Discriminative learning for structured outputs and environments

Simon Cousins
Department of Computer Science
University College London

This dissertation is submitted for the degree of
Doctor of Philosophy

2018

I would like to dedicate this thesis to my loving parents ...

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text. This dissertation contains less than 65,000 words including appendices, bibliography, footnotes, tables and equations and has less than 150 figures.

Simon Cousins
2018

Acknowledgements

First I would like to thank my supervisor, Prof. John Shawe-Taylor, for his help and guidance over these last years. The breadth and depth of his knowledge never failed to amaze me and his ability to explain complex concepts in a simple and intuitive manner was a blessing. I would like to thank all of those in the computer science department who ungrudgingly gave their time to engage in research discussions, and act as a sounding board for ideas, however ill-conceived they may have been. A special thanks to the Centre for Doctoral Training in Financial Computing for their generous scholarship, which made this thesis possible. Furthermore I would like to thank my colleagues at Stratagem Technologies and Realedge Associates who I had the pleasure to work with during my studies. They empowered me with the skills needed to bring new ideas into fruition and helped me to bridge the gap between academia and industry. Finally, I would like thank my parents for their insistent nagging and unwavering impatience. Admittedly, it was much needed and brought this thesis to its conclusion, thank you.

Abstract

Machine learning methods have had considerable success across a wide range of applications. Much of this success is due to the flexibility of learning algorithms and their ability to tailor themselves to the requirements of the particular problem. In this thesis we examine methods that seek to exploit the underlying structure of a problem and make the best possible use of the available data. We explore the structural nature of two different problems, binary classification under the uncertainty of input relationships, and multi-label output learning of Markov networks with unknown graph structures.

From the input perspective, we focus on binary classification and the problems associated with learning from limited amounts of data. In particular we pay attention to moment based methods and investigate how to deal with the uncertainty surrounding the estimate of moments using either small or noisy training samples. We present a worst-case analysis and show how the high probability bounds on the deviation of the true moments from their empirical counterparts can be used to generate a regularisation scheme that takes into consideration the relative amount of information that is available for each class. This results in a binary classification algorithm that directly minimises the worst case future misclassification rate, whilst taking into consideration the possible errors in the moment estimates.

This algorithm was shown to outperform a number of traditional approaches across a range of benchmark datasets, doing particularly well when training was limited to small amounts of data. This supports the idea that we can leverage the class specific regularisation scheme and take advantage of the uncertainty of the datasets when creating a predictor. Further encouragement for this approach was provided during the high-noise experiments, predicting the directional movement of popular currency pairs, where moment based methods outperformed those using the peripheral point of the class-conditional distributions.

From the output perspective, we focus on the problem of multi-label output learning over Markov networks and present a novel large margin learning method that leverages the correlation between output labels. Our approach is agnostic to the output graph structure and it simultaneously learns the intrinsic structure of the outputs, whilst finding a large margin separator. Based upon the observation that the score function over the complete output graph is given by the expectation of the score function over all spanning trees, we formulate the problem as an L_1-norm multiple kernel learning problem where each spanning tree over the complete output graph gives rise to a particular instance of a kernel.

We show that this approach is comparable to state-of-the-art approaches on a number of benchmark multi-label learning problems. Furthermore, we show how this method can be applied to the problem of predicting the joint movement of a group of stocks, where we not only infer the directional movement of individual stocks but also uncover insights on the input-dependent relationships between them.

Impact Statement

This thesis explores methods for making the best possible use of the data that has been made available. We approach this from two different perspectives.

The first part of the thesis will be of most interest to academics working in the field of generalisation and error bound analysis. We present an extension to current moment based approaches, addressing the issue of moment uncertainty by directly including it into an optimisation scheme to minimise worst case future performance. The framework for doing this is general and can be extended across a number of other domains. We have outlined several promising areas where we believe our approach is appropriate and can be readily adopted, which could form part of a wider research agenda in the future. This paper was published in the Machine Learning Journal and can be found here [Cousins and Shawe-Taylor, 2017]

Our new classification algorithm can be added to the arsenal of existing methods available to practitioners and distinguishes itself from other methods by having an implicit class-specific regularisation scheme. We believe it will be of particular interest to those working with small and/or noisy datasets, where our method was shown to perform favourably during experiments.

In the second part of the thesis we look at the problem of assigning labels to several output variables that are possibly related and depend on some arbitrary input. This work is of most interest to academics working in the fields of structured prediction and multiple kernel learning. We brought together these two fields by posing the problem of learning structure on a graph as multiple kernel learning with trees. This produced an intuitive scheme for dealing with problems with unknown or intractable output relationships, and is general enough to be extended beyond the binary output labellings that we considered. An alternative inference scheme over connected graphs was proposed, which provided in sights on how to guide an inference scheme based upon the observation of the input variable. This is something that could form an interesting direction for future research, and the preliminary results were presented at a *Neural Information Processing Systems* workshop on optimisation [Cousins et al.].

Practitioners are increasingly encountering high dimensional datasets where the relationships between variables is not known a priori. We have presented a new methodology that aims to find intuitive and interpretable structures that represent the underlying dataset. We believe that this approach can be deployed beyond the relatively straightforward examples we gave on conditional correlations between stocks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine learning is a data-driven learning mechanism that uses the incomplete information contained within the training examples to formulate predictions regarding unseen examples in the future. The goal is to find a function that is capable of summarising these relationships, and one that can be used to make predictions in the future. The success of these learning methods is measured by how well the function learned on the training sample generalises to unseen examples. The study of the generalisation ability of a function is outlined in the theoretical work of statistical learning theory. Broadly speaking, statistical learning theory relates the generalisation ability of a function to the hypothesis class from which it was chosen, the number of examples that were used during learning and its performance on these examples. Functions obtained from simple hypothesis classes are expected to generalise better than those learned over complex spaces. One also expects the generalisation ability of a function to improve as with the number of examples used during training increases. This is the fundamental idea that underpins machine learning; the more data that you present to the machine, the better it will get a performing the task it has been designed to do.

As humans we have a remarkable capacity for generalisation. For example, we only need to observe a few examples of a dog to be capable of accurately identifying a whole range of dogs from unseen breeds, despite the variety of shapes and sizes that they come in. For a machine to learn to identify a dog, it must first be able to recognise the underlying characteristics of what constitutes a dog, and then learn to discriminate these from other animals such as cats or wolves. This requires the use of a highly expressive hypothesis space and it needs to be trained using large amounts of data to prevent overfitting on the training examples, and allow for generalisation. Fortunately for problems such as image recognition we have access to vast datasets where complex hypothesis spaces can be used without worrying about overfitting. This is the appealing nature of a data-driven learning approach, the more data that we have available, the more we can explore complex hypothesis spaces without worrying about diminishing returns on generalisation. Unfortunately not all datasets have grown at

the same pace, and we must still face the problem of learning how to get the most out of a dataset by trading off the accuracy on training examples with the complexity of the solution. In this thesis we examine two such problems where we must seek to balance these opposing forces.

The first involves binary classification when there is a small number of training examples and/or the input dimension is large. In this setting we have limited information about the class conditional distributions that have generated the training examples. Previous methods that used empirical estimates of the class conditional moments often suffered through inaccurate moment estimates and/or overly confident generalisation guarantees. In this thesis we seek to address these issues by proposing the high-probability minimax probability machine. This approach builds upon the original formulation of the minimax probability machine [Lanckriet et al., 2003], addressing the possible dangers of assuming the empirical and true moments to be equal. We do this by deriving a new optimisation scheme that includes bounds on the deviation of the true moments from their empirical counterparts, and minimises the worst-case future misclassification rate. We see that they act as an intuitive, class specific, regularisation scheme where one penalises moment estimates based upon the relative number of observations seen in each class, and therefore our degree of confidence in their values.

The second involves multi-label learning, where one must learn a function that maps an arbitrary input to a binary output vector. Quite often in this setting there are more possible output configurations than there are training examples, which makes learning to generalise a challenging task. One solution is to take advantage of the relationships between the output variables by encoding their dependencies using a Markov network, and allow the learning algorithm access to this information. However if we have no a priori knowledge of the relationships between output variables then one could argue that it is as reasonable to train each output node individually as it is to train them as if they were completely connected to one another. The former suffers from the inability to leverage the strength of the correlation between output labellings, whereas the latter is crippled by NP-hardness of both learning and making predictions. In this thesis we seek to address these problems by presenting a novel learning method for multi-label learning over Markov networks. Our approach is agnostic to graph structure and simultaneously learns the intrinsic structure of the dataset, whilst finding a large margin predictor on it. The intrinsic graph structure is learned by framing it as a multiple kernel learning problem (MKL), where spanning trees over the complete graph define the joint feature spaces used by the base kernels and the final structure is given by a weighted combination of these spanning trees. Standard MKL methods can be used to update the tree weights and the interesting part is that we show it is possible use a maximum

spanning tree algorithm to efficiently search over the space of all exponentially many kernels, and provide an update direction. Furthermore we present a simple lemma that allows one to check whether exact inference was performed over the inferred graph. This method takes advantage of the representation of the graph as the combination of spanning trees, and only requires inference to be performed on an individual tree basis.

1.1 History

Machine learning represents the intersection of a number of well studied fields including traditional statistics, signal processing, optimisation and theoretical computer science. It is a sub-field of artificial intelligence that arguably has its roots in the realisation that equipping a computer with general intelligence was more difficult than was first conceived. This led to a kind of research that focused on what was possible to solve, and so came about machine learning, a data driven approach that can be tailored to solving a particular problem.

One of the earliest examples of a machine learning approach came during Fisher's 1936 work on linear discriminants [Fisher, 1936] and the classification of iris flowers. This approach used information regarding the observed mean and covariances of the different species of iris. It is still popular today, thanks largely in part to its interpretability and the work done by [Mika et al., 1999] to extend the learning routine into higher dimensional spaces. Another old idea that is still prevalent in the machine learning community is Rosenblatt's 1956 work on the perceptron [Rosenblatt, 1958], which provides the basis of the learning architectures employed by today's deep neural networks. Despite its current popularity, research into the use of the perceptron was largely curbed by the criticisms in [Minsky and Papert, 1988] on its inability to solve linearly inseparable problems such as the XOR problem. This led to a change in research direction and the ensuing *A.I. winter*. It wasn't until 1986 and the re-discovery of the backpropagation algorithm [Williams and Hinton, 1986] that interest resumed, with the authors showing that it was possible for errors to be propagated back through a multilayered neural network and address the shortfalls outlined by [Minsky and Papert, 1988].

At around the same time the perceptron was coming under fire, statistical learning theory [Vapnik and Chervonenkis, 1968], another important part of modern machine learning, was being developed. This approach brought together ideas from statistics and functional analysis to analyse characteristics of a function estimated on a given dataset. The main goal of the theory was to provide a statistical framework for studying how a function learned from a dataset will perform on unseen examples. It wasn't until the early 1990s that this largely theoretical analysis started to be exploited and used in the design of better algorithms for

prediction. The support vector machine [Boser et al., 1992; Cortes and Vapnik, 1995] was one of the first approaches that took advantage of the generalisation properties afforded to large margin predictors. The statistical guarantees offered by these methods, coupled with their ability to efficiently use high dimensional spaces through the representer theorem [Aronszajn, 1950; Wahba, 1990] and exploit the well developed tools of convex optimisation, resulted in a surging interest in this field of research and led to state-of-the-art performance across a wide range of problems.

These data efficient, kernel based methods largely dominated the community, in terms of both research interest and performance, up until the mid 2000s. However with the advent of high performance computing, which allows thousands of processors to run in parallel, and the availability of vast datasets such as *ImageNet* [Deng et al., 2009], it became possible to train large multilayered neural networks with millions of parameters without worrying about overfitting on the training sample. These deep neural networks represent state-of-the-art performance across a wide range of problem domains, and as such their popularity in the community has swelled. We can expect future performance improvements to be driven largely by three forces; technological advances of computing hardware, the increasing availability of large amounts of data, and a greater understanding of theoretical properties of the learning that takes places within the layers.

Artificial intelligence has always captured the imagination of the general public, thanks in part to best-selling books and blockbuster movies but the media has also paid significant attention to many of the actual successes of the research community. The most popular of these successes come in the form of mastering games of skill. Some of the earliest successes included Samuel's checkers program [Samuel, 1959] and Tesauro's temporal difference learner for playing backgammon [Tesauro, 1995]. Then came the 1997 re-match between world-chess champion Gary Kasparov and IBM's supercomputer DeepBlue [Campbell et al., 2002]. This match was marred with controversy with Kasparov claiming his defeat was because of foul play, his suspicions arising from the human like nature of several of DeepBlue's moves. In 2011 IBM made the headlines with another supercomputer, Watson [Ferrucci, 2012], beating the best contestants on the American television game show *Jeopardy!* This required not just perfect knowledge recall but also the ability to understand the hidden clues within the questions that it had to answer. In more recent successes, Google DeepMind showed that it was possible for a single general purpose algorithm to master a number of Atari games [Mnih et al., 2013] using only the information contained within the screen pixels as inputs for learning. They have also recently beaten a former world champion [Silver et al., 2016] at the ancient game of Go, a feat many believed to be out of reach for at least another ten years.

A considerable amount of research has been devoted to training machines to play these games. This helps to engage the wider public in the research efforts of the AI community because of the public's familiarity with these games and their understanding of the amount of skill that is required to master such games. Another appealing feature of using games is that their dynamics are relatively well defined and can be formulated mathematically. This means a simulator can be created, which allows the machine to learn from its own experiences playing the game, similar to how a human would. The major difference being a human's ability to use their intuition and complex planning abilities, versus the computer's ability to simulate and learn from an almost infinite amount of game experience. These successes by no means indicate that we are close to solving general artificial intelligence, however they do represent large steps towards being able to solve even more complex problems in the future.

1.2 Thesis organisation

This thesis can be divided into three main parts. We continue in Chapter 2 with a short review of statistical learning theory to help shed light on the intuition behind a number of the algorithms used throughout this thesis. We explore the notion of empirical and structural risk minimisation, and explore the concept of the complexity of a specific function class. We show how the support vector machine is directly motivated by the theory of statistical learning, and conclude the chapter by explaining how more complex representations can be efficiently incorporated into classical linear learning algorithms through the use of kernel functions.

In Chapter 3 we deviate from the support vector methodology and focus on algorithms that construct predictors using the moments of the underlying class-conditional densities rather than peripheral points. We introduce a novel binary classification algorithm, the high-probability minimax probability machine (HP-MPM), that takes into consideration the deviation of moments from their empirical counterparts, and directly minimises the worst-case future misclassification rate of the predictor. We compare the HP-MPM with traditional approaches on a number of small sample and high noise datasets, with empirical results providing evidence supporting the implicit regularisation scheme that it implements.

Chapter 4 examines the problem of learning over structured output spaces. In particular, we focus on Markov networks with unknown graph structures and develop an algorithm that is capable of simultaneously extracting information regarding the structure of the data and finding a large margin separation between correct and incorrect labellings. This approach builds on advances made in large margin methods that use a set of random spanning trees to approximate the unknown graph structure. Our approach is different in that we guide the

inclusion of new trees. It is similar to multiple kernel learning where we iterate between the addition of new kernels, which are represented by our trees, and solving a restricted SVM over the new combined kernel.

Chapter 2

Machine Learning

In this chapter we provide a short introduction to several of the key ideas that underpin modern machine learning. Through this short introduction we hope to provide the reader with an insight into how some of today's most popular learning algorithms have been constructed in the context of statistical learning theory, and provide an understanding of the intuition behind them. We begin the chapter by introducing much of the basic notation that will be used throughout this thesis and present a short overview of the main ideas behind statistical learning theory. We then discuss the support vector machine, an algorithm motivated by the generalisation guarantees offered by the theory, and conclude the chapter by discussing how kernel based methods allow linear learning algorithms to efficiently use complex, high-dimensional representations.

2.1 Learning framework

Machine learning is an inductive process whereby we are given a set of training examples and we wish to infer some relationship that will hold true in the future. The relationship is characterised by the function that we learn and its generalisation ability measures how well this relationship holds true in the future. There are three classical learning problems; classification, regression and density estimation. The first two are considered supervised in the sense that each training example consists of an input-output tuple, the output being a class label for classification and a real-valued output for regression, and the goal is to learn a function that maps inputs to outputs. Density estimation is considered unsupervised and training examples consist only of inputs. The goal here is to learn how the inputs are related to one another and find a function that models the underlying probability distribution that the examples were drawn from.

The No Free Lunch theorem [Wolpert and Macready, 1997] tells us that in order for a *best* function to exist for these tasks, we must make a number of assumptions regarding the rela-

tionship between the data that we have currently seen, and the data we expect to see in the future. In the statistical learning theory setting we assume that the data, both past and future, are sampled independently according to the same underlying distribution. This means that each new observation brings with it as much possible information about the underlying phenomenon that generates the data. Therefore as more data is presented to the learning algorithm, our ability to model the phenomenon should improve. This is the underlying principle of the data-driven approach taken by machine learning; more data means more information, which makes better predictions possible.

We now begin to formalise the learning setup and start by introducing some notation that will be used throughout the thesis. Let $\mathcal{X}$ denote the input space and $\mathcal{Y}$ the output space. We assume that the input space $\mathcal{X}$ is arbitrary and the nature of the output space $\mathcal{Y}$ depends on the problem. The focus of this chapter is on binary classification and as such we will assume that $\mathcal{Y} \in \{-1, +1\}$ for the remainder and make the further assumption that the input space $\mathcal{X}$ is Euclidean. The training data is denoted by the set $\mathcal{S} = \{(x_k, y_k) \in \mathcal{X} \times \mathcal{Y} \mid k = 1, \ldots, m\}$, and we use this to learn a function $f : \mathcal{X} \to \mathcal{Y}$ mapping input observations to class membership. We consider a probabilistic framework, where we assume that each example (x_k, y_k) is sampled independently from some fixed but unknown probability distribution $P(X, Y)$. Our goal is to model the probability of an observation $x \in \mathcal{X}$ belonging to a specific class $y \in \mathcal{Y}$, therefore we are interested in the distribution $P(Y|X)$. This can be obtained through Bayes law where

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} = \frac{P(X, Y)}{P(X)} .$$

In the case of binary classification, if we are able to compute $P(Y|X)$ then for each $x \in \mathcal{X}$ the optimal prediction function $f^*(x)$ would assign the labelling $y \in \mathcal{Y}$ with maximum a posterior probability i.e.

$$f^*(x) = \underset{y \in \mathcal{Y}}{\mathbf{argmax}}\, P(Y = y|x) . \tag{2.1}$$

This is known as Bayes rule and we now discuss how to formalise this in terms of loss functions and expected risk. We define the risk of a function f as

$$R(f) := P(f(x) \neq Y) = \mathbb{E}\left[\ell(f(X), Y\right] , \tag{2.2}$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to \{0, 1\}$ is the loss function and the expectation is taken with respect to the probability distribution $P(X, Y)$. For binary classifiers a natural choice of loss function is given by the indicator $\mathbb{1}\{f(x) \neq y\}$, which returns one if the function $f(x)$ predicts incorrectly, and zero otherwise. If we plug this loss function into expression 2.2 we see that

the function given in 2.1 corresponds to the one that minimises its values. Unfortunately we do not have access to the joint probability distribution $P(X, Y)$ and we must use other means to minimise the expected risk.

Empirical risk minimisation

Given our limited amount of training data $\mathcal{S}$, we want to be able to find a function f from the space of possible functions $\mathcal{F}$ so that we come as close as possible to minimising the expected risk. A direct approach would be to examine the risk of a function on the training sample $\mathcal{S}$ and use this information to construct the predictor. To do this we replace the expected risk $R(f)$ over $P(X, Y)$ with the empirical risk $\hat{R}(f)$ over the training sample $\mathcal{S}$, where

$$\hat{R}(f) := \frac{1}{m} \sum_{k=1}^{m} \ell(f(x_k), y_k) \,. \tag{2.3}$$

The goal of the learning algorithm is to search over the space of all possible functions $\mathcal{F}$ and return the function $f \in \mathcal{F}$ that minimises the empirical risk. This approach is known as empirical risk minimisation (ERM) and plays a crucial role in much of statistical learning theory (SLT).

One can encounter a number of problems when implementing this inductive process. For example, if the space of all possible functions is infinite then it is possible that there are infinitely many functions $f \in \mathcal{F}$ capable of obtaining zero empirical risk. The difficultly now amounts to choosing which of these candidate functions is the best. Another potential pitfall arises when we work with datasets where the dimensionality of the data is large compared to the number of training samples. In this setting it is relatively easy to find a function that obtains low empirical risk, however the performance of the function may fail to generalise to unseen examples. This is known as over-fitting and is a result of the hypothesis space $\mathcal{F}$ being overly complex for the training data that it was trained using. The opposite can also occur where the best classifier from the hypothesis space results in a large number of errors on the training data, indicating that the hypothesis is not complex enough for the training data that we have seen. This is known as under-fitting, and later we will discuss how it, and overfitting, can be handled using regularisation methods and/or following a structural risk minimisation (SRM) scheme. For now we discuss the necessary and sufficient conditions that a learning machine must possess in order to be sure that the minimisation of the empirical risk can lead to small values of actual risk. This is known as the consistency of the learning process, and it can be stated more formally by ensuring that the following

expression converges in probability

$$\left|R(f_m) - \hat{R}(f_m)\right| \to 0 \;\text{ as }\; m \to \infty \tag{2.4}$$

where $f_m \in \mathcal{F}$ is the function chosen by the learning machine on $\mathcal{S}$. We know by the law of large numbers that for any fixed function $f \in \mathcal{F}$, the empirical risk will converge to the expected risk. However for ERM, the function f_m is dependent on the training data $\mathcal{S}$, and we need to be sure that the function returned by the learning process satisfies the expression above. The following theorem states what is required for the ERM to be consistent:

Theorem 1 *[Vapnik and Chervonenkis, 1991]. One-sided uniform convergence in probability i.e.*

$$\lim_{m\to\infty} P\left[\sup_{f\in\mathcal{F}}\left(R(f) - \hat{R}(f)\right) > \epsilon\right] = 0 \;\; \forall\epsilon > 0$$

is a necessary and sufficient condition for the consistency of ERM.

This theorem states that in order for a learning process to be consistent the empirical risk of the *worst* function in $\mathcal{F}$ must converge in probability to its expected risk. Therefore assuming that consistency is a desired feature of the learning process, any analysis of the ERM principle should be conducted in a worst-case setting.

Suppose we assume consistency for a given function class $\mathcal{F}$, we know that as the number of samples tends to infinity, ERM will converge upon the function $f_m \in \mathcal{F}$ that minimises the expected risk. However in reality we only ever have a finite amount of data to learn from, and therefore we would like to understand the rate of convergence for a particular class of functions $\mathcal{F}$. Intuitively this depends on the complexity of the function class used during the learning, where complexity can be defined in a number of different ways e.g. the covering number, Rademacher complexity or the VC dimension. Each of these methods basically measure the number of possible functions that the class $\mathcal{F}$ can generate, and thus its capacity to fit the data. Given that this is a light introduction to SLT, we will only touch briefly on the VC (Vapnik-Chervonenkis) dimension h of the function class $\mathcal{F}$ that f is chosen from. This measures the maximum number h of points that can be shattered in all possible 2^h configurations using functions $f \in \mathcal{F}$. For example, if we consider binary classification for linear functions with n-dimensional inputs, the maximum number of points that can be shattered is $n + 1$.

We want to form bounds on the probability that the empirical risk of any function on sample

$\mathcal{S}$ will not deviate by much from the expected risk on the function i.e.

$$P\left[\sup_{f\in\mathcal{F}}(R(f)-\hat{R}(f,\mathcal{S}))>\epsilon\right]<H(\mathcal{F},m,\epsilon)\,,$$

where H is some function that depends on the number of training samples m, the maximum deviation of empirical and expected risks ϵ and properties of the function class $\mathcal{F}$, namely the VC dimension h. By fixing the value of $H(\mathcal{F},m,\epsilon)=\delta$, and solving with respect to ϵ, we can make statements regarding the deviation of the expected and empirical risk that hold true for high probability i.e. with probability at least $1-\delta$ over the random draw of the training sample $\mathcal{S}$

$$R(f)\leq\hat{R}(f,\mathcal{S})+\tilde{H}(\mathcal{F},m,\delta)\,, \tag{2.5}$$

where $\tilde{H}$ measures the uncertainty in the estimate of the expected risk of the function f. Intuitively the value of $\tilde{H}$ decreases as the number of samples m increases, and increases as the complexity of the function class or the desired level of precision increases. Note that this bound (2.5) holds true for each function $f\in\mathcal{F}$, not just the one that minimises the empirical risk.

Structural risk minimisation

These bounds are the principle behind the learning setting known as structural risk minimisation (SRM) [Vapnik and Chervonenkis, 1974]. In this setting one has access to a nested subset of function classes

$$\mathcal{F}_1\subset\mathcal{F}_2\subset\cdots\subset\mathcal{F}_n\ldots$$

such that the VC dimension h_k of each subset S_k satisfies

$$h_1\leq h_2\leq\cdots\leq h_n\ldots\,.$$

For each subset of functions $\mathcal{F}_k$ ERM is performed and the learning algorithm then selects the function that minimises the bound on the expected risk i.e.

$$f_m=\underset{k}{\textbf{argmin}}\left(\underset{f\in\mathcal{F}_k}{\textbf{argmin}}\;\hat{R}(f)+\tilde{H}(\mathcal{F}_k,m,\delta)\right)$$

We mentioned earlier the concept of over/under-fitting and we can see that the SRM trades off the fit on the data, given by the empirical risk $\hat{R}$, with the complexity of the function represented by the penalty term $\tilde{H}$. As the subset index n increases, the empirical risk

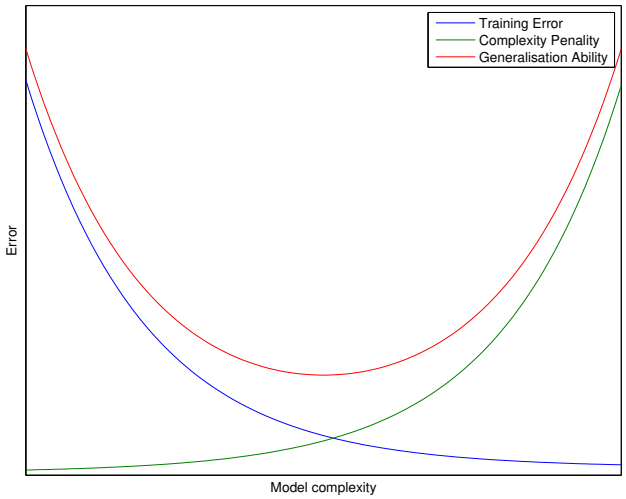


Fig. 2.1 The trade off between over and under fitting. We see that as the model complexity increases it is capable of better performance on the training sample. However this comes with an additional complexity penalty in its generalisation ability. In SRM we look for the sweet spot between complexity and training error represented by the minimum of the generalisation term.

will decrease since all functions $f \in \mathcal{F}_{i-1}$ are also in $\mathcal{F}_i$. However the value of the penalty term in the bound will increase as we use a more complex function class. When there is a large amount of data relative to the VC-dimension i.e. $m/h > 20$, then one can expect that the minimisation of the empirical risk is what dominates the determination of the solution. However if m/h is small then the relationship between the complexity of the functions across subsets plays a larger role in determining what function the SRM selects. One of the most famous bounds is given by Vapnik and Chervenonkis, and is the basis behind the formulation of the support vector machine that we discuss later in this chapter

Theorem 2 *[Vapnik and Chervonenkis, 1974] Let h denote the VC-dimension of the function class $\mathcal{F}$, $\mathcal{S}$ a training sample of m examples and let $\hat{R}$ define the empirical risk given in 2.3. For all $\delta > 0$ and $f \in \mathcal{F}$, the following bound regarding the expected risk of f holds true*

$$R(f) \leq \hat{R}(f, \mathcal{S}) + \sqrt{\frac{h\left(\log \frac{2m}{h} + 1\right) - \log(\delta/4)}{m}}, \tag{2.6}$$

with probability at least $1 - \delta$ for $m > h$ over the random draw of the training sample $\mathcal{S}$.

SRM provides a principled way to select the function that minimises expected risk, however it can be difficult to implement in practice. For example we need to have access to a nested subset of function classes of known VC-dimension. This can be done rather easily

for functions that are linear in parameters and a nested sequence can be obtained by simply increasing the polynomial degree of the function class, however it is not so straightforward when we consider non-linear approximating functions.

Regularisation

A perhaps more straightforward approach uses a large function class $\mathcal{F}$ and directly penalises the complexity of $f \in \mathcal{F}$ using a penalisation (regularisation) function $\Omega : \mathcal{F} \to \mathbb{R}$. This results in the minimisation of the regularised risk functional given by

$$f_m = \min_{f \in \mathcal{F}} \hat{R}(f) + \lambda \Omega(||f||)\,, \tag{2.7}$$

where $\lambda \in \mathbb{R}$ is the regularisation parameter that controls the trade off between the complexity of the solution and the fit of the data. The regularisation function is a non-negative, non-decreasing function that takes as input the norm of the function f . Similar to SRM, one has to solve expression 2.7 several times for different values of λ, and often one has to withhold another part of the training data to evaluate the quality of the solution obtained from minimising the regularisation risk functional. One popular procedure for this is known as cross-validation, where the training data is split in to K disjoint subsets of $\mathcal{S}$, a model is then trained using the data in $\mathcal{S} \setminus \mathcal{S}_k$ and evaluated on $\mathcal{S}_k$. The best choice of regularisation, λ, is given by the value that performs the best when averaged across all K subsets $\mathcal{S}_k$, and the final function is learned using the full training sample with the best performing value of λ.

2.2 From theory to practice: the support vector machine

The bounds found by SRM methods are often very loose, and most modern machine learning methods apply the notion of regularisation coupled with cross-validation in order to find the function that best approximates the one with lowest expected risk. We now present the support vector machine (SVM), a state of the art classification algorithm, whose regularisation scheme mimics the SRM bounds.

Suppose the dataset $\mathcal{S} = \{(x_k, y_k)\}_{k=1}^m$, where $x_k \in \mathbb{R}^m$ and $y_k \in \{-1, +1\}$, can be perfectly separated by the hyperplane $\mathcal{H}(w, b) := \{x \in \mathcal{X} \,|\, \langle w, x\rangle = b\}$. This means that the following relationship holds true

$$y_k \left(\langle w, x_k\rangle - b\right) > 0 \;\; \forall \;\; k = 1, \ldots, m\,. \tag{2.8}$$

There may be many such hyperplanes where the relationship 2.8 holds true and the empirical risk on $\mathcal{S}$ is zero, however what we are looking for is the optimal hyperplane. Given that we

are considering the space of linear functions $\mathcal{F} := \{f : \mathcal{X} \to \mathbb{R} \,|\, f(x) = \langle w, x\rangle - b\}$, each of these hyperplanes has a VC-dimension of $n+1$, and therefore the SRM principle is unable to distinguish between separating hyperplanes using this interpretation of complexity. Before we describe the complexity measure considered by the SVM we introduce the concept of a margin and discuss the uniqueness of hyperplanes. Assuming the condition 2.8 holds true, the margin of a hyperplane $\mathcal{H}(w, b)$ on $\mathcal{S}$ is given by the smallest distance of a datapoint (x_k, y_k) to the hyperplane i.e.

$$\Gamma(w, b, \mathcal{S}) := \min_{k=1,\ldots,m} \frac{y_k \left(\langle w, x_k\rangle - b\right)}{||w||} \,.$$

If a dataset can be separated by some hyperplane defined by (w, b), then it can also be separated by some scalar multiple of these quantities. Therefore there are infinitely many separating hyperplanes and we must use a method to ensure that each hyperplane is defined uniquely. To do this we use the canonical hyperplane approach, where each separating hyperplane satisfies the following,

$$y_k \left(\langle w, x_k\rangle - b\right) \geq 1 \;\forall\; k = 1, \ldots, m \,.$$

The SVM considers the complexity of a function in terms of its norm $||w||$, where a larger norm is considered to be a more complex function. Therefore the goal is to find the minimum norm hyperplane that separates the data. From a SRM perspective, we can can view this in terms of a nesting scheme of hypothesis classes of increasing norm i.e.

$$\mathcal{F}_n = \{f : \mathcal{X} \to \mathbb{R} \,|\, f(x) = \langle w, x\rangle - b, ||w|| \leq \Lambda_n\} \,,$$

where $\Lambda_1 \leq \cdots \leq \Lambda_n \leq \ldots$. It seems reasonable to assume that when $\Lambda_k < \Lambda_{k+1}$, the complexity of the function classes satisfy $h(\mathcal{F}_{\Lambda_k}) \leq h(\mathcal{F}_{\Lambda_{k+1}})$. Theoretical justification of this nesting sequence is provided in the following theorem regarding the VC-dimension of a canonical hyperplane with γ-margin where $\gamma = \Gamma(w, b, \mathcal{S}) = \frac{1}{||w||}$.

Theorem 3 *Let $x \in \mathcal{X}$ be bounded by a sphere of radius R i.e. $||x|| \leq R$. Then the set of γ-margin separating hyperplanes has VC-dimension bounded by the inequality*

$$h \leq \min\left(\frac{R^2}{\gamma^2},\; n\right) + 1 \tag{2.9}$$

In the canonical hyperplane setting, finding the minimum norm hyperplane is equivalent to finding the one that maximises the margin on $\mathcal{S}$. Geometrically the intuition behind this approach is that by maximising the distance between any two examples of different classes, we minimise the chances new unseen examples will fall on the wrong side of the hyperplane.

The SVM optimisation problem is given by

$$\min_{w,b} \frac{1}{2}||w||^2 \text{ s.t. } y_k\left(\langle w, x_k\rangle - b\right) \geq 1, \;\; k = 1, \ldots, m\,. \tag{2.10}$$

This is referred to as the primal problem formulation, and can be solved using a standard quadratic optimiser. However we will see later that it is most commonly solved using the dual version of the problem. The dual is constructed by introducing Lagrange multipliers (dual variables) $\alpha_k \geq 0$ for the margin constraints on each example $k = 1, \ldots, m$. This results in the Lagrangian

$$L(w, b, \alpha) := \frac{1}{2}||w||^2 - \sum_k \alpha_k \left(y_k\left(\langle w, x_i\rangle - b\right) - 1\right) \tag{2.11}$$

Solving for the primal variables w and b we see that at optimality

$$\frac{\partial L}{\partial w} = 0 \quad \text{and} \quad \frac{\partial L}{\partial b} = 0\,,$$

which results in the following properties for the optimal hyperplane

- Each dual variable is non-negative and there is an even distribution of weight placed upon the observations from each class

$$\sum_k \alpha_k y_k = 0\,, \;\; \alpha_k \geq 0, \;\; k = 1, \ldots, m\,. \tag{2.12}$$

- The optimal hyperplane is represented using a linear combination of the input vectors x_k, weighted according to dual variable α_k and take the sign of the class y_k,

$$w = \sum_k \alpha_k y_k x_k, \;\; \alpha_k \geq 0, \;\; k = 1, \ldots, m \tag{2.13}$$

- The Karush-Kuhn-Tucker (KKT) conditions

$$\alpha_k \left(y_i(\langle w, x_i\rangle - b) - 1\right) = 0, \;\; k = 1, \ldots, m\,, \tag{2.14}$$

  state that only those examples that appear on the margin have non-zero weight. This is where the term *support vector* comes from since it is only these examples that support the hyperplane and determine the solution.

By plugging these values into the Lagrangian, and taking into consideration the KKT conditions, we obtain the functional

$$L(\alpha) = \sum_k \alpha_k - \frac{1}{2}\sum_k \sum_j \alpha_k \alpha_j y_i y_k \langle x_k, x_j\rangle\,, \tag{2.15}$$

and the goal of the SVM is maximise this functional subject to the constraints on the dual variables given in 2.12. This results in the quadratic program given by

$$\begin{aligned} &\max_{\alpha} \sum_k \alpha_k - \frac{1}{2} \sum_k \sum_j \alpha_k \alpha_j y_i y_k \langle x_k, x_j \rangle \\ &\text{s.t.} \sum_k \alpha_k y_k = 0,\ \alpha_k \geq 0,\ k = 1, \ldots, m\,. \end{aligned} \tag{2.16}$$

The SVM makes a prediction y^* for new example $x^* \in \mathcal{X}$ according to

$$y(x^*) = \begin{cases} 1 & \text{if } \langle w, x^* \rangle - b \geq 0 \\ 0 & \text{otherwise} \end{cases}, \tag{2.17}$$

where value of the *bias* term b is found by selecting a support vector from each class x_+ and x_-, and using the KKT conditions 2.14 to get

$$b = \frac{1}{2} \left(\langle w, x_+ \rangle + \langle w, x_- \rangle \right)\,. \tag{2.18}$$

So far we have only discussed the case where the data is assumed to be perfectly separable, however the modern day version of the SVM [Cortes and Vapnik, 1995] deals with the case of inseparable data through the introduction of slack variables. These slack variables allow violations of the margin condition and the optimal solution often contains a number of data points lying within the margin of the hyperplane, and even some falling on the wrong side of it. The primal optimisation is now given by

$$\min_{w,b,\xi} \ \frac{1}{2}||w||^2 + C \sum_k \xi_k \tag{2.19}$$

$$\text{s.t.}\ \ y_k \left(\langle w, x_k \rangle - b \right) \geq 1 - \xi_k,\ \xi_k \geq 0,\ k = 1, \ldots, m\,, \tag{2.20}$$

where ξ_k are the slack variables and C is the regularisation parameter that trades off training data accuracy with the complexity of the solution. Larger values of C are likely to be associated with fewer errors on the training dataset, however this comes at the cost of a smaller margin on the dataset and a tendency for the function to overfit on the training data.

The dual formulation takes a very similar form to before and is given by

$$\begin{aligned} &\max_{\alpha} \sum_k \alpha_k - \frac{1}{2} \sum_k \sum_j \alpha_k \alpha_j y_k y_j \langle x_k, x_j \rangle \\ &\text{s.t.} \ \sum_k \alpha_k y_k = 0\,,\ 0 \leq \alpha_k \leq C,\ \ k = 1, \ldots, m \end{aligned} \tag{2.21}$$

From the KKT and complementary slackness conditions, we once again expect the solution

to consist of a sparse set of non-zero dual variables where

- if $\alpha_k = 0$ then $y_k\left(\langle w, x_k\rangle - b\right) \geq 1$ and $\xi_k = 0$,
- if $\alpha_k = C$ then $y_k\left(\langle w, x_k\rangle - b\right) < 1$ and $\xi_k \geq 0$,
- if $\alpha \in (0, C)$ then $y_k\left(\langle w, x_k\rangle - b\right) = 1$ and $\xi_k = 0$,

This implementation overcame the requirement of separable data imposed by the original formulation, and was shown to have strong generalisation guarantees on real world datasets. It was noted in [Cortes and Vapnik, 1995] that it often exhibited better out-of-sample performance than the *hard-margin* approach even on separable data. This was due to the added flexibility of the *soft-margin* that allowed a larger margin to be obtained on the training data. The large margin principle behind the SVM has also been successfully applied to the problem of regression [Drucker et al., 1997], density estimation [Schölkopf et al., 2001] and structured prediction [Tsochantaridis et al., 2004], and later in Chapter 4 we formulate our multilabel learning over an unknown graph structure using this principle.

2.3 Kernel methods

Kernel methods typically refer to a two-step approach for solving machine learning problems. The first step involves the mapping the input vectors to a high-dimensional *feature* space, and the second implements a learning algorithm to find a linear solution in that space. The benefit of this approach stems from the use of the *kernel trick* [Aizerman et al., 1964; Boser et al., 1992; Vapnik, 1998]. If the learning algorithm can be formulated so that the feature mappings appear only as inner products then these high-dimensional spaces can be used without having to explicitly compute them. The inner product between feature mappings is known as the *kernel function* and provided this can be computed efficiently, the cost of using these high dimensional spaces is only polynomially with respect to the size of the dataset. We now outline the key components of the kernel methods. A feature mapping

$$\phi : \mathcal{X} \to \mathcal{H}\,, \tag{2.22}$$

takes the original input observations $x \in \mathcal{X}$ and maps them to some feature space $\mathcal{H}$, which is typically of a much larger dimension than the original space. The kernel function is given by the inner product of feature mappings

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \quad \text{with} \quad k(x, x') = \langle \phi(x), \phi(x')\rangle\,,$$

where $\langle \cdot, \cdot \rangle$ corresponds to the inner product in the feature space $\mathcal{H}$. For classification purposes, the hope is that in this new feature space observations belonging to different

classes can be better separated, therefore making the task of classification easier. To help motivate this we refer back to the XOR problem, which represented a significant obstacle during the early days of neural networks. We consider the two dimensional version of the problem where $x = (x_1, x_2) \in \mathbb{R}^2$ and the class membership function is given by

$$y(x) = \begin{cases} 1 & \text{if } \operatorname{sign}(x_1) \neq \operatorname{sign}(x_2) \\ -1 & \text{otherwise} \end{cases}$$

It is well known that the XOR dataset is not separable in the original linear space $\mathcal{X} = \mathbb{R}^2$, however we see that by implementing the feature mapping

$$\phi : \mathbb{R}^2 \to \mathbb{R}^3 \quad \text{where} \quad (x_1, x_2) \to (x_1^2, x_2^2, \sqrt{2}x_1x_2)\,, \tag{2.23}$$

the data becomes perfectly separable in the new space. This is shown in Figure 2.3 , where we observe that the hyperplane given by $w_\phi = (0, 0, -1)$ and $b = 0$ is capable of perfectly separating the dataset. This simple example highlights the benefits of working with feature mappings, however given its small dimension the benefits of using kernel methods for this problem are not overly clear.

The most attractive features of kernel based methods are their flexibility and efficiency. In terms of their flexibility we will present a theorem that states that the kernel function k can be chosen arbitrarily, provided there is a guarantee on the existence of a corresponding feature mapping. This allows us to implicitly use feature spaces that we may not even know how to explicitly compute, provided the kernel function satisfies certain properties.

Definition 1 *Positive definite kernel*
A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a positive definite kernel iff it is symmetric i.e. $k(x, x') = k(x', x)$ for any two $x, x' \in \mathcal{X}$, and is positive definite i.e.

$$\sum_{k=1}^{m}\sum_{j=1}^{m} a_k a_j k(x_k, x_j) \geq 0\,,$$

for any sample of $m > 0$ objects $x_1, \ldots, x_m \in \mathcal{X}$, and any choice of real numbers $a_1, \ldots, a_m \in \mathbb{R}$.

Before we present the theorem, we first mention that the original input observations are mapped to a Hilbert space. This is simply a complete inner-product space, which generalises the notion of a Euclidean space, and the operations that one can perform on it to spaces with finite and even infinite dimensions.

Theorem 4 *For any positive definite kernel k on space $\mathcal{X}$, there exists a Hilbert space $\mathcal{H}$*

and a mapping $\phi : \mathcal{X} \to \mathcal{H}$ *such that*

$$k(x, x') = \langle \phi(x), \phi(x') \rangle, \quad \textit{for any} \quad x, x' \in \mathcal{X}\,.$$

This allows us to think of the kernel function as inducing a feature mapping. The dimension of these mappings are normally very large, which makes explicitly working with them intractable. Fortunately for many popular kernels there are efficient methods for computing the inner product that avoids having to ever explicitly compute the mapping $\phi(x)$. Consider for example the kernel used during the XOR problem. This feature map contains all monomials of degree two and its inner product can be computed using

$$\begin{aligned} k(x, \tilde{x}) = \langle \phi(x), \phi(\tilde{x}) \rangle &= \left\langle (x_1^2, x_2^2, \sqrt{2}x_1x_2), (\tilde{x}_1^2, \tilde{x}_2^2, \sqrt{2}\tilde{x}_1\tilde{x}_2) \right\rangle && (2.24) \\ &= x_1^2\tilde{x}_1^2 + 2x_1x_2\tilde{x}_1\tilde{x}_2 + x_2^2\tilde{x}_2^2 && (2.25) \\ &= \langle x, \tilde{x} \rangle^2\,. && (2.26) \end{aligned}$$

The more general form of this kernel is known as the polynomial kernel and its feature mapping includes all monomials up to the degree $d > 0$

$$k(x, x') = (\langle x, x' \rangle + c)^d, \quad \text{where} \quad c \geq 0\,. \tag{2.27}$$

For input data of dimension n, the dimension of the feature space N grows at a rate of $\binom{n+d}{d}$ making it computationally challenging to work with $\phi(x)$ explicitly for large values of n and d. On the other hand, the kernel function only requires n operations to compute the inner product in the original space, which is followed by the raising of this value to a given power. This is considerably more efficient for large n and d. Note that when using polynomial kernels we have to be mindful of the fact that as d increases, the absolute value of the inner products have a tendency to drift towards either zero or infinity as d increases.

Another popular choice of kernel is the Gaussian kernel,

$$k(x, x') = \exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right)\,. \tag{2.28}$$

where $\sigma > 0$ is referred to as the smoothness parameters of the kernel. Kernels are often thought of as similarity measures and the Gaussian kernel measures similarity in terms of the Euclidean distance between points in $\mathcal{X}$. Points $x, x' \in \mathcal{X}$ that are far from one another have a small value of $k(x, x')$. However as the value of σ increases the effective *distance* between points decreases and this is captured by a larger value of $k(x, x')$. This is where the interpretation of smoothness comes from, the larger the value of σ the greater the distance required for observations to be thought of as dissimilar. The feature map corresponding to

this kernel is actually infinite dimensional, however through the kernel trick we can use it efficiently.

