UNIVERSITY OF CALIFORNIA
RIVERSIDE

Towards Information-Economical Classification and Ranking

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Kin Fai Kan

August 2008

Dissertation Committee:

Dr. Christian R. Shelton, Chairperson
Dr. Eamonn Keogh
Dr. Neal Young

The Dissertation of Kin Fai Kan is approved:

_____

_____

_____

Committee Chairperson

University of California, Riverside

Acknowledgments

Firstly, I would like to thank my advisor Dr. Christian Shelton for his guidance and support. Christian helped me tremendously in creating this dissertation. He introduced me the sequential stopping problem and kindly let me be a co-author of the "Chained boosting" paper. Most importantly, he taught me to use the empirical risk principle to solve machine learning problems. Christian also helped me a lot to survive the pressure of PhD study. He did not only tolerate my slow progress but also tried his best to satisfy my needs and demands.

Secondly, I would like to thank Dr. Eamonn Keogh, Dr. Neal Young, and Dr. Victor Zordan for their advice and encouragement. They have been very nice to me. Although I did not work with them, they spent time chatting with me and shared their ideas and experiences with me. I am also grateful to the professors in CS, EE, and Statistics department from whom I learned a lot of interesting stuff.

Thirdly, I am thankful to the friends and classmates I met at UCR. Particularly, I would like to mention Dragomir Yankov and Teddy Yap. They are always there to listen to my half-baked ideas, silly opinions, and whining!

Lastly, I would like to thank my whole family. Their support and belief in me are my source of energy to move forward in the face of hardship. I dedicate this dissertation to my mother, who always thought about me before herself.

ABSTRACT OF THE DISSERTATION

Towards Information-Economical Classification and Ranking

by

Kin Fai Kan

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, August 2008
Dr. Christian R. Shelton, Chairperson

The standard formulations of classification and ranking are rather strict in terms of information usage and may incur high information costs. Standard classification assumes that we need all the attributes to classify every test example. This is neither cheap nor necessary because the attributes can be expensive to obtain and quite often we can predict the label of a test example by looking at a small subset of the attributes. A more economical approach is to acquire the attributes of test examples sequentially and weigh the benefit and cost of acquiring more attributes at every step. To do this, we need to learn a sequence of decision rules that together minimizes the penalty costs due to misclassification and information acquisition costs. We focus on rejecting negative examples as early as possible and present the catenary support vector machine (catSVM).

Standard classification learns a prediction function from a set of labeled examples.

However, it is often expensive (and sometimes impossible) to obtain labels of individual examples. One economical alternative is to learn from aggregate label information which are easily available and cheap to obtain. We propose an SVM method to learn classification from group label proportions and provide a theoretical bound on its generalization error. The idea of learning from aggregate label information is also useful for ranking. Standard ranking relies on the relative preferences between individual examples for training which can be expensive or difficult to obtain. We propose a probabilistic model for the relative preferences among groups of individual examples and present several estimation methods.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Classification is a fundamental problem in machine learning and statistics. Given a training set consisting of a number of input objects together with their output labels, the goal of classification is to learn a model to predict the output labels of future input objects. Usually input objects are represented by feature vectors of real numbers and in binary classification, output labels are either $+1$ or -1 indicating positive or negative class. Classification has numerous real-world applications. For example, a mortgage company migth use classification to predict whether a customer is likely to repay a loan based on her personal information. Also, biologists use classification to predict the function of new proteins based on their sequence structures. Table 1.1 shows some training and test examples for the task of classifying mortgage applicants. The input objects are mortgage applicants who are described by age, sex, marital status, and income. The output labels indicate whether the applicants repaid the loans. The widely used classifiers are decision tree, k-nearest-neighbor classifier,

Age	Sex	Marital status	Income	Repay?
20	M	Single	10k	No
25	M	Married	60k	Yes
36	F	Single	200k	Yes
48	M	Divorced	90k	No

(a) Training examples

Age	Sex	Marital status	Income	Repay?
27	M	Single	50k	?
55	F	Married	110k	?

(b) Test examples

Table 1.1: An example of classification task: mortgage applicants classification

naive Bayes classifier, neural network, support vector machine, and logistic regression [7].

Ranking is a relatively new problem compared to classification. It has gained considerable attention in the machine learning community in recent years. Given a training set consisting of pairs of input objects together with their preference relations, the goal of ranking is to learn a scoring function that assigns a higher score to a more preferred input object. A scoring function induces an ordering over all possible input objects. Ranking has important applications in information retrieval. For instance, search engines rank web pages depending on their relevance to a user's query. It is reasonable to assume that a web page is more relevant than the others if a user clicks to visit that web page. Thus, we can extract preference relations between web pages from query logs and use the extracted preference relations to learn the relevance function for a query. Ranking is closely related to classification. In fact, we can reduce ranking into binary classification by regarding a pair of input objects as a single input object and the preference relation between a pair of input objects as a binary label. A number of classification algorithms have been adapted to solve ranking, namely RankBoost [19], RankSVM [27], and RankNet [9].

Although information is everywhere, useful information often comes with a price. The

standard formulations of classification and ranking are rather strict in terms of information usage and hence not ideal in many scenarios. First, they do not take the cost of information collection into account during testing. Standard classification assumes that we need all input features to test every example. This is neither economical nor necessary because input attributes of a test example do not come for free and quite often we can predict the label of a test example by just looking at a small subset of input features. A more economical approach is to obtain input features sequentially at the test phase and weigh the benefit and the cost of acquiring more input features at every step. In this way, we reformulate binary classification as sequential stopping problem. For instance, to diagnose heart disease, a doctor conducts a number of tests on a patient. Clearly, there is no need to perform all relevant tests for every patient. The common practice is to conduct medical tests sequentially. Based on the currently available test results, the test costs, as well as the penalty costs for mis-diagnosis, a doctor decides whether to rule out the possibility of heart disease completely or to perform additional tests to confirm the diagnosis.

Second, standard classification and ranking rely on labels and preference relations about individual examples which are often expensive (and sometimes impossible) to obtain. Semi-supervised learning [11] provides one solution. It uses plenty of unlabeled examples in addition to a small number of labeled examples to train a classifier (or a ranking function). Recently, some researchers have proposed to learn classification and ranking from aggregate statistics [16, 29, 36, 4, 25]. In this setting, a training set consists of groups

Age	Sex	Ethnicity	Income	For?
20	M	Latino	10k	
25	M	Asian	60k	3 Yes
36	F	White	200k	1 No
48	M	Black	90k	
69	M	Black	30k	
71	F	White	150k	1 Yes
54	F	White	500k	3 No
87	M	White	0	

(a) Training examples

Age	Sex	Ethnicity	Income	For?
27	M	Latino	50k	?
55	F	Black	110k	?
33	F	White	300k	?

(b) Test examples

Table 1.2: An example of group statistics classification: voter classification

of individual examples and each group is associated with some aggregate statistics about individual labels. Different forms of aggregate statistics have been considered. Multiple instance learning [16] relies on aggregate label that indicates the presence of any positive example in a group multiple examples. Group statistics classification [29, 36] relies on the fraction of positive examples in each group. Multiple instance ranking [4, 25] relies on the knowledge of the group that contains the highest ranking example in a collection of groups of individual examples. Since aggregate statistics are often cheap and easy to obtain, learning from aggregate statistics is a promising solution to reduce the collection cost of labeled information and has many potential applications. For example, a political party wants to find out who would vote for their presidential candidate. It may be difficult to know exactly how individual people vote. It is straightforward to find out how many votes each candidate gets in different areas of the country. Together with an existing database of registered voters, a political party can learn a model to predict how individual people vote. Table 1.2 illustrates what the training and test data may look like. Table 1.2(a) shows two groups of voters from two different areas of the country. The input objects (i.e., the voters)

are described by age, sex, ethnicity, and income. The output labels indicate whether the voters vote for the candidate and they are unknown. But we know that the candidate gets 3 votes out of 4 in the first group and he just gets one vote in the second group. Another possible application is to use multiple instance ranking to learn the relevance of sections or paragraphs in web documents based on the relative preferences of web documents. Suppose someone enters a search query “multiple instance ranking”. This dissertation may be returned as one search result because it contains a chapter on multiple instance ranking. It may be helpful if search engines do not only find this dissertation relevant but can also identify the most relevant chapter. We can represent this dissertation by multiple document frequency vectors, one for each chapter, and rank the chapters based the relative preferences at the document level (which can be extracted from query logs).

In this dissertation, we study sequential stopping problem, group statistics classification, and multiple instance ranking. These new formulations of classification and ranking pose significant challenges. The sequential stopping problem requires learning a sequence of decision rules to achieve a common goal. Group statistics classification and multiple instance ranking requires dealing with aggregate statistics which are ambiguous in nature. Our contributions are summarized as below.

1. We propose catenary support vector machine (catSVM) to solve sequential stopping problem and provide a generalization bound for catSVM.

2. We propose a SVM approach to learn to classify individual examples from group

statistics and provide a generalization bound for group statistics classification.

3. We propose a probabilistic model for multiple instance ranking and derive methods to estimate the model.

Chapter 2

Background

In this chapter, we will review some fundamental concepts. First, we will look at the principle of empirical risk minimization and common loss functions in classification and regression. Then, we will review boosting, support vector machines, and some theory of kernel functions. We will also discuss how to derive boosting algorithms and SVM from the risk minimization principle. After that, we will see how to minimize an important class of nonconvex functions – the difference of convex functions.

2.1 Empirical Risk Minimization

The task of classification (or regression) is to predict the output label of an example from its input features. Let $\mathcal{X}$ be the space of input attributes and $\mathcal{Y}$ be the space of output label. We want to learn a function $f : \mathcal{X} \to \mathcal{Y}$ that predicts the output label of any test example well. We posit that test examples are drawn independently from an unknown but

fixed probability distribution $P(x, y)$ (the i.i.d. assumption). Suppose we have a function $L(y, f(x))$ that can quantify the loss of predicting $f(x)$ when the true output label is y. It makes sense to find a predictive function $f(x)$ that minimizes the risk $E_P[L(Y, f(X))]$. Since $P(x, y)$ is unknown, we approximate it using a set of labeled training examples $\{x_i, y_i\}_{i=1}^n$. Thus, we seek to find a predictive function $f(x)$ that minimizes the empirical risk

$$E_{\hat{P}}[L(Y, f(X))] = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i))$$

where $\hat{P}$ is a uniform distribution on the n training examples. Under mild conditions on $L(y, f(x))$, one can prove that the empirical risk converges to the true risk as the number of training examples increases [50]. However, for a small number of training examples, the true risk of a predictive function $f(x)$ can be much greater than the empirical risk especially if the class of predictive functions we consider has a high complexity. This problem is known as overfitting and can be avoided by adding a regularization term to encourage simple $f(X)$. We introduce a complexity term $R(f)$ and minimize the regularized risk

$$E_{\hat{P}}[L(Y, f(X))] + \lambda R(f),$$

where λ is parameter that controls the tradeoff between the empirical risk and the complexity of the predictive function.

So far, we have not mentioned how to pick the loss function L. We cannot overstate the importance of picking the *right* loss function. There are two criteria to consider.

1. Does the loss function capture the actual loss well?

2. Is the loss function easy to optimize?

In binary classification, the output label of an example can either be $+1$ or -1. The natural loss function is the 0-1 loss.

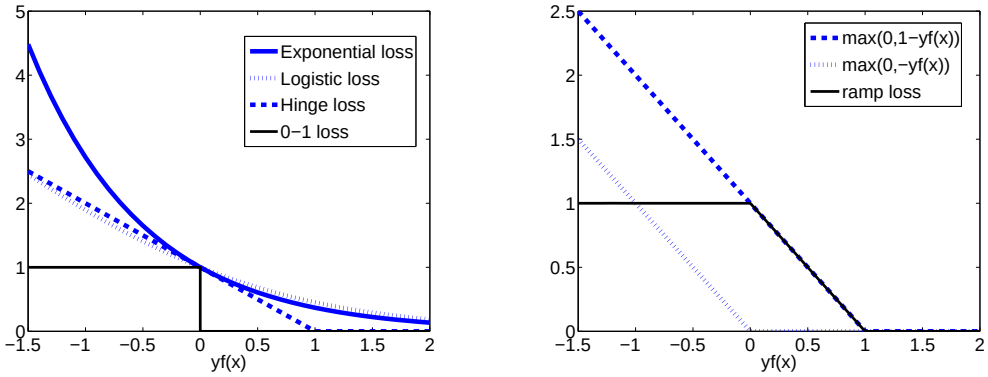
$$L_{0-1}(y, f(x)) = \begin{cases} 1 & \text{if } yf(x) \leq 0 \\ 0 & \text{otherwise.} \end{cases}$$

Here we assume that $f(x)$ is a real-valued function and an example is classified as $+1$ if $f(x)$ is positive and -1 otherwise. Therefore, we have a classification error if y and $f(x)$ have opposite signs. The major drawback of the 0-1 loss is that it is a discontinuous function and hence difficult to optimize. A number of alternative loss functions have been proposed. AdaBoost [20] uses exponential loss, logistic regression [7] uses logistic loss, and support vector machines [51] use hinge loss.

$$L_{Exp}(y, f(x)) = e^{-yf(x)}$$

$$L_{Logit}(y, f(x)) = log(1 + e^{-yf(x)})$$

$$L_{Hinge}(y, f(x)) = max(0, 1 - yf(x))$$



(a) (b)

Figure 2.1: Classification loss functions: (a) convex loss functions, (b) ramp loss

Figure 2.1(a) depicts exponential loss, logistic loss, hinge loss, as well as the 0-1 loss as a function $yf(x)$. Exponential loss and logistic loss are continuous and differentiable while hinge loss is continuous but not differentiable at $yf(x) = 1$. Importantly, exponential loss, logistic loss, and hinge loss are all convex in $f(x)$. They have one unique minimum and they can be optimized using efficient convex algorithms. However, since they are unbounded above for classification mistakes, they do not capture the 0-1 loss very well. With an unbounded loss function, it may appear better off to make many classification mistakes than just classify one single example wrongly. On the other hand, Ramp-SVM [13] and ψ-learning [40] use ramp loss which is always bounded above by 1 (also known as truncated hinge loss). Ramp loss can be expressed as the difference of two convex functions and hence is not convex.

$$L_{ramp}(y, f(x)) = max(0, 1 - yf(x)) - max(0, -yf(x))$$

Figure 2.1(b) shows ramp loss as a function of $yf(x)$.

For linear regression, the most popular loss function is the squared error

$$L_{SE}(y, f(x)) = (y - f(x))^2.$$

The squared error leads to an analytic solution, so the optimization is very efficient. It is well-known that the squared error can be interpreted as implicitly assuming independent Gaussian noise in output labels. However, the squared error is not appropriate if the assumption of independent Gaussian noise does not hold. One alternative, used in support vector regression [38], is ϵ-insensitive loss

$$L_\epsilon(y, f(x)) = \begin{cases} |y - f(x)| - \epsilon & \text{if } |y - f(x)| \geq \epsilon \\ 0 & \text{otherwise.} \end{cases}$$

When ϵ is chosen to be zero, ϵ-insensitive reduces to the absolute error. ϵ-insensitive loss is convex in $f(x)$, albeit it does not have an analytic solution. It is more appropriate than the squared error if the loss increases slowly with the prediction error. Figure 2.2 shows the squared error and ϵ-insensitive loss as a function of $y - f(x)$.

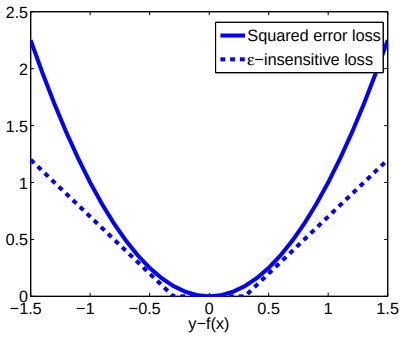


Figure 2.2: Regression loss functions

2.2 Boosting

Boosting is a technique that combines a number of weak learners into a strong learner. Roughly speaking, a weak learner is simple and not very accurate (barely better than random guess) while a strong learner is more sophisticated and can attain high accuracy. Boosting trains one weak classifier at a time on a weighted version of the given training set. Initially, every training example has equal weight. After a weak classifier is trained, the weights of the training examples that are wrongly (correctly) classified are increased (decreased). The intuition is to give more attention to the more difficult training examples in the later iterations. In addition to updating the weights of training examples, a weight is computed for every weak classifier based on their accuracy on the weighted training examples. The final strong classifier is specified by the weighted voting majority of the weak classifiers.

