definite precision matrix. The weight matrix $\mathbf{U}$ will be further constrained to limit the number of hyper-parameters (see the next subsection for further comments).

3.1.4 Feature relevance

Achieving accurate predictions is not the only possible objective for the regression task. As discussed in the previous chapters, in most preliminary analysis, practitioners are also interested in understanding the dependency between inputs and targets. However, the concept of understanding is rather abstract and subjective. With little domain knowledge or physical model in mind, the task of looking for interpretation is very vague. A simple approach to get some understanding from the learning task is by estimating the actual relevance of each feature for the prediction. This requires that the inputs have a vectorial representation and their features are interpretable.

There is no formal definition of the notion of relevance. Intuitively, the relevance of a feature should be related to the mutual information it has with the targets. But this definition is too restrictive. For example, it happens that each feature of a group has individually low mutual information with the targets while together, the group can predict the targets very well. Hence, in this case the features are relevant for the prediction.

An idea would be to assess the relevance after constructing the model by analysing the gradient of the function $f(\mathbf{x})$ with respect to the features. A more

principled approach is to associate a relevance parameter with each feature directly in the model. These relevance parameters must be adjusted during learning such that the additional relevance estimation is provided as a by-product of the prediction task. Of course, the relevance parameters might increase the capacity of the model and lead to over-fitting.

Automatic Relevance Determination One way to introduce relevance parameters in similarity-based models is to work with a diagonal weighting matrix $\mathbf{U} = \text{diag}\{[u_1, \ldots, u_D]\}$ in (3.5). Smaller precisions u_d correspond to features x_d less relevant for the prediction. This procedure is called an *Automatic Relevance Determination* (ARD) scheme [82, 95, 110]. A similar scheme exists for models based on a dot product (see Section 3.2.3). We will present in Section 3.3.3 a constrained ARD scheme allowing the identification of parts of the NIR spectra which are relevant for the regression task.

As mentioned above, one approach to optimising these relevance parameters is the marginal likelihood. However, other approaches have been proposed in the literature. Early strategies [79, 55, 50] proposed updating the relevance parameters based on the training criterion; however, this procedure often leads to over-fitting. A more robust approach is to base the tuning algorithm on a model selection criterion, such as the marginal likelihood. This is the technique used in [12] and [29], out of the probabilistic context. Recent literature in kernel methods tries to answer the more general question of global kernel optimisation; see for example [102, 74].

Feature selection Feature selection can be considered as a hard relevance estimation framework: selected features are relevant, others are not. See [49] and reference inside for a review of feature selection. As feature selection reduces the size of the input space, it is favourable for models whose capacity is linked to this size. Additionally, feature selection is advantageous in reducing the computational cost of learning and prediction. One thing that can be misleading with feature selection is that unselected features could still have high mutual information with the targets, but they are not selected because they do not contain more information than the already selected features.

Feature selection has already been used for NIR spectroscopy regression, for example in [13]. However, we do not intend to compare the ARD procedure with feature selection here.

Additive models Additive models are a particular model configuration that assume the contribution of features to the targets can be separated in additive terms

$$f(\mathbf{x}) = \sum_{d=1}^{D} f_d(x_d) \; . \tag{3.6}$$

Like the linear model, this configuration cannot represent nonlinear interactions between the features. Certainly, this is too simplistic for many real world appli-

cations. On the other hand, it is possible to visualise the average nonlinear dependency between each feature and the targets. Note that there are extensions to this formulation allowing the addition of multivariate terms $f_{d_{1,2}}(x_{d_1}, x_{d_2})$ which would encode nonlinear interactions between inputs. We will not be working with this model later on in this chapter. One can look in [52, 63] for additional details.

3.2 Overview of regression methods

In this section, we give an overview of some standard regression methods used by the machine learning community. The goal here is to give all the necessary information to understand the different aspects compared in section 3.4. One can find more thorough discussions of these methods in the references. In order to keep the description concise here, we have to omit many interesting extensions and learning strategies.

3.2.1 Linear Models

Standard linear Model The best known model applicable to vector inputs is the standard linear model. For this model, the dependency function is estimated by $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$, where $\mathbf{w} \in \mathbb{R}^D$ is a vector of weights and $b \in \mathbb{R}$ is the bias. The weights and bias are the parameters of the model. Note that this standard linear model is based on a dot product between inputs and parameters. For the ease of notation, we will use a slightly modified formulation where the bias contribution comes from an additional feature. Noting $\mathbf{x}^{(new)} = [1; \mathbf{x}]$, the model is more simply written as $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}^{(new)}$, where $\mathbf{w} \in \mathbb{R}^{D+1}$. This is the assumed representation for the developments below. As introduced in the preliminaries, it is convenient to train the model on the Mean Squared Error:

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{N} \sum_n (t_n - \mathbf{w}^\top \mathbf{x}_n)^2 \,. \tag{3.7}$$

The solution is

$$\hat{\mathbf{w}} = \mathbf{H}^{-1} \mathbf{X} \mathbf{t} \tag{3.8}$$

where $\mathbf{t} = [t_1, \ldots, t_N]^\top$, $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top$and $\mathbf{H} = \mathbf{X}^\top \mathbf{X}$.

Basis expansion Instead of working with the original inputs vectors $\mathbf{x}$, one could first transform them

$$\mathcal{B} : \mathbf{x} \in \mathbb{R}^D \rightarrow \phi \in \mathbb{R}^M : \phi = [\mathcal{B}_1(\mathbf{x}), \ldots, \mathcal{B}_M(\mathbf{x})]^\top \tag{3.9}$$

The set of functions represented by $\mathcal{B}$ is called a *basis expansion*. The principle is to work with the new features ϕ and construct a linear model on them, where this time $\mathbf{w} \in \mathbb{R}^M$. Again, the bias could correspond to one of the features.

Working with a basis expansion allows the encoding of nonlinear transformations of the inputs within the basis functions while keeping the learning procedure very simple. We will denote $\Phi = [\phi_1, \ldots, \phi_N]^\top$ and $\mathbf{H}_\phi = \Phi^\top \Phi$. There is no limit on the number of bases and consequently the capacity of the model is unbounded. Many basis expansions have the universal approximator property; see for example [104]. Note also that the parameters do not appear anymore in a dot product with the inputs.

Choosing a particular basis expansion can also be seen as doing a particular form of pre-processing of the original input data. We have already pointed out that expert knowledge should help to choose the representation, and equivalently the basis expansion. However, it is common to work with some well known basis expansions. When D is small (not bigger than 3), splines [14] - i.e. piecewise polynomial functions - or wavelets [32] are standard basis expansions. For larger input dimensionalities, radial basis expansions are more common. In Section 3.3.1, we will use the B-splines basis expansion for pre-processing the data. However, the regression task itself will be performed using the Gaussian *Radial Basis Functions* (RBF) expansion. Each basis is defined by

$$\mathcal{B}_\mathcal{G}(\mathbf{x}; \mathbf{c}, \sigma_x) \equiv \mathcal{G}(\mathbf{x}, \mathbf{c}; \sigma_x) \; . \tag{3.10}$$

This is the first appearance of (3.3). In this expression, $\mathbf{c} \in \mathbb{R}^D$ is at the centre of the basis from which the Euclidean distance is computed; the scale parameter σ_x specifies the local volume around $\mathbf{c}$ in which the basis is active. Note that the concept of locality for higher dimensional space is not very relevant as discussed in Subsection 3.2.2. There are different strategies to handle the inner parameters of the basis functions, σ_x and $\mathbf{c}$ in the case of the isotropic Gaussian RBF. The most pragmatic manner is to invoke a heuristic and fix some of these parameters at the beginning of the learning procedure. For example, the centres of the Gaussian RBF are typically set via vector quantisation of the inputs (with a k-means algorithm, or any other quantisation methods) [55]. These parameters could be called pre-processing parameters as they are not modified afterwards.

The second approach is to consider the basis parameters as model parameters that should be adjusted on the same training criterion as the weights $\mathbf{w}$. Two remarks must be made however: First, the basis parameters modify the nonlinear dependency between inputs and targets and this greatly increases the capacity of the model relative to another weight. So the tendency to over-fit will appear rapidly. The other remark is that training the basis parameters is not as simple as the weights. It is generally necessary to derive the expression of the gradient of the training criterion with respect to these basis parameters and resort to an optimisation technique (e.g. scaled conjugate gradient [92]). This second approach for adjusting the basis parameters is at the heart of the Multi-Layer Perceptron model presented below.

Finally, the third approach is to treat (some of) the basis parameters as hyper-parameters of the model space and adjust them with a model selection procedure.

Regularisation Even though the linear model is quite simple, working with a large dimensional feature space will inevitably lead to over-fitting. Adding a regulariser term in the training criterion is a way to reduce the capacity of a model configuration and thus circumvent over-fitting. One way to define a regulariser for the linear models is by a power sum expression of the weights

$$\mathcal{C}(f;\lambda,\nu) = \lambda \sum_d |w_d|^\nu \ . \tag{3.11}$$

The regularisation parameter λ controls the trade-off between the training loss $\mathcal{L}_\mathcal{E}$ and the regulariser in the training criterion $\mathcal{E}$. This parameter cannot be adjusted based on the training criterion because the optimal value would always be zero, i.e. no regularisation. It must be considered as a hyper-parameter and adjusted at the model selection stage. In the experiments, we will use a resampling method to adjust this regularisation trade-off parameter. It could also be possible to use other theoretically justified criteria (like the AIC, BIC, GCV [54]) to avoid to have a repetitive learning scheme. The second regularisation parameter ν specifies the geometry of the regularisation. This parameter is usually set *a priori* and not optimised. The classical choice is to set $\nu = 2$, such that

$$\mathcal{C}(f;\lambda) = \lambda \mathbf{w}^\top \mathbf{w} \ . \tag{3.12}$$

This is called a *ridge* regularisation (or *weight decay* regularisation) [58]. The regularisation is isotropic around the origin. One can show that the influence of this regulariser is to bias the weights toward the principal directions of the feature vectors $\mathbf{x}$ (or ϕ if a basis expansion is used). The advantage of using the ridge regulariser is that the training is no more complicated than (3.8): one just needs to replace $\mathbf{H}$ by $\mathbf{H} + \lambda \mathbf{I}_D$.

Another typical choice is $\nu = 1$ (called the *LASSO* [133]) because it leads to a sparse model. Indeed, as λ increases, an increasing number of weights have their optimal value exactly at zero. The training criterion is not convex anymore, but there exist efficient algorithms to rapidly find a local optimum of this training criterion [41]. Note that the sparsity induced by this regulariser makes the model simple to handle and it simplifies the analysis of the weights after training. To be precise, any $\nu \leq 1$ is likely to return a sparse model.

Spline regularisation The regularisation is sometimes defined directly on the function $f(\mathbf{x})$ to be estimated. This is used in the *smoothing spline* regularisation scheme for example. This regulariser attributes higher prior cost to non-smooth functions. Often, the integrated second derivative of the function is used to measure how non-smooth the function is:

$$\mathcal{C}(h;\lambda) = \lambda \int_\mathcal{X} f''(\mathbf{x})^2 \, d\mathbf{x} \ . \tag{3.13}$$

Smooth functions have low second derivative and are thus less strongly regularised. In practice, the regularisation can be rewritten

$$\mathcal{C}(h;\lambda) = \lambda \mathbf{w}^\top \mathbf{S}'' \mathbf{w} \tag{3.14}$$

where $\mathbf{S}''$ is a $M \times M$ matrix and $\mathbf{S}''_{ij} = \int_{\mathcal{X}} \mathcal{B}''_i(x)\mathcal{B}''_j(x)\, dx$. The smoothing spline regularisation is thus a kind of non-isotropic ridge regularisation. Again, optimising the weights is very easy; one needs to replace $\mathbf{H}$ by $\mathbf{H}_\phi + \lambda S''$ in (3.8).

PLS regression Another strategy to adjust the flexibility of a linear model is to reduce the dimensionality of the input vectors by projecting them on a small dimensional linear subspace. The projection (which is not necessarily orthogonal), is expressed by $\mathbf{z} = \mathbf{P}^\top \mathbf{x}$, where $\mathbf{P}$ is a $D \times J$ matrix. The dimensionality of the subspace J is a structural hyperparameter modifying the model space size. It is usually selected using a resampling method. Principal Component Analysis is a way to derive a linear subspace by keeping the J dominant components. However, it is unsupervised and in this sense it is suboptimal. In contrast, *Partial Least Square* (PLS) [147] is another projection procedure using the targets to derive the linear subspace. PLS is particularly efficient when the original features are numerous and highly correlated. This is typically the case for NIR spectroscopy spectra.

The PLS algorithm proceeds by finding the successive score vectors $\mathbf{z}_j = \mathbf{X}\mathbf{p}_j$ ($\mathbf{z}_j \in \mathbb{R}^N$ and $\mathbf{p}_j \in \mathbb{R}^D$) in order to maximise the covariance between $\mathbf{z}_j$ and the residual target vector, $\mathbf{r}_j = \mathbf{t} - \mathbf{y}_{j-1}$, where $\mathbf{y}_{j-1}$ is the vector of prediction at the previous step. Initialising with $\mathbf{y}_0 = \mathbf{0}_N$, i.e. a vector of zeros, the (simplified) PLS algorithm proceeds by computing

$$\begin{aligned}
\mathbf{p}_j &= \frac{\mathbf{X}^\top \mathbf{r}_j}{\|\mathbf{X}^\top \mathbf{r}_j\|} \\
\mathbf{Z}_j &= [\mathbf{z}_1, \ldots, \mathbf{z}_j] \\
\boldsymbol{w}_j &= \left(\mathbf{Z}_j^\top \mathbf{Z}_j\right)^{-1} \mathbf{Z}_j^\top \mathbf{t} \\
\mathbf{y}_j &= \mathbf{X}\mathbf{w}_j \,.
\end{aligned} \tag{3.15}$$

See [131] for a more thorough description of the algorithm and its applications.

Marginal likelihood on linear models We have seen in Section 1.3.2 that the Bayesian framework can be used to derive a higher-level likelihood criterion by marginalising over the parameters of the model. The hyper-parameters can then be optimised using this marginal likelihood. Note however that this criterion is still a likelihood criterion and in this sense it can lead to over-fitting if the capacity of the hyper-parameters space is sufficiently large. We present here a simple procedure following the derivation of the Laplace approximation (Section 1.3.3) which allows the efficient adjustment of regularisation hyper-parameters for linear models [80]. Moreover, the procedure can be used to optimise the basis hyper-parameters, although the optimisation is more costly.

To apply the Bayesian framework, one needs to specify a conditional target noise distribution $P(t|\mathbf{x})$ and a prior distribution on the parameters (i.e. the weights) $P(\mathbf{w})$. The procedure that we derive here makes the simple assumptions that the noise is additive homoscedastic Gaussian $t|\mathbf{x} \sim \mathcal{N}(f(\mathbf{x}), \sigma_\epsilon^2)$ and the prior is a multivariate Gaussian distribution centred at the origin $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{A}^{-1})$ where $\mathbf{A}$ is the inverse of the prior covariance matrix (i.e. a prior precision matrix). One advantage of these modelling assumptions is that the Laplace procedure is exact because the posterior over the parameters is truly a multivariate Gaussian distribution.

Using the notations of Section 1.3.3, we have

$$g(\mathbf{w}) = \frac{\tau}{2}\boldsymbol{\epsilon}^\top\boldsymbol{\epsilon} + \frac{1}{2}\mathbf{w}^\top\mathbf{A}\mathbf{w} \tag{3.16}$$

where $\boldsymbol{\epsilon}$ is a vector of length N of the target-prediction differences $\epsilon_n = t_n - y_n$ and τ is the inverse of the noise variance (also called the noise precision). One can recognise in this expression a mean square error criterion regularised by a non-isotropic ridge term. There is a slight difference due to the additional noise hyper-parameter τ.

The normalisation factors associated with the likelihood and the prior are the Gaussian normalisation factors

$$Z_\ell = \left(\frac{2\pi}{\tau}\right)^{\frac{N}{2}} \quad , \qquad Z_0 = \left(\frac{(2\pi)^M}{\mid \mathbf{A} \mid}\right)^{\frac{1}{2}} \quad . \tag{3.17}$$

One can show that the maximum a posteriori estimate of $\mathbf{w}$ is

$$\bar{\mathbf{w}} = \beta\Sigma_w\Phi^\top\mathbf{t} \tag{3.18}$$

where the posterior precision matrix is simply

$$\Sigma_w^{-1} = \tau\mathbf{H}_\phi + \mathbf{A} \ . \tag{3.19}$$

The exact log-marginal-likelihood is then

$$\begin{aligned} \log P(\mathbf{t}|\mathbf{X};\tau,\mathbf{A}) = &-\frac{\tau}{2}\boldsymbol{\epsilon}^\top\boldsymbol{\epsilon} - \frac{1}{2}\bar{\mathbf{w}}^\top\mathbf{A}\bar{\mathbf{w}} \\ &+ \frac{N}{2}\log\tau + \frac{1}{2}\log\mid\mathbf{A}\mid \\ &- \frac{1}{2}\log\mid\tau\mathbf{H}_\phi + \mathbf{A}\mid + C^{te} \ . \end{aligned} \tag{3.20}$$

Instead of working with a gradient based optimisation procedure to adjust $(\tau, \mathbf{A})$, a simple re-estimation strategy works well in practice [80]. Setting the gradient of the log marginal likelihood to zero,

$$\frac{\partial \log P(\mathcal{D}|\tau,\mathbf{A})}{\partial\tau} = -\frac{1}{2}\boldsymbol{\epsilon}^\top\boldsymbol{\epsilon} + \frac{N}{2\tau} - \frac{1}{2\tau}\mathrm{tr}\left(\mathbf{I} - \Sigma_w\mathbf{A}\right) = 0 \ , \tag{3.21}$$

the noise hyper-parameter is re-estimated with

$$\frac{1}{\tau} \leftarrow \frac{\boldsymbol{\epsilon}^\top \boldsymbol{\epsilon}}{N - \zeta} \tag{3.22}$$

where $\zeta = \mathrm{tr}\left(\mathbf{I} - \Sigma_w \mathbf{A}\right)$ is sometimes called the *number of well determined parameters*. "Well determined" should be understood in the sense that the information in the data are responsible for the estimate of ζ parameters, while the $M - \zeta$ other parameters are primarily determined by the prior. We see in (3.22) that this number plays a role equivalent to the degree of freedom used in the frequentist correction to the bias of a maximum likelihood estimate of the noise variance.

For the prior hyper-parameters, the derivation is similar. Let us consider the particular case of a diagonal prior precision matrix, with a different hyper-parameter for each weight, $\alpha = [\alpha_1, \ldots, \alpha_M]^\top$, $\mathbf{A} = \mathrm{diag}\{\alpha\}$, a diagonal matrix; we have then

$$\frac{\partial \log P(\mathcal{D}|\tau, \mathbf{A})}{\partial \alpha_m} = -\frac{1}{2}\bar{w}_m^2 + \frac{1 - \alpha_m \left(\Sigma_w\right)_{mm}}{2\alpha_m} = 0 \tag{3.23}$$

and the re-estimation of the prior precision hyper-parameters are

$$\frac{1}{\alpha_m} \leftarrow \frac{\bar{w}_m^2}{\zeta_m} \tag{3.24}$$

for all m, where $\zeta_m = 1 - \alpha_m(\Sigma_\theta)_{mm}$. One can verify that $0 \leq \zeta_m \leq 1$. This particular regularisation configuration with the marginal likelihood optimisation is known as the *relevance vector machine* [135]. The interesting property of this configuration is that in general, when we are working with a large feature space, many of the regularisation hyper-parameters take their optimal value at $\alpha_m = \infty$, leading to an equivalent pruning of the corresponding weights, $w_m = 0$. The final model is thus very sparse. In [138] a more efficient update strategy is proposed making use only of small analytical updates instead of the re-estimation formulas.

To be complete, when some of the features are constrained to have the same prior hyper-parameter, $\alpha_m = \alpha_j^0$ for $m \in \mathrm{M}_j$, the update is then

$$\frac{1}{\alpha_j^0} \leftarrow \frac{\sum_{m \in \mathrm{M}_j} w_m^2}{\sum_{m \in \mathrm{M}_j} \zeta_m} \,. \tag{3.25}$$

The process of computing the posterior statistics (mean and covariance) and re-estimating the hyper-parameters must be repeated until convergence. It is always a good practice to verify that the marginal likelihood increases after each iteration and that it is not converging to indeterminacy due to $\tau \to \infty$.

One could also consider updating the basis function parameters using this marginal likelihood criterion. It is in general not possible to derive simple re-estimation formula but one can always compute the gradient of (3.20) and

proceed with a general optimisation algorithm. In practice, combining the re-estimation formula and the gradient optimisation on the basis parameters is rather tricky. We would recommend either gradient optimisation for all the hyper-parameters (we will come back in Section 3.2.4 on this strategy) or combine the re-estimation formula with a resampling method to adjust a single basis hyper-parameter. Typically for the Gaussian RBF expansion, the isotropic scale parameter σ_x would be selected by a resampling scheme while the centres are set by the quantisation heuristic.

Generalized Linear Models Until now, we have worked with the square error training criterion and the Gaussian noise distribution. We have seen that training a model on those criteria makes use of linear algebra. Nevertheless, the linear model is not tied to this training criterion. A natural extension to the linear model is the Generalized Linear Models (GLM) framework [98, 90]. The idea is to link a linear activation $\eta = \mathbf{w}^\top \mathbf{x}$ with the mean of the conditional target distribution

$$\mathbb{E}_{t|\mathbf{x}}\{t\} = \mu = g(\eta) \; . \tag{3.26}$$

where $g(\cdot)$ is called the *link function*[1]. Note that η is the standard symbol used in the literature for the activation but one should be careful here not to confuse the activation with a hyper-parameter. The conditional target distribution is further constrained to belong to the exponential family of distributions. Distributions of the exponential family for 1-dimensional random variable can be expressed in the following form

$$P(t|\theta, \psi) = \exp\left\{\frac{t\theta - b(\theta)}{a(\psi)} + c(t, \psi)\right\} \tag{3.27}$$

In this expression, ψ is sometimes called the dispersion (or nuisance) parameter, θ is the canonical parameter and $a(\cdot)$, $b(\cdot)$, $c(\cdot, \cdot)$ are some known functions. One can show that the relation between the mean μ and the canonical parameter is $\mu = b'(\theta)$. Hence, the conditional distribution can be parameterised directly with the weights, i.e. $P(t|\mathbf{x}; \mathbf{w}, \psi)$.

Many standard distributions belong to this family: the Bernoulli, the binomial, the multinomial, the Poisson, the negative-binomial, the Gaussian, the Gamma, the Beta... to cite just a few. Note that the list contains both discrete and continuous distributions. Working with discrete distributions like the Bernoulli and the Multinomial, it is possible to tackle respectively binary and multi-class classification with the GLM framework. Actually, one could use any parametric conditional distribution instead of a member of the exponential family. The reason why the family is often useful is that it contains already a large variety of distributions and the updates of the weights is generally fast using simple iterative methods (Newton-Raphson or *iteratively reweighted least*

[1]The name "link function" is also used for the inverse function $g^{-1}(\mu)$ by Statistics community.

squares procedures). More information on the update procedure is given in Appendix A.2. Like linear models, one can extend the Generalized Linear Models by working with basis expansions and add a regulariser to control the capacity of the models.

In Chapter 4, we will make use of the GLM formulation on a regression task in the context of collaborative filtering.

3.2.2 Locally Weighted Learning

A fundamental assumption in the regression learning is that the conditional distribution of the targets $P(t|\mathbf{x})$ is continuous with respect to the inputs, possibly apart from some frontier regions. This means that inputs close to each other should have a similar conditional target distribution and thus locally a simple model should suffice to represent the variability of targets (or some useful statistics of $P(t|\mathbf{x})$ like the mean). To derive a prediction for a query input $\mathbf{x}_0$, it is intuitively appealing to construct a simple local model, based preferentially on the observed samples close to this query $\mathbf{x}_0$. That is exactly the idea of locally weighted learning [8, 78, 44].

The key point of this approach is to specify a distance measure and to define how this measure of closeness translates to contributions in the construction of a local model. For vectorial data, it is normal to work with the Euclidean distance defined in (3.2). As the Euclidean distance is isotropic, it assumes that the variability of targets is *a priori* the same in all directions. If the data justifies it, one can also work with less common distances measure, like Riemannian metrics, the Manhattan distance... [46].

The contribution of samples to the construction of a local model around $\mathbf{x}_0$ is evaluated by a weighting function $\mathcal{W}(\mathbf{x};\mathbf{x}_0)$ which is a monotonically decaying function of the distance to the query vector - the further to the query point, the lower its contribution. A classical weighting function is the squared exponential (3.3)

$$\mathcal{W}_{\mathcal{G}}(\mathbf{x};\mathbf{x}_0,\sigma_x) \equiv \mathcal{G}(\mathbf{x},\mathbf{x}_0;\sigma) \;, \tag{3.28}$$

where σ_x specifies the neighbourhood around $\mathbf{x}_0$, i.e. the size of the region around $\mathbf{x}_0$ which has an influence on the construction of the local model. Other weighting functions always have a similar scale parameter to adjust the neighbourhood size. Note that after the Gaussian RBF, this is a second use of the squared exponential (or Gaussian) function.

In the local weighted framework, the parameters θ of the model are adjusted in order to minimise the training loss $\mathcal{L}_{\mathcal{E}}$ locally at the query point $\mathbf{x}_0$. Mathematically, the task is formulated by

$$\hat{\theta}_0 = \underset{\theta\in\Theta}{\operatorname{argmin}} \sum_n \mathcal{W}(\mathbf{x}_n;\mathbf{x}_0)\mathcal{L}_{\mathcal{E}}\left(t_n, f(\mathbf{x}_n;\theta)\right) \;. \tag{3.29}$$

The prediction for the query point is then $y_0 = f(\mathbf{x}_0;\hat{\theta}_0)$.

The simplest model to work with is the local constant, i.e. $f(\mathbf{x}, \theta) = y_0$ on the neighbourhood of $\mathbf{x}_0$. For this simple model, when the training loss is the square error loss, we find that the prediction y_0 should be the weighted empirical mean

$$y_0 = \frac{\sum_n \mathcal{W}(\mathbf{x}_n; \mathbf{x}_0) t_n}{\sum_n \mathcal{W}(\mathbf{x}_n; \mathbf{x}_0)} \ . \tag{3.30}$$

This is known as the *Nadaraya-Watson estimator* [94, 146]. One could also work with a local linear model, $y(\mathbf{x}) = \mathbf{w}_0^\top \mathbf{x}$, or any more complex model. When $\sigma_x \to \infty$, the local model becomes a global model as all samples have the same contribution (or weight) to the training criterion. The smaller σ_x the higher the tendency to over-fit because there are fewer points contributing to the training of the parameters. So the scale parameter governs the capacity of the model. Adding a regulariser to the training criterion could of course be considered to further reduce the capacity of the local model. A good property of locally weighted learning is that it is a consistent learning procedure when the size of the neighbourhood decreases slowly as the number of training samples increases [44].

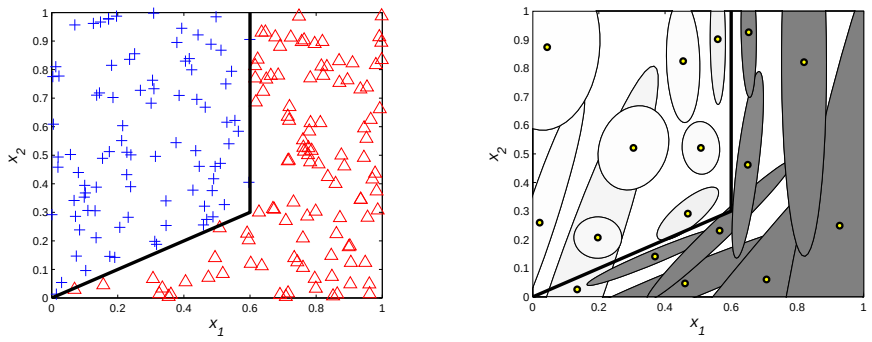
One can play with many variants of the local weighting scheme; for example by choosing different local models or trying different weighting functions. More importantly, the definition of the neighbourhood is essential. The simplest approach to defining the neighbourhood is to fix the distance measure from the start (e.g. the isotropic Euclidean for vector inputs), and to work with a global neighbourhood scaling parameter σ_x. This hyper-parameter is often adjusted by a resampling method. Choosing a global neighbourhood scale is not necessarily the best solution since the density of input samples can vary widely; points lying apart from the main clusters will thus have very few local points to construct the model with confidence. The simplest correction is to link the neighbourhood scale with the distance to the closest neighbours. This is the approach followed by the K-Nearest Neighbours (K-NN) methods. The neighbourhood scale could for example be

$$\sigma_K(\mathbf{x}_0) = \lambda_x \sigma_K(\mathbf{x}_0) \tag{3.31}$$

where $\sigma_K(\mathbf{x}_0)$ is the mean distance of the $\mathbf{x}_0$ to its K nearest neighbours (K is chosen *a priori*), and λ_x is a factor adjusting the size of the neighbourhood.

Another extension to the neighbourhood definition is to work with an adjustable, possibly anisotropic, distance measure. For example, one could use a weighted Euclidean distance, defined in (3.5). This definition implies that local distances are stretched in directions parallel to the dominant eigenvectors of the precision matrix $\mathbf{A}$, so the points in these directions contribute less to the construction of the local model. The consequence is that the dependency function $f(\mathbf{x})$ is allowed to be less smooth in those directions. Note that there is a redundancy to adjust in the same time $\mathbf{A}$ and σ_x, so one can set $\sigma_x = 1$.

Automatic DANN To make the framework even more flexible, the anisotropy could depend on the query point $\mathbf{A}(\mathbf{x}_0)$. The difficulty in working with such



(a) (b)

Figure 3.1: (a) Training samples, (b) shape of final neighbourhood derived with the automatic DANN method.

local models is to have the learning procedure deriving a good configuration for the neighbourhood size and anisotropy. In the paper *Automatic adjustment of Discriminant Adaptive Nearest-Neighbours* [35], we proposed an ad-hoc procedure to adjust $\mathbf{A}(\mathbf{x}_0)$ for a classification task. The idea is to estimate locally the most discriminant direction and to define the neighbourhood based on this direction. A nearest neighbour classifier is then used to make a prediction at the query point. The discriminant direction is estimated by a local *Linear Discriminant Analysis* (LDA) model. Our contribution to improve the framework proposed in [53] has been to specify some heuristic model selection criteria to set the size of the local LDA model and to set the size and anisotropy of the final NN-classifier neighbourhood. Figure 3.1 illustrates the idea on a 2-dimensional toy example. Figure 3.1(a) corresponds to the training samples and Figure 3.1(b) shows the neighbourhood obtained by the procedure for some query locations. We see that the neighbourhoods are stretched in directions perpendicular to the decision frontier such that the contribution to the NN-classifier of samples from the correct class should be higher than with an isotropic neighbourhood model.

In practice, we observed that the framework does perform better than an isotropic distance measure on low-dimensional datasets but it shows its limit for higher dimensional space. This is understandable as the number of hyper-parameters to adjust increases with the dimensionality and over-fitting inevitably appears. More fundamentally, the concept of neighbourhood in higher dimensional spaces looses its attractiveness. One can show that pairwise distances tend to concentrate around a mean value as the inherent dimensionality of the input space increases [46]. Consequently, all the samples tend to contribute equally to the local model, which become rather a global model. For this reason, other regression methods might be more appropriate in higher

dimensional space.

3.2.3 Multi-Layer Perceptron

The Multi-Layer Perceptron (MLP) is the model which made feed-forward neural network so famous. The principle is to combine simple processing functions called *neurons*. A neuron is represented by

$$z_{out} = g(\mathbf{w}^\top \mathbf{z}_{in}) \; . \tag{3.32}$$

It is thus a linear transformation of an entry vector $\mathbf{z}_{in} \in \mathbb{R}^{D_{in}}$ passed through a smooth link function $g(\cdot)$, returning a scalar z_{out}. Neurons set in parallel form a layer. In general, all neurons of a layer have the same link function, hence the output of a layer can be written by

$$\mathbf{z}_{out} = g(\mathbf{W}\mathbf{z}_{in}) \tag{3.33}$$

where $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_M]^\top$ is a matrix of weights, $\mathbf{z} \in \mathbb{R}^M$ and M is the number of neurons in the layer. The function $g(\cdot)$ is applied element-wise. The complete MLP model is composed of successive layers, let us say L layers; propagating the processing through the layers, we can compute the successive inner states

$$\mathbf{z}^{(l)} = g^{(l)}(\mathbf{W}^{(l)}\tilde{\mathbf{z}}^{(l-1)}) \; , \; l = 1 \ldots L \; . \tag{3.34}$$

In this notation, $\mathbf{z}^0 = \mathbf{x}$ and $y = z^{(L)}$ (for one dimensional targets) and the "tilde" on entry vectors $\tilde{\mathbf{z}}^{(l-1)}$ indicates that we are working with an additional bias term, i.e. $\tilde{\mathbf{z}} = [1; \mathbf{z}]$. The model is thus configured by setting the number of layers, the number of neurons in each layer and the types of link functions. The training stage corresponds to the optimisation of the different layers of weights $\mathbf{W}^{(l)}$.

There are two reasons that explain the success of this model. First, it has been proved [61, 11] that the model is parsimonious and has the universal approximator property already with a single layer of nonlinear neurons (followed by a linear one). The parsimony property is important as it means that for a given precision of the approximation function, the MLP requires much fewer weights (in general) than a linear basis expansion model. The second reason for its success comes from practical considerations. One can see that if only the weights of the last layer are updated, the MLP is simply a (generalised) linear model with the basis functions set by the fixed weights of the previous layers. As all the layers of weights are optimised - based on the learning criteria - the MLP can be viewed as a (generalised) linear model where the basis parameters are adjusted on the training criterion as suggested in Section 3.2.1. The advantage of the MLP over the other linear basis expansion configurations is that the gradient with respect to the layers of weights can be efficiently computed because the weights only appear in linear dot products followed by element-wise nonlinear transformation. The computation of the gradient is known as the *back-propagation* algorithm. It is obtained by applying the rules

of differentiation to combined functions. Inserting the gradient evaluations in a sophisticated optimiser (like the *scaled conjugate gradient* [92], quasi-Newton methods...), it is possible to train large MLP networks in a reasonable amount of time.

The drawback is that the MLP can easily become too flexible. Hence, the difficulty of learning a MLP is to control its capacity and avoid over-fitting. Also, convergence to bad local optima of the learning criterion is an issue for some datasets. By bad local optima we mean that the value of the training criterion at the local optima is still much higher than at the global optimum. Working with standardised inputs generally reduces the problem. In addition, a pragmatic solution to deal with bad local minima is to optimise the model from several random initialisations of the weights and keep the best one. A better approach is to form a committee model with the networks obtained from the different initialisations. This procedure is close in spirit to the marginalisation of probabilistic models. This generally reduces the tendency towards over-fitting. The problem is that this iterative re-training of an MLP can get computationally very expensive.