We return to the SVM algorithm and focus on the dual formulation given in 2.21. We can now replace the inner products $\langle x, x' \rangle$ with an arbitrary kernel $k(x, x')$ and the optimisation becomes

$$\begin{aligned} &\max_{\alpha} \sum_k \alpha_k - \frac{1}{2} \sum_k \sum_j \alpha_k \alpha_j y_k y_j k(x_k, x_j) \\ &\text{s.t. } \sum_k \alpha_k y_k = 0\,,\ 0 \leq \alpha_k \leq C,\ \ k = 1, \ldots, m\,. \end{aligned} \tag{2.29}$$

The kernel function represents the inner product of the feature mappings $\phi(x)$ and the prediction function is given by

$$y^*(x) = \begin{cases} 1 & \text{if } \ \langle w, \phi(x) \rangle = \sum_k \alpha_k y_k k(x_k, x) \geq b \\ -1 & \text{otherwise} \end{cases}\,. \tag{2.30}$$

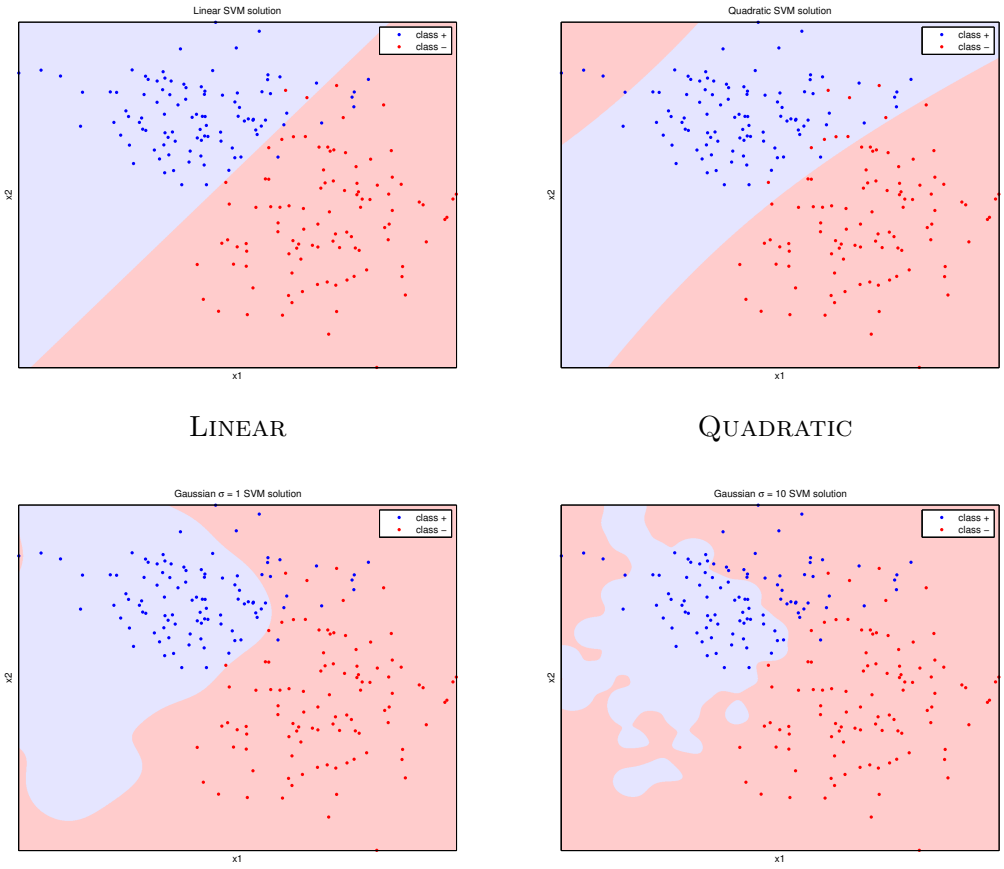
In Figures 2.2 and 2.3 we illustrate the nature of the solutions derived from different kernels. Recall that linear patterns are found in the feature space, however these translate to non-linear relationships in the original space, as shown in Figure 2.2. When it comes to choosing a kernel for a particular task, prior knowledge of the dataset one is learning over is key to selecting an appropriate one. For example, on the linear dataset the function found by the Gaussian kernel with small σ is unnecessarily complex for the problem at hand, and would be prone to overfitting. On the other hand, on the circle dataset, any function found using the linear kernel will struggle to get an accuracy of more than 50%.

We conclude our introduction to kernels by outlining a simple but powerful theorem that ties together notions of regularisation and kernels. To do this we view things from a reproducing kernel Hilbert space point of view. We think of the feature space $\mathcal{H}$ as a Hilbert space of functions $f \in \mathcal{H}$ coupled with a reproducing kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that

$$\phi(x) = k(\cdot, x) \ \ \text{and} \ \ \forall\, x \in \mathcal{X},\ \forall\, f \in \mathcal{H},\ \langle f, k(\cdot, x) \rangle = f(x)\,. \tag{2.31}$$

Theorem 5 *Representer theorem [Wahba, 1990]*
Suppose we have a dataset $\mathcal{S} = \{x_1, \ldots, x_m\}$ and the problem is to minimise the regularised risk functional given by

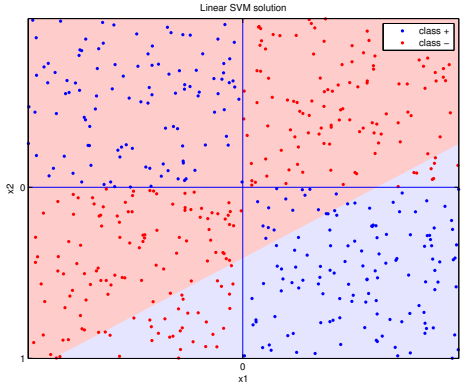
$$\min_{f \in \mathcal{H}} \ \hat{R}(f) + \Omega(||f||^2)\,, \tag{2.32}$$



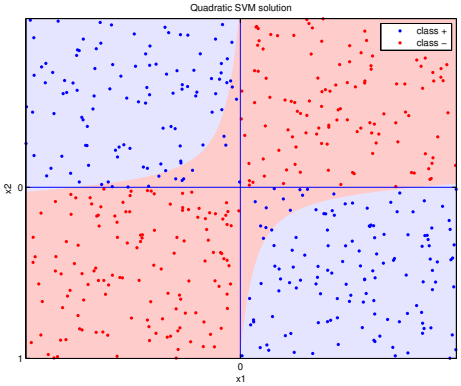
LINEAR QUADRATIC

GAUSSIAN $\sigma = 1$ GAUSSIAN $\sigma = 0.1$

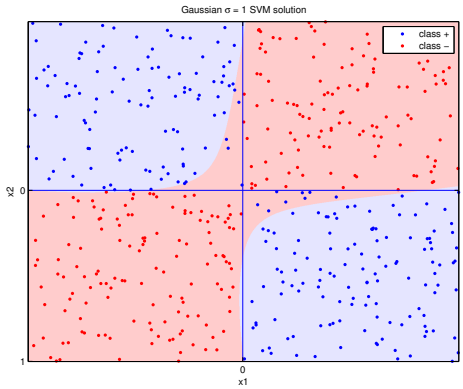
Fig. 2.2 Decision boundaries found on a training set generated by two multivariate normal distributions. The optimal decision boundary is given by a linear hyperplane and we see that the linear representation finds a good decision boundary. The decision boundary of the quadratic function performs well on the bottom right quadrant, however the quadratic nature of the solution means that it will incorrectly classify points in the top left quadrant. The Gaussian kernels find irregular decision boundaries with the one with small width finding an particularly over complicated pattern for the problem at hand.



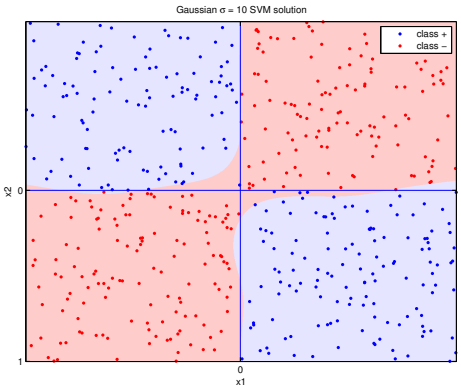
LINEAR



QUADRATIC

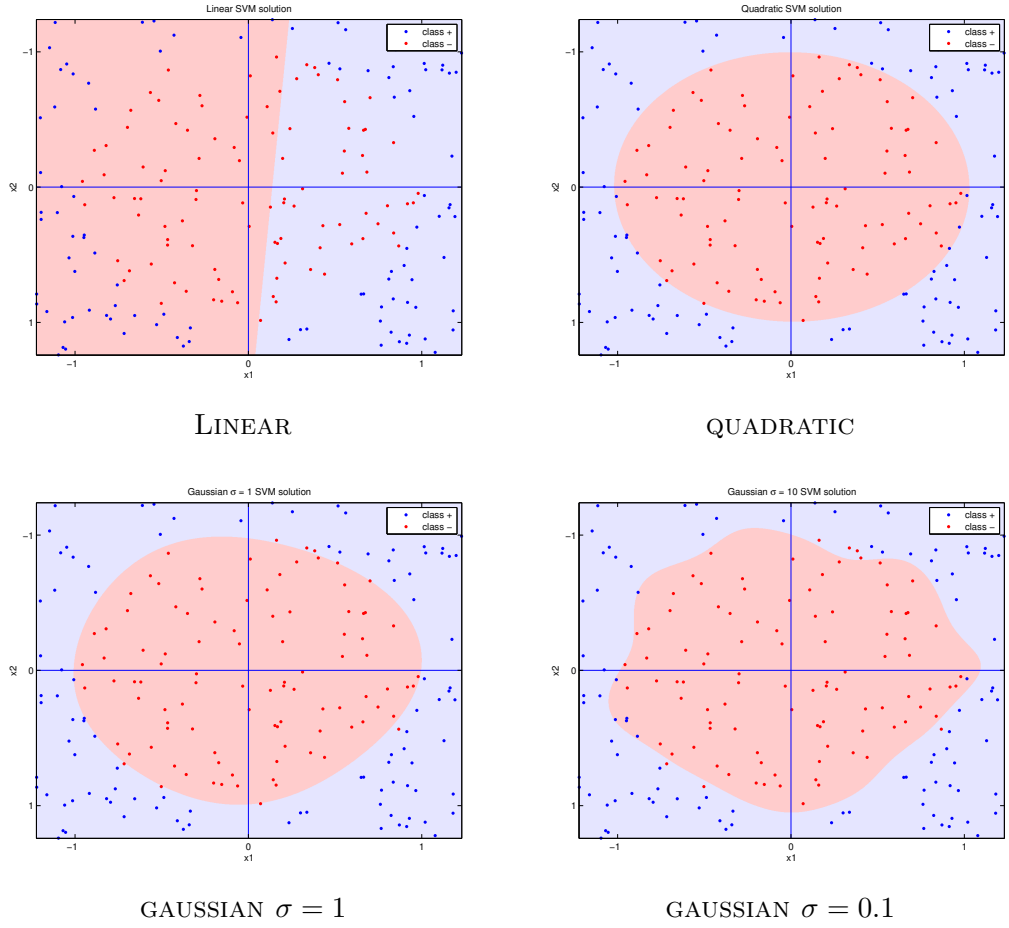


GAUSSIAN $\sigma = 1$



GAUSSIAN $\sigma = 0.1$

Fig. 2.3 Decision boundaries found on a training set generated by XOR dataset, where $y_k = -1$ if $sign(x_{k,1}) = sign(x_{k,2})$ and $y_k = 1$ otherwise. The linear representation places along the diagonal and fails to perform better than randomly, whereas the quadratic representation finds a good decision boundary due to the product between the different dimensions. The Gaussian representations find good decision boundaries, with the smaller width kernel being capable of better approximating the optimal boundaries determined by the x_1 and x_2 axes.



LINEAR QUADRATIC

GAUSSIAN $\sigma = 1$ GAUSSIAN $\sigma = 0.1$

Fig. 2.4 Decision boundaries found on a training set generated by circle dataset. If $x_{k,1}^2 + x_{k,2}^2 \leq 1$ then $y_k = -1$, otherwise $y_k = 1$. The linear representation is unable to find a separating hyperplane that is better than random guessing and the quadratic representation finds a smooth circular decision boundary that almost perfectly separates the data. Both Gaussian representations perform well finding a circle like pattern for classification, however we see the smaller width Gaussian has a slightly less smooth decision boundary.

where $\Omega : \mathbb{R} \to \mathbb{R}$ is the regularisation term, which takes as input the norm of the function f and is a strictly increasing monotonic function. The solution $f^* \in \mathcal{H}$ to this problem is given by

$$f^* = \sum_{k=1}^{m} \alpha_k \phi(x_k) = \sum_{k=1}^{m} \alpha_k k(x_k, \cdot)$$

and

$$\forall\, x \in \mathcal{X}. \quad f^*(x) = \sum_{k=1}^{m} \alpha_k k(x_k, x) \tag{2.33}$$

Proof *Let the function be comprised of two components $f = f_{\mathcal{S}} + f_{\perp}$, where $f_{\mathcal{S}} \in \mathcal{F}_{\mathcal{S}}$ represents the projection of f onto the space spanned by the dataset $\mathcal{S}$, and $f_{\perp}$ is orthogonal to the space $\mathcal{F}_{\mathcal{S}}$ i.e. $f_{\perp} \perp \mathcal{F}_{\mathcal{S}}$. An evaluation of the function is given by*

$$f(x_k) = \langle f, k(x_i, \cdot)\rangle = \langle f_{\mathcal{S}}, k(x_k, \cdot\rangle) + \langle f_{\perp}, k(x_k, \cdot)\rangle = \langle f_{\mathcal{S}}, k(x_k, \cdot)\rangle\,,$$

and we see that $f_{\perp}$ has no influence on the evaluation of the function on training samples $\mathcal{S}$. However, there is a non-negative contribution by $f_{\perp}$ to the norm of the function

$$||f||^2 = ||f_{\mathcal{S}}||^2 + ||f_{\perp}||^2 \geq ||f_{\mathcal{S}}||^2$$

Therefore $f_{\perp}$ can have no influence on reducing the empirical risk $\hat{R}$, however it does contribute positively to the penalisation term Ω, and it should therefore not contribute be part of the optimal solution. ■

Earlier we mentioned that regularisation encourages a *smooth* solution but we can now also see that it has considerable computational advantages. This theorem tells us that the solution we are looking for resides in the space spanned by the original dataset, which makes the optimisation problem much easier and it can dramatically reduce the space over which we have to search for a solution. Furthermore this supports and explains the representation of the SVM solution found through the Lagrangian expression 2.13.

2.4 Summary

In this chapter we provided a short introduction to several of the key ideas that underpin much of modern machine learning methods. The goal of classification is to find the function that minimises the expected rate of misclassification, however without knowledge of the joint distribution over inputs and outputs, we must somehow make do with an approximation of

this. Empirical risk minimisation was presented as a natural alternative to the minimisation of expected risk, however we saw that it often suffered from problems calibration with respect to over and under fitting. Structural risk minimisation sought to overcome the calibration issues by taking into consideration the complexity of the function class and sample size by minimising high-probability bounds of the expected risk. These bounds are often so weak that they became invalid, and in practice most modern methods select a function by minimising a regularised risk functional, coupled with a cross validation scheme. Later in Chapter 3 we will present a new algorithm, the high-probability minimax probability machine, that directly minimises the high-probability worst case future misclassification rates, and in practice we saw that an approximation to this approach had to be used in conjunction with cross-validation to choose the desired function.

The support vector machine was presented as an example of an algorithm motivated by statistical learning theory. We saw that it minimised a regularised risk functional and showed that the complexity of a function could be related to its norm. We focused on the dual form of the solution, which led to our discussion of kernel based methods. If the input vectors appeared only as inner products in the learning algorithm then the kernel trick could be applied, meaning that very high dimensional feature spaces could be used without having to ever explicitly compute them. In Chapter 4 we make use of both the support vector methodology and kernel based methods in the construction of our large margin solution to learning over unknown graph structures. The main differences between our approach and the original SVM is that our feature mapping is over the joint input-output space and there are an exponential number of constraints for each training sample, however much of the same theory and machinery can be applied to it.

Chapter 3

Linear discriminants: moments, uncertainty and applications

In the previous chapter we introduced the support vector machine SVM, a state-of-the-art classifier, which is largely motivated by the insights provided by statistical learning theory. We showed that the SVM constructs a prediction rule, that is linear in feature space, using a set of data points (supports) that would appear to reside on the periphery of the class-conditional densities. Using this perspective on how the SVM constructs its solution, one could imagine that in high noise environments, such as financial markets, the solution that it finds may struggle to generalise well on unseen examples. This could be a result of the solution being comprised of examples that are atypical of the true underlying distributions, and thus offer little insight into where the boundaries between the two classes reside. In this chapter we present a different approach to binary classification. Rather than constructing the predictor using a set of points that reside on the periphery of the class-conditional densities, we present algorithms that make use of the first and second order moments of class-conditional densities. The hope is that by using the moments of a distribution, rather than peripheral points, we are able to find solutions that are more robust to high noise environments.

We begin the chapter by introducing one of the oldest and best-known moment based approaches, Fisher's discriminant analysis FDA. We show that this approach is capable of producing the Bayes optimal solution when both class-conditional densities are normally distributed and share the same covariance matrix. However, this assumption rarely holds true in reality and Fisher's discriminant struggles to provide generalisation guarantees outside of this idyllic setting. This leads us to consider the Minimax Probability Machine MPM, a distributional free approach for binary classification that directly minimises the worst case probability of misclassifying future data points. The only information that is required a

priori is knowledge of the means and covariances of the class-conditional densities. In practice however, empirical moments have to be used and it is likely that in high noise/small sample environments, the accuracy of these moment estimates can be poor. This can lead to suboptimal predictors and overly confident expectations regarding the future misclassification error rate. To address this we formulate a new version of the MPM, one that takes into consideration the uncertainty of the empirical moments and provides guarantees on future misclassification error rates that hold with high-probability. We observe that the incorporation of moment uncertainty introduces a natural regularisation component into the optimisation scheme, where a smaller number of observations for a particular class results in greater uncertainty regarding its distribution, thus warranting additional regularisation. This is an often overlooked component of binary classifiers, where the class-conditional distributions are traditionally jointly regularised, ignoring the relative amount of information that is available for each class.

The formulation of a high-probability approach to MPM's, and the derivation of a suitable algorithm for solving the optimisation problem, represents the main theoretical contribution of this chapter. From a practical perspective, we evaluate the performance of the classifiers on a range of UCI benchmark datasets, whilst also experimenting under a number of different noise regimes. Furthermore, we present a number of experiments relating to the prediction of financial time series, and present a general case for favouring classifiers constructed using moments rather than supports, when predicting the movement of financial instruments.

3.1 Fisher's discriminant analysis

The basic idea behind discriminant analysis is to find a function that is capable of accurately discriminating between input data, $\mathcal{X} \subseteq \mathbb{R}^n$, belonging to different classes. The goal is to find a discriminant function $f : \mathcal{X} \to \mathbb{R}^d$, that evaluates $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$ to be similar when $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar, and different otherwise. Traditionally the similarity of observations $\mathbf{x}_i$ and $\mathbf{x}_j$ is measured using their class membership y_i and y_j, whereas the similarity of $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$ is measured using their Euclidean distance. This chapter focuses on linear predictors for binary classification tasks, which result in discriminant functions defined by $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$, where $\mathbf{w} \in \mathbb{R}^n$. In order to perform the analysis and find the optimal $\mathbf{w}$, we must first define an evaluation criterion J that measures the discriminatory power of the function. More formally, given some criterion $J(\mathbf{w})$ and a set of constraints C on the projection $\mathbf{w}$, the discriminant analysis procedure is given by

$$\mathbf{w}_* = \mathbf{argmax}\ J(\mathbf{w})\ \text{s.t.}\ \mathbf{w} \in C.$$

The evaluation criterion $J(\mathbf{w})$ is chosen at the practitioners discretion depending on what characteristics they may wish to exploit in the data, or what restrictions they may want to enforce upon the projections. For a good discussion of how to choose the evaluation criterion, see [Mika et al., 2003]. Our focus here, is on the criterion used in Fisher's discriminant analysis FDA [Fisher, 1936], namely finding a projection vector that maximises the distance between the projected class means, whilst maintaining a small degree of variance around these means. These two quantities, the distance between projected class means and the projected class variances, are referred to in the literature as between class scatter and within class scatter, respectively.

Given that we are not privy to the joint probability distribution over input and output spaces, to construct our discriminant we must make use of a training set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$, of m labelled training samples. Let $S_1 = \{\mathbf{x}_i \,|\, y_i = 1\}$ and $S_0 = \{\mathbf{x}_i \,|\, y_i = 0\}$ denote the training examples belonging to each class. The empirical class means $\hat{\boldsymbol{\mu}}_1$, $\hat{\boldsymbol{\mu}}_0$ and covariance matrices $\hat{\Sigma}_1$, $\hat{\Sigma}_0$ are given by

$$\hat{\boldsymbol{\mu}}_j = \frac{1}{m_j} \sum_{\mathbf{x} \in S_j} \mathbf{x} \quad \text{and} \quad \hat{\Sigma}_j = \frac{1}{m_j} \sum_{\mathbf{x} \in S_j} (\mathbf{x} - \hat{\boldsymbol{\mu}}_j)(\mathbf{x} - \hat{\boldsymbol{\mu}}_j)^T \quad \text{for } j = 0, 1,$$

where $m_j = |S_j|$ is the number of training samples belonging to S_j, where $m_1 + m_0 = m$. We see that by projecting the data onto the direction $\mathbf{w}$, the projected class means are given by

$$\hat{\mu}_j = \frac{1}{m_j} \sum_{\mathbf{x} \in S_j} \mathbf{w}^T \mathbf{x} = \mathbf{w}^T \hat{\boldsymbol{\mu}}_j \,.$$

Similarly the projected class variances are given by

$$\hat{\sigma}_j^2 = \frac{1}{m_j} \sum_{\mathbf{x} \in S_j} (\mathbf{w}^T \mathbf{x} - \hat{\mu}_j)^2 = \frac{1}{m_j} \sum_{\mathbf{x} \in S_j} (\mathbf{w}^T \mathbf{x} - \mathbf{w}^T \hat{\boldsymbol{\mu}}_j)^2 = \mathbf{w}^T \hat{\Sigma}_j \mathbf{w}.$$

The goal is to maximise the distance between these projected means, whilst ensuring the variance around them remains small. Mathematically we measure a projection vector's ability to do this by the expression:

$$J(\mathbf{w}) = \frac{(\hat{\mu}_1 - \hat{\mu}_0)^2}{m_1 \hat{\sigma}_1^2 + m_0 \hat{\sigma}_0^2} = \frac{\mathbf{w}^T (\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0)(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0)^T \mathbf{w}}{\mathbf{w}^T (m_1 \hat{\Sigma}_1 + m_0 \hat{\Sigma}_0) \mathbf{w}} = \frac{\mathbf{w}^T C_B \mathbf{w}}{\mathbf{w}^T C_W \mathbf{w}}, \tag{3.1}$$

where C_B and C_W represent the between-class and within-class scatter, respectively. Note that the within-class scatter C_W is a weighted combination of the variances around each class, where the weighting depends on the relative proportions of each class in the training sample.

3.1.1 Finding the solution

It is well known that the maximisation of (3.1) has a global optimum and that this can be found by solving the generalised eigenproblem $C_B\mathbf{w} = \lambda C_w\mathbf{w}$, and setting $\mathbf{w}$ to the eigenvector with the maximum eigenvalue. To show this, first we differentiate $J(\mathbf{w})$ with respect to $\mathbf{w}$

$$\begin{aligned}\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} &= \frac{(\mathbf{w}^T C_B\mathbf{w})C_W\mathbf{w} - (\mathbf{w}^T C_W\mathbf{w})C_B\mathbf{w}}{(\mathbf{w}^T C_W\mathbf{w})^2} = 0 \\ &\implies (\mathbf{w}^T C_W\mathbf{w})C_B\mathbf{w} = (\mathbf{w}^T C_B\mathbf{w})C_W\mathbf{w} \\ &\implies C_B\mathbf{w} = \left(\frac{\mathbf{w}^T C_B\mathbf{w}}{\mathbf{w}^T C_W\mathbf{w}}\right) C_W\mathbf{w}. \end{aligned} \tag{3.2}$$

It is clear that (3.2) is a generalised eigenproblem where $\mathbf{w}$ is the eigenvector and $\left(\frac{\mathbf{w}^T C_B\mathbf{w}}{\mathbf{w}^T C_W\mathbf{w}}\right) = J(\mathbf{w})$ is its corresponding eigenvalue. Therefore in order to maximse $J(\mathbf{w})$ we simply use the eigenvector corresponding to the largest eigenvalue of the eigenproblem $C_B\mathbf{w} = \lambda C_W\mathbf{w}$. Alternatively given that the solution is invariant to scaling i.e. $J(\lambda\mathbf{w}) = J(\mathbf{w})$ for $\lambda > 0$, we can use the generalised eigenproblem to derive a simple expression for $\mathbf{w}$ by ignoring the scalar constants

$$\begin{aligned}\lambda C_W\mathbf{w} = C_B\mathbf{w} = (\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0)(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0)^T\mathbf{w} &\propto (\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0) \\ \implies \mathbf{w} = C_W^{-1}(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0).&\end{aligned}$$

Furthermore, in [Duda et al., 2012; Mika, 2002] it was shown that Fisher's discriminant analysis was equivalent to the problem of least squares regression when the output variables are set to the class labels.

3.1.2 Optimality

The original goal of FDA was not to directly construct a classifier but rather to project the input data into a lower dimensional space were discrimination between classes would become easier. Therefore it is often referred to as a supervised dimensionality reduction technique, and shares a number of similarities with its unsupervised counterpart, principal components analysis. Despite this, it turns out that under a specific set of conditions, the direction found using FDA is equivalent to that given by the Bayes optimal classifier i.e. the one that minimises the expected misclassification rate over the joint probability distribution of $\mathcal{X} \times \mathcal{Y}$. At first glance it may not seem obvious that the maximisation of $J(\mathbf{w})$ would produce a solution that replicated the Bayes optimal classifier, however it turns out that if both class-conditional densities are normally distributed and share the same covariance structure i.e. $\Sigma = \Sigma_1 = \Sigma_0$, then the direction found using Fisher's discriminant is equal to

that used in the Bayes optimal classifier, up to an arbitrary scaling factor. To see this, let $p = P(y = 1)$ and recall that the class conditional probabilities $P(\mathbf{x}|y = j)$ for a normally distributed vector are given by

$$P(\mathbf{x}|y = j) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_j)^T\Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_j)\right),$$

where $|A|$ refers to the determinant of the matrix $A \in \mathbb{R}^{d\times d}$. Bayes rule states that

$$P(y = 1|\mathbf{x}) = \frac{pP(\mathbf{x}|y = 1)}{pP(\mathbf{x}|y = 1) + (1 - p)P(\mathbf{x}|y = 0)},$$

therefore if $P(y = 1|\mathbf{x}) \geq 0.5$ then the Bayes optimal classifier predicts that $\mathbf{x}$ belongs to class 1, and class 0 otherwise. By comparing the log probabilities we see that $\log P(y = 1|\mathbf{x}) \geq \log P(y = 0|\mathbf{x})$ is equivalent to

$$\begin{aligned} &-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_1)^T\Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_1) + \log p \geq -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_0)^T\Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_0) + \log(1 - p) \\ &\implies (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^T\Sigma^{-1}\mathbf{x} - (\boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0^T\Sigma^{-1}\boldsymbol{\mu}_0) + \log p - \log(1 - p) \geq 0 \\ &\implies \mathbf{w}^T\mathbf{x} - b \geq 0, \end{aligned} \tag{3.3}$$

and we recover the same projection vector $\mathbf{w}$ as that found using Fisher's discriminant i.e.

$$\mathbf{w} = \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0).$$

Note that the bias term b for the Bayes optimal classifier is given by

$$b = \frac{1}{2}\mathbf{w}^T(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_0) - \log p + \log(1 - p), \tag{3.4}$$

which provides an insight on how to best use the direction found by FDA as a classifier. Therefore by estimating p using $\hat{p} = \frac{m_1}{m_1+m_0}$, along with the empirical moments $\hat{\boldsymbol{\mu}}_j$ and $\hat{\Sigma}_j$, we can compute b accordingly, and classify points using $g(\mathbf{x}) = \mathbf{I}[\mathbf{w}^T\mathbf{x} - b \geq 0]$.

When implementing the algorithm, Bayes optimality can only be assured if the empirical estimates of the moments are indeed true. Given the limited size of the training sample, the best estimates of the class means are given by their empirical counterparts, and the best estimate of the shared covariance structure is given by a weighted combination of the individual covariances i.e. $m\Sigma \approx m_1\hat{\Sigma}_1 + m_0\hat{\Sigma}_0 = C_W$. It is important to point out that the optimality of the FDA solution relies heavily on a specific set of conditions that the class-conditional densities must satisfy. A straightforward extension to FDA is quadratic discriminant analysis, which considers the situation where the class-conditional densities are once again normally distributed, however they are allowed to have different covariances.

Following a similar technique to above, it can be shown to provide Bayes optimal solution. Given that our interest here is in linear discriminants, we refer the interested reader to [Friedman et al., 2001] for more details.

3.2 Minimax probability machine (MPM)

So far we have discussed the case where the underlying distributions generating the data are normal and showed that when the two classes share the same covariance structure the direction found by Fisher's discriminant is indeed optimal. In this section we present the Minimax Probability Machine MPM [Lanckriet et al., 2003]; a moment based linear predictor that directly minimises the probability of misclassification on future examples in a worst-case setting. The only information that is required for this is prior knowledge of the class means and covariances. No further assumptions regarding the underlying distributions is required, and the algorithm works by minimising the maximum probability of misclassification, which is taken with respect to all class-conditional densities with a given mean and covariance. The use of class conditional densities and their associated error rates is similar in respect to the methods proposed in [Marchand and Shawe-Taylor, 2002; Sokolova et al., 2002], however these methods assume that the decision boundary can be constructed using a logical combination of a small set of data derived features, whereas this method uses the first and second moments.

3.2.1 Formulation

We begin the formulation of the MPM by briefly mentioning the Chebyshev Inequality, a classical result in probability theory that describes a fundamental relationship between the moments of a random variable. The Chebyshev Inequality states that for any probability distribution of a random variable $x \in \mathbb{R}$, *nearly all* all the values are close to the mean. More precisely, for any probability distribution of a random variable with mean μ and standard deviation σ, the probability that x is further than $v > 0$ standard deviations from the mean μ is upper bounded by $1/v^2$ i.e.

$$P\{|x - \mu| \geq v\sigma\} \leq \frac{1}{v^2}.$$

This concept was indirectly applied in Fisher's discriminant where we looked to minimise the projected variances of the classes and maximise the distances between the projected means, however no explicit guarantees on the probability of misclassification were used and the underlying distributions were assumed to be normal. The MPM is more direct and uses the following multivariate generalisation of the Chebyshev Inequality [Bertsimas and Popescu, 2005; Marshall and Olkin, 1960] to optimise over the worst-case setting of probability dis-

tributions generating the data:

$$\sup_{\mathbf{x}\sim\mathcal{D}} \mathbb{P}[\,\mathbf{x}\in\mathcal{J}\,] = \frac{1}{1+d^2} \text{ where } \; d^2 = \inf_{\mathbf{x}\in\mathcal{J}} (\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}), \tag{3.5}$$

where $\mathbf{x}$ is a random vector, $\mathcal{J}$ is a given convex set, and the supremum is taken over all distributions $\mathcal{D}$ for $\mathbf{x}$ that have mean $\boldsymbol{\mu}$ and covariance Σ. This theorem states that the maximum probability that a random vector $\mathbf{x}\sim\mathcal{D}$ belongs to a convex set $\mathcal{J}$ is related to the minimum Mahalanobis distance d^2 from the centre of the distribution $\boldsymbol{\mu}$ to the set $\mathcal{J}$. In [Lanckriet et al., 2003], the authors showed that when $\mathcal{J}$ is the upper half-space defined the separating hyperplane $\mathcal{H}(\mathbf{w},b) := \{\mathbf{x} \mid \mathbf{w}^T\mathbf{x} = b\}$, this distance d^2 admits a closed form expression given by

$$d^2 = \inf_{\mathbf{w}^T\mathbf{x}\geq b} (\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}) = \begin{cases} \frac{(b-\mathbf{w}^T\boldsymbol{\mu})^2}{\mathbf{w}^T\Sigma\mathbf{w}} & \text{if } \mathbf{w}^T\boldsymbol{\mu} < b \\ 0 & \text{if } \mathbf{w}^T\boldsymbol{\mu} \geq b \end{cases}. \tag{3.6}$$

Returning to the problem of binary classification, let $\mathbf{x}_1$ and $\mathbf{x}_0$ be random vectors drawn from class-conditional densities with mean and covariances given by $(\boldsymbol{\mu}_1, \Sigma_1)$ and $(\boldsymbol{\mu}_0, \Sigma_0)$, respectively. The goal of a linear classifier is to find the projection vector $\mathbf{w}$ and bias b that minimise the probability of future misclassification i.e. points lying on the wrong side of the separating hyperplane $\mathcal{H}(\mathbf{w},b)$. This goal can be represented explicitly using the following optimisation problem

$$\begin{aligned} \max_{\mathbf{w},b,\omega} \; \omega \quad \text{s.t.} \quad & \inf_{\mathbf{x}_1\sim\mathcal{D}_1} P(\mathbf{w}^T\mathbf{x}_1 \geq b) \geq \omega \\ & \inf_{\mathbf{x}_0\sim\mathcal{D}_0} P(\mathbf{w}^T\mathbf{x}_0 \leq b) \geq \omega, \end{aligned}$$

where $\omega \in [0,1]$ is the minimum probability that examples are labelled correctly in the future, and $\mathcal{D}_j$ for $j = 0, 1$ are the set of distributions with moments (μ_j, Σ_j).

To see this, let our classifier predict that $\mathbf{x}$ belongs to class 1 if $\mathbf{w}^T\mathbf{x} \geq b$. The maximum probability that a point drawn from $\mathcal{D}_1$ resides on the wrong side of this hyperplane $\mathcal{H}(\mathbf{w},b)$ is given by

$$\sup_{\mathbf{x}_1\sim\mathcal{D}_1} P(\mathbf{w}^T\mathbf{x}_1 < b) = \frac{1}{1+d^2} = 1-\omega.$$

Therefore the minimum probability that a random vector $\mathbf{x}_1$ resides on the correct side of the hyperplane is greater than ω. Using the closed form expression for the Mahalanobis distance given in (3.6), assuming that $\mathbf{w}^T\boldsymbol{\mu}_1 > b$, we derive the following statement

$$\inf_{\mathbf{x}_1\sim\mathcal{D}_1} P(\mathbf{w}^T\mathbf{x}_1 \geq b) \geq \omega \iff -b + \mathbf{w}^T\boldsymbol{\mu}_1 \geq \kappa(\omega)\sqrt{\mathbf{w}^T\Sigma_1\mathbf{w}},$$

where $\kappa(\omega) = \sqrt{\omega/(1-\omega)}$, and the optimisation problem to be re-written as

$$\max_{\mathbf{w},b,\omega} \ \omega \quad \text{s.t.} \quad -b + \mathbf{w}^T\boldsymbol{\mu}_1 \geq \kappa(\omega)\sqrt{\mathbf{w}^T\Sigma_1\mathbf{w}}$$
$$b - \mathbf{w}^T\boldsymbol{\mu}_0 \geq \kappa(\omega)\sqrt{\mathbf{w}^T\Sigma_0\mathbf{w}}.$$

This allowed the authors in [Lanckriet et al., 2003] to present the following theorem concerning the MPM optimisation scheme:

Theorem 6 *If $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_0$ then the minimax probability decision problem does not have a meaningful solution and the worst case misclassification probability is given by $1 - \omega_* = 1$. Otherwise an optimal hyperplane $\mathcal{H}(\mathbf{w}_*, b_*)$ exists and can be determined by solving the convex optimisation problem*

$$\kappa_*^{-1} := \min_{\mathbf{w}} \sqrt{\mathbf{w}^T\Sigma_1\mathbf{w}} + \sqrt{\mathbf{w}^T\Sigma_0\mathbf{w}} \quad s.t. \quad \mathbf{w}^T(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) = 1, \tag{3.7}$$

and setting b to the value

$$b_* = \mathbf{w}_*^T\boldsymbol{\mu}_1 - \kappa_*\sqrt{\mathbf{w}_*^T\Sigma_1\mathbf{w}_*}, \tag{3.8}$$

where $\mathbf{w}_*$ is the optimal solution to (3.7). The optimal worst-case misclassification probability is given by

$$1 - \omega_* = \frac{1}{1+\kappa_*^2} = \frac{\left(\sqrt{\mathbf{w}_*^T\Sigma_1\mathbf{w}_*} + \sqrt{\mathbf{w}_*^T\Sigma_0\mathbf{w}_*}\right)^2}{1 + \left(\sqrt{\mathbf{w}_*^T\Sigma_1\mathbf{w}_*} + \sqrt{\mathbf{w}_*^T\Sigma_0\mathbf{w}_*}\right)^2}. \tag{3.9}$$

If either Σ_1 or Σ_0 is positive definite, the optimal hyperplane is unique.

To gain a better understanding of the optimisation problem (3.7) we can refer to its dual formulation, which provides an appealing geometrical interpretation. Assuming $\boldsymbol{\mu}_1 \neq \boldsymbol{\mu}_0$ and at least either Σ_1 or Σ_0 is positive definite, the dual problem can be written as

$$\min_{\kappa,\mathbf{u},\mathbf{v}} \ \kappa \ : \ \boldsymbol{\mu}_1 + \Sigma_1^{1/2}\mathbf{u} = \boldsymbol{\mu}_0 + \Sigma_0^{1/2}\mathbf{v}, \ ||\mathbf{u}|| \leq \kappa, \ ||\mathbf{v}|| \leq \kappa.$$

For a given value $\kappa \geq 0$, we consider two ellipsoids centred around the class means whose shape is controlled by their covariance matrices:

$$\mathcal{E}_1(\kappa) = \{\mathbf{x}_1 = \boldsymbol{\mu}_1 + \Sigma_1^{1/2}\mathbf{u} : \ ||\mathbf{u}|| \leq \kappa\} \ \text{and} \ \mathcal{E}_0(\kappa) = \{\mathbf{x}_0 = \boldsymbol{\mu}_0 + \Sigma_0^{1/2}\mathbf{v} : \ ||\mathbf{v}|| \leq \kappa\}.$$

These ellipsoids represent the set points where the Mahalanobis distance to the class means is less than κ. Clearly as the value of κ increases these sets will intersect. For the optimal κ

the ellipsoids will be tangent to one another and the MPM hyperplane is then the common tangent to these sets. This is shown in Figure 3.1, where we are increasing the value of κ until the two ellipsoids are tangent to one another.

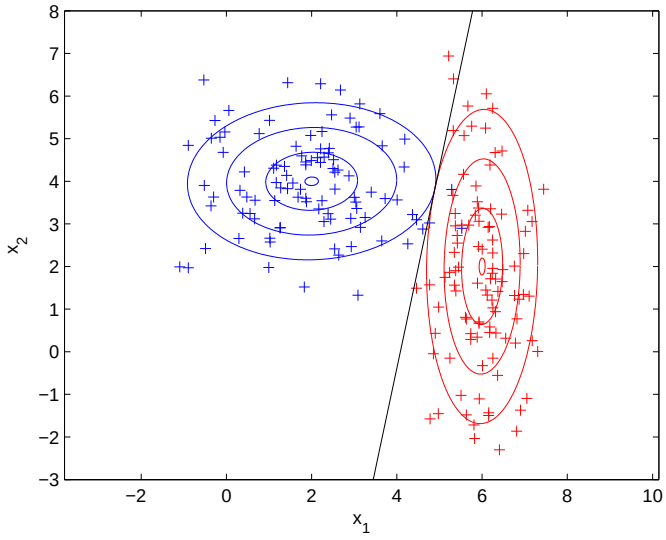


Fig. 3.1 Minimax probability machine visualisation

The authors [Lanckriet et al., 2003] showed that it was possible to solve the optimisation (3.7) using an iterative least-squares scheme, which has a worst-case complexity $\mathcal{O}(d^3)$. Furthermore, their empirical results show that the MPM approach to classification is competitive with the state-of-the-art SVM, thus providing evidence in support of the MPM as an efficient and effective approach for binary classification. These encouraging results resulted in increased attention from the community. In [Huang et al., 2004] the authors identified an oversight in the original MPM formulation: the assumption that the prior probability of each class is the same. The authors showed that if the prior probabilities of the classes differed, then it was no longer optimal to minimise a single worst-case future misclassification rate but rather one should minimise a weighted combination of class specific worst-case misclassification rates. The weights correspond to their prior class probability and this formulation became known as the minimum error MPM ME-MPM. An alternative approach for dealing with imbalanced data was presented in [Osadchy et al., 2015], where the authors optimised an objective that used the SVM hinge loss for the less frequent class and the minimax loss formulation for the abundant class. A transductive minimax probability machine was proposed in [Huang et al., 2014], here unlabelled test points were assigned classes based upon their ability to minimise the worst case error bound and it was shown to be especially competitive with the transductive SVM on semi-supervised learning tasks. Similar to the transductive

setting, in [Huang et al., 2015] the authors focus on the problem of clustering by assigning unlabelled data to clusters in an attempt to optimise criterion defined by the MPM framework.

The solutions found by the FDA and MPM are very similar in nature with both constructing the predictor using the first and second order moments of the class-conditional densities. To make the connection clearer, the scalar invariance of $J(\mathbf{w})$ with respect to $\mathbf{w}$ allows us to select an arbitrary value for the denominator in (3.1) and introduce the constraint $\mathbf{w}^T(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0) = 1$. This translates the FDA into an optimisation problem that takes the same form as the MPM. The FDA assumes that both classes share the same covariance structure, and in practice the best estimate for this is given by $m\hat{\Sigma} = m_1\hat{\Sigma}_1 + m_0\hat{\Sigma}_0$, which results in the following optimisation problem

$$\min_{\mathbf{w}} \sqrt{\mathbf{w}^T\hat{\Sigma}\mathbf{w}} \quad \text{s.t.} \quad \mathbf{w}^T(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0) = 1.$$

This is equivalent to the MPM formulation if both classes have the same covariance structure. Furthermore, if we assume that the prior probability of each class is the same then it is easy to show that the bias term found using the MPM in 3.8 is equivalent to that implied by the Bayes optimal solution i.e. $b_* = \mathbf{w}_*^T(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_0)$. This allows the FDA to be interpreted in the same context as MPM, where the optimal hyperplane is determined by hyperplane that is tangential to the initial intersection of ellipsoids having identical size and shape, that are centered at the respective class means. In Figure 3.2 we show the difference between the solutions found using the MPM approach of different class-conditional covariances and the FDA's use of a pooled covariance. We see that the pooling of the covariance matrices results in a loss of predictive power

3.2.2 Moment uncertainty

The only assumption that was made during the formulation of the MPM was that the class-conditional means and covariances are known in advance. However in practice, the true moments are not known and during the implementation of the algorithm their empirical counterparts must be used instead. In [Lanckriet et al., 2003] the authors handled the use of empirical moments by using a specific uncertainty model, which they believed to be realistic from a statistical point of view but more importantly was numerically tractable for the optimisation problem. The uncertainty sets were given by

$$\begin{aligned}
\mathcal{X}_1 &= \left\{(\boldsymbol{\mu}_1 - \hat{\boldsymbol{\mu}}_1)^T\Sigma_1^{-1}(\boldsymbol{\mu}_1 - \hat{\boldsymbol{\mu}}_1) \leq \nu^2,\ \ ||\Sigma_1 - \hat{\Sigma}_1||_F \leq \rho\right\} \\
\mathcal{X}_0 &= \left\{(\boldsymbol{\mu}_0 - \hat{\boldsymbol{\mu}}_0)^T\Sigma_0^{-1}(\boldsymbol{\mu}_0 - \hat{\boldsymbol{\mu}}_0) \leq \nu^2,\ \ ||\Sigma_0 - \hat{\Sigma}_0||_F \leq \rho\right\},
\end{aligned}$$

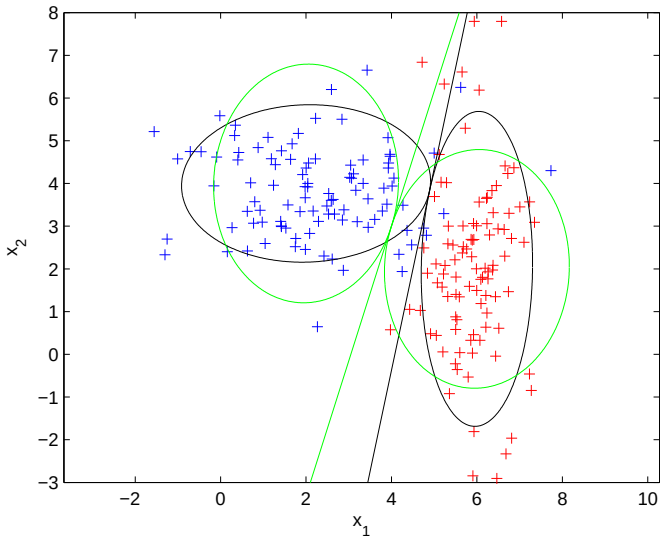


Fig. 3.2 In this figure we show the geometrical differences of the solutions found using MPM and FDA . The green plots represent the FDA ellipsoids and hyperplane, whereas the black plots correspond to the MPM. In the MPM solution we see that the difference in the covariance structures permits a better solution that better separates the two classes, however that the pooling together of covariance matrices results in a sub-optimal hyperplane formed using the FDA approach.

where $\hat{\boldsymbol{\mu}}_1, \hat{\boldsymbol{\mu}}_0$ and $\hat{\Sigma}_1, \hat{\Sigma}_0$ are the nominal empirical estimates of the class means and covariances, respectively. The value $\nu \geq 0$ controls the maximum Mahalanobis distance of the true mean from the empirical estimate and $\rho \geq 0$ controls the deviation of the true covariance from the empirical one, where $||\cdot||_F$ is the Frobenius norm.