There are many variants of boosting. Perhaps the most well-known one is AdaBoost proposed by Freund and Schapire [20]. AdaBoost computes the multiplicative update of

the weight of the i-th training example (x_i, y_i) using the following equation.

$$w_i \leftarrow w_i e^{-\alpha_t y_i h_t(x_i)}$$

where h_t is the new weak classifier and α_t is its weight. The multiplicative update is less than 1 if $y_i = h_t(x_i)$ and is greater than 1 if $y_i \neq h_t(x_i)$. The weight α_t of h_t is

$$\alpha_t = \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}$$

where ϵ_t is the error rate of h_t on the weighted training set. Mason et al. [33] points out that AdaBoost can be understood as performing gradient descent in a function space to minimize the empirical exponential loss. This risk minimization perspective leads to the AnyBoost framework [33]. Given a loss function L, the objective is to minimize the empirical risk $J(F) = \frac{1}{n} \sum_{i=1}^{n} L(y_i F(x_i))$. We start with $F(X) \equiv 0$ and add one step $\alpha f(X)$ to $F(X)$ iteratively. The direction of the step $f(X)$ is chosen to match the negative gradient of the empirical risk $\nabla J(F)$. The step size α is chosen to minimize $J(F)$ along $f(X)$. To match $f(X)$ with $-\nabla J(F)$, we minimize the inner product between $f(X)$ and

$\nabla J(F)$

$$\min_{\{f(x_i)\}_{i=1}^n} \sum_{i=1}^{n} f(x_i) y_i L'(y_i F(x_i))$$

$$\Leftrightarrow \min_{\{f(x_i\}_{i=1}^n} \sum_{i=1}^{n} (-f(x_i) y_i)(-L'(y_i F(x_i)))$$

$$\Leftrightarrow \min_{\{f(x_i)\}_{i=1}^n} \sum_{i=1}^{n} I[f(x_i) \neq y_i] D(i)$$

where $D(i) = \frac{-L'(y_i F(x_i))}{Z}$ and Z is the normalization term. Note that D is a valid probability distribution because L is a loss function and should have a negative derivative. Therefore, every descent step is equivalent to finding the best classifier f on the weighted training examples.

2.3 Support Vector Machines

The support vector machine (SVM) is one of the most important algorithms in machine learning [38]. SVM is based on a simple intuition. A hyperplane that separates data of different classes with a large margin is likely to generalize well to future examples. The bigger the margin, the better the classification rule generalizes to future examples. SVM was originally designed for binary classification and it has been extended to linear regression [17], one-class classification [37], multiclass classification [14], ordinal regression [12], and ranking [27]. It has also inspired maximum-margin-based algorithms for clustering [58], feature selection [23], distance metric learning [56], kernel learning [30], matrix

factorization [44], and structured prediction [48].

2.3.1 Linear SVM

Let $\{(x_i, y_i)\}_{i=1}^n$ be a training set with $(x_i, y_i) \in \Re^d \times \{-1, +1\}$. The SVM optimization problem can be formulated as a quadratic program.

$$\begin{aligned} \min_{w,b,\{\xi_i\}_{i=1}^n} & \sum_{i=1}^{n} \xi_i + \lambda \|w\|^2 \\ \text{s.t.} \quad & y_i(w \cdot x_i - b) \geq 1 - \xi_i, && \forall 1 \leq i \leq n \\ & \xi_i \geq 0, && \forall 1 \leq i \leq n \end{aligned}$$

In this formulation, w is the normal vector of the hyperplane and the inverse of its norm $\frac{1}{\|w\|}$ is chosen to be the size of the margin. The positive examples should be on the side of the separating plane pointed to by the normal vector while the negative examples should be on the opposite side. Thus, the constraints imply that every x_i should be further away from the separating hyperplane than the size of the margin. The slack variables $\{\xi_i\}_{i=1}^n$ are used to give SVMs some flexibility to allow some noisy examples or outliers to fall within the margin or even on the wrong side of the separating hyperplane. The objective function is the sum of the slack variables and the squared norm of (the normal vector of) the separating hyperplane. λ is a parameter used to control the tradeoff between the slack variables and the norm of the separating hyperplane. By solving the SVM quadratic program, we obtain a hyperplane that separates the examples with a large margin.

Notice that the SVM quadratic program can be re-formulated as an unconstrained optimization problem.

$$\min_{w,b} \quad \sum_{i=1}^{n} \max(0, 1 - (w \cdot x_i - b)) + \lambda \|w\|^2$$

The first term is the empirical hinge loss. The second term can be regarded as a regularizer that prefers a separating hyperplane with a small ℓ_2 norm. Therefore, we can also interpret SVM as an example of regularized risk minimization. This risk minimization perspective often provides a more convenient way to design new SVM algorithms for new problems than the geometric perspective does.

2.3.2 Nonlinear SVM

The linear SVM is good for (almost) linearly separable data. For nonlinear separable data, we can use kernel functions to implicitly map the original data space to a (possibly infinitely dimensional) feature space and find a separating hyperplane in the feature space. Let $\mathcal{X}$ be the space of input objects. A kernel function $K(x, x')$ measures the similarity between $x, x' \in \mathcal{X}$. $K(x, x')$ is a positive semi-definite kernel if for any subset $\{x_i\}_{i=1}^m \subseteq \mathcal{X}$, $\sum_{i=1}^m \sum_{j=1}^m K(x_i, x_i) c_i c_j \geq 0$ for any real numbers $c_{i\,i=1}^m$. Every positive semi-definite kernel function corresponds to the inner product in some feature space. For instance, the polynomial kernel $K(x, x') = (x \cdot x' + c)^q$ corresponds to a mapping to an $\binom{d+q}{q}$-dimensional feature space, containing all monomials of the form $x_{i_1} x_{i_2} \ldots x_{i_\ell}$ that are up to order q.

Below is the dual formulation of SVM. It can be obtained by setting the derivative of the Lagrangian of the primal SVM to zero with respect to the primal variables w, b, and $\{\xi_i\}_{i=1}^n$.

$$\begin{aligned}
&\max_{\{\alpha_i\}_{i=1}^n} \sum_{i=1}^{n} \alpha_i - \frac{1}{2\lambda} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j \\
&\text{s.t.} \quad \sum_i \alpha_i y_i = 0 \\
&\qquad 0 \leq \alpha_i \leq 1, \qquad\qquad\qquad\qquad \forall 0 \leq i \leq n
\end{aligned}$$

The key observation is that in the dual formulation the training data only appears in the form of inner product. We can replace the inner products by a kernel function. In doing so, we can find a separating hyperplane in a feature space without explicitly calculating the feature space representation of the training data. The technique is known as the *kernel trick* [38]. The kernel trick does not only enable SVM to deal with nonlinear separable data, but also makes SVM applicable to data that are not represented by real-valued vector (e.g., strings of letters).

Alternatively, we can obtain a similar result by resorting to the theory of reproducing kernel Hilbert space (RKHS). Let us review some basics about reproducing kernel Hilbert space. We regard the kernel function evaluated at x, $K(\cdot, x)$ as a function that measures the similarity between x and any object in $\mathcal{X}$. Informally speaking, the reproducing kernel Hilbert space $\mathcal{H}$ associated with kernel function $K(x, x')$ is the space of

functions that are linear combination of kernel function evaluated at objects in $\mathcal{X}$. Let $f(\cdot) = \sum_{i=1}^{m} \alpha_i K(\cdot, x_i)$ and $g(\cdot) = \sum_{i=1}^{m'} \beta_i K(\cdot, x'_i)$ be two functions from the RKHS associated with kernel function $K(x, x')$. The inner product between $f(\cdot)$ and $g(\cdot)$ is defined as

$$< f, g >= \sum_{i=1}^{m} \sum_{j=1}^{m'} \alpha_i \beta_j K(x_i, x'_j) \quad .$$

The corresponding RKHS norm is $\|f\|_{\mathcal{H}} = \sqrt{< f, f >}$. Intuitively, to make $\|f\|_{\mathcal{H}}$ small, we require α_i's to be small in magnitude.

Consider a general risk minimization problem.

$$\min_{f \in \mathcal{H}} L(f, \{x_i, y_i\}_{i=1}^{n}) + \Omega(\|f\|_{\mathcal{H}})$$

where $\{x_i, y_i\}_{i=1}^{n}$ are the training data, L is some function depending on y_i's and the values of f at x_i's, and Ω is a monotonically increasing function. The Representer theorem states that every minimizer of the above problem admits a representation of the form

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i K(\cdot, x_i) \quad .$$

The Representer theorem can be proved by the orthogonality argument. For details of the proof and a more formal treatment of reproducing kernel Hilbert space, readers can refer to [38]. Compared to the kernel trick, the Representer theorem is especially attractive when the dual formulation of the optimization problem is difficult to derive. Applying the

Representer theorem to the primal SVM, we obtain

$$\begin{aligned}
\min_{\{\alpha_i\}_{i=1}^n, b, \{\xi\}_{i=1}^n} & \sum_{i=1}^{n} \xi_i + \lambda \sum_{i,j} \alpha_i \alpha_j K(x_i, x_j) \\
\text{s.t.} \quad & y_i \left(\sum_{j=1}^{n} \alpha_j K(x_j, x_i) - b \right) \geq 1 - \xi_i, \qquad \forall 1 \leq i \leq n \\
& \xi_i \geq 0, \qquad \forall 1 \leq i \leq n
\end{aligned}$$

2.4 The Concave-Convex Procedure

So far we have focused on convex loss functions. However, we might like to consider nonconvex loss functions. In general, it is difficult to minimize a nonconvex function efficiently. However, some efficient algorithms have been developed for some special class of nonconvex functions. We focus on the difference of convex (D.C.) functions and the Concave-Convex procedure. D.C. functions are nonconvex and may have multiple local minima. D.C. functions are very common and appear in kernel methods with missing data [41], kernel selection [2], ramp-SVM [13], sparse PCA [45], and semi-supervised learning [13].

The Concave-Convex procedure was first introduced by Yuille and Rangarajan [59] to solve minimization problems whose objective functions can be expressed as the sum of a convex part and a concave part. While Yuille and Rangarajan considered only linear constraints, Smola et al. [41] generalized the CCCP to handle concave-convex constraints. The

CCCP is an iterative procedure. In each iteration, it replaces the concave parts of the objective function and the constraints by their first-order Taylor approximations. The resulting problem is convex and can be solved using efficient convex minimization algorithms.

Consider the following optimization problem:

$$\begin{aligned} \min \quad & f_0(x) - g_0(x) \\ \text{s.t.} \quad & f_i(x) - g_i(x) \leq c_i \quad \forall i \end{aligned}$$

where f_i and g_i are real-valued convex and differentiable functions on $\Re^n$ for $i \in \{0, \ldots, m\}$, and $c_i \in \Re$ for $i \in \{1, \ldots, m\}$. The CCCP computes $x^{(t+1)}$ from $x^{(t)}$ by solving the following convex optimization problem.

$$\begin{aligned} & \min_x f_0(x) - \left(g_0(x^{(t)}) + \nabla g_0(x^{(t)})^T (x - x^{(t)})\right) \\ & \text{s.t.} \quad f_i(x) - \left(g_i(x^{(t)}) + \nabla g_i(x^{(t)})^T (x - x^{(t)})\right) \leq c_i \quad \forall i \end{aligned}$$

It can be shown that the CCCP converges to a local minimum [41]. In case of a non-global minimum, one may restart the CCCP with a different $x^{(0)}$. Notice that the CCCP can be seen as a special case of D.C. programming. Tao and An [46] states that the D.C. minimization algorithm (DCA) often converges to a global solution.

Chapter 3

Caternary Support Vector Machine

Many problems require making sequential decisions. For these problems, the benefit of acquiring further information must be weighed against the costs. In this chapter, we describe the *catenary support vector machine* (catSVM), a margin-based method to solve sequential stopping problems. We provide theoretical guarantees for catSVM on future testing examples. We evaluated the performance of catSVM on UCI benchmark data and also applied it to the task of face detection. The experimental results show that catSVM can achieve a better cost tradeoff than single-stage SVM and chained boosting.

3.1 Introduction

In many problems, information are obtained sequentially and the benefit of further information acquisition must be weighed against the costs at every step. In product testing, parts are inspected throughout the manufacturing process. Humans or computers must decide

whether to continue manufacturing or whether to stop (in case the piece is not salvageable). In medical diagnosis, doctors, patients, and insurers must decide whether the current information is sufficient to make a decision or whether to conduct the next of a bank of tests.

In object detection in images, a similar problem is faced. Scanning an image for an object of interest takes processing time. If the image can be scanned more quickly or at a lower resolution (reducing the number of pixels to be examined), the detection can be sped up. In doing so, the speed of detection must be weighed against the accuracy of detection.

Most classification methods assume full information about testing examples and are thus not suitable for sequential decision making scenarios. Recently, we proposed chained boosting to solve sequential stopping problem [39]. We assume that the relative costs of stopping at each stage are known and can be made explicit. Given the stopping costs for each training example, the goal is to minimize the cost of the decision rules applied to future examples. The difficulty of the problem lies in the fact that the decisions in later stages depend on what happens in early stages. Motivated by the success of support vector machines (SVMs) in many classification problems, we present the *catenary support vector machine* (catSVM), a novel margin-based method to solve sequential stopping problems.

3.2 Related work

We are interested in direct estimation of a sequence of decision rules. For this reason we are not considering density estimation (like a hidden Markov model) followed by a cost analysis to derive the decision rules. This rules out approaches like influence diagrams [24] as we would like to skip the density estimation step.

Our formulation (see the next section) appears similar to cascade classification [52, 57, 43] in that there are stages of classification. For applications like face detection, negative examples are far more frequent than positive examples. Rejecting negative examples as quickly as possible is crucial to the speed of the classification process. Viola and Jones [52] propose an iterative approach to train the cascade. In each iteration, a new stage is added to the cascade and a new stage classifier is trained to achieve a very low false negative rate and an approximately 50% false positive rate using a modified AdaBoost algorithm. Stages are added to the cascade until the number of false positives is reduced below a small number on a validation set. Bi et al. [5] propose to use 1-norm SVM as the stage classifiers in the cascade. Like Viola and Jones, their approach trains the stage classifiers sequentially from the first stage to the last stage. In every stage, an 1-norm SVM is trained to minimize the sum of the weighted errors and the regularization term.

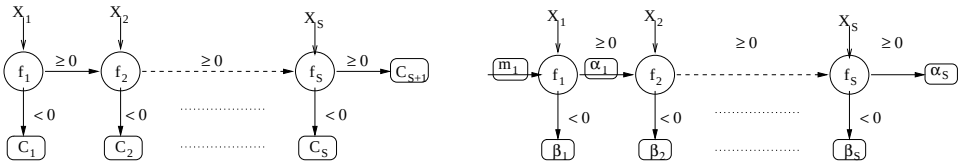
There are two major differences between our problem formulation and cascade classification. First, although classification speed is important, we are mainly concerned about the

costs of gathering information (i.e., the feature costs). Also, while cascade classification requires the user to choose the desired false negative rate and false positive rate at every stage, we assume that the feature costs and the misclassification costs are explicitly specified by the user and our algorithm automatically determines the best tradeoff between the feature cost and the two types of errors.

Second, we optimize the stage classifiers as a group to maximize the overall performance of the processing pipeline. The problem formulation allows the information available to change at each stage. Thus, false-positive and false-negative rates at each stage are not sufficient. It matters *which* positive examples are incorrectly rejected at a stage, not just *how many*. In particular, examples for whom further processing would still result in the incorrect classification should be rejected, while those for whom further information would clarify their classification should be saved.

Cascade classification and catSVM are both "staged" classifiers, but they are more complementary than competitive. The former attempts to speed up the computation of a single classification task (fixed information) by exploiting the asymmetric distribution of examples while the latter attempts to speed up a decision task by exploiting the correlation between data sources gathered at different times. One could well imagine using cascade classifiers at each stage within the framework shown here.

Recently, Dundar and Bi [18] consider the problem of jointly optimizing cascaded SVM classifiers. However, they ignore the difference of rejecting an example at different stages.



(a) (b)

Figure 3.1: An example of a processing pipeline

They formulate a non-convex and non-linear objective function and propose a cyclic optimization algorithm to optimize it.

3.3 Sequential Stopping Problem

We make no assumptions about the structure of the costs. Rather, we assume that each training example carries a cost vector indicating the costs of stopping after each stage. These costs may increase, decrease, or have any other arbitrary relationship with the stage index. The costs might be function of a "label" or might be different for each example.

Let S be the number of stages in the processing pipeline. Denote the feature vector and the costs of an example by x and c, respectively. Let x_j be the components of x that are available at the j-th stage. Let c_j be the total cost of rejecting the example at the j-th stage and c_{S+1} be the total cost of accepting it (allowing it to "pass" at each decision). We assume that c is drawn from a known set $\mathcal{C}$. In the case of binary classification, $\mathcal{C}$ might be of cardinality 2: one sequence of costs for positive examples, and one sequence for negative examples. In general, $\mathcal{C}$ can be of any size. The only requirement is that the maximum magnitude of the members of $\mathcal{C}$ be bounded. Figure 3.1(a) shows an example of a processing pipeline.

Denote the classifier at the j-th stage by f_j and the entire processing pipeline by f. A positive value for f_j indicates that processing should continue, while a negative value indicates processing should stop. Denote the 0-1 indicator function by $I[\cdot]$. The loss for an example is therefore

$$L(f(x), c) = \sum_{j=1}^{S} \left(c_j I\left[f_j(x_j) < 0\right] \prod_{k=1}^{j-1} I\left[f_k(x_k) \geq 0\right] \right) + c_{S+1} \prod_{k=1}^{S} I\left[f_k(x_k) \geq 0\right] \ . \quad (3.1)$$

The goal is to find S classifiers, one for each stage, which together minimize $\mathbf{E}[L(f(x), c)]$. Although we do not know the true distribution of (x, c), we can use the empirical loss as a surrogate. We let $\{(X_1, C_1), \ldots, (X_N, C_N)\}$ denote the training set and X_{ij} denote the features of X_i that are available at the j-th stage. Analogously, we let C_{ij} denote the cost associated with X_i at the j-th stage.

