Effective and Efficient Optimization Methods for Kernel Based Classification Problems

by

Aditya Tayal

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2014

Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Aditya Tayal
2014

Abstract

Kernel methods are a popular choice in solving a number of problems in statistical machine learning. In this thesis, we propose new methods for two important kernel based classification problems: 1) learning from highly unbalanced large-scale datasets and 2) selecting a relevant subset of input features for a given kernel specification.

The first problem is known as the rare class problem, which is characterized by a highly skewed or unbalanced class distribution. Unbalanced datasets can introduce significant bias in standard classification methods. In addition, due to the increase of data in recent years, large datasets with millions of observations have become commonplace. We propose an approach to address both the problem of bias and computational complexity in rare class problems by optimizing area under the receiver operating characteristic curve and by using a rare class only kernel representation, respectively. We justify the proposed approach theoretically and computationally. Theoretically, we establish an upper bound on the difference between selecting a hypothesis from a reproducing kernel Hilbert space and a hypothesis space which can be represented using a subset of kernel functions. This bound shows that for a fixed number of kernel functions, it is optimal to first include functions corresponding to rare class samples. We also discuss the connection of a subset kernel representation with the Nyström method for a general class of regularized loss minimization methods. Computationally, we illustrate that the rare class representation produces statistically equivalent test error results on highly unbalanced datasets compared to using the full kernel representation, but with significantly better time and space complexity. Finally, we extend the method to rare class ordinal ranking, and apply it to a recent public competition problem in health informatics.

The second problem studied in the thesis is known as the feature selection problem in literature. Embedding feature selection in kernel classification leads to a non-convex optimization problem. We specify a primal formulation and solve the problem using a second-order trust region algorithm. To improve efficiency, we use the two-block Gauss-

Seidel method, breaking the problem into a convex support vector machine subproblem and a non-convex feature selection subproblem. We reduce possibility of saddle point convergence and improve solution quality by sharing an explicit functional margin variable between block iterates. We illustrate how our algorithm improves upon state-of-the-art methods.

Acknowlegements

I would like to thank my supervisors Professor Yuying Li and Professor Thomas Coleman for allowing me explore the field of machine learning and optimization. I have benefitted tremendously in both research and life skills during the course of my PhD years.

I thank my committee members, Stephen Wright (external), Mu Zhu, Peter Forsyth and Justin Wan, for taking the time to review the thesis and give me valuable comments.

I would also like to thank fellow mates in the Scientific Computing lab, Kai Ma, Ken Chan, Eddie Cheung, Nick Nian, Haofan Zhang, Parsiad Azimzadeh and Swathi Amarala, for engaging and memorable discussions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Kernel methods are a popular choice in solving a number of problems in statistical machine learning. The key benefit of kernel methods is separation of the training algorithm from data representation, encoded via a positive semi-definite kernel. Kernels allow us to estimate complex or nonlinear functions in a hypothesis space by implicitly estimating a linear model in a (usually high-dimensional) mapped space, known as the feature space. A kernel computes the dot product of two points in this feature space. Therefore, we can estimate linear models in the feature space without explicitly working in the high-dimensional space, as long as we can formulate the estimation process entirely in terms of dot products or kernel evaluations.

The model estimation process or learning algorithm can be decoupled from the specifics of the application area [35]. Domain knowledge or data structure for a particular application can be incorporated by choosing an appropriate kernel specification. For example, state-of-the-art vision classification systems often employ vocabulary-based representations, which can be effectively expressed via histogram kernels [5], or syntax based kernels can be defined over abstract structures, such as strings, trees or graphs [e.g. see 47, and references therein]. Thus a kernel can be used to capture salient features to represent the data space of a learning problem.

The learning algorithm is usually formulated as an optimization problem. Since the kernel is positive semi-definite, this often leads to a convex optimization problem—which has a unique minimum if it exists, and where any local minimum is also a global minimum. Thus the learning algorithm does not involve heuristic choices, such as learning rates or initial points. Moreover, the same optimization algorithm can be used to solve problems with different kernel specifications. In this respect, kernel methods can be considered modular.

The theory of kernels is quite old, dating back to 1909 with Mercer [68]. The interpretation of kernels as dot products in a feature space was introduced into machine learning in 1964 by Aizerman et al. [7] on the method of potential functions. However, its possibilities were not fully understood until relatively recently by the introduction of the support vector machine by Boser et al. [15] in 1992.

By working implicitly in a high-dimensional feature space, kernels increase the risk of overfitting and ill-posedness of learning problems. That is, since the number of parameters is large compared to the number of training samples, a solution is likely to fit the training samples well, but have poor generalization performance on new unseen data. The statistical learning theory proposed by Vapnik and Chervonenkis [97] addresses this issue. In the context of a binary classification problem, they show that generalization error depends on the *capacity* of the hypothesis space, rather than the dimension of the problem, and that the capacity of linear discriminants can be controlled by maximizing the margin of the hyperplane with respect to training samples. Therefore kernels can safely be used to learn complex patterns with good generalization performance. The support vector machine (SVM) method employs this insight to learn a linear discriminant that maximizes margin while minimizing empirical error. Indeed, the success of SVMs in practical problems, backed by theoretical foundation, quickly led to its wide-spread popularity and revitalized interest in kernel methods.[1]

SVM model learning is formulated as a convex quadratic optimization problem with linear constraints. Traditionally, in the primal formulation, the problem is stated in terms of the parameters or weights of a linear discriminant function. The Lagrange dual formulation leads to a problem (and solution) that can be expressed entirely in terms of dot products, which can be replaced by kernel evaluations. This traditional perspective has led to the common misconception that in order to solve kernel SVMs one must resort to solving the dual optimization problem. However, recently Chapelle [25] shows that kernel SVMs can be solved using a primal formulation just as effectively.

In this thesis, we investigate two challenges associated with kernel based classification and illustrate how a primal approach can lead to effective and efficient solutions. The two problems are briefly described below. Background and relevant literature for each problem is reviewed in detail in Chapters 3 and 7, respectively, where they are introduced.

[1] Since Vapnik and Chervonenkis' result, other generalization results based on Bayesian statistics, compression schemes and stability analysis have also been posited to explain the performance of support vector machines and other kernel methods.

1. **Large-Scale Rare Class Learning:** Rare class problems are characterized by highly unbalanced class distributions. Unbalanced datasets introduce significant bias in standard classification methods. Also, due to the increase of data in recent years, large datasets with millions of observations have become commonplace. We propose a primal solution to address both the problem of bias and computational complexity in rare class problems by optimizing area under the receiver operating characteristic curve and by using a rare class only kernel representation, respectively. We justify the approach theoretically and experimentally.

2. **Feature Selection:** Feature selection refers to the process of selecting an optimal subset of input features to improve generalization error and model interpretability, by discarding irrelevant or redundant inputs. We develop an embedded feature selection method for kernel support vector machines based on a primal formulation. We propose an effective and efficient solution to solve the resulting non-convex problem using second-order optimization techniques and illustrate how our approach improves upon state-of-the-art methods.

1.1 Contributions

The contributions with respect to large-scale rare class learning are summarized below.

1. We propose to maximize area under curve (AUC) of the receiver operator characteristic instead of minimizing empirical error in the support vector machine formulation for unbalanced problems. We argue that the AUC is a more appropriate empirical loss function for rare class problem than classification error. This results in a regularized biclass ranking problem, which is a special case of RankSVM [54]. RankSVM has generally been used in the context of ranking (e.g web page ranking) with linear models. Its application to nonlinear rare class learning has not been highlighted in previous literature.

2. To solve the dual optimization problem for kernel RankSVM requires $O(m^6)$ time and $O(m^4)$ space, where m is the number of data samples. Chapelle and Keerthi [26] show a primal approach can be used to solve RankSVM in $O(m^3)$ time and $O(m^2)$ space. We propose a modification to kernel RankSVM, that takes specific advantage of the unbalanced class distirbution, to achieve $O(mm_+)$ time and $O(mm_+)$

space, where m_+ is the number of rare class examples. The idea is inspired by Zhu et al. [114], in which the posterior probability density is estimated with an adaptive bandwidth kernel density estimator over rare class samples and locally adjusted by the density of the background class. Using similar assumptions, we show the optimal solution can be approximately expressed as a linear combination of rare class kernel functions. In contrast to Zhu et al. [114], we use a regularized loss minimization approach to minimize a ranking loss objective while restricting the solution to a linear combination of rare class kernel evaluations. Additionally, in our method, the kernel does not need to be kernel density estimator, but can represent an arbitrary Mercer kernel. We call this method RankRC, since it enforces a Rare Class solution.

3. We view kernel RankRC as an approximation to kernel RankSVM, and mathematically investigate the quality of approximation. Specifically, we establish an upper bound on the difference between the optimal hypotheses of RankSVM and RankRC. This bound is established by observing that any L_2-regularized loss minimization problem, with a hypothesis restricted to an arbitrary subset of points, is equivalent to the L_2-regularized loss minimization problem with a hypothesis instantiated by the full data set but using the orthogonal projection of the original feature mapping.

4. We show that the upper bound suggests that, under the assumption that a hypothesis is instantiated by any subset of data points of a fixed cardinality, it is optimal to choose data points from the rare class first. Hence, this bound provides additional theoretical justification for RankRC.

5. We demonstrate that the Nyström kernel approximation method is equivalent to solving a kernel regularized loss problem instantiated by a subset of data points corresponding to the selected columns. Consequently, theoretical bounds for Nyström kernel approximation methods can be established based on perturbation analysis of the orthogonal projection in the feature mapping. We demonstrate that this approach provides tighter bounds in comparison to established bounds based on the perturbation of the kernel matrix [32], which can be arbitrarily large depending on the condition number of the kernel approximation matrix.

6. We empirically compare RankRC to other methods on several datasets and illustrate predictive and computational advantages.

7. We extend the biclass RankRC formulation to multi-level ranking and apply it to a recent competition problem sponsored by the Heritage Health Provider Competition. The problem illustrates how RankRC can be used for ordinal regression where one ordinal level contains the vast majority of examples. We compare performance of RankRC with other methods and demonstrate computational and predictive advantages.

The contributions listed above also appear in Tayal et al. [90] and Tayal et al. [91].

The contributions with respect to feature selection are summarized below:

1. We invoke the Representer Theorem to formulate a primal embedded feature selection SVM problem and use a smoothed hinge loss function to obtain a simpler bound constrained problem. We solve the resulting non-convex problem using a generalized trust-region algorithm for bound constrained minimization.

2. To improve efficiency we propose a two-block alternating optimization scheme, in which we iteratively solve (a) the standard SVM problem and (b) a smaller non-convex feature selection problem. Importantly, we propose a novel alternate optimization method by sharing a single perspective variable. We establish mathematical conditions under which this perspective variable sharing AO method avoids saddle points. For SVM feature selection, the perspective variable explicitly represents the margin. We provide computational evidence to illustrate that this helps avoid suboptimal local solutions. Moreover, by focussing on maximizing margin in the feature selection problem—a critical quantity for generalization error—we are able to further improve solution quality.