Suppose we are only uncertain about the true value of the mean ($\nu > 0$ and $\rho = 0$). Using the uncertainty set $\mathcal{X}_1$, the worst-case setting corresponds to the lowest possible value of the projection of $\boldsymbol{\mu}_1$ on to $\mathbf{w}$[1] i.e.

$$\min_{\boldsymbol{\mu}_1:(\boldsymbol{\mu}_1-\hat{\boldsymbol{\mu}}_1)^T\Sigma_1^{-1}(\boldsymbol{\mu}_1-\hat{\boldsymbol{\mu}}_1)\leq\nu^2} \mathbf{w}^T\boldsymbol{\mu}_1 = \mathbf{w}^T\hat{\boldsymbol{\mu}}_1 - \nu\sqrt{\mathbf{w}^T\Sigma_1\mathbf{w}}.$$

This transforms the relationship given in (3.11) to

$$\inf_{\mathbf{x}_1\sim\mathcal{X}_1} P(\mathbf{w}^T\mathbf{x}_1 \geq b) \geq \omega \iff -b + \mathbf{w}^T\hat{\boldsymbol{\mu}}_1 \geq (\kappa(\omega)+\nu)\sqrt{\mathbf{w}^T\Sigma_1\mathbf{w}}.$$

Letting ν be the same for both classes, the optimisation proceeds as before in (3.7), with the same optimal hyperplane $\mathcal{H}(\mathbf{w}_*, b_*)$ and κ_* being found. However when it comes to

[1]Similarly for $\boldsymbol{\mu}_0$ the worst-case projection corresponds to the maximum possible value.

estimating the worst-case misclassification probability, in the robust version κ_* is substituted with $\kappa_*^{\text{rob}} = \kappa_* - \nu$, which results in a more conservative estimate

$$1 - \omega_* = \frac{1}{1 + (\kappa_* - \nu)^2}. \tag{3.10}$$

Note that this only permits a feasible solution if $\kappa_* \geq \nu$, however when this condition is satisfied, the optimal hyperplane $\mathcal{H}(\mathbf{w}, b)$ is the same as that recovered in the original formulation.

We now consider the case where there is only uncertainty regarding the covariance matrix ($\nu = 0$ and $\rho > 0$). Under the uncertainty set $\mathcal{X}_j$ the worst-case setting for the covariance matrix is given by

$$\max_{\Sigma_j : ||\Sigma_j - \hat{\Sigma}_j||_F \leq \rho} \mathbf{w}^T \Sigma_j \mathbf{w} = \mathbf{w}^T \left(\hat{\Sigma}_j + \rho I_n \right) \mathbf{w},$$

which effectively adds a regularisation term to the empirical covariance matrix. The relationship given in (3.11) is now

$$\inf_{\mathbf{x}_1 \sim \mathcal{X}_1} P(\mathbf{w}^T \mathbf{x}_1 \geq b) \geq \omega \iff -b + \mathbf{w}^T \hat{\boldsymbol{\mu}}_1 \geq (\kappa(\omega) + \nu) \sqrt{\mathbf{w}^T \left(\hat{\Sigma}_1 + \rho I_n \right) \mathbf{w}},$$

and the optimisation can proceed as in (3.7) with the inclusion of a regularisation term in each of the empirical covariance matrices. Note that as ρ increases, reflecting the an increase in uncertainty regarding the covariance matrices, we effectively increase the amount of regularisation making the class-conditional covariances increasingly spherical. The geometrical interpretation of the MPM would indicate that there will be a tendency for the projection vector $\mathbf{w}_*$ to rotate towards the vector joining the means as the level of uncertainty increases, and for the bias term to shift to the mid-way point on the line joining the class-means. Given that the uncertainty in means in this setting doesn't effect the hyperplane, we can interpret the projection vector given by the difference between means as the optimal solution in the face of high uncertainty i.e. given that our information is very weak, the difference between the means offers the most robust solution. To include both measures of uncertainty, one simply substitutes the covariance matrices Σ_j with ρ-regularised ones $\Sigma_j + \rho$ in the original optimisation (3.7), and use (3.10) to compute worst-case misclassification probability.

3.3 High-probability minimax probability machine (HP-MPM)

In [Lanckriet et al., 2003] a main motivation for the choice of uncertainty set was the ease in which it could be incorporated into the original optimisation scheme. Despite this appealing feature, what this approach failed to provide was a solid statistical explanation of why these uncertainty sets where being used, nor did it provide guidance or insight on how best to

select the value of the parameters ν and ρ to reflect the uncertainty. Furthermore they failed to take into consideration the influence that the uncertainty on the covariance matrix has on the Mahalanobis distance between the true mean and the empirical one. In this section we seek to address these concerns by presenting an optimisation scheme that minimises the worst case probability of misclassification whilst taking into consideration the uncertainty regarding the use of empirical moments. The key feature of the MPM approach is the if and only if statement

$$\inf_{\mathbf{x}_1 \sim \mathcal{D}_1} P(\mathbf{w}^T \mathbf{x}_1 \geq b) \geq \omega \iff -b + \mathbf{w}^T \boldsymbol{\mu}_1 \geq \kappa(\omega) \sqrt{\mathbf{w}^T \Sigma_1 \mathbf{w}}. \tag{3.11}$$

Given that we do not know the true moments of the distribution we have to ask ourselves, what conditions must the empirical moments satisfy in order to be sure, with high probability, that the true moments satisfy the condition outlined in (3.11). If we can find these conditions that we can say, with high-probability, that the hyperplane we find will have a classification error no worse than $1 - \omega$.

3.3.1 HP-MPM Formulation

In this section we seek to address the shortfalls of the uncertainty sets provided in [Lanckriet et al., 2003] and present conditions that ensure the inequality (3.11) is satisfied with high-probability when using empirical moments. We begin by using a result presented in [Shawe-Taylor and Cristianini, 2003] concerning the high-probability upper bounds on the deviation of true moments $(\bar{\mathbf{x}}, \Sigma)$ from their empirical counterparts $(\hat{\mathbf{x}}, \hat{\Sigma})$: high-probability in the sense that the probability that the true value of the moment deviates from the empirical one by more than $\epsilon \in \mathbb{R}$ is less than $\delta \geq 0$. They showed that the following holds true with probability at least $1 - \delta$:

$$||\bar{\mathbf{x}} - \hat{\mathbf{x}}||_2 \leq \frac{R}{\sqrt{m}} \left(2 + \sqrt{2 \log \frac{1}{\delta}} \right) \quad \text{and} \quad ||\Sigma - \hat{\Sigma}||_F \leq \frac{2R^2}{\sqrt{m}} \left(2 + \sqrt{2 \log \frac{2}{\delta}} \right), \tag{3.12}$$

where $||\cdot||$ and $||\cdot||_F$ denote the L_2 and Frobenius norms, respectively, $R > 0$ is the radius of the smallest sphere containing the support of $\mathcal{X}$ i.e. for all $\mathbf{x} \in \mathcal{X}$, $||\mathbf{x}|| \leq R$, and m is the number of observations that were used to construct the empirical moments. The authors examined the implications of the these deviation bounds on the MPM guarantees, showing that the high-probability worst-case estimate for the future misclassification errors can differ significantly from that found using (3.7). Their focus was on finding what the high-probability worst-case future misclassification rate was, given that the predictor was constructed using the original MPM formulation (3.7). Whereas our focus is on designing an optimisation scheme that directly minimises the high-probability bound on future misclassification by taking into consideration the uncertainty in the empirical moments.

To do this we begin by reviewing how moment uncertainty affects the key equivalence relationship given in (3.11). To simplify the analysis, we assume that the weight vector lies within the unit-ball defined by the L_2-norm i.e. $||\mathbf{w}|| \leq 1$. We want to find a high-probability bound on the deviation of the values in the inequality (3.11) when using empirical and true moments in the expression.

Proposition 7 *Let $\hat{\mathbf{x}}$ and $\hat{\Sigma}$ be the empirical mean and covariance matrix of a sample of m points drawn independently according some probability distribution $\mathcal{D}$ with mean $\bar{\mathbf{x}}$ and covariance matrix Σ. The weight vector $||\mathbf{w}|| \leq 1$ where $\mathbf{w} \neq \mathbf{0}$, and $b \in \mathbb{R}$ are given such that $\mathbf{w}^T\hat{\mathbf{x}} \leq b$. Then if*

$$b - \mathbf{w}^T\hat{\mathbf{x}} \geq \sqrt{\kappa(\omega)^2\mathbf{w}^T\hat{\Sigma}\mathbf{w} + T} \tag{3.13}$$

where

$$T = \frac{4R^2}{\sqrt{m}}\left(2 + \sqrt{2\ln\frac{2}{\delta}}\right) + \kappa(\omega)^2\frac{2R^2}{\sqrt{m}}\left(2 + \sqrt{2\ln\frac{2}{\delta}}\right)$$

then with probability at least $1 - \delta$ over the draw of the random sample

$$b - \mathbf{w}^T\bar{\mathbf{x}} \geq \kappa(\omega)\sqrt{\mathbf{w}^T\Sigma\mathbf{w}} \quad \textit{and} \quad \inf_{\mathbf{x}\sim\mathcal{D}} P(\mathbf{w}^T\mathbf{x} \geq b) \geq \omega.$$

Proof *To prove this we show that if*

$$(b - \mathbf{w}^T\hat{\mathbf{x}})^2 - \kappa(\omega)^2\mathbf{w}^T\hat{\Sigma}\mathbf{w} \geq T$$

then with probability at least $1 - \delta$

$$(b - \mathbf{w}^T\bar{\mathbf{x}})^2 - \kappa(\omega)^2\mathbf{w}^T\Sigma\mathbf{w} \geq 0.$$

We do this by bounding the high-probability differences in the value of the expressions on the left hand side of the inequalities

$$\begin{aligned}
&\left|(b - \mathbf{w}^T\hat{\mathbf{x}})^2 - \kappa(\omega)^2\mathbf{w}^T\hat{\Sigma}\mathbf{w} - (b - \mathbf{w}^T\bar{\mathbf{x}})^2 + \kappa(\omega)^2\mathbf{w}^T\Sigma\mathbf{w}\right| \\
&\leq ||\hat{\mathbf{x}} - \bar{\mathbf{x}}||\,(2b + ||\hat{\mathbf{x}} + \bar{\mathbf{x}}||) + \kappa(\omega)^2\left|\mathbf{w}^T\hat{\Sigma}\mathbf{w} - \mathbf{w}\Sigma\mathbf{w}\right| \\
&\leq ||\hat{\mathbf{x}} - \bar{\mathbf{x}}||4R + \kappa(\omega)^2||\hat{\Sigma} - \Sigma||_F.
\end{aligned}$$

The proof is completed by using the bounds on the empirical moments presented (3.12) with

δ replaced with $\delta/2$,

$$\begin{aligned}&\left|(b-\mathbf{w}^T\hat{\mathbf{x}})^2-\kappa(\omega)^2\mathbf{w}^T\hat{\Sigma}\mathbf{w}-(b-\mathbf{w}^T\bar{\mathbf{x}})^2+\kappa(\omega)^2\mathbf{w}^T\Sigma\mathbf{w}\right|\\&\leq\frac{4R^2}{\sqrt{m}}\left(2+\sqrt{2\ln\frac{2}{\delta}}\right)+\kappa(\omega)^2\frac{2R^2}{\sqrt{m}}\left(2+\sqrt{2\ln\frac{2}{\delta}}\right).\end{aligned}$$

Note that the bound comes into play when we consider

$$(b-\mathbf{w}^T\hat{\mathbf{x}})^2-\kappa(\omega)^2\mathbf{w}^T\hat{\Sigma}\mathbf{w}\geq(b-\mathbf{w}^T\bar{\mathbf{x}})^2-\kappa(\omega)^2\mathbf{w}^T\Sigma\mathbf{w},$$

and it holds true regardless of the bound if the inequality is reversed. ■

To formulate the HP-MPM optimisation scheme we use each classes corresponding inequality (3.13), where the class specific uncertainty is captured by the term T_j for $j=0,1$ with

$$\begin{aligned}T_j&=\frac{4R^2}{\sqrt{m_j}}\left(2+\sqrt{2\ln\frac{2}{\delta}}\right)+\kappa(\omega)^2\frac{2R^2}{\sqrt{m_j}}\left(2+\sqrt{2\ln\frac{2}{\delta}}\right)\\&=2A_j+\kappa(\omega)^2A_j,\end{aligned}$$

where

$$A_j=\frac{2R^2}{\sqrt{m_j}}\left(2+\sqrt{2\ln\frac{2}{\delta}}\right). \tag{3.14}$$

We can drop the dependence of the optimisation on ω by noting the monotonic relationship it has with $\kappa(\omega)$, and using the constraint $||\mathbf{w}||\leq 1$ and using the inequalities (3.13), the minimax program becomes

$$\begin{aligned}\max_{\mathbf{w},b,\kappa}\ \kappa\quad\text{s.t.}\quad&||\mathbf{w}||\leq 1\\&-b+\mathbf{w}^T\hat{\mathbf{x}}_1\geq\sqrt{2A_1+\kappa^2\left(\mathbf{w}^T\hat{\Sigma}_1\mathbf{w}+A_1\right)}\\&b-\mathbf{w}^T\hat{\mathbf{x}}_0\geq\sqrt{2A_0+\kappa^2\left(\mathbf{w}^T\hat{\Sigma}_0\mathbf{w}+A_0\right)}.\end{aligned}$$

Corollary 8 *For $j=0,1$, let $\hat{\mathbf{x}}_j$ and $\hat{\Sigma}_j$ be the empirical mean and covariance matrix of m_j points drawn independently from distributions D_j with true mean $\bar{\mathbf{x}}_j$ and covariance matrix Σ_j, and let A_j be defined according to (3.14). If $||\hat{\mathbf{x}}_1-\hat{\mathbf{x}}_0||\leq\sqrt{2A_1}+\sqrt{2A_0}$ then the high probability* MPM *decision problem does not have a meaningful solution and the worst-case misclassification probability is given by $1-\omega_*=1$. Otherwise an optimal hyperplane*

$\mathcal{H}(\mathbf{w}_*, b_*)$ exists and can be determined by solving the optimisation problem given by

$$\max_{\mathbf{w},\kappa} \kappa \quad s.t. \quad ||\mathbf{w}|| \leq 1 \tag{3.15}$$
$$\mathbf{w}^T(\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_0) = \sqrt{2A_1 + \kappa^2\left(\mathbf{w}^T\hat{\Sigma}_1\mathbf{w} + A_1\right)} + \sqrt{2A_0 + \kappa^2\left(\mathbf{w}^T\hat{\Sigma}_0\mathbf{w} + A_0\right)},$$

and setting b to the value

$$b_* = \mathbf{w}_*^T\hat{\mathbf{x}}_1 - \sqrt{2A_1 + \kappa_*^2\left(\mathbf{w}_*^T\hat{\Sigma}_1\mathbf{w}_* + A_1\right)} = \mathbf{w}_*^T\hat{\mathbf{x}}_0 + \sqrt{2A_0 + \kappa_*^2\left(\mathbf{w}_*^T\hat{\Sigma}_0\mathbf{w}_* + A_0\right)},$$

where $\mathbf{w}_*$ and κ_* are the optimal solutions to (3.15). Then with probability at least $1-\delta$ over the draws of the random sample, the optimal worst-case misclassification probability is given by

$$1 - \omega_* = \frac{1}{1+\kappa_*^2}.$$

When presented with a new input observation $\mathbf{x}'$, we make our prediction according to what side of the optimal hyperplane the point resides i.e. we predict that $y' = 1$ if $\mathbf{w}_*^T\mathbf{x}' - b_* \geq 0$, and that $y' = 0$ otherwise.

3.3.2 Optimisation Scheme

The optimisation problem given in (3.15) can not be solved using the same approach taken in [Lanckriet et al., 2003] because of the unit L_2-norm restriction on $\mathbf{w}$, and the presence of the uncertainty terms A_j under the square root. To solve this problem we propose the use of an auxiliary function $h(\mathbf{w}, \kappa)$ in conjunction with an alternating update scheme over $\mathbf{w}$ and κ. The auxiliary function is given by

$$h(\mathbf{w}, \kappa) = \mathbf{w}^T(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0) - \sqrt{2A_1 + \kappa^2\left(\mathbf{w}^T\hat{\Sigma}_1\mathbf{w} + A_1\right)} - \sqrt{2A_0 + \kappa^2\left(\mathbf{w}^T\hat{\Sigma}_0\mathbf{w} + A_0\right)}.$$

Note that the class uncertainty terms require the computation of span of the data i.e. find R such that $||\mathbf{x}|| \leq R$ for all $\mathbf{x} \in \mathcal{X}$. During implementation this will have to be estimated from the training sample or can be enforced by some normalisation scheme that is independent of the learning algorithm.

This auxiliary function will be used during the intermediate steps of the optimisation scheme, which lead us towards the optimal solution. Note that the goal of the optimisation scheme is to find the maximum value of κ for which there exists a $\mathbf{w}$ that satisfies the constraints. In this approach, we incrementally adjust the value $\mathbf{w}$ to allow for an increase in the value of κ . At convergence, we can no longer find a $\mathbf{w}$ that increases the value of the auxiliary

function. Therefore we are no longer able to increase the value of κ and we have reached the optimal solution.

Initialisation

To initialise the optimisation, we begin with $\kappa = 0$, and find the value of $\mathbf{w}$ that maximises $h(\mathbf{w}, \kappa)$ subject to the constraints that $||\mathbf{w}|| \leq 1$. This has a closed form solution $(\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_0)/||\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_0|| = \arg\max_{||\mathbf{w}||\leq 1} h(\mathbf{w}, 0)$, and provides the conditions necessary for a meaningful solution to the high probability MPM decision problem i.e. we require that $\max_{||\mathbf{w}||\leq 1} h(\mathbf{w}, 0) > 0$ in order to be able to find a positive value of κ in the next step of the optimisation scheme.

w-step

For non-initialisation $\mathbf{w}$-steps, the goal is maximise the value of the auxiliary function by performing gradient ascent subject to our constraint $||\mathbf{w}|| \leq 1$. We can show that $h(\mathbf{w}, \kappa)$ is a concave in $\mathbf{w}$ and therefore every local optimum will be a global optimum. Therefore we can use standard constrained optimisation tools to solve this intermediate problem. Note that we do not need to run these constrained optimisations to convergence, we simply need the value of the auxiliary function to increase, in order to allow for a larger value of κ in the next step. We view this as a constrained maximisation subject to some implicit degree of regularisation imposed by the value of κ.

κ-step: In order to continue the optimisation, we require that the $\mathbf{w}$-step results in a strictly positive value for the auxiliary function, $h(\mathbf{w}, \kappa) > 0$. If this is not the case then the optimisation has converged, and we have reached the optimal solution. If $h(\mathbf{w}, \kappa) > 0$, we can increase the value of κ to κ' such that the value of the auxiliary function is zero i.e. $h(\mathbf{w}, \kappa') = 0$. This can be performed using a simple line-search procedure, or by finding the roots of a quadratic expression involving κ. Note that in order for the optimisation to progress we must find κ' such that $\kappa' > \kappa$. To simplify the range of the line-search we observe an upper bound on the value of κ', namely $\kappa' \leq \kappa_u = ||\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_0||/\left(\sqrt{\mathbf{w}^T\hat{\Sigma}_1\mathbf{w}} + \sqrt{\mathbf{w}^T\hat{\Sigma}_0\mathbf{w}}\right)$.

Optimal solution

We prove that the optimal solution for the weight vector $\mathbf{w}_*$ will have a unit L_2-norm i.e. $||\mathbf{w}_*|| = 1$. To do this suppose that $||\mathbf{w}_*|| < 1$, we know that at optimality $h(\mathbf{w}_*, \kappa_*) = 0$ and that $\mathbf{w}' = \mathbf{w}_*/||\mathbf{w}_*||$ is also a feasible solution. We show that $h(\mathbf{w}', \kappa_*) > 0$ and that

Algorithm 1 HP-MPM Optimisation Scheme

Input: $\hat{\mathbf{x}}_j, \hat{\Sigma}_j, A_j$ for $j = 0, 1$, tolerance $\epsilon_\kappa > 0$ and $\epsilon_{\mathbf{w}} > 0$
Initialise: $\kappa = 0$, $\mathbf{w} = (\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_0)/||\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_0||$ and $converged = false$
if $||\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_0|| \leq \sqrt{2A_1} + \sqrt{2A_0}$ **then**
  while: (not $converged$)
    $converged = true$
    Find by line-search $\kappa' \in [\kappa, \kappa_u]$ such that $h(\mathbf{w}, \kappa') = 0$
    $\mathbf{w}' = \arg\max_{||\mathbf{w}|| \leq 1} h(\mathbf{w}, \kappa)$
    if: $(|\kappa' - \kappa| > \epsilon_\kappa) \ \vee \ (h(\mathbf{w}', \kappa') > \epsilon_{\mathbf{w}})$
      then $converged = false$
    end if
    $\mathbf{w} \leftarrow \mathbf{w}'$, $\kappa \leftarrow \kappa'$
  end while
end if
$\mathbf{w}_* = \mathbf{w}$, $\kappa_* = \kappa$ and $b_* = \mathbf{w}_*^T \hat{\mathbf{x}}_1 - \sqrt{2A_1 + \kappa_*^2 \left(\mathbf{w}_*^T \hat{\Sigma}_1 \mathbf{w}_* + A_1 \right)}$

$\mathbf{w}_*$, where $||\mathbf{w}_*|| < 1$, can not be the optimal solution. To see this observe that

$$\begin{aligned}\sqrt{2A_j + \kappa_*^2 \left(\mathbf{w}'^T \hat{\Sigma}_j \mathbf{w}' + A_j \right)} &= \sqrt{2A_j + \kappa_*^2 \left(\frac{1}{||\mathbf{w}_*||^2} \mathbf{w}_*^T \hat{\Sigma}_j \mathbf{w}_* + A_j \right)} \\ &< \frac{1}{||\mathbf{w}_*||} \sqrt{2A_j + \kappa_*^2 \left(\mathbf{w}_*^T \hat{\Sigma}_j \mathbf{w}_* + A_j \right)}.\end{aligned}$$

Using this inequality in the auxiliary function $h(\mathbf{w}', \kappa_*)$ we see that

$$h(\mathbf{w}', \kappa_*) > \frac{1}{||\mathbf{w}_*||} h(\mathbf{w}_*, \kappa_*),$$

where we know by the monotonicity of $h(\mathbf{w}, \kappa)$ with respect to κ, that there exists $\kappa' > \kappa_*$, satisfying the constraints in (3.15). Therefore $(\mathbf{w}_*, \kappa_*)$ can not be the optimal solution to this problem.

Concavity of h in $\mathbf{w}$

At first glance it may not be obvious that the function $h(\mathbf{w}, \kappa)$ is concave with respect to $\mathbf{w}$, however we will now show that it is indeed the case. During the $\mathbf{w}$-step, we are maximising the function $h(\mathbf{w}, \kappa)$ subject to $\mathbf{w}$ residing in the unit-ball. Therefore in order to be sure we have obtained the maximum over the line search, we require that our function be concave. In order to check the concavity of the function h the Hessian of $h(\mathbf{w})$ must be negative

semi-definite. Recall that the derivative can be written as

$$\nabla h(\mathbf{w}) = (\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0) - \kappa^2 \left(\frac{\hat{\Sigma}_1 \mathbf{w}}{\sqrt{2A_1 + \kappa^2(\mathbf{w}^T \hat{\Sigma}_1 \mathbf{w} + A_1)}} + \frac{\hat{\Sigma}_0 \mathbf{w}}{\sqrt{2A_0 + \kappa^2(\mathbf{w}^T \hat{\Sigma}_0 \mathbf{w} + A_0)}} \right).$$

To help with presentation we will break the Hessian down into class-wise parts and show that it is indeed negative semi-definite for each class-component, and is thus negative semi-definite in full. Dropping subscript notation, the contribution of the class-component to the Hessian is

$$\nabla_{\mathbf{w}} \left(\frac{\hat{\Sigma}\mathbf{w}}{\sqrt{2A + \kappa^2(\mathbf{w}^T \hat{\Sigma} \mathbf{w} + A)}} \right) = \frac{(2A + \kappa^2(\mathbf{w}^T \hat{\Sigma}\mathbf{w} + A))\hat{\Sigma} - \kappa^2 \hat{\Sigma}\mathbf{w}\mathbf{w}^T\hat{\Sigma}}{\left(\sqrt{2A + \kappa^2(\mathbf{w}^T \hat{\Sigma}\mathbf{w} + A)}\right)^3}.$$

We know that the denominator is always positive, therefore we need to check whether the matrix created on the numerator is positive semi-definite, since we are taking the negative value of this in our Hessian of h. To see this we observe that

$$\mathbf{u}^T \left((2A + \kappa^2(\mathbf{w}^T \hat{\Sigma}\mathbf{w} + A))\hat{\Sigma} - \kappa^2 \hat{\Sigma}\mathbf{w}\mathbf{w}^T\hat{\Sigma} \right) \mathbf{u} \geq \kappa^2 \left((\mathbf{w}^T \hat{\Sigma}\mathbf{w})(\mathbf{u}^T \hat{\Sigma}\mathbf{u}) - (\mathbf{u}^T \hat{\Sigma}\mathbf{w})^2 \right) \geq 0,$$

which we have obtained by using the Cauchy-Schwartz inequality, and the positiveness of A. This proves that the Hessian of $h(\mathbf{w}, \kappa)$, with respect to $\mathbf{w}$, is negative semi-definite, which means that it is concave and therefore all local-maxima are global maxima.

3.3.3 Interpretation

In this section we discuss the interpretation the HP-MPM solution and how it compares to the original MPM formulation. We begin by examining the uncertainty term A_j associated with the empirical moments of each class. Clearly as the number of observations decrease we become less confident in the empirical moment estimates and the value of uncertainty term grows. At optimality the bias term $b_* \in \mathbb{R}$ is given by

$$b_* = \mathbf{w}_*^T \hat{\boldsymbol{\mu}}_1 - \sqrt{2A_1 + \kappa_*^2(\mathbf{w}_*^T \hat{\Sigma}_1 \mathbf{w}_* + A_1)} = \mathbf{w}_*^T \hat{\boldsymbol{\mu}}_0 + \sqrt{2A_0 + \kappa_*^2(\mathbf{w}_*^T \hat{\Sigma}_0 \mathbf{w}_* + A_0)}.$$

One consequence of increasing the value of the uncertainty A_1 is that we expect the hyperplane to shift in the direction of $\hat{\boldsymbol{\mu}}_0$. This is caused by the decrease in the bias term since larger values of A_1 fall under the negative square root term. The HP-MPM predicts that a new observation $\mathbf{x}$ belongs to class 1 when $\mathbf{w}_*^T \mathbf{x} \geq b$. Following the movement of the hyperplane, we now expect that the classifier is more likely to predict that a new observation belongs to class 1. Therefore we see the HP-MPM is naturally more inclined to position the hyperplane closer to the class that we are more confident about its empirical estimates.

In the original robust formulation, the uncertainty of the covariance matrices was captured by adding a regularisation term to the diagonal of these matrices. We see that because of the constraint $||\mathbf{w}|| = 1$, the HP-MPM approach implicitly adds a regularisation term to the covariance matrices, where the magnitude is equal to the uncertainty term for that class i.e.

$$\mathbf{w}^T\hat{\Sigma}_j\mathbf{w} + A_j = \mathbf{w}^T\left(\hat{\Sigma}_k + I_n A_j\right)\mathbf{w} = \mathbf{w}^T\tilde{\Sigma}_j\mathbf{w}$$

Therefore an increase in the value of A_1 results in a larger amount of regularisation on the covariance matrix $\hat{\Sigma}_1$. Returning to the geometric interpretation, this results in an increasingly spherical in nature ellipsoid centred at $\hat{\boldsymbol{\mu}}_1$. This in turn causes the optimal projection vector to rotate towards a solution dominated by the covariance of class 0, more precisely it rotates in the direction of $\hat{\Sigma}_0^{-1}(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0)$. Intuitively we can explain this by considering the situation where the uncertainty term A_1 dominates the covariance matrix $\hat{\Sigma}_1$, therefore small changes in the orientation of $\mathbf{w}$ will have very little influence on the value taken on by $\mathbf{w}^T\tilde{\Sigma}_1\mathbf{w}$ and therefore $h(\mathbf{w}, \kappa)$. Therefore the orientation of $\mathbf{w}$ will be more sensitive to the shape of $\hat{\Sigma}_0$, which will cause our optimal solution to rotate in the direction of $\hat{\Sigma}_0^{-1}(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0)$ i.e. the solution becomes increasingly dominated by the shape of the other class

The original MPM fails to take into consideration the fact that empirical moments may have been computed using only a small number of samples, these moment estimates are likely to be erroneous and the MPM approach is therefore subject to providing unrealistic estimates on the future misclassification rates. By taking into consideration the errors in the empirical moments we can provided a truer estimate of worst-case future misclassification errors. Furthermore by taking into consideration the relative number of observations for each class, and therefore the expected accuracy of the empirical moments, we are better able to position the hyperplane in a position where we can be more confident about the likelihood of future observations falling on either side.

We can view the original MPM as looking for the point of intersection between two ellipsoids centered at the class means, where the shape of the ellipsoids are determined by the covariance matrices and their size is controlled by the value of κ i.e. for $j = 0, 1$

$$\mathcal{E}_j(\kappa) = \left\{\mathbf{x} = \bar{\mathbf{x}}_j + \Sigma_j^{1/2}\mathbf{u} \;:\; ||\mathbf{u}|| \leq \kappa\right\}$$

Clearly as the size of κ increases these ellipsoids will eventually overlap. However the optimal hyperplane is given by the common tangent to the ellipsoids at the first point of their tangency. During our optimisation scheme, we alternate between allowing these ellipsoids, albeit a penalised version of them, to intersect i.e. $h(\mathbf{w}, \kappa) = 0$, and rotating $\mathbf{w}$ to provide

additional space for the ellipsoids to expand into at the next stage of the optimisation. We can view the moment uncertainty as introducing a regularisation component to the covariance matrices, along with a penalty regarding the location of the means. The regularised ellipsoids we consider in the high-probability setting are given by

$$\hat{\mathcal{E}}_j(\kappa) = \left\{ \mathbf{x} = \hat{\mathbf{x}}_j + \tilde{\Sigma}(\kappa)_j^{1/2}\mathbf{u} : \ ||\mathbf{u}|| \leq \kappa, \ \tilde{\Sigma}(\kappa)_j = \hat{\Sigma}_j + I_d \left(A_j + \frac{2A_j}{\kappa^2} \right) \right\}. \tag{3.16}$$

Using the geometrical interpretation, we see that as the value of κ grows, the effective regularisation on the covariance matrix decreases. This results from a relative reduction in the role played by the mean uncertainty in the square root term. Intuitively, as we move away from the means, with increasing values of κ, the point of origin becomes less important and we focus more on the underlying shape of the ellipsoid. In Figure 3.3 we show how the ellipsoids change as we increase κ up until their point of tangency. The intermediate solutions where the auxiliary function $h(\mathbf{w}, \kappa) = 0$, represent hyerplanes that are tangential to the ellipsoids but where the ellipsoids are not tangential to one another.

3.3.4 Kernel Formulation

So far we have explored the notion of finding an optimal linear decision boundary. Geometrically we saw that the worst-case bound on the future misclassification rate depends on both the distance between the class means, and the shape of the ellipsoids that their covariance matrices determine. However, it is often the case that by mapping the inputs into some higher-dimensional feature space there is a greater degree of separation between the two classes, and thus we should be able to reduce the worst-case future misclassification rate. Kernel methods [Shawe-Taylor and Cristianini, 2004; Vapnik, 1998] are able to take advantage of these higher-dimensional feature spaces without having to explicitly compute them, and have proven a useful tool for many classification algorithms. All of the methods that we have discussed thus far can make use of the kernel-trick. It was first shown to apply to Fisher's discriminant in [Mika et al., 1999], here the authors showed that by considering moments of the class-conditional densities in high-dimensional feature spaces, the kernelised version of FDA (KFDA) offered performance that was competitive with state-of-the art results. This resulted in a resurgence of interest in the use of FDA for classification; [Mika et al., 2003] examined the relationships with unsupervised techniques such as principal component analysis (PCA), [Mika et al., 2001a,b] sought efficient implementation procedures for the KFDA and [Mika, 2002] provided thorough, in-depth analysis of KFDA. The kernelisation of the MPM algorithm was presented in the original work [Lanckriet et al., 2003], and we closely follow their derivation to present the HP-MPM in its kernelised form.

The goal is to map the original input data to some higher dimensional feature space where

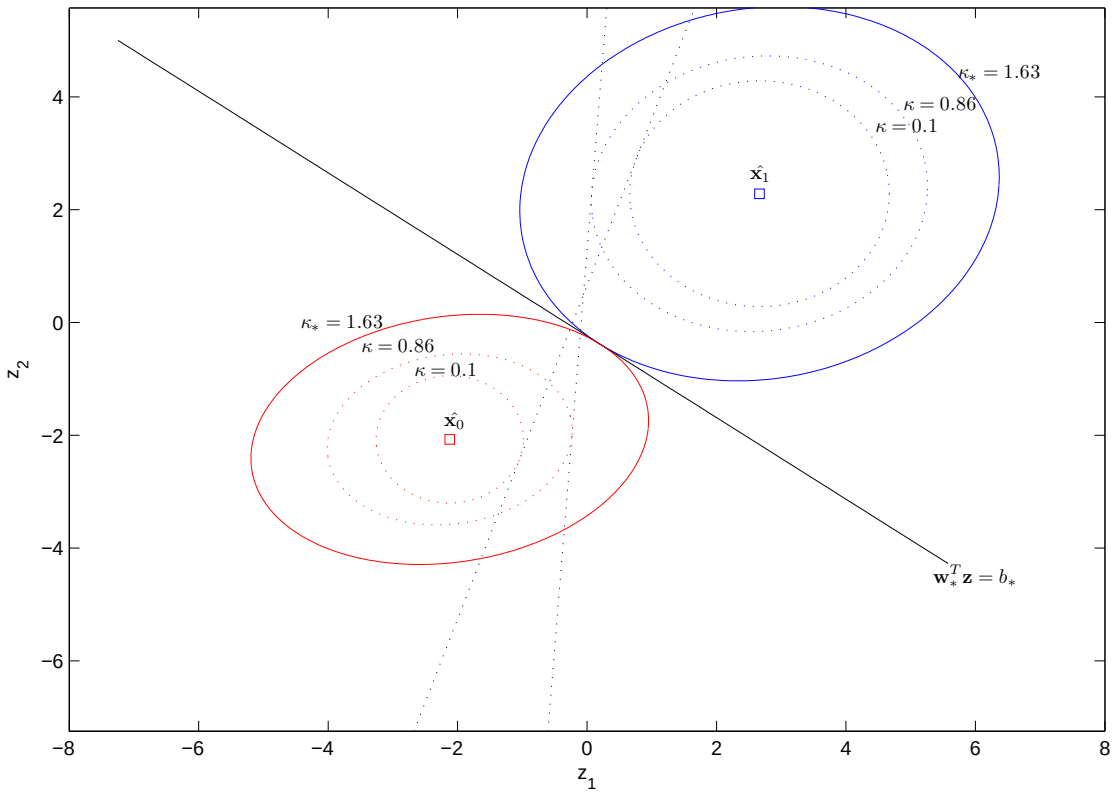


Fig. 3.3 Geometric interpretation of the high-probability MPM and the intermediate solutions produced during the optimisation scheme. We can see that in the beginning, for small values of κ, the penalised (regularised) covariance matrices are almost spherical. As the value of κ increases, and we move away from the class means, the ellipsoids begin to take on a shape increasingly determined by the sampled covariance matrix, however there still remains the regularisation caused by the uncertainty in the value of the covariance matrix. We see that the intermediate solutions $h(\mathbf{w}, \kappa) = 0$ result in hyperplanes that are tangential to the each classes ellipsoid, however these ellipsoids are only tangential to one another at the optimal solution. In the samples used to generate this solution, $m_1 = 20$ and $m_0 = 200$, explaining the larger size of the ellipsoid for class 1.

the data is better separated, allowing for a larger value of ω and thus obtaining a lower value on the probability of future misclassification. To do this we introduce the feature mapping $\phi : \mathcal{X} \to \mathcal{F}$ such that the linear decision boundary in this space is given by hyperplane $\mathcal{H}(\mathbf{w}, b) = \{\phi(\mathbf{x}) \in \mathcal{F} \ : \ \mathbf{w}^T\phi(\mathbf{x}) = b\}$. Note that this linear decision boundary in feature space corresponds to a non-linear decision boundary in $\mathcal{X}$. The data is mapped according to

$$\begin{aligned}\mathbf{x}_1 &\to \phi(\mathbf{x}_1) \sim \mathcal{D}_1^\phi \\ \mathbf{x}_0 &\to \phi(\mathbf{x}_0) \sim \mathcal{D}_0^\phi,\end{aligned}$$

where $\mathcal{D}_j^\phi$ for $j = 0, 1$ are the set of distributions with mean $\boldsymbol{\mu}_j^\phi$ and covariance Σ_j^ϕ. To find the optimal hyperplane in $\mathcal{F}$ we follow the same optimisation problem given in (3.15), where we substitute the original empirical moments with their feature space counterparts $\hat{\boldsymbol{\mu}}_j^\phi$ and $\hat{\Sigma}_j^\phi$ for each class $j = 0, 1$. We also have to take into consideration the influence that the mapping to the feature space has on the span of the data as this appears in the uncertainty terms A_j for each class j, however we will omit the superscript on A_j to help improve the clarity of the presentation. In order to make use of the kernel-trick, we have to show that the feature mappings enter the optimisation scheme only in terms of their inner-product $\langle\phi(\mathbf{x}), \phi(\mathbf{x}')\rangle = K(\mathbf{x}, \mathbf{x}')$, where $K : \mathcal{F} \times \mathcal{F}$ is the kernel function corresponding to the feature mapping ϕ. This allows the use of high-dimensional feature spaces without having to explicitly compute them thus making them tractable to work with.

To do this, first we have to show that any optimal solution to (3.15) must lie in the space spanned by the input data. To prove this, suppose the optimal solution is given by $\mathbf{w}_* = \mathbf{w}_s + \mathbf{w}_o$, where $\mathbf{w}_s$ is the projection of $\mathbf{w}$ onto the span of the input data and $\mathbf{w}_o$ is orthogonal to the space spanned by the input data. We show that the value of $\mathbf{w}_o$ plays no role in our ability to satisfy the first constraint in (3.15), however it does play a part in the unit L_2-norm restriction $||\mathbf{w}|| \leq 1$. Therefore if we removed this orthogonal component and scaled our $\mathbf{w}_s$ so that it resided on the unit-ball we will show that it results in an increase in the auxiliary function, thus permitting an increase in the value of κ in the next round of the optimisation scheme. Therefore a solution containing an orthogonal component can never be optimal.

The empirical means and covariances are linear combinations of the input data, and it is straightforward to show that

$$\begin{aligned}\mathbf{w}^T(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0) &= \mathbf{w}_s^T(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0) \\ \mathbf{w}^T\hat{\Sigma}_j\mathbf{w} &= \mathbf{w}_s^T\hat{\Sigma}_j\mathbf{w}_s.\end{aligned}$$

Therefore the value of the auxiliary function evaluated at $\mathbf{w}$ and $\mathbf{w}_s$ are the same i.e.

$h(\mathbf{w}_*, \kappa) = h(\mathbf{w}_s, \kappa)$. Under the assumption that the covariance matrices are positive semi-definite, we show that the auxiliary function increases with scale i.e. $h(t\mathbf{w}, \kappa) \geq h(\mathbf{w}, \kappa)$ for all $t \geq 1$ and that the optimal solution must reside on the unit-ball $||\mathbf{w}_*|| = 1$.

$$h(t\mathbf{w}, \kappa) = t\left(\mathbf{w}^T(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0) - \sqrt{2\frac{A_1}{t^2} + \kappa^2\left(\mathbf{w}^T\hat{\Sigma}_1\mathbf{w} + \frac{A_1}{t^2}\right)} - \sqrt{2\frac{A_0}{t^2} + \kappa^2\left(\mathbf{w}^T\hat{\Sigma}_0\mathbf{w} + \frac{A_0}{t^2}\right)}\right)$$
$$\geq t\left(\mathbf{w}^T(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0) - \sqrt{2A_1 + \kappa^2\left(\mathbf{w}^T\hat{\Sigma}_1\mathbf{w} + A_1\right)} - \sqrt{2A_0 + \kappa^2\left(\mathbf{w}^T\hat{\Sigma}_0\mathbf{w} + A_0\right)}\right) = th(\mathbf{w}, \kappa).$$

Therefore by replacing $\mathbf{w} = \mathbf{w}_a + \mathbf{w}_o$ with $\mathbf{w}_a/||\mathbf{w}_a||$, where $||\mathbf{w}_a|| \leq 1$, we see that the value of our auxiliary function increases, thus permitting a larger value of κ at optimality. Therefore a solution containing a component orthogonal to the span of the input data can not be optimal and the optimal solution must be given by

$$\mathbf{w}_* = \sum_{i=1}^{m} \alpha_i \mathbf{x}_i,$$

where $\alpha_i \in \mathbb{R}$ for all $i = 1, \ldots, m$. To take full advantage of the kernel-trick, and avoid having to explicitly evaluate the feature mappings, we now have to show that the feature mappings appear in the optimisation problem only as inner-products.

The kernel matrix $\mathbf{K}$ is given by $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ for all $i, j = 1, \ldots, m$. The first m_1 rows and last m_0 rows of $\mathbf{K}$ are denoted $\mathbf{K}_1$ and $\mathbf{K}_0$, respectively:

$$\mathbf{K} = \begin{pmatrix} \mathbf{K}_1 \\ \mathbf{K}_0 \end{pmatrix}.$$

The class row averages, $\mathbf{l}_1^T$ and $\mathbf{l}_0^T$, are m-dimensional vectors given by

$$\left(\mathbf{l}_1^T\right)_i = \frac{1}{m_1}\sum_{j=1}^{m_1} K(\mathbf{x}_j, \mathbf{x}_i) \text{ and } \left(\mathbf{l}_0^T\right)_i = \frac{1}{m_0}\sum_{j=m_1+1}^{m} K(\mathbf{x}_j, \mathbf{x}_i).$$

We create the block-row-averaged kernel matrix $\mathbf{L}$ by setting the row average of $\mathbf{K}_1$ and $\mathbf{K}_0$ equal to zero by:

$$\mathbf{L} = \begin{pmatrix} \mathbf{K}_1 - \mathbf{1}_{m_1}\mathbf{l}_1^T \\ \mathbf{K}_0 - \mathbf{1}_{m_0}\mathbf{l}_0^T \end{pmatrix} = \begin{pmatrix} \sqrt{m_1}\,\mathbf{L}_1 \\ \sqrt{m_0}\,\mathbf{L}_0 \end{pmatrix},$$

where $\mathbf{1}_m$ is a column vector of ones of dimension m. The empirical moment estimates in

the feature space are given by

$$\hat{\boldsymbol{\mu}}_1^\phi = \frac{1}{m_1}\sum_{i=1}^{m_1}\phi(\mathbf{x}_i) \text{ and } \hat{\Sigma}_1^\phi = \frac{1}{m_1}\sum_{i=1}^{m_1}\left(\phi(\mathbf{x}_i)-\hat{\boldsymbol{\mu}}_1^\phi\right)\left(\phi(\mathbf{x}_i)-\hat{\boldsymbol{\mu}}_1^\phi\right)^T$$
$$\hat{\boldsymbol{\mu}}_0^\phi = \frac{1}{m_0}\sum_{i=m_1+1}^{m}\phi(\mathbf{x}_i) \text{ and } \hat{\Sigma}_0^\phi = \frac{1}{m_0}\sum_{i=m_1+1}^{m}\left(\phi(\mathbf{x}_i)-\hat{\boldsymbol{\mu}}_0^\phi\right)\left(\phi(\mathbf{x}_i)-\hat{\boldsymbol{\mu}}_0^\phi\right)^T$$

The solution is given by $\mathbf{w} = \sum_{i=1}^m \alpha_i\phi(\mathbf{x}_i)$, we see that the components of the optimisation problem become

$$\mathbf{w}^T(\hat{\boldsymbol{\mu}}_1^\phi - \hat{\boldsymbol{\mu}}_0^\phi) = \boldsymbol{\alpha}^T(\mathbf{l}_1-\mathbf{l}_0), \quad \mathbf{w}^T\hat{\Sigma}_1^\phi\mathbf{w} = \boldsymbol{\alpha}^T\mathbf{L}_1^T\mathbf{L}_1\boldsymbol{\alpha} \quad \text{and} \quad \mathbf{w}^T\hat{\Sigma}_0^\phi\mathbf{w} = \boldsymbol{\alpha}^T\mathbf{L}_0^T\mathbf{L}_0\boldsymbol{\alpha}.$$

We saw earlier that the solution is given by $\mathbf{w} = \sum_{i=1}^m \alpha_i\phi(\mathbf{x}_i)$, and therefore the components of the optimisation become

$$\mathbf{w}^T(\hat{\phi}_1 - \hat{\phi}_0) = \boldsymbol{\alpha}^T(\mathbf{l}_1-\mathbf{l}_0), \ \mathbf{w}^T\hat{\Sigma}_1^\phi\mathbf{w} = \boldsymbol{\alpha}^T\mathbf{L}_1^T\mathbf{L}_1\boldsymbol{\alpha} \ \text{ and } \ \mathbf{w}^T\hat{\Sigma}_0^\phi\mathbf{w} = \boldsymbol{\alpha}^T\mathbf{L}_0^T\mathbf{L}_0\boldsymbol{\alpha}.$$

This allows us to write the kernelised version of the HP-MPM as

$$\max_{\boldsymbol{\alpha},\kappa}\ \kappa \text{ s.t. } ||\mathbf{w}||^2 = \boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha} \le 1$$
$$\boldsymbol{\alpha}^T(\mathbf{l}_1-\mathbf{l}_0) = \sqrt{2A_1+\kappa^2\left(\boldsymbol{\alpha}^T\mathbf{L}_1^T\mathbf{L}_1\boldsymbol{\alpha}+A_1\right)} + \sqrt{2A_0+\kappa^2\left(\boldsymbol{\alpha}^T\mathbf{L}_0^T\mathbf{L}_0\boldsymbol{\alpha}+A_1\right)}.$$

The same alternating optimisation procedure can be used to find the optimal values κ_* and $\boldsymbol{\alpha}_*$, and the optimal value of the bias term is given by

$$b_* = \boldsymbol{\alpha}_*^T\mathbf{l}_1 - \sqrt{2A_1+\kappa_*^2\left(\boldsymbol{\alpha}_*^T\mathbf{L}_1^T\mathbf{L}_1\boldsymbol{\alpha}_*+A_1\right)} = \boldsymbol{\alpha}_*^T\mathbf{l}_0 + \sqrt{2A_0+\kappa_*^2\left(\boldsymbol{\alpha}_*^T\mathbf{L}_0^T\mathbf{L}_0\boldsymbol{\alpha}_*+A_0\right)}$$

As with the linear case, when presented with a new input observation $\mathbf{x}'$, we predict that $y' = 1$ if $\boldsymbol{\alpha}_*^T\mathbf{k}_{\mathbf{x}'} - b_* \ge 0$, where $(\mathbf{k}_{\mathbf{x}'})_i = k(\mathbf{x}_i,\mathbf{x}')$, and $y' = 0$ otherwise. We should point out that we have no reason to expect that the solution $\boldsymbol{\alpha}_*$ will be sparse i.e. many $(\alpha_*)_i = 0$, and therefore the computational cost at prediction will be linear in the size of the training sample.