In [39], we propose a chained boosting algorithm to train a pipeline of ensemble classifiers. We construct an upper bound of the loss function 3.1 by replacing every 0-1 indicator function by an exponential function.

$$L(f(x), c) \leq \sum_{j=1}^{S} \left(c_j e^{-f_j(x_j)} \prod_{k=1}^{j-1} e^{f_k(x_k)} \right) + c_{S+1} \prod_{k=1}^{S} e^{f_k(x_k)} \ . \quad (3.2)$$

The training procedure is greedy. One weak classifier is added to a chosen stage at a time such that the addition would decrease the upper bound of the loss function by the largest amount. The upper bound 3.2 has the advantage of being convex. However, it may be too

loose to approximate the actual loss function 3.1 well. We demonstrate evidence to this effect in our experimental results.

3.4 Catenary Support Vector Machine

We choose to use linear threshold functions as stage classifiers and derive an optimization procedure based on an upper bound of the empirical loss.

3.4.1 Loss bound

We start by re-writing the loss function (3.1) in terms of incremental costs.

$$L(f(x), c) = m_1 + \sum_{j=1}^{S} \left(\prod_{k=1}^{j-1} I\left[f_k(x_k) \geq 0\right] \right) \left(\alpha_j I\left[f_j(x_j) \geq 0\right] + \beta_j I\left[f_j(x_j) < 0\right] \right) \tag{3.3}$$

where for $j = 1, \ldots, S$,

$$m_j = \begin{cases} \min\left(m_{j+1}, c_j\right) & \text{if } j < S, \\ \min\left(c_{j+1}, c_j\right) & \text{if } j = S; \end{cases}$$

$$\alpha_j = \begin{cases} m_{j+1} - m_j & \text{if } j < S, \\ c_{j+1} - m_j & \text{if } j = S; \end{cases}$$

$$\beta_j = c_j - m_j \quad .$$

In words, m_j is the minimal cost at stage j or later. α_j is the incremental increase in the minimal cost by continuing processing and β_j is the incremental cost of stopping processing. Note that either α_j or β_j is positive but not both. We denote the incremental costs associated with X_i at the j-th stage by m_{ij}, α_{ij}, and β_{ij}. Figure 3.1(b) shows a processing pipeline with the incremental costs.

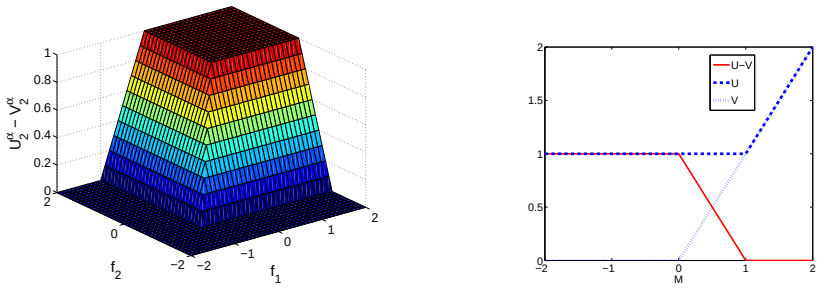
It is hard to minimize (3.3) directly. We define a upper bound for $L(f(x), c)$ and minimize the upper bound instead.

$$\hat{L}(f(x), c) = m_1 + \sum_{j=1}^{S} \left[\alpha_j \Big(U_j^{\alpha}(x) - V_j^{\alpha}(x) \Big) + \beta_j \Big(U_j^{\beta}(x) - V_j^{\beta}(x) \Big) \right] \tag{3.4}$$

where

$$\begin{aligned} U_j^*(x_j) &= \max\big(1, M_j^*(x)\big) \\ V_j^*(x_j) &= \max\big(0, M_j^*(x)\big) \\ M_j^{\alpha}(x) &= \max\left(-f_1(x_1), \ldots, -f_{j-1}(x_{j-1}), -f_j(x_j)\right) \\ M_j^{\beta}(x) &= \max\left(-f_1(x_1), \ldots, -f_{j-1}(x_{j-1}), f_j(x_j)\right) \quad . \end{aligned}$$

The wildcard '*' represents either α or β. The key idea of deriving Equation (3.4) is to use the difference of two $\max$ functions to upper bound the conjunction of indicator functions. Figure 3.2(a) shows an example when the conjunction consists of two indicator functions. Note that we simply bound the conjunction by a multivariate ramp function. A



(a) (b)

Figure 3.2: Multivariate ramp loss: (a) $I[f_1(x) \geq 0] \cdot I[f_2(x) \geq 0] \leq U_2^\alpha(x) - V_2^\alpha(x)$, (b) U_j^* and V_j^* as a function M_j^*

different multivariate ramp function has been used to approximate the disjunction in [31]. Figure 3.2(b) shows U_j^* and V_j^* as a function M_j^*. It illustrates an alternative to express the one-dimensional ramp function as the difference of two hinge functions. The way used in Figure 2.1 is not applicable here because the corresponding hinge functions are not convex in the arguments of M_j^*.

We formulate the following optimization problem.

$$\min \sum_{i=1}^{N} \hat{L}(f(X_i), C_i) + \lambda\Omega(\|f_1\|_{\mathcal{H}_1}, \ldots, \|f_S\|_{\mathcal{H}_S}) \tag{3.5}$$

where Ω is some monotonically increasing function. The first term measures the empirical loss and the second term is the regularization term, measured with respect to a set of reproducing kernel Hilbert spaces $\{\mathcal{H}_j\}$.

3.4.2 Catenary Support Vector Optimization

We begin with linear classifiers $f_j(x_j) = w_j \cdot x_j + b_j$ and an ℓ_2 regularization term $\Omega(\|f_1\|_{\mathcal{H}_1}, \ldots, \|f_S\|_{\mathcal{H}_S}) = \sum_{j=1}^{S} \|w_j\|_2^2$. Note that U_j^* and V_j^* are all convex functions in $\{(w_1, b_1), \ldots, (w_S, b_S)\}$. But the difference of two convex functions, $U_j^*(x_j) - V_j^*(x_j)$, is non-convex. Thus, Problem (3.5) is not a convex optimization problem.

We re-formulate Problem (3.5) as the following constrained optimization problem.

$$
\begin{aligned}
\min \quad & \textstyle\sum_{i=1}^{N} \sum_{j=1}^{S} \left(\alpha_{ij} \xi_{ij}^{\alpha} + \beta_{ij} \xi_{ij}^{\beta} \right) + \lambda \sum_{j=1}^{S} \|w_j\|_2^2 \\
\text{s.t.} \quad & \xi_{ij}^{\alpha} \geq -w_k \cdot X_{ij} - b_k - V_j^{\alpha}(X_i) \quad \forall i, k \leq j \\
& \xi_{ij}^{\beta} \geq -w_k \cdot X_{ij} - b_k - V_j^{\beta}(X_i) \quad \forall i, k < j \\
& \xi_{ij}^{\beta} \geq w_j \cdot X_{ij} + b_j - V_j^{\beta}(X_i) \quad \forall i, j \\
& \xi_{ij}^{\alpha}, \xi_{ij}^{\beta} \geq 1 \quad \forall i, j
\end{aligned}
\tag{3.6}
$$

Note that we have dropped the constant term, $\sum_{i=1}^{N} m_{i,1}$, from the objective. In principle, we can leave V_j^* in the objective. But moving V_j^* to the constraints appears to give better empirical results. Also, it is possible to have different tradeoff parameters for each classifier stage.

To solve Problem (3.6), we resort to the Concave-Convex procedure (CCCP) (see Chapter 2). Let $\mathbf{w} = (w_1, \ldots, w_S)$ and $\mathbf{b} = (b_1, \ldots, b_S)$. In each iteration, we need to replace V_j^* in the constraints by its first-order Taylor expansion at the current estimates of $\mathbf{w}$ and

$\mathbf{b}$. Notice that V_j^* are non-smooth functions. When we calculate its Taylor expansion, we use its subgradient. For the pointwise maximum function $h(x) = \max_{1 \le i \le m} h_i(x)$, its subdifferential at x, $\partial h(x)$, is the convex hull of the subdifferentials of the "active" functions at x, i.e., $\partial h(x) = \mathbf{H_{Convex}}\{\partial h_i(x) | h_i(x) = h(x)\}$. Thus, by simple calculus, we obtain that, for $j = 1, \ldots, S$,

$$\frac{\partial V_j^*(x; \mathbf{w}, \mathbf{b})}{\partial \mathbf{w}} = \begin{cases} \{\vec{0}\} & \text{if } M_j^*(x) < 0, \\ \Big\{(-\tau_1 x_1, \ldots, -\tau_{j-1} x_{j-1}, \sigma \tau_j x_j, \mathbf{0}) & \\ \qquad \Big| \tau_k \ge 0,\ \sum_{k=1}^{j} \tau_k \le 1 \Big\} & \text{if } M_j^*(x) = 0, \\ \Big\{(-\tau_1 x_1, \ldots, -\tau_{j-1} x_{j-1}, \sigma \tau_j x_j, \mathbf{0}) & \\ \qquad \Big| \tau_k \ge 0,\ \sum_{k=1}^{j} \tau_k = 1 \Big\} & \text{if } M_j^*(x) > 0; \end{cases} \tag{3.7}$$

where

$$\tau_k = 0 \quad \begin{array}{c} \text{if } k < j \text{ and } M_j^*(x) \neq -w_k \cdot x_k - b_k \\ \text{or if } k = j \text{ and } M_j^*(x) \neq \sigma(w_k \cdot x_k + b_k) \end{array}$$

$$\sigma = \begin{cases} -1 & \text{if *}= \alpha, \\ +1 & \text{if *}= \beta. \end{cases}$$

and $\mathbf{0}$ denotes padding zeroes of appropriate length.

Similarly, we can obtain $\frac{\partial V_j*(x;\mathbf{w},\mathbf{b})}{\partial \mathbf{b}}$ by replacing x_k's by 1's in Equation (3.7). In the

experiments, we pick the subgradient where

$$\tau_k = \begin{cases} c & \text{if } k \text{ is the largest index s.t.} \\ & \quad \text{either } k < j \text{ and } M_j^*(x) = -w_k \cdot x_k - b_k, \\ & \quad \text{or } k = j \text{ and } M_j^*(x) = \sigma(w_k \cdot x_k + b_k), \\ 0 & \text{otherwise,} \end{cases}$$

where $c = \frac{\rho}{\rho+1}$ if $M_j^*(x) = 0$ or $c = 1$ if $M_j^*(x) > 0$, and ρ is the number of active functions.

Since only one of α_{ij} or β_{ij} is nonzero, we need only consider the constraints associated with one of ξ_{ij}^{α} or ξ_{ij}^{β}. The number of constraints in Program (3.6) is quadratic in the number of stages. We can re-write it so that the number of constraints depends linearly on the number of stages.

$$\begin{aligned} \min \quad & \textstyle\sum_{i=1}^{N} \sum_{j=1}^{S} \left(\alpha_{ij} \xi_{ij}^{\alpha} + \beta_{ij} \xi_{ij}^{\beta} \right) + \lambda \sum_{j=1}^{S} \|w_j\|_2^2 \\ \text{s.t.} \quad & \xi_{ij}^{\alpha} \geq \eta_{ij} - V_j^{\alpha}(X_i) && \forall i, j \\ & \xi_{ij}^{\beta} \geq \eta_{i,j-1} - V_j^{\beta}(X_i) && \forall i, j \\ & \xi_{ij}^{\beta} \geq w_j \cdot X_{ij} + b_j - V_j^{\beta}(X_i) && \forall i, j \\ & \xi_{ij}^{\alpha}, \xi_{ij}^{\beta} \geq 1 && \forall i, j \\ & \eta_{ij} \geq -w_j \cdot X_{ij} - b_j && \forall i, j \\ & \eta_{ij} \geq \eta_{i,j-1} && \forall i, j \end{aligned} \tag{3.8}$$

3.4.3 Extensions

One important advantage of SVM is that it can handle non-linearly separable data with an appropriate kernel function. Likewise, we can also kernelize catenary support vector machine by the Representer theorem [38]. Denote the kernel matrix for the j-th stage by K_j and its i-th column by $K_j(\cdot, i)$. The only changes to Program (3.8) are (i) replacing the regularization term in the objective by $\lambda \sum_j^S w_j' K_j w_j$, and (ii) replacing the feature vector X_{ij} by $K_j(\cdot, i)$. We are free to choose different kernels for different stages. Furthermore, instead of using a ℓ_2 regularization term, we may use ℓ_1 regularization term to promote sparsity. In doing so, we need to solve a linear program instead of a quadratic program in every iteration of the CCCP.

3.4.4 An alternative loss bound

In the above derivation, we view the loss of an example as the sum of losses incurred in each stage and then derive a upper bound using ramp functions. This is not the only way to do it. Alternatively, we can view the loss of an example as the $\max$ of losses incurred in each stage and obtain the following loss bound:

$$\tilde{L}(f(x), c) = d_0 + \max\left(\left\{d_j(U_j^\alpha(x) - V_j^\alpha(x))\right\}_{j=1}^S, d_{S+1}(U_{S+1}^\beta(x) - V_{S+1}^\beta(x))\right) \quad (3.9)$$

where $d_0 = \min(c_1, \ldots, c_{S+1})$ and $d_j = c_j - d_0$. Note that $\tilde{L}$ is no greater than $\hat{L}$. It is not difficult to see that we can formulate a constrained optimization problem with $\hat{L}$ and employ the CCCP to solve it. Unfortunately, our preliminary experimental results showed that the CCCP is not effective in optimizing $\tilde{L}$. We leave the problem of optimizing $\tilde{L}$ as an open problem.

3.5 Performance bounds

We provide theoretical bounds on how well the catenary support vector machine will generalize to future test examples. We need the following definitions.

Definition 3.5.1. *Let μ be a probability distribution on a set $\mathcal{X}$ and suppose that $X_1, \ldots, X_n$ are independent samples selected according to μ. Let F be a class of functions mapping from $\mathcal{X}$ to $\Re$. Define the random variable*

$$\hat{R}_n(F) = \mathbf{E}\left[\sup_{f \in F} \left| \frac{2}{n} \sum_{1}^{n} \sigma_i f(X_i) \right| \;\middle|\; X_1, \ldots, X_n \right] ,$$

where $\sigma_1, \ldots, \sigma_n$ are independent uniform $\{\pm 1\}$-valued random variables. The Rademacher complexity *of F is $R_n(F) = \mathbf{E}\hat{R}_n(F)$. Similarly, define the random variable*

$$\hat{G}_n(F) = \mathbf{E}\left[\sup_{f \in F} \left| \frac{2}{n} \sum_{1}^{n} g_i f(X_i) \right| \;\middle|\; X_1, \ldots, X_n \right] ,$$

where $g_1, \ldots, g_n$ are independent Gaussian random variables with zero mean and unit

variance. The Gaussian complexity *of* F *is* $G_n(F) = \mathbf{E}\hat{G}_n(F)$.

The Rademacher and Gaussian complexity measure the capability of F in fitting random values. The higher the complexity, the more prone to overfitting F is. We can now state the theorem, bounding the true risk by the empirical risk and the Gaussian complexity of the classes of the stage classifiers:

Theorem 3.5.1. *Let* L *and* $\hat{L}$ *be as in Equations (3.1) and (3.4). Let* $\gamma_j = \max_{c \in \mathcal{C}}(\alpha_j + \beta_j)$ *and* $\Lambda = \max_{f(x),c} \hat{L}(f(x), c)$. *Let* $F_1, \ldots, F_S$ *be the sequence of the classes of the stage classifiers. Let* $(X_i, C_i)_{i=1}^N$ *be independently selected according to some fixed probability measure* P. *Then, for any integer* N *and any* $0 < \delta < 1$, *with probability at least* $1 - \delta$ *over samples of size* N, *every sequence* $f_1, \ldots, f_S$ *in* $F_1 \times \ldots \times F_S$ *satisfies*

$$\mathbf{E}[L] \leq \hat{\mathbf{E}}_N[\hat{L}] + \kappa \sum_{j=1}^{S} \left(\sum_{\ell=j}^{S} \gamma_\ell \right) G_N(F_j) + \Lambda \sqrt{\frac{8 \ln \frac{2}{\delta}}{N}}$$

for some constant κ.

Proof. Theorem 8 of [3] implies

$$E[L(f(X), C)] \leq \hat{E}_N[\hat{L}(f(X), C)] + 2R_N(\hat{L} \circ F) + + \Lambda \sqrt{\frac{8 \ln(2/\delta)}{N}}$$

and therefore we need only bound the $R_N(\hat{L} \circ F)$ term to demonstrate our theorem. For

our case, we have

$$\begin{aligned} R_N(\hat{L} \circ F) &= E \sup_{f \in F} \frac{1}{N} \Big| \sum_{i=1}^{N} \sigma_i \hat{L}(f(X_i), C_i) \Big| \\ &= E \sup_{f \in F} \frac{1}{N} \Big| \sum_{i=1}^{N} \sum_{j=1}^{S} \gamma_{ij} \left(U_j^{\gamma}(X_i) - V_j^{\gamma}(X_i) \right) \Big| \\ &\leq \sum_{j=1}^{S} E \sup_{f \in F} \frac{1}{N} \Big| \sum_{i=1}^{N} \sigma_i \gamma_{ij} \left(U_j^{\gamma}(X_i) - V_j^{\gamma}(X_i) \right) \Big| \\ &= \sum_{j=1}^{S} \gamma_j R_N(U_j^{\gamma} - V_j^{\gamma}) \end{aligned}$$

The inequality comes from the fact that $|a + b| \leq |a| + |b|$ and the convex property of the $\sup$ function.