3. We compare our methods to generalized multiple kernel learning and other leading nonlinear feature selectors, and show that our approach improves results.

The contributions to the feature selection problem also appear in Tayal et al. [92].

1.2 Outline

The remainder of the thesis is organized as follows. Chapter 2 reviews support vector machines and kernels. Chapters 3 to 6 develop RankRC for large-scale rare class learning:

Chapter 3 introduces the RankRC model, Chapter 4 presents analytical results for RankRC, Chapter 5 shows the empirical results, and Chapter 6 extends the RankRC to multi-class ranking. Chapter 7 develops the primal feature selection method for kernel support vector machines. We conclude in Chapter 8 with summary remarks and potential extensions.

Chapter 2

Background

In this chapter, we briefly review support vector machines (SVMs), kernel induced feature spaces, and the primal optimization method for kernel SVM based on the Representer theorem. The chapter largely draws from material in Cristianini and Shawe-Taylor [35], Hastie et al. [52], Schölkopf et al. [82], Schölkopf and Smola [83], Vapnik [97].

2.1 Support Vector Classification

Support vector machines learn a linear discriminant rule (hyperplane) that separates two classes of data instances. This is known as the binary classification problem. Consider a set of m training examples, $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_m, y_m)\}$, where $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$ are data instances and $y_i \in \{+1, -1\}$ are corresponding class labels (without loss of generality). We seek a linear discriminant rule, $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$, such that the class label of an observation $\mathbf{x}$ is given by $\hat{y}_i = \text{sign}(f(\mathbf{x}))$. In other words, we find a hyperplane in d-dimensional space, $\mathbf{w}^T\mathbf{x} + b = 0$, that separates the space into two half spaces, each corresponding to one of the class labels.

2.1.1 Linearly Separable Data

If the data is linearly separable, then there are an infinite number of linear discriminants that can separate the two classes (e.g see Figure 2.1a-c). According to the statistical learning theory of Vapnik [97], the optimal linear discriminant is the one that maximizes geometric margin between the two classes (Figure 2.1d), since it minimizes a bound on generalization error irrespective of the dimensionality of the space. Mathematically, this can be stated as a

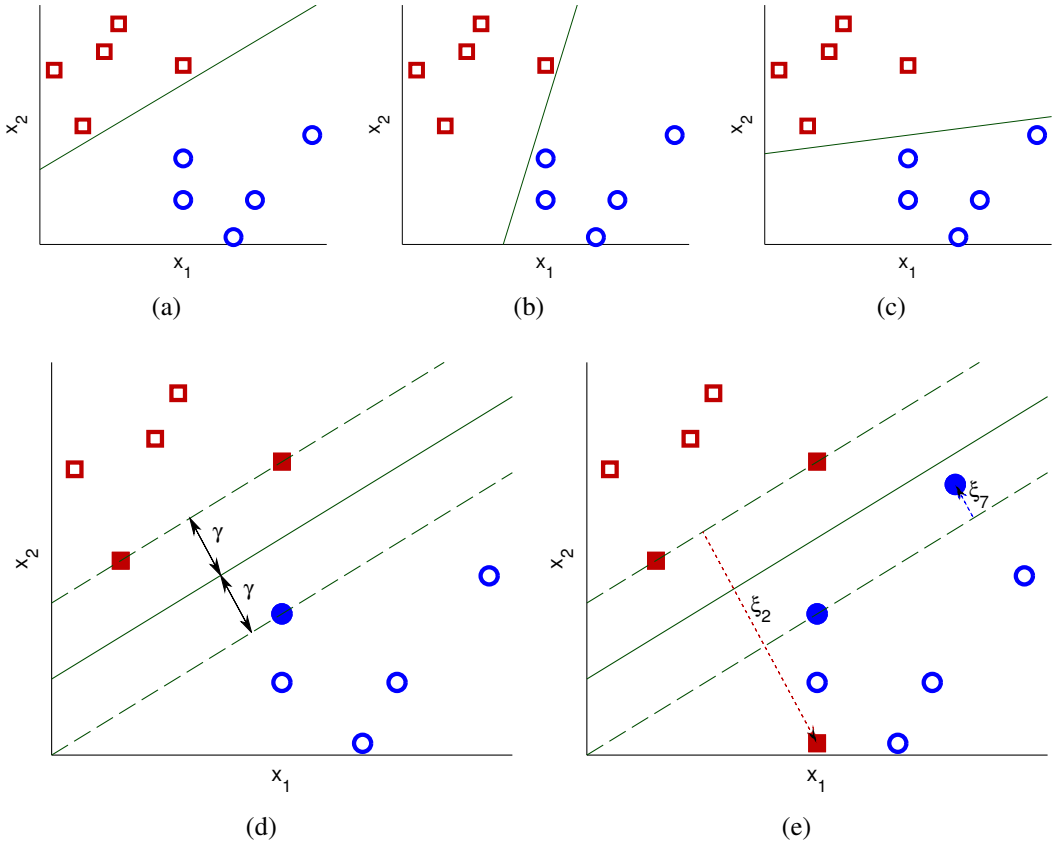


Figure 2.1: Example dataset in $(x_1, x_2) \in \mathbb{R}^2$ space consisting of two classes with (red) squares representing one class and (blue) circles representing the other class. (a)-(c) In the separable case, there are many (infinite) linear discriminants which can separate the two classes perfectly. Three such discriminants are shown. (d) According to generalization bounds, the optimal linear discriminant is the one that maximizes the geometric margin, γ, between the two classes. (e) For inseparable or noisy datasets, we allow points to cross the margin, e.g. $\mathbf{x}_2$ and $\mathbf{x}_7$, and penalize the violations, ξ_2 and ξ_7. In (d) and (e), support vectors are shown as filled in circles.

convex optimization problem. Note, the hyperplane associated with $(\mathbf{w}, b)$ does not change upon rescaling to $(\lambda\mathbf{w}, \lambda b)$, for $\lambda \in \mathbb{R}_+$. The scaling affects the margin as measured by the function output as opposed to the geometric margin. The absolute value of function output at the closest point is called the functional margin. We can optimize the geometric margin by fixing the functional margin to 1 and minimizing the norm of the weight vector.

Specifically, setting functional margin to 1 implies,

$$\mathbf{w}^T\mathbf{x}_+ + b = 1$$
$$\mathbf{w}^T\mathbf{x}_- + b = -1$$

where $\mathbf{x}_+$ and $\mathbf{x}_-$ are the closest points on the positive and negative sides of the hyperplane defined by $(\mathbf{w}, b)$. The geometric margin, γ, is then given by

$$\begin{aligned}\gamma &= \frac{1}{2}\left(\frac{\mathbf{w}^T\mathbf{x}_+ + b}{\|\mathbf{w}\|_2} - \frac{\mathbf{w}^T\mathbf{x}_- + b}{\|\mathbf{w}\|_2}\right) \\ &= \frac{1}{2\|\mathbf{w}\|_2}\left((\mathbf{w}^T\mathbf{x}_+ + b) - (\mathbf{w}^T\mathbf{x}_- + b)\right) \\ &= \frac{1}{\|\mathbf{w}\|_2}\end{aligned}$$

Thus maximizing margin is equivalent to minimizing the norm of the weight vector. The convex optimization problem that solves for the maximum margin discriminant is,

$$\begin{aligned}\min_{\mathbf{w},b} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 \\ \text{subject to} \quad & y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1, \quad i = 1, \ldots, m\,,\end{aligned}$$

where the constraint enforces a functional margin of 1 on both sides of the hyperplane.

2.1.2 Inseparable Data

Real data is often noisy and there may in general be no linear discriminant that can separate the data (see Figure 2.1e). To handle this case SVM uses penalized slack variables, ξ_i, $i = 1, \ldots, m$, which allow margin constraints to be violated:

$$\begin{aligned}\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{m}\xi_i \\ \text{subject to} \quad & y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \xi_i, \quad i = 1, \ldots, m \\ & \xi_i \geq 0, \quad i = 1, \ldots, m\,.\end{aligned} \tag{2.1}$$

Here, C is a penalty parameter which balances the tradeoff between margin and violations. In practice, it is usually determined by cross-validation.

2.1.3 Dual Formulation

We can transform Problem (2.1) into its corresponding Lagrange dual problem. Since the optimization problem is a convex quadratic programming problem, there is no duality gap. Therefore, an optimal solution of the primal problem is given by the dual problem. The dual formulations provides additional insight into the SVM problem. Also, it leads directly to a kernel approach to solve the problem in an implied feature space.

The Lagrangian for Problem (2.1) is

$$L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \mathbf{r}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i\left(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \mathbf{x}_i\right) - \sum_{i=1}^{m} r_i\xi_i \,,$$

with $\alpha_i \geq 0$ and $r_i \geq 0$ for dual feasibility. Stationarity implies,

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{m} y_i\alpha_i\mathbf{x}_i = \mathbf{0} \,, \qquad \frac{\partial L}{\partial \xi_i} = C - \alpha_i - r - i = 0 \,, \qquad \frac{\partial L}{\partial b} = \sum_{i=1}^{m} y_i\alpha_i = 0 \,.$$

Substituting these relations into the Lagrangian, we obtain the following dual objective function:

$$L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \mathbf{r}) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T\mathbf{x}_j \,.$$

The constraint $C - \alpha_i - r_i = 0$ together with $r_i \geq 0$ implies $\alpha_i \leq C$. Therefore, the dual optimization problem, which maximizes L or equivalently minimizes $-L$, is given by,

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T\mathbf{x}_j - \sum_{i=1}^{m}\alpha_i \\ \text{subject to} \quad & \sum_{i=1}^{m} y_i\alpha_i = 0 \\ & 0 \leq \alpha_i \leq C, \quad i = 1, ..., m \,, \end{aligned} \tag{2.2}$$

where the constraints enforce stationarity conditions and dual feasibility. The Karush-Kuhn-Tucker (KKT) complementarity conditions provide useful information about the

structure of the solution. These conditions state that a solution, $(\mathbf{w}^*, b^*, \boldsymbol{\xi}^*, \boldsymbol{\alpha}^*)$ must satisfy,

$$\alpha_i^* \left(y_i({\mathbf{w}^*}^T \mathbf{x}_i + b^*) - 1 + \xi_i^* \right) = 0, \quad i = 1, ..., m$$
$$\xi_i^* \left(\alpha_i^* - C \right) = 0, \quad i = 1, ..., m \, .$$

This implies a non-zero slack variable, $\xi_i^* \neq 0$, can only occur when $\alpha_i^* = C$. These points are the margin violations, as their functional margin is less than 1, and their geometric margin is less than $1/\|\mathbf{w}\|_2$. Points where $0 < \alpha_i^* < C$ imply, $\xi_i^* = 0$, and $y_i({\mathbf{w}^*}^T \mathbf{x}_i + b^*) = 1$. These points lie at the margin, with a geometric distance of $1/\|\mathbf{w}\|_2$ from the hyperplane. Points corresponding to $\alpha_i^* = 0$ lie beyond the margin in the correct half-space.