The capacity of a MLP comes from the large number of weights, so one way to control it is to reduce this number. The number of weights in the first layer is proportional to the dimensionality of the input vectors; it is thus advantageous to reduce this dimensionality as much as possible during the pre-processing stage. This is the reason why the MLPs are often constructed on the dominant principal components of the input vectors. The other obvious way to control the number of weights is by means of the number of neurons. Of course, there exist also more elaborate strategies trying to automatically prune some of the weights.

As in the linear models, the capacity of the MLP model can also be controlled by a regularisation term defined on the weights. As the different layers do not contribute to the targets with the same strength, it might be good to have a different regulariser for each layer. Again, with several hyper-parameters to adjust (the regularisation parameters, the number of neurons...) it is impractical to carry out a resampling model selection. For the regularisation parameters, the marginal-likelihood with the Laplace procedure is applicable [81]. Note however that the Laplace procedure is not exact for the MLP contrarily to the linear models. It means that the posterior distribution over weights may be very different from a Gaussian distribution. In particular, the posterior distribution might have a large asymmetric tail. Besides, computing the Hessian of (3.16) with respect to the weights is not easy, see [15].

A particular configuration of regulariser for a MLP model is to impose a common prior precision on the weights of the first layer interacting with the same entry of the input vectors

$$w_{md}^{(1)} \sim \mathcal{N}(0, \frac{1}{\alpha_d}) \ , \ m = 1 \dots M^{(1)} \ , \ d = 1 \dots D \ . \tag{3.35}$$

This regularisation scheme is also called an *Automatic Relevance Determination*

(ARD) scheme in the literature (see section 3.1.4); the larger the regularisation parameter α_d, the less relevant the corresponding feature x_d. Note that both ARD for the scale parameters of $\mathcal{G}(\cdot)$ and the MLP model have similar consequences: features with lower relevance contribute less to the prediction. But the principle is not identical. Here, we are adjusting the regulariser (or prior) parameters, while the precision parameters of the weighted Euclidean distance modify the function space itself.

3.2.4 Gaussian Process

Formally speaking, a Gaussian Process (GP) is a way to define a prior distribution $P(f)$ on the function space $\mathcal{H}$ [112]. The important feature is that this distribution is completely specified by a mean function $\mu(\mathbf{x}) = \mathbb{E}_f\{f(\mathbf{x})\}$ and a covariance function $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \mathbb{E}_f\{(f(\mathbf{x}) - \mu(\mathbf{x})) \cdot (f(\mathbf{x}') - \mu(\mathbf{x}'))\}$, where the expectation is over $P(f)$. To represent this prior distribution, we write

$$f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), \mathcal{K}(\mathbf{x}, \mathbf{x}')) \; . \tag{3.36}$$

In practice, this definition means that any set of samples $\{(\mathbf{x}_n, t_n)\}_{n=1}^N$ associated to a function picked from this Gaussian Process follows a prior conditional Gaussian distribution

$$\mathbf{t}|\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K}) \tag{3.37}$$

where $\boldsymbol{\mu} = [\mu(\mathbf{x}_1), \ldots, \mu(\mathbf{x}_N)]^\top$ and $\mathbf{K}$ is the $N \times N$ covariance matrix, with entries $\mathbf{K}_{ij} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$.

It is useful to illustrate the principle on a 1-dimensional regression example. To keep it simple, let us set $\mu(x) = 0$. This is a frequent choice (after centring the targets) when we have little prior knowledge about the function $f(x)$ to estimate. Once more, the squared exponential radial function $\mathcal{G}(\cdot)$ defined in (3.3) appears in the definition of a valid covariance function,

$$\mathcal{K}_{\mathcal{G}}(\mathbf{x}, \mathbf{x}'; \sigma_x) \equiv \mathcal{G}(\mathbf{x}, \mathbf{x}'; \sigma_x) \tag{3.38}$$

and is called the (isotropic) Gaussian covariance here as we are working with vectorial inputs. The curves in Figure 3.2 are generated with this Gaussian covariance making use of the property (3.37) with a fine grid of inputs $[x_1 < x_2 < \ldots < x_N]$ covering the range. We see clearly the influence of the scale parameter σ_x : the smaller this scale parameter, the less smooth the curves generated from the prior.

Let us come back to the regression task. Assuming that we have an observed set of input-target pairs $\mathcal{D} = \{(\mathbf{x}_n, t_n)\}_{n=1}^N$ and the objective is to predict the target at a new input location x_0, it is easy to show that the posterior distribution of predictions according to this GP model is also Gaussian

$$P(t_0|\mathbf{x}_0, \mathbf{t}, \mathbf{X}) = \frac{\mathcal{N}(\mathbf{t}_0|\boldsymbol{\mu}_0, \mathbf{K}_0)}{\mathcal{N}(\mathbf{t}|\boldsymbol{\mu}, \mathbf{K})} = \mathcal{N}(t_0|y_0, \sigma_0^2) \tag{3.39}$$

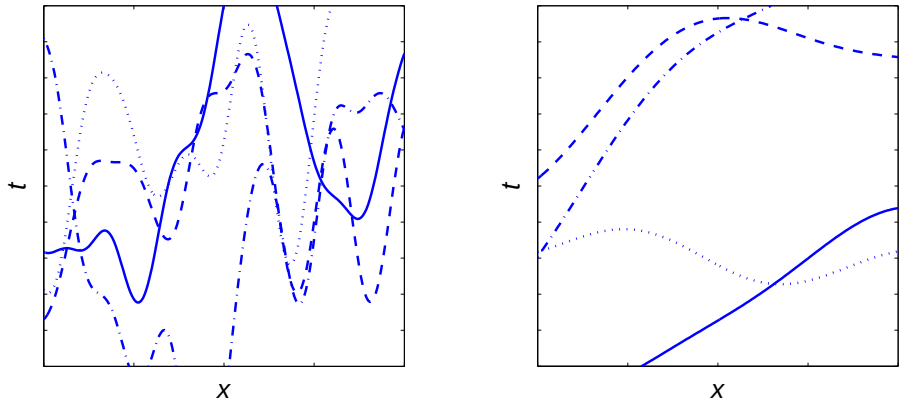


Figure 3.2: Illustration of functions picked from the Gaussian Process prior distribution with a Gaussian covariance function. (left) Small scale σ_x, (right) larger scale.

where

$$\begin{array}{rl} y_0 = & \mu(\mathbf{x}_0) + \mathbf{k}_0^\top \mathbf{K}^{-1}\mathbf{t} \\ \sigma_0^2 = & k_{00} - \mathbf{k}_0^\top \mathbf{K}^{-1}\mathbf{k}_0 \end{array} \tag{3.40}$$

In these notations,

$$\begin{aligned} \mathbf{t}_0 &= [t_0; \mathbf{t}] \\ \boldsymbol{\mu}_0 &= [\mu(\mathbf{x}_0); \boldsymbol{\mu}] \\ \mathbf{k}_0 &= [\mathcal{K}(\mathbf{x}_1, \mathbf{x}_0), \ldots, \mathcal{K}(\mathbf{x}_N, \mathbf{x}_0)]^\top \\ k_{00} &= \mathcal{K}(\mathbf{x}_0, \mathbf{x}_0) \\ \mathbf{K}_0 &= \begin{pmatrix} k_{00} & \mathbf{k}_0^\top \\ \mathbf{k}_0 & \mathbf{K} \end{pmatrix} \end{aligned} \tag{3.41}$$

As such, the model is assuming that the targets are noise free. Fitting the model with the covariance function (3.38) would interpolate the observed samples. In fact, it is very simple to add a Gaussian noise assumption in a Gaussian Process. This assumption can be encoded by an additional contribution to the covariance function:

$$\mathcal{K}_\epsilon(\mathbf{x}, \mathbf{x}') = \tau^{-1}\delta(\mathbf{x}, \mathbf{x}') \tag{3.42}$$

where $\delta(\mathbf{x}, \mathbf{x}') = 1$ if $\mathbf{x} = \mathbf{x}'$ and 0 otherwise. It is also common to adjust the global scale of the mean prediction function with a multiplicative factor α such that the complete (isotropic) Gaussian covariance with additive Gaussian noise is

$$\mathcal{K}(\mathbf{x}, \mathbf{x}') = \frac{1}{\alpha}\mathcal{K}_\mathcal{G}(\mathbf{x}, \mathbf{x}'; \sigma_x) + \tau^{-1}\delta(\mathbf{x}, \mathbf{x}') \tag{3.43}$$

Following the Bayesian approach, the natural way to adjust the (hyper-)parameters η of a covariance function (i.e. τ, α, σ_x here) is to optimise the

marginal likelihood of the model. In the Gaussian Process formulation, there is no explicit parameter space, hence the marginalisation applies directly over the functions $f(x)$,

$$P(\mathbf{t}|\mathbf{X};\eta) = \int_{\mathcal{H}} P(\mathbf{t}|\mathbf{X}, f(\mathbf{x})) \cdot \mathcal{GP}(f(\mathbf{x})|\mu(\mathbf{x}), \mathcal{K}(\mathbf{x}, \mathbf{x}';\eta))\, df \tag{3.44}$$

which might look complicated but the resulting expression was already given in (3.37). Interestingly, the definition of a Gaussian Process shortcuts the parametric space and directly specify a marginal likelihood. This gives us a very concise expression for the log-marginal likelihood:

$$\log P(\mathbf{t}|\mathbf{X};\eta) = -\frac{N}{2}\log 2\pi - \frac{1}{2}\log \mid \mathbf{K} \mid -\frac{1}{2}\mathbf{t}^\top \mathbf{K}^{-1}\mathbf{t}\ . \tag{3.45}$$

The optimisation is usually done with a gradient ascent iterative method. The gradient with respect to a particular covariance parameter η is

$$\frac{\partial}{\partial \eta}\log P(\mathbf{t}|\mathbf{X}) = \frac{1}{2}\mathbf{t}^\top \mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \eta}\mathbf{K}^{-1}\mathbf{t} - \mathrm{tr}\left(\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \eta}\right)\ . \tag{3.46}$$

As we have already discussed in Chapter 1, the marginal likelihood is still likely to over-fit in spite of the inner regularisation (the "*Occam factor*"). This occurs when the covariance is too flexible and the prior over the function space does not correspond well to the underlying data distribution. Additionally, the marginal-likelihood can have multiple local optima as shown in the illustration below.

The learning problem of a GP model is illustrated in Figure 3.3. A few input-target pairs are observed and the goal is to find the best values for (τ, σ_x) according to the marginal likelihood (note that the hyper-parameter α is not adjusted to make the illustration simpler). The lower left plot shows that the marginal likelihood is non convex and has at least two local optima; the dominant optimum is marked by a cross. The other optimum is locate on a crest of the marginal likelihood at small σ_x values. The two upper plots in Figure 3.3 shows the posterior predictive uncertainty computed with (3.40) for two different initialisations. The initialisations are located by the triangles in the lower left plot. We see that the model with a smaller scale σ_x (upper left) closely fits the observations. The higher capacity of this model gives rise to a rapidly growing uncertainty between the observations. Obviously, starting from this initialisation, the model converges to the second optimum. For the other initialisations (upper right), the initial noise is not well estimated as the observed data do not belong to the posterior predicted distribution. However, it is clear that this initialisation converges to the dominant optimum. The predicted uncertainty at the dominant optimum is shown on the lower right plot. The outer region is the uncertainty attributed to the noise while the inner region is the uncertainty attributed to the estimation of the mean function.

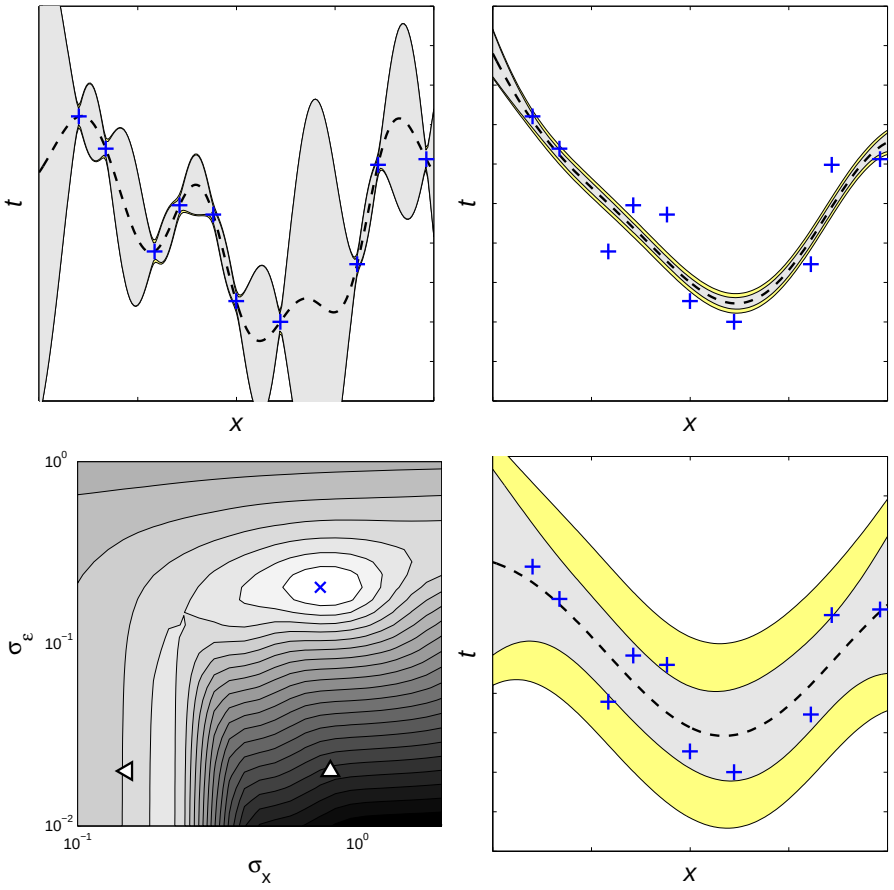


Figure 3.3: Illustration of a Gaussian Process learning problem. Lower left: marginal likelihood with respect to the scale σ_x and the noise standard deviation $\sigma_\epsilon = \tau^{-\frac{1}{2}}$ (brighter regions correspond to high values of the marginal likelihood). Other plots: predicted posterior uncertainty represented by $\pm 2\sigma_0$ (see Eq. (3.40)) around the mean function (outer region: standard deviation of noise, inner region: estimation uncertainty). Upper left: initialisation at a small σ_x (located at $\triangleleft$), upper right: initialisation at a larger σ_x ($\triangle$), lower right: global optimum ($\times$).

Link with linear model and Kernel trick In fact, we have already met a Gaussian Process earlier in the regression methods. Indeed, re-arranging formula (3.20), one can show that the marginal likelihood of the linear model with Gaussian noise and multivariate Gaussian prior distribution on the weights is

$$\mathbf{t}|\mathbf{X} \sim \mathcal{N}(0, \frac{1}{\tau}\mathbf{I}_N + \Phi^\top \mathbf{A}^{-1}\Phi) \tag{3.47}$$

a multivariate Gaussian distribution. This is the exact defining property of the Gaussian Process, Formula (3.37). We recognise in the covariance matrix the Gaussian noise term plus a weighted dot product of the inputs in their basis expansion representation:

$$\mathcal{K}(\mathbf{x}, \mathbf{x}') = \phi^\top \mathbf{A}^{-1}\phi' + \tau^{-1}\delta(\mathbf{x}, \mathbf{x}') \tag{3.48}$$

As was suggested earlier, one could optimise the basis parameters based on the marginal likelihood, using the gradient of the covariance matrix with respect to the basis parameters:

$$\frac{\partial \mathbf{K}}{\partial \eta_\phi} = \frac{\partial \Phi}{\partial \eta_\phi}\mathbf{A}^{-1}\Phi^\top + \Phi\mathbf{A}^{-1}\frac{\partial \Phi^\top}{\partial \eta_\phi} \tag{3.49}$$

and this expression can be introduced in (3.45).

This link with the linear model gives us further insight into the role played by the covariance function. One can show that computing any valid covariance function[2] is equivalent to computing a weighted dot product in a feature space. So the covariance function is in the same time defining the linear representation space and the regulariser in this representation. In fact, the features can stay implicit and they may even be infinite dimensional because all the interactions between inputs appear inside the covariance function. This property is referred to as *the kernel trick*. The covariance function is called a kernel because the implicit feature space it defines is a *Reproducing Kernel Hilbert Space*. The kernel trick is very useful as a simple covariance expression can represent in fact some clever interactions between the inputs.

One thing to note is that the dominant computational cost of the model is the inversion of an $N \times N$ matrix when it is formulated as a Gaussian Process while the linear model approach with an explicit basis expansion requires the inversion of an $M \times M$ matrix. The GP formulation allows working with very large feature spaces, but it leads to computational problems when applied to large datasets, typically with several thousands of input-target couples. However, there are some pragmatic solutions to construct approximate GP solution on larger datasets, see [112] and references therein.

During the past decade, the attention of the research community has shifted from the very versatile neural network models (the MLP in particular) toward

[2]To be valid the covariance function must satisfy the Mercer conditions, i.e. the covariance matrix associated to any set of inputs $\{\mathbf{x}_n\}$ must have non-negative eigenvalues only.

kernel methods like the GP and the support vector machines [121, 123]. The explanation is that all the difficulty of representing the data adequately, defining a regulariser, training the model and optimising the hyper-parameters for these kernel models stands in the definition and adjustment of the kernel function. The modelling procedure is thus greatly simplified. Besides, one can show that when working with an infinite dimensional implicit feature space, many kernel methods are consistent without any particular tuning.

Gaussian Processes were originally proposed for geostatistics applications [89]. It is interesting to know that the *Least-Squares Support Vector Regression* [130] method is equivalent to the GP in a non-probabilistic setting (i.e. the method is based on the square error and ridge regularisation instead of the Gaussian noise model and Gaussian prior distribution).

3.3 Manipulation of (smooth) functional data

The data on which the regression task is carried out in Section 3.4 are Near Infrared (NIR) spectra [100]. The key feature of these spectra is that they represent intrinsic smooth 1-dimensional curves. One can find information on the physical origin of NIR spectra and the devices used to record them in e.g. [140]. In practice, the spectra are finely discretised and we are thus given high dimensional ($D = 100 \ldots 1000 \ldots$) vectors as inputs. In this section, we present strategies to handle these high dimensional vectors and to take into account their intrinsic smooth functional nature.

As described below, functional processing [108] can be useful at three stages of the learning procedure. First, it can be used to work with a filtered and more compact representation of the spectra. Second, the functional aspects can guide us toward clever pre-processing methods. Third, one can apply natural constraints on the parameters of some of the models by imposing that these parameters take a smooth functional form.

3.3.1 Functional representation

Initially, each discretised curve is represented by a set of samples $\{(l_d, x_d)\}_{d=1}^{D}$. The inputs $\{l_d\}$ will be called the *evaluation points* of the curve and the corresponding outputs are the curve *values* x_d. Representing this curve by a functional entity $x(l)$ is essentially a simple 1-dimensional regression task. It is convenient to work with a basis expansion approach (Section 3.2.1):

$$x(l) = \sum_k \mathcal{B}_k(l) q_k \tag{3.50}$$

where $\mathbf{q} = [q_1, \ldots, q_K]^\top$ and $\mathcal{B}_k(l)$ for $k = 1 \ldots K$ is the basis expansion. Let us also specify the range of curves by $[l_{min}, l_{max}]$.

There are several reasons one might wish to go through this functional representation. The first reason is that each curve could be initially represented

by a different number of samples or different knot positions. This is illustrated in Figure 3.4. It is not possible to directly apply the regression methods to the curves if they do not have a common representation, as a dot product or a distance between irregularly discretised curves are not properly defined. The functional representation is thus a manner to obtain a common representation for which the dot product and distance are defined and simple to compute. For convenience, we will use the same basis expansion for all curves in the set. The dot product between two curves is then

$$\begin{aligned} < x(l), x'(l) > &= \int_{l_{min}}^{l_{max}} x(l) \cdot x'(l)\, dl \\ &= \mathbf{q}^\top \mathbf{S} \mathbf{q}' \end{aligned} \tag{3.51}$$

where $\mathbf{q}$ and $\mathbf{q}'$ are the weights in the basis expansion of $x(l)$ and $x'(l)$ respectively and $\mathbf{S}$ is a $K \times K$ matrix, with its entries defined by

$$\mathbf{S}_{ij} = \int_{l_{min}}^{l_{max}} \mathcal{B}_i(l) \cdot \mathcal{B}_j(l)\, dl \tag{3.52}$$

In the same way, the Euclidean distance is computed by

$$\mathfrak{D}_e\left(x(l), x'(l)\right) = \|\mathbf{q} - \mathbf{q}'\|_{\mathbf{S}} \tag{3.53}$$

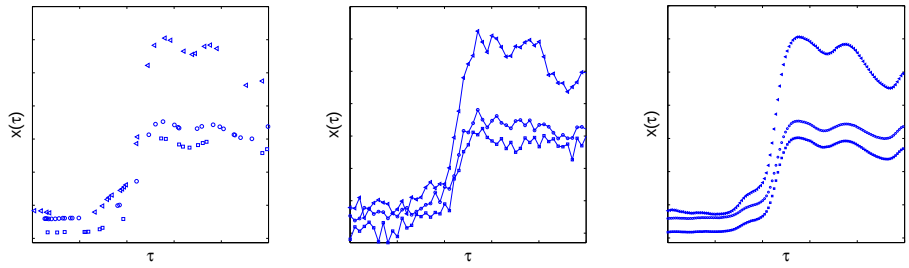
It represents the area between the two curves. We see that the common basis expansion representation makes the computation of these dot product and Euclidean distance as simple as working on a lower dimensional vector representation of the weight coefficients $\mathbf{q}$.

A second reason to use the functional representation is that is might help filter out some of the noise in the original curves. This is illustrated in Figure 3.4(b). Obviously, the dot product and Euclidean distance between noisy curves are also affected by the noise.

Finally, the functional representation is a pragmatic way to reduce the memory requirement and computational cost of finely discretised curve, like the ones shown in Figure 3.4(c).

For NIR spectroscopy at least, using a B-spline basis expansion makes sense as we know that the curves are intrinsically smooth. If the curves were spikier there would certainly be better choices of bases, for example the wavelet basis expansion. In general, the initial number of evaluation points for each curve is quite large such that the constructed functional representation is relatively insensitive to the number and position of the bases. A simple way to proceed is to look for a sufficient number of regularly spaced bases; starting with a small number, it is then increased until the expansion is large enough to represent adequately the original curves. This should already filter out some of the noise, but one can further apply smoothing spline regularisation (3.14).

It is true that for the NIR spectra the functional representation is usually not essential to be able to apply regression methods. In general, the spectra



(a) (b) (c)

Figure 3.4: Interest of using a functional representation: (a) irregular knot positions, (b) noisy original samples, (c) finely discretised functions.

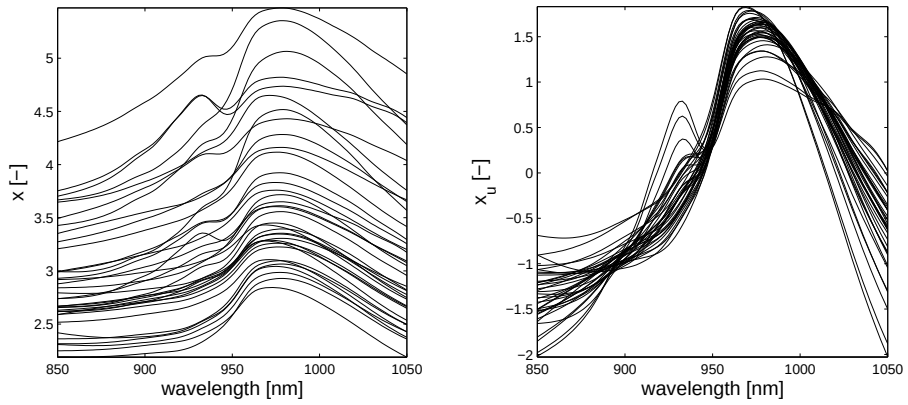
are all recorded with the same measuring spectrometer so the number and position of the evaluation points are the same. Also, the noise issue is not a real concern for this application because the spectrometer generally smoothes the spectra internally. Finally, if one is not concerned with the memory space required when the spectra are finely discretised, there is little advantage in working with the basis expansion representation. Nevertheless, the approach is general for curves and from the point of view of the parameterisation, reducing the dimensionality of the representation is interesting. Moreover, the number of bases necessary to represent the spectra well is related to the compexity of the spectra, contrarily to the number of evaluation points.

3.3.2 Functional pre-processing

We stated in the preliminaries of this chapter that a common sense pre--processing step when working with features of different units is to standardise the data (i.e. to centre and to normalise each feature). For functional data, the units are the same and it is not necessarily wise to perform this standardisation. The intention here is to make the relevant similarities and differences between curves more visible. With little prior knowledge however, one has to assume what might be relevant in the curves and what is probably not.

In Figure 3.5(a), the original spectra of one of the datasets used below - the "Tecator" dataset - are plotted. Using this original representation in the regression methods, it is common forthe dominant contribution to the dot product and the Euclidean distance between spectra to come from their global average and global vertical scale (i.e. on the value axis). The actual shape of the curve has only a secondary contribution. Moreover, it is known that these global aspects (average and scale) can vary considerably with the preparation of the samples before they are recorded by the spectrometer.

A simple pre-processing consists in normalising each curve separately, i.e. retrieve the mean value of the curve and divide by the standard deviation of



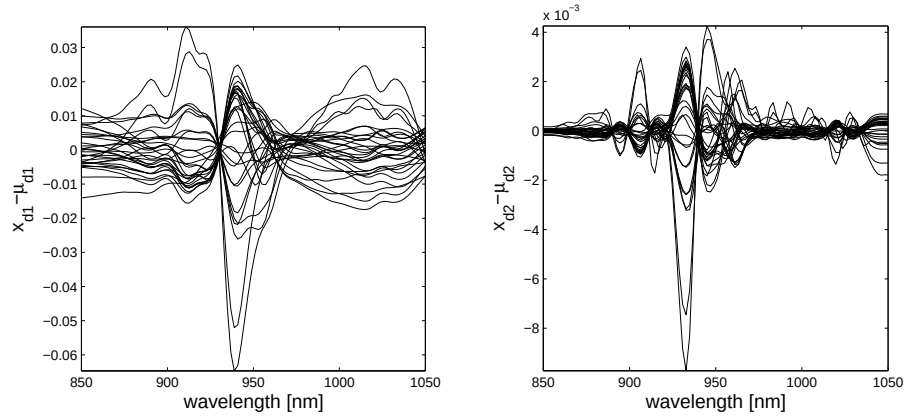
(a) original (b) normalized

Figure 3.5: (a) Original spectra of the Tecator dataset, (b) same spectra after normalisation.

its vertical variability. Figure 3.5(b) shows the same Tecator spectra after normalisation. The normalised spectra look much more similar, except maybe one or two which could be considered as outliers. The systematic procedure to spot outliers is described in Section 3.4.4.

Another approach to highlight the differences in the shape of the curves is to work with their derivatives. The first and second derivatives of the spectra for the same "Tecator" dataset are shown in Figure 3.6(a) and (b). To be precise, we subtracted the empirical mean of these derivative curves such that one can better see where the variability of the transformed curves lies. The role of the transformation is to give more importance *a priori* to some parts of the curve where the variability is higher (in the new representation). Here, we see that the second derivative representation gives more prior importance on the middle part of the spectra. Note that the computation of the derivatives, and any more complex pre-processing, will inevitably bring some estimation error. For derivatives in particular, the estimates at the limits of the range deteriorates rapidly.

Keeping only the normalised curves in constructing the regression model is not a good idea because the original mean and standard deviation of the curves may still contain relevant information for the prediction. In order to keep all the original information, one should combine the different components. For example, after normalisation of the spectra, we should work with the triplet of components $\mathbf{x}_u \equiv (\mathbf{q}_u, m_0, s_0)$, where $\mathbf{q}_u \in \mathbb{R}^J$ is the vector of weights of the normalised curves in the basis representation, m_0 is the original mean curve level and s_0 the original vertical standard deviation. For models based on a dot product, the simplest procedure would be to concatenate the component $\mathbf{x}_u^{(concat)} = [\mathbf{q}_u; m_0; s_0]$, and work with these extended vectors. To give equal *a*



(a) 1st derivative (b) 2nd derivative

Figure 3.6: First and second derivative representations of the normalised Tecator spectra.

priori importance to each component, one should first centre the components and scale them by their respective empirical standard deviations.

The principle is the same for distance based methods. Again, to give *a priori* equal importance to the different components, one should work with a weighted Euclidean distance

$$\begin{aligned}\mathfrak{D}_{we}(\mathbf{x}_u, \mathbf{x}'_u)^2 =& \frac{1}{\sigma_q^2}\|\mathbf{q}_u - \mathbf{q}'_u\|^2_{\mathbf{S}} \\ &+ \frac{1}{\sigma_\mu^2}(m_0 - m'_0)^2 + \frac{1}{\sigma_s^2}(s_0 - s'_0)^2 \end{aligned} \tag{3.54}$$

The scaling factors $(\sigma_q, \sigma_\mu, \sigma_s)$ could be heuristically set to the empirical mean of the Euclidean distance of their respective components such that each component has *a priori* approximately the same contribution to the prediction.

One might think of applying other functional pre-processing steps. We are not aware of other typical pre-processing applied to NIR spectra. For other applications, a classical functional pre-processing is the *curve registration* (also referred to as *dynamic time warping* in the engineering literature) [21, 109]. The principle is to look for shifts, stretches and compression in the horizontal axis of the curves in order to align similar shapes, corresponding to particular events in temporal processes for example.

3.3.3 Functionally constrained parameters

We have shown that the functional representation of the curves by a linear basis expansion is a convenient way to handle the original curves and to compute the

standard vector operations (dot products and Euclidean distance). We have also seen that for some regression models, the dimensionality of the parameters and/or hyper-parameters follows the dimensionality of the input space. This is the case for linear models without basis expansion (and the Partial Least Squares method in particular); it is also the case for the weights of the first layers of a MLP, and the precision parameters of a diagonally weighted Euclidean distance (3.5).

Working with a high-dimensional parameter space can be problematic from a computational point of view and from the learning point of view as the model may over-fit. In fact, this is not necessarily true; for example the PLS is more or less unaffected by the dimensionality of the inputs because the capacity of the model is well controlled thanks to the low dimensionality of the projection subspace. But for the two other models, the size of the parameters space is an issue.

Intuitively, parameters interacting with smooth (functional) entities should also have some smoothness constraints on them. The idea is thus to express these parameters with a B-spline basis expansion, and eventually to further constrain the smoothness by a smoothing spline regulariser (3.13) [51].

Dot product Constraining the weights appearing in the dot product to be expressed by a B-spline basis expansion [115] (without loss of generality, we take the same basis expansion as the inputs here), $w(l) = \sum_k \mathcal{B}_k(l)\mathbf{w}_k$, the dot product between functional inputs and parameters is simply

$$< x(l), w(l) >= \mathbf{q}^\top \mathbf{S}\mathbf{w} \tag{3.55}$$

where $\mathbf{S}$ is defined in (3.52). The MLP learning algorithm can be modified to work with this weighted dot product. The main difficulty of the MLP is still to derive an efficient model selection procedure; adding a smoothing regulariser does not simplify the task as one has to select an additional regulariser trade-off parameter. We did not follow this approach in the experiments.

Smooth ARD in weighted distance We will now show how the functional smoothing can be introduced in the precision parameters of a weighted Euclidean distance. This distance can then be introduced in a Gaussian covariance function in the Gaussian Process model and be adjusted on the marginal likelihood criterion. The procedure can thus be assimilated to a functionally smoothed Automatic Relevance Determination scheme.

A weighted Euclidean distance between functional inputs is computed by

$$\mathfrak{D}_{we}\left(x(l), x'(l); u(l)\right)^2 = \int_{l_{min}}^{l_{max}} \left(x(l) - x'(l)\right)^2 \cdot u(l)\, dl \tag{3.56}$$

where $u(l)$ is a positive precision function (i.e. the square inverse of a scaling function). This precision function could be represented by

$$u(l) = \frac{1}{\sigma_x^2}\, g\left(\sum_k \mathcal{B}_k(l)\gamma_k\right) \tag{3.57}$$

where the function $g(\cdot)$ assures that the precision function is always positive such that there is no constraint on the optimised coefficients $\gamma = [\gamma_1, \ldots, \gamma_K]^\top$. Again, the expansion needs not be the same as the curve basis expansion. A possibility is to use the logistic link function, $g(a) = (1 + \exp(-a))^{-1}$. With this choice, the scale parameter σ_x controls the upper bound of the precision function.

The drawback of this formulation is that the integration in (3.56) is not as simple as isolating the coefficients and computing once and for all the integrals over the basis functions. Instead, we prefer to come back to discretised curves on a fine grid of evaluation points $\{l_1 < \ldots < l_D\}$ and to compute the weighted Euclidean distance (with the coefficient of the basis representation) by

$$\mathfrak{D}_{we}\left(\mathbf{q}, \mathbf{q}'; \mathbf{U}\right)^2 = \Delta\mathbf{q}^\top \mathbf{B}^\top \mathbf{U}\mathbf{B}\Delta\mathbf{q} \tag{3.58}$$

where $\Delta\mathbf{q} = \mathbf{q} - \mathbf{q}'$, $\mathbf{B}_{dk} = \mathcal{B}_k(l_d)$ and $\mathbf{U} = \operatorname{diag}\{[u(l_1), \ldots, u(l_D)]\}$.

Working with this distance measure in the Gaussian covariance function

$$\mathcal{K}_{\mathcal{G}}(\mathbf{q}, \mathbf{q}'; \sigma_x, \gamma) = \exp\left(-\frac{1}{2}\mathfrak{D}_{we}\left(\mathbf{q}, \mathbf{q}'; \mathbf{U}\right)^2\right) \tag{3.59}$$

the gradient with respect to the coefficients γ is computed by

$$\begin{aligned}
\frac{\partial \mathcal{K}_{\mathcal{G}}(\mathbf{q}, \mathbf{q}')}{\partial \gamma_k} &= -\frac{1}{2}\mathcal{K}_{\mathcal{G}}(\mathbf{q}, \mathbf{q}') \cdot \frac{\partial r^2}{\partial \gamma_k} \\
\frac{\partial r^2}{\partial \gamma_k} &= \Delta\mathbf{q}^\top \mathbf{B}^\top \frac{\partial \mathbf{U}}{\partial \gamma_k}\mathbf{B}\Delta\mathbf{q} \\
\frac{\partial \mathbf{U}}{\partial \gamma_k} &= \operatorname{diag}\left\{\frac{1}{\sigma_x^2} g'\left(\mathbf{B}\gamma\right) : \mathbf{b}_k\right\}
\end{aligned} \tag{3.60}$$

where we noted $r \equiv \mathfrak{D}_{we}\left(\mathbf{q}, \mathbf{q}'; \mathbf{U}\right)$, $\mathbf{b}_k$ the kth column of the matrix $\mathbf{B}$, and $g'(\cdot)$ the derivative of the link function applied element-wise. The gradient can be used in (3.46) to derive the optimal precision parameters γ.