3.4 Experiments

In this section we examine the performance of the proposed HP-MPM and compare it to the original MPM, and two other popular binary classification algorithms, Fisher's discriminant (FDA) and the support vector machine (SVM). In Table 3.1 we provide a summary of the datasets taken from the UCI repository, `http://archive.ics.uci.edu/ml/`, and the toy dataset used in [Lanckriet et al., 2003], that we have used in our experiments. We have

included details regarding the number of observations, the dimension of the input space and the relative class frequencies to help support our argument regarding the importance of including information regarding the moment uncertainty into the derivation of the predictor.

All of the datasets were normalised so that each feature had zero mean and unit variance. To handle missing values, as in the *vote* dataset, we simply computed the means and standard deviations of each feature using the available data, performed standard normalisation on them and then set the values of the missing data to zero post-normalisation.Each dataset was randomly partitioned 50 times into training, validation and test samples, and we report the average performance over all test samples. During the experiments we varied the size of the training sample between 10% and 70%, in increments of 10%, of the full dataset to investigate how the algorithms performed with various amounts of information. The size of the validation set was fixed at 20% and the remaining data was used for testing. The goal of these experiments was to evaluate the benefits of considering moment uncertainty in the construction of the predictor, and to understand the relative gains that its inclusion have as we change the number of training points.

Table 3.1 Overview of the UCI datasets used during the experiments.

Dataset	Observations	Features	Class 1
ADULT	48844	123	23.93
AUSTRALIAN	690	14	44.49
BCI	400	117	50.00
BREAST	682	10	64.96
DIABETES	768	8	65.10
DIGIT1	1500	241	48.93
GERMAN	1000	24	70.00
HEART	920	13	44.67
IONOSPHERE	351	32	64.10
RINGNORM	7400	20	49.51
SONAR	208	60	53.37
SPLICE	3175	60	51.91
TOY	120	2	50.00
TWONORM	7400	20	50.04
VOTE	435	16	38.62

One of the main motivations of the formulation of the HP-MPM was to correct for overly confident estimates on the worst-case future misclassification rate. This was done through the introduction of high-probability bounds on the deviation of the true moments from the empirical counterparts, where we set the value of $\delta = 0.05$. However, we noticed that during the experiments that these high-probability bounds appeared to be too restrictive in many

Table 3.2 Linear experiments: we show how the performance of the classification algorithms on the datasets vary as the amount of data used during training changes. The best performing results for each dataset and training proportion are reported in **bold** typeface.

		Training proportion						
		0.1	0.2	0.3	0.4	0.5	0.6	0.7
AUSTRALIAN	MPM	84.21	85.88	85.89	86.08	86.04	86.25	86.39
	HP-MPM	**85.01**	**86.29**	**86.23**	**86.18**	**86.43**	86.31	86.55
	SVM	84.81	86.15	86.07	85.80	85.63	85.87	86.02
	FDA	83.85	85.66	85.92	86.13	86.20	**86.60**	**86.87**
BCI	MPM	58.30	62.25	56.55	70.81	76.49	79.74	81.55
	HP-MPM	**61.70**	**66.84**	**71.50**	**77.30**	**80.18**	**82.33**	**82.96**
	SVM	60.39	66.25	70.83	74.17	75.90	78.82	79.41
	FDA	58.66	57.72	53.80	57.12	60.88	63.97	68.03
BREAST	MPM	96.20	97.04	97.10	97.22	97.22	**97.33**	**97.23**
	HP-MPM	**97.12**	**97.10**	**97.18**	**97.24**	**97.24**	97.29	97.22
	SVM	96.58	96.77	96.85	96.97	97.05	97.07	97.03
	FDA	92.54	93.76	94.50	94.50	94.50	94.58	94.46
DIABETES	MPM	72.74	74.12	75.02	75.11	74.99	74.97	74.86
	HP-MPM	73.14	73.86	74.76	74.75	74.41	74.49	74.53
	SVM	**74.40**	**75.97**	**76.30**	76.29	76.48	76.55	76.74
	FDA	73.97	75.32	75.94	**76.61**	**76.51**	**76.68**	**76.91**
DIGIT1	MPM	73.47	75.08	85.92	89.53	91.38	92.16	92.90
	HP-MPM	**92.99**	**93.80**	**94.30**	94.53	94.63	94.62	94.60
	SVM	91.66	93.38	94.20	**94.69**	**95.10**	**95.25**	**95.70**
	FDA	80.09	75.77	85.82	89.35	91.42	92.17	92.83
GERMAN	MPM	68.96	70.84	71.55	72.08	72.24	72.49	72.70
	HP-MPM	69.54	71.26	71.86	72.34	72.26	72.50	72.79
	SVM	**71.66**	**73.82**	**74.55**	**74.99**	75.51	75.82	76.03
	FDA	71.50	73.64	74.45	74.85	**75.58**	**76.10**	**76.47**
HEART	MPM	78.02	79.54	80.20	80.72	81.33	**81.84**	**82.21**
	HP-MPM	**79.36**	**80.15**	**80.62**	80.87	**81.34**	81.56	81.96
	SVM	78.77	79.62	80.36	**80.98**	81.29	81.69	81.97
	FDA	77.47	79.33	80.01	80.61	81.01	81.64	81.99
IONOSPHERE	MPM	72.45	78.73	80.49	81.21	81.70	82.29	82.62
	HP-MPM	**82.18**	**82.93**	83.35	82.88	83.18	83.39	83.11
	SVM	80.68	82.91	**83.51**	**83.83**	**84.18**	**84.64**	**84.19**
	FDA	68.60	74.09	76.41	78.04	79.63	80.06	80.38
RINGNORM	MPM	74.43	74.70	74.88	74.93	74.94	74.94	74.86
	HP-MPM	76.51	76.71	76.81	76.80	76.86	76.91	76.79
	SVM	**76.55**	**76.88**	**76.88**	**77.05**	**77.03**	77.09	77.06
	FDA	76.19	76.63	76.74	76.82	76.91	**77.12**	**77.07**
SONAR	MPM	63.59	61.80	57.42	67.61	70.36	71.76	75.47
	HP-MPM	**69.88**	**73.84**	**74.71**	74.93	**76.71**	**76.01**	**77.41**
	SVM	67.51	72.95	74.08	**75.31**	75.84	75.32	77.20
	FDA	62.37	59.88	55.11	64.49	68.28	70.44	73.73
SPLICE	MPM	81.52	83.21	83.83	**84.23**	84.36	84.55	**84.89**
	HP-MPM	**82.15**	**83.31**	**83.87**	84.20	84.35	**84.67**	84.80
	SVM	81.74	82.96	83.55	84.14	**84.37**	84.40	84.84
	FDA	81.12	82.98	83.68	84.11	84.25	84.56	84.71
TOY	MPM	89.92	93.17	94.00	94.41	**94.36**	94.45	94.50
	HP-MPM	**91.16**	**93.32**	**94.22**	**94.57**	94.32	**94.50**	**94.89**
	SVM	90.42	93.13	93.75	94.24	93.50	93.09	93.38
	FDA	86.12	90.04	91.10	93.18	92.71	93.18	93.62
TWONORM	MPM	97.59	97.65	97.68	97.68	97.74	97.76	97.80
	HP-MPM	**97.67**	**97.69**	**97.71**	97.70	97.75	97.76	97.82
	SVM	97.53	97.60	97.67	**97.72**	**97.75**	**97.77**	**97.84**
	FDA	97.53	97.63	97.67	97.67	97.73	97.74	97.81
VOTE	MPM	92.86	95.28	95.75	95.71	95.74	95.95	96.03
	HP-MPM	**94.95**	95.42	95.58	95.53	95.37	95.51	95.59
	SVM	94.47	95.20	94.99	95.35	95.26	95.41	95.81
	FDA	93.37	**95.51**	**95.97**	**96.00**	**96.17**	**96.40**	**96.19**

settings and we were unable to generate meaningful solutions i.e. $\omega_* = 0$. To overcome this deficiency we propose to use the moment uncertainty terms as a form of regularisation, and during the experiments we use a validation procedure to choose what fraction of the true moment uncertainty we should use. More precisely, rather than using A_j we used some fractional amount $\hat{A}_j = \nu A_j$ of the full uncertainty, where $\nu \in \{0.05, 0.1, 0.15, 0.2, 0.3, 0.5, 1\}$. For the parameter selection process, in each training and test sample we had a distinct validation set that was used to evaluate the performance of the predictor generated for the particular regularisation parameter. For each of these training, validation and test sets we evaluated the performance of the predictor (parameter) with the best validation set accuracy on the test sample. The same method to choose the regularisation parameter for the SVM and FDA , where SVM's *capacity* parameter was selected from $C \in \{10^{-3}, \ldots, 10^{3}\}$, and the FDA's regularisation term chosen from $\lambda \in \{10^{-3}, \ldots, 10^{3}\}$. The MPM optimisation solves a series of least-squares sub-problems and we used cross validation to choose the regularisation parameter from $\{10^{-6}, \ldots, 10^{0}\}$ in these sub-problems.

Linear experiments

In Table 3.2 we examine the performance of the linear based classification algorithms and show how their performance varies as we change the size of the training sample used to construct the predictor. As one would expect, in general the performance on the test samples improves as more training examples are presented to the algorithm during training. However in the SONAR dataset we see a drop in the performance of the MPM and FDA predictors as we increase the fraction of the dataset used in training from 0.1 to 0.3. This can be explained by the relatively small number of observations that were used to construct the empirical moments, which determine each algorithms decision boundary. On the other hand we see that the HP-MPM and the SVM are relatively robust to the use of small training samples, and we observe the benefits of the regularisation scheme implemented by the HP-MPM , and note the benefits of constructing the decision boundary using peripheral points, as advocated by the SVM, rather than poorly estimated empirical moments.

From Table 3.2 we can observe that when using minimal amounts of training data i.e. 10% of the full dataset, the HP-MPM method is nearly always the top performing algorithm. As the size of the training sample increases, the advantage of the HP-MPM begins to erode and its performance comes in line with the original MPM . This is to be expected in the case of large amounts of available data since we know that the HP-MPM will eventually converge towards the original MPM solution as moment uncertainty decreases to zero.

Kernel experiments

In Table 3.3 we present the performance of the kernelised version of the algorithms. Here

Table 3.3 Kernel experiments: we show how the performance of the classification algorithms on the datasets vary as the amount of data used during training changes. The best performing results for each dataset and training proportion are reported in **bold** typeface.

		Training proportion						
		0.1	0.2	0.3	0.4	0.5	0.6	0.7
AUSTRALIAN	MPM	83.71	85.67	85.71	85.93	86.06	86.12	86.23
	HP-MPM	**85.06**	**85.91**	**86.16**	**86.01**	**86.15**	**86.25**	**86.49**
	SVM	84.83	85.84	85.53	85.29	85.30	85.53	85.90
	FDA	79.55	81.68	82.22	82.22	81.99	82.21	83.28
BCI	MPM	59.35	62.71	**69.88**	**73.33**	**75.32**	**76.10**	**77.79**
	HP-MPM	59.14	**65.00**	69.01	72.23	73.90	74.21	74.89
	SVM	**59.73**	64.65	69.40	72.56	74.07	74.36	75.79
	FDA	52.43	53.50	54.41	56.09	58.47	58.62	61.05
BREAST	MPM	96.31	97.08	**97.19**	**97.23**	**97.20**	**97.17**	97.09
	HP-MPM	**97.07**	**97.18**	97.03	97.12	97.11	97.16	**97.09**
	SVM	96.52	96.76	96.89	96.84	96.91	96.84	96.67
	FDA	95.92	96.01	95.97	95.69	96.00	95.72	95.64
DIABETES	MPM	72.50	74.25	74.95	75.08	75.09	74.88	74.98
	HP-MPM	73.15	74.43	74.68	74.81	74.44	74.55	74.32
	SVM	**74.39**	**75.99**	**76.21**	**76.52**	**76.47**	**76.76**	**76.79**
	FDA	69.32	71.23	72.05	72.70	72.75	72.93	73.56
DIGIT1	MPM	90.23	92.57	94.52	95.66	96.21	96.19	96.68
	HP-MPM	**93.73**	**96.01**	96.84	**97.23**	97.46	97.43	97.58
	SVM	92.96	95.85	**96.88**	97.22	**97.51**	**97.62**	97.66
	FDA	91.84	94.65	95.94	96.72	97.21	97.34	**97.68**
GERMAN	MPM	69.99	71.24	71.75	72.06	72.37	72.35	72.49
	HP-MPM	70.63	71.47	72.07	72.37	72.43	72.65	72.99
	SVM	**71.62**	**73.80**	**74.51**	**74.90**	**75.25**	**75.68**	**76.43**
	FDA	69.97	70.15	69.97	69.87	69.89	70.17	70.15
HEART	MPM	78.07	79.66	**80.32**	80.78	81.14	**81.75**	**82.32**
	HP-MPM	**79.19**	**80.08**	80.14	80.79	80.89	81.58	81.71
	SVM	78.84	79.68	80.23	**80.82**	**81.15**	81.46	81.57
	FDA	76.82	77.91	78.01	78.45	78.90	79.04	78.99
IONOSPHERE	MPM	80.93	83.14	87.18	89.10	90.25	90.78	91.38
	HP-MPM	**91.04**	**93.38**	**93.89**	**94.15**	94.44	94.61	94.58
	SVM	86.76	92.88	93.82	94.00	**94.53**	**95.07**	**95.62**
	FDA	86.92	86.47	89.31	91.12	91.98	92.67	93.62
RINGNORM	MPM	97.76	97.82	97.84	97.85	97.83	97.84	97.83
	HP-MPM	**98.51**	98.52	98.53	98.51	98.50	98.51	98.49
	SVM	98.47	**98.56**	**98.56**	**98.57**	**98.55**	**98.54**	**98.55**
	FDA	96.89	95.68	95.65	95.64	95.61	95.59	95.59
SONAR	MPM	66.37	71.85	76.00	78.52	80.96	82.83	84.60
	HP-MPM	**70.09**	**76.37**	78.61	**81.34**	**84.00**	**84.68**	**86.87**
	SVM	68.90	75.88	**78.94**	81.02	83.52	84.54	86.53
	FDA	65.66	72.90	76.81	80.16	82.84	84.63	86.80
SPLICE	MPM	82.90	85.00	85.90	86.51	86.81	86.81	86.81
	HP-MPM	**85.19**	**87.87**	**89.23**	**89.95**	**90.40**	**90.40**	**90.40**
	SVM	84.70	87.54	88.87	89.82	90.26	90.26	90.26
	FDA	71.25	72.49	79.97	82.46	83.87	83.87	83.87
TOY	MPM	87.92	92.52	93.45	94.24	94.00	94.45	94.50
	HP-MPM	**91.00**	**93.57**	94.05	**94.53**	**94.71**	**94.55**	**95.25**
	SVM	90.77	93.35	**94.15**	94.18	94.29	94.00	94.75
	FDA	88.08	90.39	91.70	92.29	92.57	92.45	92.62
TWONORM	MPM	97.57	97.64	97.68	97.69	97.69	97.70	97.70
	HP-MPM	**97.72**	**97.74**	**97.76**	**97.77**	**97.77**	**97.77**	**97.76**
	SVM	97.69	97.72	97.73	97.71	97.69	97.72	97.70
	FDA	97.61	97.64	97.67	97.68	97.67	97.67	97.63
VOTE	MPM	92.40	94.95	95.70	95.72	95.76	**96.07**	**96.09**
	HP-MPM	**94.91**	**95.51**	**95.72**	**95.84**	**95.80**	95.88	96.00
	SVM	94.42	95.19	95.14	95.64	95.63	95.81	95.66
	FDA	92.68	94.01	94.60	94.26	94.31	94.14	93.53

we used the popular Gaussian kernel $k(\mathbf{x}, \mathbf{x}') = \exp(-||\mathbf{x} - \mathbf{x}'||^2/\sigma)$, where the width of the kernel $\sigma \in \{10^{-3}, \ldots, 10^3\}$ was chosen using the same validation scheme outlined earlier. We see that in general each algorithm's performance is similar to its performance in the linear setting, however there are noticeable improvements on the ionosphere, ringnorm and sonar datasets when using the Gaussian kernel. This suggests that these input spaces are better separated with a non-linear decision boundary, whereas for the others a simple linear decision boundary will suffice. In the kernelised form we see that the MPM approach to classification , MPM or HP-MPM , is extremely competitive with the SVM, being the top performing algorithm for a large proportion of the dataset/training set size combinations.

Table 3.4 GERMAN dataset: we evaluate the performance (classification accuracy) of selecting the bias term for the HP-MPM according to its performance on the validation set. We see that this simple approach to adjusting the decision boundary, represented in column bHP-MPM , improves the performance of the HP-MPM , correcting for its implicit assumption that classes are equally likely, and brings its performance inline with the SVM.

Fraction	MPM (%)	HP-MPM (%)	bHP-MPM (%)	SVM(%)	FDA (%)
0.1	69.99	70.63	**73.04**	71.62	69.97
0.2	71.24	71.47	**74.33**	73.80	70.15
0.3	71.75	72.07	**75.22**	74.51	69.97
0.4	72.06	72.37	**75.42**	74.90	69.87
0.5	72.37	72.43	**76.08**	75.25	69.89
0.6	72.35	72.65	**76.11**	75.68	70.17
0.7	72.49	72.99	**76.72**	76.43	70.15

It would appear as though the validation procedure used to determine the parameters for the kernelised form of FDA failed to ensure that increased training data resulted in an improvement in the performance. This could be due to inappropriate values of λ used during regularisation, however there is very little guidance in the literature on a suitable degree of regularisation, whereas the HP-MPM has a simple range $\nu \in [0, 1]$ from which to choose. Furthermore, it is straightforward to work out what maximum value of ν will result in $\kappa > 0$ i.e. the conditions for non-zero κ require $\nu \in [0, 1]$ to satisfy $||\hat{\mathbf{x}}_1 - \hat{\mathbf{x}}_0|| \geq \sqrt{2\nu A_1} + \sqrt{2\nu A_0}$.

Accounting for class imbalance

The MPM schemes are generally competitive with the other approaches, however they seem to perform comparatively poorly, some 3% worse than the SVM, on the GERMAN dataset. This weakness of the MPM was previously identified in [Huang et al., 2004], and is due to the MPM 's prior assumption that the prior probability of each class is the same. We know from Table 3.1 that this is not the case for the GERMAN dataset, and that the probability of belonging to class 1 is much higher than class 0. One could foresee the HP-MPM making

this situation potentially even worse given the nature in which in constructs its solution, and its natural bias towards placing the decision boundary closer to the mean of the class where moment uncertainty is lower i.e. the one with more observations. This is illustrated in Figure 3.3, where we see the hyperplane is positioned nearer to the mean of class 0 because the training sample consists of many more observations from this class. Fortunately this is not the case and we see that the HP-MPM 's performance is similar to that of the MPM . This is largely a result of the low levels of confidence in future performance i.e. small κ_*, which results in large levels of implicit regularisation for both classes as seen in the expression for the ellipsoids (3.16). This results in a decision boundary that is not overly biased towards predicting that new observations belong to the minority class. A simpler explanation of its similar performance can be given by the validation procedure that was used to determine what degree ν of regularisation to choose i.e. more likely that a smaller value of ν was used since it would place the decision boundary less close to the mean of the more common class, and therefore not be overly biased towards predicting that a new observation belongs to the less probable class.

To improve the performance of the HP-MPM on unbalanced training samples, we propose a simple solution that adjusts the bias term used in the construction of the decision boundary. During the training step we use the optimal weight vector $\mathbf{w}_*$ found using the standard HP-MPM algorithm, and then select the bias term to be the one that maximises the accuracy on the training set. These weight vectors and biases are then evaluated on the validation sample. Alternatively one could use the validation set to set the bias term, however this could be thought of as given this a glimpse of additional training samples and therefore an unfair advantage. Geometrically, this adjustment corresponds to a translational movement of the hyperplane where its direction $\mathbf{w}$ remains the same. In the GERMAN dataset, this corresponds to shifting the decision boundary in the direction of the mean of class 0, since we want to increase the probability that a new observation is predicted to belong to class 1. In Table 3.4 we show the results obtained on the GERMAN dataset by selecting the value of the bias term, when using the Gaussian kernel. We see that this simple approach to selecting the value of the bias term b, represented by column bHP-MPM in Table 3.4, improves the performance of the HP-MPM , correcting its implicit assumption that classes are equally likely, and brings its performance in line with the SVM. This would suggest that the direction $\mathbf{w}$ found by the HP-MPM is a useful method for discriminating between classes, and the bias term can be selected to take into consideration the relative class probabilities. However in doing so, the worst-case error rates that are found using the HP-MPM are no longer valid as we have repositioned the location of the separating hyperplane. In our conclusions we discuss a possible extension to the HP-MPM that similar to the MEMPM suggested in [Huang et al., 2004] can take into consideration the relative class frequencies in the construction of

the hyperplane.

Large dataset

We evaluate the performance of the proposed method on the relatively large ADULT dataset with the results presented in Table 3.5. This table reports the performance using the linear version of all proposed methods. We see that the bHP-MPM approach is the top performing approach up until 5,000 training examples are provided to the learning algorithm, after which the SVM becomes the top performing predictor. This supports our argument that in the case of limited data availability the incorporation of moment uncertainty can improve the performance of predictors. As the number of data points increases and our information of the class-conditional distribution improves, the worst-case assumptions and the regularisation imposed by the HP-MPM , may hinder the construction of predictions, whereas the SVM is able to take advantage of better knowledge of the true periphery of the class-conditional distributions. Note that these experiments were conducted using the primal form of the problem and we have avoided having to find a very large dimensional dual solution. We discuss possible methods for computing sparse solutions to our HP-MPM during our conclusions to this chapter.

Table 3.5 ADULT dataset lines experiment: we evaluate the performance (classification accuracy) of the proposed algorithm on a large scale dataset. The number of training samples m is varied and we observe the changes in classifier performance. We see that with a small number of training examples the bHP-MPM tends to outperform the other approaches, with its relative advantage deteriorating as m increases.

m	MPM (%)	HP-MPM (%)	bHP-MPM (%)	SVM(%)	FDA (%)
50	60.63	78.13	**79.35**	78.41	73.01
100	74.63	78.30	**81.37**	79.87	76.42
200	77.90	79.03	**82.37**	81.41	78.52
500	79.52	79.63	**83.30**	82.83	81.08
1000	80.08	80.07	**83.79**	83.53	82.61
5000	80.47	80.44	84.34	**84.46**	84.23
10000	80.56	80.52	84.49	**84.65**	84.43

Impact of label noise

A key motivation for this chapter was our belief that predictors constructed using the moments of a distribution would be more robust to high-noise environments than those constructed using peripheral points. For example, a small number of mislabelled examples could quite significantly adjust the shape of the covariance matrix when the number of examples is not large. To test this we re-visit the UCI datasets and examine the performance of the algorithms under different levels of label noise, varying from 0% to 30% in increments of 5%. Label noise in this scenario refers to the random switching of the class labelling i.e.

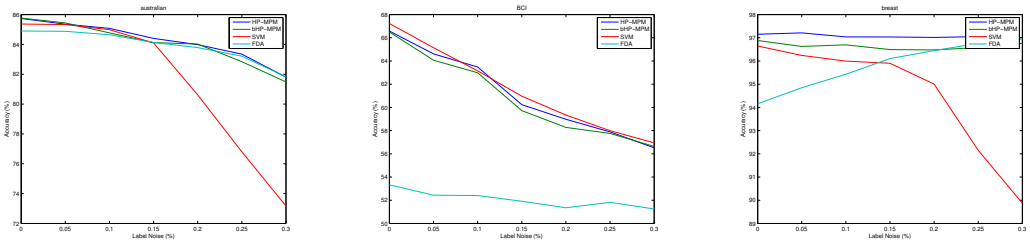
10% noise corresponds to a 10% chance that a given label will be switched. In all of the experiments we performed 50 iterations varying the levels noise, in Figures 3.4 and 3.5 we report the performance of the algorithms when we use 20% and 50%, respectively, of the data during training. We follow the same cross validation procedure as in our original experiments to select the required parameters for each algorithm.

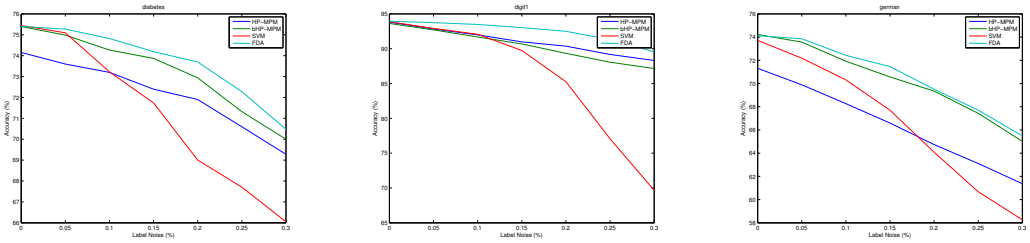
In Figures 3.4 and 3.5 we see that the performance of the HP-MPM is fairly robust to the introduction of noise in the label space. It performs competitively with the other algorithms, and for many noise levels performs the best. We observe that on the BREAST, DIGIT1, HEART, RINGWORM and TWONORM datasets, the performance of the HP-MPM algorithms degrades only marginally as the level of label noise increases. The behaviour of the FDA on these datasets except for BREAST is relatively similar, indicating the robustness of solutions constructed using the moments of the class-conditional densities. However on the BREAST dataset, we see that the performance of the FDA algorithm improves as we increase the level of label noise. One possible explanation for this is due to the value of the bias term b that depends on the relative probabilities of each class. By randomly flipping labels, it is likely that we reduce the class imbalances and this results in a better positioning of the FDA hyperplane. The same argument may be used to explain why the performance of the HP-MPM methods don't decrease for this dataset.

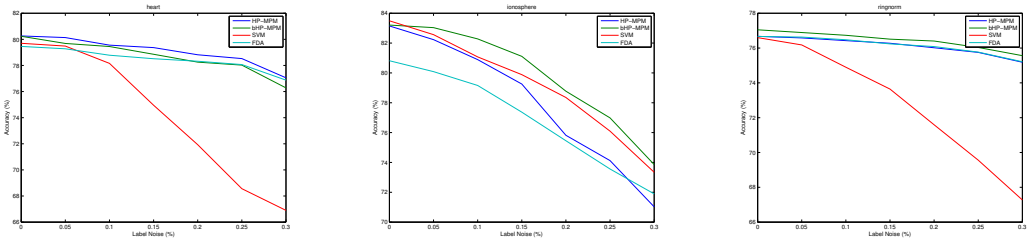
The performance of the SVM appears to be significantly more affected for large levels of label noise, and we see a sharp drop in the performance of the SVM once a certain level of noise has been reached. The reason for this degradation could be explained by the construction of the solution using outliers, which in this case happen to come from a different distribution and are therefore more misleading than discriminative. In the moment based classifiers, we would expect the label flipping to have a lesser influence on the predictor and this is evident in the results shown in Figures 3.4 and 3.5.

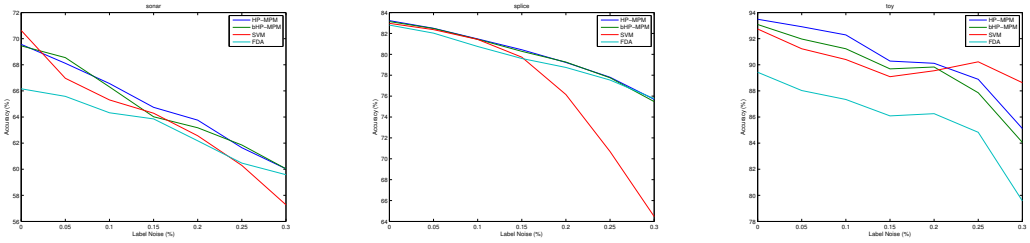
Currency movement prediction

We conclude our experiments by testing the performance of the different classification algorithms on predicting the daily price movement of four common currency pairs. The daily foreign exchange (FX) data was freely downloaded from `http://www.dukascopy.com`, and ranges from October 2008 to October 2014. The currency pairs that we investigated were; EUR-GBP, EUR-USD, EUR-GBP and AUD-USD. We now describe the classification setting used in the experiments. Let the opening price of the currency at day t be given by p_t, we represent the input space using a range of n-past log returns. For example, if $n = 3$, then









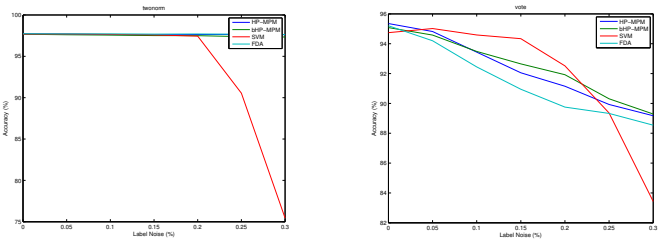


Fig. 3.4 The impact that flipping training labels has on the performance of the different classification algorithms, when training using 0.2 of the full dataset.

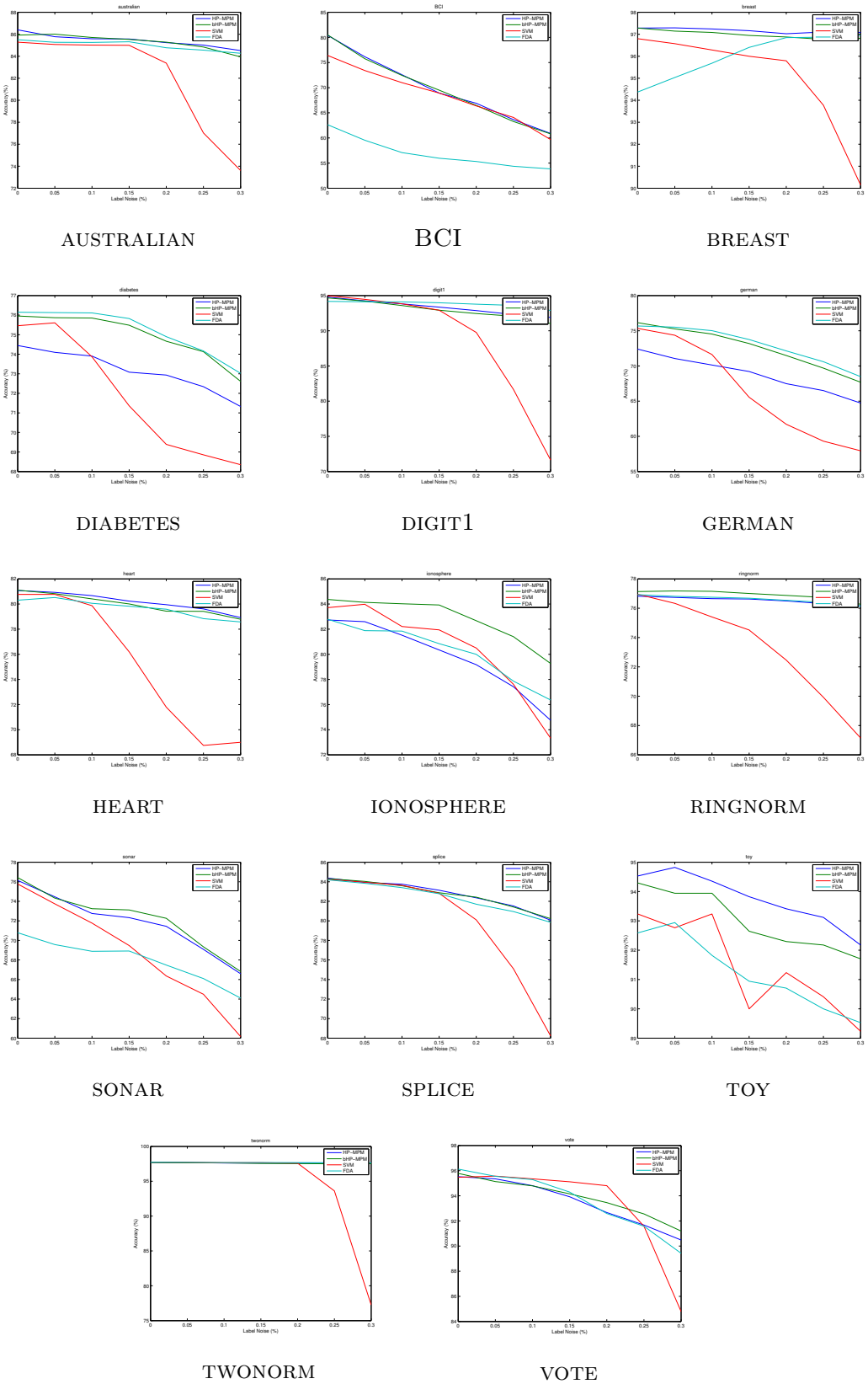


Fig. 3.5 The impact that flipping training labels has on the performance of the different classification algorithms, when training using 0.5 of the full dataset.

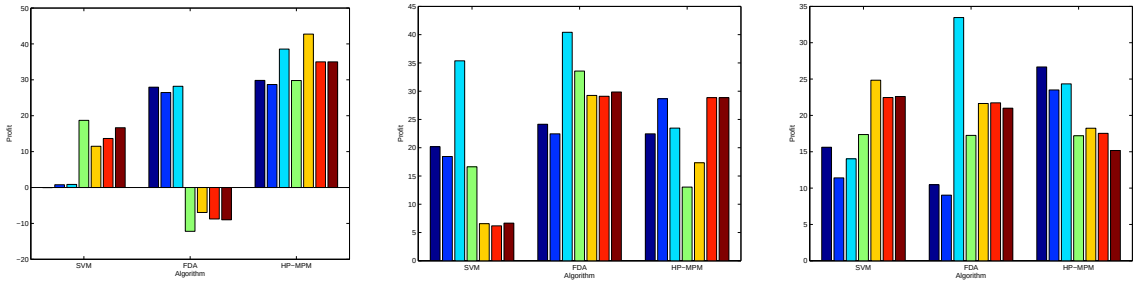
the input representation $\mathbf{x}_t$ at time t is given by

$$\mathbf{x}_t = [\log(p_t/p_{t-1}), \log(p_{t-1}/p_{t-2}), \log(p_{t-2}/p_{t-3})].$$

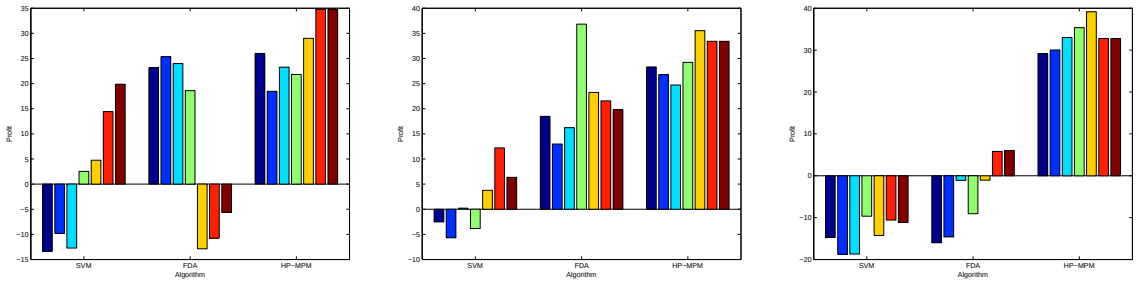
Given $\mathbf{x}_t$, our goal is to make a prediction whether we believe the price at the next time step will be higher than the current i.e. $y_t = 1$ if $p_{t+1} \geq p_t$, and $y_t = 0$ otherwise. In evaluating the performance of the algorithms we recorded not only the accuracy of the predictions, but also the hypothetical profit that would be made had we made a decision according to the advice of the predictor i.e. if we predicted the price to increase from t to $t+1$, then our return r_t would be the change in price over this time $r_t = (p_{t+1} - p_t)/p_t$. Similarly if we predicted the price would fall over this time horizon $r_t = (p_t - p_{t+1})/p_t$.

To train the model we implement a simple sliding window procedure that uses a fixed size number of examples (training window) to construct the predictor, which is then refreshed after a given number of observations (test window). By updating the predictor over time it is hoped that the predictor will be able to account for fact that the data is most likely not identically and independently distributed. Unfortunately we are unable to use the same validation technique that we used on the previous experiments, as it is likely that the most recent observations are the most important to the derivation of the predictor and we cannot make predictions based on observations in the future. Therefore in the results presented in Figures 3.6-3.7 we have shown the performance of all of regularisation parameters for each classification algorithm. Given the multitude of different settings for these experiments and the limited space, we present only the results obtained when predictor is refreshed every 10 days, the input space is described using the last 5 log returns and we allow the training window to vary between 50, 100 and 150 days.

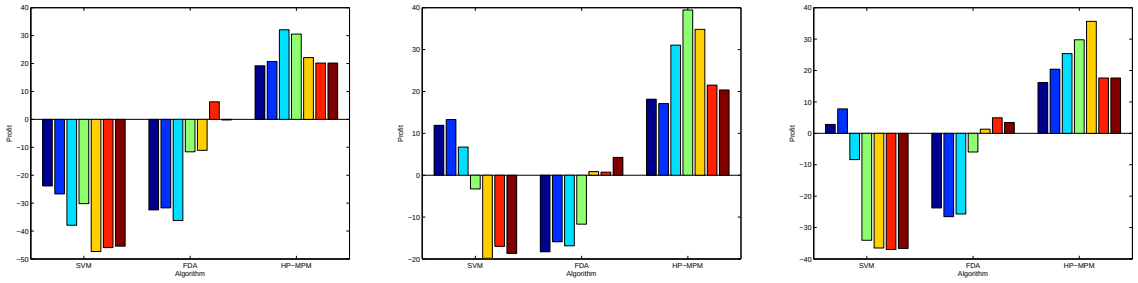
In Figure 3.6 we present the hypothetical profits that would have been generated having traded on the prediction of the algorithms. We see that the HP-MPM approach performs consistently well across varying degrees of regularisation i.e. ν. It only fails to make profits on the AUD/USD currency pair, however its returns are often considerably better than those generated by the FDA or SVM. Similarly the accuracy of the HP-MPM is consistently on par with, if not exceeding that, of the other algorithms. On these datasets it would appear that the moment based algorithms, FDA and HP-MPM , perform better in terms of accuracy the SVM. We believe that this is largely due to the nature in which the solutions are constructed. The SVM will construct its solutions using points that it believes to lie on the boundary of the class-conditional distributions, whereas the moment based solutions are defined by the mean and covariances i.e. the majority of the data, rather than the outliers. Therefore when it comes to finding predictors in high-noise environments, the SVM will be constructing its solutions based on these outlying points rather than constructing it using the points that



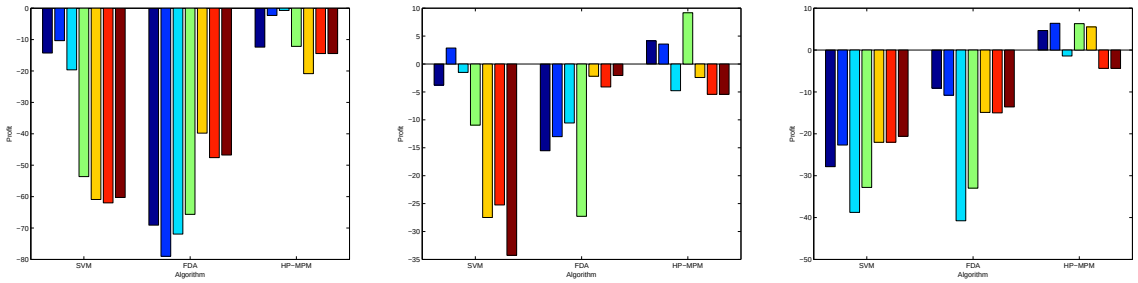
EUR/USD - 50 days | EUR/USD - 100 days | EUR/USD - 150 days



EUR/GBP - 50 days | EUR/GBP - 100 days | EUR/GBP - 150 days

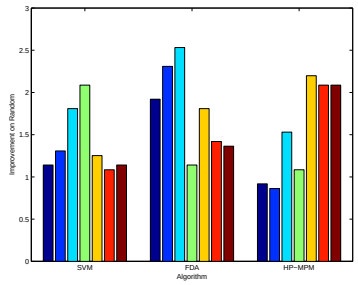


GBP/USD - 50 days | GBP/USD - 100 days | GBP/USD - 150 days

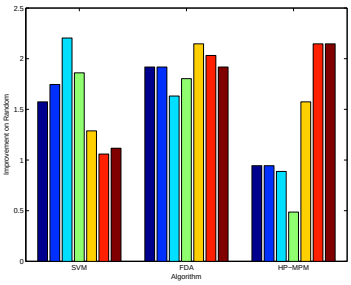


AUD/USD - 50 days | AUD/USD - 100 days | AUD/USD - 150 days

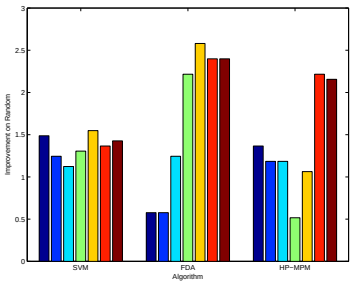
Fig. 3.6 Currency experiments: profit on trading decisions advised by the different algorithms. We see that the HP-MPM performs consistently well across the majority of settings (training window and regularisation). However, all of the algorithm seem to struggle with the AUD/USD currency pair.



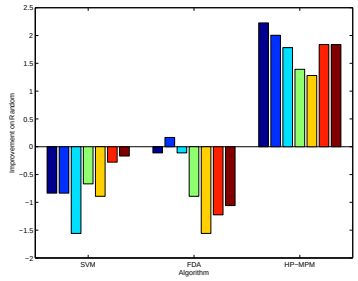
EUR/USD - 50 days



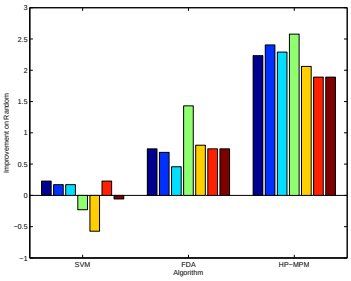
EUR/USD - 100 days



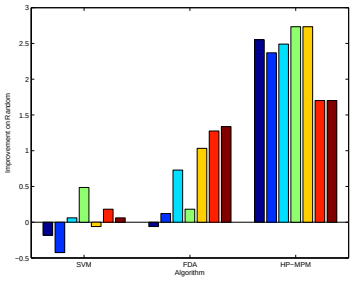
EUR/USD - 150 days



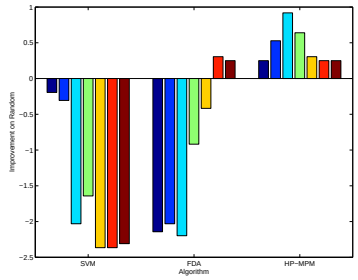
EUR/GBP - 50 days



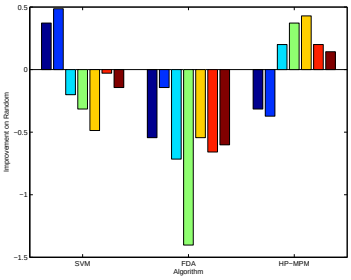
EUR/GBP - 100 days



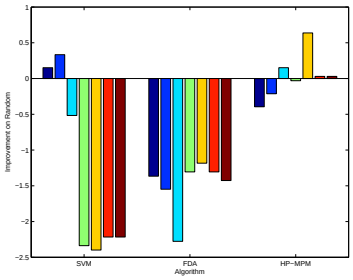
EUR/GBP - 150 days



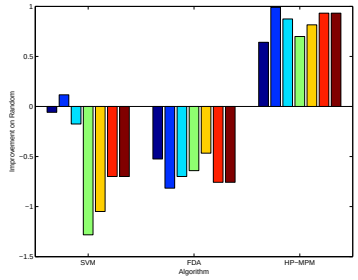
GBP/USD - 50 days



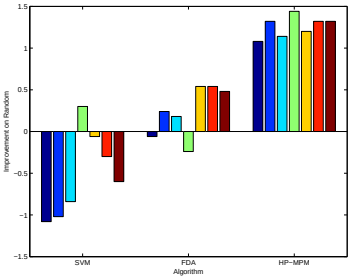
GBP/USD - 100 days



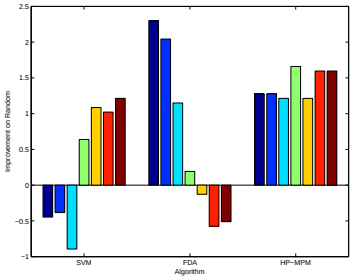
GBP/USD - 150 days



AUD/USD - 50 days



AUD/USD - 100 days



AUD/USD - 150 days

Fig. 3.7 Currency experiments: improvement of accuracy on random guessing i.e. improvement over 50% correct. The HP-MPM appears to be the most consistently performing algorithm and only does worse than random guessing on two particular parameterisations. The other algorithms appear to perform quite considerably worse in terms of accuracy, with the SVM only consistently better than random for the EUR/USD currency pair.

define the mass of the distribution.