From the first stationary condition, we have $\mathbf{w} = \sum_{i=1}^m y_i \alpha_i \mathbf{x}_i$. Thus, the optimal hypothesis can be expressed in the dual representation as,

$$f(\mathbf{x}) = {\mathbf{w}^*}^T \mathbf{x} + b = \sum_{i=1}^m y_i \alpha_i^* \mathbf{x}_i^T \mathbf{x} + b^* \, . \tag{2.3}$$

The value of b^* can be determined by solving $y_i f(\mathbf{x}_i) = 1$ for any i with $0 < \alpha_i^* < C$, due to the complementarity conditions. Points corresponding to non-zero values of α_i^* are called support vectors, since in the expression (2.3) only these points are involved. Note, slight perturbations of points that are not support vectors will not affect the solution.

2.2 Kernel Induced Feature Spaces

So far we have considered linear hypothesis functions only. Real-world applications often require more expressive hypothesis spaces. That is, the target hypothesis function cannot be expressed as a simple linear combination of the input variables, but may require abstract features of the data to be exploited. For example, consider Newton's law of gravitation as the target function, $f(m_1, m_2, r) = G m_1 m_2 / r^2$, in terms of masses m_1, m_2 and distance, r. A linear hypothesis in terms of m_1, m_2, r cannot represent f, but a change of space obtained by mapping input features, $(m_1, m_2, r) \mapsto (x, y, z) = (\ln m_1, \ln m_2, \ln r)$, gives the representation $g(x, y, z) = \ln f(m_1, m_2, r) = \ln C + \ln m_1, + \ln m_2 - 2 \ln r = c + x + y - 2z$, which can be learned using a linear function.

Kernels allow us to implicitly work in a high-dimensional derived feature space. The larger the set of mapped features, the more likely, the function to be learned can be repre-

sented. Traditionally, a large number of derived features is known to degrade generalization performance, an effect popularly known as the curse of dimensionality. However, SVMs can avoid such degradation since generalization performance depends on the geometric margin of separation, and not the dimensionality of the feature space *per se*. Moreover, a large number of derived features would pose computational challenges, but since kernels bypass the need to compute the explicit feature map, high dimensional feature spaces (with even infinite dimensions) can be used, which otherwise would be computationally intractable.

Definition 1. *A* feature map *is a vector function,* $\phi : \mathcal{X} \subseteq \mathbb{R}^d \to \mathcal{F} \subseteq \mathbb{R}^{d'}$*, which maps the input space* $\mathcal{X}$ *to a derived feature space,* $\mathcal{F}$*, i.e.*

$$\mathbf{x} = (x_1, ..., x_d) \mapsto \phi(\mathbf{x}) = (\phi_1(\mathbf{x}), ..., \phi_{d'}(\mathbf{x})) \ ,$$

where $\mathcal{F} = \{\phi(\mathbf{x}) | \mathbf{x} \in \mathcal{X}\}$.

A feature map is used to transform the data to a new space, in which a linear function is learned. The hypothesis space is $f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$, which can be a non-linear function of the input features $\mathbf{x}$.

Definition 2. *A* kernel *is a function,* $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$*, such that for all* $\mathbf{u}, \mathbf{v} \in \mathcal{X}$*,*

$$k(\mathbf{u}, \mathbf{v}) = \phi(\mathbf{u})^T \phi(\mathbf{v}) \ ,$$

where $\phi : \mathcal{X} \to \mathcal{F}$ *is a feature map.*

A kernel corresponds to the dot product in a derived feature space. In many cases, the dot product can be computed more efficiently using a simple function than by directly computing the dot product after explicitly forming the feature vectors. A few examples of common kernels are shown in Table 2.1.

Note, in the dual SVM problem (2.2) and the solution (2.3) all occurrences of data instances occur in a dot product. Therefore, we can replace these dot products with the kernel to implicitly compute the dot product in a derived feature space. This is known as

Kernel	$k(\mathbf{u}, \mathbf{v})$
Linear	$\mathbf{u}^T\mathbf{v}$
Degree-p polynomial	$(\mathbf{u}^T\mathbf{v}+b)^p$
Gaussian	$\exp\left(-\frac{\|\mathbf{u}-\mathbf{v}\|_2^2}{2\sigma^2}\right)$
Histogram intersection	$\sum_{i=1}^{d}\min\left(\|u_i\|^\alpha, \|v_i\|^\beta\right)$
Spline	$\prod_{i=1}^{d} 1+u_iv_i+u_iv_i\min(u_i,v_i)-\frac{u_i+v_i}{2}\min(u_i,v_i)^2+\frac{\min(u_i,v_i)^3}{3}$
Wave	$\frac{\theta}{\|\mathbf{u}-\mathbf{v}\|_2}\sin\frac{\|\mathbf{u}-\mathbf{v}\|_2}{\theta}$

Table 2.1: Examples of kernels, $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. The Gaussian kernel is a popular non-linear kernel, often used as a default in absence of expert knowledge about the data. It corresponds to an infinite dimensional feature space and is in the class of universal kernels, i.e. it can approximate an arbitrary continuous target function uniformly, thereby minimizing both estimation and approximation errors [70].

the "kernel trick". The resulting problem is,

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\sum_{i,j=1}^{m} y_iy_j\alpha_i\alpha_jk(\mathbf{x}_i,\mathbf{x}_j)-\sum_{i=1}^{m}\alpha_i \\ \text{subject to} \quad & \sum_{i=1}^{m} y_i\alpha_i = 0 \\ & 0 \le \alpha_i \le C, \quad i = 1,...,m\,, \end{aligned} \tag{2.4}$$

with the solution,

$$f(\mathbf{x}) = \sum_{i=1}^{m} y_i\alpha_i^*k(\mathbf{x}_i,\mathbf{x})+b^*\,. \tag{2.5}$$

The benefit of a kernel is that we do not need to know the underlying feature map in order to compute the dot product. A kernel is usually defined directly as a function, hence implicitly defining the feature space. This way we avoid the feature space not only in the computation of the dot product, but also in the design of the learning machine. The kernel can be viewed as a similarity measure between two points in the input space, $\mathcal{X}$.

Consequently, defining a kernel for an input space can be more natural than designing a complex feature space.

2.2.1 Characterization

Mercer's condition [68] characterizes what constitutes a valid kernel according to Definition 2. That is, Mercer's theorem gives necessary and sufficient conditions for a continuous symmetric function k to admit an inner product representation in some feature space. We will not go into details of the analysis but simply quote the theorem below. In this thesis, we use the term kernel to refer to functions satisfying Mercer's condition, though in the literature these are sometimes qualified as Mercer kernels.

Theorem 1. *(Mercer) If k is a continuous kernel of a positive definite integral operator on $L_2(\mathcal{X})$, where $\mathcal{X}$ is some compact space and $L_2(\mathcal{X})$ denotes the space of square-integrable functions, that is,*

$$\int_{\mathcal{X}\times\mathcal{X}} k(\mathbf{u},\mathbf{v})f(\mathbf{x})f(\mathbf{v})d\mathbf{x}d\mathbf{v} \geq 0 \,,$$

for all $f \in L_2(\mathcal{X})$, then it can be expanded as

$$k(\mathbf{u},\mathbf{v}) = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{u})\phi_i(\mathbf{v}) \,,$$

using eigenfunctions $\phi_i \in L_2(\mathcal{X})$ and eigenvalues $\lambda_i \geq 0$.

In the dual SVM problem (2.4) we see the only information used is kernel evaluations on pairwise data in the training set. This can be stored in a $m \times m$ matrix, referred to as the kernel matrix.

Definition 3. *Given a kernel k and inputs $X = \{\mathbf{x}_1, ..., \mathbf{x}_m\} \in \mathcal{X}^m$, the $m \times m$ matrix*

$$K = [k(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1}^{m} \,,$$

is called the kernel matrix *(or* Gram matrix*) of the kernel k for the finite set X.*

The conditions for Mercer's theorem are equivalent to requiring that for any finite subset of $\mathcal{X}$, the corresponding kernel matrix is positive semi-definite [e.g. see 35] . This provides an alternative characterization of a kernel that is often more useful in practice.

Proposition 2. *Let $\mathcal{X}$ be a non-empty set with $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a symmetric function on $\mathcal{X}$. Then k is a Mercer kernel if and only if the kernel matrix $K = [k(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1}^m$ is positive semi-definite (has non-negative eigenvalues) for all $m \in \mathbb{N}$, $\mathbf{x}_i \in \mathcal{X}, i = 1, ..., m$.*

2.2.2 Reproducing Kernel Hilbert Space (RKHS)

The feature map, ϕ, can also be defined as a map from $\mathcal{X}$ into the space of functions mapping $\mathcal{X}$ into $\mathbb{R}$, denoted as $\mathbb{R}^{\mathcal{X}}$,

$$\phi : \mathcal{X} \to \mathbb{R}^{\mathcal{X}} , \qquad \mathbf{x} \overset{\phi}{\mapsto} k(\cdot, \mathbf{x}).$$

Here $\phi(\mathbf{x})(\cdot) = k(\cdot, \mathbf{x})$ represents the function that assigns the value $k(\mathbf{x}', \mathbf{x})$ for any given point $\mathbf{x}' \in \mathcal{X}$.[1] Following [83], we constructively build the Hilbert space of functions. This is done by defining a vector space of functions by taking linear combinations of the form

$$f(\cdot) = \sum_{i=1}^{m} a_i k(\cdot, \mathbf{x}_i) , \quad g(\cdot) = \sum_{j=1}^{m'} b_j k(\cdot, \mathbf{x}'_j) , \tag{2.6}$$

where $m, m' \in \mathbb{N}, a_1, ..., a_m, b_1, ..., b_{m'} \in \mathbb{R}$ and $\mathbf{x}_1, ..., \mathbf{x}_m, \mathbf{x}'_1, ..., \mathbf{x}'_{m'} \in \mathcal{X}$ are arbitrary. A dot product between f and g can be defined as

$$\langle f, g \rangle = \sum_{i=1}^{m} \sum_{j=1}^{m'} a_i b_j k(\mathbf{x}_i, \mathbf{x}'_j) , \tag{2.7}$$

which can be verified as a well-defined dot product on the vector space of functions given by (2.6). It can be verified that the dot product of f with the function $k(\cdot, \mathbf{x})$ recovers $f(\mathbf{x})$, i.e. $\langle f, k(\cdot, \mathbf{x}) \rangle = f(\mathbf{x})$, which is known as the reproducing property of the kernel k. The space of functions (2.6) can be completed in the norm corresponding to the dot product (2.7) to obtain a Hilbert space $\mathcal{H}$, called a reproducing kernel Hilbert space (RKHS). Thus,