Working with a finely (discretised) precision function $u(l)$, it might be necessary to further regularise this function by a smoothing spline scheme. Hence, constructing a smooth-ARD Gaussian Process model with the covariance function (3.59) inserted in (3.43), the learning procedure consists in optimising the negative log-marginal likelihood (3.45) and an additional smoothing spline regularisation

$$\hat{\eta} = \underset{\eta}{\operatorname{argmin}} \; -\log P(\mathbf{t}|\mathbf{X}; \eta) - \lambda_s \gamma^\top \mathbf{S}'' \gamma \tag{3.61}$$

where the set of (hyper)parameters tuned on the (regularised) marginal likelihood is $\eta \equiv \{\tau, \alpha, \sigma_x, \gamma\}$, and the smoothing trade-off parameter λ_s is selected by cross-validation.

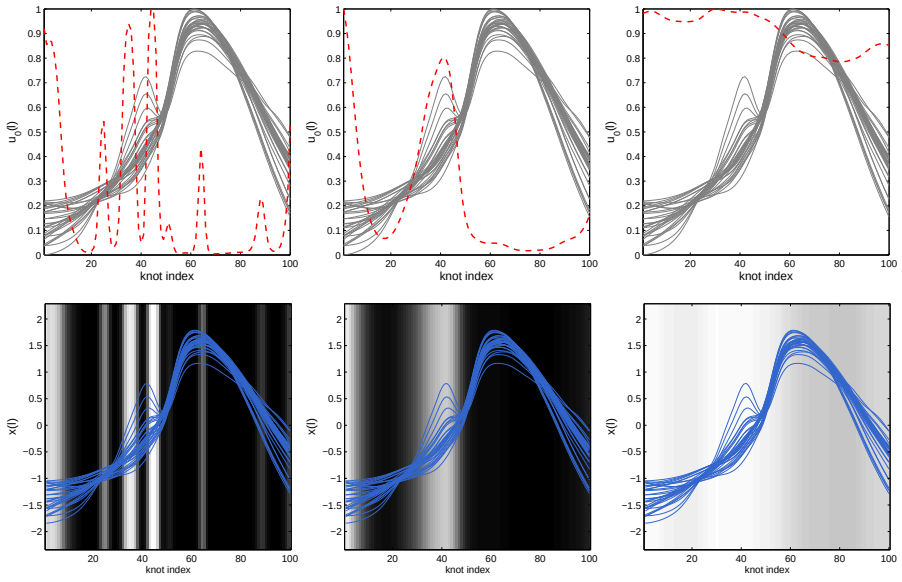


Figure 3.7: Illustration of the smooth ARD scheme on Tecator spectra. Dashed lines on the upper row represent the precision function $u(l)$ (scaled to have maximum at 1). Same information on the lower row: darker shadings correspond to lower values of the precision function.

The smooth ARD scheme is illustrated in Figure 3.7 on the Tecator spectra. The subfigures (a), (b) and (c) were obtained with an increasing smoothing regularisation, i.e. by means of an increasing λ_s. Both rows represent the same information, namely the shape of the optimised precision function $u(l)$ defined in (3.57) (to be exact, the function was scaled such that the maximum is set at 1 on the plots). Remember that regions where the precision function is high correspond to parts of the curves more relevant for the prediction. When the smoothing is low, the smooth-ARD behaves more like a feature selection scheme as it identifies narrow regions of high relevance (the extremities, the three close regions on the left and a smaller one on the right) surrounded by regions of very low relevance. With an intermediate smoothing (Figure 3.7(b)), we see that the three close regions of higher relevance have merged, while the extremities are not considered to be so relevant anymore. With a higher smoothing, the precision function becomes more or less flat; we come back to an isotropic configuration.

From the point of view of the performances only, we will see in Section 3.5 that this smooth ARD scheme on the Tecator dataset does not outperform the other methods. However, the shape of the precision function is interesting information as it tells us which parts of the spectra are most relevant for the

prediction of the targets. In this example, we clearly see that the smaller bumps in the middle left of the curves contain the most relevant information. This is confirmed by expert knowledge: the target to predict here are levels of fat content (see description of the Tecator data below) and the smaller bumps are situated in the 900-950 nm wavelength region which is precisely the region where fat constituents have the highest absorption of light.

3.4 Experimental setup

In Section 3.2, we have presented a set of methods for regression tasks. The goal here is to compare their performances on spectroscopy datasets.

3.4.1 Evaluation procedure

To evaluate the prediction performances of the different model configurations, we adopt a 5-fold cross-validation scheme repeated 20 times. In other words, the original datasets are split in five approximately equal size subsets. Each of these subsets is in turn held out of the learning stage. The predictions for the held-out samples averaged over the 5 folds give an estimate of the generalisation ability (see Chapter 1). To improve this estimate, the procedure is repeated 20 times with different splits. In total there are thus 100 learning-prediction runs for each model configuration.

3.4.2 Model configurations

We have presented in Subsection 3.1.3 the different aspects of the regression methods that we intend to compare; to summarise, the goal is to compare similarity based methods making use of the radial Gaussian function processing, dot product methods against similarity based methods, and the marginal likelihood against the cross-validation approach.

It should be noted that when a cross-validation scheme is used to select the best hyper-parameters of the model, this scheme is nested within the evaluation procedure, i.e. the learning sets in each of the 100 learning runs is further split in training and validation sets. By default, a 5-fold cross-validation is used. It is important to nest the model selection scheme to make the learning procedure independent of the evaluation and avoid introducing an optimistic selection bias in the evaluation.

The Gaussian Process with the covariance specified in (3.43) and isotropic Euclidean distance will be our base configuration for the comparison of the performances. The reason of this choice is that most of the other configurations can be seen as slight variations or extensions to this base configuration. Moreover, as all the hyper-parameters (τ, α, σ_x) are adjusted using the marginal likelihood, it requires only very little intervention from the practitioner, which is an important quality. This configuration will be noted $\mathcal{GP}_0$.

Influence of the pre-processing We have already pointed out that computing the Euclidean distance between the original spectra is probably not the best approach to exploit the information in the spectra (see section 3.3.2). The goal is to verify this statement. Using the base configuration $\mathcal{GP}_0$, we start by comparing the performances on four representations of the spectra:

- $\mathcal{D}_0$: Original representation.
- $\mathcal{D}_u$: Normalisation such that each spectrum has mean zero and unit standard deviation.
- $\mathcal{D}_{d1}$: First derivative of the normalised spectra.
- $\mathcal{D}_{d2}$: Second derivative of the normalised spectra.

For the following comparisons, we will keep the pre-processing method appearing to perform best for each dataset. This representation is noted $\mathcal{D}_*$ below. As explained in Section 3.3.2, the complementary information of the average curve levels and initial standard deviations are kept as additional components and we use the heuristic procedure described in this same section to set $\{\sigma_*, \sigma_\mu, \sigma_s\}$ (σ_* is the scale of the shape component of the selected representation). There is of course no guarantee that the selected pre-processing is the one giving the best prediction performances with the other regression models. But choosing the same pre-processing simplifies the comparison of the configurations. Moreover, similarity-based models described below are closely related and thus they behave similarly using the different pre-processing methods.

Comparison with the standard PLS method PLS regression is the most widely used method applied to NIR spectroscopy data. It is therefore interesting to see if nonlinear models like Gaussian Processes perform better in general. PLS is not based on the Euclidean distance and for this reason it is less sensitive to the mean and standard deviation of original spectra than the Gaussian covariance used in the GP model. Nevertheless, we construct the PLS using two pre-processing methods:

- PLS-$\mathcal{D}_0$: on original data.
- PLS-$\mathcal{D}_*$: on the best representation according to the base configuration $\mathcal{GP}_0$.

To complete the methodology, the original vectors are standardised before constructing the PLS models and the dimensionality of the working space is set by cross-validation.

Multi-Layer Perceptron We consider two Multi-Layer Perceptron configurations with a single layer of hidden neurons. To reduce the number of weights in the first layer, we work with a B-spline representation as proposed in Subsection 3.3.3, i.e. $w_m(l) = \sum_k \mathcal{B}_k(l) w_{km}^{(1)}$, $m = 1 \dots M$, where M is the number of hidden neurons. The model is constructed as a standard MLP (for which the weights and bias are $(\{\mathbf{w}_m^{(1)}\}_m, \mathbf{b}^{(1)}, \mathbf{w}^{(2)}, b^{(2)})$) on pre-processed entries $\mathbf{q}_*^{(new)} \leftarrow \mathbf{S}\mathbf{q}_*$ (Eq. (3.55)). $\mathbf{q}_*$ is the vector of B-spline coefficients of the spectra within the representation $\mathcal{D}_*$. The mean and standard deviation of the original spectra are entered in the model as additional inputs.

The number of B-splines for the weights of the first layer is set to $K = 30$ and the number of hidden neurons is set to $M = 5$. This number might seem low but note that the PLS model can be viewed as a special case of the MLP model with only a single hidden neuron. From some side experimentation, we did not see improvement in the prediction performances when this number of hidden neurons is larger.

The Netlab (www.ncrg.aston.ac.uk/netlab/) toolbox was used for the construction of the MLP models.

- MLP-ml: The inputs and targets are standardised and isotropic prior Gaussian distributions are applied to the weights and biases, with the prior precision hyper-parameters $\alpha = [\alpha_{b1}, \alpha_{w1}, \alpha_{b2}, \alpha_{w2}]$ and the noise hyper-parameter τ adjusted by a Laplace approximation of the marginal likelihood criterion.

- MLP-cv: a single ridge regularisation trade-off λ common to all the weights and biases is adjusted by cross-validation. The inputs and targets are standardised. This limits the issue about different scales in the first and second layers of weights.

Similarity-based models Here is the list of configurations using the isotropic Euclidean distance and subsequent Gaussian function 3.3 in the processing of inputs. To give the basis expansion approach *a priori* the same information as the $\mathcal{GP}_0$, the expansion is constructed with a Gaussian radial basis function centred on each learning sample and a common scale parameter σ_x.

- $\mathcal{GP}_\phi$: Gaussian Process with an explicit feature expansion (one basis for each learning sample, as described above). We use an isotropic regulariser as in (3.47), i.e. $\alpha \mathbf{I}_N$. Like $\mathcal{GP}_0$, all hyper-parameters are adjusted on the marginal likelihood.

- $\mathcal{B} - \sigma_x$: same principle as the previous configuration but with the basis expansion approach. The isotropic prior precision α and noise parameter τ are set with the re-estimation formula of Section 3.2.1 and the isotropic scale σ_x is selected with a cross-validation scheme.

- $\mathcal{B} - \sigma_x, \alpha$: this is the unique configuration for which we allow two hyper-parameters (σ_x, α) to be set by cross-validation. The parameter α is in fact the trade-off parameter in the ridge regulariser (3.12). As there is no model for the noise, the approach is fully non probabilistic.
- $\mathcal{W} - \sigma_x$: locally weighted learning with the Gaussian isotropic weighting function and a global scale parameter σ_x adjusted by cross-validation.
- $\mathcal{GP}_0 - \sigma_x$: base configuration where the isotropic scale σ_x is selected with a cross-validation scheme, the other two parameters (τ, α) are still adapted on the marginal-likelihood.

ARD scheme Finally, we look at two Gaussian Process configurations for which the distance measure is adjusted based on the marginal likelihood.

- $\mathcal{GP}_{ard}$: the scale parameters of the different components $\{\sigma_*, \sigma_\mu, \sigma_s\}$ are also adjusted. Note that there is a redundancy with the global scale σ_x which is thus left out of the model.
- $\mathcal{GP}_{s-ard}$: this configuration applies the smooth-ARD scheme proposed in Section 3.3.3. Additionally, the component scales (σ_μ, σ_s) are also adjusted. To keep the learning feasible, we fix the size of the basis representation of the precision function to $K = 10$ and will not further smooth the function.

To summarise, we have listed in Table 3.1 the different configurations and the way their hyper-parameters are adjusted.

3.4.3 Datasets

We have used 5 publicly available NIR spectroscopy datasets in the experiments. There are actually 7 regression tasks as we predict two different outputs for two of the datasets. Note also that we are not following the training-validation splits that have been used in other publications. All the spectra for which we have an observed target are used for learning (except the assumed outliers, see below). Here is a small description for each of them.

- Tecator [132]: the dataset consists of 240 near-infrared absorbance spectra of meat samples, recorded on a Tecator Infratec Food and Feed Analyzer. Each observation consists of a 100-channel absorbance spectrum in the 850-1050 nm wavelength range. The goal is to predict the fat content (expressed in percent) of the meat samples. There are two additional outputs (water content and protein content) which are not used here.
- Orange Juice [13]: the dataset contains 218 near-infrared absorbance spectra of dry extract of orange juices. Each observation is a 700-channel spectrum in the 1100-2500 nm range. The goal is to predict the percentage of saccharose (i.e. a constituent of sugar).

ID	Configuration	Fixed	ML	CV
'A'	$\mathcal{GP}_0 - \mathcal{D}_0$	/	τ, α, σ_x	/
'A'	$\mathcal{GP}_0 - \mathcal{D}_u$	$\sigma_u, \sigma_\mu, \sigma_s$	τ, α, σ_x	/
'A'	$\mathcal{GP}_0 - \mathcal{D}_{d1}$	$\sigma_{d1}, \sigma_\mu, \sigma_s$	τ, α, σ_x	/
'A'	$\mathcal{GP}_0 - \mathcal{D}_{d2}$	$\sigma_{d2}, \sigma_\mu, \sigma_s$	τ, α, σ_x	/
'B0'	PLS-$\mathcal{D}_0$	/	/	J
'B*'	PLS-$\mathcal{D}_*$	/	/	J
'C'	MLP-ml	M, K	τ, α	/
'D'	MLP-cv	M, K	/	λ
'E'	$\mathcal{GP}_\phi$	$\sigma_*, \sigma_\mu, \sigma_s$	τ, α, σ_x	/
'F'	$\mathcal{B} - \sigma_x$	$\sigma_*, \sigma_\mu, \sigma_s$	τ, α	σ_x
'G'	$\mathcal{B} - \sigma_x, \lambda$	$\sigma_*, \sigma_\mu, \sigma_s$	/	λ, σ_x
'H'	$\mathcal{W} - \sigma_x$	$\sigma_*, \sigma_\mu, \sigma_s$	/	σ_x
'I'	$\mathcal{GP}_0 - \sigma_x$	$\sigma_*, \sigma_\mu, \sigma_s$	τ, α	σ_x
'J'	$\mathcal{GP}_{ard}$	/	$\tau, \alpha, \sigma_*, \sigma_\mu, \sigma_s$	/
'K'	$\mathcal{GP}_{s-ard}$	K	$\tau, \alpha, \sigma_x, \gamma, \sigma_\mu, \sigma_s,$	/

Table 3.1: Summary table of the the model configurations and the way their hyper-parameters are handled (Fixed: set *a priori* and not optimised, ML: marginal likelihood, CV: cross-validation). The first columns is the indicator letter used in Figures 3.9 and 3.10. Symbols: σ: scale parameter, τ: precision on noise model, α: precision on Gaussian prior, λ: ridge trade-off parameter, J: dimension of latent space, M: number of hidden neurons, K: number of B-spline coefficients, γ: coefficient of the smooth-ARD precision function.

- Apple [13]: the dataset contains 337 near-infrared absorbance spectra of apples. Each observation is a 110-channel spectrum in the 1100-2190 nm range. The targets correspond to the crushing forces necessary to insert a stem in the skin of the apples. The idea is to predict if an apple is ripe without crushing it.

- Chimiometrie 2006 [106]: the dataset was proposed for a prediction challenge at the *Chimiometrie 2006* conference. It contains 618 near-infrared spectra of dried and sieved soil samples, typical of data analysed in pedometrics[3]. The 207 validation spectra of the challenge are not used here. Each observation is a 700-channel spectrum in the 1100-2498 nm range. The prediction was aimed at three outputs, the proportion of Nitrogen, the Cation Exchange Capacity and the content of Carbon (in percentage of dry soil). We will try to predict the Nitrogen and Carbon outputs (the Nitrogen and Cation Exchange Capacity outputs are highly correlated; there is thus little interest to predict both of them). They are referred as to Chimiom-nt and Chimiom-c below. Note that a good deal of the targets are missing for each output (see numbers in Table 3.2).

- Shootout 2006 [24]: the dataset was proposed for a prediction challenge at the "The International Diffuse Reflectance Conference 2006". It contains 2761 calibration spectra of soil samples collected from all over the USA in the 350-2500 nm wavelength range (2151-channels). The challenge was asking to predict the content of three constituents in the samples. Here we will try to predict only two of these constituents, namely the clay-size fraction and the fraction of soil organic carbon. These tasks are noted respectively Shootout-cl and Shootout-soc below. As the regression methods would be too costly to train on the complete set of samples, we picked randomly 700 spectra from the set and did not use the other spectra.

The Table 3.2 gives summary information about the datasets. Further information can also be found in Appendix B.

3.4.4 Outlier detection

As explained in the preliminaries, it is necessary to remove outliers from each dataset in order to make the estimation of the performances more consistent. We have tried to adopt a standard "good sense" approach for detecting the outliers. The procedure is illustrated on the Orange Juice dataset which has two obvious outliers when we look at the original spectra. These original spectra are represented in Figure 3.8(a), and their normalised representation is visible on Figure 3.8(b). On this second subfigure, we have marked the assumed outliers with thick lines. Here is the procedure leading to this identification.

[3]Pedometrics is the application of mathematical and statistical methods for the study of the distribution and genesis of soils.

Name	l-range [nm]	t-range		N_0	D_0
Tecator	850-1050	0.9-58.5	[%]	240	100
Orange juice	1100-2500	0-95.2	[%]	218	700
Apple	1100-2190	8.3-31.1	[N]	337	110
Chimiom-nt	1100-2498	0.2-8.1	[g/Kg]	485	700
Chimiom-c	1100-2498	1-45.7	[%]	334	700
Shootout-cl	350-2500	0.1-90.5	[%]	700 (2761)	2151
Shootout-soc	350-2500	0-53.7	[%]	700 (2501)	2151

Table 3.2: Summary of dataset details. l-range is the wavelength range of the spectra, t-range is their target range, N_0 is the number of samples used in the construction of the regression models (under brackets, the original number of spectra in the dataset), D_0 the initial number of evaluation points.

We first apply a Principal Component Analysis to the spectra and keep only the dominant components such that most of the global variance in the data is still present (we chose to keep 99.5 percent). Using the empirical covariance matrix of the PCA-filtered spectra, we compute the Mahalanobis distance (i.e. the Euclidean distance weighted by the inverse covariance matrix) to the empirical mean of the spectra. In order to improve the identification of outliers, we apply the procedure to the normalised and the first derivative representations. The Mahalanobis distances of the samples for both representations (noted respectively δ_u and δ_d) are visualised with a scatter plot. This is illustrated on Figure 3.8(c). The points in this plot which look atypical for both representations or very atypical from one of these representations are marked as outliers. For this Orange Juice dataset, the three points apart from the rest in the upper right region of the plot were marked as outliers. The second step is to look at the mean and standard deviation components of the original spectra and to visualise them also with a scatter plot. This is shown on Figure 3.8(d) for the Orange Juice spectra. Here also, if some samples look very atypical they are marked as outliers (especially if they are close to be atypical in the Mahalanobis distance analysis). From this view, we spotted four additional outliers. Finally, we look at the distribution of the targets (Figure 3.8(e)) and see if other samples are not lying far from the others, at the limits of the target range. For the Orange Juice dataset, we remark that there are target outliers which were already marked from the other views. In practice, the three analyses are made concurrently. In total seven samples were marked as outliers for the Orange Juice dataset. On the normalised representation in Figure 3.8(b) we can see that the two obvious outliers were detected. The visualisation of the outlier detection procedure for the other datasets is given in the Appendix B.1.

In the end, the decision to mark a sample as an outlier is still subjective, but with little additional knowledge about the measurement procedure there is probably no perfect way to proceed.

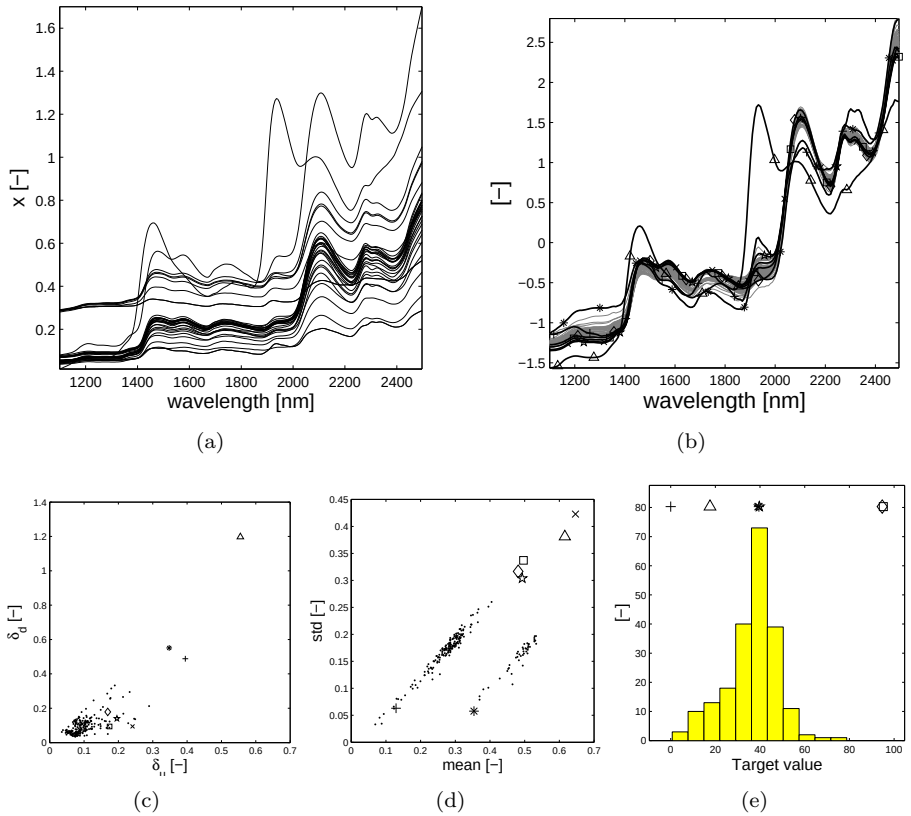


Figure 3.8: Visualisation of *Orange Juice* spectra and the outlier detection procedure. (a) Original representation, (b) normalised representation (outliers marked by thick lines), (c) Mahalanobis distances: normalised vs. first derivative representations, (d) global mean vs. standard deviation of original spectra, (d) target distribution of saccharose content. Markers correspond to the assumed outliers.

3.5 Results and discussion

The results will be analysed using the box plots of Figures 3.9 and 3.10, which show the Normalised Root Mean Squared Error (NRMSE) of the predictions (the normalisation is with respect to the empirical target standard deviation). Each box is drawn from 20 evaluations as explained in Section 3.4.1. The same results are also given in Appendix C.2 by means of a table. To simplify the analysis, we have split the model configurations in four groups, one group for each of the four columns of Figures 3.9 and 3.10.

3.5.1 Representation

On the first column of Figures 3.9 and 3.10, the base configuration $\mathcal{GP}_0$ defined above is constructed from the following four spectra representations: original, normalised, first derivative and second derivative, noted by $\{\mathcal{D}_0, \mathcal{D}_u, \mathcal{D}_{d1}, \mathcal{D}_{d2}\}$.

From the box plot analysis, we observe that the influence of the representation is not as important as we had thought. For all datasets, the performances are quite close, whatever the pre-processing. The Tecator data is an exception as we observe a big improvement for the pre-processed spectra over the original representation. There is no systematic best pre-processing but the first derivative representation does perform consistently well on the seven regression tasks. It is the best representation for four of the tasks: Tecator, Apple, Chimiom-c, Shootout-cl. Each of the three other representations is the winner for one of the other set. We have selected the normalised, original and second derivative representation respectively for the Orange juice, Chimiom-nt and Shootout-soc tasks.

The succeeding model constructions are based on the best representations identified for each task.

3.5.2 PLS-MLP

Looking at the second column of Figures 3.9 and 3.10, we compare the similarity based approach encoded in the Gaussian kernel of the Gaussian Process model with a formulation of the model making use of a dot product between parameters and spectra: the Partial Least Squares and the Multi-Layer Perceptron.

We start with the PLS model. For four of the regression tasks (Tecator, Shootout-cl, Shootout-soc and Orange juice to a lesser degree), we clearly see from the prediction performances the advantage of a nonlinear model like the GP with Gaussian kernel over a linear model like the PLS. On the other hand, this is certainly not always true and we see that the simplicity (and robustness) of the PLS method outperforms the GP for two of the tasks (Apple and Chimiom-c), and their performances are close on the last one (Chimiom-nt). Part of the explanation could be the high intrinsic noise level in the data of these tasks. The higher the noise, the more difficult the identification of pat-

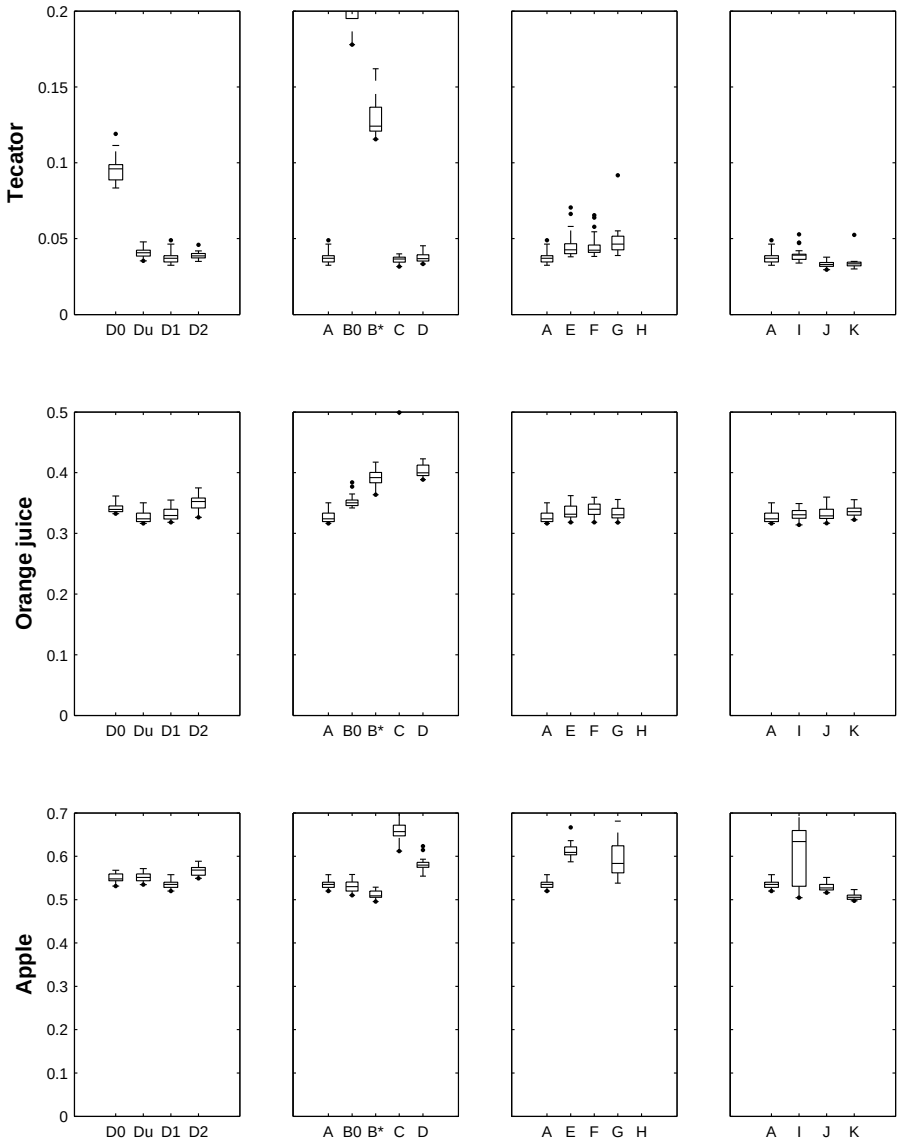


Figure 3.9: Prediction performances (NRMSE criterion) for the Tecator, Orange juice and Apple datasets. The columns correspond to the different aspects compared (see text). Letters identify the model configurations. First column: influence of the representation with the configuration $\mathcal{GP}_0$: 'D0' original, 'Du' normalised, 'D1' first derivative, 'D2' second derivative. For the other columns, the best representation from first column is used. 'A': $\mathcal{GP}_0$, 'B0': PLS on original , 'B*': PLS on best, 'C': MLP-ml, 'D': MLP-cv, 'E': $\mathcal{GP}_\phi$, 'F': $\mathcal{B} - \sigma_x$, 'G': $\mathcal{B} - \sigma_x, \lambda$, 'H': $\mathcal{W} - \sigma_x$, 'I': $\mathcal{GP}_0 - \sigma_x$, 'J': $\mathcal{GP}_{ard}$, 'K': $\mathcal{GP}_{s-ard}$.

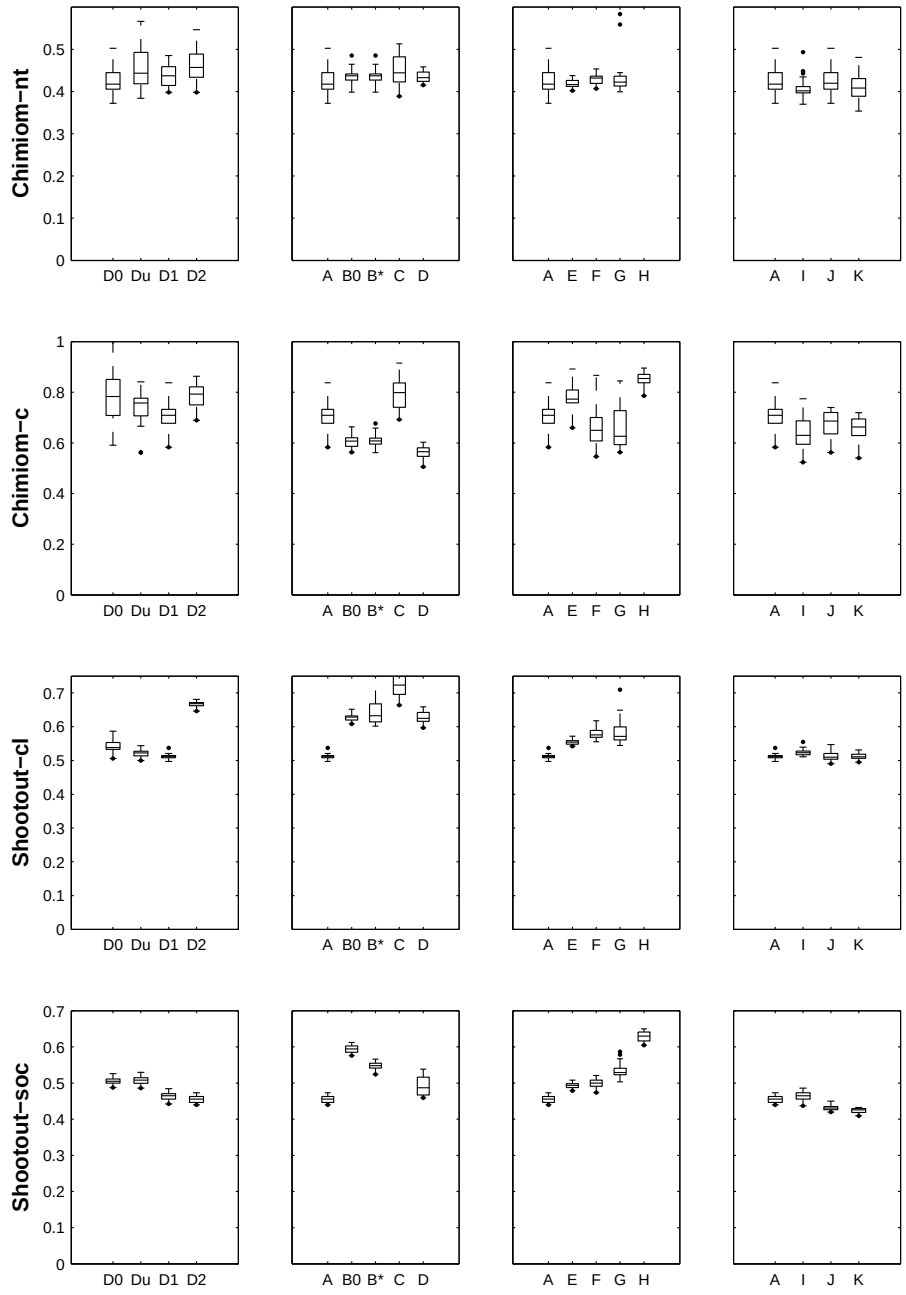


Figure 3.10: Prediction performances (NRMSE criterion) for the Chimiom-nt, Chimiom-c, Shootout-cl, Shootout-soc. See caption of Figure 3.9 for the description of the configuration identifers.

terns, especially nonlinear ones. Note that for the Chimiom-c dataset which appears to have the highest noise level, the samples on the upper part of the target scale are less numerous (Appendix B) and they seem to be quite complicated to predict; these samples have thus a big influence on the measured performances.

Furthermore, it is not surprising that a linear model performs well on (moderately) high-dimensional data. The principle of nonlinear models like the radial basis expansions or kernel methods is to convert the original data in a high-dimensional representation such that in this representation patterns are mainly linear. But spectra are already moderately high dimensional entities and from the arguments just stated, we understand that there is less necessity to pass by another representation.

The influence of the representation on the construction of the PLS model ('B0' vs. 'B*') is similar to what we observed with the $\mathcal{GP}_0$. We do not comment this influence further.

Comparing now the two MLP configurations with $\mathcal{GP}_0$, we observe the difficulty in benefitting from the high flexibility of the MLP networks. Indeed, the MLP outperforms the $\mathcal{GP}_0$ only on Chimiom-c and the performances are comparable on Tecator and Chimiom-nt. For the four other tasks, the $\mathcal{GP}_0$ is clearly performing better. We leave the discussion of the comparison of the MLP-ml and MLP-cv configuration to Subsection 3.5.4.