3.5 Conclusions and future work

In this chapter we began by introducing alternative approaches for binary classification that use the moments of the class-conditional distribution to construct prediction functions. These approaches seek to construct predictors by aggregating the information contained within the sampled examples, rather than focusing on the peripheral points. In our examination of the minimax probability machine we addressed the oversight that the worst-case future misclassification rate depends explicitly on prior knowledge of each classes mean and covariance matrix. In practice, these true moment values have to be substituted with their empirical counterparts, which are finite sample estimates of their true values. Making use of the high-probability bounds on the deviation of these estimates from their true values [Shawe-Taylor and Cristianini, 2003], we derived a new optimisation scheme that takes into account the moment uncertainty and directly minimises the worst-case future misclassification rate that holds true with high-probability.

Better bounds on moments

During our experiments we observed that in many cases the level of moment uncertainty was so large that it was unable to produce meaningful results i.e. $\kappa_* = 0$. Therefore, at the expense of statistical correctness, we proposed to use fractional quantities of the true moment uncertainty as a form of regularisation. This form of regularisation, unlike most traditional schemes, implicitly takes into consideration the relative uncertainties regarding each class i.e. through different values of A_1 and A_0. During the experiments we noted that its performance was competitive with the popular SVM and FDA approaches, however its advantage was most apparent when minimal amounts of training data were used to construct the decision boundary thus providing support for this new approach to regularisation.

To improve the correctness of this approach, future work should focus on obtaining tighter bounds on the deviation of empirical moments from their true value. It is well known that the general concentration bounds presented in [Shawe-Taylor and Cristianini, 2003], despite being valid, have a tendency to be overly conservative. To overcome this, in [Bertsimas et al., 2013] the authors suggested the use of bootstrapping methods to compute alternate thresholds, and thus drive tighter bounds on the uncertainty sets. By adopting this work into our approach it is possible that it could lead to better, statistically correct, worst-case guarantees, whilst also circumventing the problem of having to use a validation set in order to choose the regularisation terms for the HP-MPM .

Class imbalance

During our experiments we alluded to the poor performance of the HP-MPM on imbalanced datasets. This was partially due to the algorithms implicit belief that the prior probabilities were equal and this was made worse by a regularisation scheme that encourages the hyperplane to be *close* to dominant class. We attempted to circumvent this problem by selecting the positioning of the hyperplane, as given by the bias term b, using the training set accuracy as the criterion for its selection. This arbitrary positioning of the hyperplane invalidated our guarantees on the worst-case probability of misclassification, however it was shown to improve prediction performance across a number of datasets, most notably the imbalanced ones.

The original MPM suffered from a similar problem by implicitly assuming that each class appeared with equal probability. When this holds true, $P(y = 1) = p = 0.5$, it makes sense to minimise a common worst-case error rate ω, however when the data is biased towards one class, $p \neq 0.5$, we are no longer minimising the *true* worst-case error-rate. Clearly if one class has a significantly larger probability of appearing, then it makes sense to focus on minimising the worst-case error rate for this class at the expense of additional error in the less frequent class. This oversight was addressed in [Huang et al., 2004], where the authors presented the minimum error minimax probability machine ME-MPM. By introducing worst-case error rates for each class, ω_1 and ω_0, they solved the following optimisation problem

$$\max_{\mathbf{w},\omega_1,\omega_0} \; p\,\omega_1 + (1-p)\,\omega_0 \quad \text{s.t.} \quad \inf_{\mathbf{x}_1 \sim \mathcal{D}_1} P\left\{\mathbf{w}^T\mathbf{x}_1 \geq b\right\} \geq \omega_1 \qquad \inf_{\mathbf{x}_0 \sim \mathcal{D}_0} P\left\{\mathbf{w}^T\mathbf{x}_0 \leq b\right\} \geq \omega_0. \tag{3.17}$$

The authors [Huang et al., 2004] showed that (3.17) could be solved by performing a one-dimensional line search over ω_0, where intermediate solutions where found by maximising the value of ω_1 subject to a minimum accepted value of ω_0. This method is known as the biased minimax probability machine (B-MPM) [Huang et al., 2006] and can be solved using Fractional Programming methods. This approach is particularly useful for problems where there is a different cost associated with each error i.e. in medical applications where there is a more serious error in the inability to identify whether someone has a given disease, rather than incorrectly predicting that someone has a disease. The BMPM provides a means for controlling the maximum error rate for a particular class, and optimises the performance over the other class subject to this constraint. On first inspection it would seem relatively straightforward to apply the high-probability machinery that we have developed in this chapter in this context. Although we would need to use an additional empirical estimate for the relative class probabilities. If we assume with high probability the error of the prior class probabilities is less than ϵ i.e. $p \in [\hat{p} - \epsilon, \hat{p} + \epsilon]$, there will be two extreme probabilities that we have to consider. It is not currently clear what the implications of this estimation will

have on the smoothness of the original line-search method, and whether or not we would have to perform a line search over p in order to find the worst-case. One would hope that it is possible to show that the worst-case occurs at one of the extrema, which would allow us to avoid having to repeat this optimisation for many values of p. We leave this for future research and briefly discuss the extension of the method for regularisation to the other popular minimax formulation approaches.

In [Osadchy et al., 2015], the minimax principle is used for the abundant class, which would typically have a small degree of moment uncertainty and therefore it seems unlikely that applying our proposed methods would be particularly beneficial in this case. For the transductive [Huang et al., 2014] and clustering based [Huang et al., 2015] minimax approaches, the main difficulty of including moment uncertainty into the optimisation stems from the assignment of unlabelled data to classes. This effectively allows us to have control over the degree of uncertainty surrounding each class, and one could imagine a scenario where we inadvertently encourage equal numbers of observations for both classes in order to mitigate the effects of the regularisation scheme. Despite these potential difficulties, the inclusion of moment uncertainty with existing minimax approaches remains an interesting area of research.

Sparsity methods

During our discussion of the algorithm and the experiments, we briefly mentioned that we have no reason to expect a sparse solution. Whether it be a sparse primal solution i.e. many irrelevant dimensions, or a sparse dual solution, i.e. many zero dual variables. This makes our algorithm unsuitable for problems where we have both a large dimension and a large number of training points. Note that we are free to switch between the primal and dual form when we have only either a large dimension or a large number of training points.

During the formulation of the bounds in Proposition 7 we assumed that the weight vector $\mathbf{w}$ was contained within the L_2 norm unit sphere, $||\mathbf{w}|| \leq 1$. This was done to help with the clarity of our presentation during the formulation of the bounds. We also assume this unit norm restriction in the optimisation scheme, however we could take advantage of the relationship between the unit L_1 and L_2 norms i.e. $||\mathbf{w}|| \leq ||\mathbf{w}||_1$, and use the unit L_1 norm restriction during the optimisation, without worrying about violating the bounds satisfied in the proposition. This holds true since we know that a weight vector $\mathbf{w}$ satisfying $||\mathbf{w}||_1 \leq 1$ will also satisfy $||\mathbf{w}|| \leq 1$. It is well known [Tibshirani, 1996] that the L_1 norm method of regularisation, which we are effectively doing by limiting its norm, encourages sparsity in the co-ordinates of the solution $\mathbf{w}$. In order to solve this optimisation and handle the difficulties associated with its non-differentiability, one would expect to borrow ideas taken

for the methods presented in [Tibshirani, 1996; Yuan et al., 2010]. Furthermore, it would be interesting to understand the implications on the worst-case misclassification probability, if we only include a proportion of the original dimensions. For example, is it fair to treat the separability of classes with respect to different subsets of input variables? Intuitively it seems reasonable to allow the algorithm to ignore variables that don't have any discriminatory power, however given that we work with a finite sample, we might risk disregarding a potentially useful variable due to limited observations.

During the optimisation scheme proposed for the kernel version HP-MPM , each gradient step involves updating each of the m dual variables in $\boldsymbol{\alpha}$. This is a costly operation as it requires us to compute m gradients and check whether the constraint $\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha} \leq 1$ is satisfied. One would hope that we could develop specialised optimisation procedures such as those used in SVMs so as to allow our algorithm to scale to large scale datasets. One simple method, similar in respect to that proposed in [Strohmann et al., 2004], would be to select a single dual variable α_k at each iteration k and update the dual solution $\boldsymbol{\alpha}^{(k)}$ in this direction i.e. $\boldsymbol{\alpha}^{(k+1)} \leftarrow (1-\gamma)\boldsymbol{\alpha}^{(k)}$ followed by $\alpha_k^{(k+1)} \leftarrow \alpha_k^{(k)} + \gamma$, where γ is chosen so as to maximise the value of the auxiliary function $h(\boldsymbol{\alpha}^{(k+1)}, \kappa)$. This process could be repeated until a desired level of convergence is reached. An alternative may be to use approximate kernel methods such as those proposed in [Drineas and Mahoney, 2005; Williams and Seeger, 2001] that involve the use of the Nyström approximation. By implementing these methods we would have to take into consideration an approximation error associated with the computation of means and covariances using a subset of the training samples.

Final remarks

In this chapter we have presented a new algorithm for binary classification that uses the first and second moments of a class conditional distribution, along with our uncertainties regarding their values, to construct a classifier that directly minimises with high probability the worst case future rate of misclassification. Geometrically, we saw that our classifier operated in a similar manner to the original MPM , increasing ellipsoids centred at class means until their point of intersection. The difference in our approach was that the shape of each ellipsoid was adjusted using the uncertainty associated with that class, effectively acting as class specific regularisation scheme.

During our experiments we noted that on many datasets, the number of observations was insufficient to permit meaningful solutions in the minimax setting. This led to the use of fractional amounts of the class specific regularisers, and we observed promising results across a number of UCI datasets. The greatest performance improvements were seen when there were only a small number of observations used and the dimension large. This may be

going in the opposite direction to modern, big-data methods, however we feel it shows that one can still perform well with only a small amount of information, and one can leverage this uncertainty in order to make better predictions on unseen examples. We concluded our experiments by examining the problem of predicting the movement of several different currencies, and comparing the performance of algorithms using moments of the distribution, versus those that use the support of the distribution, to construct predictors. Across our experiments we saw that our HP-MPM method had the most consistent improvement over random guessing, providing evidence support the use of this method for financial predictions. The reason we feel that this method was able to perform consistently well across different parameterisations is its apparent resilience to high levels of noise, and its dependence on the shape of the underlying distributions rather than simply the peripheral points.

Chapter 4

Multi-label learning over unknown graph structures

Machine learning has a long and successful history in solving problems where there is a single output variable to predict. Traditionally, these approaches can be separated into classification, where the output is a binary decision variable, and regression where the output is a scalar quantity. These are arguably the simplest and most general learning settings, and our understanding of what it means to learn with machines has benefited significantly from the considerable research efforts devoted to these problems. As our expertise and understanding of learning in these settings grows, it is natural that we consider approaches better designed to solving real-world problems, since many of these cannot be expressed fully as a simple regression or classification task. In this chapter we study problems where there is more than one output variable to be predicted, which is often referred to in the literature as structured output prediction. Here the goal is to learn a general functional dependency between an arbitrary input space and a complex output space of interdependent variables. The dependency between outputs may reflect some combinatorial, hierarchical, spatial or temporal property related to the problem domain, where being able to capture these relationships is often as important as understanding the relationships between inputs and outputs when it comes to constructing a function for prediction.

Structured prediction has become increasingly prevalent in machine learning research with notable successes in fields such as computer vision, natural language processing and computational biology, to name but a few. This success is partly due to its ability to incorporate knowledge of the structure of the output into the learning algorithm. For example, a common problem in computer vision is assigning each pixel to a particular class (e.g. tree, cat, sky etc.) using inputs given by raw RGB pixel values. Here the incorporation of prior knowledge that neighbouring pixels are likely to represent the same thing is an essential component to

learning accurate prediction functions. However we are often faced with problems where we have no prior knowledge of the structure between output variables and we are unable to use hand-crafted structures to leverage these output correlations and improve our predictions. For example, what is the relationship between the movements in price of the financial assets that make up the stock market.

Graphical models, a popular sub-field in machine learning, provide a powerful learning framework for capturing complex relationships between output variables and allows the construction of high dimensional statistical models. However, the focus here is on forming probabilistic relationships between output variables, for the most part overlooking the case of how different inputs can give rise to different output dependencies. Graphical models could be used to estimate the correlation between different assets in the stock market and manage the risk of a stock portfolio, however this approach fails to take into consideration exogenous variables that can affect the structure within the market. If we knew the relationship between these exogenous variables and the correlation between stocks then we could significantly improve our portfolio management skills.

In this chapter we seek to bridge the gap in the literature between algorithms designed explicitly for learning the structure between output variables, and those for making predictions over known structures. In particular we focus on a particular setting of multi-label learning, where we must learn to predict the labellings of several, possibly interdependent, output variables. When we don't know the structure between the variables, there are two extreme approaches that one can take. The first assumes that each output variable is independent from one another, and the solution is to train individual predictors for each variable. The other extreme assumes that each variable is connected to one another, resulting in what is known as a complete output graph, which makes performing inference on the graph intractable. The method that we are proposing falls somewhere in between these two, defining the structure using a sparse combination of spanning trees. Later we show that we are able to overcome the intractability of inference over the graph by using a collection of spanning trees, whilst also showing that the margin obtained using these trees is at least as large as that found using the complete graph.

We continue this chapter with a short introduction to graphical models, looking at how inference can performed on them and examining how parameters and structures can be learned from data. The focus turns to the case where outputs depend on an arbitrary input and we discuss several state-of-the-art methods for structured prediction, before presenting a new method for multi-label classification. A graph agnostic learning framework that finds the structure that maximises the margin of the predictor. We conclude by investigating the

performance of our newly proposed method on a range of benchmark multi-label datasets and discuss the ramifications of this approach and the possible extensions that may exist.

4.1 Graphical models

A graph $G = (V, E)$ encodes a set of conditional independence assumptions and is defined by a set of vertices V and edges E. Each vertex $V_i \in V$ is associated with a random variable $Y_i \in \mathcal{Y}_i$. The space of possible outcomes is denoted $\mathcal{Y} = \mathcal{Y}_1 \times \mathcal{Y}_2 \times \cdots \times \mathcal{Y}_\ell$, where we have assumed there are ℓ vertices in the graph i.e. $|V| = \ell$. We denote the realisation of a random variable by $Y_i = y_i$, and the realisation of all vertices $Y = \mathbf{y}$. Furthermore, let Y_S denote a subset of random variables indexed by $S \subseteq G$, the realisation of random variables belonging to this subset is denoted $Y_s = \mathbf{y}_s$. Each edge $e \in E$ consists of a pair of vertices $e = (i, j) \in V \times V$, where the existence of an edge between V_i and V_j represents some dependency between the random variables Y_i and Y_j. We denote the labelling of edge $e = (i, j) \in E$ by $Y_e = \mathbf{y}_e \in \mathcal{Y}_i \times \mathcal{Y}_j$. A clique $C \subset G$ is a fully connected subset of vertices of G, where there exists an edge between for each pair of vertices $i, j \in C$. A maximal clique is one that cannot be extended to include further vertices without violating the fully connectedness property.

Graphical models can generally be divided into two categories based upon the nature of their edges; directed graphical models and undirected graphical models. Both models are capable of representing a family of joint probability distributions over $\mathcal{Y}$, however they differ in their factorisation and conditional independence relations. Directed graphical models are often considered more appropriate when it is possible to attribute a directionality on the dependency between variables, and thus edges. Later when it comes to learning graph structure, we assume no prior knowledge of the relationship between random variables, which makes it difficult to attribute the responsibility of dependencies. Therefore we will focus our efforts on the undirected case, and refer the reader to [Barber, 2012; Heckerman, 1998; Koller and Friedman, 2009] for further information on directed models.

Undirected graphical models are also known as Markov random fields, their name derived from the Markov properties exhibited by the random variables defined over the graph G, namely:

- Pairwise Markov Property: we have for all $i, j \in V : i \neq j \wedge (i, j) \notin E : i \perp\!\!\!\perp j \,|\, V \setminus \{i, j\}$.
- Global Markov Property: we have for all disjoint set $I \subset V, J \subset V, S \subset V$ with S being a vertex-separator set of I and J in G such that $I \perp\!\!\!\perp J \,|\, S$.

where a vertex separator is a subset of nodes in V, $\wedge$ is the logical conjunction operator,

$A \perp\!\!\!\perp B$ says that A and B are independent of one another, $A \subset B$ states that A is a subset of B and $A \setminus B$ corresponds to set A minus set B. Note that it is easy to see that the global Markov property implies the pairwise Markov property by simply taking $I = \{i\}, J = \{j\}$ and $S = V \setminus \{i, j\}$.

If a distribution of random variables satisfies the Markov properties with respect to G, and every realisation of random variables has a positive probability i.e. $P(\mathbf{y}) > 0$ for all $\mathbf{y} \in \mathcal{Y}$, then the Hammersley-Clifford theorem [Besag, 1974; Hammersley and Clifford, 1971] states that the probability distribution can be factorised over the cliques of the graph, namely

$$P(Y = \mathbf{y}) = P(\mathbf{y}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} h_C(\mathbf{y}_C), \tag{4.1}$$

where $\mathcal{C}$ is the set of all cliques in G, h_C are non-negative factors $h_C : \mathcal{Y}_C \to \mathbb{R}_+$, and Z is a normalisation constant, which ensures a valid probability distribution i.e. $\frac{1}{Z} \sum_{\mathbf{y} \in \mathcal{Y}} P(\mathbf{y}) = 1$. This is known as the partition function and is given by

$$Z = \sum_{\mathbf{y} \in \mathcal{Y}} \prod_{C \in \mathcal{C}} h_C(\mathbf{y}_C)\,.$$

These non-negative factors $h_C(\mathbf{y}_C)$ do not correspond to probabilities but rather capture the *agreement* between the configuration of random variables $\mathbf{y}_C$ within the clique C. The advantage of using these agreement factors within cliques is that they allow local information to be efficiently propagated throughout the rest of the graph, taking into consideration the conditional independencies implied by G. This propagation takes place through through the intersection of the various cliques within the graph.

We now present some examples of graphical models, considering factorisations in terms of the set of maximal cliques $\mathcal{C}$ over the graph. However later we shall make use of an over-complete, non-maximal clique, representation of the output graph to take advantage of the decomposability properties of our chosen kernels. In Figure 4.1 we present a complete graph, which is defined as a simple undirected graph in which there exists an edge between every pair of vertices. We see that there is only one maximal clique, and it is represented by the set of all vertices, namely, $C_0 = V$, and the subsequent factorisation contains a single factor. In Figure 4.2 we present a graph consisting of two maximal cliques $C_0 = \{0, 1, 2\}$ and $C_1 = \{0, 2, 3\}$, and two factors corresponding to the labelling of the random variables indexed by these cliques. In this graph the intra-clique information is propagated through the labelling of nodes $C_0 \cap C_1 = \{0, 2\}$. In Figure 4.3 we present a tree-structured graph, an undirected graph where any two vertices in the graph are connected by a unique sequence of edges. The maximal cliques correspond to the edges of the graph and we have three cliques,

$C_0 = \{0, 1\}$, $C_1 = \{1, 2\}$ and $C_2 = \{2, 3\}$ and the information is propagated through each of the clique intersections.

Throughout this chapter we make use of the fact that a product of positive functions (4.1) can be expressed as the exponential of a sum of functions. This representation of a probability distribution is known as the log-linear model and is given as

$$P(\mathbf{y}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} h_C(\mathbf{y}_C) = \exp\left(\sum_{C \in \mathcal{C}} \log h_C(\mathbf{y}_C) - \log Z \right).$$

This motivates the use of the exponential family representation

$$P(\mathbf{y}|\mathbf{w}) = \exp\left(\langle \mathbf{w}, \boldsymbol{\psi}(\mathbf{y}) \rangle - \log Z(\mathbf{w}) \right) = \exp\left(\sum_{C \in \mathcal{C}} \langle \mathbf{w}_C, \boldsymbol{\psi}_C(\mathbf{y}_C) \rangle - \log Z(\mathbf{w}) \right),$$

where $\boldsymbol{\psi} : \mathcal{Y} \to \mathbb{R}^n$ are known as the *sufficient statistics*, $\mathbf{w} \in \mathbb{R}^n$ are their associated parameters and the value of the partition function $Z(\mathbf{w})$ depends on the choice of model parameters. The sufficient statistics can be represented by the concatenation of individual clique sufficient statistics $\boldsymbol{\psi}_C(\mathbf{y}_C)$ where each clique's sufficient statistic can be thought of as a feature mapping that acts as an indicator function for a particular realisation of the clique. The dimension of each clique's sufficient statistic is given by $|\mathcal{Y}_C|$ and therefore the dimension of the model is $n = \sum_{C \in \mathcal{C}} |\mathcal{Y}_C|$. The parameters $\mathbf{w}_C \in \mathbb{R}^{|\mathcal{Y}_C|}$ associated with each clique correspond to the logarithm of the clique factors i.e. $\langle \mathbf{w}_C, \boldsymbol{\psi}_C(\mathbf{y}_C) \rangle = \log h_C(\mathbf{y}_C)$, which we refer to as *compatibility factors*. For example, consider the tree-structured graph given in Figure 4.3 where each random variable $Y_i \in \{0, 1\}$. The sufficient statistic, which we will refer to as a feature mapping, $\boldsymbol{\psi}(\mathbf{y})$ is given by

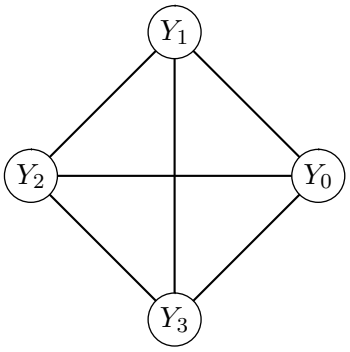
$$\boldsymbol{\psi}(\mathbf{y}) = \left(\boldsymbol{\psi}_{C_0}(\mathbf{y}_{C_0}), \boldsymbol{\psi}_{C_1}(\mathbf{y}_{C_1}), \boldsymbol{\psi}_{C_2}(\mathbf{y}_{C_2}) \right)^T.$$

Here $n = 12$ corresponding to the three different cliques, each of which two random variables and can take on four possible values i.e.

$$\boldsymbol{\psi}_C(\mathbf{y}_C) = \left(\mathbb{1}[\mathbf{y}_C = (0,0)],\ \mathbb{1}[\mathbf{y}_C = (0,1)],\ \mathbb{1}[\mathbf{y}_C = (1,0)],\ \mathbb{1}[\mathbf{y}_C = (1,1)] \right)^T.$$

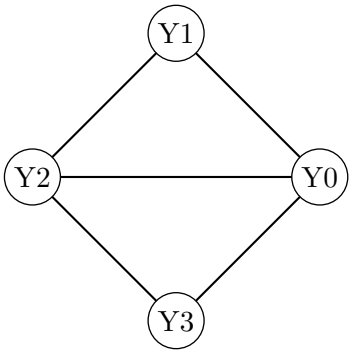
4.2 Inference over graphs

Now that we have introduced some basic concepts regarding Markov random fields MRF, we can look at how they can be used to for performing inference over given graph structures. Assuming that we have been given the graph structure and compatibility factors, there are two main types of inference that one may wish to perform on a graphical model;



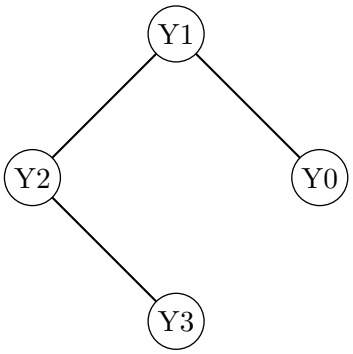
$$V = \{0, 1, 2, 3\}$$
$$E = \{(0, 1), (0, 2), (0, 3), (1, 2), (1, 3), (2, 3)\}$$
$$\mathcal{C} = \{\{0, 1, 2, 3\}\}$$
$$P(Y = \mathbf{y}) = \frac{1}{Z} h_V(y_0, y_1, y_2, y_3)$$

Fig. 4.1 Complete output graph



$$V = \{0, 1, 2, 3\}$$
$$E = \{(0, 1), (0, 2), (0, 3), (1, 3), (2, 3)\}$$
$$\mathcal{C} = \{\{0, 1, 2\}, \{0, 2, 3\}\}$$
$$P(Y = \mathbf{y}) = \frac{1}{Z} h_{C_0}(y_0, y_1, y_2) h_{C_1}(y_0, y_2, y_3)$$

Fig. 4.2 Two-clique graph



$$V = \{0, 1, 2, 3\}$$
$$E = \{(0, 1), (1, 2), (2, 3)\}$$
$$\mathcal{C} = \{\{0, 1\}, \{1, 2\}, \{2, 3\}\}$$
$$P(Y = \mathbf{y}) = \frac{1}{Z} h_{C_0}(y_0, y_1) h_{C_1}(y_1, y_2) h_{C_2}(y_2, y_3)$$

Fig. 4.3 Tree-structured graph

- The computation of marginal probability distributions. Let $S \subseteq V$ denote some subset of variables, assuming the graph is parameterised by $\mathbf{w}$, the marginal distribution $P(\mathbf{y}_S)$ is given by

$$P(\mathbf{y}_S) = \sum_{\mathbf{y}' \in \mathcal{Y} : \mathbf{y}'_S = \mathbf{y}_S} P(\mathbf{y}) = \sum_{\mathbf{y}' \in \mathcal{Y} : \mathbf{y}' = \mathbf{y}_S} \exp\left(\langle \mathbf{w}, \boldsymbol{\psi}(\mathbf{y}') \rangle - \log Z(\mathbf{w})\right) .$$

- Finding the labelling with highest probability, which is known as the *maximum a posterior* MAP configuration

$$\mathbf{y}^* = \underset{\mathbf{y} \in \mathcal{Y}}{\mathbf{argmax}}\ P(\mathbf{y}|\mathbf{w}) = \underset{\mathbf{y} \in \mathcal{Y}}{\mathbf{argmax}}\ \langle \mathbf{w}, \boldsymbol{\psi}(\mathbf{y}) \rangle .$$

Worst case analysis, for computing marginals [Cooper, 1990] and finding MAP configurations [Shimony, 1994], has shown that for general graphs these inference tasks are NP-hard to solve exactly. For example, a naive method for computing the MAP configuration would be to compute the value (ignoring the partition function) of all possible configurations and choose the one with the highest value. Even for the simple case where each random variable is binary, this would require 2^ℓ computations, which quickly becomes intractable as the size of the graph grows. Similarly to compute the marginal probability for a single random variable, $P(Y_i = y_i)$, the naive method would require the summation over $2^{\ell-1}$ configurations i.e. $P(Y_i = y_i) = \sum_{\mathbf{y}' \in \mathcal{Y} : y'_i = y_i} P(\mathbf{y})$.

Fortunately there are some graph structures where exact inference can be performed in polynomial time, for example in tree-structured graphs [Pearl, 1986], planar Ising models [Schraudolph and Kamenetsky, 2009] and graphs where the parameters defined over cliques exhibit submodularity [Greig et al., 1989]. Later we will perform inference on tree-structured graphs, which are known more generally as trees. Here one can perform these inference tasks *exactly* in linear time using recursive message-passing algorithms, similar in nature to the approaches taken in dynamic programming. The algorithms used for computing the marginal distribution and mode of the distribution work almost the same, and are referred to as sum-product algorithm and max-product algorithm, respectively. The terms *sum* and *max* refer to the operations undertaken as the algorithm works recursively through the tree-structure. To perform marginal inference, the messages passed through the tree by the sum-product algorithm are given by

$$m_{i \to j}(y_j) \propto \sum_{y_i} \left[\exp\left(\langle \mathbf{w}_{ij}, \boldsymbol{\psi}_{i,j}(y_i, y_j) \rangle + \prod_{s \in N(i) \setminus j} m_{s \to i}(y_i) \right) \right] ,$$

and the marginal distributions are obtained by setting

$$P(y_i) \propto \prod_{j \in N(i)} m_{j \to i}(y_i) \tag{4.2}$$

$$P(y_i, y_j) \propto \exp\left(\langle \mathbf{w}_{ij}, \boldsymbol{\psi}_{ij}(y_i, y_j) \rangle\right) \frac{P(y_i)P(y_j)}{m_{i \to j}(y_j) m_{j \to}(y_i)} \tag{4.3}$$

For computing the MAP configuration, the messages passed through the tree by the max-product algorithm are given by

$$m_{i \to j}(y_j) \propto \max_{y_i} \left[\exp\left(\langle \mathbf{w}_{ij}, \boldsymbol{\psi}_{i,j}(y_i, y_j) \rangle + \prod_{s \in N(i) \setminus j} m_{s \to i}(y_i) \right) \right],$$

and the MAP configuration of a particular node is given by

$$y_i = \underset{y_i \in \mathcal{Y}_i}{\mathbf{argmax}} \prod_{j \in N(i)} m_{j \to i}(y_i)\,. \tag{4.4}$$

Note that this MAP configuration is for a particular node and it is not always the case that they correspond to the global MAP configuration. Therefore in order to compute the global MAP labelling we must trace back through the tree conditioning on previously maximised variables.

For general graphs, these message-passing algorithms can be implemented by converting the graph into a clique tree structure [Robertson and Seymour, 1986] and using the junction tree-algorithm [Lauritzen and Spiegelhalter, 1988] to perform the message-passing operations over cliques in a similar manner. Inference is performed in time that is exponential with respect to the treewidth of the graph.[1] In the example graphs presented in Figures 4.1-4.3, the tree-width of the complete graph is $\ell - 1 = 3$, whereas the two-clique graph has a treewidth 2 and the tree-structured graph is obviously 1. On a graph this size, simple enumeration is a trivial solution to these inference tasks, however as the number of nodes scales, even just beyond 50, it becomes intractable to perform such brute force methods. When faced with complete graphs, or highly connected graphs, approximate inference schemes are often favoured to their more time consuming exact schemes. For example, Monte-Carlo methods such as Gibbs sampling [Geman and Geman, 1984] and the Metropolis-Hastings [Hastings, 1970] algorithm sample from the underlying distribution $P(\mathbf{y})$ to obtain estimates, or Variational methods such as Mean-field approximations convert the inference problem into an optimisation problem [Wainwright et al., 2008]. Other popular approaches include introducing relaxations over the search spaces, these are variational approaches i.e. linear and

[1] The treewidth of a graph G is one less the largest clique of the chordal graph, where a chordal graph is the result of a triangulation phase that includes there is no loop in the graph greater than length 3.

quadratic programming relaxations [Bertsimas and Tsitsiklis; Ravikumar and Lafferty, 2006; Wainwright et al., 2005]. Later we shall show how to extend the algorithm for computing the mode of a density defined over a tree, and present conditions that ensure exact inference has been performed over a graph formed by the superposition of a collection of trees.

4.3 Learning over graphs

So far we have discussed the problem of performing inference tasks such as computing the MAP labelling under the assumption that the graph structure and its parameters are known in advance. However, we might also want to learn the parameters over a particular graph structure, or learn the conditional independence relationships between the variables. These tasks are referred to as parameter learning and structure learning, respectively. In this section we discuss the case where the distribution of random variables depends on the observation of some input variable $x \in \mathcal{X}$. These types of graph are often referred to as a conditional random fields (CRFs) [Lafferty et al., 2001], where the distribution of random variables is given by

$$P(\mathbf{y}|x, \mathbf{w}) = \exp\left(\langle \mathbf{w}, \boldsymbol{\phi}(x, \mathbf{y})\rangle - \log Z(x, \mathbf{w})\right) ,$$

where $\boldsymbol{\phi} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^D$ is a feature map defined over the joint space of inputs and outputs and the partition function $Z(x, \mathbf{w}) = \sum_{\mathbf{y}} \exp(\langle \mathbf{w}, \phi(x, \mathbf{y})\rangle)$ depends on both the parameters $\mathbf{w}$ and the input observation $x \in \mathcal{X}$. The joint feature mapping can once again be broken down into factors defined over cliques so that the compatibility score can be expressed as

$$\langle \mathbf{w}, \boldsymbol{\phi}(x, \mathbf{y})\rangle = \sum_{C \in \mathcal{C}} \langle \mathbf{w}_C, \phi_C(x, \mathbf{y})\rangle$$

To give an example, consider the problem of labelling the pixels of an image as foreground or background using the RGB value of the pixels as inputs. In this example, we would expect that neighbouring pixels are likely to share the same labelling thus defining a structure over the output that can be thought of as a grid over the image. The input values of the pixels could represent the similarity in colour of particular pixels and thus help identify sharp changes in colours, which could correspond to edges and are likely to indicate a change between foreground and background.

4.3.1 Parameter Learning

We study the problem of parameter learning by using the risk minimisation principle introduced earlier. We minimise the empirical risk using a dataset of input output pairs $S = \{(x_1, \mathbf{y}_1), \ldots, (x_m, \mathbf{y}_m)\}$ and a loss function $L(x, \mathbf{y}, \mathbf{w})$. Given the expressiveness of

graphical models, a regularisation component Ω is often included in an attempt to prevent overfitting and we will see later how the choice of loss functions and regularisers give rise to familiar looking algorithms.

Maximum likelihood learning

The maximum likelihood principle is a popular approach to learning the parameters defined over a given graph structure. The goal is to maximise the value of the log likelihood expression

$$\mathcal{L}(S, \mathbf{w}) := \frac{1}{m} \sum_{k=1}^{m} \langle \mathbf{w}, \boldsymbol{\phi}(x_k, \mathbf{y}_k) \rangle - \log Z(x_k, \mathbf{w}) \,.$$

This approach was first presented in [Lafferty et al., 2001] where the authors introduced the concept of conditional random fields for discriminatively training models for natural language processing. They were able to capture the dependencies between output variables Y using much simpler graph structures by conditioning on some *input* X. The loss-function being minimised in this method is the negative log-likelihood of the conditional distribution

$$L(x, \mathbf{y}, \mathbf{w}) = -\log P(\mathbf{y}|x, \mathbf{w}) = -\langle \mathbf{w}, \boldsymbol{\phi}(x, \mathbf{y}) \rangle + \log Z(x, \mathbf{w})$$

The use of the exponential family implies the concavity of the objective function $\mathcal{L}(S, \mathbf{w})$ making the optimisation amenable to gradient-based learning methods. We see that the gradient with respect to parameters $\mathbf{w}$ is given by

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{1}{m} \sum_{k=1}^{m} \left[\boldsymbol{\phi}(x_k, \mathbf{y}_k) - \frac{\partial}{\partial \mathbf{w}} \log Z(x_k, \mathbf{w}) \right] .$$

Unfortunately the presence of the partition function results in a coupling together of all model parameters and makes the computation of the gradient NP-hard for general graph structures. To see this, observe that the gradient of the log partition function is given by

$$\frac{\partial}{\partial \mathbf{w}} \log Z(x, \mathbf{w}) = \sum_{\mathbf{y} \in \mathcal{Y}} \boldsymbol{\phi}(x, \mathbf{y}) P(\mathbf{y}|x, \mathbf{w}) = \mathbb{E}\left[\boldsymbol{\phi}(x, \mathbf{y}) | x, \mathbf{w} \right] ,$$

which requires inference over the expected value of the feature vectors. The gradient takes on the intuitive form of being the difference between the empirical mean of the features and the average expected value of the features under the conditional distributions i.e.

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{1}{m} \sum_{k=1}^{m} \boldsymbol{\phi}(x_k, \mathbf{y}_k) - \frac{1}{m} \sum_{k=1}^{m} \mathop{\mathbb{E}}_{P(\mathbf{y}|x_k, \mathbf{w})} \left[\boldsymbol{\phi}(x_k, \mathbf{y}) \right]$$

This largely coincides with the general notion of what it means to have learned, we want

to find a model, parameterised by $\mathbf{w}$, where the expectations under this model match what we have seen in our data. One further difficulty associated with the gradients is that each different input observation requires a separate inference procedure in order to compute the conditional expectation of the sufficient statistic. However one would hope that by conditioning on the input observation the inference calculations would become more tractable, however this depends on the assumed structure. For example in [Lafferty et al., 2001], by conditioning on the input features, the structure exhibited by the part of speech tagging becomes a chain-structured graph, and is therefore amenable to efficient inference.

Given the difficulties associated with computing gradients, there have been a number of methods that have been proposed in an attempt to improve learning maximum likelihood estimates. These broadly fall into two categories; computing the gradient using approximate inference schemes, and optimising an alternative, more tractable objective function. The approximate inference schemes that we alluded to earlier, such as loopy belief propagation [Pearl, 1986; Yedidia et al., 2005] or sampling based methods, could clearly operate in a black-box manner, providing estimates of the gradient. However a potential drawback of these approaches is that erroneous (approximate) gradient computations can often lead to sub-optimal solutions and result in poor rates of convergence [Koller and Friedman, 2009].

The other popular approach for dealing with the inference task involves the optimisation of an alternative objective function that circumvents the problems associated with the partition function and the summation over an exponentially large set of variables. For example, the pseudo likelihood [Besag, 1977] approach conditions each variable with respect to all others, and through Bayes rule allows the formulation of an objective function that omits the troublesome partition function, and requires only the summation over $\mathcal{Y}_i$. The pseudo-likelihood function is given by

$$\mathcal{L}_{\text{pseudo}}(\mathcal{S}, \mathbf{w}) := \frac{1}{m} \sum_{k=1}^{m} \sum_{i \in V} \log P(y_{k,i} | \mathbf{y}_{k,V \setminus i}, x_k, \mathbf{w}),$$

where $\mathbf{y}_{k,V \setminus i}$ denotes all the variables except that corresponding the i-th vertex of the k-th training example. The pseudo-likelihood is a consistent estimator of the true likelihood i.e. as the number of samples $m \to \infty$ it yields an exact solution. However in the limited data setting, using this approach often suffers by assuming that each variables neighbours are fully observable, and therefore when it comes to learning we are prone to ignoring the influence of other variables that are not immediate neighbours i.e. those variables that they are conditionally independent of given a particular set of observations.

Contrastive divergence [Hinton, 2002; Teh et al., 2003] is an another popular approach, which

introduces a randomly perturbed set of samples and compares the likelihood of these perturbed samples with those that have been observed $\mathcal{S}$. This approach does not approximate the likelihood function itself, but rather performs a stochastic estimate of the gradient. The goal is to adjust the parameters so that the likelihood of the samples actually observed is larger than those of that have been randomly perturbed. For each example $(x_k, \mathbf{y}_k)$ we assume that we have a set of V randomly perturbed output labellings $\{\mathbf{y}_k^v\}_{v=1}^V$. A simple example of the gradient used in contrastive divergence is given by

$$\frac{\partial \mathcal{L}_{\text{CD}}}{\partial \mathbf{w}} = \frac{1}{m}\sum_{k=1}^{m}\left(\boldsymbol{\phi}(x_k, \mathbf{y}_k) - \frac{1}{V}\sum_{v=1}^{V}\boldsymbol{\phi}(x_k, \mathbf{y}_k^v)\right),$$

In order to generate the contrasting samples for contrastive divergence, one must find an efficient way of sampling $\mathbf{y}_k^v$ from the distribution $P(\mathbf{y}|x, \mathbf{w})$, which can itself be intractable. However, there have been some empirical results that have shown that even with a single cycle of Markov chain Monte Carlo sampling the algorithm is capable of converging to the maximum likelihood solution.

Large margin learning

In large margin learning approaches we move away from probabilistically modelling the conditional distribution and focus on directly learning a function that returns the MAP labelling. The prediction function used for structured SVM takes the form

$$f_{\mathbf{w}}(x) := \underset{\mathbf{y}\in\mathcal{Y}}{\mathbf{argmax}} \left\langle \mathbf{w}, \boldsymbol{\phi}(x, \mathbf{y})\right\rangle .$$

We no longer require that the parameters $\mathbf{w}$ represent a meaningful estimate of the conditional distribution but rather ensure that the MAP estimates for the training sample inputs corresponds to their respective outputs. From the risk minimisation point of view, the goal of the structured SVM is to find the prediction function $f_{\mathbf{w}}$ that minimises the (regularised) empirical risk functional given by

$$\frac{1}{m}\sum_{k=1}^{m}\Delta(\mathbf{y}_k, f_{\mathbf{w}}(x_k)) + \Omega(||f_{\mathbf{w}}||^2) .$$

The loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ can be used to define a number of different structured prediction tasks. The two we are interested in are the zero-one loss and the Hamming loss (see [Nowozin and Lampert, 2011] for further examples):

- **Zero-one loss:** $\Delta(\mathbf{y}, \mathbf{y}') = \mathbb{1}[\mathbf{y} \neq \mathbf{y}']$. This is what is used in binary classification, and commonly in multi-class classification. It is less frequently used in structured

prediction problems when there is a large output space, since it implies that small discrepancies between outputs are punished the same as large.

- **Hamming loss:** $\Delta(\mathbf{y}, \mathbf{y}') = \frac{1}{\ell}\sum_{i=1}^{\ell} \mathbb{1}[y_i \neq y_i']$. A popular choice when the output can be broken down into multiple parts, where each can be judged independently whether it is an error or not.

As with the traditional SVM, we are unable to directly minimise these loss-functions due to their non-convexity. Therefore we make use of a surrogate loss function, which is given by the maximum margin loss function

$$L_{\text{mm}}(x, \mathbf{y}, \mathbf{w}) := \max_{\mathbf{y}' \in \mathcal{Y}} \left[\langle \mathbf{w}, \boldsymbol{\phi}(x, \mathbf{y}') + \Delta(\mathbf{y}, \mathbf{y}')\right] - \langle \mathbf{w}, \boldsymbol{\phi}(x, \mathbf{y})\rangle \,.$$

This acts as a convex upper bound on the loss functions and is known to act as a consistent estimator of the function that minimises the underlying loss Δ [Zhang, 2004]. The optimisation problem for the structured SVM can be written as

$$\begin{aligned} &\min_{\mathbf{w},\boldsymbol{\xi}} \ \frac{1}{2}||\mathbf{w}||^2 + C\sum_{k=1}^{m} \xi_k \\ &\text{s.t. } \langle \mathbf{w}, \boldsymbol{\phi}(x_k, \mathbf{y}_k) - \boldsymbol{\phi}(x_k, \mathbf{y})\rangle \geq \Delta(\mathbf{y}_k, \mathbf{y}) - \xi_k \ \forall\ \mathbf{y} \in \mathcal{Y}, k = 1, \ldots, m \\ &\xi_k \geq 0 \ \forall\ k = 1, \ldots m \,, \end{aligned} \tag{4.5}$$

where C is the regularisation term that trades off complexity with model fit. The constraints tell us that, for each $\mathbf{y} \in \mathcal{Y}$, we want the score of the true labelling $\mathbf{y}_k$ to be at least $\Delta(\mathbf{y}_k, \mathbf{y})$ larger than the score evaluated at $\mathbf{y}$. The presence of the slack term, ξ_k, allows these constraints to be violated and they enter into the objective function as the maximum margin losses. The structured SVM was independently developed by [Tsochantaridis et al., 2004] and [Taskar et al., 2004], however the constraints presented in 4.5 corresponds to the margin scaling version suggested by [Taskar et al., 2004].

In the optimisation problem given in 4.5, there are exponentially many constraints for each training example $(x_k, \mathbf{y}_k)$ i.e. one for each $\mathbf{y} \in \mathcal{Y}$. This constraint can be replaced with a single constraint by using the maximum margin loss, transforming the optimisation problem to

$$\min_{\mathbf{w},\boldsymbol{\xi}} \ \frac{1}{2}||\mathbf{w}||^2 + C\sum_{k=1}^{m} \xi_k \tag{4.6}$$

$$\text{s.t. } \langle \mathbf{w}, \boldsymbol{\phi}(x_k, \mathbf{y}_k)\rangle \geq \max_{\mathbf{y} \in \mathcal{Y}} \left[\langle \mathbf{w}, \boldsymbol{\phi}(x_k, \mathbf{y})\rangle + \Delta(\mathbf{y}_k, \mathbf{y})\right] - \xi_k \ \forall\ k = 1, \ldots, m \tag{4.7}$$

$$\xi_k \geq 0 \ \forall\ k = 1, \ldots m \,.$$

This notion of a single *maximally violating labelling* (or a small set of them) for each $(x_k, \mathbf{y}_k) \in S$, makes the cutting plane algorithm a suitable candidate for solving the optimisation problem. The cutting plane algorithm is described in Algorithm 2. The algorithm begins by initialising an empty set of the active constraints $\mathcal{A}_k = \emptyset$ for each training example and we specify the desired level of precision ϵ. The algorithm proceeds in rounds, iterating through the training examples and checking to see if any constraints are violated. If a constraint is violated, it is added to the active constraint set and the quadratic program QP is solved again. This process is repeated until no further constraints are added to $\mathcal{A} = \bigcup_{k=1}^{m} \mathcal{A}_k$.