[1] Note, viewing $\phi(\mathbf{x})$ as a function is compatible with the feature map perspective given in Definition 1, since a vector can represent coefficients of a function in a given feature space. For example, consider the feature map $\phi : \mathbb{R}^2 \mapsto \mathbb{R}^3$ defined by $\mathbf{x} = [x_1, x_2]^T \mapsto \phi(\mathbf{x}) = [x_1, x_2, x_1 x_2]^T$, with kernel $k(\mathbf{u}, \mathbf{v}) = [u_1, u_2, u_1 u_2][v_1, v_1, v_1, v_2]^T$. We can define a function of the features as $f(\mathbf{x}) = ax_1 + bx_2 + cx_1 x_2$, where $f : \mathcal{X} = \mathbb{R}^2 \to \mathbb{R}$, and define an equivalent representation for $f(\cdot) = [a, b, c]^T$ with $f(\mathbf{x}) = f(\cdot)^T \phi(\mathbf{x})$. The notation $f(\cdot)$ or simply f refers to the function itself, in the abstract, which may have multiple equivalent representations. Thus, while $\phi(\mathbf{x})$ can be defined as a mapping from $\mathbb{R}^2$ to $\mathbb{R}^3$, it can also be seen to define the parameters of a function that maps $\mathbb{R}^2$ to $\mathbb{R}$.

one can define a RKHS as a Hilbert space $\mathcal{H}$ of functions on a set $\mathcal{X}$ with the property that, for all $\mathbf{x} \in \mathcal{X}$ and $f \in \mathcal{H}$, the point evaluations $f \mapsto f(\mathbf{x})$ are continuous linear functionals and all points $f(\mathbf{x})$ are well defined. The Moore-Aronszajn theorem [8] states that for every Mercer kernel, k, there exists a unique RKHS and vice versa.

2.3 Representer Theorem and Training in the Primal

It is often presumed that to solve SVM with a kernel, we must solve the dual optimization problem (2.4). However, the primal SVM problem also admits the use of a kernel representation. More generally, a large class of kernel problems can be written in the primal form using the Representer Theorem [82], without resorting to the dual optimization problem.

Theorem 3. *(Representer Theorem) The solution to the following regularized loss minimization problem,*

$$\min_{\substack{\widetilde{f} \in \mathcal{H} \\ h \in span\{\psi_p\}}} \quad L(f(\mathbf{x}_1), ..., f(\mathbf{x}_m)) + \Omega(\|\widetilde{f}\|_{\mathcal{H}}) \,, \tag{2.8}$$

where $f := \widetilde{f} + h$, $\mathcal{H}$ is a RKHS associated with kernel k, i.e.

$$\mathcal{H} = \left\{ \widetilde{f} \in \mathbb{R}^{\mathcal{X}} \,|\, \widetilde{f}(\cdot) = \sum_{i=1}^{\infty} \beta_i k(\cdot, z_i), \beta_i \in \mathbb{R}, z_i \in \mathcal{X}, \|\widetilde{f}\|_{\mathcal{H}} < \infty \right\} \,,$$

$\{\psi_p\}_{p=1}^{M}$ is a set of M real-valued functions on $\mathcal{X}$, with the property that the $m \times M$ matrix $[\psi_p(\mathbf{x}_i)]_{ip}$ has rank M, $L : \mathbb{R}^m \to \mathbb{R} \cup \{\infty\}$ is an arbitrary cost function, and Ω is a strictly monotonically increasing function on $[0, \infty]$, with $\|\widetilde{f}\|_{\mathcal{H}}^2 = \langle \widetilde{f}, \widetilde{f} \rangle$ defined by (2.7), admits a representation of the form

$$f(\cdot) = \sum_{i=1}^{m} \beta_i k(\cdot, \mathbf{x}_i) + \sum_{p=1}^{M} b_p \psi_p(\cdot) \,,$$

with unique coefficients $b_p \in \mathbb{R}$, for all $p = 1, ..., M$.

Note the hypothesis space in (2.8) also allows M parametric basis functions on the input space, i.e. $\{\psi_p\}_{p=1}^{M}$. This is useful, for example to include an offset term, as seen in SVM solution (2.5), i.e. setting $M = 1$ and $\psi_1(\mathbf{x}) = 1$ for all $\mathbf{x}$, implies $h \in \mathbb{R}$, which represents a scalar offset to $\widetilde{f}$.

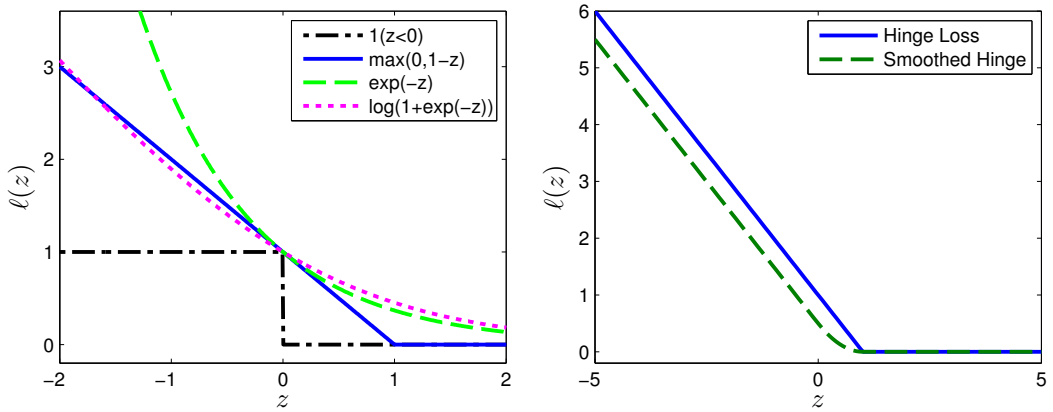
The SVM problem (2.4) is a special case of (2.8), since it can be expressed in a primal exact-penalty form as:

$$\min_{\substack{\widetilde{f}\in\mathcal{H}\\ b\in\mathbb{R}}} \quad C\sum_{i=1}^{m}\ell_h(y_i f(\mathbf{x}_i))+\frac{1}{2}\|\widetilde{f}\|_{\mathcal{H}}^2\,, \tag{2.9}$$

where $f(\mathbf{x}) = \widetilde{f}(\mathbf{x})+b$ and $\ell_h(z) = \max(0,1-z)$ measures margin violations and is known as the hinge loss function.[2] From the Representer theorem we know the solution of (2.9) has the form $f(\mathbf{x}) = \sum_{i=1}^{m}\beta_i k(\mathbf{x},\mathbf{x}_i)+b$, which also matches form (2.5) obtained using a dual optimization argument. Finally, we can substitute this form in (2.9) to obtain a primal optimization problem in terms of the unknown model variables $(\boldsymbol{\beta}, b)\in\mathbb{R}^m\times\mathbb{R}$.

Apart from the hinge loss, other loss functions can also be used for classification purposes, for example see Figure 2.2a. According to the Representer Theorem, regardless of the loss function, the solution can be expressed in the form $f(\mathbf{x}) = \sum_{i=1}^{m}\beta_i k(\mathbf{x},\mathbf{x}_i)+b$. Note, the loss functions are all convex approximations of the 0-1 loss function. The hinge-loss is robust to outliers and does not penalize correctly classified points, resulting in a sparse solution. Since the hinge-loss is non-differentiable, it can pose computational difficulties

[2]Consider replacing $f(\mathbf{x})\equiv\mathbf{w}^T\mathbf{x}+b$ in (2.1), and eliminating ξ_i's by incorporating the constraint directly into the objective as a loss function.



(a) (b)

Figure 2.2: (a) Shows different loss functions that are a convex approximation of 0-1 loss for classification problems. (b) The smoothed hinge is a differentiable approximation of the hinge loss. Here the smoothed hinge is shown with $\epsilon = 0.5$

in solving (2.9) using standard unconstrained optimization algorithms. Chapelle [25] proposes to use a differentiable approximation to the hinge loss (see Figure 2.2b),

$$\ell_\epsilon(z) = \begin{cases} (1-\epsilon)-z & \text{if } z < 1-2\epsilon \\ \frac{1}{4\epsilon}(1-z)^2 & \text{if } 1-2\epsilon \leq z < 1 \\ 0 & \text{if } z \geq 1\ , \end{cases}$$

in place of ℓ_h, in (2.9) and uses Newton's method to solve the resulting optimization problem. Note, we can opt to use a twice-differentiable smoothed function as well, however in practice, this is not necessary since the overall objective is sufficiently smooth. From a classification perspective, the smoothed hinge loss function is margin-maximizing [79] and Bayes-risk consistent [72], and offers similar benefits as the hinge loss.

Chapter 3

RankRC: Large-scale Nonlinear Rare Class Ranking

In this chapter, we introduce a new kernel based learning method for rare class problems called RankRC. Rare class problems are characterized by a highly unbalanced class distribution. In these situations, standard classification algorithms lead to biased models, since they focus on overall classification accuracy. In addition, many real-world rare class applications involve large datasets, which are prohibitive for kernel methods.

RankRC addresses both the problem of bias and computational complexity for rare class problems by optimizing area under the receiver operating characteristic curve and by using a rare class only kernel representation, respectively. This chapter motivates and develops the problem formulation and optimization algorithm for RankRC.

3.1 Introduction

In many classification problems samples from one class are extremely rare (the minority class), while the number of samples belonging to the other class are plenty (the majority class). This situation is known as the rare class problem. It is also referred to as an unbalanced or skewed class distribution problem. Rare class problems naturally arise in several application domains, for example, fraud detection, customer churn, intrusion detection, fault detection, credit default, insurance risk and medical diagnosis.

Standard classification methods perform poorly when dealing with unbalanced data, e.g. support vector machines (SVM) [55, 78, 107], decision trees [11, 29, 55, 102], neural

networks [55], Bayesian networks [39], and nearest neighbor methods [11, 109]. Most classification algorithms are driven by accuracy (i.e. minimizing error). Since minority examples constitute a small proportion of the data, they have little impact on accuracy or total error. Thus majority examples overshadow the minority class, resulting in models that are heavily biased in recognizing the majority class. Implicitly, errors from different classes are assumed to have the same costs, which is usually not true. In most problems, incorrect classification of the rare class is more expensive, for instance, diagnosing a malignant tumor as benign has more severe consequences than the contrary case.

Solutions to the class imbalance problem have been proposed at both the data and algorithm level. At the data level, various resampling techniques are used to balance class distribution, including random under-sampling of majority class instances [62], over-sampling minority class instances with new synthetic data generation [28], and focused resampling, in which samples are chosen based on additional criteria [109]. Although sampling approaches have achieved success in some applications, they are known to have drawbacks, for instance under-sampling can eliminate useful information, while over-sampling can result in overfitting. At the algorithm level, solutions are proposed by adjusting the algorithm itself. This usually involves adjusting the costs of the classes to counter the class imbalance [24, 65, 96] or adjusting the decision threshold [58]. However, true error costs are often unknown and using an inaccurate cost model can lead to additional bias.