Lemma 4 of [3] states that there exists a κ such that $R_N \leq \kappa G_N$. Theorem 14 of the same paper allows us to conclude that $G_N(U_j^{\gamma} - V_j^{\gamma}) \leq 2 \sum_{\ell=1}^{j} G_N(F_\ell)$. Taken together, this proves our result. □

Note that the second term in the bound does not depend on the regular costs c_j's directly, and it does not depend on m_1 at all. Quite often, the incremental costs α_j's and β_j's are smaller than the c_j's. Additionally, for $j < j'$, the complexity of F_j has a larger weight than that of $F_{j'}$. This may suggest that it is advantageous to use simple stage classifiers in early stages and use complex stage classifiers in later stages. We can further bound the true risk in terms of kernel functions of the stage classifiers. We need the following lemma which follows from McDiarmid's inequality [34].

Lemma 3.5.1. *Let F be a class of functions mapping to $[-1, 1]$. For any integer n,*

$$P\left\{|G_n(F) - \hat{G}_n(F)| \geq \epsilon\right\} \leq 2\exp\left(\frac{-n\pi\epsilon^2}{4}\right) .$$

Theorem 3.5.1 and Lemma 3.5.1, combined with Lemma 22 in Bartlett and Mendelson [3], imply the following theorem.

Theorem 3.5.2. *Let L and $\hat{L}$ as in Equations (3.1) and (3.4). Let $\gamma_j = \max_{c\in\mathcal{C}}(\alpha_j + \beta_j)$ and $\Lambda = \max_{f(x),c} \hat{L}(f(x), c)$. Let $F_1, \ldots, F_S$ be the sequence of the classes of stage classifiers. Let $\mathcal{X}_j$ be the feature space in the j-th stage. For $j = 1, \ldots, S$, fix B_j, and let $K_j : \mathcal{X}_j \times \mathcal{X}_j \to \Re$ be a kernel with $\sup_{x\in\mathcal{X}_j} |K_j(x, x)| < \infty$. Let $\mathcal{X}$ be the full feature space, i.e., $\mathcal{X} = \bigcup_{j=1}^{S} \mathcal{X}_j$. Suppose that $\{X_i, C_i\}_{i=1}^{N}$ are selected at random and independently according to some probability distribution P on $\mathcal{X}\times\mathcal{C}$. Then with probability at least $1 - \delta$, every function sequence $f_1, \ldots, f_S$ of the form*

$$f_j(x) = \sum_{i=1}^{N} \alpha_i K_j(x_{ij}, x)$$

with $\sum_{i_1,i_2} \alpha_{i_1}\alpha_{i_2}K_j(x_{i_1,j}, x_{i_2,j}) \leq B_j^2$ *satisfies*

$$\mathbf{E}[L] \leq \hat{\mathbf{E}}_N[\hat{L}] + \frac{\kappa}{N}\sum_{j=1}^{S}\left(\sum_{\ell=j}^{S}\gamma_\ell\right) B_j \sqrt{\sum_{i=1}^{N} K_j(x_{ij}, x_{ij})} + \left(\Lambda + \frac{1}{\sqrt{2\pi}}\sum_{j=1}^{S} j\gamma_j\right)\sqrt{\frac{8\ln\frac{2(S+1)}{\delta}}{N}}$$

for some constant κ*.*

3.6 Experimental Results

We tested catSVM on UCI benchmark data and the MIT face database. We compared the performance of catSVM to chained boosting[1] and single-stage SVM. For chained boosting, we used decision stumps as weak classifiers and set the number of rounds before the algorithm stops to 2000. For single-stage SVM and catSVM, we used the RBF kernel and set the kernel width to the median of the pairwise distance in the training set. We set the regularization parameter λ to be 1 and did not adjust it. Furthermore, for catSVM, we initialized the SVM in every stage to be zero (i.e., for $j = 1, \ldots, S,\ w_j = \vec{0}$ and $b_j = 0$). We used Mosek (www.mosek.com) to solve the quadratic programs generated by catSVM.

The single-stage SVM and our catSVM algorithms run on the same hypothesis space of RBF kernels. We ran chained boosting on a different feature space. It was not possible to use the same feature space. The boosting algorithm constructs a linear surface in the space

[1]The original chained boosting algorithm uses regular costs. We modified it to use incremental costs as well, which improved its performance.

of features that are decision stumps. That does not correspond to any easily constructed kernel. The feature space dimensions of the RBF kernel could be defined as the set of all kernel functions with one point as a training point. However, those dimensions have real values and the boosting algorithm is designed for thresholded features (or weak learners). However, our experience with these datasets suggests that boosting decision stumps and RBF kernels for SVMs have roughly the same performance on single-stage problems.

We did not compare to constructing three independent SVM classifiers and then connecting them in a chain; this would have required selecting the false-positive and false-negative costs for each classifier. One of the main purposes of our approach is to automatically adjust the classifier to achieve the desired cost results without having to manually search over such trade-off parameters.

We construct the vectors of stage costs as follows. We assign a constant feature cost to each stage that is the same for all examples (as specified in the problem set up below). It represents the cost of collecting the features, regardless of the final outcome. If the example is positive, we add an extra cost to c_i for $i \leq s$, representing an extra penalty if a positive example is rejected at any stage. If the example is negative, we add an extra cost to c_{s+1}, that is we penalize the classifier if it allows the example to pass through every stage (and therefore wrongly accepts it as a positive example).

			fn: 9, fp: 18				fn: 18, fp: 18				fn: 36, fp: 18				fn: 72, fp: 18			
boosting	train	+	76	0	0	0	70	0	0	6	38	0	0	38	16	0	0	60
		–	94	0	0	0	94	0	0	0	82	8	4	0	74	4	16	0
	test	+	43	0	0	1	39	0	0	5	24	2	0	18	18	2	1	23
		–	56	0	0	0	52	1	2	1	45	5	4	2	42	7	4	3
catSVM	train	+	76	0	0	0	6	2	6	62	2	1	1	72	0	0	2	74
		–	94	0	0	0	68	3	19	4	54	8	20	12	25	20	34	15
	test	+	44	0	0	0	9	0	5	30	7	1	3	33	7	0	4	33
		–	56	0	0	0	49	1	5	6	35	5	9	7	20	9	19	8

Table 3.1: For four different cost settings for the heart dataset, the distributions of the stages at which the examples were rejected (or accepted for the final column) for boosting and catSVM, for training and testing, and for positive and negative examples.

3.6.1 UCI Data

We report the results on the heart disease dataset from the UCI machine learning repository. We assigned the 13 attributes to three stages in descending order according to their correlation with the predicted output: four attributes to each of the first two stages and five to the last stage. The single-stage SVM was trained and tested on all the 13 attributes. We set the feature cost of each stage to the number of attributes assigned to that stage and all the preceding stages. Early rejections are treated as "normal" whereas an example that passes all stages is treated as "disease." Figure 3.3(a) shows the false negatives and false positives as the misclassification penalties vary. The three methods give very similar tradeoff between the two types of errors. Figure 3.3(b) shows the false negative and the average feature cost as the misclassification penalties vary. The average feature cost is the average number of features that must be examined before the classifier makes a decision. The feature cost of single-stage SVM is fixed at 13. We observe that catSVM does a better job in trading false negative rate for feature cost than chained boosting.

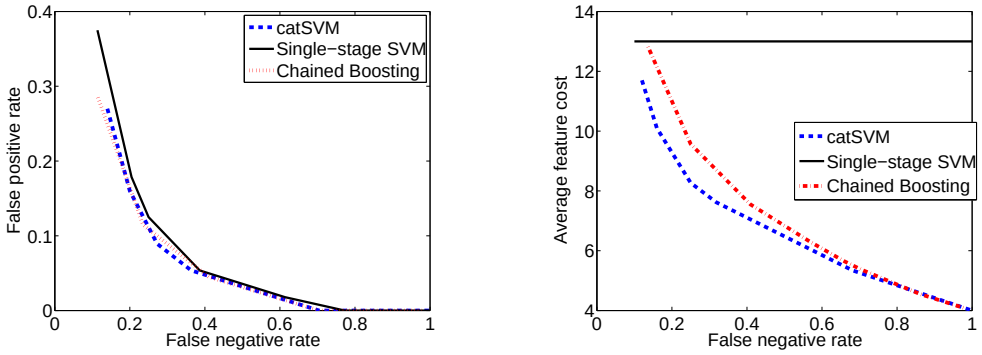
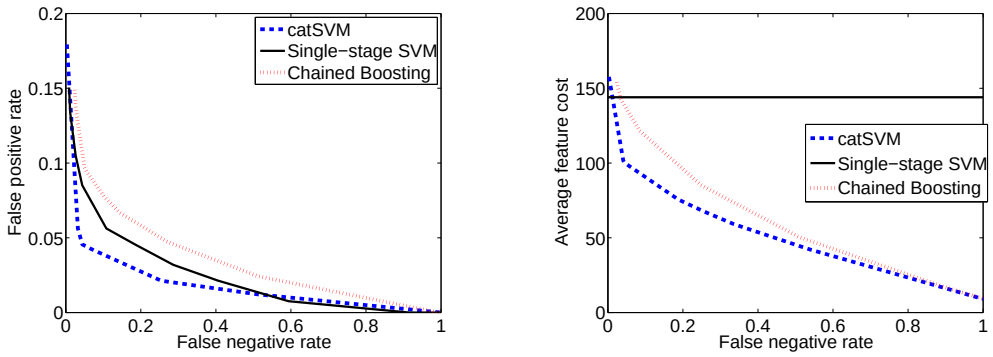


Figure 3.3: UCI heart tradeoffs: (a) False negative vs. False positive, (b) False negative vs. Average feature cost

			fn: 150, fp: 250				fn: 250, fp: 250				fn: 500, fp: 250				fn: 1000, fp: 250			
boosting	train	+	224	0	0	0	224	0	0	0	220	0	0	4	33	0	0	191
		–	376	0	0	0	376	0	0	0	375	1	0	0	342	34	0	0
	test	+	341	0	0	1	341	0	0	1	342	4	0	36	143	32	1	166
		–	654	1	3	0	653	1	3	1	646	4	4	4	533	88	19	18
catSVM	train	+	224	0	0	0	35	8	0	181	0	0	1	223	0	0	0	224
		–	376	0	0	0	278	86	10	2	104	233	36	3	96	240	37	3
	test	+	342	0	0	0	42	16	6	278	1	10	7	324	1	10	7	324
		–	658	0	0	0	457	143	28	30	170	392	63	33	152	410	62	34

Table 3.2: For four different cost settings for the face detection dataset, the distributions of the stages at which the examples were rejected (or accepted for the final column) for boosting and catSVM, for training and testing, and for positive and negative examples.

Table 3.1 shows at which stage the examples are rejected or accepted. The feature costs are 4, 8, 13, and 13. 'fn: 9, fp: 18' means the penalties for false negative and false positive are 9 and 18, respectively; therefore the cost vectors would be $[13, 17, 22, 13]$ and $[4, 8, 13, 31]$ for positive and negative examples, respectively. Note that for every penalty setting, the first three columns are the number of examples rejected in the three stages and the last column is the number of examples accepted. As the penalty of false negative increases, both chained boosting and catSVM try to accept more and more positive examples.



(a) (b)

Figure 3.4: Face detection tradeoffs: (a) False negative vs. False positive, (b) False negative vs. Average feature cost

3.6.2 Face Detection

We also validated the catenary SVM by applying it to face detection. We tested on the MIT face database [35] which contains 19-by-19 gray-scale images of faces and non-faces. For face detection, the non-face is usually the majority. Therefore, our goal is to produce a classifier that can identify non-face images by examining as low a resolution patch as possible. We built a multi-stage detection system where any early rejection is labeled as a non-face. The first stage looks at down-sampled versions of the images at a resolution of 3-by-3. The next stages do the same, at resolutions of 6-by-6 and 12-by-12. We did not examine the full 19-by-19 resolution as it did not provide significant improvement over the 12-by-12 resolution.

We assign a feature cost to each stage proportional to the total number of pixels at that stage and all the preceding stages. There are three free parameters in the problem formulation: the per pixel cost, the penalty for an incorrect face classification, and the

penalty for an incorrect non-face classification. Changing these quantities will control the tradeoff between false negatives and false positives, and between classification error and feature cost. In the experiments, we fix the per pixel cost and vary the other two quantities.

We used 600 images as the training set and 1000 images as the testing set. The single-stage SVM was trained and tested on image patches at the highest resolution, 12-by-12. Figure 3.4(a) shows the false negatives and false positives as the misclassification penalties vary. Note that catSVM can achieve a better tradeoff than single-stage SVM and chained boosting. The processing pipeline successfully improves the ability of SVM to tradeoff between the two types of errors. Figure 3.4(b) shows the false negative and the average feature cost as the misclassification penalties vary. The feature cost of single-stage SVM is fixed at 144. Chained boosting and catSVM give higher average feature costs for lower false negative rates. Note that catSVM requires a lower average feature cost than chained boosting for most false negative rates. The advantage of catSVM becomes more obvious when the false negative rate is small.

Table 3.2 shows at which stage the examples are rejected or accepted. The feature costs are 9, 45, 189, and 189. 'fn: 150, fp: 250' means the penalties for false negative and false positive are 150 and 250, respectively. As the penalty of false negative increases, both chained boosting and catSVM try to accept more and more positive examples. It is clear that catSVM is more effective in pushing the positive examples forward.

It is interesting to note that the performance of catSVM is superior to that of a single-

stage SVM (which is a regular SVM trained on the full set of features, varying the false-positive and false-negative costs) in terms of testing error. We believe this is because the hypothesis class of the earlier stages are simpler. Therefore, those decision rules have less variance for a fixed number of samples. Our algorithm then has a natural bias that helps reduce the variance with few numbers of samples. Our generalization bounds also point to this advantage.

3.7 Conclusion

We believe that for some decision-making problems, it is important to weigh the benefit against the cost of acquiring more information. We present the catenary SVM to solve one-sided early detection for binary classification. We formulate the problem as a constrained concave-convex optimization problem and solve it using CCCP. In addition, we provide data-dependent theoretical guarantee for catSVM. The experimental results show that catSVM can tradeoff misclassification error and feature cost more effectively than single-stage SVM and chained boosting.

Chapter 4

Group Statistics Support Vector Machine

Recently, Kück and de Freitas [29] formulated a new classification problem based on group statistics. In this formulation, we are given groups of instances along with the fraction of positive instances per group. The task is to learn a classifier to classify individual instances. In this chapter, we give a generalization error bound for groups statistics classification and derive a novel SVM-based algorithm that seeks to minimize the bound. Furthermore, we show how to extend our algorithm to general multiclass setting. The experimental results show that our method is competitive with Kück's and de Freitas's Bayesian sampling approach. Our method has the advantage of having fewer parameters to tune.

4.1 Introduction

In group statistics classification, we are given a number of groups of examples as the training set. We know the attributes of every training example but not their labels. The additional information we have is the class distribution of individual examples in each group. The goal is to learn a classifier for any individual example from the group statistics.

Group statistics classification can be very useful in marketing research. For example, a company wants to increase its profit of sales by sending out discount coupons. Ideally, the company wants to send coupons exclusively to people who would only make a purchase if they receive a coupon. The company does not know the identity of those potential customers. However, the company may estimate the percentage of potential customers in different regions reliably by comparing the sales number with and without a promotional campaign. Together with some existing information about individual people allows the company to study the relationship between individual purchase decision and individual properties.

Other possible applications of group statistics classification include email and image classification tasks. In email classification, it may be tedious for each user to label every single email in his/her inbox but a user may be able tell how many emails of various kinds in one's inbox. In image classification, we may be interested in identifying interesting segments in images. It is expensive to label each image segment but it may be possible to

estimate the percentage of interesting segments in each image from its caption and annotation keywords. In both cases, we may take advantage of some aggregate information to learn a classifier for individual examples.

Because of the ambiguity in the information, group statistics classification is a challenging task. We do not know which instances in a group are positive and pure (positive or negative) groups are rare. The lack of pure negative groups implies that multiple-instance learning algorithms (see the next section) are not suitable for this task. We provide a theoretical analysis for group statistics classification. From that, we derive a SVM-based algorithm that seeks to minimize a generalization error bound.

4.2 Related work

Group statistics classification is related to multiple-instance (MI) learning which was first introduced by Dietterich et al. [16] for drug activity prediction. In MI learning, the training set is presented as bags of instances. Each bag is associated with a (positive or negative) label. A positive bag label means that there is at least one positive instance in the bag. A negative bag label means that all the instances in the bag are negative. MI learning has attracted a lot of attention in the past ten years. A number of MI learning algorithms have been proposed including Diverse Density (DD) [32], EM-DD [60], Citation-kNN [54], MI kernels [22], multiple-instance SVMs [1], Bayesian model [28], MILBoost [53], and convex-hull MIL Fisher's discriminant [21].

Several researchers have considered generalizations of the standard MI learning formulation. Weidmann et al. [55] and Tao et al. [47] generalize the process that combines labels of instances to form a bag label. They assume that the label of a bag is decided by some more general threshold function rather than a disjunction. A positive bag label indicates that the number of positive instances in the bag lies within a certain range.

Group statistics classification was formulated recently by Kück and de Freitas [29]. It can be seen as a more informative variant of multiple-instance learning. Also, explicitly specifying the fraction of positive instances in each group is similar in spirit to using more general threshold functions to restrict the number of positive instances in a positive bag [55, 47]. However, the main difference is that in the group statistics, we are not limited to a single threshold for labeling every bag. This difference gives us the flexibility to handle more general applications.