3.5.3 Gaussian function processing

As described in the overview of the regression methods, there are different ways to make use of the information coming from the processing of the input vectors with a Gaussian radial function (or squared exponential, Eq. (3.3)). The third column of Figures 3.9 and 3.10 shows the comparisons of the methods using this Gaussian function processing, namely the Gaussian Process, the basis expansion and the locally weighted schemes. For this analysis, let us consider only the configurations marked 'A:' $\mathcal{GP}_0$, 'E': $\mathcal{GP}_\phi$ and 'H': $\mathcal{W} - \sigma_x$ (the others are treated in the next analysis).

We first note the complete failure of the locally weighted scheme ('H') for tackling these regression tasks. The box plots associated with this configuration are not visible for most of the tasks but they are in fact somewhere above the vertical axis limit. As discussed in Section 3.2.2, the scheme transforms to a global averaging of the observed targets because with the increasing dimensionality all samples tend to become equally close to the query point.

The difference is not as marked between the Gaussian covariance formulation $\mathcal{GP}_0$, and the Gaussian RBF approach $\mathcal{GP}_\phi$. We observe that the Gaussian covariance formulation performs better than the basis function formulation on 5 out of the 7 regression tasks (and the performances are approximately equal on the two others). Remember that the former is based on an implicit infinite-dimensional feature space while the latter makes use of an explicit finite-dimensional feature space. Apparently, the Gaussian covariance formulates a

regulariser (or prior) over the function space which is more adequate. The origin might be that in contrast to the basis expansion formulation, the regulariser in the Gaussian covariance adapts its strength automatically to the local density of samples [112].

3.5.4 Marginal Likelihood vs. cross-validation

Starting with this chapter, one of the main questions was to know if the Bayesian selection principle used to adjust the hyper-parameters of the regression models performs well for this NIR spectroscopy application. To answer this question, we can compare using Figures 3.9 and 3.10 the performance of two pairs of configurations: 'A': $\mathcal{GP}_0$ with 'I': $\mathcal{GP}_0 - \sigma_x$ and 'E': $\mathcal{GP}_\phi$ with 'F': $\mathcal{B} - \sigma_x$ and 'C': MLP-ml with 'D': MLP-cv. In each case, the pairs of models are completely equivalent except that a ridge trade-off parameter λ adjusted by cross-validation replaces the Gaussian prior parameter(s) α (adjuted on the marginal likelihood).

From the pairwise comparisons 'A'-'I' and 'E'-'F' (all based on a GP formulation), we do not see a systematic advantage in the use of one approach over the other. On most regression tasks, the performances are very close. We see a clear advantage for the cross-validation procedure only on the Chimiom-c task. This is precisely the dataset that has the highest noise level. We have some doubts about the poor performances for the cross-validation procedure on the Apple dataset. We did try to vary the initialisation states and the range over which the cross-validation makes the search to correct the issue but we had no success.

Arguably, the $\mathcal{GP}_0 - \sigma_x$ and $\mathcal{B} - \sigma_x$ configuration still make use of the marginal likelihood for adjusting the other two hyper-parameters (τ, α). This influence of the Bayesian selection principle is suppressed in the 'G': $\mathcal{B} - \sigma_x, \lambda$ configuration since its two unique hyper-parameters are selected by a grid search based cross-validation scheme. Here again, we do not observe an advantage in favour of the resampling selection scheme.

From this analysis, the Bayesian selection scheme within the Gaussian Process formulation looks perfectly valid for this application. Arguably, the number of hyper-parameters (and more fundamentally the size of the hyper-parameter space) is small enough here not to experience over-fitting issues with the marginal likelihood scheme. We will see in the next section if the marginal likelihood continues to behave well when the number of hyper-parameters increases.

However, the same cannot be said for the MLP formulation. Indeed, comparing 'C:' MLP-ml and 'D': MLP-cv, we see that the MLP-ml configuration is failing to deliver competitive predictions for all but the Tecator and Chimiom-nt tasks. There are two origins to the difference with the GP. First, the inner space of parameters of the MLP has probably larger capacity than the GP implicit inner space. A larger inner space means that the optimisation of the marginal likelihood will more easily fall into a bad local optima for which the noise in

the targets is underestimated (Figure 3.3). In addition, the Laplace approximation is probably not always accurately estimating the posterior uncertainty over the weights of the MLP. If the posterior uncertainty is underestimated, the marginal likelihood of the hyper-parameters is penalised by an Occam factor larger than it should be (and conversely if the posterior uncertainty is overestimated). These might be the reasons why the MLP-ml configuration is not performing well.

3.5.5 ARD and smooth-ARD

Our final analysis tries to determine if the marginal likelihood optimisation allows us to further increase the model capacity. More precisely, we look at the Automatic Relevance Determination scheme for the adjustment of the precision parameters in the Gaussian covariance. See [96, 28] for similar analyses. The models considered are thus 'J': $\mathcal{GP}_{ard}$ and 'K': $\mathcal{GP}_{s-ard}$ in Figures 3.9 and 3.10, compared with the base configuration 'A': $\mathcal{GP}_0$.

The difference between the $\mathcal{GP}_0$ and $\mathcal{GP}_{ard}$ is certainly not large but adjusting the precision or equivalently the scale parameters associated with the different components of the pre-processed representation (i.e. $\sigma_*, \sigma_\mu, \sigma_s$) gives a small advantage on average. The regression tasks for which the improvement is visible are the Tecator, Chimiom-c and Shootout-soc.

Allowing further flexibility by means of the smooth-ARD scheme slightly improves the performances of both the $\mathcal{GP}_0$ and the $\mathcal{GP}_{ard}$ configurations for two of the tasks: the Apple and Chimiom-nt. For the other tasks, the higher capacity of the model is well controlled and the performances are comparable with the $\mathcal{GP}_0$ and the $\mathcal{GP}_{ard}$. In [28], the comparison of an isotropic and an ARD Gaussian kernel (within the least-square support vector machine model) on 13 datasets from diverse fields shows that the ARD framework has a clear tendency to over-fit the data. The issue is can be healed by applying a higher level regularisation scheme on the ARD hyper-parameters. The spectra from NIR spectroscopy seem thus to be a favourable type data as over-fitting does not appear so quickly. Possibly, this is due to the high correlation between the spectra (i.e. the intrinsic space of the data is not so high). This also suggest that parts of the spectra are truly non relevant for the prediction and thus the assumption of the smooth-ARD scheme are appropriate.

Table 3.3 gives the ratios of the scale factors $\sigma_\mu, \sigma_s$to σ_*, related to the three components: shape, average curve level m_0 and initial standard deviation s_0. These quantities give an indication of the sensitivity estimated by the ARD scheme of the prediction function to the different components of the curves. From this table, we see that for the Tecator, Apple and shootout-soc tasks the prediction function is mainly influenced by the shape component (as the ratios of the scale parameters are large). This is precisely the tasks for which the performances improve with the ARD (and smooth-ARD) scheme(s).

Figure 3.11 shows the precision curves estimated by the smooth-ARD for all tasks except Tecator which was given in Figure 3.7. It is interesting to see

	σ_μ/σ_*	σ_s/σ_*
Tecator	460	66
Orange juice	2.0	1.1
Apple	34	880
Chimiom-nt	-	-
Chimiom-c	6.3	7.0
Shootout-cl	5.3	5.3
Shootout-soc	8.3	350

Table 3.3: ARD scheme, evaluation of the influence of the different components on the prediction function. Ratio of the scales (σ_μ, σ_s) of the mean curve and standard deviation components to the scale $\sigma_*$of the shape component. The higher this ratio, the less relevant the mean component (first column) or the standard deviation component (second column). In comparison, the ratio is one for the isotropic model $\mathcal{GP}_0$.

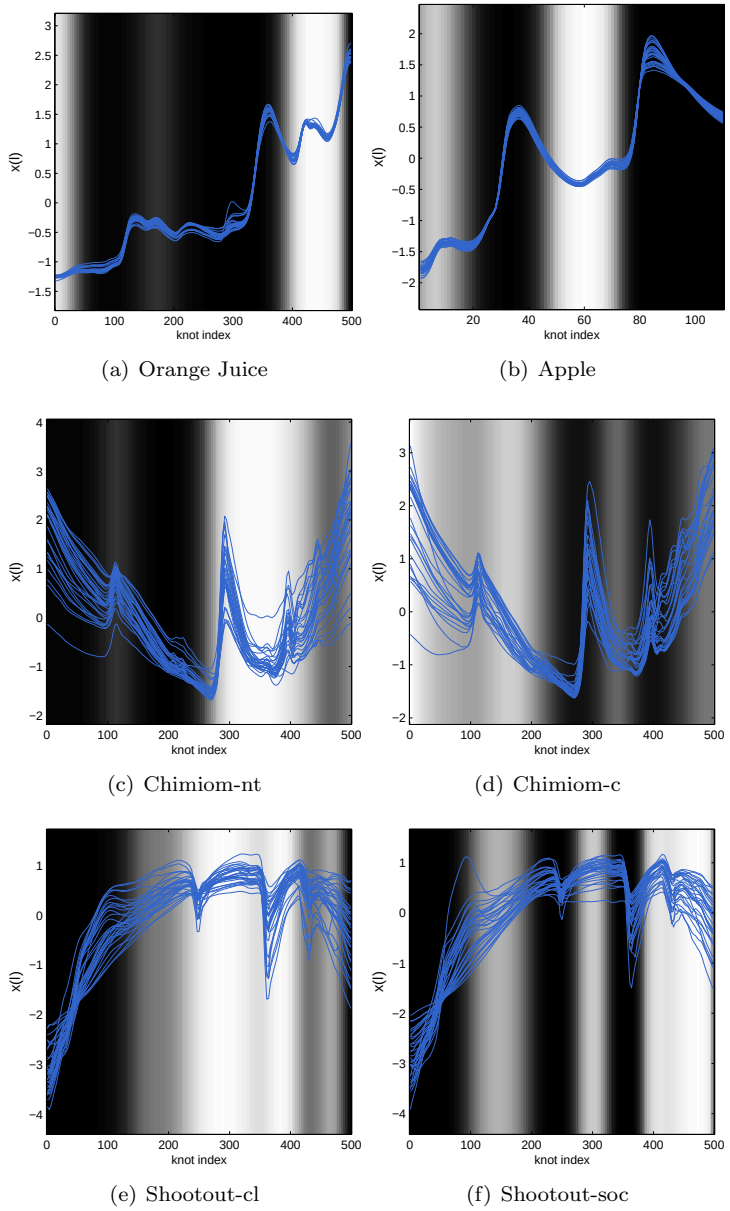
for example that for the Chimiom-nt task the relevant part of the spectra is the upper region of the wavelength range (containing the dominant peak in the mid-range region) while for the Chimiom-c task the relevant part is for the lower wavelength range. Also for the Shootout spectra, we see a clear difference between the relevant par for the two tasks (Shootout-cl and Shootout-soc). Naturally, additional domain knowledge about the region of higher absorption (or reflectance) of the analysed constituents would be useful to reinforce the information returned by the smooth-ARD scheme.

3.5.6 Summary

In short, here are the main results brought by the comparison. First, we have seen that the representation of the spectra has in general only a limited influence on the performances. We would recommend using the first derivative representation as a standard choice. We have seen that the Partial Least Squares is a good starting method in forming an idea of the prediction performances that can be obtained. Furthermore, it is sometimes difficult to improve over this method with a nonlinear model. It is probably wise to control the general adequacy of the more flexible nonlinear configurations with the data by constructing in parallel a Partial Least Squares model.

It was also interesting to observe that the Gaussian covariance formulation (with an infinite dimensional implicit feature space) performs better than the explicit basis expansion formulation on these tasks. This should encourage us to use kernel methods with high-dimensional implicit feature spaces.

The Bayesian selection principle (i.e. the marginal likelihood optimisation) turns out to be as good if not better than the cross-validation scheme for the GP configuration. In practice, both approaches can be difficult to implement properly: the Bayesian selection requires that the initialisation be not too far



(a) Orange Juice (b) Apple

(c) Chimiom-nt (d) Chimiom-c

(e) Shootout-cl (f) Shootout-soc

Figure 3.11: Smooth-ARD precision curves $u(l)$ for the different regression tasks. Lighter shadings indicate the relevant parts of the spectra for the prediction. The spectra are in their normalised representation.

from an appropriate optimum of the marginal likelihood. It is not always easy to avoid the convergence to bad local optima. On the other hand, it is no easier to determine the range of hyper-parameter(s) values to be covered by the cross-validation approach and how many repetitions are necessary (and still feasible from the computational point of view). We noted that the Automatic Relevance Determination scheme is applicable and slightly improves performance in general. More interestingly, the smooth-ARD scheme allows us to have a rapid analysis of the relevant parts of the spectra. However, we also observe the limits of the Bayesian model selection principle within the Multi-Layer Perceptron approach. Further analysis should be able to determine whether the problem is due to the capacity of the model rather than the failure of the Laplace approximation.

Chapter 4

Collaborative Filtering

Collaborative Filtering is a data analysis task appearing in many challenging applications, in particular for data mining in Internet and e-commerce. Collaborative Filtering can often be formulated as identifying patterns in a large and mostly empty rating matrix. We will take the view that ratings represent the opinions expressed by users on a set of items. Collaborative Filtering is then often part of a recommendation procedure.

After an introduction to the topic of Collaborative Filtering, we discuss two probabilistic models constructing *mixtures of latent profiles*. The principle of these models is to look for a decomposition of the users' ratings which could intuitively correspond to the particular interests of the users. Similarly to the models presented in Chapter 2, uncertainty in these models is encoded by latent indicator variables and an Expectation-Maximisation (EM) algorithm is used to adjust the main parameters. We will show how the probabilistic approach can provide a degree of interpretation about the users' interests. However, finding interpretation is fundamentally subjective and moreover there are some limits to the utility of the posterior analysis as we will see.

In the second part of the chapter, we propose an alternative model called the interlaced Generalized Linear Model (GLM) [37, 38]. This approach is based on a factorisation of the rating matrix and uses probabilistic modelling to represent uncertainty in the ratings. Intuition about the rating process can also be encoded in the model by means of prior distributions, but it is no longer possible to look for interpretation from the posterior analysis of the model. However, the main advantage over the mixture models is its simple formulation relying on the Generalized Linear Model formalism [90] and making it applicable to large datasets. We will see that an interesting feature of the interlaced GLM model is an ability to encode very simply the dominant patterns of the rating matrix. The model is constructed on three public domain datasets and its performance compared with the mixtures models.

4.1 Preliminaries

We start the section with a general description of the Collaborative Filtering task, its main applications as well as the current difficulties faced in making the method more useful. An overview of the literature on the topic is then given and finally the main notations used in the chapter are presented.

4.1.1 Introduction to Collaborative Filtering

The expansion of the Internet has brought many new challenges regarding the processing of the huge amounts of data stored and exchanged on the web. To help people find their way toward useful information, data mining engines are required. Famous examples are the search engines from Google and Yahoo that are able to propose web pages most relevant to some keywords. The Internet has also given rise to e-commerce and it is now possible to purchase the goods and services of many companies online. In parallel with e-commerce, advertisement on the web has also become a very lucrative business.

To improve the diffusion of information over the web (e.g. to propose better links related to a search, to advertise the right products to potential customers...) the mining engines need to have a pro-active attitude in trying to foresee the information a user might need before he makes an explicit request for it. In other words, the search for information should be made more automatic and personalised. Of course, for an engine to personalise the information given to a user, it needs to know a little about this user. For example, the engine needs to have access to the records of previous interactions the user has had with the system (i.e. the pages he has visited, the queries he has made...). The goal is to identify the user's interests from his past behaviour.

Collaborative Filtering is the task of identifying (filtering) the interests of the users from the similarity of their behaviour with other users (collaboration). It is based on the idea that users with similar behaviours in the past should have similar behaviours in the future. Note that the specificity of the CF approach is the collaboration, i.e. merging information coming from several users. Using content information could be an alternative to identify the users' interests. For example, if a user searches for a book on a particular subject - the subject is the content information - the engine could let him know about other books on the same subject.

The CF task can be formalised as follows. First, the system is associated with a set of items (e.g. the web pages, the products of a company...). The users can have interactions with the items, visit the corresponding pages, query for the corresponding products and give opinions about them. Interactions between users and items will be referred as ratings because they are an expression of users' interests. It is usual to distinguish between implicit and explicit ratings. Implicit ratings are interactions that do not directly encode an appreciation, e.g. consulting an item. On the other hand, explicit ratings are real appreciations returned by the user for the items he knows about. Typically, the system offers

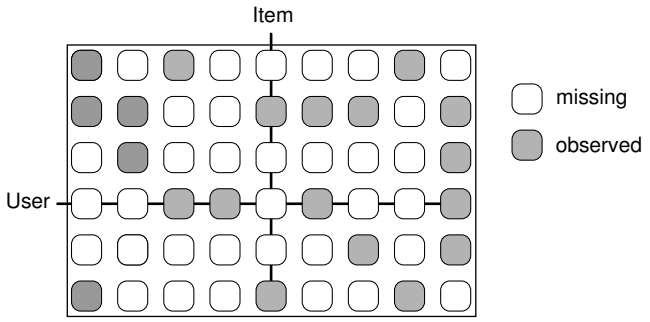


Figure 4.1: Illustration of a small rating matrix

the user the opportunity to rate any item on a ordinal scale of 5 or 10 grades. It is more demanding to collect explicit ratings, but they are usually more informative about the users' interests.

All of the collected ratings can be stored in a rating matrix, each row corresponding to a user and each column to an item belonging to the system (Figure 4.1). It is common to have systems with hundreds or even thousands of items. Consequently, each user can express ratings only for a small subset of all items and the rating matrix is mostly empty, i.e. filled with missing values. Note that this is not identical to a sparse matrix, mostly filled with zero values.

Formally speaking, the Collaborative Filtering task amounts to identifying patterns in this large rating matrix containing many missing entries. As we discussed in the Chapter 1, the concept of pattern identification is vague because it depends on the final objectives. The main objective can be to have some understanding of the different personalities of the users and to spot some correlated behaviours (e.g. those users favourable to these items seem to favour also these other items). When the search for understanding is the main objective, CF should try to represent the rating matrix by a compact and interpretable model. There is thus a trade-off to be reached between the simplicity of the model and the adequacy of the model with the data (which is measured by the learning criterion). In addition, linking the CF task with the content information like the items' categories (e.g. for books: science-fi, biography...), the prices of the products and other information is essential to help in the quest for understanding.

The second use of Collaborative Filtering is for recommendation purposes. We have already sketched the idea of recommendation, a recommender system is a data mining engine that tries to inform users about items they are likely to be interested in. This can be achieved by predicting the missing entries of the rating matrix. In particular, when the rating matrix stores explicit ratings, the items recommended by the engine are those receiving the highest predictions. Prediction of missing entries can be cast in a supervised learning framework. It must be added that in parallel with good recommendations, users want to know

what makes the system think that they should like the recommended items. It is thus useful to be able to access and interpret the model mechanisms which lead to the recommendations.

There are famous examples of recommender systems. The website Amazon.com for example always proposes links to products related to the last web pages visited. For this example, the recommender system is based on both the CF scheme and additional content information. Another illustration of a recommender system is used by the Netflix movie rental company. To improve their service, Netflix has a recommender system which makes propositions to the customers of movies that should fit their tastes. The company has launched (October 2006) a 1 million dollar prize competition on the prediction of explicit ratings collected from their customers. The goal is to improve (by more than 10 percent) the quality of prediction of their current recommender system.

Recommender systems are an illustrative application for CF. The goal of identifying patterns in a large matrix appears in many other applications. In the field of Information Retrieval, CF can be used to mine text corpora and characterize the common topics covered by a set of documents [119]. In this case, we are not referring to users' behaviours and items' preferences, but rather to document content and word saliency. The entries of the matrix store the number of occurrences of a word in a document. The matrix is thus very sparse as most words are not present in a given document. Another field where large contingency matrices appear is in microarray analysis. Here the contingency matrix collects the response levels of a set of genes (the users) to some environmental conditions (the items). One way to identify patterns in the microarrays is by bi-clustering, i.e. trying to simultaneously form groups of genes having similar responses to the same conditions [84]. This procedure is related to the mixture models described in Section 4.2. We will continue to refer to *users* and *items* terms because they are appropriate for the datasets used in the experiment section. It should be clear that depending on the context of the application other terms could be more appropriate.

The main focus of the comparison presented in Section 4.4 is the prediction performances for explicit ratings. This scheme allows the comparison of different models based on quantitative criteria. The quality of Collaborative Filtering for implicit ratings can also be evaluated with quantitative criteria, but it is less evident that these criteria are a good match to the users' preferences. Evaluation of Collaborative Filtering when the task is to identify interpretable patterns is even more complicated. In fact, there are probably only subjective reasons to prefer one model over another as interpretation cannot be quantified, at least when we have no additional expert knowledge. We have already discussed this problem in the previous chapters. For illustrative purpose, we will show how some interpretation can be obtained from the mixture models presented in Section 4.2.

In general, prediction of explicit or implicit ratings is a sub-task of a recommender system application. The exact link between prediction and the actual items to be recommended is never straightforward because recommendation is

usually expressed by an ordered list instead of the absolute values of the prediction. Also, items with less publicity around them are in general more valuable. Indeed, it makes little sense to recommend items (like blockbuster movies) that most people already know about. This is the concept of serendipity, uncovering what is not at first obvious. For a review of the difficulties relating to the evaluation of a recommender system, see [56].

4.1.2 Overview of methods

Many approaches were proposed to perform Collaborative Filtering. Only a very brief overview of the domain is given here. Early methods relied on a nearest neighbour principle: the users should get advice from the most similar users [113, 23], according to their common ratings. The main challenge is to define a good similarity between users in particular when they have only a small number of rated items in common. The similarity can also be defined from the dual view of items. In this case, it is expected that users are interested in items similar to those to which they have already given good ratings. When the contingency matrix is recognised as a bipartite graph, the definition of similarity can be formalised in terms of a distance in a graph [45]. Concepts such as random walks and mean transit times between nodes of the graph are then used to define the similarity.

A second approach to tackle Collaborative Filtering is referred in the literature to the model-based approach. The main idea is that the variability of users' rating behaviours can be described by a small set of typical rating profiles. Mixture models and in particular the *Aspect Model* [60] (also known as the *probabilistic Latent Semantic Indexing*) are simple and illustrative of this approach. The intuition behind these mixtures is that each rating expressed by a user can be attributed to one cause (or aspect). We will describe the standard mixture models in the next section.

Other interesting but more complicated variants of these mixture models were proposed in [85]. For example, the Multiple Multiplicative Factor model tries to encode multiple memberships to the mixture components. Unfortunately, it looses the simplicity of an exact Expectation-Maximisation learning procedure. There are also extensions in the field of non-parametric Bayesian statistics, notably with the introduction of a Dirichlet process in the model [111, 149, 150]. Non-parametric models are convenient from the modelling point of view as the number of components does not need to be fixed in advance; this number is automatically estimated and appears as a mode of the posterior distribution of the model parameters. It is also possible to use non-parametric models to encode multiple memberships with the so-called Chinese restaurant and Indian buffet processes. These kinds of model seem suitable for text mining, e.g. [19].

Collaborative Filtering can also be handled by factorisation of the rating matrix. The goal is to find two low rank factor matrices minimising a reconstruction error, i.e. $\mathbf{R} \approx \Phi\Omega^\top$, where $\mathbf{R}$ is the $N \times M$ rating matrix, N is the

number of users, M is the number of items; Φ and Ω are the factor matrices (respectively $N \times K$ and $M \times K$). The factor matrices Φ and Ω are full (no missing entries), so the product of these factor matrices can compute any entry of the rating matrix. This property is sometimes required by the evaluation procedure (see Section 4.4)[1]. The choice of the reconstruction error and the way the factor matrices are constrained are critical to obtain an efficient algorithm. One of the most prominent factorisation is of course the Singular Value Decomposition (SVD). It was originally proposed for text mining in [34]. For this factorisation, the reconstruction error is simply given by the Frobenius norm, $\|\mathbf{R} - \Phi\Omega^\top\|_F$, and the regularisation is adjusted by the common size K of the factor matrices. However, in order to handle missing entries correctly, it is necessary to resort to a non-standard SVD algorithm, e.g. [22, 126]. It is sometimes necessary to impose a non-negativity condition on the factor matrices in order to avoid violation of some constraints or simply to have interpretable factor matrices. The task is called a *non-negative matrix factorisation* in the literature [103]. Another way to define the reconstruction error was proposed with the concept of maximum-margin factorisation in [127, 33]. This factorisation has shown to perform very well on different datasets.

The Frobenius norm appearing in the SVD reconstruction error is a square error loss. It is known that a Gaussian noise assumption fitted by maximum likelihood would also minimise the square error. Closely related to the approach in [31], the factorisation of the rating matrix could be formulated with the assumption of other noise distributions. In particular, one could work with distributions of the exponential family as they are easy to handle and can represent a large variety of noise models. This is exactly the idea followed in the interlaced Generalized Linear Model (GLM) [38] approach and presented in Section 4.3. The advantage of using a probabilistic formulation to model the uncertainty in the ratings is that different model configurations - encoding different assumptions on the noise or different constraints on the factor matrices - can be tested without modifying the general learning procedure. Moreover, we will show that the learning procedure is very efficient as it requires only little memory and implies the resolution of many small GLM regression problems.

4.1.3 Notation

In the following, the set of items will be denoted $\mathrm{Y} = \{y_m\}_{m=1}^M$ and the set of registered users $\mathrm{U} = \{u_n\}_{n=1}^N$. Thus, the rating matrix $\mathbf{R}$ has dimension $N \times M$. The rating expressed by user u_n for the item y_m is noted r_{nm}. It is useful to have a subscript notation for the set of observed ratings only. For this purpose, the observed ratings are stored in a list of triplets $\mathcal{D} = \{(u, y, r)_l\}_{l=1}^L$. Each time the subscript l is used, it refers to an element of the list. For example, we will note u_{n_l} the user who made the lth rating. It is useful also to have a

[1] Note that some models are not able to return a rating for any user-item pair when there is not sufficient interaction between the two. This can also be viewed as a positive feature as the model will not predict a rating when a minimum information is missing.

compact notation for sets of indices. For example $l \in \mathrm{L}_n^u$ represents the set of indices such that the ratings r_l was made by the user u_n.

4.2 Mixture of latent profiles

In this section, we describe the probabilistic mixtures of latent profiles models for Collaborative Filtering. The idea is to represent the rating behaviour of the users by a simple generative model. This approach has the advantage of encoding some general intuition about the patterns in the data. From the posterior analysis of the model parameters, we will see if this intuition corresponds to some interpretable patterns.

We will look at two popular versions of this probabilistic mixture approach. The first one assumes that there are groups of users having essentially the same rating behaviour. This model is sometimes called in the literature a Bayesian clustering of the users or a mixture of multinomials. We prefer to call it the *latent profile clustering* model. The concept of profile will be defined in the next subsection. The second model allows each user to have its own rating profile (or behaviour) but this profile is constrained to be formulated as a mixture of a few typical rating profiles. The principle is thus much closer to a linear subspace projection (see Chapter 2). The intuition behind this formulation is that the typical rating profiles common to all users identify specific interests (or aspects) over the item set (e.g. for a movie database, it might be an interest for action movies, for a particular actor...). The model is also known as the *Aspect Model*, or the *probabilistic latent semantic indexing* [59, 60]. It is essentially a *latent profile subspace* model.

We start with the definition of the rating profiles. The actual algorithms for adjusting these rating profiles are described in the following subsections. The mathematical developments of this section are very similar to those in Chapter 2.

4.2.1 Rating profiles

In practice, a rating profile specifies a distribution over the ratings on the complete item set, $P(\mathbf{r}, \mathbf{y}|\boldsymbol{\theta})$, where $\mathbf{r} = [r_1, \ldots, r_M]^\top$ and $\mathbf{y} = [y_1, \ldots, y_M]^\top$ with $y_m = 1$ indicating that the mth item has received a rating. For items not rated, $y_m = 0$ and $r_m = \varnothing$. The distribution is parameterised by $\boldsymbol{\theta}$. For the prediction of explicit rating, we usually know for which item a prediction must be made and it is thus sufficient to define a conditional rating profile $P(\mathbf{r}|\mathbf{y}; \boldsymbol{\theta})$. We will refer to this second configuration as the *force prediction* scheme while $P(\mathbf{r}, \mathbf{y}|\boldsymbol{\theta})$ will be called a *free prediction* scheme [60]. We will come back later on the consequences of this distinction.

As described in the preliminaries, the most basic type of rating is a binary information: an interaction was observed between user u_n and item y_m, or no interaction was observed. For this type of rating, the rating profile is defined

by $P(\mathbf{y}|\boldsymbol{\theta})$, the probability of having a rating (i.e. an observed interaction) for each item of the set. One possibility could be to express this rating profile by means of the Bernoulli distribution:

$$P(\mathbf{y}|\boldsymbol{\theta}) = \prod_m \mathcal{B}e(y_m|\beta_m) = \prod_m \beta_m^{y_m}(1-\beta_m)^{1-y_m} \ . \tag{4.1}$$

The set of parameters is $\boldsymbol{\theta} \equiv [\beta_1, \ldots, \beta_M]^T$, with $0 \leq \beta_m \leq 1$; β_m represents the probability that a user had an interaction with item y_m. The main problem with this formulation of the rating profile is that all items appear in the computation of the profile likelihood. For very large datasets with thousands of items and users, this is not computationally manageable

An alternative to the Bernoulli encoding is the multinomial distribution (Appendix A.1). The multinomial has an additional constraint between the parameters: $\boldsymbol{\theta} \equiv \boldsymbol{\beta} = [\beta_1, \ldots, \beta_M]^T$ with $\sum_m \beta_m = 1$ and $\beta_m \geq 0$. The likelihood of the profile is now given by

$$P(\mathbf{y}|\boldsymbol{\theta}) \propto \mathcal{M}n(\mathbf{y}|\boldsymbol{\beta}, V) = V! \prod_m \beta_m^{y_m} = V! \prod_{y_m=1} \beta_m \ , \tag{4.2}$$

where $V = \sum_m y_m$ is the number of rated items. Interestingly, for the multinomial distribution the un-rated items do not contribute directly to the likelihood. This makes the computation of the likelihood much faster than the Bernoulli formulation and applicable on large rating matrices. In fact the information about un-rated items is summarised in the normalisation constraint. The proportionality sign in Eq. (4.2) indicates that a normalisation factor is required to take into account the constraint that $y_m \in \{0, 1\}$. However, this factor does not influence the optimisation of the parameters.

More generally, the multinomial distribution is appropriate for ratings representing counts, i.e. $r \in \{0, 1, 2, \ldots\}$. Similarly to the previous expression, we have

$$P(\mathbf{r}|\boldsymbol{\theta}) = \mathcal{M}n(\mathbf{r}|\boldsymbol{\beta}, \sum_m r_m) \tag{4.3}$$

The vector $\mathbf{y}$ is absent in this expression as the information about missing and observed ratings is already encoded in $\mathbf{r}$. Typically, this rating profile definition is used for document mining applications. In this case, the items correspond to the words of the vocabulary and the ratings are the counts for each word in the document.

The principle for defining rating profiles remains the same with explicit ratings, i.e. when ratings encode the opinions of the users about the items. Explicit ratings can take a value in an ordered set, either discrete $r \in \{1, \ldots, r_{max}\}$, or continuous, $r \in [r_{min}, r_{max}]$. As noted above, we will represent missing ratings with $r = \varnothing$. Let us consider first the simpler case where we only require a conditional rating distribution $P(\mathbf{r}|\mathbf{y}; \boldsymbol{\theta})$ (i.e. the force prediction scheme). One could define the profile for explicit ratings with Gaussian distributions for example. This is parameterised by a mean and a standard deviation for each item rating distribution, i.e. $\boldsymbol{\theta} \equiv \{(\mu_m, \sigma_m)\}_{m=1}^M$, and the likelihood is

$$P(\mathbf{r}|\mathbf{y};\boldsymbol{\theta}) = \prod_{y_m=1} \mathcal{N}(r_m|\mu_m,\sigma_m) \tag{4.4}$$

where the product goes only on the set of observed ratings. In practice, the ratings are always constrained to take on a value in a bounded interval and it then makes sense to use a bounded distribution like the Beta distribution (which has also two parameters). Also, when the ratings can be chosen from a discrete set, a binomial distribution or a multinomial distribution would be more natural than the Gaussian. Defining a rating profile with a continuous rating distribution when the rating values are discrete requires some precautions to avoid the likelihood degenerating to an infinite value. One way to do that with the Gaussian distribution is by imposing a lower bound on the value that the standard deviation can take.

Now, considering the free prediction scheme, i.e. $P(\mathbf{r},\mathbf{y}|\boldsymbol{\theta})$, one must combine an implicit and an explicit rating profile. The standard choice is to combine the multinomial distribution for the missing-observed implicit patterns and Gaussian distributions for the explicit rating distributions. Hence, we have $\boldsymbol{\theta} \equiv \{(\beta_m,\mu_m,\sigma_m)\}$ and the likelihood of this profile is

$$\begin{aligned} P(\mathbf{r},\mathbf{y}|\boldsymbol{\theta}) &= P(\mathbf{r}|\mathbf{y};\{(\mu_m,\sigma_m)\}) \cdot P(\mathbf{y}|\boldsymbol{\beta}) \\ &\propto \prod_{y_m=1} \beta_m \mathcal{N}(r_m|\mu_m,\sigma_m)\ . \end{aligned} \tag{4.5}$$

At this stage, it is not easy to see why one should prefer the free prediction scheme over the forced prediction scheme. In fact, this choice depends on the way the model is evaluated or used. If the goal is only to accurately predict ratings for (user, item) pairs, i.e. to have a good estimate of $P(r|u,y)$, it is probably of little use to take into account the missing ratings in the model. On the other hand, it is sometimes useful to evaluate the probability $P(r,y|u)$ of observing a particular item jointly with a rating value r. This could be used to increase the serendipity of recommendation by recommending items having high predicted ratings but low predicted probability of being observed (e.g. movies that not so many people went to see but still received very positive critics from those who have viewed these movies).