We focus on nonlinear kernel based classification methods expressed as a regularized loss minimization problem. In recent years, we have also seen a rapid increase of data, resulting in many large scale rare class problems. For example, detecting unauthorized use of a credit card from millions of transactions. Processing large datasets can be prohibitive for many nonlinear kernel algorithms, which scale quadratically to cubically in the number of examples and may require quadratic space as well.

To address the challenges associated with rare class problems and large scale learning we propose the following:

1. Instead of maximizing accuracy (minimizing error), we optimize area under curve (AUC) of the receiver operator characteristic. The AUC overcomes inadequacies of accuracy for unbalanced problems and provides a skew independent measure. It is often used as the evaluation metric for unbalanced problems and therefore it is appropriate to directly optimize it in the training process. This results in a regularized biclass ranking problem, which is a special case of RankSVM with two ordinal levels [54].

2. To solve a kernel RankSVM problem in the dual, as originally proposed in [54], requires $O(m^6)$ time and $O(m^4)$ space, where m is the number of data samples. Recently, Chapelle and Keerthi [26] proposed a primal approach to solve RankSVM, which results in $O(m^3)$ time and $O(m^2)$ space for nonlinear kernels. We propose a modification to kernel RankSVM, that takes specific advantage of the unbalanced nature of the problem, to achieve $O(mm_+)$ time and $O(mm_+)$ space, where m_+ is the number of rare class examples. The idea is inspired by Zhu et al. [114], in which the posterior probability density is estimated with an adaptive bandwidth kernel density estimator over rare class samples and locally adjusted by the density of the background class. Using similar assumptions, we show the solution can be approximately expressed as a linear combination of rare class kernel functions. In contrast to Zhu et al. [114], we use a regularized loss minimization approach to minimize a ranking loss objective, but restrict the solution to a linear combination of rare class kernel evaluations. In our method, the kernel does not need to be kernel density estimator, but can represent an arbitrary Mercer kernel. We call this method RankRC, since it enforces a Rare Class solution.

In this chapter, we motivate RankRC assuming certain properties of the class distribution and kernel choice. In Chapter 4 we analyze RankRC under general settings, and show that RankRC is optimal with respect to RankSVM for unbalanced datasets with a fixed cardinality of kernel functions.

The rest of the chapter is organized as follows. Sections 3.2 and 3.3 review the AUC measure and RankSVM. Section 3.4 develops the RankRC problem and presents justification for the rare class representation. Section 3.5 outlines the optimization algorithm used to solve RankRC.

3.2 ROC Curve

Evaluation metrics play an important role in learning algorithms. They provide ways to assess performance as well as guide modeling. For classification problems, error rate is the most commonly used metric. Consider the binary classification problem. Let $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_m, y_m)\}$ be a set of m training examples, where $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^d$, $y_i \in \{+1, -1\}$. Denote $f(\mathbf{x})$ as the inductive hypothesis obtained by training on example set $\mathcal{D}$.

The empirical error rate is defined as,

$$ErrorRate = \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}[f(\mathbf{x}_i) \neq y_i], \tag{3.1}$$

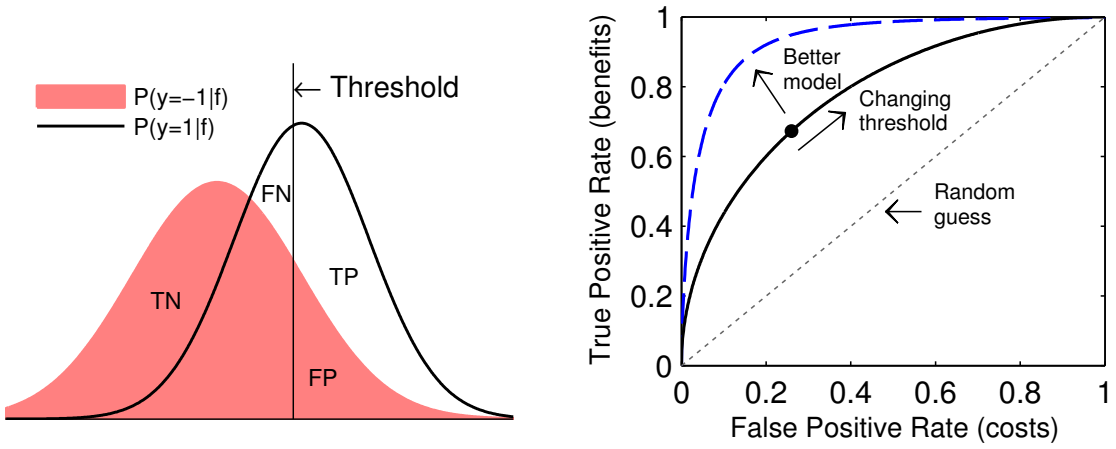
where $\mathbb{I}[p]$ denotes the indicator function and is equal to 1 if p is true, 0 if p is false. However, for highly unbalanced datasets, error rate is not appropriate since it can be biased toward the majority class [53, 66, 77, 86]. In this paper, we follow convention and set the minority class as positive and the majority class as negative. Consider a dataset that has 1 percent positive cases and 99 percent negative ones. A naive solution which assigns every example to be positive will obtain only 1 percent error rate. Indeed, classifiers that always predict the majority class can obtain lower error rates than those that predict both classes equally well. But clearly these are not useful hypotheses.

Classification performance can be represented by a confusion matrix as in Table 3.1, with m_+ denoting the number of minority examples and m_- the number of majority ones. The proportion of the two rows reflects class distribution and any performance measure that uses values from both rows will be sensitive to class skew.

The Receiver Operating Characteristic (ROC) can be used to obtain a skew independent measure [19, 69, 77]. Most classifiers intrinsically output a numerical score and a predicted label is obtained by thresholding the score. For example, a threshold of zero leads to taking the sign of the numerical output as the label. Each threshold value generates a confusion matrix with different quantities of false positives and negatives (see Figure 3.1a). The ROC graph is obtained by plotting the true positive rate (number of true positives divided by m_+) against the false positive rate (number of false positives divided by m_-) as the threshold level is varied (see Figure 3.1b). It depicts the trade-off between benefits (true positive) and costs (false positives) for different choices of the threshold. Thus it does not depend on a priori knowledge of the costs associated with misclassification. A ROC curve that

		Predicted		
		$f(\mathbf{x}) = +1$	$f(\mathbf{x}) = -1$	Total
Actual	$y = +1$	True Positives (TP)	False Negatives (FN)	m_+
	$y = -1$	False Positives (FP)	True Negatives (TN)	m_-

Table 3.1: Confusion matrix representing the results of a model for a binary classification problem.



(a) (b)

Figure 3.1: ROC analysis. (a) Different quantities of True Positives (TP), False Positives (FP), False Negatives (FN) and True Negatives (TN) are obtained as the threshold value of a model is adjusted. (b) The ROC curve plots true positive rate against false positive rate for different threshold values. The dashed (blue) ROC curve dominates the solid (black) ROC curve. The dotted (gray) ROC curve has an AUC of 0.5, indicating a model with no discriminative value.

dominates another provides a better solution at any cost point.

To facilitate comparison, it is convenient to characterize ROC curves using a single measure. The area under a ROC curve (AUC) can be used for this purpose. It is the average performance of the model across all threshold levels and corresponds to the Wilcoxon rank statistic [51]. AUC represents the probability that the score generated by a classifier places a positive class sample above a negative class sample when the positive sample is randomly drawn from the positive class and the negative sample is randomly drawn from the negative class [37]. The AUC can be computed by forming the ROC curve and using the trapezoid rule to calculate the area under the curve. Also, given the intrinsic output of a hypothesis, $f(\mathbf{x})$, we can directly compute the AUC by counting pairwise correct rankings [37]:

$$AUC = \frac{1}{m_+ m_-} \sum_{\{i:y_i=+1\}} \sum_{\{j:y_j=-1\}} \mathbb{I}\left[f(\mathbf{x}_i) \geq f(\mathbf{x}_j)\right] . \tag{3.2}$$

Incorporating the AUC in the modeling process leads to a biclass ranking problem, as discussed in the following section.

3.3 RankSVM

The modeling process can usually be expressed as an optimization problem involving a loss function and a penalty on complexity (e.g. regularization term). For most classification problems, since the performance measure is error rate, it is natural to consider minimizing the empirical error rate (3.1) as the loss function. In practice, $\mathbb{I}[\cdot]$ is often replaced with a convex approximation such as the hinge loss, logistic loss or exponential loss [10]. Specifically, using the hinge loss, $\ell_h(z) = \max(0, 1-z)$, with ℓ_2-regularization leads to the well known support vector machine (SVM) formulation [15, 97],

$$\min_{\mathbf{w}\in\mathbb{R}^d} \quad \frac{1}{m}\sum_{i=1}^{m} \ell_h\left(y_i \mathbf{w}^T \mathbf{x}_i\right) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 \,, \tag{3.3}$$

where $\lambda \in \mathbb{R}_+$ is a penalty parameter that controls model complexity. Here, the hypothesis, $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$, is assumed linear in the input space $\mathcal{X}$. Since SVMs try to minimize error rate, they can lead to ineffective class boundaries when dealing with highly skewed datasets, with resulting solutions biased toward the majority concept [107]. The literature contains several approaches to remedy this problem. Most prevalent are sampling methods and cost-sensitive learning. However, these approaches implicitly or explicitly fix the relative costs of misclassification. When the true costs are unknown, this can lead to suboptimal solutions.

Instead of minimizing error rate, we consider optimizing AUC as a natural way to deal with imbalance. Indeed, if we measure performance using AUC, it is preferable to optimize this quantity directly during the training process. In the AUC formula given in (3.2), we replace $\mathbb{I}[\cdot]$ with the hinge loss to obtain a convex ranking loss function. Thus we solve the following regularized loss minimization problem:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \quad \frac{1}{m_+ m_-} \sum_{\{i:y_i=+1\}} \sum_{\{j:y_j=-1\}} \ell_h\left(\mathbf{w}^T\mathbf{x}_i - \mathbf{w}^T\mathbf{x}_j\right) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 \,. \tag{3.4}$$

Problem (3.4) is a special case of RankSVM proposed by Herbrich et al. [54] with two ordinal levels. Like SVM, RankSVM leads to a dual problem which can be expressed in terms of dot-products between input vectors. This allows us to obtain a non-linear function through the kernel trick [15], which consists of using a kernel function, $k : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$, that corresponds to a feature map, $\phi : \mathcal{X} \to \mathcal{F} \subseteq \mathbb{R}^{d'}$, such that $\forall \mathbf{u}, \mathbf{v} \in \mathcal{X}$, $k(\mathbf{u}, \mathbf{v}) = \phi(\mathbf{u})^T\phi(\mathbf{v})$. Here, k directly computes the inner product of two vectors in a potentially high-dimensional

feature space $\mathcal{F}$, without the need to explicitly form the mapping. Consequently, we can replace all occurrences of the dot-product with k in the dual and work implicitly in space $\mathcal{F}$.