In [Tsochantaridis et al., 2004] an alternative choice of constraints is proposed, which scales the slack variables by the loss assuming a fixed margin i.e.

$$\langle \mathbf{w}, \boldsymbol{\phi}(x_k, \mathbf{y}_k) \rangle - \langle \mathbf{w}, \boldsymbol{\phi}(x_k, \mathbf{y}) \rangle \geq 1 - \frac{\xi_k}{\Delta(\mathbf{y}_k, \mathbf{y})} \;\; \forall \; \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_k \,. \tag{4.8}$$

Here we have to omit the true labelling $\mathbf{y}_k$ from the constraints to prevent division by zero. One benefit of the slack scaling approach (4.8) is that the optimal weight vector is invariant to the scaling of the loss function, whereas the margin scaling approach, as used in (4.7), requires a calibration between the scaling of the feature map and the loss function. A further disadvantage of the margin scaling approach is that it pays significant attention to outputs with large loss, which may not be in the slightest bit confusable with the true labelling. By constructing the prediction function using these highly unlikely values, we limit our ability to distinguish between similar outputs. Despite these drawbacks, the margin scaling approach has become the standard approach for structured SVM, largely due to the ease in which it can be included into the optimisation scheme.

Algorithm 2 Cutting-plane algorithm

Input: $S = \{(x_k, \mathbf{y}_k)\}_{k=1}^{m}$, $C > 0$, $\epsilon > 0$
$\mathcal{A}_k \leftarrow \emptyset$, $\xi_k = 0$ for all $k = 1, \ldots, m$
repeat
  for $k = 1, \ldots, m$
    $L(\mathbf{y}) := \Delta(\mathbf{y}_k, \mathbf{y}) + \langle \mathbf{w}, \boldsymbol{\phi}(x_k, \mathbf{y}) - \boldsymbol{\phi}(x_k, \mathbf{y}_k) \rangle$
    compute $\hat{\mathbf{y}} = \mathbf{argmax}_{\mathbf{y} \in \mathcal{Y}} L(\mathbf{y})$ *(loss augmented inference)*
    compute $\xi_k = \max\{0, \max_{\mathbf{y} \in \mathcal{A}_k} L(\mathbf{y})\}$
    if $L(\hat{\mathbf{y}}) > \xi_k + \epsilon$ **then**
      $\mathcal{A}_k \leftarrow \mathcal{A}_k \cup \{\hat{\mathbf{y}}\}$ *(constraint inclusion)*
      $\mathbf{w} \leftarrow$ solve 4.5 using constraint set $\mathcal{A} = \bigcup_{k=1}^{m} \mathcal{A}_k$
    end if
  end for
until no $\mathcal{A}_k$ changes during an iteration

The two main concerns with the cutting place algorithm are the number of times a QP has to be solved, and the exponential number of constraints that we may have to add. The QP problem is addressed by initialising the program with the last solution, hoping that the new solution will not be too far away and thus converge fast. To address the constraints, it was shown in [Tsochantaridis et al., 2004] that this optimisation can be solved to an ϵ-precision using only a polynomially sized set of constraints for each example. These features make solving the structured SVM feasible for medium sized datasets, however they can often struggle to scale to larger datasets. To handle the size of the constraint sets associated with larger problems, [Joachims et al., 2009] introduced a single slack version of the structured SVM, where the time complexity of the algorithm is independent of the number of training examples and depends linearly on the desired precision and the value of the regularisation parameter. A single slack variable ξ is used for the entire dataset, which effectively couples together constraints across all training examples,

$$\frac{1}{m}\sum_{k=1}^{m}\langle \mathbf{w}, \boldsymbol{\phi}(x_k, \mathbf{y}_k) - \boldsymbol{\phi}(x_k, \mathbf{y})\rangle \geq \frac{1}{m}\sum_{k=1}^{m}\Delta(\mathbf{y}_k, \mathbf{y}) - \xi \quad \forall\, \mathbf{y} \in \mathcal{Y}\,.$$

The key difference in this approach is that it allows support vectors to be linear combinations of training data points, leading to a smaller number of support vectors, where each is believed to contain more information and therefore provides a more robust solution.

A number of other approaches have been presented to solve the general problem posed by the structured SVM. The first margin based approach was presented in [Collins, 2002] as a new method for part-of-speech tagging. The algorithm is a generalisation of the original perceptron algorithm that is capable of dealing with problems involving structured output spaces. The structured perceptron works in an online manner, observing an input x_k and making a prediction $\mathbf{y}^* = \mathbf{argmax}_{\mathbf{y}\in\mathcal{Y}}\langle \mathbf{w}, \boldsymbol{\phi}(x_k, \mathbf{y})\rangle$ on the most likely output. If the prediction is incorrect, the model is updated according to

$$\mathbf{w}' \leftarrow \mathbf{w} + \boldsymbol{\phi}(x_k, \mathbf{y}_k) - \boldsymbol{\phi}(x_k, \mathbf{y}^*)\,,$$

which effectively increases the *compatibility* of the predictor with the true labelling $\mathbf{y}_k$ and makes it more likely for the predictor function to return the correct labelling. Recently [Lacoste-Julien et al., 2013] presented the Frank Wolfe algorithm [Frank and Wolfe, 1956] as a general purpose framework for analysing the convergence behaviour of a number of online learning methods for structured SVMs. They derived a block-coordinate version of the algorithm, breaking down the problem into blocks, each associated with constraints over a particular example. One particularly nice feature of this algorithm is that it provides a closed form expression for the optimal step-size, which avoids the problem of performing a

line-search or heuristic method to compute the size of the update step. A further advantage of this approach is that the algorithm obtains a linear rate of convergence in the number of updates made and makes use of a well-defined stopping criterion, namely an upper bound on the duality gap.

Another popular approach to dealing with the exponentially large constraint set is to make a polynomial sized reformulation of the problem. In [Taskar et al., 2004], the authors observed that the dual variables associated with constraints for each example represented a *density like* function over outputs conditional on the input x. This observation enables the optimisation to be reformulated in terms of the marginal dual variables

$$\mu(k, y_i) = \sum_{\mathbf{y}' \sim [y_i]} \alpha_k(\mathbf{y}') \quad \forall\ i \in V,\ \forall\ y_i,\ \ k = 1, \ldots, m$$
$$\mu(k, y_i, y_j) = \sum_{\mathbf{y}' \sim [y_i, y_j]} \alpha_k(\mathbf{y}) \quad \forall\ (i,j) \in E,\ \forall\ y_i, y_j,\ \ k = 1, \ldots, m\ ,$$

where $\mathbf{y}' \sim [y_i]$ refers to the labellings $\mathbf{y}' \in \mathcal{Y}$ such that $y'_i = y_i$, similarly for $\mathbf{y}' \sim [y_i, y_j]$.

The use of the marginal dual variables results in a fixed number of variables and a polynomially large sized optimisation problem, however its size (quadratic in the number of examples and edges in the graph) remains a difficult task for off-the-shelf QP solvers. Furthermore we now have the added difficulty of ensuring that the edge variables form a legal density over $\alpha_k(\mathbf{y})$ i.e. they belong to the marginal polytope of the underlying graph structure. This is a difficult task for general graphs, however for trees the restriction simply amounts to ensuring that by marginalising over the edge variables, we obtain the vertex variables i.e. $\sum_{y_j} \mu(k, y_i, y_j) = \mu(k, y_i)$. To avoid the use of standard QP solvers, an adaptation of the SMO algorithm for SVMs was presented. The optimality criteria is used to select the two most violating variables $\alpha_k(\mathbf{y})$ and $\alpha_k(\mathbf{y}')$, and then solves a one variable quadratic subproblem for $\lambda = \alpha'_k(\mathbf{y}) - \alpha_k(\mathbf{y}) = \alpha_k(\mathbf{y}') - \alpha'_k(\mathbf{y})$, the size of the update. This can be computed in terms of the marginal dual variables.

The efficiency of the large-margin learning algorithms depend heavily on being able to quickly solve the augmented inference problem

$$\mathbf{y}^* = \operatorname*{\mathbf{argmax}}_{\mathbf{y} \in \mathcal{Y}} \langle \mathbf{w}, \boldsymbol{\phi}(x_k, \mathbf{y}) \rangle + \Delta(\mathbf{y}_k, \mathbf{y})$$

Unfortunately, this can only be solved in polynomial time for a limited number of graph structures and in many cases we may have to resort to approximate inference schemes for evaluating constraints and finding update directions. This can lead to the usual difficulties in

evaluating the optimality of a solution and slow rates of convergence. However, recently there has been some work done on examining the convergence of structural SVMs when exact inference is intractable [Finley and Joachims, 2008], which has offered insights on better methods for training structured SVMs using approximate inference. They consider two types of approximate inference; *under-generating* and *over-generating*. The under-generating methods such as greedy search and loopy-belief propagation, simplify the intractable prediction problem by searching only over a subset of possible labelling. Over-generating methods do the opposite and search over a set larger than the possible labellings. For example, through linear programming relaxations they consider real-valued outputs rather than integer valued ones. Difficulties arise in these methods when it comes to ensuring that the distribution over the dual variables remains a valid one.

4.3.2 Structure Learning

During parameter learning we have assumed that the underlying graph structure is known ahead of time. The literature on structure learning for CRFs is relatively sparse and more often than not the structure used for parameter learning is constructed by hand using some trade-off between specialist domain knowledge and the tractability of the resulting graph. Within this literature, and the broader case of structure learning for MRFs, there are two main approaches for learning the structure of a graph using data; the constraint based approach and the score-based approach. The constraint-based approach constructs a set of empirical independence tests on the training sample and then searches for the graph structure that satisfies these implied relationships. The score-based approaches first define a scoring function that measures the quality of a proposed model and then searches among the space of possible models for the one with highest score. Our focus here is on the score-based approaches due to their connections with the work we propose later.

When choosing the hypothesis space in which to search, one must trade off the expressivity of a potential solution with its tractability. For example if we consider the space of tree-structured graphs for MRFs, [Chow and Liu, 1968] showed that it was possible to find the maximum likelihood tree in time quadratic in the number of vertices. The tree structure that maximises the log-likelihood corresponds to the same one that maximises the sum of the mutual information across the edges, where the mutual information between two variables Y_i and Y_j is given by

$$I(Y_i, Y_j) := \sum_{y_i \in \mathcal{Y}_i} \sum_{y_j \in \mathcal{Y}_j} P(y_i, y_j) \log \left(\frac{P(y_i, y_j)}{P(y_i)P(y_j)} \right)$$

The Chow-Liu tree algorithm finds the tree-structure that maximises the likelihood of the observed data. This is an important result that tells us we can find the tree-structure and

parameters that maximise the likelihood of a given data sample in only quadratic time. Although real-world problems rarely fit into this sparse representation and tree-structured models are likely to miss out on important, higher-order interactions between variables. Naturally one would like to extend this algorithm to include the set of low tree-width graphs [Bach and Jordan, 2001], however [Srebro, 2001] showed that it is NP-hard to find the optimal bounded tree-width graph structure when we consider tree-widths greater than one. In [Bradley and Guestrin, 2010], a conditional random field variant of the Chow-Liu tree was presented, however they alluded to the difficulties associated with computing these trees, noting that it becomes increasingly intractable as the size of the input space increases.

Algorithm 3 Chow-Liu Tree Algorithm

1: **Input:** $S = (\mathbf{y}_1, \ldots, \mathbf{y}_m)$,complete graph $G = (V, E)$
2: **for** each $(i, j) \in E$
3: Compute mutual information $w_{ij} = I(Y_i; Y_j)$
4: **end for**
5: For graph G and edge weights $\mathbf{w}$, compute maximum weighted spanning tree T
6: **Return** T

When considering possible score-function candidates, the likelihood function seems like an obvious choice, however we quickly encounter the familiar problem of overfitting since the likelihood can be arbitrarily improved by increasing the expressivity of the graph structure permitted. Various regularisation techniques have been borrowed from traditional supervised learning literature to address this issue; L_2-norm and L_1-norm regularisation, Bayesian Information criteria and Laplacian approximations being popular approaches. One of the most interesting of these approaches is arguably the use of the L_1 regulariser, which has been used in the Lasso model [Tibshirani, 1996] for regression to encourage sparsity amongst the model parameters. This was introduced to MRFs in [Lee et al., 2006; Riezler and Vasserman, 2004], and although proposed in [Lee et al., 2006], the first application of the block L_1 regularisation on conditional random fields was presented in [Schmidt et al., 2008]. The block L_1 regulariser encourages sparsity across the inclusion of edges into the learned model in the hope that it will encode a tractable graph structure. This approach avoids the need to perform a combinatorial approach to feature selection and guarantees to find a globally optimal solution at convergence, however the usual difficulties associated with the calculation of the gradient still exist in the run-up to convergence. The L_1 regularisation encourages sparsity in the model, however it does not necessarily result in tractable models. In [Meshi et al., 2013], the authors proposed a novel regulariser, the circuit rank regulariser, for learning large margin tree predictors. This regulariser directly penalises non-tree structures and results in an optimisation procedure that can be solved using a convex-concave procedure.

The inherent difficulties associated with learning the structure from data has led to a number of studies where the underlying graph structure is approximated using a random sample of tractable subgraphs. In [Pletscher et al., 2009] the authors used a mixture of random spanning trees to approximate an intractable graph and use it to compute the maximum likelihood estimate of the CRF parameters. This approach benefits from the tractability of the tree distributions, which results in efficient gradient calculations during training over the approximated graph structure. This work is similar to the approach taken in [Meila and Jordan, 2000] for learning a MRF using a mixture of trees model. Here, the authors assumed that there was some hidden variable responsible for explaining some of the variability across the data samples. The main distinction between these works is that [Pletscher et al., 2009] focus on CRFs rather than MRFs, and that the spanning trees it uses are sampled uniformly at the beginning. In [Meila and Jordan, 2000], a fixed number of trees are used, however during each expectation-maximisation [Dempster et al., 1977] step of the optimisation algorithm, each tree is adjusted by running a Chow-Liu algorithm using a weighted training sample. These weights represent the expected probability that an observation belongs to a particular tree component, and the weighted observations are used to compute the required mutual information values.

From the large margin perspective a similar approach was presented in [Su and Rousu, 2013]. A random set of graph structures is sampled and an individual predictor is trained over each structure. The motivation behind this work was to allow the random graph structures to capture different structural elements of the data. These relationships could efficiently be learned by sampling from a family of tractable graph structures. These individually trained predictors were treated as an ensemble and a number of different aggregation strategies were proposed to maximise the accuracy of the predictor. A more theoretical approach using a set of random spanning trees was presented in [Marchand et al., 2014]. The authors showed that with a relatively small number of spanning trees, they were capable of obtaining a large proportion of the margin that is achievable when learning a predictor over the complete output graph. In contrast to [Su and Rousu, 2013], the parameters over these trees were trained together. To address the inference problem associated with a non-tree structure, [Marchand et al., 2014] performed K-best inference on each of the trees and provided conditions under which we can be assured of exact inference over the combined structure. This work provides the main source of inspiration for the method that we propose in the next section. The main difference between our proposed method and that presented in [Marchand et al., 2014] is that we consider the space of all possible spanning trees, showing that a sparse combination of spanning trees can obtain a margin that is at least as large as that obtainable when learning takes place over the complete graph.

4.4 Large margin multi-label learning on graphs

In this section we focus on the problem of multi-label learning where we must learn a function that maps arbitrary inputs to binary outputs defined on a graph of unknown structure. We have shown graphical models to be powerful tools for representing complex dependencies between random variables. There was however a trade-off between the complexity of the underlying graph structure and our ability to efficiently solve inference and learning tasks. We saw that tree-structured graphs could solve these tasks efficiently, however they rarely represent the relationships we see in real-world problems. In this section we adopt the view of representing more complex graphs using a combination of spanning trees. This is a well known approach for both inference [Wainwright et al., 2001] and parameter learning [Marchand et al., 2014; Meila and Jordan, 2000; Pletscher et al., 2009]. However, unique about our approach is the representation of the problem in terms of multiple kernel learning over the set of all spanning trees, showing that we can efficiently consider exponentially many kernels when learning a large margin predictor. The advantage of this approach is that it is agnostic to graph structure and is capable of efficiently learning the structure that leads to the maximum margin over the dataset. Furthermore, we show the weighted combination of spanning trees permits tractable inference, learns a margin at least as large as that obtainable over the complete graph and can be solved using a convex optimisation procedure. We begin with the introduction of some notation and then proceed to discuss various aspect of the optimisation procedure.

4.4.1 Definitions and Assumptions

We consider the general supervised learning problem between an arbitrary input space $\mathcal{X}$ and a complex output space $\mathcal{Y}$, which is the set of all ℓ dimensional vectors $\mathbf{y} = (y_1, \ldots, y_\ell)$ where each $y_i \in \{0, 1\}$ is a binary variable. The space of all possible outputs is of size $|\mathcal{Y}| = 2^\ell$. Each example $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ is mapped to a joint feature space $\mathcal{H}$ by the mapping $\boldsymbol{\phi} : \mathcal{X} \times \mathcal{Y} \to \mathcal{H}$. We consider the case where the joint feature map $\boldsymbol{\phi}$ corresponds to the Markov random field defined by the graph $G = (V, E)$, which has ℓ vertices and $\binom{\ell}{2}$ undirected edges. Each vertex i corresponds to a particular output variable y_i and between each pair of output variables (y_i, y_j) there exists an edge $(i, j) \in E$. This is a particular instance of a complete graph, and we will often refer to it as the complete output graph.
We assume that the feature mapping for any input-output pair $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ is given by

$$\boldsymbol{\phi}(x, \mathbf{y}) = (\boldsymbol{\phi}_{ij}(x, y_i, y_j))_{(i,j)\in G} = (\boldsymbol{\rho}(x) \otimes \boldsymbol{\psi}_{i,j}(y_i, y_j),)_{(i,j)\in G} \tag{4.9}$$

where $\boldsymbol{\rho} : \mathcal{X} \to \mathbf{P}$ and $\boldsymbol{\psi} : \mathcal{Y} \to \boldsymbol{\Psi}$ are feature mappings for the input and output space, respectively, and $\otimes$ corresponds to the Kronecker product. Given that each output variable y_i is binary it means that each possible edge $(i, j) \in G$ can take on one of four possible values

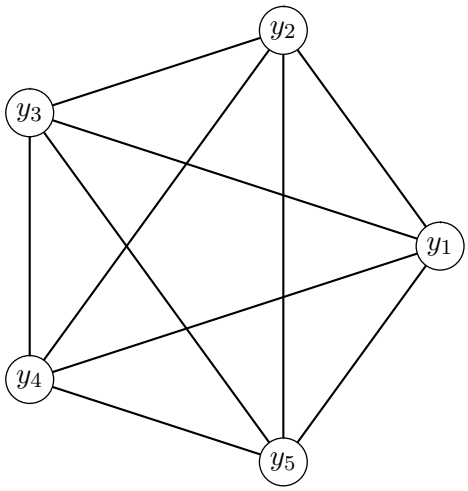


Fig. 4.4 Complete output graph: every node $i \in \{1, \ldots, 5\}$ is connected to every other $j \in \{1, \ldots, 5\} \setminus i$. There are $\binom{\ell}{2}$ edges in total.

and the output feature mappings for a particular edge labelling (y_j, y_j) corresponds to

$$\boldsymbol{\psi}_{i,j}(y_i, y_j) = \begin{pmatrix} \mathbb{1}[(y_i, y_j) = (0,0)] \\ \mathbb{1}[(y_i, y_j) = (0,1)] \\ \mathbb{1}[(y_i, y_j) = (1,0)] \\ \mathbb{1}[(y_i, y_j) = (1,1)] \end{pmatrix} \quad \text{and} \quad \boldsymbol{\phi}_{i,j}(x, y_i, y_j) = \boldsymbol{\rho}(x) \otimes \boldsymbol{\psi}_{i,j}(y_i, y_j)\,,$$

and the joint feature vector $\boldsymbol{\phi}(x, \mathbf{y})$ is given by the concatenation of all $\binom{\ell}{2}$ individual edge feature vectors. Given a weight vector $\mathbf{w}$, the predicted output $\mathbf{y}_{\mathbf{w}}(x)$ at input $x \in \mathcal{X}$ is given by the output $\mathbf{y} \in \mathcal{Y}$ that maximises the score $F(x, \mathbf{y}; \mathbf{w})$ i.e. it solves the MAP inference problem

$$\mathbf{y}_{\mathbf{w}}(x) := \operatorname*{\mathbf{argmax}}_{\mathbf{y} \in \mathcal{Y}} F(x, \mathbf{y}; \mathbf{w}) \quad \text{where} \quad F(x, \mathbf{y}; \mathbf{w}) := \langle \mathbf{w}, \boldsymbol{\phi}(x, \mathbf{y}) \rangle\,.$$

The weight vector $\mathbf{w}$ can be written as $\mathbf{w} = (\mathbf{w}_{i,j})_{(i,j) \in G}$, where $\mathbf{w}_{i,j}$ is the weight on feature mapping $\phi_{i,j}(x, y_i, y_j)$. The score function $F(x, \mathbf{y}; \mathbf{w})$ represents the compatibility of labelling $\mathbf{y} \in \mathcal{Y}$ for input $x \in \mathcal{X}$ and represented as a sum of edge compatibilities $F_{i,j}$,

$$F(x, \mathbf{y}; \mathbf{w}) = \langle \mathbf{w}, \boldsymbol{\phi}(x, \mathbf{y}) \rangle = \sum_{(i,j) \in G} \langle \mathbf{w}_{i,j}, \phi_{i,j}(x, y_i, y_j) \rangle = \sum_{(i,j) \in G} F_{i,j}(x, y_i, y_j; \mathbf{w}_{i,j}).$$

The margin $\Gamma(x, \mathbf{y}; \mathbf{w})$ achieved by predictor $\mathbf{w}$ on input-output pair $(x, \mathbf{y})$ is given by

$$\Gamma(x, \mathbf{y}; \mathbf{w}) := \min_{\mathbf{y}' \neq \mathbf{y}} \left[F(x, \mathbf{y}; \mathbf{w}) - F(x, \mathbf{y}'; \mathbf{w}) \right]$$

If we assume that the distribution of the output variables $\mathbf{y}$ conditioned on the observation of some input $x \in \mathcal{X}$ can be modelled using the exponential family. Therefore the margin on a particular example corresponds to the log ratio of the probabilities of the true labelling and that of any other labelling i.e.

$$\min_{\mathbf{y}' \neq \mathbf{y}} \log \frac{P(\mathbf{y}|x, \mathbf{w})}{P(\mathbf{y}'|x, \mathbf{w})} = \min_{\mathbf{y}' \neq \mathbf{y}} \left[\langle \mathbf{w}, \boldsymbol{\phi}(x, \mathbf{y}) \rangle - \langle \mathbf{w}, \boldsymbol{\phi}(x, \mathbf{y}') \rangle \right] = \Gamma(x, \mathbf{y}; \mathbf{w})$$

and maximising the margin corresponds to maximising the difference between the probability of the true output labelling $\mathbf{y}$ and any other labelling $\mathbf{y}' \in \mathcal{Y} \setminus \mathbf{y}$.

For any vector $\mathbf{a}$, let $||\mathbf{a}||$ and $||\mathbf{a}||_1$ denote its L_2 and L_1 norm, respectively. We assume that we are working with a normalised joint feature map such that $||\boldsymbol{\phi}(x, \mathbf{y})|| = 1$ for all $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ and that $||\phi_{i,j}(x, y_i, y_j)||$ is the same for each $(i, j) \in G$. Given that we are working with a complete graph $|E| = \binom{\ell}{2}$ and therefore $||\phi_{i,j}(x, y_i, y_j)|| = \binom{\ell}{2}^{-1}$. We assume that we have been given a labelled training set $\mathcal{S} = \{(x_k, \mathbf{y}_k)\}_{k=1}^m$, which have been sampled identically and independently from some unknown distribution over $\mathcal{X} \times \mathcal{Y}$.

4.4.2 Superposition of spanning trees

In this section we show how the score function for a complete output graph can be represented as the expectation of the scores over its spanning trees. This allows us to represent the complete output graph using a weighted combination of spanning trees, which leads to our large margin formulation of the predictor as a unit-L_1 norm combination of spanning trees. This work follows from the presentation given in [Marchand et al., 2014].

Suppose that we have a complete graph G of ℓ nodes representing a Markov random field. Let $S(G)$ represent the set of $\ell^{\ell-2}$ spanning trees of G. Each spanning tree has $(\ell - 1)$ edges, which results in each edge appearing in a total of $\ell^{\ell-2}(\ell-1)/\binom{\ell}{2} = (2/\ell)\ell^{\ell-2}$ trees in $S(G)$. Therefore, we can represent any function f defined using the edges of G according to

$$\sum_{T \in S(G)} \sum_{(i,j) \in T} f((i,j)) = \frac{2}{\ell}\, \ell^{\ell-2} \sum_{(i,j) \in G} f((i,j))$$

Let T be any spanning tree of G and $\mathbf{w}$ any predictor, we define the projection of the $\mathbf{w}$ on to tree T according to

$$(\mathbf{w}_T)_{i,j} = \begin{cases} \mathbf{w}_{i,j} & \text{if} \quad (i,j) \in T \\ 0 & \text{otherwise} \end{cases}.$$

The projection of the joint feature mapping $\boldsymbol{\phi}$ on to tree T is defined analogously. The feature

mapping defined on each edge is assumed to have the same norm, $||\phi_{i,j}(x, y_i, y_j)||^2 = \binom{\ell}{2}^{-1}$, which results in projected feature mappings with norms given by

$$||\phi_T(x, \mathbf{y})||^2 = \sum_{(i,j)\in T} ||\phi_{i,j}(x, y_i, y_j)||^2 = \frac{\ell - 1}{\binom{\ell}{2}} = \frac{2}{\ell}\,.$$

Lemma 9 ([Marchand et al., 2014]) *Let* $\hat{\mathbf{w}}_T = \mathbf{w}_T/||\mathbf{w}_T||$, $\hat{\boldsymbol{\phi}}_T = \boldsymbol{\phi}_T/||\boldsymbol{\phi}_T||$. *Let* $\mathcal{U}(G)$ *denote the uniform distribution of spanning trees on* $S(G)$*. Then we have that*

$$F(x, \mathbf{y}; \mathbf{w}) = \mathop{\mathbb{E}}_{T\sim\mathcal{U}(G)} a_T \langle \hat{\mathbf{w}}_T, \hat{\boldsymbol{\phi}}_T(x, \mathbf{y})\rangle \quad \textit{where} \quad a_T = \sqrt{\frac{\ell}{2}}||\mathbf{w}_T|| \;.$$

Moreover, for any $\mathbf{w}$ such that $||\mathbf{w}|| = 1$, we have:

$$\mathop{\mathbb{E}}_{T\sim\mathcal{U}(G)} a_T^2 = 1\,, \quad \text{and} \quad \mathop{\mathbb{E}}_{T\sim\mathcal{U}(G)} a_T \leq 1 \;.$$

Let $\mathcal{T} = \{T_1, \ldots, T_n\}$ define a sample of n spanning trees of G, where each tree T is sampled independently from the uniform distribution $\mathcal{U}(G)$ over $S(G)$. By considering the implications of the unit L_2-norm constraints on the weights a_T on the trees, they considered the unit L_2-norm conical combination $(\mathcal{W}, \mathbf{q})$ of each weight $\mathcal{W} = \{\hat{\mathbf{w}}_{T_1}, \ldots, \hat{\mathbf{w}}_{T_n}\}$ realised by the n-dimensional weight vector $\mathbf{q} = (q_1, \ldots, q_n)$, where $\sum_{i=1}^n q_i^2 = 1$. The score function under this conical combination $(\mathcal{W}, \mathbf{q})$ is given by

$$F_{\mathcal{T}}(x, \mathbf{y}, \mathcal{W}, \mathbf{q}) = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} q_i \langle \hat{\mathbf{w}}_{T_i}, \hat{\boldsymbol{\phi}}_{T_i}(x, \mathbf{y})\rangle$$

By assuming that there exists some predictor $\mathbf{w}$ defined over the complete graph that achieves margin of $\gamma > 0$ on all $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, [Marchand et al., 2014] showed with high probability, there exists a conical combination $(\mathcal{W}, \mathbf{q})$ that obtains a significant proportion of this margin. This resulted in an optimisation problem that aimed at maximising the margin obtainable by a conical combination of a randomly sampled set of spanning trees $\mathcal{T}$, and thus finding the predictor defined over the trees $\mathcal{T}$ that has the best generalisation guarantee. Rather than focusing on the implications of the unit L_2-norm for weights in Lemma 9, we concentrate on the L_1-norm constraints.

By turning the expectation into a sum over all spanning trees we see that the score function can be represented by

$$F(x, \mathbf{y}; \mathbf{w}) = \sum_{T\in S(T)} \hat{a}_T \langle \hat{\mathbf{w}}_T, \hat{\boldsymbol{\phi}}_T(x, \mathbf{y})\rangle \,,$$

where $\hat{a}_T = \sqrt{\frac{\ell}{2}}||\mathbf{w}_T||/\ell^{\ell-2}$ and $\sum_{T\in S(T)} \hat{a}_T \leq 1$. This is the weighted sum of the inner products between weight vectors and feature vectors that have been projected onto the spanning trees $T \in S(G)$, where the weights reside within the unit L_1-norm.

If we assume that there exists a predictor $\mathbf{w}$ obtaining $\gamma > 0$ for each $(x, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$, we observe that this can be achieved by assigning a weight $\hat{a}_T$ to each spanning tree $T \in S(G)$. When we express the large margin learning problem in the dual space, we see that each tree can be represented using a kernel, which receives a weight $\hat{a}_T$ in the score function. This particular weighting corresponds to a feasible solution for the L_1-norm multiple kernel learning (MKL) problem, however it is not necessarily the optimal one. This opens up the possibility of finding a weighting over the set of spanning trees that achieves a margin that is at least as large as that obtainable when learning takes place over the complete output graph.

4.4.3 L_1-Norm Multiple Kernel Learning for Spanning Trees

We consider an arbitrary set $\mathcal{W} = \{\hat{\mathbf{w}}_T\}_{T\in S(G)}$ of unit L_2-norm weight vectors where $\hat{\mathbf{w}}_T$ operate on the unit L_2-norm feature vector $\hat{\boldsymbol{\phi}}_T(x, \mathbf{y})$. Rather than using a conical combination, as in [Marchand et al., 2014], we consider a convex combination of feature weights in $\mathcal{W}$ realised by an $\ell^{\ell-2}$-dimensional vector $\mathbf{q} = \{q_T\}_{T\in S(G)}$. The score function is given by

$$F(x, \mathbf{y}, \mathcal{W}, \mathbf{q}) = \sum_{T\in S(G)} q_T \langle \mathbf{w}_T, \boldsymbol{\phi}_T(x, \mathbf{y})\rangle \quad \text{where} \quad \sum_{T\in S(G)} q_T \leq 1\,.$$

We drop the assumption that the data is perfectly separable with a hard margin $\gamma > 0$ and opt instead for the popular soft-margin approach. Following the standard soft-margin framework for constructing large margin predictors [Cortes and Vapnik, 1995], the optimisation problem is given in Definition 2.

Definition 2 *Large margin learning over G*

$$\begin{aligned} &\min_{\boldsymbol{\xi},\gamma,\mathbf{q},\mathcal{W}} \quad \frac{1}{2\gamma^2} + \frac{C}{\gamma}\sum_{k=1}^{m} \xi_k \\ &s.t. \ \min_{\mathbf{y}\neq\mathbf{y}_k} \sum_{T\in S(G)} q_T \langle \hat{\mathbf{w}}_T, \hat{\boldsymbol{\phi}}_T(x_k, \mathbf{y}_k) - \hat{\boldsymbol{\phi}}_T(x_k, \mathbf{y})\rangle \geq \gamma - \xi_k, \\ &\xi_k \geq 0, \ \forall \ k, \ \sum_{T\in\mathcal{T}} q_T = 1, \ q_T \geq 0 \ \forall \ T \in S(G). \end{aligned} \tag{4.10}$$

Due to an arbitrary scaling of $\mathbf{q}$ we will always end up with a solution where the weights $\mathbf{q}$ reside on the unit L_1-norm ball, therefore we are able to drop the inequality on the sum of the tree weights and replace it with an equality. By making the substitutions $\zeta_k = \xi_k/\gamma$ and

$\mathbf{v}_T = q_T \hat{\mathbf{w}}_T / \gamma$ in Definition 2 and observing that $(\sum_{T \in S(G)} ||\mathbf{v}_T||)^2 = 1/\gamma^2$, we see that we can maximise the size of the margin by solving the optimisation problem in Definition 3.

Definition 3 *L_1-norm MKL for Spanning Trees*

$$\min_{\mathbf{v}_T, \zeta} \frac{1}{2} \left(\sum_{T \in S(G)} ||\mathbf{v}_T|| \right)^2 + C \sum_{k=1}^{m} \zeta_k$$
$$s.t. \min_{\mathbf{y} \neq \mathbf{y}_k} \sum_{T \in S(G)} \langle \mathbf{v}_T, \hat{\boldsymbol{\phi}}_T(x_k, \mathbf{y}_k) - \hat{\boldsymbol{\phi}}_T(x_k, \mathbf{y}) \rangle \geq 1 - \zeta_k \ , \zeta_k \geq 0 \ \forall \ k,$$

where $\mathbf{v}_T$ are the weight vectors for each tree, ζ_k is the slack variable for example $(x_k, \mathbf{y}_k)$ and $C > 0$ is the slack parameter that controls the complexity of the solution. This definition is almost identical to the multiple kernel learning (MKL) formulation first introduced in [Bach et al., 2004] albeit with a different set of constraints related to the structured prediction nature of the problem. To help make our expressions clear, and as familiar as possible, we introduce $\tilde{\boldsymbol{\phi}}_T(x_k, \mathbf{y}_r) = \hat{\boldsymbol{\phi}}_T(x_k, \mathbf{y}_k) - \hat{\boldsymbol{\phi}}_T(x_k, \mathbf{y}_r)$ to denote the difference between the joint feature vectors, and make a slight abuse of notation by setting $\mathbf{w}_T \leftarrow \mathbf{v}_T$, $\xi_k \leftarrow \zeta_k$ and $\boldsymbol{\phi}_T \leftarrow \hat{\boldsymbol{\phi}}_T$. For the remainder of this paper we use the MKL formulation given in Definition 4 first presented in [Rakotomamonjy et al., 2008], which was shown to be equivalent to that given in Definition 3.

Definition 4 *L_1-norm MKL for Spanning Trees (equivalence)*

$$\min_{\mathbf{w}_T, \boldsymbol{\xi}, \boldsymbol{\lambda}} \quad \frac{1}{2} \sum_{T \in S(G)} \frac{1}{\lambda_T} ||\mathbf{w}_T||^2 + C \sum_k \xi_k \tag{4.11}$$
$$s.t. \sum_{T \in S(G)} \lambda_T = 1, \ \ \lambda_T \geq 0 \ \forall \ T \in S(G),$$
$$\min_{\mathbf{y} \neq \mathbf{y}_k} \sum_{T \in S(G)} \langle \mathbf{w}_T, \tilde{\boldsymbol{\phi}}_T(x_k, \mathbf{y}) \rangle \geq 1 - \xi_k, \ \ \xi_k \geq 0 \ \forall k. \tag{4.12}$$

In this formulation we are able to use the L_2-norms of the projected weight vectors by dividing them by λ_T, which are the weights attributed to each tree $T \in S(G)$. This has the effect of controlling the squared norm of the weight vector assigned to tree T, which results in a smooth and convex optimisation problem [Rakotomamonjy et al., 2008]. To see this, consider the case where $\lambda_T = 0$, to obtain a finite objective function we must also set $\mathbf{w}_T$ to zero, otherwise $||\mathbf{w}_T||^2/\lambda_T$ would yield an infinite value and clearly not minimise the expression. Similarly, if $||\mathbf{w}_T||$ was large we need λ_T also to be large to suppress the contribution of this weight vector to the objective function. To solve this optimisation we begin by introducing Lagrange multipliers for the constraints. Note that we substitute the minimum inequality regarding score differences with an inequality for each labelling $\mathbf{y} \in \mathcal{Y}$, which is effectively the same due to the single slack variable ξ_k being shared across each

possible labelling $\mathbf{y} \in \mathcal{Y}$. Furthermore, we introduce the zero-one loss function $\Delta(\mathbf{y}_k, \mathbf{y})$ into our expression to make it as generic as possible. Therefore the Lagrangian is given by

$$\mathcal{L}(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\lambda}, \boldsymbol{\nu}, \boldsymbol{\mu}, \omega) := \frac{1}{2} \sum_{T \in S(G)} \frac{1}{\lambda_T} ||\mathbf{w}_T||^2 + C \sum_k \xi_k - \sum_k \mu_k \xi_k + \omega \left(\sum_{T \in S(G)} \lambda_T - 1 \right) - \sum_{T \in S(G)} \nu_T \lambda_T - \sum_{k, \mathbf{y}} \alpha_k(\mathbf{y}) \left(\sum_{T \in S(G)} \langle \mathbf{w}_T, \tilde{\boldsymbol{\phi}}_T(x_k, \mathbf{y}) \rangle - \Delta(\mathbf{y}_k, \mathbf{y}) + \xi_k \right),$$

where the dual variables $\alpha_k(\mathbf{y})$ correspond to the margin based constraints for labelling $\mathbf{y} \in \mathcal{Y}$ on example $(x_k, \mathbf{y}_k)$. The other Lagrange multipliers μ_k ensures the non-negativity of the slack variables ξ_k, the non-negativity of kernel weights is controlled by ν_T and to ensure their unitary sum we use ω. Differentiating with respect to the primal variables we observe that for primal optimality we require that

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_T} = 0 \ , \ \frac{\partial \mathcal{L}}{\partial \lambda_T} = 0 \ \text{and} \ \frac{\partial \mathcal{L}}{\partial \xi_k} = 0 \,, \tag{4.13}$$

and as a result of this we know that

- the sum of the dual variables for each example $(x_k, \mathbf{y}_k)$ is upper bounded by the regularisation term C,

$$\sum_{\mathbf{y}} \alpha_k(\mathbf{y}) = C - \mu_k \,. \tag{4.14}$$

- the weight vector attributed to each tree $\mathbf{w}_T$ is given by a linear combination of the training samples, as implied by the representer theorem, and scaled according to the weight attached to that tree λ_T

$$\mathbf{w}_T = \lambda_T \sum_{\mathbf{y}} \alpha_k(\mathbf{y}) \tilde{\boldsymbol{\phi}}_T(x_k, \mathbf{y}) \tag{4.15}$$

- the weight attributed to each tree λ_T relates to the norm of the corresponding tree weight vector $\mathbf{w}_T$

$$\lambda_T = \frac{||\mathbf{w}_T||}{\sqrt{2(\omega - \nu_T)}} \tag{4.16}$$

The dual variables $\alpha_k(\mathbf{y})$ are used throughout the construction of weight vector $\mathbf{w}$, where

$\alpha_k(\mathbf{y}) > 0$ tells us that a margin of $\Delta(\mathbf{y}_k, \mathbf{y})$ is not obtained on that particular labelling

$$\alpha_k(\mathbf{y}) > 0 \implies \sum_{T \in S(G)} \langle \mathbf{w}_T, \tilde{\boldsymbol{\phi}}_T(x_k, \mathbf{y}) \rangle = \Delta(\mathbf{y}_k, \mathbf{y}) - \xi_k \,,$$
$$\alpha_k(\mathbf{y}) = 0 \implies \sum_{T \in S(G)} \langle \mathbf{w}_T, \tilde{\boldsymbol{\phi}}_T(x_k, \mathbf{y}) \rangle \geq \Delta(\mathbf{y}_k, \mathbf{y}) - \xi_k \,.$$

As a consequence of the optimality conditions for slack variables, we see that

$$\xi_k > 0 \implies \mu_k = 0 \quad \text{and} \quad \sum_{\mathbf{y}} \alpha_k(\mathbf{y}) = C \,.$$

This tells us that we have used all of the weight afforded to example $(x_k, \mathbf{y}_k)$ in order to try and ensure the margin condition for each labelling $\mathbf{y} \in \mathcal{Y}$ is satisfied.

Examining the optimality conditions for the kernel weights λ_T, we see that if a tree has non-zero weight $\lambda_T > 0$ then $\nu_T = 0$. When we refer to the *unscaled* norm of $\mathbf{w}_T$, we mean its value prior to scaling by λ_T i.e.

$$\frac{||\mathbf{w}_T||}{\lambda_T} = \sqrt{\sum_{k,\mathbf{y}} \sum_{j,\mathbf{y}'} \alpha_k(\mathbf{y}) \alpha_j(\mathbf{y}') \tilde{K}_T(x_k, \mathbf{y}; x_j, \mathbf{y}')} \,.$$

Using 4.16, we see that each tree with positive weight must have the same unscaled norm and those with zero weight must have an unscaled norm that is less than or equal to those with positive weight,

$$\frac{1}{2} \sum_{k,\mathbf{y}} \sum_{j,\mathbf{y}'} \alpha_k(\mathbf{y}) \alpha_j(\mathbf{y}') \tilde{K}_T(x_k, \mathbf{y}; x_j, \mathbf{y}') = \omega \quad \forall \quad \{T \in S(G) : \lambda_T > 0\} \,, \tag{4.17}$$
$$\frac{1}{2} \sum_{k,\mathbf{y}} \sum_{j,\mathbf{y}'} \alpha_k(\mathbf{y}) \alpha_j(\mathbf{y}') \tilde{K}_T(x_k, \mathbf{y}; x_j, \mathbf{y}') \leq \omega \quad \forall \quad \{T \in S(G) : \lambda = 0\} \,. \tag{4.18}$$

Substituting the primal optimality conditions back into the Lagrangian it transforms into the dual optimisation given in Definition 5. This dual problem is difficult to optimise due to the last constraint, which involves a quadratic constraint for each kernel. This difficulty arises before we even consider the problems associated with optimising over an exponentially large set of kernels.

Definition 5 *L_1-norm MKL for spanning trees (dual formulation)*

$$\min_{\omega,\boldsymbol{\alpha}} \quad \omega - \sum_{k,\mathbf{y}} \alpha_k(\mathbf{y})\Delta(\mathbf{y}_k,\mathbf{y}) \tag{4.19}$$

$$s.t. \quad \sum_{\mathbf{y}} \alpha_k(\mathbf{y}) \leq C \quad \forall \quad k = \{1,\ldots,m\} \tag{4.20}$$

$$\frac{1}{2}\sum_{k,\mathbf{y}}\sum_{j,\mathbf{y}'} \alpha_k(\mathbf{y})\alpha_j(\mathbf{y}')\tilde{K}_T(x_k,\mathbf{y};x_j,\mathbf{y}') \leq \omega \quad \forall \quad T \in S(G) \tag{4.21}$$

One approach to address the quadratic constraint on kernels would be to bring the expression 4.21 into the objective function [Bach et al., 2004], however this results in the non-differentiability of the objective function. Instead we opt for an alternating optimisation scheme, which is similar to the one used in [Argyriou et al., 2008] for multi-task feature learning. In the first step, we fix the set of spanning trees $\mathcal{T} = \{T_1,\ldots,T_n\}$ and weights $\boldsymbol{\lambda} = \{\lambda_{T_1},\ldots,\lambda_{T_n}\}$, and then solve the associated structured SVM problem. In the second step we update the weights of the kernels $\boldsymbol{\lambda}$ in a direction that decreases the value of the objective function, whilst keeping the dual variables $\boldsymbol{\alpha}$ fixed. During this step we consider the inclusion and removal of trees from $\mathcal{T}$. These two steps are iterated until we reach the required degree of convergence and the meta-algorithm is presented in Algorithm 5. To solve the optimisation efficiently we must address two key issues; the first is the *argmax* problem of finding the maximum violator and ensuring the constraints are satisfied, and the second is the identification of kernels that have non-zero weight in the solution. We note that both the space of all multi-labels and the space of all spanning trees are exponential in size, and in both cases we are prevented from exhaustively enumerating over them.

4.4.4 Efficient inference and constraint satisfaction

We begin by examining the inference task that is necessary to check whether the margin constraints in Definition 4 for the structured SVM are satisfied and to predict the labelling of an unseen test input. We assume that the set of trees $\mathcal{T} = \{T_1,\ldots,T_n\}$ and weights $\boldsymbol{\lambda} = (\lambda_{T_1},\ldots,\lambda_{T_n})$ are fixed. The dual representation of the weight vector tells that the score function can be written as a linear combination of scores defined over trees

$$F(x,\mathbf{y},\mathbf{w}) = \sum_{T\in\mathcal{T}} \langle \mathbf{w}_T, \boldsymbol{\phi}_T(x,\mathbf{y})\rangle = \sum_{T\in\mathcal{T}} F_T(x,\mathbf{y},\mathbf{w}_T) = \sum_{T\in\mathcal{T}} \lambda_T \hat{F}_T(x,\mathbf{y},\mathbf{w}_T),$$

where $\hat{F}_T(x,\mathbf{y}) = F_T(x,\mathbf{y})/\lambda_T$, is the score function evaluated on tree T when we remove the scaling factor caused by the kernel weight λ_T. For each example $(x_k,\mathbf{y}_k)$ there is a margin constraint for each possible labelling $\mathbf{y} \in \mathcal{Y}$. Fortunately only a single slack variable is required for each training example and we only have to concern ourselves with searching for the maximally violating labellings, namely the labelling $\mathbf{y}_k^* = \mathbf{argmax}_{\mathbf{y}\in\mathcal{Y}}\,[F(x_k,\mathbf{y};\mathbf{w}) +$

$\Delta(\mathbf{y}_k, \mathbf{y})]$.