Kück and de Freitas [29] propose a Bayesian sampling approach based on Markov chain Monte Carlo (MCMC). However, MCMC algorithms are tricky to implement, especially with respect to convergence criteria. It is hard to assess convergence of the Markov chains. It may take many state transitions before a Markov chain converges to its stationary distribution. And it is hard to determine how many samples are sufficient to capture the target distribution. Thus there seems to be a genuine need to consider alternatives to MCMC algorithms. Most recently, Qua et al. [36] proposed to use a conditional exponential model based on mean operator estimation. Their method seems not suitable for settings when the

number of groups is larger than the number of classes.

4.3 Group statistics classification

In this section, we give the problem definition of group statistics classification and provide a theoretical analysis of group statistics classification in the binary setting.

4.3.1 Problem Definition

Let m be the number of groups and n_i be the number of instances in the i-th group. Let p_i ($\bar{p}_i$) be the number of positive (negative) instances in the i-th group. Denote the i-th group of instances by X_i and the j-th instance in the i-th group by X_{ij}. Given a training set $\{(X_i, p_i)\}_{i=1}^m$, the task is to learn to classify individual instances from this information.

4.3.2 Theoretical Analysis

We prove a generalization bound for group statistics classification. We make two simplifying assumptions in our analysis. First, we assume that each instance in the training set can be regarded as an i.i.d. sample from the test distribution. This implies the test distribution is the same as the training distribution at the instance level. We can interpret this assumption as follows: we generate the training set by drawing i.i.d. samples from the test distribution and then dividing the samples into groups arbitrarily. Second, we assume that there is no noise in the fraction of positive instances associated with each group in the training set. Wlog, we also assume that the numbers of instances in every group are the same, *i.e.*, n_i

for $i = 1, \ldots, m$. We now state our theorem.

Theorem 4.3.1. *Let F be a class of $\{\pm 1\}$-valued functions defined on a set $\mathcal{X}$ and P be a probability distribution on $\mathcal{X} \times \{\pm 1\}$. Suppose that (x, y) is chosen according to P. Suppose that $\{(X_i, p_i)\}_{i=1}^{N}$ are chosen such that $\{(X_{ij}, Y_{ij})\}$ are independent samples from P and $p_i's$ are correct. Then, there is an absolute constant c such that for any integers m and n, with probability at least $1 - \delta$ over samples of size $m \times n$, every f in F satisfies*

$$P(y \neq f(x)) \leq \frac{1}{mn} \sum_{i=1}^{m} \left| \sum_{j=1}^{n} I[f(x_{ij}) \geq 0] - p_i \right| + \frac{2}{mn} \sum_{i=1}^{m} \min(p_i, \bar{p}_i) + c\sqrt{\frac{VCdim(F)}{mn}},$$

where VCdim(F) denotes the Vapnik-Chervonenkis dimension of F and $\bar{p}_i = n - p_i$.

Proof. Since (X_{ij}, Y_{ij}) are i.i.d. samples from P, by Theorem 1 in Bartlett and Mendelson [3], we have with probability at least $1 - \delta$,

$$P(y \neq f(x)) \leq \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} I[Y_{ij} \neq f(X_{ij}] + c\sqrt{\frac{VCdim(F)}{mn}}.$$

Notice that

$$\sum_{j=1}^{n} I[Y_{ij} \neq f(X_{ij})] \leq \left| \sum_{j=1}^{n} I[f(x_{ij}) \geq 0] - p_i \right| + 2\min(p_i, \bar{p}_i) \ .$$

Taken together, this proves our result. □

Theorem 4.3.1 says that the test error rate is no more than the average of the absolute errors in the predicted fraction of positive instances in each group in the training set, plus twice the average of the fractions of the minority class in each group, and a complexity term. The generalization bound is very intuitive. Consider two extreme cases. If the fractions of positive instances are either 0 or 1, the second term in the bound disappears and we get back the generalization bound for standard binary classification. If the fractions of positive instances are all 0.5, the second term in the bound becomes 1. This means even if we can achieve zero empirical loss, the classifier we learn can still perform arbitrarily badly on future test examples.

4.4 Our Method

The above theoretical analysis suggests that a small absolute prediction error implies a small error rate for future test examples. As a result, we propose to find a classifier f such that for any group i, the number of instances it classifies as positive is close to p_i in the ℓ_1 sense. In particular, we are interested in the class of linear classifiers, *i.e.*, $f(x) = w \cdot \phi(x) + b$ where $\phi(x)$ is the feature vector of instance x. The total empirical loss is defined as

$$L(f, \{(X_i, p_i)\}_{i=1}^m) = \sum_{i=1}^{m} \left| \sum_{j=1}^{n_i} I\left[f(X_{ij}) \geq 0\right] - p_i \right|$$

where $I\left[\cdot\right]$ is the 0-1 indicator function.

However, it is difficult to minimize the empirical loss L directly. We get around this

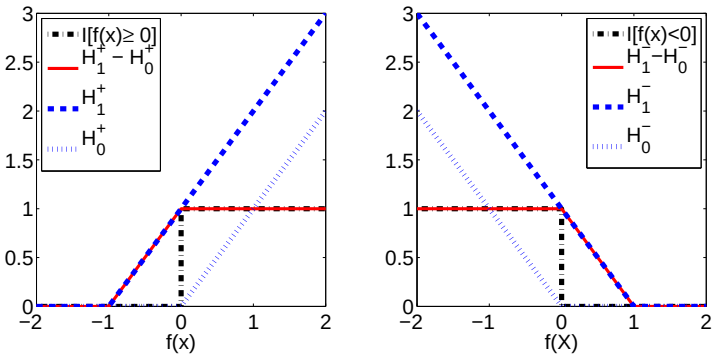


Figure 4.1: Bounding the indicator functions by ramp functions

computational difficulty by minimizing a upper bound of L. Consider the loss incurred by the i-th group.

$$\begin{aligned}
\left|\sum_{j=1}^{n_i} I[f(X_{ij}) \geq 0] - p_i\right| &= \max\left(\sum_{j=1}^{n_i} I[f(X_{ij}) \geq 0] - p_i, p_i - \sum_{j=1}^{n_i} I[f(X_{ij}) \geq 0]\right) \\
&= \max\left(\sum_{j=1}^{n_i} I[f(X_{ij}) \geq 0] - p_i, \sum_{j=1}^{n_i} I[f(X_{ij}) < 0] - \bar{p}_i\right) \\
&\leq \max\left(\sum_{j=1}^{n_i} \left(H_1^+(f(X_{ij})) - H_0^+(f(X_{ij}))\right) - p_i, \right. \\
&\qquad \left. \sum_{j=1}^{n_i} \left(H_1^-(f(X_{ij})) - H_0^-(f(X_{ij}))\right) - \bar{p}_i\right)
\end{aligned}$$

where $H_s^+(v) = \max(0, s+v)$ and $H_s^-(v) = \max(0, s-v)$.

In the last step, we upper bound every indicator function by a ramp function which can be expressed as the difference of two hinge functions (see Figure 4.1). We upper bound the loss of each group to obtain a upper bound of the total empirical loss, denoted by $\hat{L}$. We

formulate the following optimization program.

$$\min_f \quad \lambda \hat{L}(f, \{(X_i, p_i)\}_{i=1}^m) + \tfrac{1}{2}\|f\|_{\mathcal{H}}^2 \tag{4.1}$$

We introduce a regularization term $\frac{1}{2}\|f\|_{\mathcal{H}}^2$ in the objective function where $\|\cdot\|_{\mathcal{H}}$ is the reproducing kernel Hilbert space norm with kernel $K(x_1, x_2) = \phi(x_1) \cdot \phi(x_2)$. λ is a tradeoff parameter between the empirical loss and the regularization term. Note that alternatively, we can use an ℓ_1 regularizer to promote sparsity of the solution. We re-write Program (4.1) as a constrained optimization program.

$$\begin{aligned}
\min_{w,b,\xi,\eta,\tau} \quad & \lambda \textstyle\sum_{i=1}^m \xi_i + \frac{1}{2}\|f\|_{\mathcal{H}}^2 \\
\text{s.t.} \quad & \xi_i \geq \textstyle\sum_{j=1}^{n_i} \left(\eta_{ij} - H_0^+(f(X_{ij}))\right) - p_i \\
& \xi_i \geq \textstyle\sum_{j=1}^{n_i} \left(\tau_{ij} - H_0^-(f(X_{ij}))\right) - n_i \\
& \eta_{ij} \geq 1 + f(X_{ij}) \\
& \tau_{ij} \geq 1 - f(X_{ij}) \\
& \eta_{ij}, \tau_{ij} \geq 0
\end{aligned} \tag{4.2}$$

Note that the first two constraints in Program (4.2) are nonconvex. H_0^+ and H_0^- are convex because the maximum of convex functions is convex. However, the difference of two convex functions is nonconvex. Therefore, Program (4.2) is not a convex optimization problem. We employ the Concave-Convex procedure (CCCP) to solve for (w, b) itera-

tively. In every iteration, we replace H_0^+ and H_0^- by their first-order Taylor approximations at the current (w, b). The resulting program is a quadratic program and can be solved by any efficient quadratic program solver. Notice that H_0^+ and H_0^- are non-smooth functions. We need to use their subgradients to compute their first-order Taylor approximations. For the pointwise maximum function $h(x) = \max_{1 \leq i \leq m} h_i(x)$, its subdifferential at x, $\partial h(x)$, is the convex hull of the subdifferentials of the "active" functions at x, *i.e.*, $\partial h(x) = \mathbf{H_{Convex}}\{\partial h_i(x) | h_i(x) = h(x)\}$. By simple calculus, we obtain the subgradients of $H_0^+(X_{ij})$ and $H_0^-(X_{ij})$ w.r.t. (w, b):

$$\partial H_0^+(f(X_{ij})) = \begin{cases} \left\{\vec{0}\right\} & \text{if } f(X_{ij}) < 0, \\ \left\{\rho \cdot (\phi(X_{ij}), 1) \middle| 0 \leq \rho \leq 1\right\} & \text{if } f(X_{ij}) = 0, \\ \{(\phi(X_{ij}), 1)\} & \text{if } f(X_{ij}) > 0; \end{cases}$$

$$\partial H_0^-(f(X_{ij})) = \begin{cases} \left\{\vec{0}\right\} & \text{if } f(X_{ij}) > 0, \\ \left\{\rho \cdot (\phi(X_{ij}), 1) \middle| 0 \leq \rho \leq 1\right\} & \text{if } f(X_{ij}) = 0, \\ \{(\phi(X_{ij}), 1)\} & \text{if } f(X_{ij}) < 0\,. \end{cases}$$

We pick ρ to be 0.5 in our experiments.

For a linear kernel, we can solve Program (4.2) for (w, b) directly. However, we cannot do so for more powerful kernels because the corresponding feature vector space can be infinite dimensional. Fortunately, by the Representer theorem [38], we know that the optimal

solution of Program (4.2) has the form

$$f(x) = \sum_{i=1}^{m} \sum_{j=1}^{n_i} \alpha_{ij} K(x, X_{ij}) \ . \tag{4.3}$$

We can substitute Equation (4.3) into Program (4.2) and solve for (α, b) instead. Denote the kernel matrix by K. The only changes to Program (4.2) are (i) changing the regularization term in the objective to $\frac{1}{2}\alpha^T K \alpha$ and (ii) replacing the feature vector X_{ij} by $K(\cdot, X_{ij})$. In this way, we reduce the search space of f from an infinite-dimensional space of functions to space of finite dimension $\sum_{i=1}^{m} n_i + 1$.

4.5 Extension to multiclass setting

In many applications, we are interested in classifying individuals into more than two classes. For example, we might want to know which presidential candidate people vote for or through which mode of advertisement people get to know a commercial product. We show how our algorithm can be extended naturally to handle more than two classes.

Let m be the number of groups and n_i be the number of instances in the i-th group. Denote the i-th group of instances by X_i and the j-th instance in the i-th group by X_{ij}. Further, let p_i be the class distribution of the instances in the i-th group and p_{iy} be the number of instances in the i-group that belong to class y. We are given a training set $\{(X_i, p_i)\}_{i=1}^{m}$. The task is to learn a function f that maps instances $x \in \mathcal{X}$ to discrete class labels $y \in \mathcal{Y}$ from the given information.

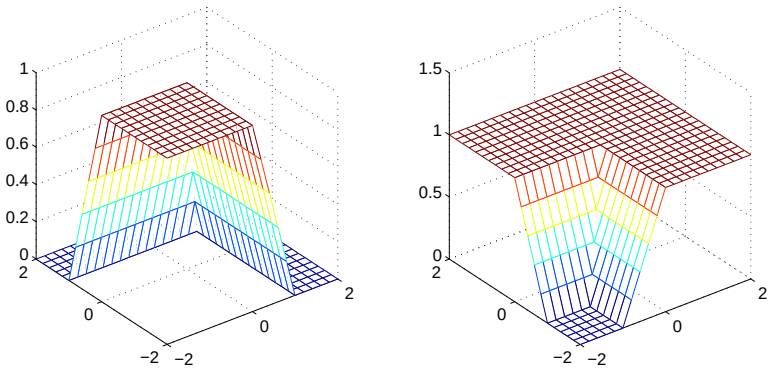


Figure 4.2: Two-dimensional ramp functions

We seek to compute a classifier f such that for any group i, the predicted class distribution is close to p_i. More specifically, we consider classifiers that take the form

$$f(x) = \arg\max_{y \in \mathcal{Y}} w \cdot \Phi(x, y).$$

We assume that $f(x, y) = w \cdot \Phi(x, y)$ measures the correctness of the association between instance x and class label y and the feature mapping function Φ maps (x, y) jointly into a suitable feature space endowed with dot product. This formulation of multiclass SVMs is very common (see e.g., [14, 8]). Thus, the total empirical loss is defined as

$$L\left(f, \{(X_i, p_i\}_{i=1}^m\right) = \sum_{i=1}^{m} \sum_{y \in \mathcal{Y}} \left| \sum_{j=1}^{n_i} I[\forall y' \neq y, f(X_{ij}, y) > f(X_{ij}, y')] - p_{iy} \right|.$$

The ℓ_1 distance between two probability vectors is also known as the variation distance. In

other words, we use the variation distance weighted by the number of instances in a group to quantify the loss. Consider a term in the (weighted) variation distance for the i-th group.

$$
\begin{aligned}
&\left| \sum_{j=1}^{n_i} I[\forall y', f(X_{ij}, y) > f(X_{ij}, y')] - p_{iy} \right| \\
&= \max \left(\sum_{j=1}^{n_i} I[\forall y', f(X_{ij}, y) > f(X_{ij}, y')] - p_{iy}, \right. \\
&\qquad \left. p_{iy} - \sum_{j=1}^{n_i} I[\forall y', f(X_{ij}, y) > f(X_{ij}, y')] \right) \\
&= \max \left(\sum_{j=1}^{n_i} I[\forall y', f(X_{ij}, y) > f(X_{ij}, y')] - p_{iy}, \right. \\
&\qquad \left. \sum_{j=1}^{n_i} I[\exists y', f(X_{ij}, y) < f(X_{ij}, y')] - \bar{p}_{iy} \right) \\
&\leq \max \left(\sum_{j=1}^{n_i} \left(\tilde{H}(g(X_{ij}, y)) - H_0^+(g(X_{ij}, y)) \right) - p_{iy} \right. \\
&\qquad \left. \sum_{j=1}^{n_i} \left(H_1^+(g(X_{ij}, y)) - H_0^+(g(X_{ij}, y)) \right) - \bar{p}_{iy} \right)
\end{aligned}
$$

where $\bar{p}_{iy} = n_i - p_{iy}$, $\tilde{H}(v) = \max(1, v)$, and $g(X_{ij}, y) = \max_{y' \neq y} \left(f(X_{ij}, y') - f(X_{ij}, y) \right)$. Figure 4.2 shows these upper bounds in the two-dimensional case.