However, since there is a Lagrange multiplier for each constraint associated with the hinge loss, the dual formulation leads to a problem in $m_+m_- = O(m^2)$ variables. Assuming the optimization procedure has cubic complexity in the number of variables and quadratic space requirements, the complexity of the dual method becomes $O(m^6)$ time and $O(m^4)$ space, which is unreasonably large for even medium sized datasets.

As noted by Chapelle [25], Chapelle and Keerthi [26], we can also solve the primal problem in the implicit feature space due to the Representer Theorem [59, 82]. This theorem states that the solution of any regularized loss minimization problem in $\mathcal{F}$ can be expressed as a linear combination of kernel functions evaluated at the training samples, $k(\mathbf{x}_i, \cdot)$, $i = 1, ..., m$. In the feature space $\mathcal{F}$, problem (3.4) corresponds to solving

$$\min_{\mathbf{w}\in\mathbb{R}^{d'}} \quad \frac{1}{m_+m_-} \sum_{\{i:y_i=+1\}} \sum_{\{j:y_j=-1\}} \ell_h\left(\mathbf{w}^T\phi(\mathbf{x}_i) - \mathbf{w}^T\phi(\mathbf{x}_j)\right) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 \,, \tag{3.5}$$

where we have replaced $\mathbf{x}$ with $\phi(\mathbf{x})$ and the hypothesis $f(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x})$, is a nonlinear function of the input space $\mathcal{X}$. Problem (3.5) cannot be solved directly, since the dimensionality, d', of the feature space is usually very high (potentially infinite). Using the Representer Theorem, the solution of (3.5) in space $\mathcal{F}$ can be written as:

$$f(\mathbf{x}) = \sum_{i=1}^{m} \beta_i k(\mathbf{x}_i, \mathbf{x}) = \sum_{i=1}^{m} \beta_i \phi(\mathbf{x}_i)^T\phi(\mathbf{x}) \,, \text{ or } \mathbf{w} = \sum_{i=1}^{m} \beta_i \phi(\mathbf{x}_i) \,. \tag{3.6}$$

By substituting (3.6) in (3.5) we can express the primal problem in terms of $\boldsymbol{\beta}$:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^m} \quad \frac{1}{m_+m_-} \sum_{\{i:y_i=+1\}} \sum_{\{j:y_j=-1\}} \ell_h\left(\sum_{r=1}^{m} \beta_r k(\mathbf{x}_r, \mathbf{x}_i) - \sum_{r=1}^{m} \beta_r k(\mathbf{x}_r, \mathbf{x}_j)\right) + \frac{\lambda}{2} \sum_{i,j=1}^{m} \beta_i\beta_j k(\mathbf{x}_i, \mathbf{x}_j) \,,$$

or more simply,

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^m} \quad \frac{1}{m_+m_-}\sum_{\{i:y_i=+1\}}\sum_{\{j:y_j=-1\}} \ell_h\left(K_{i\bullet}\boldsymbol{\beta}-K_{j\bullet}\boldsymbol{\beta}\right)+\frac{\lambda}{2}\boldsymbol{\beta}^T K\boldsymbol{\beta}\,, \tag{3.7}$$

where $K\in\mathbb{R}^{m\times m}$ is the kernel matrix, $K_{ij}=k(\mathbf{x}_i,\mathbf{x}_j)$, and $K_{i\bullet}$ denotes the ith row of K. To be able to solve (3.7) using unconstrained optimization methods such as gradient descent, we require the objective to be differentiable. We replace the hinge loss, ℓ_h, with an ϵ-smoothed differentiable approximation, ℓ_ϵ, defined as,

$$\ell_\epsilon(z)=\begin{cases}(1-\epsilon)-z & \text{if } z<1-2\epsilon\\ \frac{1}{4\epsilon}(1-z)^2 & \text{if } 1-2\epsilon\le z<1\\ 0 & \text{if } z\ge 1\,,\end{cases}$$

which transitions from linear cost to zero cost using a quadratic segment (see Figure 2.2b) and provides similar benefits as the hinge loss. Thus we can solve (3.7) using standard unconstrained optimization techniques. Since there are m variables, Newton's method would, for example, take $O(m^3)$ operations to converge.

RankSVM is popular in the information retrieval community, where linear models are the norm [e.g. see 56]. For a linear model, with d-dimension input vectors, the complexity of RankSVM can be reduced to $O(md+m\log m)$ [26]. However, many rare class problems require a nonlinear function to achieve optimal results. Solving a nonlinear RankSVM requires $O(m^3)$ time and $O(m^2)$ space [26], which is not practical for mid- to large-sized datasets. We believe this complexity is, in part, the reason why nonlinear RankSVMs are not commonly used to solve rare class problems.

In the next section we propose a modification to nonlinear RankSVMs that takes specific advantage of unbalanced datasets to achieve $O(mm_+)$ time and $O(mm_+)$ space, while not sacrificing predictive performance.

3.4 RankRC: Ranking with Rare Class Representation

To make RankSVM computationally feasible for large scale unbalanced problems, we propose to enforce a rare class representation for the decision surface. Specifically, we propose

to restrict the solution to the form

$$f(\mathbf{x}) = \sum_{\{i:y_i=+1\}} \beta_i k(\mathbf{x}_i, \mathbf{x}) \ , \tag{3.8}$$

so it consists only of kernel function realizations of the minority class. We call this RankRC to indicate a Rare Class representation, instead of a support vector representation.

We present motivation for RankRC by assuming specific properties of the class conditional distributions and kernel function. Zhu et al. [114] make use of similar assumptions, however, in their method they attempt to directly estimate the likelihood ratio. In contrast, we are using a regularized loss minimization approach.

Recall that the optimal ranking function for a classification problem is the posterior probability, $P(y = 1|\mathbf{x})$, since it minimizes the Bayes risk for arbitrary costs. From Bayes' Theorem, we have

$$P(y = 1|\mathbf{x}) = \frac{P(y = 1)P(\mathbf{x}|y = 1)}{P(y = 1)P(\mathbf{x}|y = 1) + P(y = -1)P(\mathbf{x}|y = -1)} \ . \tag{3.9}$$

In addition, any monotonic transformation of (3.9) also yields equivalent ranking capability. Dividing the numerator and denominator of (3.9) by $P(y = -1)P(\mathbf{x}|y = -1)$, we note that $P(y = 1|\mathbf{x})$ is a monotonic transformation of the likelihood ratio, denoted as

$$f(\mathbf{x}) = \frac{P(\mathbf{x}|y = 1)}{P(\mathbf{x}|y = -1)} \ , \tag{3.10}$$

which is the ranking function we are interested in obtaining.

Using kernel density estimation (also called the Parzen window estimate), the conditional density $P(\mathbf{x}|y = 1)$ can be approximated using

$$P(\mathbf{x}|y = 1) = \frac{1}{m_+} \sum_{\{i:y_i=+1\}} a_i k(\mathbf{x}, \mathbf{x}_i; \sigma) \ , \tag{3.11}$$

where $k(\mathbf{x}, \mathbf{x}_i; \sigma)$ represents a kernel density function—typically a continuous unimodal function of $\mathbf{x}$ with a peak at $\mathbf{x}_i$ and a width localization parameter $\sigma > 0$.[1] The constants

[1]The kernel density function is not the same as a Mercer kernel described in Chapter 2. The kernel density function used in the Parzen window estimate is a symmetric but not necessarily positive function that integrates to one. Mercer's kernel described in Chapter 2 is associated with a unique Hilbert space of functions called its reproducing kernel Hilbert space (RKHS). It is more general in the sense that it can be defined over abstract syntax, for example strings, trees or graphs. However, to be a valid Mercer kernel, it

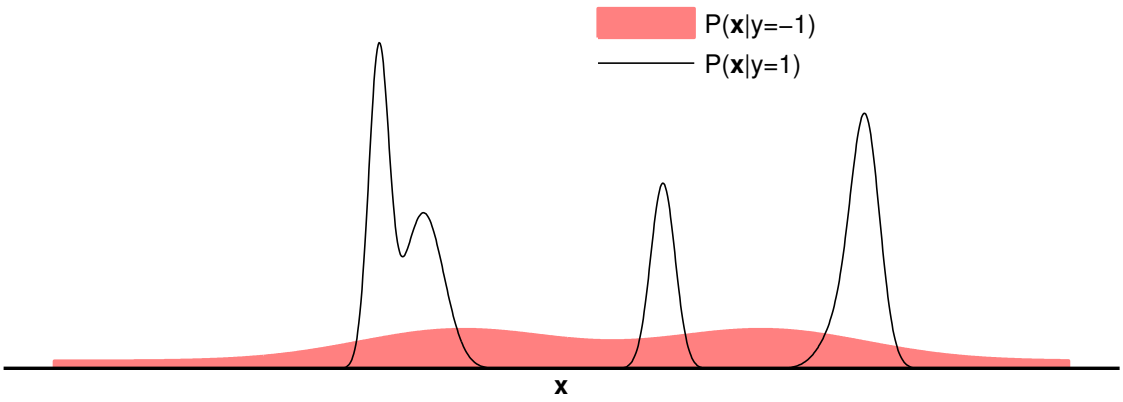


Figure 3.2: Example class conditional distributions for a rare class dataset showing that $P(\mathbf{x}|y=1)$ is concentrated with bounded support, while $P(\mathbf{x}|y=-1)$ is relatively constant in the local regions around the positive class.

a_i are used to normalize the density function and allow for a more general mixture model. For example, if we define

$$k(\mathbf{x}, \mathbf{x}_i; \sigma) = \exp\left\{\frac{||\mathbf{x}_i - \mathbf{x}||^2}{\sigma^2}\right\}$$

as the Gaussian kernel, then (3.11) is equivalent to a mixture of m_+ identical spherical normals centered at the rare class examples. This mixture model encompasses a large range of possible distributions to represent the m_+ rare examples provided.

In rare class problems, most examples are from the majority class ($y=-1$) and only a small number are from the rare class ($y=1$). It is reasonable to assume the minority class examples are concentrated in local regions with bounded support, while the majority class acts as background noise. Therefore, in a neighborhood around the minority class examples, the conditional density function $P(\mathbf{x}|y=-1)$ can be assumed to be relatively flat in comparison to $P(\mathbf{x}|y=1)$, see Figure 3.2 for instance. Assume $P(\mathbf{x}|y=-1) \approx c_i$ for each minority example i in the neighborhood of $\mathbf{x}_i$.[2] Together with (3.11), the likelihood ratio (3.10) can be written as

must be positive semi-definite (PSD). Ignoring normalization constants, well known examples of kernels that are both kernel density functions and Mercer kernels are the Gaussian kernel, multivariate Student kernel and the Laplacian kernel.