It is well known that the exact solution to the inference problem

$$\mathbf{y}_T(x) = \operatorname*{\mathbf{argmax}}_{\mathbf{y}\in\mathcal{Y}} F_T(x, \mathbf{y}; \mathbf{w}_T) = \operatorname*{\mathbf{argmax}}_{\mathbf{y}\in\mathcal{Y}} \hat{F}_T(x, \mathbf{y}; \mathbf{w}_T),$$

on an individual tree $T \in S(G)$ can be obtained in $\mathcal{O}(\ell)$ time by dynamic programming. However, given that we are looking to find the maximum value over a weighted combination of spanning trees, there is no guarantee that the multi-label that maximises the score of $\hat{F}_T$ will also be the maximiser of F. Instead it seems reasonable to expect that the top scoring multi-label will vary across individual trees $T \in \mathcal{T}$. To overcome this we use a variant of the dynamic programming algorithm [Marchand et al., 2014] that finds a K-best list of top scoring multi-labels for each tree $T \in \mathcal{T}$.

Let $\mathcal{Y}_{T,K} = \{\hat{\mathbf{y}}_{T,1}, \ldots, \hat{\mathbf{y}}_{T,K}\}$ represent the set of the K highest scores for $\hat{F}_T(x, \mathbf{y}; \mathbf{w}_T)$, with labellings ordered in descending value such that $F_T(x, \mathbf{y}_1) \geq F_T(x, \mathbf{y}_2) \geq \cdots \geq F_T(x, \mathbf{y}_K)$. Let $\mathcal{Y}_{\mathcal{T},K} = \{\mathcal{Y}_{T,K}\}_{T\in\mathcal{T}}$ to be the set of K-best lists for $T \in \mathcal{T}$. We now state a key lemma that will enable us to verify if the candidate set $\mathcal{Y}_{\mathcal{T},K}$ contains the true maximiser of F.

Lemma 10 *Let* $\mathbf{y}_K^* = \operatorname*{\mathbf{argmax}}_{\mathbf{y}\in\mathcal{Y}_{\mathcal{T},K}} F(x, \mathbf{y}; \mathbf{w})$ *be the highest scoring multi-label in* $\mathcal{Y}_{\mathcal{T},K}$*. Suppose that the score function of labelling* $\mathbf{y}_K^*$ *satisfies*

$$F(x, \mathbf{y}_K^*; \mathbf{w}) \geq \sum_{T\in\mathcal{T}} \lambda_T \hat{F}_T(x, \mathbf{y}_{T,K}; \mathbf{w}_T) = \theta_x(K).$$

Then it follows that $F(x, \mathbf{y}_K^*; \mathbf{w}) = \max_{\mathbf{y}\in\mathcal{Y}} F(x, \mathbf{y}; \mathbf{w})$*.*

Proof *Consider a multi-label* $\mathbf{y}' \notin \mathcal{Y}_{\mathcal{T},K}$*. For each* $T \in S(G)$

$$F_T(x, \mathbf{y}'; \mathbf{w}_T) = \lambda_T \hat{F}_T(x, \mathbf{y}'; \mathbf{w}_T) \leq \lambda_T \hat{F}_T(x, \mathbf{y}_{T,K}; \mathbf{w}_T).$$

Therefore the score weighted and evaluated over all trees is given by

$$F(x, \mathbf{y}'; \mathbf{w}) = \sum_{T\in S(G)} \lambda_T \hat{F}_T(x, \mathbf{y}'; \mathbf{w}_T) \leq \sum_{T\in S(G)} \lambda_T \hat{F}(x, \mathbf{y}_{T,K}; \mathbf{w}_T) \leq F(x, \mathbf{y}_K^*; \mathbf{w}).$$

■

This lemma states that we can use any K that satisfies the criteria above and be sure that $\mathbf{y}_K^*$ is the maximum scoring example on F. It does not however provide guidance in the selection

of the size of the K-best lists that we must construct in order to ensure exact inference has occurred. Note that if we have been unable to satisfy the condition for exact inference we can use $\theta_x(K)$ as an upper bound on the maximum score i.e. $\max_{\mathbf{y}\in\mathcal{Y}} F(x, \mathbf{y}; \mathbf{w}) \leq \theta_x(K)$. This allows us to infer the sub-optimality of the labelling we have chosen to update our model using, making it amenable to the techniques and analysis used in [Finley and Joachims, 2008] for structured SVMs when exact inference is intractable. This upper bound also provides us with advice on how much deeper we have to search in order to obtain exact inference, and we can use the K-th best scores on each tree to target deeper searches on particular trees.

4.4.5 Generating trees efficiently

The second step of the alternating optimisation scheme updates tree weights in a direction to decrease the value of the objective function given in 4.11. We consider an exponentially large set of kernels that correspond to the set of all possible spanning trees over G. Similar to the problem of considering constraints during the structured SVM step of the optimisation we must find an efficient way to verify whether there exists a tree that violates the conditions for optimality outlined in 4.17 and 4.18.

In the traditional MKL setup there is a fixed set of kernels, allowing one to simply iterate between updating kernel weights and solving the structured SVM until the convergence criteria is satisfied. Our problem is more difficult in that we are optimising over an exponentially large set of kernels, $|S(G)| = \ell^{\ell-2}$, making it intractable to consider them all at once. To overcome this we maintain an active set of kernels defined by the set of trees $\mathcal{T} := \{T \ : \ \lambda_T > 0\}$, where the inclusion and exclusion of trees to this set depends on their violation of the conditions for optimality. During the tree weight update stage, we assume that the structured SVM has been solved and let

$$\omega = \frac{1}{2} \sum_{T\in\mathcal{T}} \lambda_T \sum_{k,\mathbf{y}} \sum_{j,\mathbf{y}'} \alpha_k(\mathbf{y})\alpha_j(\mathbf{y}')\tilde{K}_T(x_k, \mathbf{y}; x_j, \mathbf{y}') = \frac{1}{2} \sum_{T\in\mathcal{T}} \lambda_T \, \boldsymbol{\alpha}^T \tilde{\mathbf{K}}_T \boldsymbol{\alpha} \ .$$

This corresponds to the minimum norm predictor that is possible when the output graph is represented using the particular unit L_1-norm combination of spanning trees. The conditions for optimality state that each tree with positive weight should have the same unscaled norm 4.17. This is not taken into consideration during the structured SVM and it is likely that the optimal solution will have trees $T \in \mathcal{T}$ with different norms. To correct this during MKL with a fixed set of trees we would simply re-distribute the weight in favour of the tree that has the largest unscaled norm given the current predictor. This corresponds to the tree weight λ_T with largest gradient. When considering the set of all possible spanning trees, we

must look for the tree $T \in S(G)$ with the largest gradient. This amounts to finding

$$T^* = \underset{T \in S(G)}{\mathbf{argmax}} \sum_{k,\mathbf{y}} \sum_{j,\mathbf{y}'} \alpha_k(\mathbf{y}) \alpha_j(\mathbf{y}') \tilde{K}_T(x_k, \mathbf{y}; x_j, \mathbf{y}').$$

By decomposing the kernel into its edge components we can write the objective as a sum over the edge kernels,

$$\begin{aligned} \sum_{k,\mathbf{y}} \sum_{j,\mathbf{y}'} \alpha_k(\mathbf{y}) \alpha_j(\mathbf{y}') \tilde{K}_T(x_k, \mathbf{y}; x_j, \mathbf{y}') &= \sum_{k,\mathbf{y}} \alpha_k(\mathbf{y}) \left(\hat{F}(x_k, \mathbf{y}_k) - \hat{F}(x_k, \mathbf{y}) \right) \\ &= \sum_{k,\mathbf{y}} \sum_{e \in T} \alpha_k(\mathbf{y}) \left(\hat{F}_e(x_k, \mathbf{y}_k^e) - \hat{F}_e(x_k, \mathbf{y}^e) \right) . \end{aligned}$$

In Algorithm 4 we show how the maximum violating tree is found. This is almost identical to the Chow-Liu tree algorithm, the difference being the criteria on which an edge is evaluated. The Chow-Liu tree algorithm uses the mutual information between variables to define an edge's *importance*, whereas we use the weighted margin on that edge i.e. how much more is the edge score of the true labelling $\mathbf{y}_{k,e}$ than that violating labellings i.e. $\mathbf{y}_e \neq \mathbf{y}_{k,e}$

$$\gamma_e = \sum_{k,\mathbf{y}} \alpha_k(\mathbf{y}) \left(\hat{F}_e(x_k, \mathbf{y}_{k,e}) - \hat{F}_e(x_k, \mathbf{y}_e) \right) .$$

Therefore we see that the search over an exponentially large set of trees can be reduced to a simple implementation of the maximum spanning tree algorithm, and therefore has a running time complexity of $\mathcal{O}(E \log V)$, using weighted margins on the edges, where the goal is to find the tree that maximises $\sum_{e \in T} \gamma_e$.

Recall that a dual variable $\alpha_k(\mathbf{y})$ is only non-zero when the margin criteria is not met. This allows us to interpret the tree with maximum gradient as the one that can potentially reduce margin violations on the training sample. Therefore the tree update scheme can be seen as distributing weight to edges that have the capacity to discriminate between correct and incorrect labellings.

Algorithm 4 Maximum violating tree

1: **Input:** $\mathcal{S} = \{(x_k, \mathbf{y}_k)\}_{k=1}^m$, complete output graph G, dual variables $\alpha_k(\mathbf{y})$ and kernel function $\tilde{K}$
2: **for** each $e \in G$
3: Compute weighted edge margin $\gamma_{\mathbf{e}} = \sum_{k,y} \sum_{j,\mathbf{y}'} \alpha_k(\mathbf{y}) \alpha_j(\mathbf{y}') \tilde{K}_{\mathbf{e}}(x_k, \mathbf{y}^{\mathbf{e}}; x_j, \mathbf{y}'^{\mathbf{e}})$
4: **end for**
5: compute maximum weighted spanning tree T over graph G with edge weights $\boldsymbol{\gamma}$
6: **Return** T

4.4.6 Implementation

The algorithm used for solving the L_1-norm MKL for spanning trees is presented in Algorithm 5. The optimisation procedure works by alternating between solving a structured SVM, using a fixed kernel defined by a set of weighted trees, and updating the weights attributed to each tree $T \in S(G)$. In this section we describe the key components of the optimisation scheme and discuss how they are implemented in practice.

Algorithm 5 Algorithm to find large margin using L_1-norm combination of spanning trees

```
Input: Graph G, labelled training examples S
Randomly sample T spanning trees
while not converged do
    converged = False
    structConv, α = Update dual variables (α, 𝒯)
    mklConv, 𝒯 = Update tree weights (α, 𝒯)
    if structConv and mklConv then
        converged = True
    end if
end while
Return: α, 𝒯
```

Solving the structured svm

This component of the optimisation involves solving a structured SVM where the kernel has been fixed using the current set of active trees and weights. Note that the solution of the structured SVM depends heavily on the tree weightings, and it seems reasonable to expect these weightings to fluctuate quite a lot during the start of the optimisation scheme. Therefore we favour the speed and efficiency of the optimisation scheme over its precision, especially when there is a large duality gap in the MKL component. For these reasons, we apply the Frank-Wolfe algorithm [Frank and Wolfe, 1956; Lacoste-Julien et al., 2013] to the problem of structured SVM, and are similarly mindful of the relationship between the MKL and SVM component during the update of tree weights.

The Frank-Wolfe algorithm FWA [Frank and Wolfe, 1956] was designed for solving convex optimisation problems $\min_{\boldsymbol{\alpha}\in\mathcal{A}} J(\boldsymbol{\alpha})$, where the convex feasible set $\mathcal{A}$ is compact and the objective function J is continuously differentiable. In our case the objective function is given by

$$J(\boldsymbol{\alpha}) := \frac{1}{2}\sum_{k,\mathbf{y}}\sum_{j,\mathbf{y}'}\alpha_k(\mathbf{y})\alpha_j(\mathbf{y}')\tilde{K}(x_k,\mathbf{y};x_j,\mathbf{y}') - \sum_{k,\mathbf{y}}\alpha_k(\mathbf{y})\Delta_k(\mathbf{y})\,, \tag{4.22}$$

where $\tilde{\mathbf{K}} = \sum_{T\in\mathcal{T}}\lambda_T\tilde{\mathbf{K}}_T$. We consider the sparse set of dual variables $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_m)^T$,

where each $\boldsymbol{\alpha}_k$ corresponds to the dual variables of a particular example $k \in \{1, \ldots, m\}$. The feasible set is given by $\mathcal{A} := A_1 \times \cdots \times A_m$, where $A_k = \{\, \boldsymbol{\alpha}_k \,:\, \sum_{\mathbf{y} \in \mathcal{Y}} \alpha_k(\mathbf{y}) = C \,\}$ for each $k = \{1, \ldots, m\}$.

The FWA is an iterative optimisation scheme that works in rounds, solving a linear approximation to the objective function J over a constrained domain $\mathcal{A}$. For structural SVM [Lacoste-Julien et al., 2013], the authors highlighted that the exponential nature of the dual variables makes it much more amenable to solving linear subproblems, rather than a quadratic ones. At each iteration, an update direction $\mathbf{s} \in \mathcal{A}$ is found by searching over the feasible set to find the direction that minimises the linearisation of J at $\boldsymbol{\alpha}$,

$$\mathbf{s}^* = \operatorname*{\mathbf{argmin}}_{\mathbf{s} \in \mathcal{A}} \left[J(\boldsymbol{\alpha}) + \langle \mathbf{s} - \boldsymbol{\alpha}, \nabla J(\boldsymbol{\alpha}) \rangle \right] = \operatorname*{\mathbf{argmin}}_{\mathbf{s} \in \mathcal{A}} \langle \mathbf{s} - \boldsymbol{\alpha},\, \nabla J(\boldsymbol{\alpha}) \rangle \,,$$

where $\nabla J(\boldsymbol{\alpha})$ is the gradient of J. During this step the algorithm effectively computes a linearisation of the duality gap given by

$$g(\boldsymbol{\alpha}) := \max_{\mathbf{s}' \in \mathcal{A}} \langle \boldsymbol{\alpha} - \mathbf{s}', \nabla J(\boldsymbol{\alpha}) \rangle = \langle \boldsymbol{\alpha} - \mathbf{s}^*, \nabla J(\boldsymbol{\alpha}) \rangle \,,$$

since we know that the optimal objective function is bounded according to $J(\boldsymbol{\alpha}^*) \geq J(\boldsymbol{\alpha}) - g(\boldsymbol{\alpha})$. The linearisation of the J, coupled with the block-wise constraints, amounts to finding individual search directions $\mathbf{s}_k \in A_k$ for each example $k \in \{1, \ldots, m\}$

$$\operatorname*{\mathbf{argmin}}_{\mathbf{s} \in \mathcal{A}} \langle \mathbf{s}, \nabla J(\boldsymbol{\alpha}) \rangle = \sum_k \operatorname*{\mathbf{argmin}}_{\mathbf{s}_k \in A_k} \langle \mathbf{s}_k, [\nabla J(\boldsymbol{\alpha})]_k \rangle \,,$$

where $[\nabla J(\boldsymbol{\alpha})]_k$ are the gradient components corresponding to $\boldsymbol{\alpha}_k$. These search directions $\mathbf{s}_k$ can be found by solving the augmented inference problem

$$\mathbf{y}_k^* = \operatorname*{\mathbf{argmax}}_{\mathbf{y} \in \mathcal{Y}} \left[F(x_k, \mathbf{y}) + \Delta_k(\mathbf{y}) - F(x_k, \mathbf{y}_k) \right] \,,$$

and setting $\mathbf{s}_k(\mathbf{y}^*) = C$ and $\mathbf{s}_k(\mathbf{y}) = 0$ for all $\mathbf{y} \neq \mathbf{y}^*$. Note that this represents a vertex of the feasible domain of $\mathcal{A}_k$. The dual variables $\boldsymbol{\alpha}$ are then updated using a convex combination $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \beta(\mathbf{s}^* - \boldsymbol{\alpha})$ of the current solution and the search direction $\mathbf{s}^*$, where the step size $\beta \in [0, 1]$. The algorithm is known to converge for an arbitrary step-size $\beta = 2/(t+2)$, where t is the iteration number. However, the rate of convergence can be improved by selecting β using a simple line search or computing the optimal step-size, which can be given in closed form by

$$\beta = \operatorname*{\mathbf{argmin}}_{\beta} J(\boldsymbol{\alpha} + \beta(\mathbf{s}^* - \boldsymbol{\alpha})) = \frac{\langle \boldsymbol{\alpha} - \mathbf{s}^*, \nabla J(\boldsymbol{\alpha}) \rangle}{(\mathbf{s}^* - \boldsymbol{\alpha})^T \tilde{\mathbf{K}} (\mathbf{s}^* - \boldsymbol{\alpha})} \,, \tag{4.23}$$

and the step-size is clipped so that $\beta \in [0, 1]$. Despite being a relatively efficient computation, when the number of training examples is large it is often best to use an arbitrary update step-size and avoid these computations that are quadratic with the number of active dual variables.

During our implementation we use the block-coordinate method proposed in [Lacoste-Julien et al., 2013]. This method randomly selects a single example $k \in \{1, \ldots, m\}$ and minimises the linearisation of the objective function $J(\boldsymbol{\alpha})$ with respect to the dual variables $\boldsymbol{\alpha}_k$. This approach benefits from much more friendly closed form calculations for the optimal the step-size β and has been shown [Lacoste-Julien et al., 2013] to have similar convergence properties to the full Frank-Wolfe approach. Parallelism aside, we believe that this approach makes a much better use of the inference scheme, since the full Frank-Wolfe runs inference over all m examples before updating. The violators for each example are chosen independently and do not depend on the potential changes that may be made in other examples. It is likely that a small change in one dual variable $\boldsymbol{\alpha}_k$ could result in a very different selection of violators across a number of other examples, which makes it intuitive to focus on updating a single example at a time.

During each round of the optimisation we randomly select an example $k \in \{1, \ldots, m\}$ to update. The first step involves computing the update direction, which is found using the augmented inference scheme

$$\mathbf{y}^* = \operatorname*{\mathbf{argmax}}_{\mathbf{y} \in \mathcal{Y}} \left[F(x_k, \mathbf{y}) + \Delta_k(\mathbf{y}) \right] .$$

The search direction is then set such that $\mathbf{s}_k(\mathbf{y}^*) = C$ and $\mathbf{s}_k(\mathbf{y}) = 0$ for all $\mathbf{y} \neq \mathbf{y}^*$. The size of the update step is given by

$$\beta = \frac{C \left[F(x_k, \mathbf{y}^*) + \Delta_k(\mathbf{y}^*) \right] - \sum_{\mathbf{y}} \alpha_k(\mathbf{y}) \left[F(x_k, \mathbf{y}) + \Delta_k(\mathbf{y}) \right]}{C^2 \tilde{K}(x_k, \mathbf{y}^*; x_k, \mathbf{y}^*) - 2C \sum_{\mathbf{y}} \alpha_k(\mathbf{y}) K(x_k, \mathbf{y}^*; x_k, \mathbf{y}) + \boldsymbol{\alpha}_k^T \tilde{\mathbf{K}}_{k,k} \boldsymbol{\alpha}_k} , \tag{4.24}$$

where the step-size has been clipped so that $\beta \in [0, 1]$. The update to the dual variable of example k is then given by

$$\boldsymbol{\alpha}_k \leftarrow \boldsymbol{\alpha}_k + \beta \left(\mathbf{s}_k - \boldsymbol{\alpha}_k \right) .$$

Edge potentials

Edge potentials measure the contribution that a given edge labelling makes to the score function; a measure of the compatibility of a particular edge labelling for a given input. These are used extensively throughout the optimisation scheme, during both the inference

procedure and the tree weight update scheme. For a given input observation $x \in \mathcal{X}$, edge $e = (v, v')$ and labelling $\mathbf{u} \in \mathcal{Y}_v \times \mathcal{Y}_{v'}$, the edge potential is given by

$$\begin{aligned}\hat{F}_e(x, \mathbf{u}) &= \sum_{k,\mathbf{y}} \alpha_k(\mathbf{y}) \left(K_e(x_k, \mathbf{y}_{k,e}; x, \mathbf{u}) - K_e(x_k, \mathbf{y}_e; x, \mathbf{u})\right) \\ &= \sum_{k,\mathbf{y}} \alpha_k(\mathbf{y}) k(x_k, x) \left(\mathbb{1}[(\mathbf{y}_{k,e} = \mathbf{u}] - \mathbb{1}[\mathbf{y}_e = \mathbf{u}]\right) ,\end{aligned}$$

where K_e denotes the kernel component corresponding to edge e. Computing these edge potentials on demand can be costly, and to avoid unnecessary overhead we maintain their values at all times throughout the optimisation using a simple update scheme. We begin by initialising $\hat{F}_e(x_k, \mathbf{u}) = 0$ for each edge $e \in E$, edge labelling $\mathbf{u}$ and example $k = \{1, \ldots, m\}$. To update the edge scores, suppose the dual variables associated with example $(x_k, \mathbf{y}_k)$ are to be updated according to $\boldsymbol{\alpha}_k \leftarrow (1 - \beta)\boldsymbol{\alpha}_k + \beta \mathbf{s}_k$, where $\mathbf{s}_k$ is everywhere non-zero apart from the location indexed by labelling $\mathbf{y}^*$, which takes on the value of C. The change in edge potentials is simply given by

$$\hat{F}_e(x, \mathbf{u}) \leftarrow (1 - \beta)\hat{F}_e(x, \mathbf{u}) + \beta C k(x_k, x) \left(\mathbb{1}[\mathbf{y}_{k,e} = \mathbf{u}] - \mathbb{1}[{\mathbf{y}^*}_e = \mathbf{u}]\right) .$$

To further reduce the computational burden, we observe that when the output variables are binary, we only have to compute three out of the four possible edge scores. This is due to the constraint imposed on the dual variables that the sum up to the regularisation parameter C. We see that the score function can be expressed as

$$\begin{aligned}\hat{F}_e(x, \mathbf{u}) &= \sum_{k,\mathbf{y}} \alpha_k(\mathbf{y}) k(x_k, x) \left(\mathbb{1}[\mathbf{y}_{k,e} = \mathbf{u}] - \mathbb{1}[\mathbf{y}_e = \mathbf{u}]\right) \\ &= C \sum_k k(x_k, x) \mathbb{1}[\mathbf{y}_{k,e} = \mathbf{u}] - \sum_{k,\mathbf{y}} \alpha_k(\mathbf{y}) k(x_k, x) \mathbb{1}[\mathbf{y}_e = \mathbf{u}] ,\end{aligned}$$

and therefore the sum over all possible edge labellings is $\sum_{\mathbf{u}} \hat{F}_e(x, \mathbf{u}) = 0$.

During the tree update component we use the edge potentials to compute the weighted edge margins γ_e, and plug them into the maximum spanning tree algorithm. By maintaining these edge potentials in memory we avoid having to recompute them for each example and possible edge, making it relatively computationally inexpensive to find the maximally violating tree. However, one issue is the memory requirements for storing edge potentials, which scales linearly with the number of training points and polynomially with the number of nodes in the graph.

Inference algorithm

The inference algorithm is used throughout the training stage to find the direction upon which to update the dual variables, and during the testing stage to make predictions on unseen test points. For the zero-one loss case the algorithm is pretty much the same during both training and testing. The only difference being the augmentation of the score function $F(x_k, \mathbf{y}) \leftarrow F(x_k, \mathbf{y}) + 1$ for incorrect labellings $\mathbf{y} \neq \mathbf{y}_k$. Later we will show how to efficiently include the hamming loss into the inference algorithm using augmented edge potentials but for now we focus on the zero-one loss case. An outline of the inference scheme is presented in Algorithm 6. For a fixed observation $x \in \mathcal{X}$ and predefined length of K-best list, the algorithm takes as input a set of rooted spanning trees $\mathcal{T}$, a set of weightings over these trees λ_T and the set of edge potentials

$$\hat{F}_{\mathcal{E}} = \left\{ \hat{F}_{v,v'}(y_v, y'_v) \right\}_{(v,v') \in \mathcal{E}, y_v \in \mathcal{Y}_v, y_{v'} \in \mathcal{Y}_{v'}} .$$

We have dropped the dependence of the score functions on the input as it is assumed to be fixed and we let $\mathcal{E}$ denote the set of all edges $(i, j) \in G$ with non-zero weight. By working with rooted trees the edges are implicitly oriented. The direction of the edges are denoted by $v \to pa(v)$, where $pa(v)$ denotes the parent[2] of v and the children of v are denoted by $ch(v)$. We denote T_v as the subtree of T rooted at v, and $T_{v' \to v}$ as the subtree consisting of $T_{v'}$ plus the edge $v' \to v$, and node v.

The algorithm implements a variation of the standard dynamic programming algorithm for finding MAP labelling on a tree, the only difference being that we maintain a list of the K best configurations throughout the traversal of the tree and at the end we pool together the K best labellings across trees to find the one with highest score. The dynamic programming works through the tree in reverse order, so that the children of a node are always processed before the parent. It maintains a sorted K best list of labellings of the subtrees T_v and $T_{v' \to v}$ using the following data structures:

- Score matrix P_v, where element $P_v(y, r)$ records the score of the r-th best multi-label of subtree T_v when node v is labelled as y.

- Pointer matrix C_v, where element $C_v(y, r)$ keeps track of the ranks of the child nodes $v' \in ch(v)$ in the message matrix $M_{v' \to v}$ that contribute the score $P_v(y, r)$.

- Message matrix $M_{v \to pa(v)}$, where element $M_{v \to pa(v)}(y', r)$ records the r-th best multilbael of the subtree $T_{v \to pa(v)}$ when the label of $pa(v)$ is y'.

[2] A parent node $pa(v)$ is the node adjacent to v on the path towards the root. The children $ch(v)$ of v are the nodes adjacent to v on the path away from the root.

- Configuration matrix $C_{v\to pa(v)}$, where element $C_{v\to pa(v)}(y', r)$ traces the label and rank (y, r) of child v that achieves $M_{v\to pa(v)}(y', r)$.

When processing each node, we begin by merging the K-best lists of the children of the node, which are stored in $M_{v'\to v}$, to obtain the score matrix P_v and pointer matrix C_v. This can be performed in amortised $\mathcal{O}(K)$ time per child node. Next the K-best list of $T_{v\to pa(v)}$ corresponding to the all possible labels y' of $pa(v)$ are formed. We do this by keeping the label of the head of the edge $v \to pa(v)$ fixed, and picking the best combination of labelling the tail of the edge, then selecting a multi-label of T_v consistent with that label. This results in the matrices $M_{v\to pa(v)}$ and $C_{v\to pa(v)}$, and can be performed in $\mathcal{O}(K)$ time. The iteration over a tree ends when we have reached the root v_{root} and have updated its score $P_{v_{root}}$. The multi-labels $\mathcal{Y}_{T,K}$ can be computed by tracing the pointers stored in C_v and $C_{v\to pa(v)}$. We see that an iteration on a single tree can be performed in $\mathcal{O}(K\ell)$, and therefore by repeating this process on $|\mathcal{T}|$ trees it gives the algorithm a total complexity of $\mathcal{O}(|\mathcal{T}|K\ell)$.

Algorithm 6 Algorithm to obtain K best multi-labels for collection of weighted spanning trees.

Input: Collection $\mathcal{T}$ of rooted spanning trees, active edge set $\mathcal{E}$, edge scores $\hat{F}_{\mathcal{E}}$
for $T \in \mathcal{T}$ **do**
  Initialise P_v, C_v, $M_{v\to pa(v)}$, $C_{v\to pa(v)}, \forall\ v \in G$
  $I =$ nodes indexed in post-order of tree T
  for $j = 1 : \ell$ **do**
    $v = I(j)$
    if $ch(v) \neq \emptyset$ **then**
      $P_v(y) = P_v(y) + \underset{r_v, v'\in ch(v)}{\mathbf{kmax}} \left(\sum_{v'\in ch(v)} \left(M_{v'\to v}(y, r_v)\right)\right)$
      $C_v(y) = P_v(y) + \underset{r_v, v'\in j}{\mathbf{argkmax}} \left(\sum_{v'\in ch(v)} \left(M_{v'\to v}(y, r_v)\right)\right)$
    end if
    $M_{v\to pa(v)}(y_{pa(v)}) = \underset{y,r}{\mathbf{kmax}} \left(P_v(y, r) + \hat{F}_{v,pa(v)}(y, y_{pa(v)})\right)$
    $C_{v\to pa(v)}(y_{pa(v)}) = \underset{y,r}{\mathbf{argkmax}} \left(P_v(y, r) + \hat{F}_{v,pa(v)}(y, y_{pa(v)})\right)$
  end for
  Trace back with C_v and $C_{v\to v'}$ to get $\mathcal{Y}_{T,K}$.
end for
$\mathcal{Y}_{\mathcal{T},K} = \bigcup_{T\in\mathcal{T}} \mathcal{Y}_{T,K}$
$\mathbf{y}^* = \mathbf{argmax}_{\mathbf{y}\in\mathcal{Y}_{\mathcal{T},K}} F(x, \mathbf{y}, \mathbf{w})$
$\bar{\mathbf{y}} = \mathbf{argmax}_{\mathbf{y}\in\mathcal{Y}_{\mathcal{T},K}\setminus\mathbf{y}_k} F(x, \mathbf{y}, \mathbf{w})$
$\theta_K = \sum_{T\in\mathcal{T}} \lambda_T \hat{F}_T(x, \mathbf{y}_{T,K}, \mathbf{w}_T)$
Return : $\mathbf{y}^*, \bar{\mathbf{y}}, \theta_K$

Recall that Lemma 10 states that we can be assured of exact inference if the score function satisfies a particular condition regarding the scores across the K-best list. This is not always

required during the learning phase and therefore during the early stages of the optimisation, we would encourage the use of small values of K[3] to get the optimisation moving in the direction of the optimal solution. We know that progress will be made towards the optimal solution if the duality gap is greater than zero and therefore it seems intuitive to take advantage of these relatively cheap inference steps during the early stages. Only later when we wish to be assured that the algorithm has converged would we suggest ramping up the value of K to ensure exact inference has been performed on all examples. One can also perform relatively naive updates early one by simply taking a *walk* along the graph beginning at the correctly labelling and randomly flipping the labelling so as to obtain a positive duality gap.

Tree weight updates

When making adjustments to the weights of the trees we must be weary of the fact that current solution to the structured SVM, and therefore the current set of maximum violators, depends on a fixed kernel function. By adjusting the tree weights, and the kernel, it is possible that the maximum violators for each example can change, making the previous structured SVM solution redundant. In order to limit the oscillatory nature of our solution path, we propose small adjustments to kernel weightings, which will hopefully lead to a more stable optimisation procedure.

The tree weight updates depend on the gradient of the objective function given in 4.11. To help simply our expressions, we introduce the variables υ_T

$$\begin{aligned}\upsilon_T = 2\left|\frac{\partial J}{\partial \lambda_T}\right| &= \sum_{k,\mathbf{y}}\sum_{j,\mathbf{y}'}\alpha_k(\mathbf{y})\alpha_j(\mathbf{y}')\tilde{K}_T(x_k,\mathbf{y};x_j,\mathbf{y}') \\ &= \sum_{e\in T}\sum_{k,\mathbf{y}}\sum_{j,\mathbf{y}'}\alpha_k(\mathbf{y})\left(\hat{F}_e(x_k,\mathbf{y}_{k,e}) - \hat{F}_e(x_k,\mathbf{y}_e)\right) = \sum_{e\in T}\gamma_e\,,\end{aligned}$$

represent the absolute value of the gradients with respect to each tree. During the tree weight update, Algorithm 7, we use the edge potentials to compute the weighted margins γ_e for each edge in the graph. These weighted edge margins are used to compute υ_T for the current trees $\mathcal{T}$, and to find the maximum violating tree T^* i.e. the tree with largest absolute gradient. The tree weight update can take on one of two forms and depends on whether or not the maximally violating tree T^* belongs to the current active set of trees $\mathcal{T}$. Both updates however begin the same way by examining the gradients of the current set of active trees.

[3]For example in our experiments, we typically tended to start with values of $K < 4$ until latter stages of optimisation.

Algorithm 7 Tree weight update

Input: $\mathcal{T}, T^*, \boldsymbol{\lambda}, \boldsymbol{\alpha}, \epsilon, \nu$
let $\omega = \sum_{T \in \mathcal{T}} \lambda_T \upsilon_T$ - the weighted average gradient
for $T \in \mathcal{T}$ **do**
  if $\upsilon_T < \omega \cap \lambda_T < \nu$ **then**
    $\lambda_T = 0$ and $\mathcal{T} \leftarrow \mathcal{T} \setminus T$
  end if
end for
$\lambda_T \leftarrow \lambda_T / \sum_{T' \in \mathcal{T}} \lambda_{T'}$

if $T^* \in \mathcal{T}$ **then**
  for $T \in \mathcal{T}$ **do**
    $\hat{\lambda}_T = \frac{\lambda_T \exp(\sqrt{\upsilon_T})}{\sum_{T' \in \mathcal{T}} \lambda_{T'} \exp(\sqrt{\mathrm{d}J_{T'}})}$
    $\lambda_T \leftarrow \lambda_T + \beta(\hat{\lambda}_T - \lambda_T)$
  end for
else
  $\mathcal{T} \leftarrow \mathcal{T} \cup T^*$
  $\lambda_{T^*} = \frac{1}{|T|}$
  for $T \in \mathcal{T} \setminus T^*$ **do**
    $\lambda_T \leftarrow \frac{|\mathcal{T}|-1}{|\mathcal{T}|} \lambda_T$
  end for
end if

A standard multiplicative update scheme would prevent weights from ever reaching zero and excluding trees from the active set. Therefore in order to promote a sparse set of trees, we introduce a weight threshold ν that combined with the trees gradient and the weighted average gradient of the trees allows us to truncate the tree weight to zero and effectively remove it from the active set. If $\lambda_T < \nu$ and $\upsilon_T < \omega = \sum_{T \in \mathcal{T}} \lambda_T \upsilon_T$, then we set $\lambda_T = 0$ and remove T from the active set i.e. $\mathcal{T} \leftarrow \mathcal{T} \setminus T$.

If the maximal gradient tree T^* belongs to the active set $\mathcal{T}$, then we perform a multiplicative update on these weights proportional to the magnitude of their gradient. Note that rather than implementing the full multiplicative update as suggested in [Kloft et al., 2011], we alluded earlier to the fact that we dampen the change of tree weights using some $\beta \in (0, 1)$ in an effort to stabilise the active set of trees and the set of maximally violating labels for each example. If the tree does not belong to the active set i.e. $T^* \notin \mathcal{T}$ then we include it to the active set $\mathcal{T} \leftarrow \mathcal{T} \cup T^*$, initialise its value to $\lambda_{T^*} = \frac{1}{|\mathcal{T}|}$ and redistribute the weight of the other trees so that $\sum_{T \in \mathcal{T} \setminus T^*} \lambda_T = 1 - 1/|\mathcal{T}|$. Again we abstain from making large steps in the tree weights to encourage a smooth optimisation procedure.

Marginal dual polytope representation for structured SVM

So far when solving the structured SVM we have focused on using the dual variables associated with the margin constraints. One drawback of this approach is that we are unsure of the total number of violators that we may require. This may result in a drain on memory and unnecessary overheads involving the addition and removal of dual variables from our solution. It was shown in [Tsochantaridis et al., 2004] that the number of constraints, and therefore dual variables of the optimal solution, required does not depend on the size of the label space $|\mathcal{Y}|$ but is instead polynomial with respect to both the number of training examples m, and the inverse of the desired accuracy $\epsilon > 0$. However given the nature of our optimisation we would expect to go through a large number of candidate violators before converging upon the optimal solution that has a polynomial number. In this section we review the marginal dual polytope reparameterisation, which produces a polynomial-size problem and permits the use of kernels to efficiently address complex input spaces.

In [Taskar et al., 2004] the authors observed that the dual variables $\boldsymbol{\alpha}_k$ for each example $k \in \{1, \ldots .m\}$ could be interpreted as a density function over $\mathcal{Y}$ conditioned on x_k, where the weight attached to a dual variable is proportional to our belief that we could confuse this labelling to be the correct one i.e. $P(\mathbf{y}_k = \mathbf{y}) \propto \alpha_k(\mathbf{y})$. In their formulation, they went on to show that the objective could be separated into terms involving only nodes and edges, and defined the marginal dual variables to be

$$\mu(k, e, \mathbf{u}_e) = \sum_{\mathbf{y}} \alpha_k(\mathbf{y}) \mathbb{1}[\mathbf{y}_e = \mathbf{u}_e] \text{ (edge marginal)}$$
$$\mu(k, i, u_i) = \sum_{\mathbf{y}} \alpha_k(\mathbf{y}) \mathbb{1}[y_i = u_i] \text{ (node marginal) .}$$

Note that our kernel function is defined in terms of the edge labellings and we no longer require the use of the node marginals. We now show how the objective function given in 4.22 can be reformulated in terms of the edge marginals. Beginning with the quadratic expression involving the kernel, this is given by

$$\begin{aligned}
&\sum_{k,\mathbf{y}} \sum_{j,\mathbf{y}'} \alpha_k(\mathbf{y}) \alpha_j(\mathbf{y}') \sum_{T \in \mathcal{T}} \lambda_T \tilde{K}_T(x_k, \mathbf{y}; x_j, \mathbf{y}') \\
&= \sum_{k,\mathbf{y}} \sum_{j,\mathbf{y}'} \alpha_k(\mathbf{y}) \alpha_j(\mathbf{y}') \sum_{e \in \mathcal{E}} \lambda_e \tilde{K}_e(x_k, \mathbf{y}_e; x_j, \mathbf{y}'_e) \\
&= \sum_{k,j} \sum_{e \in \mathcal{E}} \lambda_e \sum_{\mathbf{u}_e, \mathbf{u}'_e} \sum_{\mathbf{y}_e = \mathbf{u}_e} \sum_{\mathbf{y}'_e = \mathbf{u}'_e} \alpha_k(\mathbf{y}) \alpha_j(\mathbf{y}') \tilde{K}_e(x_k, \mathbf{u}_e; x_j, \mathbf{u}'_e) \\
&= \sum_{k,j} \sum_{e \in \mathcal{E}} \lambda_e \sum_{\mathbf{u}_e, \mathbf{u}'_e} \mu(k, e, \mathbf{u}_e) \mu(j, e, \mathbf{u}'_e) \tilde{K}(x_k, \mathbf{u}_e; x_j, \mathbf{u}'_e)
\end{aligned}$$

When using the zero-one loss function, we can monitor the contribution of each example $k = \{1, \ldots, m\}$ to the linear term of the objective function by recording the weight $l_k \in [0, C]$ that has been moved away from the correct labelling. To do this we begin by initialising $l_k = 0$ for each $k \in \{1, \ldots, m\}$. Then suppose we update in the direction $\mathbf{s}_k$, where $\mathbf{s}_k(\mathbf{y}^*) = C$ and $\mathbf{s}_k(\mathbf{y}) = 0$ for all $\mathbf{y} \neq \mathbf{y}^*$, and use the step-size $\beta \in [0, 1]$. Then the adjustment of l_k is given by

$$l_k \leftarrow \begin{cases} l_k + \beta C & \text{if } \mathbf{y}^* \neq \mathbf{y}_k \\ l_k - \beta C & \text{if } \mathbf{y}^* = \mathbf{y}_k \end{cases}$$

The marginal dual reparameterisation implements the same optimisation scheme as the dual variables, meaning that the calculations for computing edge potentials and finding update directions are essentially the same but just have to be computed in a different manner. For example, the edge potentials are calculated according to

$$\begin{aligned} \hat{F}_e(x, \mathbf{u}) &= \sum_k \sum_{\mathbf{y}} \alpha_k(\mathbf{y}) k(x_k, x) \left(\mathbb{1}[\mathbf{y}_{k,e} = \mathbf{u}] - \mathbb{1}[\mathbf{y}_e = \mathbf{u}] \right) \\ &= C \sum_k k(x_k, x) \mathbb{1}[\mathbf{y}_{k,e} = \mathbf{u}] - \sum_k k(x_k, x) \mu(k, e, \mathbf{u}) \,. \end{aligned}$$

To find the optimal update direction $\mathbf{y}^*$ we can simply plug in the edge potentials, computed using $\boldsymbol{\mu}$, into the inference algorithm. To replicate the update $\boldsymbol{\alpha}_k \leftarrow (1 - \beta)\boldsymbol{\alpha}_k + \beta \mathbf{s}_k$, where the non-zero entry in $\mathbf{s}_k$ is indexed by labelling $\mathbf{y}^*$, the marginal dual variables are updated according to

$$\mu(k, e, \mathbf{u}) = (1 - \beta)\mu(k, e, \mathbf{u}) + \beta \mathbb{1}[\mathbf{y}^*_e = \mathbf{u}] \,,$$

where the optimal step-size can be computed in an analogous way to 4.24. When using the marginal dual polytope representation for our problem, the total number of variables is given by $4m\binom{\ell}{2}$. To put this into perspective, when working with the dual form, we need to keep a record of the labellings associated with each violating example. From [Tsochantaridis et al., 2004] we saw that the solution could be approximately solved using a polynomial number of constraints, which upper bounds the memory requirements for the dual form. However, given the frequent use of edge labellings, it makes sense to store the violating example as its edge labellings rather than its output labellings. Therefore if we have an average of more than four violators for each example then the memory requirements for the dual form exceed those of the marginal dual polytope form.

Using Hamming loss function

So far we have presented a learning scheme that attempts to obtain a margin of magnitude at least one across the set of all possible incorrect labellings. This is a difficult task since there are $2^\ell - 1$ incorrect labellings for each example, and ℓ violators that differ from the correct labelling on only one node. If we are convinced that there should exist a large margin solution and are focused on obtaining the exact labelling for each example then one could argue that the zero-one loss function is the right approach to take. However with multi-label problems, we are often willing to sacrifice some incorrectly labelled vertices if it permits a much simpler solution. In doing so, we hope that the performance seen during training will generalise to unseen examples.

Recall that the Hamming loss function measures the average disagreement of node labellings, therefore the loss function for example k is given by

$$\Delta_k(\mathbf{y}) = \frac{1}{\ell} \sum_{i \in V} \mathbb{1}[y_{k,i} \neq y_i].$$

During the structural learning component of our optimisation, in order to find the update direction for example k, we must solve the augmented inference problem given by

$$\mathbf{y}^* = \underset{\mathbf{y} \in \mathcal{Y}}{\mathbf{argmax}} \; [F(x_k, \mathbf{y}) + \Delta_k(\mathbf{y})] \tag{4.25}$$

where $\Delta_k(\mathbf{y})$ is the Hamming loss function. In the previous inference task, for each tree we computed the top K scores and incremented their score by one if they were not the correct labelling. Unfortunately this approach cannot be taken when using the Hamming loss since the individual edge labellings contribute to the overall loss and must be taken into consideration when we traverse the tree. Note that this is not a special case because we are using a combination of spanning trees but it is also the case even when we only have a single tree upon which to perform augmented inference. We now outline how to adjust the edge potentials to include the contribution of the Hamming loss.

Suppose we have a tree structured graph $T \in S(G)$ and let $n(i)$ denote the node degree[4] of node $i \in V$ on graph T. By augmenting the edge potential of edge e and labelling $\mathbf{u} = (u, u')$ by

$$\tilde{F}_{i,j}\left(x_k, \mathbf{u}\right) = \hat{F}_{i,j}(x_k, \mathbf{u}) + \frac{1}{\ell}\left(\frac{\mathbb{1}[y_{k,i} \neq u]}{n(i)} + \frac{\mathbb{1}[y_{k,j} \neq u']}{n(j)}\right)$$

we are able to use our existing top-K algorithm and efficiently incorporate the Hamming

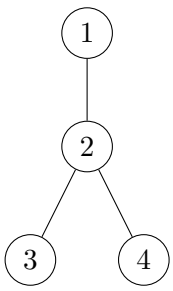
[4]The node degree of a node in a graph G is the number of nodes that are connected to node i

loss function into the inference scheme.