4.2.2 Latent profile clustering

As described above, the idea of the latent profile clustering is that each user is assumed to rate according to one rating profile among a set of typical rating profiles. Let us suppose that there are K rating profiles in the model (for the moment, K is fixed *a priori*). Each profile has its own parameterisation, hence the complete set of profile parameters is $\boldsymbol{\theta} \equiv \{\boldsymbol{\theta}_k\}_{k=1}^K$. As in the derivation of the models in Chapter 2, the membership of a user u to the profiles will be represented by an indicator vector $\mathbf{z} = [z_1,\ldots,z_K]$ and $z_k = 1$ if the user follows

the kth profile, otherwise $z_k = 0$. Then, the likelihood of the membership variable of a user and the profile parameters is simply

$$P(\mathbf{r}, \mathbf{y}|u; \mathbf{z}, \boldsymbol{\theta}) = \prod_k P(\mathbf{r}, \mathbf{y}|\boldsymbol{\theta}_k)^{z_k} \ . \tag{4.6}$$

We adopt here and for the subsequent developments the free prediction formulation since the forced prediction scheme can be treated as a subtask (see below).

There are different ways to view the membership variables. The first way is to see them as parameters of the model such that the global likelihood criterion to be optimised is

$$\{\hat{\mathrm{Z}}, \hat{\boldsymbol{\theta}}\} = \underset{\{\mathrm{Z}, \boldsymbol{\theta}\}}{\operatorname{argmax}} P(\mathbf{R}, \mathbf{Y}|\mathrm{U}; \mathrm{Z}, \boldsymbol{\theta}) = \prod_n \prod_k P(\mathbf{r}_n, \mathbf{y}_n|\boldsymbol{\theta}_k)^{z_{nk}} \tag{4.7}$$

where we noted the set of membership variables $\mathrm{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$, the vector $\mathbf{r}_n$ contains the ratings of the nth user and $\mathbf{y}_n$ is the corresponding indicator vector of observed-missing interactions. This formulation is a hard clustering scheme because the ratings of a user can only affect the optimisation of a single rating profile at a time. An algorithm similar to k-means could be used to find a (local) optimum of the clustering scheme.

Rather than this hard clustering scheme, we will prefer here to view the set of membership variables as latent variables that must be marginalised over. Hence, we need to specify a prior over these variables. The standard choice for such mixture models without additional knowledge is to take a common multinomial prior $P(\mathbf{z}|\boldsymbol{\pi})$, where $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_K]$ and π_k is the prior probability that a user follows the kth rating profile. The marginal likelihood criterion is slightly different from Eq. 4.7

$$\begin{aligned} \{\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\pi}}\} &= \underset{\{\boldsymbol{\theta}, \boldsymbol{\pi}\}}{\operatorname{argmax}} \sum_{\mathrm{Z}} P(\mathbf{R}, \mathbf{Y}|\mathrm{U}; \mathrm{Z}, \boldsymbol{\theta})\, P(\mathrm{Z}|\boldsymbol{\pi}) \\ &= \underset{\{\boldsymbol{\theta}, \boldsymbol{\pi}\}}{\operatorname{argmax}} \prod_n \sum_k P(\mathbf{r}_n, \mathbf{y}_n|\boldsymbol{\theta}_k)\pi_k \ . \end{aligned} \tag{4.8}$$

The summation on the first line of this expression stands for the marginalisation over the membership variables. The principal advantage of the latent view over the hard clustering scheme is first to make the optimisation a little more robust to bad local optima thanks to the marginalisation (although the issue still exists). More importantly, the advantage is to enter the Bayesian framework benefitting from its nice features (i.e. posterior analysis, modularity, extension of a model...).

The technique to train the parameters $(\boldsymbol{\theta}, \boldsymbol{\pi})$ is already well known (Chapter 2); we will derive an EM algorithm. Making use of the Bayes formula, the posterior distribution over the latent membership variables is

$$Q(\mathrm{Z}) \equiv P(\mathrm{Z}|\mathbf{R}, \mathbf{Y}; \boldsymbol{\theta}, \pi) \propto \prod_n \prod_k P(\mathbf{r}_n, \mathbf{y}_n|\boldsymbol{\theta}_k)^{z_{nk}} \cdot \pi_k^{z_{nk}} \ . \tag{4.9}$$

The normalisation factor is the marginal likelihood from Eq. (4.8). It is interesting to see that the posterior factorises over the user set: $Q(\mathrm{Z}) = \prod_n Q(\mathbf{z}_n)$. Looking now at the log-complete likelihood,

$$\begin{aligned}\log P(\mathbf{R}, \mathbf{Y}, \mathrm{Z}|\mathrm{U}; \boldsymbol{\theta}, \boldsymbol{\pi}) &= \log P(\mathbf{R}, \mathbf{Y}|\mathrm{U}; \mathrm{Z}, \boldsymbol{\theta}) + \log P(\mathrm{Z}|\boldsymbol{\pi}) \\ &= \sum_n \sum_k z_{nk} \left(\log P(\mathbf{r}_n, \mathbf{y}_n|\boldsymbol{\theta}_k) + \log \pi_k\right) \ , \end{aligned} \tag{4.10}$$

we see that taking the expectation of this expression with respect to the posterior distribution of the latent variables requires us to compute the so-called *responsibilities*:

$$\rho_{nk} \equiv \mathbb{E}_Q\{z_{nk}\} \propto P(\mathbf{r}_n, \mathbf{y}_n|\boldsymbol{\theta}_k) \cdot \pi_k \tag{4.11}$$

for $n = 1 \dots N$ and $k = 1 \dots K$, where the proportionality acknowledges the normalisation constraint $\sum_k \rho_{nk} = 1$. The computation of these responsibilities is the E-step of the algorithm. It is concernd with the marginalisation over the latent variables in (4.8).

The expected log-complete likelihood with the current value of the parameters is thus

$$\mathbb{E}_Q \left\{\log P(\mathbf{R}, \mathbf{Y}, \mathrm{Z}|\mathrm{U}; \boldsymbol{\theta}, \boldsymbol{\pi})\right\} = \sum_n \sum_k \rho_{nk} \left(\log P(\mathbf{r}_n, \mathbf{y}_n|\boldsymbol{\theta}_k) + \log \pi_k\right) \ . \tag{4.12}$$

The M-step consists of updating the parameters $\boldsymbol{\pi}$ and $\boldsymbol{\theta}$ in order to maximise this quantity. As both types of parameters appear in different terms of the log-complete likelihood, their update can be done separately. The maximisation with respect to $\boldsymbol{\pi}$ is analytic; the update is given by

$$\pi_k \leftarrow \frac{1}{N} \sum_n \rho_{nk} \tag{4.13}$$

for $k = 1 \dots K$.

Depending on the rating distribution used in the profile definition (e.g. Bernoulli, multinomial, Gaussian...) we are able to make an analytical update of the parameters $\boldsymbol{\theta}$ or only a partial update with a gradient ascent step. Let us consider the case of the multinomial-Gaussian formulation proposed in (4.5). The optimisation can be further divided as

$$\log P(\mathbf{r}_n, \mathbf{y}_n|\boldsymbol{\theta}_k) = \log P\left(\mathbf{r}_n|\mathbf{y}_n; \{(\mu_{mk}, \sigma_{mk})\}\right) + \log P(\mathbf{y}_n|\boldsymbol{\beta}_k) \ . \tag{4.14}$$

For the free prediction term, we obtain the analytic update

$$\beta_{mk} \leftarrow \frac{1}{L_k} \sum_{n \in \mathrm{N}_m^y} \rho_{nk} \ , \tag{4.15}$$

where $L_k = \sum_n \rho_{nk}$ is the equivalent number of ratings estimated to come from the kth profile and N_m^y represents the set of indices n such that user u_n has

rated (or had an interaction with) item y_m. The second term in (4.14), which corresponds to the forced prediction part, can also be updated analytically

$$\mu_{mk} \leftarrow \frac{1}{L_{mk}} \sum_{n \in \mathrm{N}_m^y} \rho_{nk} r_{nm} \tag{4.16}$$

$$\sigma_{mk}^2 \leftarrow \frac{1}{L_{mk}} \sum_{n \in \mathrm{N}_m^y} \rho_{nk} (r_{nm} - \mu_{mk})^2 \tag{4.17}$$

where $L_{mk} = \sum_{n \in \mathrm{N}_m^y} \rho_{nk}$. If instead of the Gaussian, we use a binomial distribution (as in the experiments below) to represent the distribution over discrete ratings $r \in \{0, \ldots, r_{max}\}$, the update of the *probability of success* parameters (Appendix A.1) is

$$\mu_{mk} \leftarrow \frac{\sum_{n \in \mathrm{N}_m^y} \rho_{nk} r_{nm}}{L_{mk} \cdot r_{max}} \ . \tag{4.18}$$

Since the responsibilities are dependent on the value of the profile parameters, the E-steps and M-steps must be iterated in order to converge to a local minimum of the marginal likelihood.

When one is interested in the forced prediction scheme, only the first term on the right hand side of (4.14) will appear in the log-complete likelihood. Similarly, in modelling the missing-observed patterns, only the second term of this same expression is present.

In Section 4.4, it is necessary to evaluate for the forced prediction scheme the rating distribution of some user-item pairs (u_n, y_m); this is given by

$$P(r_{nm} | y_m = 1; \mathbf{r}_n, \mathbf{y}_n; \boldsymbol{\theta}, \pi) = \sum_k \mathcal{N}(r_{nm} | \mu_{mk}, \sigma_{mk}^2) \rho_{nk} \ . \tag{4.19}$$

Note that the contribution of previously observed ratings is summarised in the responsibilities.

4.2.3 Aspect Model (or Latent profile subspace)

The Aspect Model assumes that each user follows his own rating profile which is constrained to be a mixture of the K base rating profiles specified by $\{\boldsymbol{\theta}_k\}$. Instead of a latent membership variable $\mathbf{z}_n$, the rating behaviour of a user u_n is represented by a mixture variable $\boldsymbol{\pi}_n = [\pi_{n1}, \ldots, \pi_{nK}]$, where $\sum_k \pi_{nk} = 1$ and $\pi_{nk} \geq 0$. The set of mixture variables for all users will be noted $\Pi = \{\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_N\}$. For this model, the likelihood of the mixture variable of a user and the the profile parameters is

$$P(\mathbf{r}_n, \mathbf{y}_n | u_n; \boldsymbol{\theta}, \boldsymbol{\pi}_n) = \prod_m \sum_k P(r_{nm}, y_{nm} | \theta_{mk}) \pi_{nk} \ . \tag{4.20}$$

In this expression, we suppose that all items contribute directly to the likelihood. As discussed in Section 4.2.1, this is not manageable for large rating matrices. We will thus require that only the observed ratings should contribute to the likelihood, i.e.

$$P(\mathbf{r}_n, \mathbf{y}_n | u_n; \boldsymbol{\theta}, \boldsymbol{\pi}_n) = \prod_{l \in \mathrm{L}_n^u} \sum_k P(r_l, y_{m_l} | \theta_{m_l k}) \pi_{nk} \ , \tag{4.21}$$

where L_n^u is the set of rating indices from the list notation (confer Section 4.1.3) such that the rating r_l belongs to the user u_n, and where the index m_l refers to the index of the item which received the lth rating.

In contrast to the previous model, we will not try to marginalise these mixture variables Π since the marginalisation cannot be done analytically, but rather keep them as parameters to be adjusted using the likelihood of the model. In the next subsection, we briefly discuss approximations for the marginalisation of these mixture parameters. The optimisation of the model likelihood is expressed by

$$\begin{aligned} \{\hat{\boldsymbol{\theta}}, \hat{\Pi}\} &= \operatorname*{argmax}_{\{\boldsymbol{\theta}, \Pi\}} P(\mathbf{R}, \mathbf{Y} | \mathrm{U}; \Pi, \boldsymbol{\theta}) \\ &= \operatorname*{argmax}_{\{\boldsymbol{\theta}, \Pi\}} \prod_n \prod_{l \in \mathrm{L}_n^u} \sum_k P(r_l, y_{m_l} | \theta_{m_l k}) \pi_{nk} \ . \end{aligned} \tag{4.22}$$

The direct optimisation of this expression with a gradient method is certainly not obvious. The trick here is again to add some latent indicator variables and derive an EM algorithm. For this model, the mixture variables appear in the contribution to the likelihood of each observed rating. It is then necessary to associate a latent indicator variable to each rating, i.e. $\mathbf{z}_l = [z_{l1}, \ldots, z_{lK}]$ with $z_{lk} = 1$ if the rating r_l was generated from the kth rating profile. The complete set of latent indicator variables is again denoted by Z. The derivation of the EM algorithm is similar to the previous model. The log-complete likelihood is

$$\log P(\mathbf{R}, \mathbf{Y}, \mathrm{Z} | \mathrm{U}; \Pi, \boldsymbol{\theta}) = \sum_n \sum_{l \in \mathrm{L}_n^u} \sum_k z_{lk} \left(\log P(r_l, y_{m_l} | \theta_{m_l k}) + \log \pi_{nk} \right) \ , \tag{4.23}$$

to be compared with Eq. (4.10). We see that for the E-step we need to compute the responsibilities of the latent variables of each observed rating

$$\rho_{lk} \equiv \mathbb{E}_Q\{z_{lk}\} \propto P(r_l, y_{m_l} | \theta_{m_l k}) \pi_{n_l k} \tag{4.24}$$

with the constraints $\sum_k \rho_{lk} = 1$. The index n_l refers to the user who made the lth rating.

The M-step can again be performed separately on the mixture parameters Π and the profile distribution parameters $\boldsymbol{\theta}$. For the mixture variables, we have

$$\pi_{nk} \leftarrow \frac{1}{L_n} \sum_{l \in L_n^u} \rho_{lk} \tag{4.25}$$

where L_n is the number of rating made by the nth user.

Considering again the multinomial-Gaussian profile definition (Eq. (4.5)), the update for the parameters are

$$\beta_{mk} \leftarrow \frac{1}{L_k} \sum_{l \in \mathrm{L}_m^y} \rho_{lk} \tag{4.26}$$

$$\mu_{mk} \leftarrow \frac{1}{L_{mk}} \sum_{l \in \mathrm{L}_m^y} \rho_{lk} r_l \tag{4.27}$$

$$\sigma_{mk}^2 \leftarrow \frac{1}{L_{mk}} \sum_{l \in \mathrm{L}_m^y} \rho_{lk} (r_l - \mu_{mk})^2 \tag{4.28}$$

where L_m^y is the set of rating indices such that r_l is attributed to item y_m, $L_k = \sum_m \rho_{lk}$ and $L_{mk} = \sum_{l \in \mathrm{L}_m^y} \rho_{lk}$. If we prefer only to model the forced prediction scheme, one does not need to consider the contribution of $\boldsymbol{\beta}$ in (4.24) and (4.26). Finally, the estimated rating distribution for a particular pair (u_n, y_m) is

$$P(r_{nm}|y_m = 1; \mathbf{r}_n, \mathbf{y}_n; \boldsymbol{\theta}, \boldsymbol{\pi}_n) = \sum_k \mathcal{N}(r_{nm}|\mu_{mk}, \sigma_{mk}^2)\rho_{nmk} \quad \text{(free prediction)} \tag{4.29}$$

$$\text{or} \qquad = \sum_k \mathcal{N}(r_{nm}|\mu_{mk}, \sigma_{mk}^2)\pi_{nk} \quad \text{(forced prediction)} \tag{4.30}$$

where $\rho_{nmk} \propto \beta_{mk} \cdot \pi_{nk}$ and $\sum_k \rho_{nmk} = 1$.

4.2.4 Model selection

As the discussion presented in Section 2.3.2, model selection with probabilistic mixture models can be tackled via different approaches. Until now, we have assumed that the number of profiles, K, was fixed. In practice, this number is generally unknown at the beginning and is thus a hyper-parameter to select. To evaluate which number suits the data best, we have to use a model selection procedure. A simple procedure is to fit the model for different values of the number of profiles and then keep the number of profiles that minimises a loss function (usually the negative log likelihood) on a validation set. If there are no other hyper-parameter to adjust, this is the recommended method because it is simple and robust. We use this procedure in the experimental section of this chapter.

The other strategy is to directly adjust the number of profiles using a learning criterion. The problem with the current formulation of the probabilistic profile clustering and the Aspect Model is that the likelihood always increases for an increasing number of profiles as the flexibility of the model is unbounded.

We can not set the number of profiles with the training criterion. But the solution is well known in the probabilistic framework: marginalisation! Ideally, both mixture parameters Π and profile parameters $\boldsymbol{\theta}$ should be given a prior probability and marginalised out of Eq. (4.22). The new learning criterion for the Aspect Model would then be

$$P(\mathbf{R}, \mathbf{Y}|\mathrm{U}; K) = \underset{K}{\operatorname{argmax}} \int_{\Pi,\Theta} \sum_{\mathrm{Z}} P(\mathbf{R}, \mathbf{Y}|\mathrm{Z}, \boldsymbol{\theta}) \cdot P(\mathrm{Z}|\Pi) \cdot P(\boldsymbol{\theta}, \Pi|K)\, d\Pi\, d\Theta \ . \tag{4.31}$$

To derive an EM algorithm for this marginal likelihood, we need to compute the posterior distribution over $(\mathrm{Z}, \boldsymbol{\theta}, \Pi)$. Unfortunately, this posterior is never analytical. So the solution is to resort to MCMC sampling methods or work with an approximate inference framework. This second solution is adopted in the model named latent Dirichlet allocation [20, 86]. The principle is to use a mean field approximation to the posterior distribution

$$P(\mathrm{Z}, \boldsymbol{\theta}, \Pi|\mathbf{R}, \mathbf{Y}, \mathrm{U}; K) \approx Q(\mathrm{Z})Q(\boldsymbol{\theta})Q(\Pi) \ , \tag{4.32}$$

in conjunction with conjugate prior distributions $P(\boldsymbol{\theta}|K) \cdot P(\Pi|K)$. In particular, the conjugate prior distribution of the (multinomial) mixture parameters Π is a common Dirichlet prior distribution $P(\Pi|K) = \prod_n \mathcal{D}i(\boldsymbol{\pi}_n|\boldsymbol{\alpha})$. It is then possible to derive a variational EM algorithm, with E-steps and M-steps similar to the ones derived in the previous subsections (see references).

In comparison to the selection of the number of profiles based on a validation set, this procedure has the advantage of using all of the (observed) ratings for training, which should reduce the variance of the estimated parameters. Of course, the additional assumptions on the prior over the parameters are responsible for a small increase in the bias of the estimator. From the practical point of view, there is no real computational benefit over a single pass validation scheme because the model must also be constructed with different number of profiles.

As stated in the preliminaries, it is possible to automatically adjust the number of profiles using non-parametric Bayesian models, with so-called Dirichlet processes (e.g. [70]). This type of processes allows the model to have possibly as many profiles as there are users. In practice, learning is carried out by sampling methods (frequently, a Gibbs sampling procedure is used). The simplest model with a Dirichlet process is known as the Chinese Restaurant process: at each step of the sampling algorithm, there is a set of active profiles and all users are associated with one of these active profiles, except the user currently sampled. From this state, we can compute the probability of membership of the current user to any of these active profiles or to a new profile. At the end, the chains of posterior profile membership of the users should stabilise around the best number of active profiles. The main difficulty in constructing such a model is that with a basic sampling strategy, the chains tend to be too correlated to mix in a reasonable amount of time. It is thus necessary to design more efficient sampling methods.

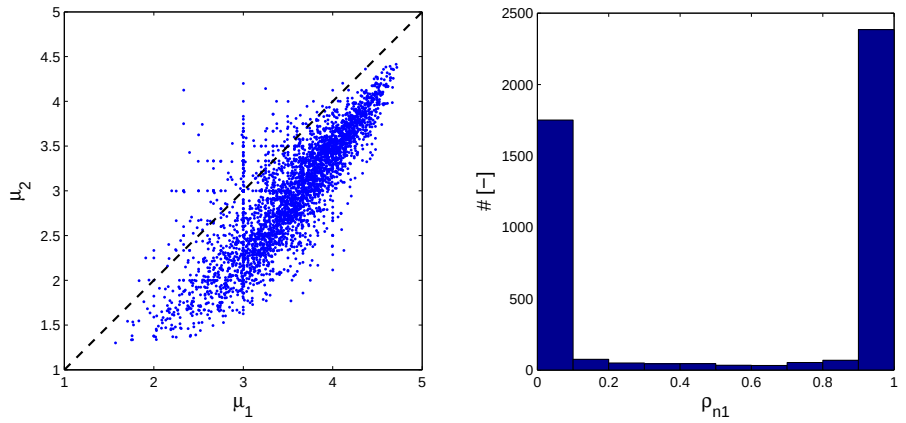
Choosing the number of profiles is not the only way to regularise the mixture models presented above. To avoid numerical issues (in particular, to avoid a degenerate M-step), it is often useful to add a small regularisation term to the distribution parameters $\boldsymbol{\theta}$ by means of a prior distribution. Hence, the training criterion becomes a maximum a posteriori criterion instead of a maximum likelihood. We would recommend to set the parameters of the prior over $\boldsymbol{\theta}$ to a common sense value since the flexibility of the model is already adjusted with the number of profiles. Regularisation can also be obtained by adding global constraints on the parameters. For example, working with explicit ratings and Gaussian rating profiles, one could constrain the variance to be the same for all the item distribution, i.e. $\forall m, k : \sigma_{mk}^2 = \sigma^2$.

4.2.5 Analysis of profiles

Throughout the thesis, we have tried to show how probabilistic modelling can be used to tackle different data analysis tasks. One of the important aspects of the approach is that prior intuition about the patterns in the data can be encoded by means of a simple generative process. For the Collaborative Filtering task, the intuition has led us to decompose the rating in typical rating profiles which might correspond to the different users' interests. At this stage of the analysis, one should wonder whether this intuition is legitimate and whether we can identify some understanding from the estimated profiles. We have already discussed the difficulty of interpretation in the preliminaries of this chapter and previously in the thesis. The main problem is that interpretation cannot be expressed by a quantitative criterion optimised on the generalisation performances.

In this section, we will illustrate the posterior analysis of rating profiles on the MovieLens dataset which is used in Section 4.4.3. The dataset stores the explicit ratings of users on a discrete scale $[1, \ldots, 5]$ expressing their satisfactions regarding a set of movies (the items). Further information can be found below. We use this dataset because it contains proportionally many more ratings than the usual Collaborative Filtering datasets, and more importantly because one can link the posterior analysis of the rating profiles with prior knowledge about the movies making the search for understanding easier.

Let us start the analysis with a very simple configuration: a probabilistic profile clustering model is constructed with only two components (or rating profiles) $K = 2$, and the rating distributions are modelled by binomial distributions (see Appendix (A)). Hence, the rating distributions are fully specified by the estimated probabilities of success of the binomial distributions, which we note μ_{mk}. Figure 4.2(a) shows a scatter plot of the estimated means of the profile distributions (μ_{m1} against μ_{m2}). There is thus one point in this scatter plot for each movie. What we see is that the means of the rating distributions between the two profiles are highly correlated. In other words, the order of preference over the movies according to both profiles is essentially the same. We notice also that the mean ratings for the first profile are systematically higher



(a) (b)

Figure 4.2: Probabilistic profile clustering with two components applied on MovieLens. (a) Mean of first vs. mean of second profile distributions. (b) Histogram of the posterior memberships of the users (the responsibilities) to the first profile.

than for the second. Hence, a user following the first rating profile tends to give systematically higher rating (on average) than a user following the second profile. Furthermore, from Figure 4.2(b) showing the posterior memberships of users to the profiles (Eq. (4.13)), we see that for most users we have strong posterior confidence about the rating profile they are following since most responsabilities are close to 0 and 1 (responsibilities close to one indicate that with strong confidence the users is following the first profile). Basically, the model has partitioned the set of users according to their tendency to rate rather above the average movie rating or below this average. Even when the model is constructed with more profiles, this separation remains clearly visible.

The observation highlights an important aspect of the MovieLens dataset and most of the Collaborative Filtering datasets actually, namely that the dominant pattern is a common order of preference over the item set and the main variability is not the expression of diverse users' interests but the tendency they have to give rather high or low ratings. This makes the identification of patterns representing the particular users' interests harder as it is masked by the dominant patterns.

The problem of the inconsistency of the users' rating tendencies was discussed in other works. For example in [67], the authors propose to further separate the behaviours of the users with an additional mixture hierarchy in the model. Another way to tackle this issue is by formulating the task as an ordinal regression task instead of regression on absolute rating values [151]. We will see that the dominant patterns and the users' rating tendencies are

encoded naturally in the model proposed in the next section.

Next, we will show the analysis of the profiles estimated with the Aspect Model. As explained above, these profiles should intuitively correspond to the different interests users might have about the movies. We would like to verify this fact. For this purpose, we construct both a free prediction implicit rating (i.e. $P(\mathbf{Y}|\mathrm{U};\boldsymbol{\theta},\Pi)$ is maximised) and a free prediction explicit rating scheme (i.e. $P(\mathbf{R},\mathbf{Y}|\mathrm{U};\boldsymbol{\theta},\Pi)$ is maximised). The explicit ratings scheme uses additional information and we want to see if this information helps finding more intuitive profiles. The models are constructed on a subset of the MovieLens dataset; only the 500 most rated movies are considered here.

To select the best number of profiles, we train the models on a subset of the rating matrix (20 percent of the observed ratings randomly selected were held out of the training stage). Figure 4.3(a) and (b) shows the negative log likelihood curves for the implicit and explicit rating schemes respectively. We see from the validation curve that the selected number of profiles is 7 for both the implicit ratings and explicit ratings schemes. We will come back shortly on the similarity of these curves.

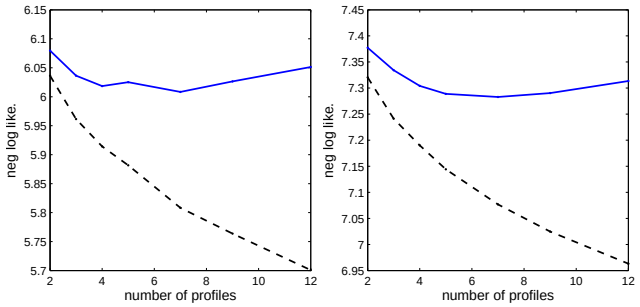
The movies in the MovieLens dataset are classified in $J = 14$ categories. Figure 4.4 shows the distribution of observed ratings among the movie categories. Let us note $\mathbf{c}_m$ the 14-dimensional vector of indicator variable for the membership of the mth movie to the categories; $\mathbf{c}_{mj} = 1$ if the mth movie belongs to the jth category, and $\mathbf{c}_{mj} = 0$ otherwise. The proportion of ratings for each category (i.e. the length of each bar in the histogram) is computed from

$$h_{j0} \propto \sum_m L_m \frac{\mathbf{c}_{mj}}{\sum_{j'} \mathbf{c}_{mj'}} , \tag{4.33}$$

for $j = 1 \dots J$, where L_m is the number of rating attributed to the mth movie and the proportionality stands for the normality constraints $\sum_j h_{j0} = 1$.

The question is to know if the profiles identified by the Aspect Model are related to particular categories. To answer this question, we look at the same empirical rating distribution among categories for each profile estimated by the model. Let us note by h_{jk} the proportion of rating attributed to the kth profile that correspond to movies belonging to the jth category. This proportion is again computed from (4.4) where L_m is replaced by $L_{mk} = \sum_{l \in \mathrm{L}_m^y} \rho_{lk}$ and the responsibilities ρ_{lk} are computed from Eq. (4.24). These distributions are on the histograms of Figures 4.5 and 4.6,(a) and (b), respectively for the implicit and explicit ratings schemes. Additionally, the list of titles of the movies receiving the largest number of ratings (computed by L_{mk}) for each profile of the implicit rating scheme is given in Appendix C.

Looking at the histograms in Figure 4.5(a) for implicit ratings, we clearly see that each profile corresponds to a particular type of movie. Profile 1 has a dominant Action/Thriller component, Profile 2 rather concentrates on the Sci-Fi category, Profile 3 on comedies, Profile 4 on romantic comedies (related also to Children's and Musical categories), Profile 5 corresponds to classic Drama/War movies (see the titles in Appendix C), Profile 6 is obviously identified as the



(a) Implicit ratings (b) Explicit ratings

Figure 4.3: Selection of the best number of profiles

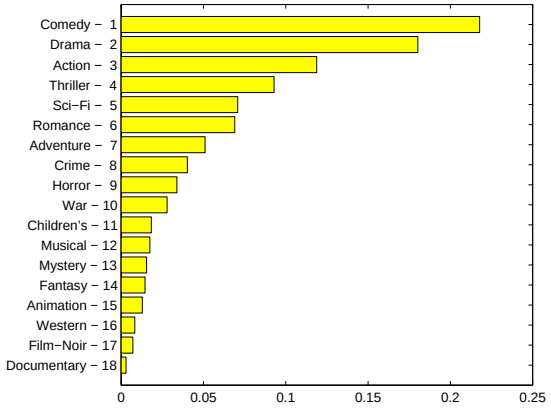
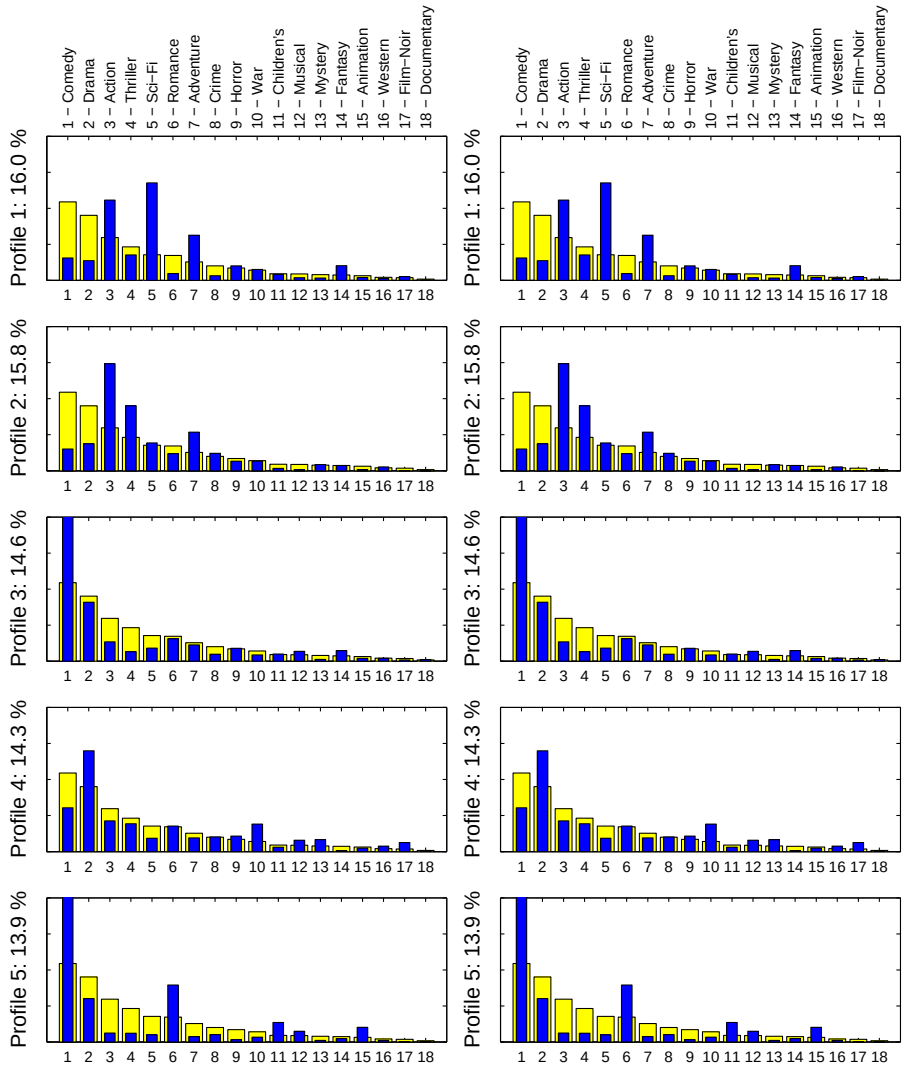
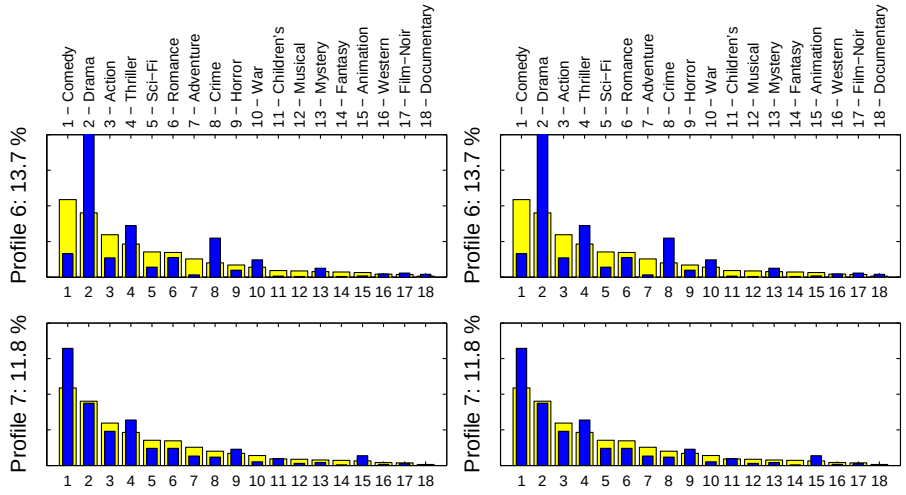


Figure 4.4: MovieLens dataset: distribution of ratings among the movie categories.



(a) Implicit ratings (b) Explicit ratings

Figure 4.5: Distributions of ratings among the movie categories for each estimated profile with a free prediction Aspect Model. Profile 1 to 5 (profiles continues in Figure 4.6. (a) Implicit ratings, (b) explicit ratings. The histogram behind the profile histograms indicates the global rating distribution (Figure 4.7).



(a) Implicit ratings (b) Explicit ratings

Figure 4.6: Distributions of ratings among the movie categories for each estimated profile with a free prediction Aspect Model. Profile 6 and 7 (see Figure 4.5 for other profiles).

Drama/Crime profile. We have no particular interpretation for Profile 7. The proportion of ratings attributed to each profile (computed by $\frac{1}{L}\sum_m L_{mk}$) is stated on the left of the histograms. This information tells us that all the profiles have about the same importance.

Comparing with the histograms of the explicit rating scheme, we observe a close similarity between the profiles found by both models. Profile 1 looks like a mixture of the Profiles 1 and 2 of the implicit scheme. Also, we notice a swap between Profiles 4 and 5. Profile 2 and 7 are less easy to interpret.

Obviously, the information brought by the explicit ratings is not playing a large role for the estimation of the profiles. This could have been anticipated. Indeed, the variability of explicit ratings for movies is generally quite limited (see histogram in Figure 4.7, most ratings are around 3 and 4). People knows which type of movie they like and from the comments they hear about a movie they know if they might want to see it, limiting the occurrence of low ratings.