[2]We do not make this more precise since we are mainly interested in motivating an approximate form.

$$f(\mathbf{x}) \approx \sum_{\{i:y_i=+1\}} \frac{a_i k(\mathbf{x}_i, \mathbf{x})}{c_i} , \tag{3.12}$$

which only uses kernel functions at rare class points. The form (3.12) is equivalent to the rare class representation (3.8). In contrast to (3.6), this form takes specific advantage of the conditional density structure often found in rare class problems.

The rare class form (3.8) implies that

$$\mathbf{w} = \sum_{\{i:y_i=+1\}} \beta_i \phi(\mathbf{x}_i) . \tag{3.13}$$

By substituting (3.13) in the regularized ranking loss problem (3.4), we obtain the following RankRC problem in m_+ variables,

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^{m_+}} \quad \frac{1}{m_+ m_-} \sum_{\{i:y_i=+1\}} \sum_{\{j:y_j=-1\}} \ell_h \left(K_{i+}\boldsymbol{\beta} - K_{j+}\boldsymbol{\beta}\right) + \frac{\lambda}{2}\boldsymbol{\beta}^T K_{++}\boldsymbol{\beta} . \tag{3.14}$$

Here, K_{i+} denotes ith row of K with column entries corresponding to the positive class, and $K_{++} \in \mathbb{R}^{m_+ \times m_+}$ is the square submatrix of K corresponding to positive class entries.

Although we motivated the rare class kernel formulation assuming certain properties of the class conditional distribution and kernel functions, we may still expect the rare class representation to perform adequately under more general settings. In Chapter 4 we shall analyze RankRC in a general setting, and show that for an unbalanced dataset, the rare class representation is optimal with respect to RankSVM when a fixed number of kernel functions are used. In the next section, we discuss the optimization method and algorithm complexity.

3.5 Optimization Algorithm and Complexity

As discussed earlier, we can replace the hinge loss, ℓ_h, with the ϵ-smoothed differentiable approximation, ℓ_ϵ to obtain a differentiable objective function:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^{m_+}} \quad \frac{1}{m_+m_-}\sum_{\{i:y_i=+1\}}\sum_{\{j:y_j=-1\}} \ell_\epsilon\left(K_{i+}\boldsymbol{\beta}-K_{j+}\boldsymbol{\beta}\right)+\frac{\lambda}{2}\boldsymbol{\beta}^T K_{++}\boldsymbol{\beta}\,. \tag{3.15}$$

To solve (3.15) we can use several approaches, which are discussed below.

3.5.1 Linearization

Since K_{++} is a positive semi-definite matrix, it has an eigen-decomposition which can be expressed in the form, $K_{++} = U\Lambda U^T$, with U being an orthogonal matrix (i.e. $U^TU = I$) and Λ a diagonal matrix containing non-negative eigenvalues of K_{++}. Let $\mathbf{w} = \Lambda^{\frac{1}{2}}U^T\boldsymbol{\beta}$, then

$$\boldsymbol{\beta} = U\Lambda^{\dagger\frac{1}{2}}\mathbf{w}\,, \tag{3.16}$$

where $\Lambda^\dagger$ denotes the pseudoinverse of Λ. We can substitute (3.16) in (3.15) to obtain the following linear (hypothesis) space problem,

$$\min_{\mathbf{w}\in\mathbb{R}^{m_+}} \quad \frac{1}{m_+m_-}\sum_{\{i:y_i=+1\}}\sum_{\{j:y_j=-1\}} \ell_\epsilon\left(K_{i+}U\Lambda^{\dagger\frac{1}{2}}\mathbf{w}-K_{j+}U\Lambda^{\dagger\frac{1}{2}}\mathbf{w}\right)+\frac{\lambda}{2}\|\mathbf{w}\|_2^2\,. \tag{3.17}$$

That is, Problem (3.17) is equivalent to Problem (3.4) with data points given by $\mathbf{x}_i = (K_{i+}U\Lambda^{\dagger\frac{1}{2}})^T = \Lambda^{\dagger\frac{1}{2}}U^TK_{i+}^T \in \mathbb{R}^{m_+}$, $i = 1,...,m$. Therefore we can use the algorithm described in Chapelle and Keerthi [26] to solve the linear ranking problem in $O(mm_+ + m\log m) = O(mm_+)$ time. The cost of computing $\mathbf{x}_i = K_{i+}U\Lambda^{\dagger\frac{1}{2}}, i = 1,...,m$, is $O(mm_+^2)$. The cost of factoring K_{++} is $O(m_+^3)$. Therefore the total time is $O(mm_+^2 + m_+^3)$. Once we solve for optimal $\mathbf{w}$ we can use (3.16) to obtain $\boldsymbol{\beta}$ for subsequent testing purposes. Also, since we only need kernel entries $\{K_{ij} : y_i = 1, j = 1,...,m\}$, the method uses $O(mm_+)$ space.

3.5.2 Unconstrained Optimization

We can also directly solve (3.15) using standard unconstrained optimization methods. Gradient only methods, such as steepest descent and nonlinear conjugate gradient do not require estimation of the Hessian. Although this makes each iteration much cheaper, convergence can be slow, especially near the solution. In contrast Hessian based algorithms, such as Newton's method can obtain quadratic convergence near the solution, but each iteration

can be expensive. In Newton's method, the pth iterate is updated according to

$$\boldsymbol{\beta}^{(p+1)} = \boldsymbol{\beta}^{(p)} + \mathbf{s} \,,$$

where the step, $\mathbf{s}$, is obtained by minimizing the quadratic Taylor approximation around the current iterate $\boldsymbol{\beta}^{(p)}$:

$$\min_{\mathbf{s}} \quad \mathbf{s}^T \mathbf{g}^{(p)} + \frac{1}{2} \mathbf{s}^T H^{(p)} \mathbf{s} \,, \tag{3.18}$$

where $H^{(p)}$ and $\mathbf{g}^{(p)}$ are the Hessian and gradient of the objective at $\boldsymbol{\beta}^{(p)}$, respectively. Problem (3.18) has a closed form solution given by

$$\mathbf{s} = -\left(H^{(p)}\right)^{-1} \mathbf{g}^{(p)} \,.$$

Since $H^{(p)}$ is a $m_+ \times m_+$ matrix, this involves $O(m_+^3)$ cost in each iteration. To avoid this, we can use the truncated Newton method in which $H^{(p)}\mathbf{s} = -\mathbf{g}^{(p)}$ is solved using linear conjugate gradient. Here, the Hessian is not computed explicitly and the method iteratively approximates the solution using Hessian-vector products. Since each iteration in the linear conjugate gradient algorithm leads to a descent direction, we can terminate early while still improving convergence.

A drawback of (truncated) Newton's method is that convergence can be guaranteed only from a certain neighbourhood of the solution. If the initial point is not chosen close enough to the solution, the method can be slow to converge, or fail altogether. Therefore we consider a subspace-trust-region method, which combines the benefit of a truncated Newton step with steepest descent. In our tests, we found that the subspace-trust-region method converges with significantly fewer iterations than the truncated Newton method.

The idea behind the trust-region method is to solve (3.18) while constraining the step, $\mathbf{s}$, to a neighborhood around the current iterate, in which the approximation is trusted:

$$\begin{aligned} \min_{\mathbf{s}} \quad & \frac{1}{2} \mathbf{s}^T H^{(p)} \mathbf{s} + \mathbf{s}^T \mathbf{g}^{(p)} \\ \text{s.t.} \quad & ||\mathbf{s}||_2 \leq \Delta^{(p)} \,. \end{aligned} \tag{3.19}$$

The trust region radius, $\Delta^{(p)}$, is adjusted at each iterate according to standard rules, for example it is decreased if the solution obtained is worse than the current iterate. Problem

(3.19) can be solved accurately [e.g see 18], however, the solution uses the full eigen-decomposition of $H^{(p)}$. To avoid this computation, in the subspace-trust-region method, Problem (3.19) is restricted to a two-dimensional subspace spanned by the gradient, $\mathbf{g}^{(p)}$, and an approximate Newton direction, $\mathbf{s}_2$, which can be obtained by solving $H^{(p)}\mathbf{s}_2 = -\mathbf{g}^{(p)}$ using linear conjugate gradient [21]. The idea behind this choice is to ensure global convergence, while maintaining fast local convergence. Once the subspace has been computed, solving (3.19) costs $O(1)$ time, since in the subspace the problem is only two-dimensional. The implementation we use is provided in Matlab's optimization toolbox, `fminunc/fmincon`.

Computing Gradient and Hessian-Vector Product

We describe how we can compute the gradient and Hessian-vector product for Problem (3.15) efficiently. Let $K_{\bullet+} = [K_{ij}]_{i=1,\ldots,m,y_j=1} \in \mathbb{R}^{m\times m_+}$ denote the rectangular submatrix of K with columns indexed by the positive class. Consider the expanded matrix

$$A = [K_{i+} - K_{j+}]_{i:y_i=1,j:y_j=-1} \in \mathbb{R}^{m_+m_-\times m_+},$$

consisting of the differences of rows in $K_{\bullet+}$ corresponding to all pairwise preferences. In our computation we do not explicitly form matrix A, rather we note that A can be expressed as a sparse matrix product:

$$A = PK_{\bullet+},$$

where $P \in \mathbb{R}^{m_+m_-\times m}$ is a sparse matrix that encodes a pairwise preference. That is, if $y_i > y_j$, then there exists a row r in P such that $P_{ri} = 1, P_{rj} = -1$ and the rest of the row is zero. Let A_r denote the rth row of A. Then the ranking loss expression in (3.15) can be written as,

$$\begin{aligned}
&\sum_{\{i:y_i=+1\}}\sum_{\{j:y_j=-1\}} \ell_\epsilon\left(K_{i+}\boldsymbol{\beta} - K_{j+}\boldsymbol{\beta}\right) \\
&= \sum_{r=1}^{m_+m_-} \ell_\epsilon(A_r\boldsymbol{\beta}) \\
&= \sum_{r=1}^{m_+m_-} \mathbb{I}[r\in\mathcal{L}](1-\epsilon-A_r\boldsymbol{\beta}) + \sum_{r=1}^{m_+m_-} \mathbb{I}[r\in\mathcal{Q}]\frac{1}{4\epsilon}(1-A_r\boldsymbol{\beta})^2 ,
\end{aligned} \tag{3.20}$$

where $\mathcal{L} = \{r : A_r\boldsymbol{\beta} < 1-2\epsilon\}$ is the set of pairwise differences which are in the linear portion of ℓ_ϵ, and $\mathcal{Q} = \{r : 1-2\epsilon \le A_r\boldsymbol{\beta} < 1\}$ is the set which fall in the quadratic part. Denote