Using the above upper bound, we can obtain a upper bound of the total empirical loss

and denote it by $\hat{L}$. We formulate the following optimization program to optimize $\hat{L}$.

$$\begin{aligned}
\min \quad & \lambda \textstyle\sum_{i=1}^{m} \sum_{y \in \mathcal{Y}} \xi_{iy} + \frac{1}{2}\|f\|_{\mathcal{H}}^2 \\
s.t. \quad & \xi_{iy} \geq \textstyle\sum_{j=1}^{n_i} \left(\eta_{ijy} - H_0^+(g(X_{ij}, y))\right) - p_{iy} \\
& \xi_{iy} \geq \textstyle\sum_{j=1}^{n_i} \left(\tau_{ijy} - H_0^+(g(X_{ij}, y))\right) - \bar{p}_{iy} \\
& \eta_{ijy} \geq f(X_{ij}, y') - f(X_{ij}, y) \\
& \tau_{ijy} \geq 1 + f(X_{ij}, y') - f(X_{ij}, y) \\
& \eta_{ijy} \geq 1, \tau_{ijy} \geq 0
\end{aligned} \tag{4.4}$$

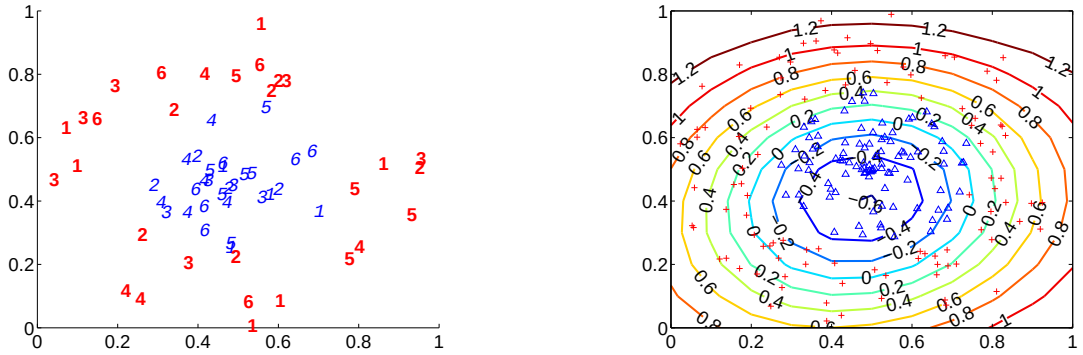
Program (4.4) is nonconvex because the first two (group-level) constraints are nonconvex. We employ CCCP again to solve for w iteratively. In every iteration, we compute the subgradient of $H_0^+(g(X_{ij}, y))$ w.r.t. w as follows.

$$\partial H_0^+ \left(g(X_{ij}, y)\right) = \begin{cases} \left\{\vec{0}\right\} & \text{if } g(X_{ij}, y) < 0, \\ \left\{ \sum_{y' \in \mathcal{A}} \rho_{y'} \delta\Phi(X_{ij}, y') \middle| \sum_{y' \in \mathcal{A}} \rho_{y'} \leq 1 \right\} & \text{if } g(X_{ij}, y) > 0, \\ \left\{ \sum_{y' \in \mathcal{A}} \rho_{y'} \delta\Phi(X_{ij}, y') \middle| \sum_{y' \in \mathcal{A}} \rho_{y'} = 1 \right\} & \text{if } g(X_{ij}, y) > 0, \end{cases}$$

where $\mathcal{A} = \left\{y' \middle| g(X_{ij}, y) = f(X_{ij}, y') - f(X_{ij}, y)\right\}$, $0 \leq \rho_{y'} \leq 1$, and $\delta\Phi(X_{ij}, y') = \Phi(X_{ij}, y') - \Phi(X_{ij}, y)$.

In our experiments, we pick $\rho_{y'}$ to be $0.5/|\mathcal{A}|$ if $g(X_{ij}, y) = 0$ and $1/|\mathcal{A}|$ if $g(X_{ij}, y) > 0$.

To apply non-linear kernels, we resort to the Representer Theorem which says that the



(a) (b)

Figure 4.3: Circle and ring: (a) The six groups in training set, (b) The contour lines of the learned classifier function and the test data

optimal solution of Program (4.4) takes the form below.

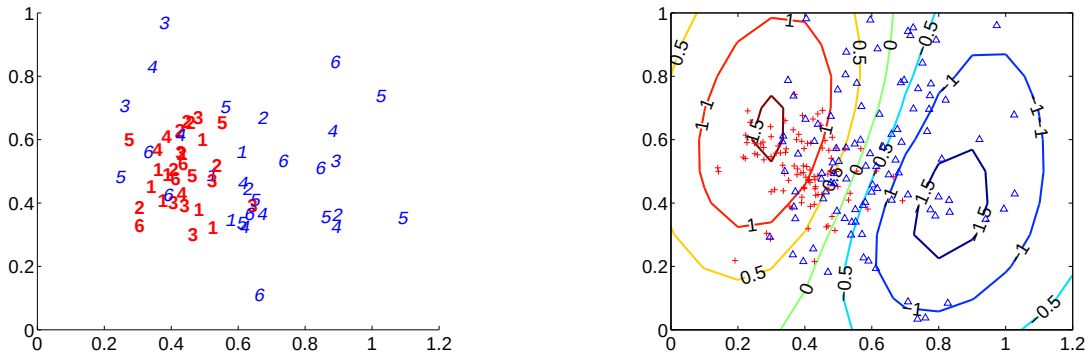
$$f(x,y) = \sum_{i=1}^{m} \sum_{j=1}^{n_i} \sum_{y' \in \mathcal{Y}} \alpha_{ijy'} K((x,y),(X_{ij},y')). \tag{4.5}$$

We can substitute Equation 4.5 into Program (4.4) and solve for α. In this way, we reduce the search space of f from infinite dimensions to $|\mathcal{Y}| \sum_{i=1}^{m} n_i$ dimensions.

4.6 Experimental results

4.6.1 2D toy datasets

We tested our method on two 2D toy datasets. The main advantage of using 2D datasets is that we can visualize the decision boundary. Neither dataset is linearly separable. We chose to use a Gaussian kernel and set the kernel width parameter to the median pairwise distance in the training set. We set the tradeoff parameter between the empirical loss and



(a) (b)

Figure 4.4: Two Gaussian clouds: (a) The six groups in training set, (b) Contour lines of the learned classifier function and the test data

the regularization term to be 1. We started the CCCP with the zero classifier (i.e., $w = \vec{0}$ and $b = 0$).

The first toy dataset is a (negative) circle surrounded by a (positive) ring. The two classes are uniformly distributed radially and spherically. The training set consists of six groups with ten instances each. Three groups contain six positive instances and four negative instances. The other three groups contain four positive instances and six negative instances. Figure 4.3(a) shows the training set with instances in each group denoted by their group IDs. The positive instances are red and bold while the negative instances are blue and italic. Figure 4.3(b) shows the contour lines of the classifier function learned by our method and the test data. The second dataset is overlapping Gaussian clouds, generated from two isotropic Gaussian distributions. The training set consists of six groups with ten instances each. The fraction of positive instances is 0.7 in three groups and 0.3 in the other three. Figure 4.4(a) shows the training set and Figure 4.4(b) shows the learned classifier.

We observe that our method finds a decision boundary that is close to the ground truth. For the circle and ring dataset, the training and test error rates on the first dataset are 0.067 and 0.012. The training and test error rates on the Gaussian clouds dataset are 0.183 and 0.245, respectively. Note that training error is a meaningful measure as the training data were not individually labeled.

4.6.2 USPS and 20newsgroups datasets

Next we tested our method on two benchmark datasets for classification,[1] namely the USPS handwritten digits and 20newsgroups dataset. We report results on '3' vs '8' and '1' vs '7' from the USPS data and comp vs sci and rec vs talk from the 20newsgroups data. These pairs are relatively difficult to differentiate. To create the training set, we randomly partitioned the examples into groups of the same size. A test set consisted of individual examples. We use a test set of 500 and 1000 examples for the USPS data and 20newsgroups data, respectively. In every experiment, we generated ten independent training and test sets and reported the average results.

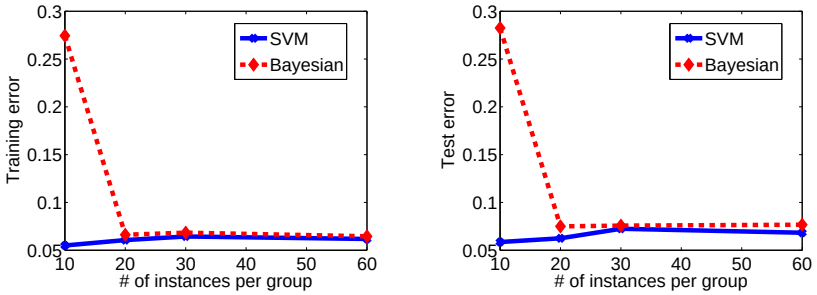
We compare the performance of our method to that of Kück and de Freitas's Bayesian sampling approach [29]. We obtained the code for the Bayesian sampling approach from Carbonetto's website.[2] We chose to use a Gaussian kernel for both methods and set the kernel width parameter to the median pairwise distance in the training set. For our method, we coarsely tuned the tradeoff parameter on a validation set and started the CCCP with the

[1] http://www.cs.toronto.edu/∼roweis/data.html
[2] http://www.cs.ubc.ca/∼pcarbo/objrecls/index.html

zero classifier. For the Bayesian sampling approach, we set the parameters according to the suggestions of Carbonetto et al. [10]. We set $a = 1$ and set b to correspond to a feature selection prior of approximately 10 active kernel centers. We bestowed near uninformative priors on the hyper-parameters: $\mu = \nu = 0.01$ and $\mu_\alpha = \nu_\alpha = 0.01$. We set the stabilization term on the covariance prior $\epsilon = 0.01$. We also coarsely tuned the confidence parameter χ on a validation set. We generated 10000 samples from the posterior distribution of the probit classifier after a burn-in period of 10000 samples.

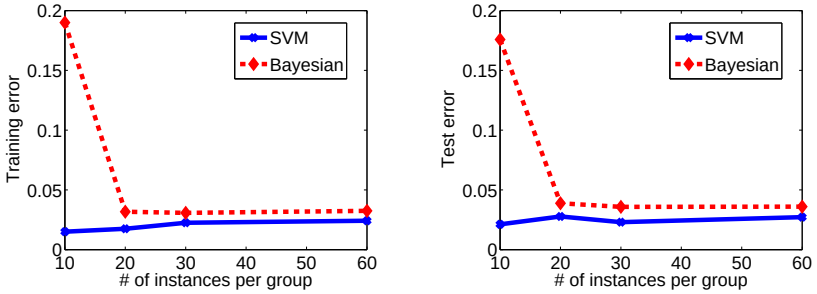
In our first experiment, we examined how the number of instances in a group affects the performance of the two methods when the total number of instances is fixed. We used a total of 120 instances for the USPS data and 240 instances for the 20newsgroups data. We set the fraction of positive instances to 70% in one half of the groups and 30% in the other half. Figure 4.5 and 4.6 depict the training and test errors for different number of instances per group. The two methods have very close performance. The more instances per group, the more ambiguous the information is. However, we observe that coarse group statistics may provide sufficient information to learn a good classifier.

In our second experiment, we examined how the fraction of positive instances in a group affects the performances. Half of the groups have positive as the majority class and the other half have negative as the majority class. We decrease the percentage of instances from the majority class in a group from 100% to 60%. For the USPS data, we fixed the number of groups to 6 and the number of instances per group to 20. For the 20newsgroups



(a) ‘3’ vs ‘8’ (training error)

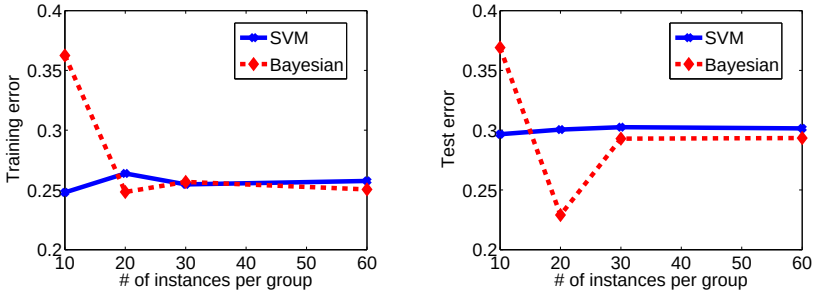
(b) ‘3’ vs ‘8’ (test error)



(c) ‘1’ vs ‘7’ (training error)

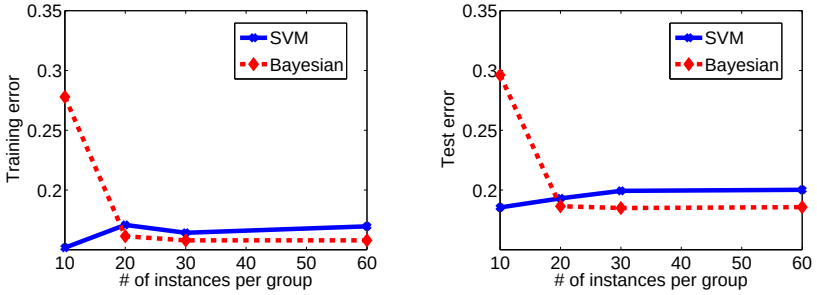
(d) ‘1’ vs ‘7’ (test error)

Figure 4.5: Effect of number of instances per group (USPS data)



(a) comp vs sci (training error)

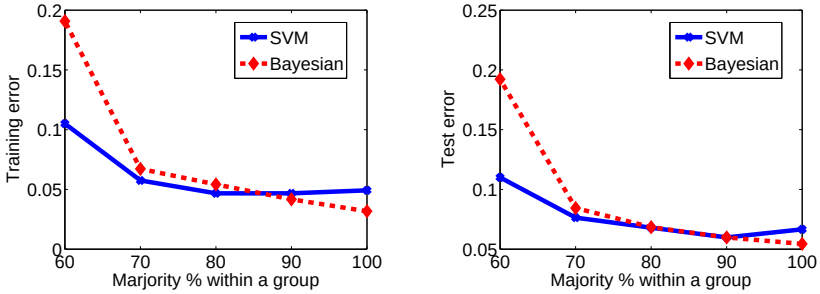
(b) comp vs sci (test error)



(c) rec vs talk (training error)

(d) rec vs talk (test error)

Figure 4.6: Effect of number of instances per group (20newsgroups data)



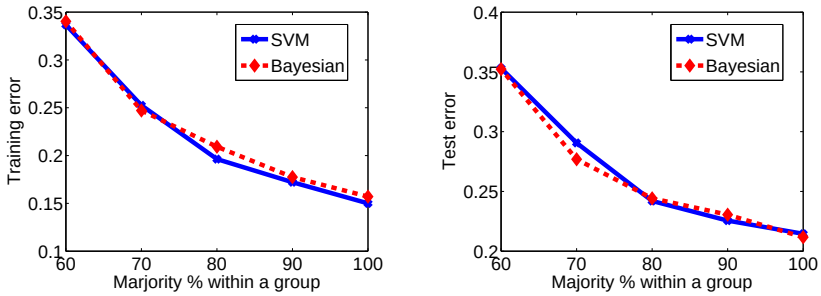
(a) ‘3’ vs ‘8’ (training error) (b) ‘3’ vs ‘8’ (test error)

(c) ‘1’ vs ‘7’ (training error) (d) ‘1’ vs ‘7’ (test error)

Figure 4.7: Effect of group homogeneity (USPS data)

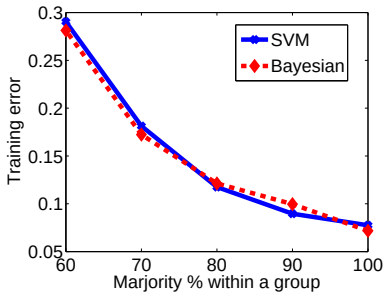
data, we used 12 groups and 20 instances per group. Figure 4.7 and 4.8 depict the training and test errors using the two methods. The two methods have similar performance, As expected, the error rates increase as the majority percentage decreases.

In our third experiment, we tested the robustness of the two methods by introducing noise to the group statistics. We generated a training set as follows. For each group, we flip an unbiased coin. If it is a head, we draw each of its instances from the positive class (one at a time) with a probability of 0.75. If it is a tail, we draw each of its instances from the positive class with a probability of 0.25. To add noise to the group statistics, we flip another unbiased coin for each group. If it is a head, we add a constant to the number of positive instances. Otherwise, we subtract the same constant from the number of positive instances.

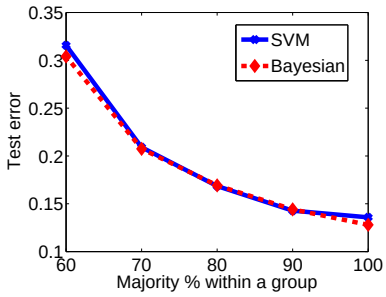


(a) comp vs sci (training error)

(b) comp vs sci (test error)



(c) rec vs talk (training error)

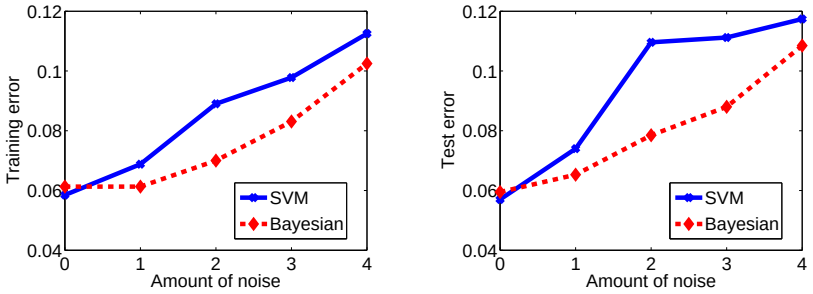


(d) rec vs talk (test error)

Figure 4.8: Effect of group homogeneity (20newsgroups data)

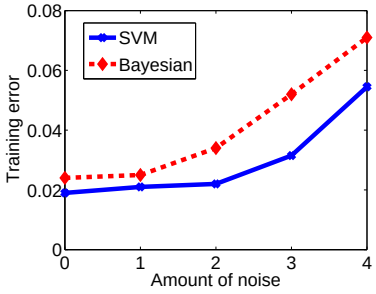
For the USPS data, we set the number of groups to 10 and the number of instances per group to 20. For the 20newsgroups data, we set the number of groups to 20 and the number of instances per group to 20. Figure 4.9 and 4.10 show the training and test errors as we increase the magnitude of the noise from 0 to 4. The Bayesian sampling approach performs better on the USPS data while our method has the edge on the 20newsgroups data. We observe that even when the noise magnitude is relatively large, the performance of both methods does not degrade by much.