From this analysis, we have seen that the implicit rating scheme is adequate for identifying groups of items which seem to correspond quite well to particular interests. Furthermore, the particular interests of a user u_n is identified by his mixture variable $\boldsymbol{\pi}_n$. On the other hand, the current formulation of the implicit-explicit rating scheme could certainly be improved. We have the feeling that the information encoded in the explicit ratings may be useful to identify preferences at a more subtle level. Also the rating tendencies of users should certainly be encoded in the model. Of course, the main difficulty is to include these ideas in

the model and still keep the learning procedure efficient to tackle real datasets.

4.3 Interlaced GLM

We have seen that the representation of users and items in the mixture models is not identical. The user behaviour is summarised with the responsibilities $\{\boldsymbol{\rho}_n\}$ in the latent profile cluster model, and with mixture variables $\{\boldsymbol{\pi}_n\}$ in the Aspect Model. On the other hand, the items are represented with the set of K rating distributions defining the profiles. In this section, we are looking for a simpler representation of the users and items which would reduce the number of parameters necessary to describe the rating distributions and avoid the computational cost of the EM algorithm. But we want to keep a probabilistic view on the rating process, and to be able to easily modify the overall model by changing the assumptions on the rating distributions and the prior parameter distributions.

4.3.1 Model Description

The fundamental step in designing a Collaborative Filtering model is to define the way users and items information (contained in the observed ratings) is encoded in the model. The idea with the interlaced Generalize Linear Model (GLM) formulation is to encode the users and items with a dual feature representation. In other words, we associate with each user u_n a feature vector $\phi_n \in \mathbb{R}^K$, and symmetrically with each item y_m a feature vector $\omega_m \in \mathbb{R}^K$. For the moment, we assume that the dimension of the feature space K is fixed *a priori*. The user and item features are the main parameters of the model. They summarise the information about the rating process.

The next assumption of the proposed model is that the appreciations of items expressed by the users can be estimated by a dot product between their respective feature representations: the appreciation of item y_m for user u_n is evaluated as $\eta_{nm} = \phi_n^\top \omega_m$. The intuition behind this model is similar to the Aspect Model. The user features can be understood as different sensibilities to a set of aspects describing the items, and the item features correspond to their content with respect to these different aspects. The estimated appreciation comes from an addition of positive and/or negative contributions over all the aspects.

Now, it is necessary to transform the appreciation into a predicted rating distribution. This is done using the GLM formalism [90]. Remember from Section 3.2.1 that a GLM is defined by a link function matching the appreciation (or activation, in the GLM terms) to the mean of the rating distribution, $\mu = g(\eta)$, and a conditional distribution $p(r|\mu;\psi)$ chosen from the exponential family (see Appendix A.2) with the parameter(s) ψ allowing the adjustment of the dispersion of ratings around the mean. The important feature of the Collaborative Filtering task is that both ϕ and ω are parameters to optimized,

unlike a standard GLM where only one of the vectors is a parameter. For this reason, we call it an *interlaced* GLM framework. In fact, the procedure is applicable to distributions not coming from the exponential family. We will come back to this family later.

One can see that this model performs a factorisation of the rating matrix. Indeed, writing the feature matrices $\Phi = [\phi_1, \dots, \phi_N]^\top$ and $\Omega = [\omega_1, \dots, \omega_M]^\top$, we see that predictions for the mean ratings are computed by $\bar{\mathbf{R}} = g(\Phi\Omega^\top)$. The mean rating would be the optimal prediction if the goal was to minimise the least squares loss. For other losses (e.g. the mean absolute error) one can always derive the optimal prediction from the estimated rating distribution $P(r|\mu;\psi)$. Following the usual procedure in probabilistic modelling, the model is trained on the negative log likelihood criterion. Adjusting the feature matrices using this criterion is expressed by

$$\begin{aligned}\{\hat{\Phi}, \hat{\Omega}\} &= \underset{\{\Phi,\Omega\}}{\operatorname{argmin}} \left\{ -\log P(\mathbf{R}|\bar{\mathbf{R}};\psi) \right\} \\ &= \underset{\{\Phi,\Omega\}}{\operatorname{argmin}} \left\{ -\sum_l \log P\left(r_l | g(\phi_{n_l}^\top \omega_{m_l});\psi\right) \right\} . \end{aligned} \tag{4.34}$$

The summation comes from the hypothesis of conditional independence of ratings given the feature representation. Note also that this summation runs over the observed ratings only (indexed by l) because there is no contribution to the likelihood for unobserved ratings. This is essential in order to be applicable on large Collaborative Filtering tasks. However, as discussed for example in [87], unobserved ratings are informative about the type of items a user is interested in and knowing this information can improve the prediction. The model described in this paper does not make use of this information.

In order to have a model that generalises well on unobserved ratings, we need to adjust its capacity. As usual, there are two approaches to adjust capacity: Either we select the structural parameter defining the dimensionality of the parameter space, in this case the feature dimensionality K, or we adjust the capacity by means of a regularisation term. As the regularisation term is a continuous function of the hyper-parameter(s), this second approach allows a smoother adjustment. Keeping a probabilistic framework, it is natural to define the regularisation term by a prior distribution on the parameters (Chapter 1), leading to a minimum negative log *a posteriori* learning criterion

$$\{\hat{\Phi}, \hat{\Omega}\} = \underset{\{\Phi,\Omega\}}{\operatorname{argmin}} \left\{ -\log P(\mathbf{R}|\bar{\mathbf{R}};\psi) - \log P(\{\phi_n\}, \{\omega_m\}|\alpha) \right\} , \tag{4.35}$$

where α is a set of hyper-parameters for the prior distribution. In the same spirit as the derivation in Section 3.2.1, one could regularise the features with Gaussian prior distributions

$$P(\{\phi_n\}, \{\omega_m\}|\alpha) = \prod_n \mathcal{N}(\phi_n|0, \alpha^u \mathbf{I}_K) \prod_m \mathcal{N}(\omega_m|0, \alpha^y \mathbf{I}_K) , \tag{4.36}$$

where $\mathbf{I}_K$ is the $K \times K$ identity matrix, α^u and α^y are the precision parameters (i.e. inverse variance) of the Gaussian distributions. We will discuss in the Subsection 4.3.3 a pragmatic approach to optimize the parameters and hyper-parameters of this model. But before let us give some indications on how to choose the different distributions appearing in the model.

4.3.2 Choosing the configuration

The assumption of a Gaussian noise distribution is common in regression problems. It is known that fitting a Gaussian noise model is equivalent to fitting a least squares criterion. Besides, if the identity $\mu = \eta$ is used as link function and if there is no regularization term, the interlaced GLM is equivalent to minimizing the Frobenius norm $\|\mathbf{R} - \Phi\Omega^\top\|_F$ for a given feature dimensionality. In that case, there is an indeterminacy in the solution as any invertible matrix $\mathbf{G}$ of size $K \times K$ inserted in the factorization $(\Phi\mathbf{G})(\mathbf{G}^{-1}\Omega^\top)$ gives the same Frobenius norm. We have already mentioned in Section 4.1.2 that the SVD factorisation also minimise this norm, but with the requirement of minimum reconstruction error for the successive components. So, both approaches would give the same mean rating prediction $\bar{\mathbf{R}}$, but with different factor matrices.

One of the advantages of using the probabilistic framework is that different feature representations and regularisation terms can be tested without changing the general optimization technique (see the next subsection). A Gaussian noise with the identity link is certainly not the only acceptable choice. There are several reasons which would suggest using another noise model. First, it is not really satisfactory to represent a distribution over a bounded range of rating with the unbounded Gaussian. It is more appropriate to work with a bounded distribution like the Beta for continuous ratings (the original ratings should then be scaled/translated to fit the $[0, 1]$ range), or a binomial for discrete ratings $r \in [0, \ldots, r_{max}]$

$$\mathcal{B}n(r|\mu; r_{max}) = \frac{r_{max}!}{r!(r_{max} - r)!} \left(\frac{\mu}{r_{max}}\right)^r \left(1 - \frac{\mu}{r_{max}}\right)^{r_{max}-r} . \tag{4.37}$$

To use this Binomial, it might also be necessary to scale and translate the original rating range. Additionally, a saturated link function must be used to avoid predicting a mean outside the rating range; this can be done for example with the logistic link

$$\frac{\mu}{r_{max}} = \frac{1}{1 + \exp(-\eta)} \tag{4.38}$$

Another reason why the Gaussian noise model (and the least square fitting criterion) might not be the best choice is that this model is sensitive to atypical ratings. Some users might try to mislead the Collaborative Filtering model by entering random ratings. In particular if they give randomly high and low ratings, these ratings can have a big influence on the estimated features while they do not contribute to the identification of the inherent patterns in the rating

matrix. Actually, the Binomial distribution is also sensitive to atypical ratings. One could use more robust noise models like the Student-t distribution or a power binomial distribution $P(r|\mu;\nu) \propto \mathcal{B}n(r|\mu, r_{max})^{\nu}$. Unfortunately, these distributions are not members of the exponential family and their optimization (see next subsection) is more demanding.

One can also play on the prior distribution to encode intuitive considerations in the model. For example, it could be a good idea to constrain the user features to be positive, making them closer to the concept of sensibilities to different aspects: the more sensible a user is to a particular aspect, the higher the contribution of the associated feature to the ratings. The positivness can be enforced by an exponential prior on the user features

$$P(\{\phi_n\}|\alpha) = \prod_n \prod_d \mathcal{E}xp(\phi_{nd}|\alpha^u) \tag{4.39}$$

where the exponential distribution is defined by $\mathcal{E}xp(\phi|\alpha^u) \propto \exp(-\alpha^u\phi)$. One could also use a Gamma distribution to have more control on the prior distribution.

So far, the parameterisation of the noise and prior distributions was kept as concise as possible, imposing a common dispersion ψ for all ratings and common prior parameters α^u and α^y for all user and item features respectively. Intuitively, the rating profiles of users present great diversity; some users are very predictable, other much less and the same can be said about the items. One could think of more flexible configurations where each user (item) has its own prior parameters α^u_n (α^y_m). Also the dispersion parameter ψ could depend on the users and items. However, as discussed in the next subsection, the difficulty is then to define an efficient optimization strategy for such a flexible configuration.

As discussed in Section 4.2.5, the transformation of appreciations into rating values is not consistent between users. Some users tend to give systematically higher ratings than others, and yet they have a very similar order of preference over the item set. With the factorisation approach, it is simple to represent user specific shifts in the rating distributions. One just needs to fix the value of one of the item features, for example $\omega_{m1} = 1$. The corresponding user feature ϕ_{n1} is a contribution to the appreciation that is independent of the rated item.

4.3.3 Learning procedure

As usual, there are two levels in the optimisation of the interlaced GLM. The inner level corresponds to training the features Φ and Ω. The outer level corresponds to the hyper-parameters, namely the number of features K, the rating dispersion parameter ψ, and the prior distribution parameters α.

Let us first consider that the hyper-parameters are fixed. A way to optimise the features is to update one feature vector at a time. The optimizations of the

regularized loss (Eq. 4.35) with respect to ω_m and ϕ_n are respectively

$$\hat{\omega}_m = \underset{\omega_m}{\operatorname{argmin}} \left\{ -\sum_{l \in \mathrm{L}_m^y} \log P(r_l | g(\phi_{n_l}^\top \omega_m), \psi) - \log P(\omega_m | \alpha^y) \right\} \tag{4.40}$$

and

$$\hat{\phi}_n = \underset{\phi_n}{\operatorname{argmin}} \left\{ -\sum_{l \in \mathrm{L}_n^u} \log P(r_l | g(\phi_n^\top \omega_{m_l}), \psi) - \log P(\phi_n | \alpha^u) \right\} \tag{4.41}$$

where L_m^y (L_n^u) is the set of indexes associated to the ratings of item y_m (user u_n). Each feature vector update is a standard regression problem with typically less than a few hundreds learning pairs (ϕ_{n_l}, r_l) (Eq. (4.40)) or (ω_{m_l}, r_l) (Eq. (4.41)). Finding a local optimum of this criterion is fast, especially with distributions from the exponential family. One can apply a few *iteratively reweighted least squares* steps [63], or any other gradient descent algorithm. The general gradient and Hessian expressions for the GLM framework are given in Appendix. All feature vectors ω_m and ϕ_n are to be updated in turn and the complete procedure must be repeated until convergence. The procedure can be applied to large datasets as memory requirements are small. Furthermore, the process could be parallelized.

For the adjustment of the hyper-parameters, we propose a pragmatic approach. One thing we have already discussed is that both the number K of features and the prior distribution control the flexibility of the model. In general, there is no great advantage in adjusting both simultaneously. In the experiments below, we compare the optimization of K and of α separately.

Let us look again at the Gaussian prior distribution in Eq. 4.36. Two precision hyper-parameters appear (α^u and α^y). However, as user and item features interact multiplicatively, it is sufficient to set the precision over users (or items) to an arbitrary value and adjust only the precision of the other set. Another point that we have already encountered in Section 3.2.1 is that the dispersion hyper-parameter ψ generally has a dual influence to the regularisation (or prior) hyper-parameters in the non-probabilistic view. The assumption of small dispersion can be compensated by a weak prior distribution (i.e. a small precision α), although the duality is exact only for Gaussian noise and Gaussian prior distributions. It is thus convenient to fix ψ from an estimate of the noise dispersion (Eq. (A.14)). The global rating variance is a sensible starting point and it can be improved after construction of the model from a set of held out ratings. This pragmatic procedure leaves us with only one common prior parameter to be adjusted by a resampling technique. As the Gaussian prior distribution is closely related to the ridge regularisation, the single regularization (or prior) hyper-parameter is called the *ridge* parameter in the experiments.

Obviously, this procedure is closer to a pragmatic regularization scheme than a Bayesian framework. It would be interesting to fit the dispersion and prior parameters on a marginal likelihood criterion as in Chapter 3.

Moreover, this type of optimization procedure should also be applicable for more flexible configurations as suggested in the Subsection 4.3.2 where for example all user and item features have their own hyper-parameters. The marginalisation requires the estimation of the posterior distribution of the feature vectors, which unfortunately is intractable and besides, we would probably loose the small computational load of the proposed pragmatic interlaced GLM optimization. An approximation scheme is thus necessary. One could wonder whether applying the simple update rules for the hyper-parameters derived in Section 3.2.1 [80] for a regression problem would be a good approach. Unfortunately, these updates lead to over-fitting because both user and item features are adjusted at the same time, deviating thus from the assumptions of the regression framework. The design of an efficient marginalization framework is left for a future work.

4.4 Experiments

The objective of this section is to evaluate the performance of mixture models and some configurations of the interlaced GLM for the prediction of explicit ratings. In contrast to Section 4.2.5, our discussion here is based on quantitative performance rather than subjective interpretation. Three publicly available datasets are used for the comparison: MovieLens, Jester, Libimseti. These datasets are described below.

4.4.1 Evaluation procedure

The models are evaluated with a 4-fold cross-validation procedure. The set of users is first divided randomly into four groups containing the same number of users. Each of these four user sets (and the corresponding ratings) is in turn held out of the training stage. For each fold, we have Φ_{train} and Φ_{test} (for the interlaced GLM model), the matrices storing the features of the users in the training set (three of the four user sets) and in the test set respectively. Given this split, a model is constructed on the rating of the users in the training set, returning an estimate for Φ_{train} and Ω. During the test stage, we start by selecting some of the ratings of the users in the test set. These are the test ratings on which the prediction performances are evaluated. Then, the feature matrix Φ_{test} is adjusted on the remaining ratings. It requires a single run over each test user features as Ω is not modified anymore. In comparison, for the latent profile clustering model, it is necessary to compute the responsibilities (4.11) of the test users to be able to make prediction with Eq. (4.19). Of course, the profile parameters $\boldsymbol{\theta}$ are not modified anymore after the training phase. For the Aspect Model, the mixture parameters of the test users, Π_{test}, must be estimated. Unfortunately, it cannot be done analytically and it is thus necessary to apply the EM algorithm to evaluate these parameters.

Once the features of the test users are estimated, predictions can be made

for the test ratings. The performances are evaluated with two standard loss functions: the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE). Note that for the MAE loss, the optimal prediction is the median of the predicted rating distribution, while for the RMSE loss it is the mean. Finally, we summarise the performance by the mean of these estimates over the four folds.

Below, we have used two schemes to select test ratings. The first one is called the *all-but-one* scheme: a single rating (selected randomly) is retrieved for each test user. This procedure should be a fair estimation of the average prediction performances, but it is biased because datasets usually contain users having rated a minimum number of ratings (typically 20).

The second scheme, called here the *given-*L_{test} scheme, consists in keeping only L_{test} ratings for each user in the test set. Their other ratings are used for evaluation. So, this scheme concentrates on the dependence of prediction performances with the number of observed ratings for a user. We did not compute the performances of the mixtures models for this scheme.

4.4.2 Model configurations

In the experiments below, we compare the latent profile clustering and subspace model as well as five configurations of the interlaced GLM. Here are short descriptions of the different configurations. For the selection of the hyper-parameters, we systematically used the same validation procedure, leaving 20 percent of the current learning data for the validation set. Also, in order to apply the binomial distribution on continuous ratings, we transform the ratings during learning to a discrete range with 5 equal bins.

- LP-clustering: latent profile clustering model with the forced prediction scheme and the rating distribution modelled by binomial distributions. The optimal number of clusters is selected in the range [1,10] with a validation procedure.

- Aspect Model (or latent profile subspace): the same settings as the LP-clustering configuration are used, i.e. the binomial distribution for the rating distribution and the number of profiles is found by a validation procedure.

- Common preference: baseline configuration of the interlaced GLM configuration. In this configuration, $K = 2$ and we constrain $\phi_{n1} = 1$ and $\omega_{m2} = 1$. There is no constraint on ϕ_{n2} and ω_{m1}. In this way, there is no direct interaction between the user and item features. Predictions are composed of an item average rating plus a user tendency to deviate from the item average. Consequently, all users have the same order of preference over the items. We use an identity link function and Gaussian noise. With the Gaussian noise model, predicted ratings can fall outside the bounds. When it happens, the prediction is set to the exceeded bound.

- Gauss-K : configuration using the identity link function and the Gaussian noise distribution. A weak Gaussian prior distribution is applied such that we are almost at the maximum likelihood fit. The capacity is adjusted by the single pass validation procedure on the dimensionality K of feature vectors.

- Gauss-ridge: here also, the identity link function is used with the Gaussian noise distribution. However, the capacity is adjusted with a common ridge parameter (i.e. the variance of the Gaussian prior distribution over items) and this parameter is selected by the validation procedure. The dimensionality of the features is set to $K = 8$, which is found to be sufficiently high as we will see in the discussion section.

- Binom-ridge: same configuration as Gauss-ridge except that a logistic link function and binomial dispersion are used (Eq. (4.37)). We keep $K = 8$.

- Gauss-Expon: this configuration is supposed to be closer to the intuition of the rating process. We keep the Gaussian noise model and linear link, but the user features are forced to be positive with an exponential prior distribution (Eq. (4.39)) and fixed $\alpha^u = 1.$. Additionally, we allow user specific shifts in the rating distribution similarly to the common preference configuration. Thus $\omega_{m1} = 1$ and ϕ_{n1} is given a weak Gaussian prior. The only hyper-parameter to be selected by validation is still the ridge parameter associated with the prior item feature distribution.

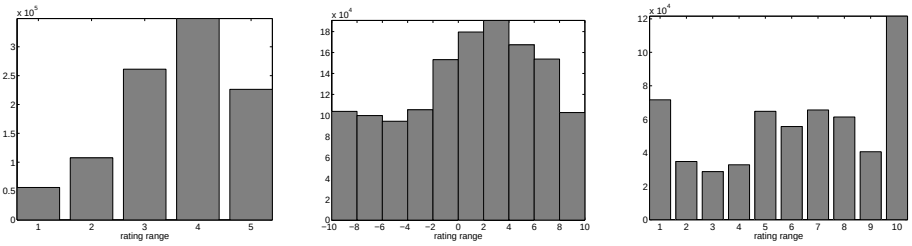
For all configurations, the same validation procedure is used to select the ridge hyper-parameter: Before training a model on one of the 4-fold splits of the evaluation procedure, the learning ratings are further split randomly, leaving 20 percent of them aside for the validation.

4.4.3 Datasets

The models are compared on three publicly available datasets (see below). In order to keep the time of the training-evaluation procedure to around 4 hours for each configuration (on a 3.0Mhz processor with 1.5 Gb RAM and Matlab coding), we have had to work with only a subset of the ratings for two of the datasets as described below.

- MovieLens[2]: The dataset is distributed by GroupLens Research at the University of Minnesota. It contains 6040 users, 3900 movies (the items), and approximately 1 million discrete ratings collected from users registered in the service in 2000. The ratings of the movies are on a discrete scale from 1 to 5. Each user has at least 20 ratings encoded.

[2]Available at http://www.grouplens.org/



(a) MovieLens (b) Jester (c) LibimSeTi

Figure 4.7: Global rating distributions for the three datasets

- Jester[3]: From the Jester Online Joke Recommender System [48], the dataset contains 73,421 users, 100 jokes (the items), and 4.1 million continuous ratings on the interval $[-10, 10]$. In the experiments, only 24,000 randomly selected users were considered.
- LibimSeTi[4]: Collected from the LibimSeTi dating agency [25], the complete dataset contains 17,359,346 anonymous ratings of 168,791 profiles of members made by 135,359 LibimSeTi users until April 4, 2006. The ratings are on a discrete scale from 1 to 10. In the experiments, we picked randomly 10,000 users, and 10,000 rated profiles of members (the "items") having at least received 20 ratings from the subset of users.

Figure 4.7 shows histograms of the observed distributions of ratings for the three datasets. Note that Jester ratings were discretised in 10 bins, but they are intrinsically continuous ratings. We can see that for all three datasets there is a high concentration of ratings at a position a bit above mid-range. We assume that this corresponds in the mind of the users to a default appreciation - not a bad item, neither a very good one. Another thing to note is the small proportion of ratings in the MovieLens dataset lower than mid-range, contrarily to the other two sets. We guess it is easier for the users to know in advance what movies they should avoid (from synopsis, critics...) than for the jokes or the personal profiles (LibimSeTi). And anyway, it is more demanding to watch a bad movie than read a bad joke or evaluate a profile that is not particularly desirable. The last thing to note from these histograms is the high number of extreme ratings (1 or 10) for the LibimSeTi dataset.

Table 4.1 stores summary information about the three datasets.

4.4.4 Discussion

Let us start the discussion from the prediction performances for the all-but-one scheme. The results are given in Table 4.2.

[3]Available at http://www.ieor.berkeley.edu/~goldberg/jester-data/
[4]Available at http://www.ksi.ms.mff.cuni.cz/~petricek/data/

	N	M	L	Range	$\mathbb{E}_{\mathcal{D}}\{r\}$	med L_n^u	med L_m^y
MovieLens	6.04e3	3.88e3	1.00e6	$\{1,\ldots,5\}$	3.58	96	123
Jester	24.0e3	100	1.35e6	$[-10,10]$	0.74	52	12.7e3
LibimSeTi	10.0e3	10.0e3	0.58e6	$\{1,\ldots,10\}$	6.14	36	34

Table 4.1: Summary for the three (reduced) datasets. $\mathbb{E}_{\mathcal{D}}\{r\}$is the empirical mean, med L_n^y is the median number of rating observed by user and med L_m^y is the median number of rating observed by items.

	MovieLens		Jester		LibimSeTi	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
LP-clustering	0.682	0.916	3.43	4.36	1.24	1.88
LP-subspace	0.688	0.903	3.89	4.46	1.44	2.16
Common Pref.	0.695	0.937	3.49	4.41	1.24	1.81
Gauss-K	0.695	0.928	3.33	4.30	1.28	1.95
Gauss-ridge	0.663	0.892	3.32	4.25	1.27	1.83
Binom-ridge	0.656	0.886	3.33	4.26	1.30	1.87
Gauss-Expon	0.648	0.881	3.29	4.31	1.34	1.92

Table 4.2: Rating prediction performances of the all-but-one scheme. The numbers correspond to the mean of a 4-fold procedure.

In comparison, the best performances found in the literature (to our knowledge) for MovieLens are a MAE of 0.652 (in [33]) and for Jester a MAE of 3.26 (in [26]). We are not aware of a similar evaluation on the LibimSeTi dataset. We can see that the performances obtained with the interlaced GLM are very similar to these state-of-the-art performances. We must state that we observe about 5-7 percent difference between the evaluations on the different folds. Reducing the variance of the estimation of the performances by repeating the 4-fold procedure would certainly change the figures in the table a little. Unfortunately, the evaluation then becomes computationally very intensive.

Analysing the mixture models performances, we notice that there is no advantage of the subspace model over the clustering model. On the MovieLens dataset, both models perform similarly while on the Jester and LibimSeTi datasets, the Aspect Model performs surprisingly badly (possibly, due to convergence to bad local minima). The number of profiles selected was always very low ($K = 4$ for MovieLens and $K = 2$ for Jester and LibimSeTi). Compared with the interlaced GLM formulation, the mixture models performances look clearly behind on both MovieLens and Jester datasets. It is difficult to tell the main factor causing this difference. There are several possible explanations. We have already stated the problem of user specific rating tendencies which are not well encoded in the mixture models. Additionally, for an equivalent number of parameters, the interlaced GLM formulation is probably less subject to overfitting because the rating distributions in this model are more dependent on

one another than in the mixture formulation.

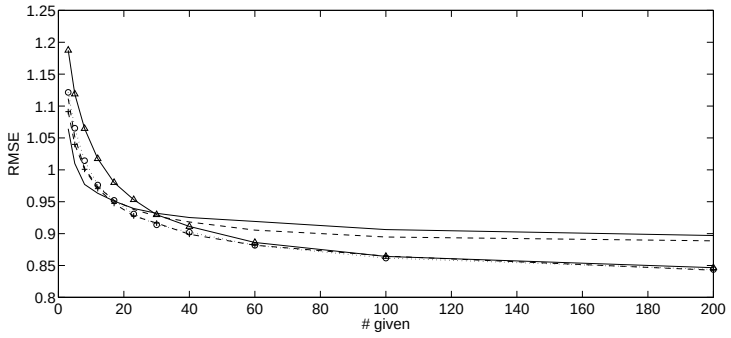
We also observe from Table 4.2 that adjusting the prior distribution parameter rather than the dimensionality of the features is consistently advantageous. Indeed, the Gauss-ridge configuration always performs better than the Gauss-K configuration. In fact, we have noticed that the best dimension K was very low for all datasets (around $K = 3$). The model quickly over-fits above this number. The choice of $K = 8$ for the other configurations looks thus more than sufficient. Note that mixture models would probably also benefit from a smoother regularisation scheme than selecting the number of components in the mixture but it is less evident to see how this regularisation should be defined.

There does not seem to be a real difference between the Gaussian and the binomial noise models as both configurations perform similarly on all datasets. For the last interlaced GLM configuration (i.e. the Gauss-Expon), we see that the positivity constraint imposed by the exponential prior distribution leads to good performances on the MovieLens and Jester datasets. It is a bit weaker on LibimSeTi. One reason might be that this last configuration is more easily trapped in bad local optima (due to the positivity constraints) when the average number of ratings shared between users is low, like in the LibimSeTi dataset.

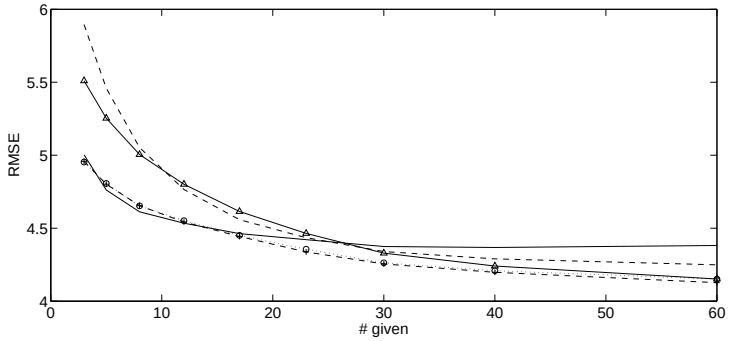
We finish the analysis of Table 4.2 with the common preference configuration. This configuration performs only a few percent worse than the best configurations on MovieLens and Jester. Actually, on LibimSeTi it is the best configuration. This observation is in agreement with the discussion in Section 4.2.5, namely that users tend to agree on item's evaluations (up to a user specific rating shift) and consequently the dominant patterns in the rating matrix are well expressed by global averages. Nevertheless, a few percent improvement can be of practical importance as demonstrated by the Netflix Prize[5]. For the LibimSeTi dataset, our feeling is that the dataset is too empty to detect something other than global averages.

Let us turn to the given-L_{test} scheme. As stated above, only the interlaced GLM models are evaluated here (due to lack of time to implement and run the scheme on the mixture models). The performances are plotted in Figure 4.8, representing the mean RMSE for rating predictions. Each curve corresponds to one of the interlaced GLM configurations. As expected, we see that the prediction improves (on average) as the number of observed ratings used to fit the user features increases. It can also be seen that the common preference performs best when the number of observed ratings by user is lower than (around) 15 ratings. Again, this does not apply to the LibimSeTi set for the same reasons as given above. Beyond this number of 15 ratings, the Gauss-ridge and Binomial-ridge are performing better than the common preference configuration. This observation would incite us to design a more elaborate regularisation scheme for which the regularisation decreases as the number of observed ratings increases. In fact, designing a fully Bayesian scheme would perform this personalised regularisation automatically. These figures also show

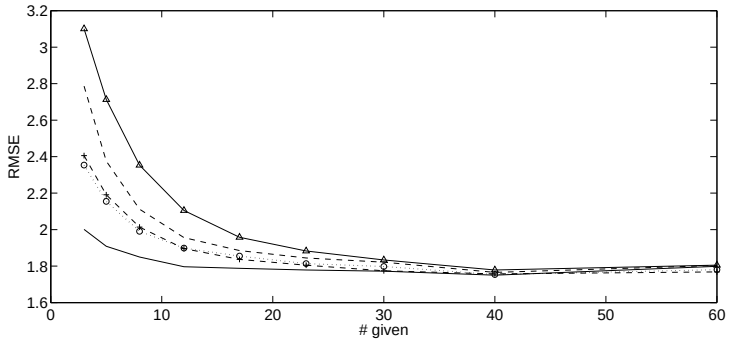
[5]http://www.netflixprize.com/. The objective is to improve a baseline method by 10 percent on the RMSE criterion.



(a) MovieLens



(b) Jester



(c) LibimSeTi

Figure 4.8: Prediction performance (RMSE criterion) using the given-L_{test} evaluation procedure. Line identifier: $(-)$ Common preference, $(--)$ Gauss-K, $(+--)$ Gauss-ridge, $(\circ\cdot\cdot)$ Binom-ridge, $(\triangle\ -)$ Gauss-Expon

the weaker performances of the Gauss-Expon configuration when there are few observed ratings.

4.5 Conclusion

In this chapter, we have presented two approaches to tackle the Collaborative Filtering task: a mixture approach and a matrix factorisation approach. We have illustrated the use of mixture models on the MovieLens dataset to analyse patterns from the Collaborative Filtering scheme and identify some interpretation. Interestingly, we could match the patterns (i.e. the rating profiles) with the intuition of users' interests. However, we are conscious that this illustration would not translate so easily on other datasets mainly for two reasons. First the rating matrix is often extremely empty and the users tend to share only very few ratings with other users (e.g. the LibimSeTi dataset). Consequently it is difficult to benefit from the collaboration. Secondly, the concept of interests or aspects is not relevant for every application and it is not necessarily possible to link the profiles with interpretation.

In the second part of the chapter, we focused on the explicit rating prediction task. For this purpose, we proposed a simplified model, called the interlaced GLM. The formulation of this model is mathematically well founded and its optimisation is relatively straightforward which makes the model applicable to large Collaborative Filtering datasets. Furthermore, the prediction performances of this interlaced GLM are clearly better than those obtained with the mixture models. Actually, the interlaced GLM performs similarly to the best performances stated in the literature on the same datasets.

We identified a weak point in the learning procedure of the interlaced GLM model which is the regularisation scheme common to all users. An improvement would be to personalise the adjustment of the regularisation parameters (and dispersion parameters). We would probably see that users having expressed few ratings have their feature vectors more constrained (i.e. more biased by the prior distribution) than those with many ratings. Another improvement would be to take into account in the model the missing-observed rating information, similarly to the free prediction scheme in the mixture models.

Experiments have highlighted the fact that the average order of preferences over the item set is the dominant pattern of these rating matrices. One can wonder if the lack of marked variability in the users' preferences could be due to the framework used to collect the data. In particular, rating items over a single appreciation scale might mask the expression of diversity. We have also the feeling that the batch approach to tackle Collaborative Filtering analysis is the main barrier to improve its usefulness. While the user's interests are probably quite constant over time, the daily needs (e.g. the movie I would like to rent today) can obviously vary a lot. Certainly, Collaborative Filtering for recommendation would benefit from an online processing of the users exchange of information with the system. Unfortunately, it is more complicated to have

data and design models with an online service.

Conclusion

Throughout the thesis, we have presented the probabilistic modelling framework for data analysis, illustrated its features, advantages, but also identified its weaknesses. We based our study more specifically on the practical aspects of data analysis. This brought us to consider different learning tasks and different applications as well as to design empirical evaluations of the framework on real data (i.e. near-infrared spectroscopy and collaborative filtering datasets). We review here the main elements brought by our analysis.

One aspect of probabilistic modelling that makes the framework very attractive is the Bayesian selection principle since it is potentially possible to fit any hyper-parameter in a model, provided the marginal likelihood over the parameters can be computed and optimised. Also from a practical point of view, this method simplifies the learning procedure as it is not necessary to repeat the learning process, and the marginal likelihood criterion is common for all the hyper-parameters. Consequently, the Bayesian selection method allows us to work with flexible models having many hyper-parameters in their formulation. In addition, Bayesian selection avoids the caveats of working with too much flexibility like the standard neural networks formulation (and the Multi-Layer Perceptron in particular) thanks to its inner regularisation principle. At least, this is what the theory states; however the theory presumes that the model assumptions match the data distribution. From the experimental study on regression in Chapter 3, we have tried to assess whether the Bayesian selection framework, or more precisely marginal likelihood maximisation, can be adequate for tackling real data analysis applications. The results we obtained were quite positive. We compared Bayesian selection directly with the (supposedly) more robust cross-validation procedure to select a single hyper-parameter and found out that the latter procedure on average was not performing better. On the other hand, adding more flexibility to the model by means of an Automatic Relevance Determination scheme showed only subtle improvements. We would not recommend its use only based on the performances but we showed that the relevance scheme can additionally be used to spot relevant parts of the spectra for the regression task. Obviously, it makes little sense to generalise these observations about the Bayesian selection procedure as it is really dependent on the data. Nevertheless, the study illustrates an application where Bayesian selection is clearly competitive with resampling methods.