During the K-best inference algorithm (Algorithm 6), we adjust the message passing matrices to use $\tilde{F}_{v,v'}$ rather than $\hat{F}_{v,v'}$ i.e.

$$M_{v\to pa(v)}(y_{pa(v)}) = \underset{y,r}{\mathbf{kmax}} \left(P_v(y,r) + \tilde{F}_{v,pa(v)}(y_v, y_{pa(v)})\right)$$

By doing this we ensure that we take into consideration the full contribution to the Hamming loss of the involved nodes when computing the intermediate K-best lists, before moving up the tree. This small adjustment allows us to incorporate the Hamming loss into our existing framework for performing inference, thus allowing us to efficiently solve the augmented inference problem given in 4.25. Note that the augmented edge potentials are different across trees, and we have to perform this augmentation on a tree by tree basis during inference. Once the K-best lists have been found on each tree and we have $\mathcal{Y}_{\mathcal{T},K}$, we simply evaluate the scores $F(\mathbf{y})$ for each $\mathbf{y} \in \mathcal{Y}_{\mathcal{T},K}$ and then augment it with their respective Hamming loss afterwards.



$$\begin{aligned}
\tilde{F}_{3,2}(y_3, y_2) &= \hat{F}_{3,2}(y_3, y_2) + \frac{\mathbb{1}[y_3 = 1]}{1} + \frac{\mathbb{1}[y_2 = 1]}{3} \\
\tilde{F}_{4,2}(y_4, y_2) &= \hat{F}_{4,2}(y_4, y_2) + \frac{\mathbb{1}[y_4 = 1]}{1} + \frac{\mathbb{1}[y_2 = 1]}{3} \\
\tilde{F}_{2,1}(y_2, y_1) &= \hat{F}_{2,1}(y_2, y_1) + \frac{\mathbb{1}[y_2 = 1]}{3} + \frac{\mathbb{1}[y_1 = 1]}{1} \\
\tilde{F}_T(0,1,1,0) &= \hat{F}_{2,1}(1,0) + \hat{F}_{3,2}(1,1) + \hat{F}_{4,2}(0,1) + 3\frac{1}{n(2)} + 1\frac{1}{n(3)} \\
&= \hat{F}_T(0,1,1,0) + \Delta_{\mathbf{y}}(0,1,1,0)
\end{aligned}$$

Fig. 4.5 Example tree structure showing how the augmentation of the edge potentials works, where the correct labelling is given by $\mathbf{y} = (0,0,0,0)$.

4.4.7 Experiments

In this section we evaluate our L_1 norm spanning tree algorithm (TA) on a number of datasets and compare its performance to a number of state-of-the-art learning algorithms. We use ten multi-label datasets, taken from a wide range of domains including biological, chemical and text classification. A summary of the datasets is provided in Table 4.1. The NCI60 dataset is a panel of 60 diverse human cancer cell lines that have been used for screening compounds relating to anticancer activity. The Fingerprint dataset (FP) uses as input molecular mass spectra data to predict molecular substructures. The ENRON dataset contains emails from Enron employees that are categorised according to the content of the email. The MEDICAL dataset maps clinical free text to a number of medical based codings. The SCENE dataset uses

image data to determine whether certain classes are present within the image. The CAL500 dataset was collected by asking humans to listen to 500 different songs and categorise them using a survey designed capture the semantic associations between music and words. The YEAST dataset uses information about the yeast cells to determine the localisation site of each cell. The EMOTIONS dataset uses a number of audio based features to determine which emotions where being expressed in a particular song. The CIRCLE10 and CIRCLE50 datasets have been synthetically generated according to the process outlined in [Bian et al., 2012].

DATASET	EXAMPLES	LABELS	FEATURES	DENSITY
EMOTIONS	593	6	72	0.31
YEAST	2417	14	103	0.30
SCENE	2407	6	294	0.18
ENRON	1702	53	1001	0.06
CAL500	502	174	68	0.15
FP	490	286	490	0.17
CANCER	4547	60	4547	0.18
MEDICAL	978	45	1449	0.03
CIRCLE10	1000	10	3	0.85
CIRCLE50	1000	50	3	0.71

Table 4.1 Summary statistics of the datasets used during the experiments.

For comparison, we select the following learning models to compare to our newly proposed $\text{TA}_{0/1}$ and TA_{HAM} models, which have been trained using the zero-one loss function and the Hamming loss function, respectively. The SVM is used as a single target classifier, where each node is optimised and predicted independently. Multitask feature learning (MTL) [Argyriou et al., 2008] is a multi-label classifier that assumes that the label specific functions are related such that they share a small subset of features. Max-margin conditional random fields (MMCRF) [Rousu et al., 2007] is a multi-label classifier that uses the structure of the output graph that connects multiple labels and uses the loopy belief propagation algorithm for approximate inference on the general graph. Maximum average marginal aggregation (MAM) [Su and Rousu, 2013] is a multi-label ensemble model that trains a set of random tree based learners separately and performs the final approximate inference on a union graph of the edge potential functions of the trees. Finally, we evaluate the L_2-random spanning tree (RTA) [Marchand et al., 2014] algorithm, which uses a fixed random sample of spanning trees and learns a large margin predictor. We train this algorithm using both the zero-one loss $\text{RTA}_{0/1}$ and Hamming loss functions, RTA_{HAM}, just as we do for the TA methods.

We use 5-fold cross validation to select the model parameters and compute the results. The margin slack parameter C is evaluated over the set $\{0.01, 0.1, 1, 10, 100, 1000\}$ and the number of initial trees $\mathcal{T}$ is tested over $\{5, 10, 20, 40\}$. Each method is presented with the same

Dataset	0/1 Loss (%)							
	SVM	MTL	MMCRF	MAM	$\text{RTA}_{0/1}$	RTA_{HAM}	$\text{TA}_{0/1}$	TA_{HAM}
Emotions	77.8	74.5	71.3	69.6	69.5	**66.3**	68.4	*67.2*
Yeast	85.9	88.7	93.0	86.0	83.6	*77.7*	81.9	**77.3**
Scene	47.2	55.2	72.2	94.6	*29.6*	30.2	**27.5**	29.7
Enron	99.6	99.6	92.7	87.9	88.2	*87.7*	89.4	**87.1**
Cal500	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
FP	99.0	100.0	99.6	99.6	**96.7**	98.0	97.6	*97.5*
NCI60	56.9	53.0	63.1	60.0	*50.7*	52.9	**50.3**	51.6
Medical	91.8	91.8	63.8	63.1	*29.1*	36.7	**28.3**	31.8
Circle10	28.9	33.2	20.3	17.7	3.7	4.0	*3.5*	**3.3**
Circle50	69.8	72.3	**38.8**	46.2	68.4	52.8	48.5	*46.1*

Table 4.2 Prediction performance of each algorithm in terms of 0/1 loss. The best performing algorithm is highlighted with **boldface**, the second best is in *italic.*

Dataset	Microlabel Loss (%)							
	SVM	MTL	MMCRF	MAM	$\text{RTA}_{0/1}$	RTA_{HAM}	$\text{TA}_{0/1}$	TA_{HAM}
Emotions	22.4	20.2	20.1	*19.5*	21.9	**18.8**	21.4	19.7
Yeast	20.0	20.7	21.7	20.1	23.7	*19.8*	22.8	**19.3**
Scene	9.8	11.6	18.4	17.0	8.9	**8.8**	8.3	*8.5*
Enron	6.4	6.5	6.2	*5.0*	5.7	5.3	5.7	**4.7**
Cal500	*13.7*	13.8	*13.7*	*13.7*	23.0	13.8	15.8	**13.3**
FP	*10.3*	17.3	10.5	10.5	*10.3*	10.7	*10.3*	**9.8**
NCI60	15.3	16.0	14.6	*14.3*	16.3	14.9	16.3	**13.3**
Medical	2.6	2.6	2.1	2.1	*1.0*	2.1	**0.9**	1.1
Circle10	4.7	6.3	2.6	2.5	*0.6*	*0.6*	*0.6*	**0.5**
Circle50	5.7	6.2	**1.5**	*2.1*	6.1	3.8	4.2	*2.1*

Table 4.3 Prediction performance of each algorithm in terms of microlabel loss. The best performing algorithm is highlighted with **boldface**, the second best is in *italic.*

initial set of trees $\mathcal{T}$ where RTA maintains an equal weighting for each tree and TA, adjusts weights whilst adding and removing trees according to a criteria defined by the violation of the KKT conditions for optimality. For each dataset we use a linear kernel for the input space and the size of our K-best list is restricted by the number of labels in that particular dataset. We report both the multi-label and micro-label losses, where multi-label loss indicates whether or not the correct labelling was predicted i.e. $\mathbf{y}_{\mathbf{w}}(x_k) = \mathbf{y}_k$, and the micro-label accuracy measures fraction of incorrectly labelled output nodes i.e. the Hamming loss.

In Tables 4.2 and 4.3 we show the zero-one and micro-label loss, respectively, obtained by the different learning algorithms on our datasets. We observe that when training using the zero-one loss, the performance of the spanning tree approaches are in general inferior to

those trained using the Hamming loss. This would indicate that the algorithms struggle to find a fixed large margin that holds true for all possible violators. This is not very surprising given the exponential number of violators that the condition has to be satisfied for. The exceptions to this are the SCENE and MEDICAL datasets, which would suggest they are more amenable to large margin analysis. For example, in the SCENE dataset it is very likely that an image will contain one of a fixed set of category combinations, and it may be more of a multiclass task rather than multi-label. It would appear that minimising the Hamming loss allows the algorithms to focus on directing the predictor towards a solution in the *right region*, where it can trade off small Hamming losses with a smoother solution, and in general better performance in terms of both zero-one and Hamming loss. We are a little disheartened by the poor performance of the zero-loss function formulation of the TA algorithm on several of the datasets, however we saw in the last chapter that the SVM struggled on difficult tasks where there was no margin, and it would appear this multi-label extension also struggles on datasets where there is no clear separation between the true labelling and all possible violators.

These early experiments go some way to supporting the use of this approach to multi-label classification over unknown graph structures, however there is still much work that can be done to improve both the performance across these datasets and the general functionality of the algorithm. We touch briefly on the latter towards the end of this chapter but for now we just want to point out that the limited performance improvements that we see may be partially due to the datasets that we are using, and that they may not be particularly conducive to the methods we are proposing. A similar result was seen in the early days of multiple kernel learning, where the performance improvements were limited until it was used on the *correct* datasets. We hope that further research will allow these datasets to be found, and the algorithm can take advantage of its ability to find the margin maximising graph structure.

Stock market example

We apply our learning framework to the problem of predicting the joint movement of a set of 13 large cap stocks from the S&P 500, which represent a range of different market sectors. Our goal is to simultaneously uncover and exploit the relationships between the movements of stocks, and identify the most likely joint price movement. For example if we are confident that the price of IBM will go up and there is a strong correlation between IBM and Microsoft, then it should influence the likelihood that the price of Microsoft will also increase. The weights attributed to edges can be thought of as an alternative measure of stock correlation. Traditional approaches to making portfolio decisions use the covariance matrix between assets to form diversified portfolios. Our approach effectively encodes dependencies between stocks, and it can be interpreted as some form of conditional covariance function.

Here the strengths of the relationships between stocks is dependent on some input variable that captures specific characteristics of the current market setting.

We use daily price data ranging from January 2012 to December 2014. Let $p_{t,i}$ be the opening price of the i-th stock at time t, and let $\delta_{t,i} = p_{t,i} - p_{t-1,i}$ be the change in price from time $t-1$ to t. The output vector at time t is given by $\mathbf{y}_t \in \{0,1\}^{13}$, where each co-ordinate $y_{t,i}$ is an indicator function for stock i, indicating whether the price of stock i increases over the coming time period i.e. $y_{t,i} = \mathbb{I}[\delta_{t+1,i} \geq 0]$. We use a simple representation for the input space, which corresponds to whether the current opening price is higher or lower than yesterday's opening i.e. $x_{t,i} = \text{sign}(\delta_{t,i})$. These price changes are combined to give the input observation x_t, and we make use of a linear kernel to represent the similarities between input observations.

The time series nature of the data means that it is not amenable to use of cross-validation approaches for optimising model parameters. Instead we propose a very simple method, which borrows inspiration from the field known as *learning from expert advice* [Cesa-Bianchi and Lugosi, 2006]. We let each parameterisation (C, T) correspond to a particular expert indexed by j. The exponential weights algorithm commonly used throughout the field of learning with expert advice is given by Algorithm 8. The algorithm begins by initialising each expert j with a weight v_j proportional to its performance on the training sample. At each time step t, the predictions $\hat{\mathbf{y}}_{t,j}$ of each expert j are weighted according to v_j to form an overall prediction for time t given by $\hat{\mathbf{y}}_{t,i} = \mathbb{1}[\sum_{j=1}^{P} v_j \hat{y}_{t,i,j} \geq 0]$, where $\hat{y}_{t,i,j}$ corresponds to the j-th expert's prediction of stock i at time t. Based upon the performance of the prediction, the weight of each expert is updated and the process repeated until the end. In the structured prediction task, we measure the performance of an algorithm using the Hamming loss. The idea behind this very vanilla approach is to ensure that we do almost as well as the best predictor in hindsight, however more complicated variants could be investigated if we were to use the algorithms proposed in [Herbster and Warmuth, 1998] for tracking the best expert i.e. in this setting the best parameterisation is allowed to change over time. However, the goal here is not to evaluate methods for time series prediction but rather assess whether or not this approach to predicting the movement of a portfolio has any potential.

To compare the performance of our algorithm, for each stock we train an SVM classifier using a variety of parameterisations and combine them together using the expert weighting framework to form a prediction on the test sample. These individual stock predictions are then combined to give the overall prediction at each time step. Note that each stock has been trained individually using an SVM and individually handled using the expert learning framework. We present both the results of training the SVM using the combined input space

Algorithm 8 Expert learning over model parameterisations

Input: learning rate $\alpha > 0$, different parameterisations (C, T) indexed by $j = 1, \ldots, P$, hamming losses $L_j = \sum_{k=1}^{m} L_{j,k}$ on the training sample.
Initialise weights $v_j = \exp(-\alpha L_j) / \sum_k \exp(-\alpha L_k)$
for $t = 1, \ldots T$ **do**
  $\hat{y}_{t,i} = \mathbb{1}[\sum_{j=1}^{P} v_j \ \hat{y}_{t,i,j} \geq 0]$
  $v_j \leftarrow v_j \exp(-\alpha L_{j,k+t})$ for each $j = 1, \ldots, P$
  $v_j \leftarrow v_j / \sum_{p=1}^{P} v_p$ for each $j = 1, \ldots, P$
end for

$\mathbf{x}_t$ (SVM$_J$), and training each stock using only its own observations $x_{t,i}$ (SVM$_I$), once again we use the linear kernel to measure the similarities between observations.

In Table 4.4 we present the results obtained from the stock market experiments. Along with the microlabel and 0/1 loss, we present the hypothetical return that would have been obtained if we would have followed the investment decision implied from the expert learning algorithms prediction. The return on an investment at time t is given by $r_t = 100 \sum_{i=1}^{\ell} \hat{y}_{t,i} \delta_{t+1} / p_t$ There is one free parameter in this version of the expert learning algorithm, the learning rate α, and we present results for a range of $\alpha \in \{0.01, 0.05, 0.1, 0.2, 0.5, 1\}$. From the results we see that our proposed L_1-norm tree approximation method is the top performing method across the three different measures on all but one choice of parameterisation of the learning algorithm. We see that the random spanning tree structure fairs unfavourably to both the TA method and the independent SVM methods. One possible explanation for its poor performance is due to the arbitrary dependencies that we force between pairs of stocks, which hinders rather than helps the performance of the predictor. The SVM methods treat each stock independently and will not suffer from this setback, and the dependency structure is learned in the TA method to maximise the margin that is possible on the training sample.

This preliminary experiment would support the idea that the structure implied by a large margin predictor can be leveraged to make predictions capable of outperforming machine learning methods that use either random dependencies, RTA, or assume independence SVM, between the stocks. In future work one can expect to see further performance improvements if we experiment with a variety of input space representations. For example, rather than sharing a single input kernel over the entire graph, we could breakdown the input kernel over the edges themselves. This would come with additional computational and memory requirements, since during training we would have to replace a separate kernel of size $m \times m$ with a separate one over each of the $|E|$ edges.

Alpha	Microlabel Loss (%)				0/1 Loss (%)				Return (%)			
	RTA	TA	SVM$_J$	SVM$_I$	RTA	TA	SVM$_J$	SVM$_I$	RTA	TA	SVM$_J$	SVM$_I$
0.01	*98.6*	**98.0**	100.0	99.7	50.0	**48.3**	*48.6*	49.6	5.1	**138.1**	*109.6*	-29.2
0.05	*98.6*	**98.0**	100.0	99.7	49.9	**48.4**	49.1	*49.0*	4.0	**139.1**	*91.7*	16.4
0.10	*98.9*	**98.0**	100.0	99.7	50.3	**48.1**	*48.6*	49.0	-14.7	**142.6**	*88.7*	-15.2
0.20	*98.6*	**98.0**	100.0	100.0	50.0	**48.4**	*48.5*	49.1	-6.3	**127.3**	*115.7*	-12.2
0.50	**98.6**	98.9	100.0	100.0	49.4	**48.0**	48.8	*48.6*	20.8	**151.3**	*70.5*	43.4
1.00	**98.6**	99.2	100.0	100.0	49.2	**48.4**	48.9	*48.5*	43.6	**131.3**	*58.8*	54.3

Table 4.4 Comparison of algorithm performance on stock market dataset in terms of microlabel loss, 0/1 loss and percentage return of an investment strategy following this advice. The best performing algorithm for each evaluation criteria is highlighted with **boldface**, the second best is in *italic.*

4.4.8 Extensions

In this section we examine a number of extensions that can be made to the original formulation of our learning algorithm. One of these extensions is similar to the use of the marginal dual polytope and focuses on reparameterising the weights on an exponentially large number of trees, to weights over a polynomial number of edges. We also present a heuristic for guiding the inference scheme on an example basis, and adapt the edge potentials to efficiently perform augmented inference using the Hamming loss function and include unary (label specific) potentials.

Augmenting edge potentials and inference

In our discussions thus far we have focused on constructing functions using edge potentials, however in the original CRFs [Lafferty et al., 2001], there were explicit terms for the contribution of individual nodes to the overall compatibility of a labelling i.e. the score function would be represented by

$$F(x, \mathbf{y}) = \sum_{i \in V} F_i(x, y_i) + \sum_{(i,j) \in G} F_{i,j}(x, y_i, y_j)\,.$$

These node potentials are given by

$$F_i(x, u) = \sum_{k, \mathbf{y}} \alpha_k(\mathbf{y}) k(x_k, x) \left(\mathbb{1}[y_{k,i} = u] - \mathbb{1}[y_i = u] \right)$$

To incorporate the unary potentials into the inference scheme, we would simply follow the same process as we did to incorporate an incorrect node labelling during the Hamming loss variant of the optimisation. Namely for a given tree T where $n(i)$ defines the node degree of node i on T, the augmented edge potentials $\tilde{F}_{i,j}$ for example k and labelling $\mathbf{u} = (u_i, u_j)$

are given by

$$\tilde{F}_{i,j}(x_k, \mathbf{u}) = \hat{F}(x_k, \mathbf{u}) + \left(\frac{F_i(x_k, u_i)}{n(i)} + \frac{F_j(x_k, u_j)}{n(j)} \right)$$

Intermediate tree gradients

The alternating optimisation scheme means that there is quite a bit of back and forth between updating dual variables and updating tree weights. It is reasonable to expect that the progress made during violator updates may be eroded by the update of tree weights, and vice versa. In our current setting, we run the FW scheme for a fixed number of steps and then perform a tree weight update step. Upon closer inspection one can observe that the changes to the tree gradients are computed as a by-product of the FW update scheme, and we therefore have access to this gradients throughout the FW scheme. To see this observe that a particular the $T \in \mathcal{T}$, the gradient is given by

$$\begin{aligned} \mathrm{d}J_T &= \frac{1}{2} \sum_{k,\mathbf{y}} \sum_{j,\mathbf{y}'} \alpha_k(\mathbf{y}) \alpha_j(\mathbf{y}') \tilde{K}_T(x_k, \mathbf{y}; x_j, \mathbf{y}') \\ &= \frac{1}{2} \sum_{e} \sum_{k,\mathbf{y}} \sum_{j,\mathbf{y}'} \alpha_k(\mathbf{y}) \alpha_j(\mathbf{y}') \tilde{K}_e(x_k, \mathbf{y}; x_j, \mathbf{y}') \; = \sum_e \mathrm{d}J_e \,, \end{aligned}$$

where $\mathrm{d}J_e$ are the gradients with respect to a particular edge $e \in E$. For a given edge $e \in E$, the change in its gradient caused by the update of $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \beta(\mathbf{s} - \boldsymbol{\alpha})$ is given by

$$\mathrm{d}J_e \leftarrow \mathrm{d}J_e + \frac{1}{2}\beta^2 (\mathbf{s} - \boldsymbol{\alpha})^T \tilde{K}_e (\mathbf{s} - \boldsymbol{\alpha}) + \beta (\mathbf{s} - \boldsymbol{\alpha})^T \tilde{K}_e \boldsymbol{\alpha} \,.$$

We observe that these terms are required to compute the step size β in 4.23. Therefore one could simply track the value of the edge gradients, adjusting them when we update the dual variables, and then make small adjustments to the weights on each tree according to the MKL scheme. Although this doesn't directly consider the whole space of possible kernels like the full MKL step, we only use the currently active trees, one would imagine that it would make some progress towards shifting the weight in the direction of the optimal solution.

Trees for inference

We have shown that a unit L_1-norm combination of spanning trees produces a large margin predictor. The combination of spanning trees gives rise to a score function that can be represented using the active edges in the graph

$$F(x, \mathbf{y}) = \sum_T \lambda_T \hat{F}_T(x, \mathbf{y}) = \sum_e \lambda_e \hat{F}_e(x, \mathbf{y}_e) = \sum_{(i,j) \in \mathcal{E}} F_e(x, \mathbf{y}_e),$$

where $F_e(x, \mathbf{u}) = \lambda_e \hat{F}_e(x, \mathbf{u})$ are the scaled edges scores. We observe that edges not present in the spanning trees have zero weight and have no contribution to the score function, therefore $F_{i,j} = 0$ for all $(i, j) \notin \mathcal{E}$.

When it comes to inference we are looking to find the largest scoring labelling, and we have done this by performing inference over the trees that were used to construct the predictor. We will show that we are not restricted to performing inference over a fixed set of spanning trees and we can provide conditions under which we are sure of exact inference when using an alternative set of spanning trees.

To see this, let $\mathcal{T}$ denote the set of spanning trees used by the predictor and the let the set of active edges be denoted by $\mathcal{E}$. We can split the active set of edges into two disjoint subsets $\mathcal{E}_1$ and $\mathcal{E}_0$ where $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_0$ and $\mathcal{E}_1 \cap \mathcal{E}_0 = \emptyset$. The score function can be represented by

$$F = \sum_{e \in \mathcal{E}} F_e = \sum_{e \in \mathcal{E}_1} F_e + \sum_{e \in \mathcal{E}_0} F_e.$$

We will refer to $\mathcal{E}_0$ as the residual edge set, and assume that $\mathcal{E}_1$ is an edge set representing the union of a set spanning trees $\mathcal{T}_1$. The contribution of the residual edge set to the score function is upper bounded by

$$\nu = \sum_{e \in \mathcal{E}_0} \max_u F_e(u)$$

Let $\mathcal{T}_1$ be the set of spanning trees that define edge set $\mathcal{E}_1$, we will show that by using the K-best inference scheme we can present conditions showing that exact inference has been performed.

Lemma 11 *Let* $\mathbf{y}^* = \mathbf{argmax}_{\mathbf{y} \in \mathcal{Y}_{\mathcal{T}_1, K}} F(\mathbf{y})$ *be the top scoring multi-label from the set* $\mathcal{Y}_{\mathcal{T}_1, K}$ *and let* $\nu = \sum_{e \in \mathcal{E}_0} \max_u F_e(\mathbf{u})$. *If*

$$F(\mathbf{y}^*) \geq \sum_{T \in \mathcal{T}_1} F_T(\mathbf{y}_{T,K}) + \nu \,,$$

then $\mathbf{y}^* = \mathbf{argmax}_{\mathbf{y} \in \mathcal{Y}} F(\mathbf{y})$.

Proof *Suppose there exists a multi-label* $\mathbf{y} \notin \mathcal{Y}_{\mathcal{T}_1, K}$ *where* $F(\mathbf{y}) > F(\mathbf{y}^*)$ *then*

$$\begin{aligned} F(\mathbf{y}) &= \sum_{e \in \mathcal{E}} F_e(\mathbf{y}) = \sum_{T \in \mathcal{T}_1} F_T(\mathbf{y}) + \sum_{e \in \mathcal{E}_0} F_e(\mathbf{y}) \leq \sum_{T \in \mathcal{T}_1} F_e(\mathbf{y}) + \nu \\ &\leq \sum_{T \in \mathcal{T}_1} F_T(\mathbf{y}_{T,K}) + \nu. \end{aligned}$$

Therefore we see that $F(\mathbf{y}) \leq F(\mathbf{y}^*)$ *and that* $\mathbf{y}^* = \mathbf{argmax}_{\mathbf{y} \in \mathcal{Y}} F(\mathbf{y})$.

■

The question we must ask is how best to generate the set of trees $\mathcal{T}_1$ so that we can be assured of exact inference with the smallest possible amount of computational effort. One intuitive method for achieving this is to think of the residual ν as a penalty for not including edges $\mathcal{E}_0$ into the inference scheme. This is the maximum contribution to the score function unaccounted for due to the exclusion of the edges $\mathcal{E}_0$. We can set about minimising the value of ν by repeatedly sampling trees from $S(G)$ in a greedy manner. To do this we assign each edge in the graph a weight $\omega_{i,j} = \max_u F_{i,j}(u)$, which corresponds to its contribution to the residual ν. We can then use a maximum spanning tree algorithm to compute the tree that minimises the value of ν. We set the value of the weights present in the sampled tree T to zero i.e. $\omega_{i,j} = 0$ if $(i,j) \in T$ and repeat this process until the value of ν reaches a given threshold. An alternative approach would be to apply a halving algorithm to the weights over the graph. One would imagine that this more moderate approach to edge removal would allow the discriminative edges to be present in in more of the spanning trees, and thus help during the inference stage to focus on important edges. Note that this process of generating trees for inference is likely to be different across examples $k \in \{1, \ldots, m\}$. This approach addresses the difficulties associated with performing inference using a fixed set of trees, where the discriminative power of these trees is likely to vary across training examples. It is reasonable to expect that edges will have a different level of importance across examples i.e. some edges will be particularly discriminative for some examples and not for others. We see that this method focuses on ensuring that we have the maximum possible scoring edges included in our search phase.

Implicit tree learning

In the last section we provided conditions showing that it was possible to perform exact inference using an arbitrary set of spanning trees. This showed that we were able to use trees that were different from the ones explicitly used to construct the predictor and presented a simple heuristic for selecting trees for inference. Following on from this, we now view the problem of assigning weight to the trees used in the construction of the predictor, and present a tree-free method for learning. In our method we propose to forget about maintaining an explicit set of trees with non-zero weight during learning but rather think of it along the same lines as the marginal dual polytope. In the context of trees, we simply maintain a weight attributed to each possible edge, where the weight represents the sum of the weights of all trees that have this edge present.

Rather than maintaining weights for an exponentially large set of possible trees we can sum-

marise this information into $\binom{\ell}{2}$ weights defined over edges, where $\lambda_e = \sum_{T \in S(G)} \lambda_T \mathbb{1}[e \in T]$. During the MKL update step we proceed as before, computing the tree using the maximum spanning tree principle, and shift the distribution of tree weights in the direction of the edges present in this tree. These conditional gradient steps ensure that we remain in the polytope defined by the unit L_1 norm combination of spanning trees. And by leveraging the inference scheme that we discussed in previous section, we no longer have to worry about maintaining knowledge of the trees that have non-zero weight in the ensemble as we are still able to perform inference using knowledge of the edge weights.

Be revisiting the condition for ϵ-optimality during multiple kernel learning, we see that the expression can be represented in terms of weighted edge margins γ_e and further reduced to involve total edge weightings λ_e. To see this let $\omega^* = \max_{T \in S(G)} \upsilon_T$ be the maximum gradient with respect to a tree weighting, we see that

$$\omega^* - \sum_{T \in \mathcal{T}} \lambda_T \upsilon_T = \omega^* - \sum_{T \in \mathcal{T}} \lambda_T \sum_{e \in T} \gamma_e = \omega^* - \sum_{e \in \mathcal{E}} \lambda_e \gamma_e < \epsilon \,.$$

Combining this reparameterisation of tree weights with the trees for inference scheme we see that we are capable of running the optimisation scheme without explicitly maintaining a fixed set of trees. Instead we generate trees when required for inference and summarise the tree weightings by maintaining a distribution of weights over edges.

4.5 Summary

In this chapter we discussed the use of graphical models for inference and learning, in particular we addressed the problem of large margin learning for multi-label output spaces. We presented a new method of multi-label learning that was agnostic to output graph structure and sought to bridge the gap between methods for parameter and structure learning. The goal of our algorithm is to simultaneously learn the intrinsic structure of the dataset whilst finding a large margin predictor on it. The solution was formulated as a multiple kernel learning problem where we assigned weights to spanning trees, which were then combined to given the overall graph structure.

To address the difficulties of inference on highly-connected graphs, we presented a Lemma that provided conditions to ensure exact inference had been performed on the graph composed of a combination of spanning trees, and that this could be performed in $\mathcal{O}\left(|\mathcal{T}|K\ell\right)$ time. Later we presented insights on how in this inference scheme can be developed in the future to use arbitrary trees and input dependent trees. We hope that these extensions should allow exact inference to be guaranteed using a smaller K-best list. The multiple

kernel learning framework was used to find the combination of spanning trees, thus graph structure, that maximised the margin on the training dataset. To the best of our knowledge this is the first such approach that connects multiple kernel learning with structure learning on graphical models. Furthermore, given that we were able to consider the space of exponentially many trees, in $\mathcal{O}(E \log V)$ time, via the maximum spanning tree algorithm on edge potentials makes this approach particularly interesting from both a graphical model perspective and wider multiple kernel learning approaches such as those used in [Bach, 2008] for group lasso.

Using a number of benchmark datasets we compared the performance of our new approach to several popular methods from the literature. In general we saw a small degree of improvement when using our structured learning approach, with the hamming loss variant performing on average better than the zero-one loss approach. We put this improvement down to the difficult obtaining a large margin on all exponentially many violators, which made it difficult for the algorithm to find smooth solutions that generalised well. When testing on the prediction of stock price movements, we saw the graph agnostic approach outperformed other methods, where structure was either assumed or ignored, in terms of both accuracy and return on the implied trading strategy. These preliminary results are somewhat encouraging, however we still believe the best of this approach is still to come. From a practical point of view, given that the alternating optimisation scheme uses adaptations of well known algorithms, Frank-Wolfe, dynamic programming and minimum spanning tree algorithms, one would expect it to be able to scale to much larger datasets than those that have been tested here.

We present this work at a time where deep learning largely dominates the machine learning research agenda, making considerable advances in fields such as image and speech recognition where the deep architectures are capable of learning complex relationships between variables. However, the cost of these architectures is a lack of transparency in the function that has been learned. We offer an alternative learning scheme, one that can explicitly learn relationships between output variables in an intuitive way through the combination of trees that capture relationships. We hope that the discussions we have outlined regarding future work will encourage researchers to pursue this line of research. This should lead to efficiency improvements in the implementation of the algorithm, enabling its application to larger real world graph structures where its agnostic approach may have considerable advantages to static counterparts.

Chapter 5

Final remarks

In this thesis we presented new methods that aim to make the best use of the data that is available. We began by considering the problem of binary classification for datasets where there are only a small number of observations. To aid with the design of our predictor, we incorporated the uncertainty of the empirical moments into an optimisation scheme that minimises the worst case future misclassification rate. Intuitively we saw that as the number of observations for a particular class increased, our confidence in the value of sampled moments grew, which allowed them to have a greater influence on the shape of the discriminant. We likened this approach to an implicit regularisation scheme that took into consideration the relative amount of information we have for each class.

Experimentally this new approach favoured well against popular methods, such as support vector machines and Fisher's discriminant, on a wide range of benchmark datasets. Furthermore we examined the performance of a number of classifiers in predicting the directional movement of foreign exchange rates using the relationship between the current price and past prices. For this problem, we saw that moment based approaches generally outperformed their support vector counterparts. We attributed this partly to the noisy market signal and also to the ability of moment based approaches to summarise the information rather than constructing solutions on the basis of the outliers.

We discussed several extensions to the initial HP-MPM that could be the focus of future research. One of the most straightforward was the incorporation of the class probabilities into the optimisation scheme. This would build upon the work done on minimum error probability error machines [Huang et al., 2004], but should also be able to take advantage of the methods we outlined earlier. A more challenging direction would be the design of an efficient algorithm for constructing a solution using only a subset of the input variables and an examination of the consequences on generalisation guarantees. Further work should consider how we employee kernel-based methods, which become computationally challenging as the

number of samples increase. Our approach sought to overcome difficulties with small sample sizes but it could also be used to penalise classes when we use a sparse selection of examples to make kernel approximations i.e. use the approximation residuals as the regularisers.

In the second half of this thesis we presented novel methods for multi-label classification. For this problem we have to learn a mapping between an arbitrary input variable to the labelling of output nodes on a graph. If we know the exact relationship between these output variables then it clearly makes the prediction problem easier; however, seldom in the real world is this the case and we must infer structural relationships from the data that we have observed. We presented a new approach that simultaneously learns a large margin predictor whilst uncovering the intrinsic structure of the dataset. The alternating optimisation procedure iterates between finding a large margin predictor for a fixed output structure and improving the structure to increase the margin potential. The output structure was represented using a linear combination of spanning trees, where each tree was given a weight and edge weights were given by the superposition of the spanning trees. We showed that using a linear combination of spanning trees, it was possible to obtain a margin that is at least as large as that when using the complete output graph. We cast this as an optimisation problem in terms of multiple kernel learning, where each spanning tree defined a kernel and our goal was to find the optimal combination of these kernels. Despite there being exponentially many trees to choose from, we showed that the direction of steepest descent i.e. the tree maximally violating MKL optimality conditions, could be found using a simple implementation of the maximum spanning tree algorithm.

The tree based representation of the output structure also allowed us to address the problem of exact inference i.e. finding the most likely output labelling for a particular input. We presented conditions that guaranteed exact inference using our weighted combination of spanning trees. This involved a K-best dynamic programming scheme over each of the weighted trees, and it was used during both learning, to find update directions, and prediction, to find most likely labelling. The need for exact inference wasn't so great during learning, especially in the early stages, since we only needed to find update directions so the size of K could typically remain small. During prediction we systematically increased the value of K until exact inference guarantees had been satisfied or we reached a specific threshold but unfortunately we were unable to provide guarantees of the maximum depth one would have to go to in order to be assured of high-probability exact inference. We discussed several methods to improve the inference scheme, a simple one being to vary K according to the weight of the tree so that more time is spent searching on trees that contribute most to the overall score function of the ensemble of trees. Another approach showed that it was possible to perform exact inference using a set of input dependent spanning trees, however

an additional residual penalty had to paid.

We compared the performance of this new approach to a number of multi-label learning methods, including independently trained SVMs and a random variant of our approach [Marchand et al., 2014]. In general our method performed on par with these existing approaches, seeing marginal improvements across a number datasets. However we believe that we have only scratched the surface of the relative benefits of this approach and that this is largely due to its current algorithmic implementation, which prevented us from trying it on larger output graphs and datasets.

References

A Aizerman, Emmanuel M Braverman, and LI Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control*, 25:821–837, 1964.

Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

Francis R Bach. Consistency of the group lasso and multiple kernel learning. *Journal of Machine Learning Research*, 9(Jun):1179–1225, 2008.

Francis R Bach and Michael I Jordan. Thin junction trees. In *Advances in Neural Information Processing Systems*, pages 569–576, 2001.

Francis R Bach, Gert RG Lanckriet, and Michael I Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the twenty-first international conference on Machine learning*, page 6. ACM, 2004.

David Barber. *Bayesian reasoning and machine learning.* Cambridge University Press, 2012.

Dimitris Bertsimas and Ioana Popescu. Optimal inequalities in probability theory: A convex optimization approach. *SIAM Journal on Optimization*, 15(3):780–804, 2005.

Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6.

Dimitris Bertsimas, Vishal Gupta, and Nathan Kallus. Data-driven robust optimization. *arXiv preprint arXiv:1401.0212*, 2013.

Julian Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 192–236, 1974.

Julian Besag. Efficiency of pseudolikelihood estimation for simple gaussian fields. *Biometrika*, pages 616–618, 1977.

Wei Bian, Bo Xie, and Dacheng Tao. Corrlog: Correlated logistic models for joint prediction of multiple labels. In *International Conference on Artificial Intelligence and Statistics*, pages 109–117, 2012.

Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

Joseph K Bradley and Carlos Guestrin. Learning tree conditional random fields. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 127–134, 2010.

Murray Campbell, A Joseph Hoane, and Feng-hsiung Hsu. Deep blue. *Artificial intelligence*, 134(1-2):57–83, 2002.

Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

C Chow and C Liu. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory*, 14(3):462–467, 1968.

Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics, 2002.

Gregory F Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial intelligence*, 42(2-3):393–405, 1990.

Corinna Cortes and Vladimir N Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995.

Simon Cousins and John Shawe-Taylor. High-probability minimax probability machines. *Machine Learning*, 106(6):863–886, 2017.

Simon Cousins, John Shawe-Taylor, Mario Marchand, Juho Rousu, and Hongyu Su. Multiple kernel learning for prediction on unknown graph structures.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

Petros Drineas and Michael Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6(Dec): 2153–2175, 2005.

Harris Drucker, Chris Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. *Advances in neural information processing systems*, pages 155–161, 1997.

Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

David A Ferrucci. Introduction to "this is watson". *IBM Journal of Research and Development*, 56(3.4):1–1, 2012.

Thomas Finley and Thorsten Joachims. Training structural svms when exact inference is intractable. In *Proceedings of the 25th international conference on Machine learning*, pages 304–311. ACM, 2008.

R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Human Genetics*, 7(2):179–188, 1936.

Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer Series in Statistics, 2001.

Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741, 1984.

Dorothy M Greig, Bruce T Porteous, and Allan H Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 271–279, 1989.

Johan Hammersley and Peter Clifford. Markov fields on finite graphs and lattices. 1971.

W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

David Heckerman. A tutorial on learning with bayesian networks. In *Learning in graphical models*, pages 301–354. Springer, 1998.

Mark Herbster and Manfred K Warmuth. Tracking the best expert. *Machine Learning*, 32 (2):151–178, 1998.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Gao Huang, Shiji Song, Zhixiang Eddie Xu, and Kilian Weinberger. Transductive minimax probability machine. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 579–594. Springer, 2014.

Gao Huang, Jianwen Zhang, Shiji Song, and Zheng Chen. Maximin separation probability clustering. In *AAAI*, pages 2680–2686, 2015.

Kaizhu Huang, Haiqin Yang, Irwin King, Michael R Lyu, and Laiwan Chan. The minimum error minimax probability machine. *The Journal of Machine Learning Research*, 5:1253–1286, 2004.

Kaizhu Huang, Haiqin Yang, Irwin King, and Michael R Lyu. Imbalanced learning with a biased minimax probability machine. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 36(4):913–923, 2006.

Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009.

Marius Kloft, Ulf Brefeld, Sören Sonnenburg, and Alexander Zien. Lp-norm multiple kernel learning. *Journal of Machine Learning Research*, 12(Mar):953–997, 2011.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. 2009.

Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. Block-coordinate frank-wolfe optimization for structural svms. pages 53–61, 2013.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

G.R.G. Lanckriet, L.E. Ghaoui, C. Bhattacharyya, and M.I. Jordan. A robust minimax approach to classification. *The Journal of Machine Learning Research*, 3:555–582, 2003.

Steffen L Lauritzen and David J Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 157–224, 1988.

Su-In Lee, Varun Ganapathi, and Daphne Koller. Efficient structure learning of markov networks using *l*_1-regularization. In *Advances in neural Information processing systems*, pages 817–824, 2006.

Mario Marchand and John Shawe-Taylor. The set covering machine. *Journal of Machine Learning Research*, 3(Dec):723–746, 2002.

Mario Marchand, Su Hongyu, Emilie Morvant, Johu Rousu, and John Shawe-Taylor. Multilabel structured output learning with random spanning trees of max-margin markov networks. In *Proceedings of Neural Information Processing Systems (NIPS)*, 2014.

A.W. Marshall and I. Olkin. Multivariate chebyshev inequalities. *The Annals of Mathematical Statistics*, 31(4):1001–1014, 1960.

Marina Meila and Michael I Jordan. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1(Oct):1–48, 2000.

Ofer Meshi, Elad Eban, Gal Elidan, and Amir Globerson. Learning max-margin tree predictors. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, UAI 2013, Bellevue, WA, USA, August 11-15, 2013*, 2013.

Sebastian Mika. *Kernel fisher discriminants*. PhD thesis, Universitätsbibliothek, 2002.

Sebastian Mika, Gunnar Ratsch, Jason Weston, Bernhard Scholkopf, and KR Mullers. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop.* IEEE, 1999.

Sebastian Mika, Gunnar Rätsch, and Klaus-Robert Müller. A mathematical programming approach to the kernel fisher algorithm. *Advances in Neural Information Processing Systems*, 2001a.

Sebastian Mika, Alexander Smola, and Bernhard Schölkopf. An improved training algorithm for kernel fisher discriminants. *In proceedings AISTATS 2001*, pages 98–104, 2001b.

Sebastian Mika, Gunnar Rätsch, Jason Weston, Bernhard Schölkopf, Alex Smola, and Klaus-Robert Müller. Constructing descriptive and discriminative nonlinear features: Rayleigh coefficients in kernel feature spaces. 2003. doi: 10.1.1.10.2471.

Marvin Minsky and Seymour Papert. *Perceptrons*. MIT press, 1988.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Sebastian Nowozin and Christoph H Lampert. Structured learning and prediction in computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 6(3–4):185–365, 2011.

Margarita Osadchy, Tamir Hazan, and Daniel Keren. K-hyperplane hinge-minimax classifier. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1558–1566, 2015.

Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1986.

Patrick Pletscher, Cheng Soon Ong, and Joachim M Buhmann. Spanning tree approximations for conditional random fields. In *AISTATS*, pages 408–415, 2009.

Alain Rakotomamonjy, Francis Bach, Stéphane Canu, Yves Grandvalet, et al. SimpleMKL. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

Pradeep Ravikumar and John Lafferty. Quadratic programming relaxations for metric labeling and markov random field map estimation. In *Proceedings of the 23rd international conference on Machine learning*, pages 737–744. ACM, 2006.

Stefan Riezler and Alexander Vasserman. Incremental feature selection and l1 regularization for relaxed maximum-entropy modeling. In *EMNLP*, pages 174–181. Citeseer, 2004.

Neil Robertson and Paul Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of algorithms*, 7(3):309–322, 1986.

Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Juho Rousu, Craig Saunders, Sandor Szedmak, and John Shawe-Taylor. Efficient algorithms for max-margin structured classification. 2007.

Arthur Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

Mark W Schmidt, Kevin P Murphy, Glenn Fung, and Rómer Rosales. Structure learning in random fields for heart motion abnormality detection. In *CVPR*, volume 1, page 2, 2008.

Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, pages 1443–1471, 2001.

Nicol Schraudolph and Dmitry Kamenetsky. Efficient exact inference in planar ising models. In *Advances in Neural Information Processing Systems*, pages 1417–1424, 2009.

John Shawe-Taylor and Nello Cristianini. Estimating the moments of a random vector with applications. *In Proceedings of GRETSI 2003 Conference*, 2003.

John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

Solomon Eyal Shimony. Finding maps for belief networks is np-hard. *Artificial Intelligence*, 68(2):399–410, 1994.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Marina Sokolova, Mario Marchand, Nathalie Japkowicz, and John Shawe-Taylor. The decision list machine. In *Advances in Neural Information Processing Systems*, pages 921–928, 2002.

Nathan Srebro. Maximum likelihood bounded tree-width markov networks. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 504–511. Morgan Kaufmann Publishers Inc., 2001.

Thomas Strohmann, Andrei Belitski, Gregory Grudic, and Dennis DeCoste. Sparse greedy minimax probability machine classification. *Advances in Neural Information Processing Systems*, 16, 2004.

Hongyu Su and Juho Rousu. Multilabel classification through random graph ensembles. In *Asian Conference on Machine Learning*, pages 404–418, 2013.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. *Advances in neural information processing systems*, 16:25, 2004.

Yee Whye Teh, Max Welling, Simon Osindero, and Geoffrey E Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4(Dec): 1235–1260, 2003.

Gerald Tesauro. Temporal difference learning and TD-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

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

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*, page 104. ACM, 2004.

Vladimir N Vapnik. *Statistical learning theory*, volume 2. Wiley New York, 1998.

Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Rep. Academy Sci. USSR*, 1968.

Vladimir N Vapnik and A Ya Chervonenkis. *Theory of Pattern Recognition [in Russian].* Nauka, 1974.

Vladimir N Vapnik and A Ya Chervonenkis. The necessary and sufficient conditions for consistency of the method of empirical risk minimization. *Pattern Recognition and Image Analysis 1*, pages 284–305, 1991.

Grace Wahba. *Spline models for observational data.* SIAM, 1990.

Martin J Wainwright, Tommi Jaakkola, and Alan S Willsky. Tree-based reparameterization for approximate inference on loopy graphs. In *Advances in neural information processing systems*, pages 1001–1008, 2001.

Martin J Wainwright, Tommi S Jaakkola, and Alan S Willsky. Map estimation via agreement on trees: message-passing and linear programming. *IEEE transactions on information theory*, 51(11):3697–3717, 2005.

Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

Chris Williams and Matthias Seeger. Using the nystroem method to speed up kernel machines. *Advances in Neural Information Processing Systems 13*, 2001.

Ronald J. Williams and Geoffrey E. Hinton. Learning representations by back-propagating errors. *Nature*, 323(6088):533–538, 1986.

David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

Jonathan S Yedidia, William T Freeman, and Yair Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.

Guo-Xun Yuan, Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. A comparison of optimization methods and software for large-scale l1-regularized linear classification. *Journal of Machine Learning Research*, 11(Nov):3183–3234, 2010.

Tong Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, pages 56–85, 2004.