Lastly, we report some results using our method in multiclass settings. We chose the joint kernel function $K((x, y), (x', y')) = k(x, x')\delta(y, y')$ where $k(x, x')$ is the Gaussian kernel and $\delta(y, y')$ is the Dirac kernel (i.e., $\delta(y, y') = 1$ if $y = y'$ and zero otherwise). For

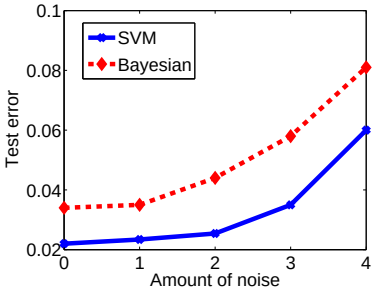


(a) '3' vs '8' (training error)

(b) '3' vs '8' (test error)

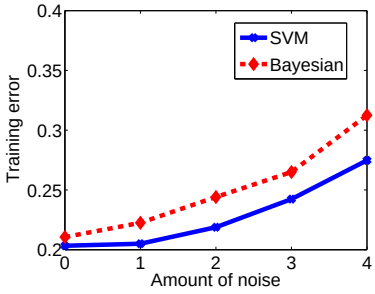


(c) '1' vs '7' (training error)

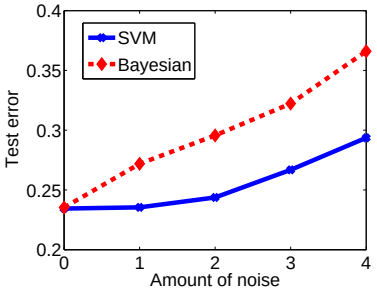


(d) '1' vs '7' (test error)

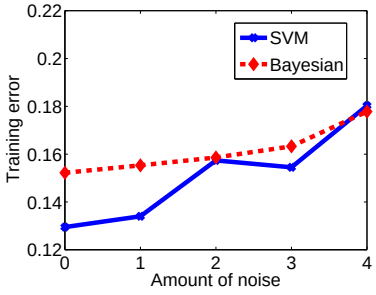
Figure 4.9: Effect of noise in group statistics (USPS data)



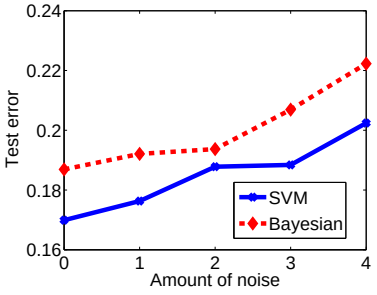
(a) comp vs sci (training error)



(b) comp vs sci (test error)



(c) rec vs talk (training error)



(d) rec vs talk (test error)

Figure 4.10: Effect of noise in group statistics (20newsgroups data)

	USPS	20newsgroups
Training error (noiseless)	0.076	0.442
Test error (noiseless)	0.082	0.454
Training error (noisy)	0.102	0.484
Test error (noisy)	0.103	0.504

Table 4.1: Performance on multiclass classification

the USPS dataset, we focused on '3,' '5,' '6,' and '8.' For the 20newsgroups dataset, we used all four classes. To create a training set, we partition 500 instances randomly into 50 groups of 10 instances. To introduce noise to the groups statistics, for each class in a group, we add or subtract 1 from the number of instances belonging to that class uniformly at random. We normalize the resulting group statistics so that the total number of instances in a group is consistent. Table 4.1 shows the errors using our method.

4.7 Conclusion

We present a theoretical analysis for group statistics classification. From that, we derive a novel SVM-based method that seeks to minimize a generalization error bound. We also consider the problem in the more general multiclass setting and give the first solution. We evaluated our method on 2D toy datasets and some benchmark datasets for classification. The experimental results show that the SVM-based method compares competitively with Kück and de Freitas's Bayesian sampling approach. The SVM-based method has fewer parameters to tune than the Bayesian sampling approach and can be viewed as a deterministic method.

Chapter 5

Multiple Instance Ranking

In multiple instance ranking (MIRank), we are given a number of collections of groups of individual instances along with the identity of the group that contains the instance with the highest rank in every collection. The goal is to learn ranking over the set of possible individual instances. We use the learned ranking function to predict the group containing the highest ranking instance for any given collection. This MIRank problem has applications in computational chemistry and information retrieval. In this chapter, we describe a simple and efficient approach to tackle MIRank problems.

5.1 Introduction

The problem of multiple instance ranking (MIRank) is to learn a ranking function over individual instances from the preference relations among groups of individual instances. In this setting, we are given a number of collections of groups of individual instances. We

know for each collection, the group that contains the highest ranking instance but not the exact identity of the instance with the highest rank. This MIRank problem was proposed by Bergeron et al. [4] and it was studied independently by Hu et al [25].

One important application of MIRank is the hydrogen abstraction problem in computational chemistry. We follow the description in [4]. The goal is to build a model that predicts, for each molecule, the site of abstraction of a hydrogen atom during metabolism. In order to accomplish this, individual hydrogen atoms are first grouped together according to molecular equivalence: hydrogens are placed within the same group if and only if the abstraction of any hydrogen from within the group would result in the same metabolised molecule. In this way, groups are equivalent representations of potential sites of metabolism. Note that experimental data do not show which individual hydrogen is abstracted during metabolism, but rather only to which group the hydrogen atom belongs. Clearly, we can view the hydrogen abstraction problem as a multiple instance ranking problem. We can view molecules as collections, a group of hydrogen atoms as a group of individual instances, and the abstracted hydrogen as the highest ranking instance.

Another application of MIRank is information retrieval. Suppose we want to rank images according to their relevance to a particular topic. We collect pairs of images and the preference relation between each pair (e.g., image A is more relevant to image B). Since an image can contain more than one thing, we represent every image as a set of segments. Furthermore, we may want to identify the most relevant segment in an image as well. We

can learn a ranking function in the MIRank setting. We can view every image pair as a collection and every image as a group of individual instances.

Like multiple-instance classification and group statistics classification, the challenge of the MIRank problem comes from the ambiguity in the label information in the training data. We propose a probabilistic model for the MIRank problem. The probabilistic model is based on a generalized Bradley-Terry model [26]. We show that optimizing the probabilistic model is equivalent to approximate empirical risk minimization. The resulting optimization problem is unconstrained and can be solved by efficient numerical algorithms (e.g., quasi-newton and scaled conjugate gradient [6]). We also explore boosting in the MIRank setting. Furthermore, we extend the relevance vector machine to the MIRank setting for automatic feature selection.

5.2 Related work

Multiple-instance (MI) learning is a popular research topic in machine learning (see Chapter 4 for a more detail discussion). Most previous MI works focus on classification. Recently, Bergeron et al. [4] and Hu et al. [25] independently considered extending MI learning to ranking. Bergeron et al. propose to solve MIRank problems using successively linear programming. On the other hand, Hu et al. consider three variants of MIRank, namely using the average, the $\max$, and the approximate softmax of the instances to represent each group. The first case be solved as a standard quadratic program. The second case can be

solved as a sequence of quadratic programs using the Concave-Convex procedure (CCCP) (see Chapter 2). The third case also involves solving a sequence of quadratic program but it is not clear if it has any convergence guarantee. Although Bergeron et al. and Hu et al. report some success in their experiments, their methods can be computationally expensive. A more scalable and efficient method is highly desirable. Another drawback of their methods is the lack of a probabilistic meaning (i.e., what is the probability that a group is the one containing the highest ranking instance in a collection?).

5.3 Problem Definition

We need some notation to formally state the MIRank problem.

$\mathcal{X}$	the space of feature vectors representing individual instances
$X^{(k)}$	collection k
$X_i^{(k)}$	group i in collection k
$X_{ij}^{(k)}$	feature vector of instance j in group i in collection k
$Y^{(k)}$	the identity of the group that contains the highest ranking instance in collection k

Table 5.1: Notation for MIRank

Given a training set $\{\left(X^{(k)}, Y^{(k)}\right)\}$, the goal of MIRank is to learn a ranking function $f : \mathcal{X} \rightarrow \Re$. The learned ranking function f is later used to predict which group contains the highest ranking instance for any test collection.

5.4 A Probabilistic Model

The Bradley-Terry model [15] is a popular model for paired comparisons:

$$P(\text{individual } i \text{ beats individual } j) = \frac{\pi_i}{\pi_i + \pi_j} \tag{5.1}$$

where π_i is the (positive) skill of the i-th individual. Huang et al. [26] extend it for paired team comparisons:

$$P(\text{team } A \text{ beats team } B) = \frac{\sum_{a \in A} \pi_a}{\sum_{a \in A} \pi_a + \sum_{b \in B} \pi_b} \quad . \tag{5.2}$$

For the MIRank problem, we say that a group g is the favorite group if it contains the highest ranking instance in a collection and we assume

$$P(\text{group } g \text{ is labeled as the favorite group}) = \frac{\sum_j e^{f(X_{gj})}}{\sum_{i \neq g} \sum_j e^{f(X_{ij})}} \quad . \tag{5.3}$$

Note that unlike [15] and [26], we consider the "skill" of an individual instance as a function of its feature vector. This difference allows us to rank instances outside the training set as well. The ranking function f can be found by maximum likelihood estimation (MLE) or maximum a posterior (MAP) estimation (if a prior distribution is assumed over f).

Approximate empirical risk minimization interpretation

Interestingly, we can also motivate the generalized Bradley-Terry model from the risk minimization perspective. We define the loss incurred by a collection $(X, Y = g)$ as

$$\max_i \max_j f(X_{ij}) - \max_j f(X_{gj}) \quad . \tag{5.4}$$

If group g contains the instance with the highest skill, the loss is zero. Otherwise, the loss is the difference between the highest skill in the collection and that in group g. The $softmax$ function is a *soft* version of the max function.

$$softmax(z_1, \ldots, z_n) = \log(\sum_{i=1}^{n} e^{z_i}) \quad . \tag{5.5}$$

Replacing max by $softmax$ in Equation 5.4, we obtain

$$\log\left(\sum_i \sum_j e^{f(X_{ij})}\right) - \log\left(\sum_j e^{f(X_{gj})}\right) \tag{5.6}$$

which is the negative log-likelihood using Equation 5.3. Therefore, minimizing the approximate loss function is equivalent to maximizing the likelihood in the generalized Bradley-Terry model.

5.5 Estimation methods

5.5.1 Linear model

The first method seeks to learn a linear ranking function $f(x) = w \cdot x$. We assume a isotropic Gaussian prior on w. The posterior distribution of w is

$$\begin{aligned} P(w|\{(X^{(k)}, Y^{(k)})\}) &\propto \prod_k P(Y^{(k)}|X^{(k)}, w)P(w) \\ &\propto \prod_k \frac{\sum_j e^{w \cdot X^{(k)}_{Y^{(k)}j}}}{\sum_i \sum_j e^{w \cdot X^{(k)}_{ij}}} \cdot e^{-\frac{\lambda}{2} w \cdot w} \quad . \end{aligned} \tag{5.7}$$

The maximum a posterior (MAP) estimate of w is obtained by solving

$$\min_w \ \mathcal{L}_{MAP} = \sum_k \left(\log \left(\sum_i \sum_j e^{w \cdot X^{(k)}_{ij}} \right) - \log \left(\sum_j e^{w \cdot X^{(k)}_{Y^{(k)}j}} \right) \right) + \frac{\lambda}{2} w \cdot w \quad . \tag{5.8}$$

Since log-sum-exp is convex, the objective function is the difference of two convex functions. Equation (5.8) is a nonconvex unconstrained optimization problem. We can solve for a local optimum of w using the CCCP or directly using numerical algorithms such as quasi-Newton and scaled conjugate gradient [6]. Denote the objective function in Equation (5.8) by $\mathcal{L}_{MAP}$. The corresponding gradient and Hessian matrix are

$$\frac{\partial \mathcal{L}_{MAP}}{\partial w} = \sum_k (-z_k + \tilde{z}_k) + \lambda w \tag{5.9}$$

$$\frac{\partial^2 \mathcal{L}_{MAP}}{\partial w^2} = \sum_k \left(z_k z_k^T - \mathcal{Z}_k - \tilde{z}_k \tilde{z}_k^T + \widetilde{\mathcal{Z}}_k \right) + \lambda \mathrm{I} \tag{5.10}$$

where I denotes the identity matrix and

$$z_k = \frac{\sum_j e^{w \cdot X^{(k)}_{Y^{(k)}j}} X^{(k)}_{Y^{(k)}j}}{\sum_j e^{w \cdot X^{(k)}_{Y^{(k)}j}}} \tag{5.11}$$

$$\tilde{z}_k = \frac{\sum_i \sum_j e^{w \cdot X^{(k)}_{ij}} X^{(k)}_{ij}}{\sum_i \sum_j e^{w \cdot X^{(k)}_{ij}}} \tag{5.12}$$

$$\mathcal{Z}_k = \frac{\sum_j e^{w \cdot X^{(k)}_{Y^{(k)}j}} X^{(k)}_{Y^{(k)}j} X^{(k)\ T}_{Y^k j}}{\sum_j e^{w \cdot X^{(k)}_{Y^{(k)}j}}} \tag{5.13}$$

$$\widetilde{Z}_k = \frac{\sum_i \sum_j e^{w \cdot X^{(k)}_{ij}} X^{(k)}_{ij} X^{(k)T}_{ij}}{\sum_i \sum_j e^{w \cdot X^{(k)}_{ij}}} \tag{5.14}$$

5.5.2 Ensemble model

The second method seeks to learn an ensemble model $f(x) = \sum_t \alpha_t h_t(x)$ where h_t's are some weak classifiers. We choose the negative log-likelihood

$$\mathcal{L}_{ML} = \sum_k \left(\log \left(\sum_i \sum_j e^{f\left(X^{(k)}_{ij}\right)} \right) - \log \left(\sum_j e^{f\left(X^{(k)}_{Y^{(k)}j}\right)} \right) \right) \tag{5.15}$$

as the loss function and minimizes it using the AnyBoost framework [33] (see section 2.2). The functional gradient of the negative log-likelihood is

$$\frac{\partial \mathcal{L}_{ML}}{\partial f\left(X_{uv}^{(k)}\right)} = \begin{cases} \frac{e^{f\left(X_{uv}^{(K)}\right)}}{\sum_i \sum_j e^{f\left(X_{ij}^{(K)}\right)}} & \text{if } u \neq Y^{(k)}; \\ \frac{e^{f\left(X_{uv}^{(K)}\right)}}{\sum_i \sum_j e^{f\left(X_{ij}^{(K)}\right)}} \left(-\frac{\sum_{i \neq Y^{(k)}} \sum_j e^{f\left(X_{ij}^{(K)}\right)}}{\sum_j e^{f\left(X_{Y^{(k)}j}^{(K)}\right)}} \right) & \text{if } u = Y^{(k)} \ . \end{cases} \tag{5.16}$$

Therefore, in every iteration, we set the label and weight of instance X_{uv}^k as $sign\left(\frac{\partial \mathcal{L}_{ML}}{\partial f\left(X_{uv}^k\right)}\right)$ and $\frac{abs\left(\frac{\partial \mathcal{L}_{ML}}{\partial f\left(X_{uv}^k\right)}\right)}{\sum_k \sum_i \sum_j abs\left(\frac{\partial \mathcal{L}_{ML}}{\partial f\left(X_{ij}^k\right)}\right)}$, respectively. We find the weak classifier $h_t(x)$ that minimizes the weight error on the training set and then use line search to find its weight α_t that minimizes $\mathcal{L}_{ML}$.