A second beneficial property of the probabilistic modelling framework is that it is highly modular. In the second chapter, we showed how the modularity nature of the probabilistic framework can be used to transform a nice model, the Probabilistic PCA (and its mixture extension), into an even more useful one, the (mixture of) robust probabilistic PCA(s). One advantage over other robust extensions to PCA is that there is only a single additional parameter for each component which is directly optimised in the learning criterion. In practice, working with a more robust learning formulation makes a real difference, at least for small datasets where outliers have a significant influence. We have also discussed the property of modularity to extend the mixture models presented in the context of Collaborative Filtering. The idea was also exploited in the interlaced GLM as it is straightforward to modify the noise assumptions and prior distributions over the parameters. Unfortunately, making use of the modularity of the probabilistic framework is not always as simple as the configurations quickly give rise to intractable inference. For the robust extension of the Probabillistic PCA, we could develop an exact Expectation-Maximisation algorithm. On the other hand, we would have like to add intuitively justified constraints on parameters of the mixtures models in the Collaborative Filtering application, but the optimisation becomes much more complicated as the posterior distribution are non analytic. Also for the interlaced GLM model, the learning procedure would probably have benefited from an additional hierarchy allowing us to adapt user specific parameters. But here also, the formulation deviates from the standard scheme and it is no longer possible to derive a simple optimisation procedure.

The other feature of the learning procedure in which we were interested is to see how modelling can be useful for deriving understanding about the patterns in the data, and from there to understand the system itself. We have discussed the possibility of encoding intuition in the model for example with mixture and latent variables. Intuition is often a good starting point in defining the model. A nice property of the probabilistic modelling framework is that the posterior analysis is readily available in the posterior distribution of the latent variables. Using the probabilistic modelling approach to encode intuition was illustrated in the mixture models for tackling Collaborative Filtering. From the posterior analysis of the mixture components (or profiles), we could find some intuitive understanding, giving us hints that the patterns are not too far from our intuition. We have shown that other aspects come into play for the Collaborative Filtering data, like variability corresponding to the tendency that users have to rate rather high or low. In this respect, the interlaced GLM formulation was advantageous as variability of rating tendencies could be encoded very naturally in the model.

The quest for understanding is essentially a search for a simple model that has meaningful parameters. Unfortunately the requirement of simplicity of the model comes along with the risk that the model is highly biased such that it is impossible to have a good fit with the patterns present in the data. It is difficult to tell if a simple model with high bias offers useful information as it depends

on the dataset. For example, looking for a low-dimensional representation in data that are intrinsically high-dimensional can lead to a misleading analysis. The practitioner should know at least partially what information he is looking for in constructing a low-dimensional representation. On the other hand, the mixture models can be arbitrarily flexible (reducing thus the bias) by working with a sufficient number of components. However, doing so we can loose in terms of higher variance of the estimation (although the model selection procedure should find the right trade-off), but more importantly here we loose the simplicity of the analysis and the hope to find some interpretations from the parameters. All the difficulty lies precisely in finding the simple model which at the same time is not too biased.

In the end, we have the feeling that we have stirred a pot of unsolved problems and could only propose limited answers. We would like to point out some directions which we think could be valuable for research in data analysis, concerning the probabilistic modelling framework in particular.

One of the criticisms addressed to the probabilistic framework and the Bayesian selection principle is its lack of robustness when there is a mismatch between the assumptions made by the model and the data (either caused by the noise model, or simply because the constraints encoded in the hierarchy are not appropriate). For tackling this issue, combining the Bayesian selection with other selection strategies is probably a good solution. The difficulty is in keeping the learning procedure as simple as possible. In this sense, resampling methods are not always a viable solution due to their computational cost and the difficulty in finding a good region of the hyper-parameter space. There are certainly interesting combinations to be found with other estimators derived from learning theory. This is an ongoing research topic.

As explained above, the modularity of probabilistic modelling is very attractive but it is difficult to fully benefit from this modularity since the inference (i.e. the computation of posterior and marginal likelihood) becomes intractable when the model deviates from the standard combinations of distributions. There is hope that new approximattion schemes will be proposed to make the modularity even more useful. Already, variational approximation methods are becoming widespread in the derivation of variational Expectation-Maximisation algorithms. However, the technique is mainly used within a mean field approximation scheme which has also its constraints and limitations (see [17]). Other approximations are being proposed like the expectation-propagation approach. There is certainly some interesting work to be done there.

Ideally, the learning procedure should be made as automatic as possible in order to facilitate the work of the practitioner. His role should be limited to finding good model configurations matching the patterns in the data and he should not be concerned by the learning procedure itself. In fact, all models exposed in this thesis need some expertise to be used as they are subject to optimisation issues like multiple local optima. Common sense initialisation often avoids convergence to bad local optima, but at the beginning of the

analysis it is not always possible to know what a common sense initialisation is. This is inherent in non-convex optimisation and not only in probabilistic modelling. Designing simple learning schemes is a necessity in order to make the use of data analysis methods more widespread. There is however little hope that the issue can be completely solved and linear models which require less optimisation expertise have still a bright future. Also we have seen from the two real applications considered (regression on spectroscopy and Collaborative Filtering) that simple models (the PLS in the case of spectroscopy, the common preference configuration in the case of the Collaborative Filtering task) can already perform well. Trying to construct more complex configurations can improve the performances by some percent certainly, but not radically modify the results. One should wonder whether this gain is worth the effort.

Working on real datasets turned out to be a real challenge not only because of the important computational costs (in particular for the Collaborative Filtering datasets). There are several origins to the difficulties in working on real applications. First, formulating the problem is rarely simple when handling real data, especially at the stage of an exploratory analysis. Indeed, without much knowledge of the inherent patterns in the data, there is little reason to prefer one loss function over another, yet this choice dictates which patterns will be found. Second, we have copiously discussed the case of erroneous or atypical measurements and we have shown how important it is to care for these data. In fact, the way data are collected is an even more important aspect of the data as it can considerably bias the estimated patterns. One should be careful that the collecting procedure is sufficiently close to the exploitation conditions for the inference derived from the model to be meaningful. We did formulate some limits about the batch approach to analyse Collaborative Filtering datasets. This is an example where it would be useful to add a feed-back between the model inference and the collecting procedure; online configuration would certainly make sense. More generally, improving the synergy of the collecting procedure and modelling is necessary if we want to improve the performances and general usefulness of the analysis. For this purpose, active learning is interesting. The principle is to try to identify where additional data are required to improve estimation of the model. The probabilistic approach is appropriate as it has ready for use (with some precautions) assessments of the estimation uncertainty. Unfortunately, the synergy between collecting data and modelling is not easy to investigate in the context of academic research.

There are many great challenges still to be solved in the data analysis research field. These challenges concern theoretical aspects such as the improvement and simplification of the current estimators. There are also challenges concerning the design of computationally efficient learning strategies (in particular to handle the growing size of datasets in some applications). And finally, there is a need for simple data analysis tools to help the practitioner in his task. The probabilistic modelling framework has strong arguments to be part of the solutions...

Appendix A

Mathematical definitions

A.1 Probability Distributions

A.1.1 Bernoulli

The Bernoulli distribution defines the probabilities associated to a binary test (e.g. flipping a coin).

$$\mathcal{B}e(x|\beta) = \beta^x(1-\beta)^{1-x} \tag{A.1}$$

where $x \in \{0,1\}$, and if $x = 1$ represents a success (and $x = 0$ a failure), β is then the probability of success, $(0 \leq \beta \leq 1)$.

A.1.2 Binomial

The Binomial distribution gives the probability that for a number L of Bernoulli tests (see definition above), the number of successes be x.

$$\mathcal{B}n(x|\beta, L) = \frac{L!}{x!(L-x)!}\,\beta^x(1-\beta)^{1-x} \tag{A.2}$$

We have $x \in \{0, 1, \ldots, L\}$ and β is the probability of success $(0 \leq \beta \leq 1)$.

A.1.3 Gamma

The Gamma is a probability distribution defined over a positive random variable $x \geq 0$ such that

$$\mathcal{G}a(x|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)}\,x^{\alpha-1}\,e^{-\beta x} \tag{A.3}$$

with the constraint that $\alpha > 0$ and $\beta > 0$. The function $\Gamma(\cdot)$ appearing in the normalisation factor is known as the Gamma function.

A.1.4 Multinomial

The multinomial is a generalisation of the binomial distribution in the case where there are K possible outcomes for each test (instead of 2 for the simpler binomial). This is encoded by $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_K]^\top$, where β_k represent the probability of observing the kth outcome. We have the usual constraints $0 \leq \beta_k \leq 1$ and $\sum_k \beta_k = 1$.

$$\mathcal{M}n(\mathbf{x}|\boldsymbol{\beta}, L) = \frac{L}{\prod_k x_k!} \prod_{k=1}^{K} \beta_k^{x_k} \tag{A.4}$$

where $\mathbf{x} = [x_1, \ldots, x_K]^\top$ is the number of occurrences for each outcome and $L = \sum_k x_k$ is the total number of tests.

A.1.5 Gaussian (or Normal)

The multivariate Gaussian distribution of a random variable $\mathbf{x} \in \mathbb{R}^D$ with mean $\boldsymbol{\mu} \in \mathbb{R}^D$ and covariance matrix $\boldsymbol{\Sigma}$ (which is by definition symmetric positive definite) is defined by

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mid 2\pi\boldsymbol{\Sigma} \mid^{-\frac{1}{2}} \; e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})} \tag{A.5}$$

A.1.6 Student-t

The multivariate Student-t distribution of a random variable $\mathbf{x} \in \mathbb{R}^D$ with mean $\boldsymbol{\mu} \in \mathbb{R}^D$ is

$$\mathcal{S}t(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \nu) = \frac{\Gamma(\nu/2 + D/2)}{\Gamma(\nu/2)} \mid \nu\pi\boldsymbol{\Sigma} \mid^{-\frac{1}{2}} \left(1 + \frac{1}{\nu}(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right)^{-\frac{\nu+1}{2}} \tag{A.6}$$

where$\boldsymbol{\Sigma}$ is a $D \times D$ symmetric positive definite matrix (proportional to the covariance matrix of the distribution), and ν is called the degree of freedom of the distribution. When $\nu \to \infty$ the Student-t becomes a Gaussian distribution.

Physical principle Mid-IR spectra is very efficient at identifying pure substances because they tend to have very characteristic spectrum in this range with peaks attributed to particular bonds.

A.2 Exponential family and GLM framework

Let us first remind the definition of the exponential family (Eq. (3.27)). Distributions of a 1-dimensional random variable T belonging to the family can always be written as

$$P(t|\theta; \psi) = \exp\left\{\frac{t\theta - b(\theta)}{a(\psi)} + c(t, \psi)\right\} \tag{A.7}$$

In this expression, ψ is sometimes called the dispersion (or nuisance) parameter and θ the canonical parameter and $a(\cdot)$, $b(\cdot)$, $c(\cdot,\cdot)$ are some specific functions. One can show that the mean and variance of the distributions are given by

$$\mathbb{E}\{T\} = \mu = b'(\theta) \tag{A.8}$$

$$Var\{T\} = b''(\theta)a(\psi) = v(\mu)a(\psi) \tag{A.9}$$

The function $v(\mu)$ will be useful in the subsequent derivations.

In the Generalized Linear Model formalism, we are looking for a conditional target distribution $P(t|\mathbf{x};\beta,\psi)$ belonging to the exponential family. The mean of the distribution is imposed to be $\mu = g(\eta)$ where $\eta = \mathbf{x}^\top\beta$ and $g(\cdot)$ is called the link function (also called the inverse link function sometimes). If $\eta = \theta$ then $g(\cdot)$ is said to be the canonical link function.

Suppose that a set of input-targets pairs $\mathcal{D} = \{(\mathbf{x}_n, t_n)\}_{n=1}^N$ is observed, then the log likelihood of the parameters (β,ψ) is

$$\ell(\mathcal{D}) \equiv \log P(\{t_n\}|\{\mathbf{x}_n\};\beta,\psi) = \sum_n^N \frac{t_n\theta_n - b(\theta_n)}{a(\psi)} + c(t_n,\psi) \tag{A.10}$$

where $\theta_n = b'^{-1}(\mu_n)$, $\mu_n = g(\eta_n)$ and $\eta_n = \mathbf{x}_n^\top\beta$. The log likelihood is often optimized with respect to the vector β by a Newton-Raphson procedure. Differentiating (A.10) with respect to β leads to the following expressions (A.11) and (A.12) for the gradient and Hessian respectively

$$\nabla_\beta\ell(\mathcal{D}) = \sum_n^N (t_n - \mu_n)\frac{g'(\eta_n)}{v(\mu_n)}\mathbf{x}_n \tag{A.11}$$

$$\mathcal{H}_\beta\ell(\mathcal{D}) = \sum_n^N \left(\frac{g'(\eta_n)^2}{v(\mu_n)} - (t_n - \mu_n)\frac{g''(\eta_n)v(\mu_n) - g'(\eta_n)^2v'(\mu_n)}{v(\mu_n)^2}\right)\mathbf{x}_n\mathbf{x}_n^\top \tag{A.12}$$

where $\eta_n = \mathbf{x}_n^\top\beta$. The Newton-Raphson update step for β is then

$$\beta^{(new)} \leftarrow \beta^{(old)} - \mathcal{H}_\beta\ell(\mathcal{D})^{-1}\nabla_\beta\ell(\mathcal{D}) \tag{A.13}$$

Additionally, an estimate (see [63]) of the dispersion parameter can be computed from

$$a(\hat{\psi}) = \sum_n^N \frac{(t_n - \mu_n)^2}{v(\mu_n)} \tag{A.14}$$

If the regression problem is further regularized by means of a prior distribution $P(\beta)$, one should add the contribution of this prior distribution to the gradient and the Hessian.

Appendix B

Datasets

B.1 Near-Infrared spectroscopy

This section gives additional information about the near-infrared spectroscopy datasets used in the Chapter 3. The original spectra are plotted as well as the outlier detection procedure (described in Section 3.4.4).

B.1.1 Tecator

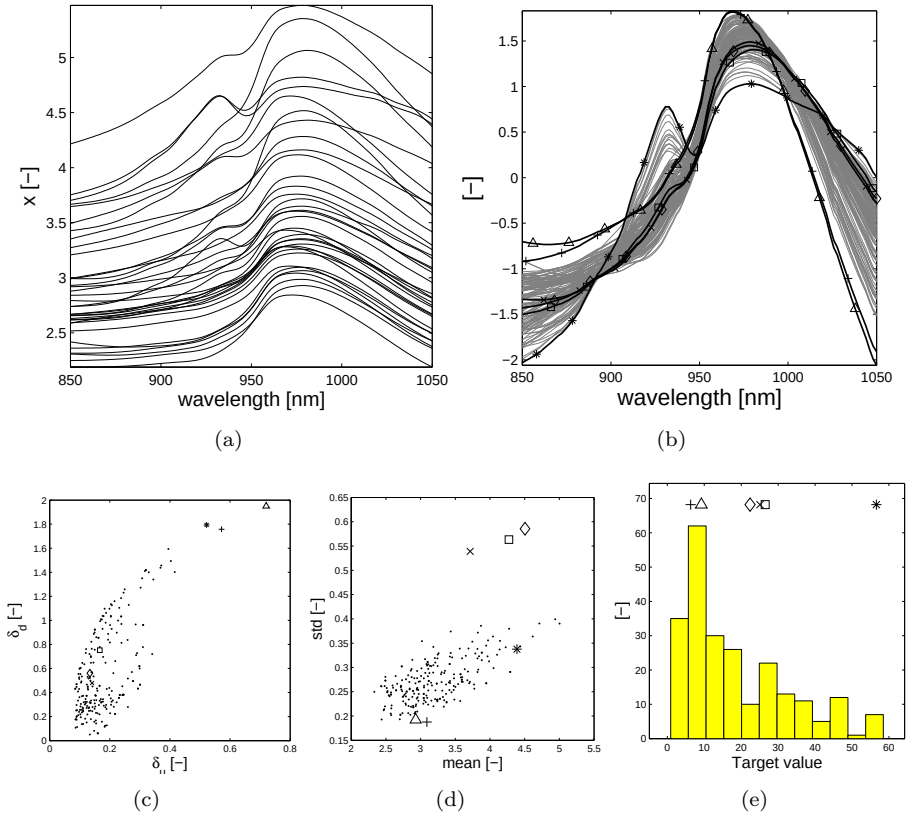
These data are recorded on a Tecator Infratec Food and Feed Analyzer working in the wavelength range 850 - 1050 nm by the Near Infrared Transmission (NIT) principle [132]. Each sample contains finely chopped pure meat with different moisture, fat and protein contents. The percentage of fat, water and protein obtained by analytic chemistry are reported.

From the 240 original spectra, 6 outliers where detected. Figure B.1 presents the original spectra and the outlier detection procedure.

B.1.2 Orange Juice

The dataset was provided by Pr. Marc Meurens, Université catholique de Louvain, BNUT unit. The dataset is available at "www.dice.ucl.ac.be/mlg". Please acknowledge the origin if you use it and/or publish about it. The dataset contains 218 near-infrared absorbance spectra of dry extract of orange juices. Each observation is a 700-channel spectrum in the 1100-2500 nm range. The goal is to predict the percentage level of saccharose.

From the 218 spectra, 7 outliers were detected. The visualisation of the procedure is given in Figure 3.8 on page 101.



(a) (b) (c) (d) (e)

Figure B.1: Visualisation of the *Tecator* spectra and the outlier detection procedure. (a) Original representation, (b) normalised representation (outliers marked by thick lines), (c) Mahalanobis distances: normalized vs. first derivative representations, (d) global mean vs. standard deviation of original spectra, (d) target distribution of fat content. Markers correspond to the assumed outliers.

B.1.3 Apple

The dataset was provided by Mohamed Hanafi from ENITIAA/ INRA (Unit of Sensometry and Chemometrics), Nantes (FRANCE). The dataset contains 337 near-infrared absorbance spectra of apples. Each observation is a 110-channel spectrum in the 1100-2190 nm range. The targets correspond to the crushing forces necessary to insert a stem in the skin of the apples.

In Figure B.2 the original spectra and their normalised representation are shown. Note that the spectra look already pre-processed and no outlier was detected.

B.1.4 Chimiometrie 2006

The dataset was proposed at the conference Chimiometrie 2006 [106] for a prediction competition. It contains 618 near-infrared spectra of dried and sieved soil samples. Each observation is a 700-channel spectrum in the 1100-2498 nm range. The prediction was aimed at three outputs, the proportion of Nitrogen, the Cation Exchange Capacity and the percentage of Carbon. Additionally, the dataset contains 207 validation spectra with missing targets.

Figure B.3 shows the original spectra and their normalised representation. From the calibration dataset, 11 outliers were detected.

B.1.5 Shootout 2006

The dataset was provided by (i) David J. Brown, Montana State University, Department of Land Resources and Environmental Sciences, USA; (ii) the National Soil Survey Center Soil Survey Laboratory (NSSC-SSL), Lincoln, NE; (iii) and by the Council for Near Infrared Spectroscopy, 128 N. Deer Run Drive, Lincoln University, PA 19352 via the International Diffuse Reflectance Conference (IDRC) and ShootOut2006.

The dataset contains 2761 calibration spectra and 1423 validation spectra of soil samples collected from all over the USA. Each spectrum is a 2151-channel recording in the 350-2500 nm wavelength range. There are 8 constituents measures associated with the spectra: (1) Kaolinite [KK], (2) Montmorillonte [MT], (3) Vermiculite [VR], (4) clay-size fraction (CL) [clay_g_kg], (5) cation exchange capacity (CEC) [cec_nh4], (6) inorganic carbon [IC_g_kg], (7) soil organic carbon (SOC) [SOC_g_kg], and (8) sand-size fraction [sand_g_kg]. The challenge was asking the prediction for three of these constituents: the clay-size fraction, the cation exchange capacity and the fraction of soil organic carbon.

Figure B.4 represents the Shootout spectra and the procedure used to detect the 31 outliers (out of the 2151 calibration spectra). Note that for the analysis of Chapter 3, only the clay size fraction and the soil organic carbon outputs were used.

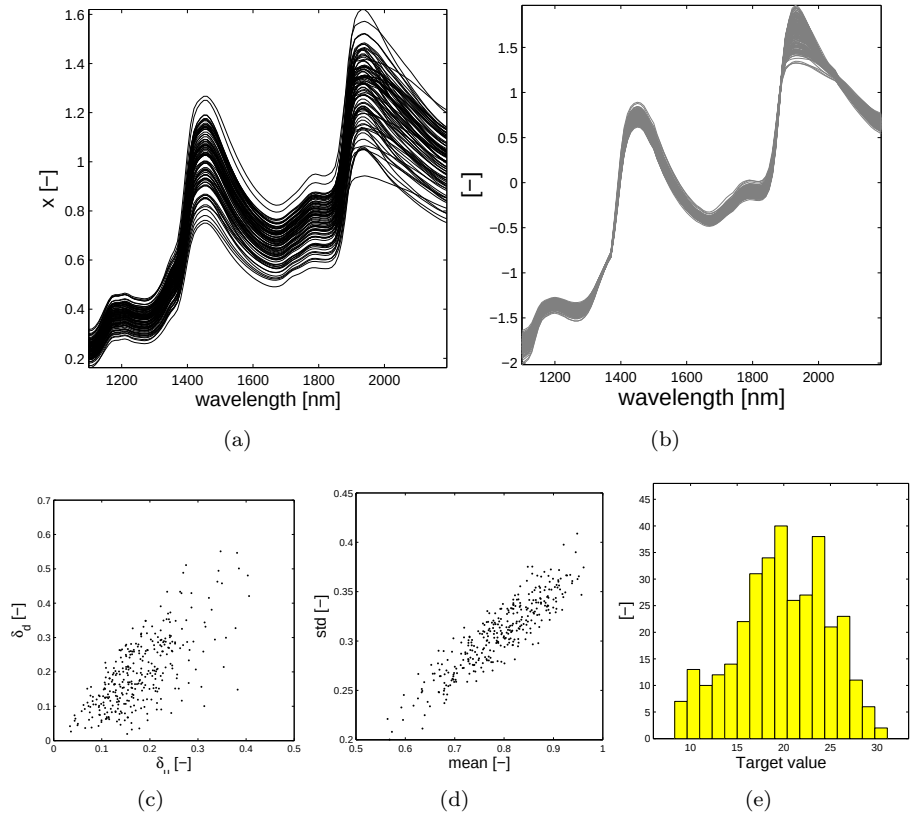
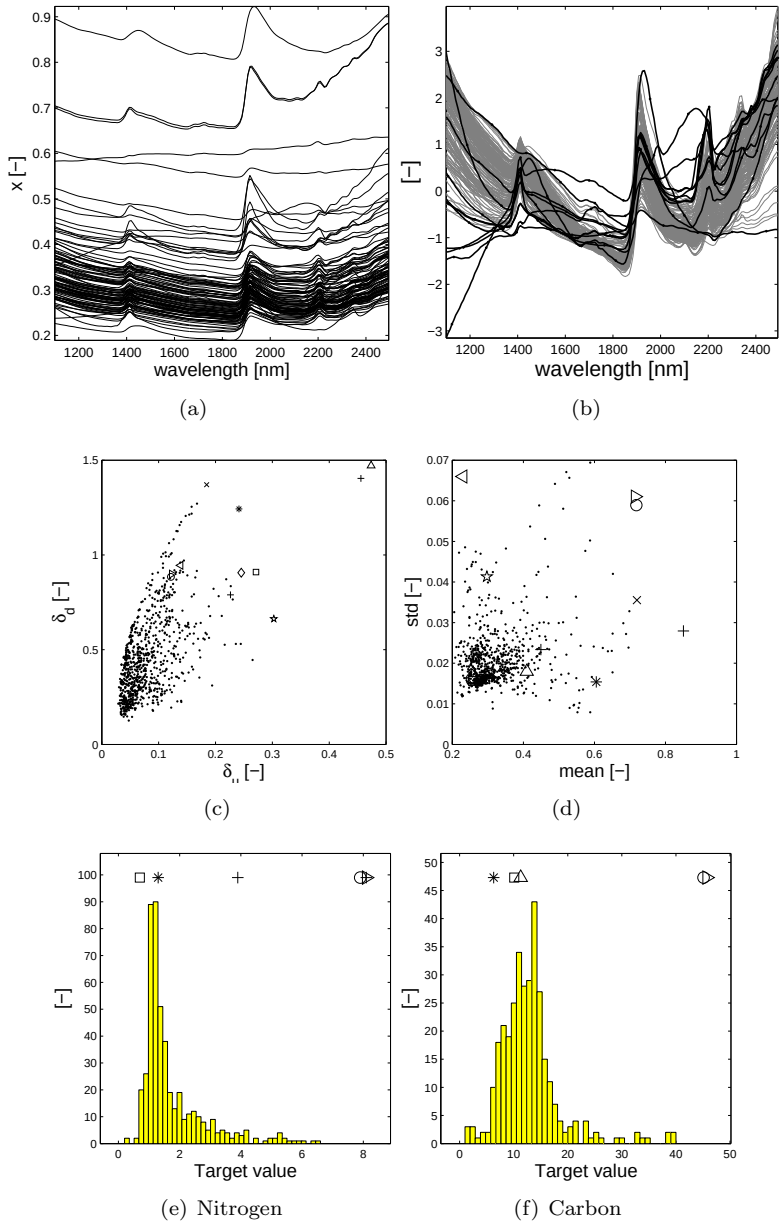


Figure B.2: Visualisation of the *Apple* spectra and the outlier detection procedure. (a) Original representation, (b) normalised representation (outliers marked by thick lines), (c) Mahalanobis distances: normalised vs. first derivative representations, (d) global mean vs. standard deviation of original spectra, (e) target distribution of crushing force applied. Markers correspond to the assumed outliers.

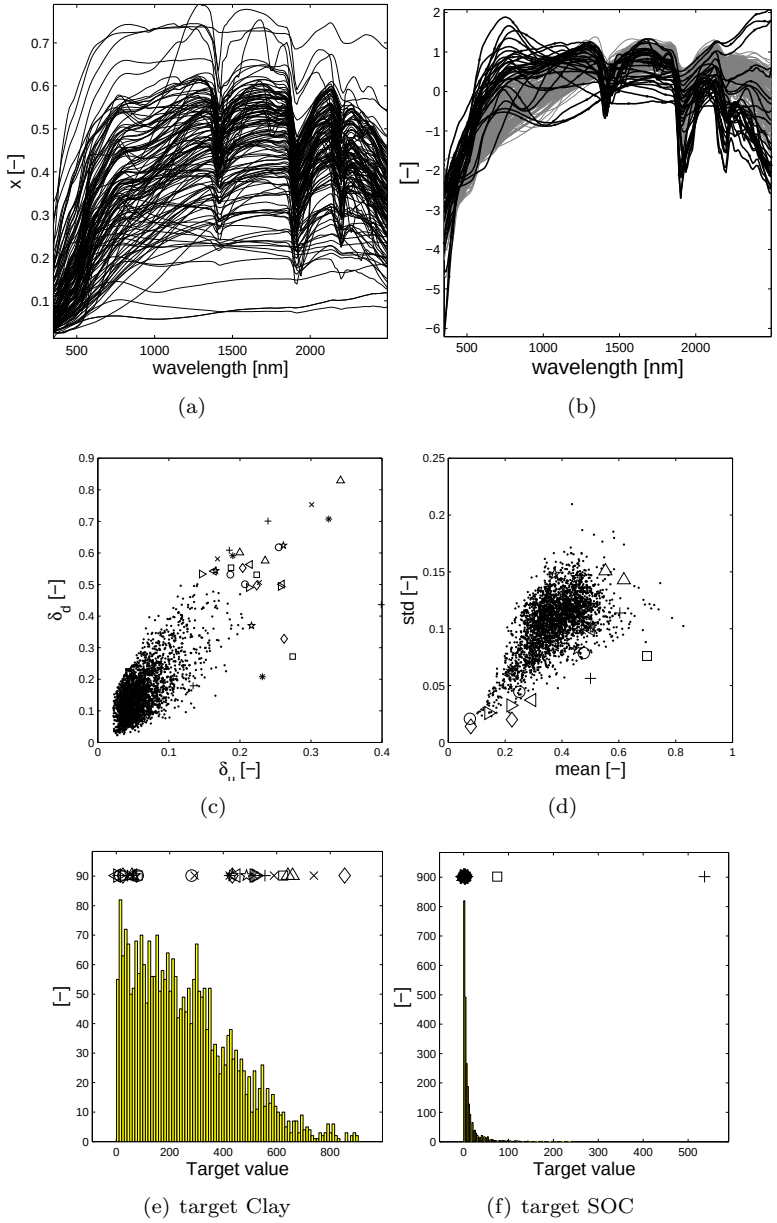


(a) (b)

(c) (d)

(e) Nitrogen (f) Carbon

Figure B.3: Visualisation of the *Chimiometrie 2006* spectra and the outlier detection procedure. (a) Original representation, (b) normalised representation (outliers marked by thick lines), (c) Mahalanobis distances: normalised vs. first derivative representations, (d) global mean vs. standard deviation of original spectra, target distributions of (e) proportion of Nitrogen, (f) the percentage of Carbon. Markers correspond to the assumed outliers.



(a) (b)

(c) (d)

(e) target Clay (f) target SOC

Figure B.4: Visualisation of the *Shootout 2006* spectra and the outlier detection procedure. (a) Original representation, (b) normalised representation (outliers marked by thick lines), (c) Mahalanobis distances: normalised vs. first derivative representations, (d) global mean vs. standard deviation of original spectra, target distributions of (e) the clay size fraction, (f) the soil organic carbon. Markers correspond to the assumed outliers.

Appendix C

Complementary results

C.1 Probabilistic latent linear subspaces

Here are further developments for the derivation of the EM algorithm in Section 2.3.1 on page 51.

C.1.1 Posterior distributions

To compute the posterior distributions of the latent variables, we write

$$\begin{aligned} &P(\{\boldsymbol{\chi}_n\}, \{\mathbf{u}_n\}, \{\mathbf{z}_n\} | \{\mathbf{y}_n\}) \\ &\quad = P(\{\boldsymbol{\chi}_n\} | \{\mathbf{u}_n\}, \{\mathbf{z}_n\}, \{\mathbf{y}_n\}) P(\{\mathbf{u}_n\} | \{\mathbf{z}_n\}, \{\mathbf{y}_n\}) P(\{\mathbf{z}_n\} | \{\mathbf{y}_n\}) \\ &\quad = \prod_n \prod_k P(\mathbf{x}_{nk} | u_{nk}, z_{nk} = 1, \mathbf{y}_n) P(u_{nk} | z_{nk} = 1, \mathbf{y}_n) P(z_{nk} = 1 | \mathbf{y}_n) \end{aligned} \quad \text{(C.1)}$$

Note that the latent variables $\mathbf{x}_{nk}$ and z_{nk} are not defined if $z_{nk} = 0$, so the factors $P(\mathbf{x}_{nk} | u_{nk}, z_{nk} = 0, \mathbf{y}_n)$ and $P(u_{nk} | z_{nk} = 0, \mathbf{y}_n)$ do not appear in the posterior distribution.

The different factors of this posterior distribution can be identified by applying the Bayes formula and making use of the factorised model defined in (2.18)-(2.21).

The posterior distributions of the low-dimensional latent vectors are given by

$$\begin{aligned} P(\mathbf{x}_{nk} | u_{nk}, z_{nk} = 1, \mathbf{y}_n) &\propto P(\mathbf{y}_n | \mathbf{x}_{nk}, u_{nk}, z_{nk} = 1)\, P(\mathbf{x}_{nk} | u_{nk}, z_{nk} = 1) \\ &\propto \mathcal{N}(\mathbf{y}_n | \mathbf{W}_k \mathbf{x}_{nk} + \boldsymbol{\mu}_k, u_{nk} \tau_k \mathbf{I}_D)\, \mathcal{N}(\mathbf{y}_{nk} | \mathbf{0}, u_{nk} \mathbf{I}_J) \\ &= \mathcal{N}(\mathbf{x}_{nk} | \tau_k \mathbf{C}_k \mathbf{W}_k^\top (\mathbf{y}_n - \boldsymbol{\mu}_k), \frac{1}{u_{nk}} \mathbf{C}_k) \end{aligned} \quad \text{(C.2)}$$

for all n and k and where $\mathbf{C}_k^{-1} = \tau_k \mathbf{W}_k^\top \mathbf{W}_k + \mathbf{I}_J$.

The posteriors of the precision variables are obtained from

$$\begin{aligned} P(u_{nk}|z_{nk}=1,\mathbf{y}_n) &\propto \int P(\mathbf{y}_n|\mathbf{x}_{nk},u_{nk},z_{nk}=1) \\ &\qquad \cdot P(\mathbf{x}_{nk}|u_{nk},z_{nk}=1)\,d\mathbf{x}_{nk}\; P(u_{nk}|z_{nk}=1) \\ &\propto \mathcal{N}(\mathbf{y}_n|\boldsymbol{\mu}_k,\frac{1}{u_{nk}}\boldsymbol{\Sigma}_k)\;\mathcal{G}a(u_{nk}|\tfrac{\nu_k}{2},\tfrac{\nu_k}{2}) \\ &= \mathcal{G}a(u_{nk}|\tfrac{D+\nu_k}{2},\tfrac{(\mathbf{y}_n-\boldsymbol{\mu}_k)^\top\boldsymbol{\Sigma}_k^{-1}(\mathbf{y}_n-\boldsymbol{\mu}_k)+\nu_k}{2}) \end{aligned} \tag{C.3}$$

for all n and k and where $\boldsymbol{\Sigma}_k = \mathbf{W}_k\mathbf{W}_k^\top + \tau_k^{-1}\mathbf{I}_D$. In this derivation, we made explicit the marginalisation over the latent vectors $\mathbf{x}_{nk}$ which is described in (2.4)-(2.6).

As the Gamma distribution is the conjugate distribution of the precision parameter of a Gaussian distribution, the posteriors of the precision parameters are also Gamma distributions.

Finally, the posterior probabilities of the indicator variables (called the *responsibilities)* are given by

$$\begin{aligned} P(z_{nk}=1|\mathbf{y}_n) &\propto \int\int P(\mathbf{y}_n|\mathbf{x}_{nk},u_{nk},z_{nk}=1)\,P(\mathbf{x}_{nk}|u_{nk},z_{nk}=1) \\ &\qquad \cdot P(u_{nk}|z_{nk}=1)\,d\mathbf{x}_{nk}\,du_{nk}\;P(z_{nk}=1) \\ &\propto \mathcal{S}t(\mathbf{y}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k,\nu_k)\;\pi_k \end{aligned} \tag{C.4}$$

for all n and k. In the first line, we made explicit the marginalisation discribed in (2.12)-(2.15). The posterior probability that the observation $\mathbf{y}_n$ was generated by the kth component is thus

$$P(z_{nk}=1|\mathbf{y}_n) = \frac{\pi_k\mathcal{S}t(\mathbf{y}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k,\nu_k)}{\sum_k \pi_k\mathcal{S}t(\mathbf{y}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k,\nu_k)} \tag{C.5}$$

C.1.2 Log complete likelihood

$$\begin{aligned} &P(\{\mathbf{y}_n\},\{\boldsymbol{\chi}_n\},\{\mathbf{u}_n\},\{\mathbf{z}_n\}) \\ &\quad = P(\{\mathbf{y}_n\}|\{\boldsymbol{\chi}_n\},\{\mathbf{u}_n\},\{\mathbf{z}_n\})P(\{\boldsymbol{\chi}_n\}|\{\mathbf{u}_n\},\{\mathbf{z}_n\})P(\{\mathbf{u}_n\}|\{\mathbf{z}_n\})P(\{\mathbf{z}_n\}) \\ &\qquad = \prod_n\prod_k \mathcal{N}(\mathbf{y}_n|\mathbf{W}_k\mathbf{x}_{nk}+\boldsymbol{\mu}_k,\frac{1}{u_{nk}\tau_k}\mathbf{I}_D)^{z_{nk}}\mathcal{N}(\mathbf{x}_{nk}|\mathbf{0},\frac{1}{u_{nk}}\mathbf{I}_J)^{z_{nk}} \\ &\qquad\qquad \cdot \mathcal{G}a(u_{nk}|\tfrac{\nu_k}{2},\tfrac{\nu_k}{2})^{z_{nk}}\pi_k^{z_{nk}} \end{aligned} \tag{C.6}$$

$$
\begin{aligned}
&\log P(\{\mathbf{y}_n\},\{\boldsymbol{\chi}_n\},\{\mathbf{u}_n\},\{\mathbf{z}_n\}) \\
&\quad= \sum_n \sum_k z_{nk} \Bigg\{ \frac{D}{2}\log\frac{u_{nk}\tau_k}{2\pi} - \frac{u_{nk}\tau_k}{2}\|\mathbf{y}_n - \boldsymbol{\mu}_k - \mathbf{W}_k\mathbf{x}_{nk}\|^2 \\
&\qquad\qquad + \frac{J}{2}\log\frac{u_{nk}}{2\pi} - \frac{u_{nk}}{2}\|\mathbf{x}_{nk}\|^2 \\
&\qquad - \log\Gamma\left(\frac{\nu_k}{2}\right) + \frac{\nu_k}{2}\log\frac{\nu_k}{2} + \left(\frac{\nu_k}{2}-1\right)\log u_{nk} - \frac{\nu_k}{2}u_{nk} \\
&\qquad\qquad + \pi_k \Bigg\} \qquad \text{(C.7)}
\end{aligned}
$$

Setting $Q(\cdot)$ to the posterior distribution derived above and taking the expectation of (C.7), we see that all we need to compute is $\bar{\rho}_{nk} \equiv \mathbb{E}_Q\{z_{nk}\}$, $\log\tilde{u}_{nk} \equiv \mathbb{E}_Q\{\log u_{nk}\}$, $\bar{u}_{nk} \equiv \mathbb{E}_Q\{u_{nk}\}$, $\bar{\mathbf{x}}_{nk} \equiv \mathbb{E}_Q\{\mathbf{x}_{nk}\}$ and $\bar{\mathbf{S}}_{nk} \equiv \mathbb{E}_Q\{\mathbf{x}_{nk}\mathbf{x}_{nk}^\top\}$. This last expectation comes from the following rearrangement in the first line

$$\mathbf{x}_{nk}^\top\mathbf{W}_k^\top\mathbf{W}_k\mathbf{x}_{nk} = \mathrm{tr}\left(\mathbf{W}_k\mathbf{x}_{nk}\mathbf{x}_{nk}^\top\mathbf{W}_k^\top\right) \qquad \text{(C.8)}$$

The resulting expectations are given in (2.30)-(2.34). Differentiating the expected log complete likelihood with respect to the parameters $\boldsymbol{\theta} \equiv \{(\mathbf{W}_k, \boldsymbol{\mu}_k, \tau_k, \nu_k, \pi_k)\}_{k=1}^K$ and equating to zero, we obtain the update rules in (2.35)-(2.39).

C.2 Regression on NIR spectroscopy

Tables C.1 and C.2 records the numerical values associated to the Figures 3.9 and 3.10 on page 104.

C.3 Collaborative Filtering

List of the most rated movies for each of the 7 profiles identified by the implicit rating aspect model (see Section 4.2.5 on page 126).

PROFILE 1: 16.3 %

1) Mission: Impossible (1996) | Action | Adventure | Mystery
2) Face/Off (1997) | Action | Sci-Fi | Thriller
3) Rock, The (1996) | Action | Adventure | Thriller
4) Speed (1994) | Action | Romance | Thriller
5) Fugitive, The (1993) | Action | Thriller
6) Independence Day (ID4) (1996) | Action | Sci-Fi | War
7) Armageddon (1998) | Action | Adventure | Sci-Fi | Thriller
8) Hunt for Red October, The (1990) | Action | Thriller
9) True Lies (1994) | Action | Adventure | Comedy | Romance
10) Die Hard 2 (1990) | Action | Thriller
11) Twister (1996) | Action | Adventure | Romance | Thriller
12) Clear and Present Danger (1994) | Action | Adventure | Thriller
13) Air Force One (1997) | Action | Thriller
14) Mummy, The (1999) | Action | Adventure | Horror | Thriller
15) Batman Returns (1992) | Action | Adventure | Comedy | Crime

PROFILE 2: 15.4 %

1) Star Wars: Episode VI - Return of the Jedi (1983) | Action | Adventure | Romance | Sci-Fi | War
2) Star Wars: Episode V - The Empire Strikes Back (1980) | Action | Adventure | Drama | Sci-Fi | War
3) Star Wars: Episode IV - A New Hope (1977) | Action | Adventure | Fantasy | Sci-Fi
4) Raiders of the Lost Ark (1981) | Action | Adventure
5) E.T. the Extra-Terrestrial (1982) | Children's | Drama | Fantasy | Sci-Fi

	Tecator		Orange juice		Apple	
	std. targets: 14.36		13.38		4.955	
$\mathcal{GP}_0$-D0	0.0958	(0.00831)	0.341	(0.00762)	0.55	(0.0104)
$\mathcal{GP}_0$-Du	0.0409	(0.00322)	<u>0.327</u>	(0.00984)	0.552	(0.0104)
$\mathcal{GP}_0$-D1	<u>0.0377</u>	(0.00436)	0.333	(0.0108)	<u>0.535</u>	(0.00961)
$\mathcal{GP}_0$-D2	0.0389	(0.00244)	0.351	(0.0113)	0.567	(0.0107)
A: $\mathcal{GP}_0$	0.0377	(0.00436)	**0.327**	(0.00984)	0.535	(0.00961)
B0: PLS-$\mathcal{D}_0$	0.204	(0.0128)	0.353	(0.0109)	0.531	(0.0138)
B*: PLS-$\mathcal{D}_*$	0.129	(0.0115)	0.391	(0.0133)	0.511	(0.00912)
C: MLP-ml	0.036	(0.0024)	0.59	(0.0664)	0.66	(0.0237)
D: MLP-cv	0.0376	(0.00314)	0.404	(0.0104)	0.582	(0.0149)
E: $\mathcal{GP}_\phi$	0.0462	(0.00892)	0.335	(0.0111)	0.613	(0.018)
F: $\mathcal{B} - \sigma_x$	0.0456	(0.0079)	0.34	(0.0111)	0.912	(0.0187)
G: $\mathcal{B} - \sigma_x, \lambda$	0.0488	(0.0108)	0.333	(0.0109)	0.644	(0.239)
H: $\mathcal{W} - \sigma_x$	0.227	(0.00685)	0.591	(0.012)	0.837	(0.0097)
I: $\mathcal{GP}_0 - \sigma_x$	0.0395	(0.00457)	0.331	(0.00883)	0.616	(0.0863)
J: $\mathcal{GP}_{ard}$	**0.033**	(0.00193)	0.334	(0.0131)	0.529	(0.00884)
K: $\mathcal{GP}_{s-ard}$	**0.034**	(0.00445)	0.337	(0.00958)	**0.506**	(0.00659)

Table C.1: NRMSE prediction performance of the regression configurations applied to the NIR spectroscopy datasets (Chapter 3). The empirical standard deviation of the target values is given in the second line. The table records the mean and standard deviation (under brackets) computed over the 20 repetitions of the 5-fold cross-validation scheme (Section 3.4). Bold results indicate the best configurations for each task (with a 95% confidence t-test to assess the significance of the difference). Underlined results indicate the selected preprocessing. Results continued in Table C.2

	Chimiom-nt		Chimiom-c		Shootout-cl		Shootout-soc	
	std. targets: 1.167		6.204		182		22.87	
$\mathcal{GP}_0$-D0	<u>0.437</u>	(0.0606)	0.8	(0.135)	0.543	(0.0183)	0.506	(0.00973)
$\mathcal{GP}_0$-Du	0.467	(0.0673)	0.737	(0.0733)	0.521	(0.0108)	0.508	(0.0116)
$\mathcal{GP}_0$-D1	0.437	(0.0261)	<u>0.708</u>	(0.0597)	**0.512**	(0.00766)	0.463	(0.0102)
$\mathcal{GP}_0$-D2	0.462	(0.0402)	0.787	(0.0478)	0.666	(0.00756)	<u>0.455</u>	(0.00978)
A: $\mathcal{GP}_0$	0.437	(0.0606)	0.708	(0.0597)	**0.512**	(0.00766)	0.455	(0.00978)
B0: PLS-$\mathcal{D}_0$	0.438	(0.0189)	0.605	(0.0265)	0.628	(0.0107)	0.594	(0.01)
B*: PLS-$\mathcal{D}_*$	0.438	(0.0189)	0.61	(0.0263)	0.658	(0.0654)	0.547	(0.0103)
C: MLP-ml	0.449	(0.0334)	0.789	(0.0621)	0.725	(0.0351)	0.929	(0.109)
D: MLP-cv	0.434	(0.0119)	**0.564**	(0.0242)	0.628	(0.0186)	0.492	(0.0261)
E: $\mathcal{GP}_\phi$	0.418	(0.00933)	0.781	(0.0487)	0.555	(0.00897)	0.493	(0.00801)
F: $\mathcal{B} - \sigma_x$	0.429	(0.0116)	0.66	(0.0746)	0.588	(0.0477)	0.5	(0.0129)
G: $\mathcal{B} - \sigma_x, \lambda$	0.437	(0.0466)	0.657	(0.0852)	0.587	(0.0403)	0.536	(0.0209)
H: $\mathcal{W} - \sigma_x$	0.729	(0.0147)	0.851	(0.0296)	0.789	(0.00669)	0.629	(0.0142)
I: $\mathcal{GP}_0 - \sigma_x$	**0.409**	(0.0267)	0.644	(0.0707)	0.524	(0.00999)	0.465	(0.0119)
J: $\mathcal{GP}_{ard}$	0.437	(0.0604)	0.676	(0.0483)	**0.512**	(0.0137)	0.432	(0.00717)
K: $\mathcal{GP}_{s-ard}$	**0.425**	(0.0694)	0.655	(0.0473)	**0.513**	(0.00887)	**0.424**	(0.00645)

Table C.2: NRMSE prediction performance, continued from Table C.1.

6) Star Wars: Episode I - The Phantom Menace (1999) | Action | Adventure | Fantasy | Sci-Fi
7) Terminator 2: Judgment Day (1991) | Action | Sci-Fi | Thriller
8) Terminator, The (1984) | Action | Sci-Fi | Thriller
9) Alien (1979) | Action | Horror | Sci-Fi | Thriller
10) Aliens (1986) | Action | Sci-Fi | Thriller | War
11) Blade Runner (1982) | Film-Noir | Sci-Fi
12) Jurassic Park (1993) | Action | Adventure | Sci-Fi
13) Men in Black (1997) | Action | Adventure | Comedy | Sci-Fi
14) Abyss, The (1989) | Action | Adventure | Sci-Fi | Thriller
15) Star Trek: The Wrath of Khan (1982) | Action | Adventure | Sci-Fi
———————————
PROFILE 3: 14.8 %
———————————
1) Stand by Me (1986) | Adventure | Comedy | Drama
2) Airplane! (1980) | Comedy
3) Princess Bride, The (1987) | Action | Adventure | Comedy | Romance
4) Ferris Bueller's Day Off (1986) | Comedy
5) Beetlejuice (1988) | Comedy | Fantasy
6) When Harry Met Sally... (1989) | Comedy | Romance
7) Monty Python and the Holy Grail (1974) | Comedy
8) Ghostbusters (1984) | Comedy | Horror
9) Breakfast Club, The (1985) | Comedy | Drama
10) Big (1988) | Comedy | Fantasy
11) Rain Man (1988) | Drama
12) Raising Arizona (1987) | Comedy
13) Back to the Future (1985) | Comedy | Sci-Fi
14) Fish Called Wanda, A (1988) | Comedy
15) Christmas Story, A (1983) | Comedy | Drama
———————————
PROFILE 4: 14.2 %
———————————
1) Forrest Gump (1994) | Comedy | Romance | War
2) Toy Story (1995) | Animation | Children's | Comedy
3) As Good As It Gets (1997) | Comedy | Drama
4) There's Something About Mary (1998) | Comedy
5) Clueless (1995) | Comedy | Romance
6) Aladdin (1992) | Animation | Children's | Comedy | Musical
7) League of Their Own, A (1992) | Comedy | Drama
8) Full Monty, The (1997) | Comedy
9) Groundhog Day (1993) | Comedy | Romance
10) Ghost (1990) | Comedy | Romance | Thriller
11) Four Weddings and a Funeral (1994) | Comedy | Romance
12) My Cousin Vinny (1992) | Comedy
13) Lion King, The (1994) | Animation | Children's | Musical
14) Wayne's World (1992) | Comedy
15) American President, The (1995) | Comedy | Drama | Romance
———————————
PROFILE 5: 14.0 %
———————————
1) Godfather, The (1972) | Action | Crime | Drama
2) Casablanca (1942) | Drama | Romance | War
3) One Flew Over the Cuckoo's Nest (1975) | Drama
4) Godfather: Part II, The (1974) | Action | Crime | Drama
5) Butch Cassidy and the Sundance Kid (1969) | Action | Comedy | Western
6) Wizard of Oz, The (1939) | Adventure | Children's | Drama | Musical
7) Psycho (1960) | Horror | Thriller
8) North by Northwest (1959) | Drama | Thriller
9) Annie Hall (1977) | Comedy | Romance
10) Taxi Driver (1976) | Drama | Thriller
11) Graduate, The (1967) | Drama | Romance
12) Gone with the Wind (1939) | Drama | Romance | War
13) Chinatown (1974) | Film-Noir | Mystery | Thriller
14) Citizen Kane (1941) | Drama
15) Amadeus (1984) | Drama
———————————
PROFILE 6: 13.7 %
———————————
1) Silence of the Lambs, The (1991) | Drama | Thriller
2) Braveheart (1995) | Action | Drama | War
3) Shawshank Redemption, The (1994) | Drama
4) Saving Private Ryan (1998) | Action | Drama | War
5) Fargo (1996) | Crime | Drama | Thriller
6) Pulp Fiction (1994) | Crime | Drama
7) L.A. Confidential (1997) | Crime | Film-Noir | Mystery | Thriller
8) Usual Suspects, The (1995) | Crime | Thriller
9) GoodFellas (1990) | Crime | Drama
10) Schindler's List (1993) | Drama | War
11) Titanic (1997) | Drama | Romance
12) Good Will Hunting (1997) | Drama
13) Reservoir Dogs (1992) | Crime | Thriller
14) Sixth Sense, The (1999) | Thriller
15) Sling Blade (1996) | Drama | Thriller
———————————
PROFILE 7: 11.5 %
———————————
1) American Beauty (1999) | Comedy | Drama
2) Being John Malkovich (1999) | Comedy
3) Gladiator (2000) | Action | Drama
4) Shakespeare in Love (1998) | Comedy | Romance

```
5) X-Men (2000) | Action | Sci-Fi
6) Austin Powers: The Spy Who Shagged Me (1999) | Comedy
7) Sixth Sense, The (1999) | Thriller
8) American Pie (1999) | Comedy
9) High Fidelity (2000) | Comedy
10) Chicken Run (2000) | Animation | Children's | Comedy
11) Election (1999) | Comedy
12) Erin Brockovich (2000) | Drama
13) Talented Mr. Ripley, The (1999) | Drama | Mystery | Thriller
14) Fight Club (1999) | Drama

15) Blair Witch Project, The (1999) | Horror
```

Bibliography

[1] D. M. Allen. The relationship between variable selection and data augmentation and a method for prediction. *Technometrics*, 16:125–127, 1974.

[2] A. Ambroladze, E. Parrado-Hernandez, and J. Shawe-Taylor. Tighter pac-bayes bounds. In *Advances in Neural Information Processing Systems 19*, 2007.

[3] C. Archambeau. *Probabilistic Models in Noisy Environments - And their Application to a Visual Prosthesis for the Blind*. PhD thesis, Université catholique de Louvain, Louvain-la-Neuve, Belgium, 2005.

[4] C. Archambeau, N. Delannay, and M. Verleysen. Robust probabilistic projections. In *ICML 2006, 23th International Conference on Machine Learning*, pages 33–40, Pittsburgh (USA), 2006.

[5] C. Archambeau, N. Delannay, and M. Verleysen. Robust probabilistic projections. In W. W. Cohen and A. Moore, editors, *23rd International Conference on Machine Learning (ICML)*, pages 33–40. ACM, 2006.

[6] C. Archambeau, N. Delannay, and M. Verleysen. Mixtures of robust probabilistic principal component analysers. In *ESANN 2007, European Symposium on Artificial Neural Networks Advances in Computational Intelligence and Learning*, Bruges (Belgium), 2007.

[7] C. Archambeau and M. Verleysen. Manifold constrained variational mixtures. In E. O. W. Duch, J. Kacprzyk and S. Zadrozny, editors, *Artificial Neural Networks: Formal Models and Their Applications*, Lecture Notes in Computer Science (LNCS 3697), pages 279–284. Springer, 2005.

[8] C. Atkeson, A. Moore, and S. Schaal. Locally weighted learning. *AI Review*, 11:11–73, April 1997.

[9] H. Attias. Inferring parameters and structure of latent variable models by variational bayes. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 21–30, San Francisco, CA, 1999. Morgan Kaufmann.

[10] F. Bach and M. Jordan. A probabilistic interpretation of canonical correlation analysis. (688), 2005.

[11] A. R. Barron. Universal Approximation Bounds for Superpositions of a Sigmoidal Function. *IEEE Trans. Information Theory*, 39(3):930–945, May 1993.

[12] Y. Bengio. Gradient-based optimization of hyperparameters. *Neural Comput.*, 12(8):1889–1900, 2000.

[13] N. Benoudjit, D. François, M. Meurens, and M. Verleysen. Spectrophotometric variable selection by mutual information. *Chemometrics and Intelligent Laboratory Systems*, 74(2), 2004.

[14] G. Birkhoff and C. R. de Boor. Piecewise polynomial interpolation and approximation. In H. L. Garabedian, editor, *Proc. General Motors Symposium*, pages 164–190, New York, Amsterdam, 1965. Elsevier.

[15] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[16] C. M. Bishop. Bayesian pca. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 382–388, Cambridge, MA, USA, 1999. MIT Press.

[17] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[18] C. M. Bishop, M. Svensen, and C. K. I. Williams. Gtm: The generative topographic mapping. *Neural Computation*, 10(1):215–234, 1998.

[19] D. Blei, T. Griffiths, M. Jordan, and J. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. In *In Advances in Neural Information Processing Systems 16*. Cambridge, MA, MIT Press, 2004.

[20] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[21] L. Bookstein. *Morphometric Tools for Landmark Data: Geometry and Biology*. Cambridge University Press, Cambridge, Mass, 1991.

[22] M. Brand. Incremental singular value decomposition of uncertain data with missing values. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part I*, pages 707–720, London, UK, 2002. Springer-Verlag.

[23] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[24] D. J. Brown, K. D. Shepherd, M. G. Walsh, M. D. Mays, and T. G. Reinsch. Global soil characterization with vnir diffuse reflectance spectroscopy. *Geoderma*, 2006.

[25] L. Brozovsky and V. Petricek. Recommender system for online dating service. In *Proceedings of Znalosti 2007 Conference*, Ostrava, 2007. VSB.

[26] J. Canny. Collaborative filtering with privacy via factor analysis. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 238–245, New York, NY, USA, 2002. ACM Press.

[27] G. C. Cawley and N. L. C. Talbot. Fast leave-one-out cross-validation of sparse least-squares support vector machines. *Neural Networks*, 17(10):1467–1475, December 2004.

[28] G. C. Cawley and N. L. C. Talbot. Preventing over-fitting during model selection via bayesian regularisation of the hyper-parameters. *Journal of Machine Learning Research*, 8:841–861, 2007.

[29] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Mach. Learn.*, 46(1-3):131–159, 2002.

[30] K.-M. Chung, W.-C. Kao, C.-L. Sun, L.-L. Wang, and C.-J. Lin. Radius margin bounds for support vector machines with the rbf kernel. *Neural Computation*, 15(11):2643–2681, 2003.

[31] M. Collins, S. Dasgupta, and R. E. Schapire. A generalization of principal components analysis to the exponential family. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[32] I. Daubechies. Orthonormal bases of compactly supported wavelets ii: variations on a theme. *SIAM J. Math. Anal.*, 24(2):499–519, 1993.

[33] D. DeCoste. Collaborative prediction using ensembles of maximum margin matrix factorizations. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 249–256, New York, NY, USA, 2006. ACM Press.

[34] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[35] N. Delannay, C. Archambeau, and M. Verleysen. Automatic adjustment of discriminant adaptive nearest neighbor. In *ICPR 2006, 18th International Conference on Pattern Recognition*, pages vol. 2, pp. 525–528, Hong Kong (China), 2006.

[36] N. Delannay, F. Rossi, B. Conan-Guez, and M. Verleysen. Functional radial basis function networks. In *ESANN 2004, European Symposium on Artificial Neural Networks*, pages 313–318, Bruges (Belgium), 2004.

[37] N. Delannay and M. Verleysen. Collaborative filtering with interlaced generalized linear models. In *ESANN 2007, European Symposium on Artificial Neural Networks Advances in Computational Intelligence and Learning*, Bruges, 2007.

[38] N. Delannay and M. Verleysen. Collaborative filtering with interlaced generalized linear models. *submitted to Neurocomputing*, 2007.

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

[40] L. Devroye, L. Györfi, and G. Lugosi. *A probabilistic theory of pattern recognition*. Springer-Verlag, New York, 1996.

[41] B. Efron, T. Hastie, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

[42] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, London, 1993.

[43] B. S. Everitt. *An Introduction to Latent Variable Models*. Chapman and Hall, 1983.

[44] J. Fan and I. Gijbels. *Local Polynomial Modelling and Its Applications*. Chapman & Hall, 1996.

[45] F. Fouss, J.-M. Renders, A. Pirotte, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):355–369, 2007.

[46] D. François. *High-dimensional data analysis: optimal metrics and feature selection*. Ph.d., Université catholique de Louvain, Louvain-la-Neuve, Belgium, 2007.

[47] Z. Ghahramani and G. E. Hinton. The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, Dept. of Computer Science, University of Toronto, 1996.

[48] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[49] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003.

[50] B. Hammer and T. Villmann. Generalized relevance learning vector quantization. *Neural Networks*, 15(8-9):1059–1068, 2002.

[51] T. Hastie, A. Buja, and R. Tibshirani. Penalized discriminant analysis. *Annals of Statistics*, 23:73–102, 1995.

[52] T. Hastie and R. Tibshirani. Generalized additive models. *Statistical Science*, 1:297–318, 1986.

[53] T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(6):607–616, 1996.

[54] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2001.

[55] S. Haykin. *Neural networks, A comprehensive Foundation*. Macmilan, New York, 1994.

[56] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.

[57] G. E. Hinton, P. Dayan, and M. Revow. Modeling the manifolds of images of handwritten digits. *IEEE Transactions on Neural Networks*, 8:65–74, 1997.

[58] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67, 1970.

[59] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, New York, NY, USA, 1999. ACM Press.

[60] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, 2004.

[61] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366, 1989.

[62] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 1933.

[63] W. Härdle, M. Müller, S. Sperlich, and A. Werwatz. *Nonparametric and Semiparametric Models*. Springer-Verlag, Heidelberg, Germany, 2004.

[64] K. Huang, Y. Ma, and R. Vidal. Minimum effective dimension for mixtures of subspaces: A robust GPCA algorithm and its applications. In *2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 631–638, 2004.

[65] P. Huber. *Robust Statistics*. John Wiley and Sons, New York, 1981.

[66] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[67] R. Jin, L. Si, and C. Zhai. A study of mixture models for collaborative filtering. *Information Retrieval*, 9(3):357–382, 2006.

[68] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[69] I. T. Jolliffe. *Principal Component Analysis, second edition*. Springer-Verlag, New York, 2002.

[70] M. I. Jordan. Dirichlet processes, chinese restaurant processes and all that. Tutorial presentation at the NIPS Conference, 2005.

[71] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, pages 1137–1145, 1995.

[72] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[73] T. Kohonen. *Self-Organizing Maps*. Springer Verlag, New York, 1997.

[74] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[75] N. D. Lawrence. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of Machine Learning Research*, 6:1783–1816, 2005.

[76] J. A. Lee and M. Verleysen. *Nonlinear Dimensionality Reduction*. Springer, to be published in 2007.

[77] C. Liu and D. B. Rubin. ML estimation of the t distribution using EM and its extensions, ECM and ECME. *Statistica Sinica*, 5:19–39, 1995.

[78] C. Loader. *Local Regression and Likelihood*. 1999.

[79] D. G. Lowe. Similarity metric learning for a variable-kernel classifier. Technical report, Vancouver, BC, Canada, Canada, 1993.

[80] D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.

[81] D. J. C. MacKay. A practical bayesian framework for backpropagation networks. *Neural Comput.*, 4(3):448–472, 1992.

[82] D. J. C. MacKay. Bayesian non-linear modelling for the prediction competition. In *ASHRAE Transactions, V.100, Pt.2*, pages 1053–1062, Atlanta Georgia, 1994. ASHRAE.

[83] D. J. C. MacKay. *Information theory, inference, and learning algorithms.* Cambridge University Press, Cambridge, 2003.

[84] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 1(1):24–45, 2004.

[85] B. Marlin. Collaborative filtering: A machine learning perspective. Master thesis, 2004.

[86] B. Marlin. Modeling user rating profiles for collaborative filtering. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2004. MIT Press.

[87] B. M. Marlin, R. S. Zemel, S. Roweis, and M. Slaney. Collaborative filtering and the missing at random assumption. In *UAI 2007: Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, 2007.

[88] R. A. Marrona, D. R. Martin, and V. J. Yohai. *Robust Statistics: Theory and Methods.* Wiley, New York, 2006.

[89] G. Matheron. The intrinsic random functions and their applications. *Advances in Applied Probability*, 5(3):439–468, 1973.

[90] P. McCullagh and J. A. Nelder. *Generalized Linear Models.* Chapman and Hall, New York, 1989.

[91] T. P. Minka. Expectation propagation for approximate bayesian inference. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 362–369, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[92] M. F. Moller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533, 1993.

[93] J. Moody and C. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294, 1989.

[94] E. A. Nadaraya. On estimating regression. *Theory of probability and its applications*, 10:186–190, 1964.

[95] R. M. Neal. *Bayesian Learning for Neural Networks.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996.

[96] R. M. Neal. *Neural networks and machine learning*, chapter Assessing relevance determination methods using DELVE, pages 97–129. Springer-Verlag, Berlin, 1998.

[97] R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer, 1998.

[98] J. A. Nelder and R. W. M. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society. Series A*, 135(3):370–384, 1972.

[99] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2000.

[100] K. H. Norris and J. R. Hart. Principles and methods of measuring moisture in liquids and solids. In *Proceedings of the 1963 International Symposium on Humidity and Moisture*, pages 19–25, NY, 1965.

[101] M. Oja, S. Kaski, and T. Kohonen. Bibliography of self-organizing map (som) papers: 1998-2001 addendum.

[102] C. Ong, A. Smola, and R. Williamson. Learning the kernel with hyperkernels. *Journal of Machine Learning Research*, 6:1043–1071, 07 2005.

[103] P. Paatero and U. Tapper. Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values. *Envirometrics*, 5:111–126, 1994.

[104] J. Park and I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3(2):246–257, 1991.

[105] D. Peel and G. J. McLachlan. Robust mixture modelling using the t distribution. *Statistics and Computing*, 10:339–348, 2000.

[106] J. F. Pierna and P. Dardenne. Soil regression as a chemometric challenge at chimiométrie 2006. *submitted to Chemometrics and Intelligent Laboratory Systems*, 2007.

[107] T. Poggio and F. Girosi. Networks for approximation and learning. *Proc. IEEE*, 78(9):1481–1497, September 1990.

[108] J. Ramsay and B. Silverman. *Functional Data Analysis*. Springer Series in Statistics. Springer Verlag, June 1997.

[109] J. O. Ramsay and X. Li. Curve registration. *Journal of the Royal Statistical Society: Series B*, 60(2):351–363, 1998.

[110] C. E. Rasmussen. *Evaluation of Gaussian Processes and other Methods for Non-Linear Regression*. PhD thesis, Department of Computer Science, University of Toronto, Canada, 1996.

[111] C. E. Rasmussen and Z. Ghahramani. Infinite mixtures of gaussian process experts. In S. G. Z. Dietterich, Thomas G.; Becker, editor, *Advances in Neural Information Processing Systems 14*, 2002.

[112] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, Massachusetts, 2006.

[113] P. Resnik, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *ACM, Conference on Computer Supported Cooperative Work*, pages 175–186. ACM, 1994.

[114] S. Richardson and P. Green. On bayesian analysis of mixtures with an unknown number of components. *J. Roy. Statist. Soc.*, 59:731–792, 1996.

[115] F. Rossi and B. Conan-Guez. Functional multi-layer perceptron: a non-linear tool for functional data analysis. *Neural Networks*, 18(1):45–60, January 2005.

[116] F. Rossi, N. Delannay, B. Conan-Guez, and M. Verleysen. Representation of functional data in neural networks. *Neurocomputing*, 64:183–210, 2005.

[117] S. T. Roweis. EM algorithms for PCA and SPCA. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10 (NIPS)*. The MIT Press, 1998.

[118] D. B. Rubin and D. T. Thayer. EM algorithms for ML factor analysis. *Psychometrika*, 47(1):69–76, 1982.

[119] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[120] L. K. Saul, T. Jaakkola, and M. I. Jordan. Mean field theory for sigmoid belief networks. Technical report, Cambridge, MA, USA, 1996.

[121] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, Mass., 2002.

[122] J. Shao and D. Tu. *The Jackknife and Bootstrap*. Springer Series in Statistics, 1995.

[123] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

[124] S. Shoham. Robust clustering by deterministic agglomeration EM of mixtures of multivariate t-distributions. *Pattern Recognition*, 35(5):1127–1142, 2002.

[125] P. Sollich and C. K. I. Williams. Using the equivalent kernel to understand gaussian process regression. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1313–1320, Cambridge, MA, 2005. MIT Press.

[126] N. Srebro and T. Jaakkola. Weighted low rank approximations. In *ICML 2003: 20th International Conference on Machine Learning*, pages 720–727, 2003.

[127] N. Srebro, J. D. M. Rennie, and T. S. Jaakola. Maximum-margin matrix factorization. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances In Neural Information Processing Systems 17*. MIT Press, 2005.

[128] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society B*, 36(2):111–147, 1974.

[129] M. Stone. An asymptotic equivalence of choice of model by cross-validation and akaike's criterion. *Journal of the Royal Statistical Society B*, 39(1):44–47, 1977.

[130] J. Suykens, T. V. Gestel, J. D. Brabanter, B. D. Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002.

[131] M. Tenenhaus. *La régression PLS. Théorie et pratique*. Technip., Paris, 2002.

[132] H. H. Thodberg. A review of Bayesian neural networks with an application to near infrared spectroscopy. *IEEE Trans. on Neural Networks*, 7(1):56–72, 1996.

[133] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996.

[134] A. N. Tikhonov and V. Y. Arsenin. *Solutions of ill-posed problems*. V. H. Winston & Sons, Washington, D.C., 1977.

[135] M. E. Tipping. Sparse Bayesian learning and the Relevance Vector Machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

[136] M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2):443–482, 1999.

[137] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society B*, 61:611–622, 1999.

[138] M. E. Tipping and A. . Faul. Fast marginal likelihood maximisation for sparse bayesian models. In C. M. Bishop and B. J. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, San Francisco, 2003. Morgan Kaufmann.

[139] D. Titterington, A. Smith, and U. Makov. *Statistical Analysis of Finite Mixture Distributions*. Wiley, 1985.

[140] E. Tony Davies. A celebration of near infrared spectroscopy. NIR News, Vol. 16, October 2005.

[141] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[142] V. N. Vapnik. *The nature of statistical learning theory.* Springer, 2000.

[143] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probab. and its Applications*, 16(2):264–180, 1971.

[144] G. Wahba. *Splines model for observational data.* CBMS-NSF Regional Conference Series in Applied Mathematics, 1990.

[145] S. Waterhouse, D. MacKay, and T. Robinson. Bayesian methods for mixtures of experts. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 351–357. The MIT Press, 1996.

[146] G. S. Watson. Smooth regression analysis. *Sankhya*, A26:359–372, 1964.

[147] H. Wold. Estimation of principal components and related models by iterative least squares. In P. Krishnaiah, editor, *Multivariate Analysis.* Academic Press, 1966.

[148] L. Xu and A. L. Yuille. Robust principal component analysis by self-organizing rules based on statistical physics approach. *IEEE Transactions on Neural Networks*, 6(1):131–143, 1995.

[149] K. Yu, V. Tresp, and S. Yu. A nonparametric hierarchical bayesian framework for information filtering. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 353–360, New York, NY, USA, 2004. ACM Press.

[150] K. Yu, S. Yu, and V. Tresp. Dirichlet enhanced latent semantic analysis. In R. G. Cowell and Z. Ghahramani, editors, *AISTATS 2005*, pages 437–444. Society for Artificial Intelligence and Statistics, 2005.

[151] S. Yu, K. Yu, V. Tresp, and H.-P. Kriegel. Collaborative ordinal regression. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 1089–1096, New York, NY, USA, 2006. ACM Press.

[152] H. Zhu and R. Rohwer. No free lunch for cross-validation. *Neural Compution*, 8(7):1421–1426, 1996.