5.6 Automatic Feature Selection

In many applications, model interpretation is just as important as prediction accuracy and a sparse model is highly desirable. Also, it is troublesome to tune the parameter λ in the MAP estimation method manually. We apply a feature selection technique originally proposed for the relevance vector machine (RVM) [49] in MIRank. Instead of assuming an isotropic Gaussian prior on w, we assume that different components of w may have different variances, i.e., $w \sim N(\vec{0}, A^{-1})$ where $A = diag(\alpha_1, \ldots, \alpha_d)$. The type-II maximum

likelihood method is to find A that maximizes the marginal likelihood

$$P(\{Y^{(k)}\}|\{X^{(k)}\}; A) = \int \prod_k P(Y^{(k)}|X^{(k)}, w)P(w; A)dw \quad . \tag{5.17}$$

However, the integral is intractable for the generalized Bradley-Terry model. Denote the logarithm of the integrand by $\Psi(w)$. We approximate $\Psi(w)$ using the second-order Taylor expansion at the current MAP estimate $\hat{w}^{MAP}$.

$$\Psi(w) \approx \Psi(\hat{w}_{MAP}) + \frac{1}{2}(w - \hat{w}_{MAP})^T \nabla^2 \Psi(w)(w - \hat{w}_{MAP}) \tag{5.18}$$

where $\nabla^2 \Psi(w) = \frac{\partial^2 \Psi(w)}{\partial w^2}$. The resulting integral is easy to compute.

$$P(\{Y^{(k)}\}|\{X^{(k)}\}; A) = \int e^{\Psi(w)} dw \tag{5.19}$$

$$\approx e^{\Psi(\hat{w}_{MAP})} \int e^{\frac{1}{2}(w-\hat{w}^{MAP})^T \nabla^2 \Psi(w)(w-\hat{w}_{MAP})} dw \tag{5.20}$$

$$= \prod_k P(Y^{(k)}|X^{(k)}, \hat{w}_{MAP})P(\hat{w}_{MAP}; A)(2\pi)^{d/2} \left| -\nabla^2 \Psi(w) \right|^{-1/2} \tag{5.21}$$

The logarithm of the approximate marginal likelihood is

$$\log P(\{Y^{(k)}\}|\{X^{(k)}\}; A)$$
$$\approx \sum_k \log P(Y^{(k)}|X^{(k)}, \hat{w}_{MAP}) - \frac{1}{2}\hat{w}_{MAP}^T A \hat{w}_{MAP} + \frac{1}{2}\log|A| - \frac{1}{2}\log\left|-\nabla^2\Psi(w)\right|. \tag{5.22}$$

The derivative can be approximated as

$$\frac{\partial \log P(\{Y^{(k)}\}|\{X^{(k)}\}; A)}{\partial A} \approx -\frac{1}{2}\hat{w}_{MAP}\hat{w}_{MAP}^T + \frac{1}{2}A^{-1} - \frac{1}{2}\nabla^{-2}\Psi(w) \tag{5.23}$$

where $\nabla^{-2}\Psi(w)$ is the matrix inverse of $\nabla^2\Psi(w)$. Taking into account that A is a diagonal matrix, we have

$$\frac{\partial \log P(\{Y^{(k)}\}|\{X^{(k)}\}; A)}{\partial \alpha_i} \approx \frac{1}{2}\hat{w}_{MAP,i}^2 + \frac{1}{2\alpha_i} - \frac{1}{2}\nabla^2\Psi(w) \tag{5.24}$$

where $\nabla^{-2}\Psi(w)_{ii}$ is the (i,i)-element of $\nabla^{-2}\Psi(w)$. Setting the derivative to zero, we obtain the following update rule.

$$\alpha_i^{new} = \frac{1}{\hat{w}_{MAP,i}^2 + \nabla^{-2}\Psi(w)_{ii}} \tag{5.25}$$

The final algorithm is described as Algorithm 1. We fix A to find the MAP estimate $\hat{w}_{MAP}$ and then update A using Equation (5.25). We repeat the two steps until A converges. After

Algorithm 1 MIRank-RVM

initialize $w = \vec{0}$ and A as identity matrix
repeat
find w that maximizes $\Psi(w)$, the log of the integrand in Equation (5.17),
by numerical optimization methods
update the diagonal elements of A using Equation (5.25)
until A does not change

a number of iterations, some α_i's become very large and we can remove those irrelevant features from further consideration. Often times, this procedure results in a sparse model that is good for model interpretation and generalization.

5.7 Experiments

Following [4], we tested our methods on the CYP3A4 substrate dataset[1] and two versions of a census dataset (census-16h and census-16l) from the Data for Evaluating Learning in Valid Experiments (DELVE) repository[2]. The CYP3A4 substrate dataset is made up of 227 small drug-like compounds. A series of 36 descriptors for each hydrogen atom for all molecules are calculated. For each molecule, the goal is to predict which group a hydrogen atom is abstracted, and it is not known exactly which hydrogen is abstracted.

The census dataset consists of 22784 towns spread among the 50 states of the United States of America. This study only considered the 3054 towns of more than 10000 inhabitants. Each town is assigned a 5-digit Federal Information Processing Standard (FIPS) place code (not a zip code). Typically, this dataset is used in a regression setting to model

[1] http://reccr.chem.rpi.edu/MIRank/
[2] http://www.cs.toronto.edu/∼delve/

Dataset	successive LP [4]	MAP	MAP+RVM	Boosting
CYP3A4 substrate	$70.9\% \pm 6.9$	$69.9\% \pm 5.9$	$72.2\% \pm 5.5$	$71.9\% \pm 6.7$
Census-16H	$60.3\% \pm 15.1$	$58.1\% \pm 14.5$	$62.2\% \pm 14.3$	$55.6\% \pm 16.8$
Census-16L	$57.5\% \pm 16.0$	$62.5\% \pm 15.2$	$64.7\% \pm 13.4$	$47.8\% \pm 16.0$

Table 5.2: Rank-2 prediction accuracies

the response – which is the town's median housing unit price. The census-16h and census-16l datasets differ in their features: each consists of 16 features drawn from the 1990 census. These datasets are fitted into the multiple instance ranking framework as follows. States are collections, divisions of towns are groups and towns are individual instances. For each state, towns whose place codes begin with the same digit are assigned to the same division. As no place code starts with '0', there are up to 9 divisions per state. The task is to predict, for each state, the division that contains the town with the highest median housing unit price.

The experimental design is as follows. Each dataset was randomly split into training, validation and testing subsets consisting of 60%, 20% and 20% of the collections, respectively. The parameters of the algorithms (λ for the linear model method and the number of weak learners T for the ensemble model method) are tuned on the validation set. The experiment was repeated 32 times for each dataset. The average accuracy is reported in Table 5.2, along with the standard deviation. Note that a prediction is considered as correct if the top two guesses contain the most preferred group in the collection. The same metric is employed in [4]. We can see that the MAP estimation of a linear model has a similar performance to successive linear programming. The MAP linear model has a slightly

lower accuracy than successive linear programming on the CYP3A3 substrate dataset and census-16H dataset but is more accurate on census-16L dataset. The MAP estimation combined with automatic feature selection gives the best accuracy on all three datasets. Our boosting approach used decision stumps [7] as the weak learners. It achieved the second best accuracy on the CYP3A4 substrate dataset, but it did not perform well on the two census datasets because of overfitting. Although the performance of the boosting approach is somewhat disappointing, we believe that it can be a good choice when the optimal ranking function is nonlinear.

5.8 Conclusion

We propose a probabilistic model for multiple instance ranking and present methods to learn the model. Our first method learns a linear ranking function by MAP estimation while our second method learns an ensemble ranking function by MLE method. Both methods involve solving an unconstrained optimization problem, which is nonconvex but belongs to the class of the difference of convex functions. Furthermore, we extend the feature selection technique used in the relevance vector machine to the MIRank setting. Our experimental results show that the MAP linear model has a similar performance to successive linear programming. The MAP estimation with automatic feature selection produced the best prediction accuracy. Our boosting method seems susceptible to overfitting and did not do well.

Chapter 6

Conclusion

We argue that standard classification and ranking suffer from high information acquisition costs at both the test stage and the training stage. To reduce the information acquisition cost during testing, we can obtain information about test examples sequentially. Based on the currently available information, we decide whether to reject the example or to obtain further information. This sequential stopping framework is especially suitable for product testing in manufacturing where a good product item need to pass every test and a product item is flagged as faulty immediately if it fails a test. We propose catenary support vector machine (catSVM) to the sequential stopping problem and give a generalization bound for it. We demonstrate the advantage of catSVM over single-stage SVM and its closest competitor, chained boosting, on the UCI heart dataset and on a face detection task.

Labels of training examples are often expensive to obtain. To reduce the label collection cost, we can perform learning from aggregate statistics. We propose a SVM method

to learn to classify individual examples from group label proportions. We provide a simple theoretical analysis for group statistics classification. Our experiments show that our SVM method has comparable performance as a Bayesian sampling approach. Also, we propose several methods to learn to rank individual examples from preference relations at the group level. Our experimental results show that the MAP estimation of a linear model with automatic feature selection produces the best accuracy while boosting seems susceptible to overfitting. Learning from aggregate statistics opens new opportunities for data users and can benefit many applications. But it also raises privacy concerns because people may be able to infer private individual actions or preferences from public aggregate statistics.

There are a number of directions for future work. The sequential stopping framework we consider can only reject test examples early but it cannot accept any examples until the last stage. In some applications, there are many easy positive examples that can be accepted by looking at the first few features. Extending catSVM to two-sided early detection is an useful next step. Another direction is improve the scalability of catSVM. The size of the catSVM optimization problem depends on the product of the number of training examples and the number of stages in the processing pipeline. Currently, we use a generic interior-point solver to solve the quadratic program in each iteration of the CCCP. But the interior-point solver quickly runs out of memory as the problem size increases. It may be possible to make catSVM optimization more scalable using cutting-plane methods.

Moreover, it would be interesting to tighten the generalization bound for group statistics

classification. The assumptions in our analysis are admittedly rather strong. It would be desirable to relax some of the assumptions. Especially, it would be very useful to consider the case when the group label proportions are noisy. It would also be interesting to extend the theoretical analysis to multiclass setting. In addition, our solution for multiclass group statistics classification involves solving quadratic programs with the number of constraints quadratic in the number of classes. A generic interior-point solver does not scale to a large number of classes. Again, cutting-plane methods and bundle methods [42] may be of help. Finally, extending our methods to other variants of multiple instance ranking (e.g., allowing multiple individual instances in different groups to tie for the top spot) is an interesting topic.

Bibliography

[1] Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems*, volume 15, 2003.

[2] Andreas Argyriou, Raphael Hauser, Charles A. Micchelli, and Massimiliano Pontil. A DC-programming algorithm for kernel selection. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 41–48, 2006.

[3] Peter L. Bartlett and Shahar Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 2:463–482, 2002.

[4] Charles Bergeron, Jed Zaretzki, Curt Breneman, and Kristin P. Bennet. Multiple instance ranking. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

[5] Jinbo Bi, Senthil Periaswamy, Kazunori Okada, Toshiro Kubota, Glenn Fung, Marcos Salganicoff, and Bharat Rao. Computer aided detection via asymmetric cascade of sparse hyperplane classifiers. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*, 2006.

[6] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[7] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 1st edition edition, 2007.

[8] Antonie Bordes, Léon, Patrick Gallinari, and Jason Weston. Solving multiclass support vector machines with LaRank. In *Proceedings of the 24th International Conference on Machine Learning*, pages 89–96, 2007.

[9] Christopher J.C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 89–96, 2005.

[10] Peter Carbonetto, Gyuri Dorkó, Cordelia Schmid, Hendrik Kück, and Nando de Freitas. Learning to recognize objects with little supervision. *International Journal of Computer Vision*, 77(1–3):219–237, 2008.

[11] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[12] Wei Chu and S. Sathiya Keerthi. New approaches to support vector ordinal regression. In *Proceedins of International Conference on Machine Learning*, pages 145–152, 2005.

[13] Ronan Collobert, Jason Weston, and Leon Bottou. Trading convexity for scalability. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

[14] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

[15] Herbert A. David. *The method of paired comparisons*. Oxford University Press, New York, second edition edition, 1988.

[16] Thomas G. Dietterich, Richard H. Lathrop, and Thomás Lozanao-Pérez. Solving multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1–2):31–71, 1997.

[17] Harris Drucker, Chris J.C. Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. In *Advances in Neural Information Processing Systems*, volume 9, pages 155–161. the MIT Press, 1996.

[18] Murat Dundar and Jinbo Bi. Joint optimization of cascaded classifiers for computer aided detection. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Rocognition*, 2007.

[19] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. In *Proceedings of the 15th International Conference on Machine Learning*, pages 170–178, 1998.

[20] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

[21] Glenn Fung, Murat Dundar, Balaji Krishnapuram, and R. Bharat Rao. Multiple instance learning for computer aided diagnosis. In *Advanced in Neural Information Processing Systems*, volume 19, 2007.

[22] Thomas Gärtner, Peter A. Flach, Adam Kowalczyk, and Alex J. Smola. Multi-instance kernels. In *Proceedings of the 19th International Conference on Machine Learning*, pages 179–186, 2002.

[23] Ran Gilad-Bachrach, Amir Navot, and Naftali Tishby. Margin based feature selection – theory and algorithms. In *Proceedings of the 21st International Conference on Machine Learning*, pages 43–50, 2004.

[24] R. A. Howard and J. E. Matheson. Influence diagrams. In R. A. Howard and J. E. Matheson, editors, *Readings on the Principles and Applications of Decision Analysis*, volume 2, pages 719–762. Strategic Decision Group, 1984. article dated 1981.

[25] Yang Hu, Mingjing Li, and Nenghai YU. Multiple-instance ranking: Learning to rank images for image retrieval. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Rocognition*, 2008.

[26] Tzu-Kuo Huang, Chih-Jen Lin, and Ruby C. Weng. A generalized bradley-terry model: From group competition to individual skills. In *Advanced in Neural Information Processing Systems*, volume 17, pages 601–608. The MIT Press, 2005.

[27] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 133–142, 2002.

[28] Hendrik Kück, Peter Carbonetto, and Nando de Freitas. A constrained semi-supervised learning approach to data association. In *Proceedings of the 8th European Conference for Computer Vision, Part III*, pages 1–12, 2004.

[29] Hendrik Kück and Nando de Freitas. Learning to classify individuals based on group statistics. In *Proceedings of the Twenty-First International Conference on Uncertainty in Artificial Intelligence*, 2005.

[30] Gert R. G. Lanckriet, Nello Cristianini, Peter L. Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. In *Proceedings of the 19th International Conference in Machine Learning*, pages 323–330, 2002.

[31] Yufeng Liu, Xiaotong Shen, and Hani Doss. Multicategory ψ-learning and support vector machine: computational tools. *Journal of Computational and Graphical Statistics*, 14(1):219–236, 2005.

[32] Oded Maron and Tomás Lozano-Pérez. A framework for multiple-instance learning. In *Advanced in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.

[33] Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Boosting algorithms as gradient descent. In *Advances in Neural Information Processing Systems*, volume 12, pages 512–518. The MIT Press, 2000.

[34] C. McDiarmid. On the method of bounded differences. *Surveys in Combinatorics*, pages 148–188, 1989.

[35] MIT. CBCL face database #1, 2000. http://cbcl.mit.edu/cbcl/software-datasets/FaceData2.html.

[36] Novi Quadrianto, Alex Smola, Tiberio Caetano, and Quoc Viet Le. Estimating labels from label proportions. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

[37] Bernhard Schölkopf, John Platt, J. Shawe-Taylor, Alex Smola, and R. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, July 2001.

[38] Bernhard Schölkopf and Alex J. Smola. *Learning with kernels*. The MIT Press, Cambridge, MA, 2002.

[39] Christian R. Shelton, Wesley Huie, and Kin Fai Kan. Chained boosting. In *Advanced in Neural Information Processing Systems*, 2007.

[40] Xiaotong Shen, George C. Tseng, Xuegong Zhang, and Wing Hung Wong. On ψ-learning. *Journal of the American Statistical Association*, 98(463):724–734, 2003.

[41] Alex J. Smola, S.V.N. Vishwanathan, and Thomas Hofmann. Kernel methods for missing variables. In *Proceedings of the Tenth Workshop on Artificial Intelligence and Statistics*, 2005.

[42] Alex J. Smola, S.V.N. Vishwanathan, and Quoc Le. Bundle methods for machine learning. In *Advanced in Neural Information Processing Systems*, 2008.

[43] Jan Šochman and Jiří Matas. Waldboost — learning for time constrained sequential detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 150–156, 2005.

[44] Nathan Srebro, Jason D. M. Rennis, and Tommi S. Jaakkola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems*, volume 17, pages 1329–1336, Cambridge, MA, 2005. the MIT Press.

[45] Bharath K. Sriperumbudur, David A. Torres, and Gert R. G. Lanckriet. Sparse eigen-methods by D.C. programming. In *Proceedings of the 24th International Conference on Machine Learning*, pages 831–838, 2007.

[46] Pham Dinh Tao and Le ThiHoai An. A D.C. optimization algorithm for solving the trust-region subproblem. *SIAM Journal on Optimization*, 8(2):476–505, 1998.

[47] Qingping Tao, Stephen Scott, N. V. Vinodchandran, and Thomas Takeo Osugi. SVM-based generalized multiple-instance learning via approximate box counting. In *Proceedings of the 21st International Conference on Machine Learning*, pages 101–108, 2004.

[48] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In *Advances in Information Processing Systems*, volume 16, 2004.

[49] Michael E. Tipping. The relevance vector machine. In *Advances in Neural Information Processing Systems*, volume 22, pages 652–658, 2000.

[50] Vladmir Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, 1998.

[51] Vladmir Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, 2nd edition, 1999.

[52] Paul Viola and Michael Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. In *Advances in Neural Information Processing Systems 14*, pages 1311–1318, 2002.

[53] Paul A. Viola, John Platt, and Cha Zhang. Multiple instance boosting for object detection. In *Advances in Neural Information Processing Systems*, volume 17, 2006.

[54] Jun Wang and Jean-Daniel Zucker. Solving the multiple-instance problem: A lazy learning approach. In *Proceedings of the 17th International Conference on Machine Learning*, pages 1119–1126, 2000.

[55] Nils Weidmann, Eibe Frank, and Bernhard Pfahringer. A two-level learning method for generalized multi-instance problems. In *Proceedings of the 14th the European Conference on Machine Learning*, pages 468–479, 2003.

[56] Kilian Weinberger, John Blitzer, and Lawrence Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems*, volume 18, pages 1473–1480. the MIT Press, 2006.

[57] Jianxin Wu, Matthew D. Mullin, and James M. Rehg. Linear asymmetric classifier for cascade detectors. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 988–995, 2005.

[58] Linli Xu, James Neufeld, Bryce larson, and Dale Schuurmans. Maximum margin clustering. In *Advances in Neural Information Processing Systems*, volume 17, Cambridge, MA, 2005. the MIT Press.

[59] Alan L. Yuille and Anand Rangarajan. The concave-convex procedure (CCCP). In *Advanced in Neural Information Processing Systems*, volume 14, Cambridge, MA, 2002. the MIT Press.

[60] Qi Zhang and Sally A. Goldman. EM-DD: An improved multiple-instance learning. In *Advances in Neural Information Processing Systems*, volume 14, Cambridge, MA, 2002. the MIT Press.