$\mathbf{e} \in \mathbb{R}^{m_+ m_-}$ as a vector of ones. Define $\mathbf{e}^{\mathcal{L}} \in \mathbb{R}^{m_+ m_-}$ as a binary vector where $\mathbf{e}^{\mathcal{L}}_r = 1$ if $r \in \mathcal{L}$ and $\mathbf{e}^{\mathcal{L}}_r = 0$ if $r \notin \mathcal{L}$. Also define $I^{\mathcal{Q}} \in \mathbb{R}^{m_+ m_- \times m_+ m_-}$ as a diagonal matrix, where $I^{\mathcal{Q}}_{rr} = 1$, if $r \in \mathcal{Q}$, and $I^{\mathcal{Q}}_{rr} = 0$, if $r \notin \mathcal{Q}$. Then (3.20) is equivalent to

$$\begin{aligned}&\left(\mathbf{e}^{\mathcal{L}}\right)^T ((1-\epsilon)\mathbf{e} - A\boldsymbol{\beta}) + \frac{1}{4\epsilon}(\mathbf{e} - A\boldsymbol{\beta})^T I^{\mathcal{Q}} (\mathbf{e} - A\boldsymbol{\beta}) \\ &= \left(\mathbf{e}^{\mathcal{L}}\right)^T ((1-\epsilon)\mathbf{e} - PK_{\bullet+}\boldsymbol{\beta}) + \frac{1}{4\epsilon}(\mathbf{e} - PK_{\bullet+}\boldsymbol{\beta})^T I^{\mathcal{Q}} (\mathbf{e} - PK_{\bullet+}\boldsymbol{\beta}) \ .\end{aligned}$$

Therefore the objective function in (3.15) can be expressed as

$$F(\boldsymbol{\beta}) \triangleq \frac{1}{m_+ m_-}\left[\left(\mathbf{e}^{\mathcal{L}}\right)^T ((1-\epsilon)\mathbf{e} - PK_{\bullet+}\boldsymbol{\beta}) + \frac{1}{4\epsilon}(\mathbf{e} - PK_{\bullet+}\boldsymbol{\beta})^T I^{\mathcal{Q}} (\mathbf{e} - PK_{\bullet+}\boldsymbol{\beta})\right] + \frac{\lambda}{2}\boldsymbol{\beta}^T K_{++}\boldsymbol{\beta} \ . \tag{3.21}$$

We obtain the gradient by taking the derivative of (3.21) with respect to $\boldsymbol{\beta}$:

$$\begin{aligned}\mathbf{g} \triangleq \frac{\partial F}{\partial \boldsymbol{\beta}} &= \frac{1}{m_+ m_-}\left[-\left(\mathbf{e}^{\mathcal{L}}\right)^T PK_{\bullet+} + \frac{1}{2\epsilon} PK_{\bullet+} I^{\mathcal{Q}} (PK_{\bullet+}\boldsymbol{\beta} - \mathbf{e})\right] + \lambda K_{++}\boldsymbol{\beta} \\ &= \frac{1}{m_+ m_-}\left[-\left(\left(\mathbf{e}^{\mathcal{L}}\right)^T P\right) K_{\bullet+} + \frac{1}{2\epsilon}\left(P\left(K_{\bullet+}\left(I^{\mathcal{Q}}P\right)(K_{\bullet+}\boldsymbol{\beta})\right) - P\left(K_{\bullet+}\left(I^{\mathcal{Q}}\mathbf{e}\right)\right)\right)\right] + \lambda K_{++}\boldsymbol{\beta} \ .\end{aligned} \tag{3.22}$$

In the last expression we have used brackets to emphasize the order of operations that leads to an efficient implementation by avoiding the computation of $A = PK_{\bullet+}$. It can be verified that the time required is $O(mm_+)$.

We obtain the Hessian by taking the derivative of (3.22) with respect to $\boldsymbol{\beta}$:

$$H \triangleq \frac{\partial^2 F}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T} = \frac{1}{2\epsilon m_+ m_-}\left(PK_{\bullet+} I^{\mathcal{Q}} PK_{\bullet+}\right) + \lambda K_{++} \ .$$

Note the Hessian requires computing A. However, for the linear conjugate gradient method we only require computing $H\mathbf{s}$ for some vector $\mathbf{s}$. In this case, we can avoid computing A by using the following order of operations:

$$H\mathbf{s} = \frac{1}{2\epsilon m_+ m_-}\left(P\left(K_{\bullet+}\left(I^{\mathcal{Q}}P\right)(K_{\bullet+}\mathbf{s})\right)\right) + \lambda K_{++}\mathbf{s} \ .$$

The time required to compute $H\mathbf{s}$ is also $O(mm_+)$.

In the subspace-trust-region method we use a maximum of 25 conjugate gradient it-

erations.[3] We found the solution usually converges in a constant number of trust region iterations. Since each iteration requires $O(mm_+)$ time, the total time required by the algorithm is $O(mm_+)$. Total space is also $O(mm_+)$.

Finally, we note that we can slightly improve the time required to compute the gradient and Hessian-vector product by first sorting the values of $K_{\bullet+}\beta$ or $K_{\bullet+}\mathbf{s}$. Though this does not improve the *big-O* efficiency, it does reduce the constant factor. We refer the interested reader to [26] for details on a method which can be adapted for the nonlinear RankRC objective (3.15).

3.6 Summary

We use a ranking loss function to tackle the problem of learning from unbalanced datasets. Minimizing biclass ranking loss is equivalent to maximizing the AUC measure, which overcomes the inadequacies of accuracy, used by conventional classification algorithms. The resulting regularized loss minimization problem corresponds to a biclass RankSVM problem. We propose a modification to RankSVM, called RankRC, that takes advantage of the rare class situation by restricting the solution to a linear combination of rare class kernel functions. This allows us to solve the nonlinear ranking problem in $O(mm_+)$ time and $O(mm_+)$ space, thus enabling us to solve problems which are too large for kernel RankSVM. We motivate this formulation by assuming certain properties of the class distribution often found in rare class problems and kernel choice. In the next chapter we provide further theoretical justification for the RankRC model.

[3] We use diagonal preconditioning and warm-starts as λ is varied from high to low.

Chapter 4

Theoretical Properties of RankRC

In this chapter we analyze properties of RankRC and obtain the following results.

- We mathematically establish an upper bound on the difference between the optimal hypotheses of RankSVM and RankRC. This bound is established by observing that a regularized loss minimization problem with a hypothesis instantiated with points in a subset is equivalent to a regularized loss minimization problem with a hypothesis instantiated by the full data set but using the orthogonal projection of the original feature mapping.
- We show that the upper bound suggests that, under the assumption that a hypothesis is instantiated by a subset of data points of a fixed cardinality, it is optimal to choose data points from the rare class first. Hence, this bound provides additional theoretical justification for RankRC.
- We demonstrate that the Nyström kernel approximation method is equivalent to solving a kernel regularized loss problem instantiated by a subset of data points corresponding to the selected columns. Consequently, theoretical bounds for Nyström kernel approximation methods can be established based on the perturbation analysis of the orthogonal projection in the feature mapping. We demonstrate that this approach provides tighter bounds in comparison to perturbation analysis based on a kernel matrix approximation, which can be arbitrarily large depending on the condition number of the approximation matrix.

We note that some of our results in this chapter are general and apply to any regularized loss minimization problem. Therefore, these results can be useful to analyze and devise

algorithms for other approximate kernel problems as well.

4.1 Comparison of RankRC with RankSVM

In this section we analytically compare the solution of RankRC with RankSVM. In particular, we establish a bound for the difference between the solution of RankSVM and a solution in which the hypothesis is restricted to an arbitrary subset of kernel functions. This bound shows that it is optimal to first include kernel functions that correspond to points from the rare class when the dataset is unbalanced.

4.1.1 Projected Mapping Equivalence

We first establish equivalence between instantiating a hypothesis using a subset of training points and instantiating a hypothesis using the full training set but with the feature mapping equal to the orthogonal projection of the original mapping. We show this is true for an arbitrary loss function. Subsequently, we use this result to bound the difference between the RankSVM classifier and a classifier that is restricted to a subset of kernel functions, by conducting a stability analysis for the RankSVM optimization problem under a projected feature map perturbation.

For the purpose of analysis, we shall work explicitly in the high-dimensional feature space. Let $\phi : \mathcal{X} \to \mathcal{F} \subseteq \mathbb{R}^{d'}$, denote a feature map corresponding to the kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, such that $\forall \mathbf{u}, \mathbf{v} \in \mathcal{X}$, $k(\mathbf{u}, \mathbf{v}) = \phi(\mathbf{u})^T \phi(\mathbf{v})$. For an arbitrary loss function, $L : \mathbb{R}^m \to \mathbb{R}$, and regularization parameter, $\lambda \in \mathbb{R}_+$, consider the following regularized loss minimization problem in space $\mathcal{F}$,

$$\min_{\mathbf{w} \in \mathbb{R}^{d'}} \quad L\left(\mathbf{w}^T \phi(\mathbf{x}_1), ..., \mathbf{w}^T \phi(\mathbf{x}_m)\right) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 \,. \tag{4.1}$$

Here, the hypothesis, $f_\phi(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$, can be nonlinear in the input space, $\mathcal{X}$, but is linear in the high-dimensional feature space, $\mathcal{F}$. We use the subscript ϕ in f_ϕ to indicate the feature map used in the hypothesis. Note, RankSVM is a special case of (4.1) using a ranking loss for L. From the Representer Theorem, the solution of (4.1) is of the form

$$f_\phi(\mathbf{x}) = \sum_{i=1}^{m} \beta_i k(\mathbf{x}_i, \mathbf{x}) = \sum_{i=1}^{m} \beta_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) \,, \quad \text{or} \quad \mathbf{w} = \sum_{i=1}^{m} \beta_i \phi(\mathbf{x}_i) \,. \tag{4.2}$$

This implies that the optimal hypothesis can always be represented using the full training set and the solution vector $\mathbf{w} \in \mathcal{S} = \text{span}\{\phi(\mathbf{x}_i) : i = 1, ..., m\}$ is a linear combination of all the points in feature space.

Now consider restricting the hypothesis to an arbitrary subset of kernel functions, indexed by $\mathcal{R} \subseteq \{1, ..., m\}$:

$$\bar{f}_\phi(\mathbf{x}) = \sum_{i \in \mathcal{R}} \beta_i k(\mathbf{x}_i, \mathbf{x}) = \sum_{i \in \mathcal{R}} \beta_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) \;, \quad \text{or} \quad \mathbf{w} = \sum_{i \in \mathcal{R}} \beta_i \phi(\mathbf{x}_i) \;, \tag{4.3}$$

with $\bar{f}_\phi(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$. We use the overline in $\bar{f}_\phi$ to indicate a restricted hypothesis. Subsequently, we shall refer to (4.3) as the $\mathcal{R}$-subset representation or classifier. In this case, the solution vector, $\mathbf{w} \in \mathcal{S}_\mathcal{R} = \text{span}\{\phi(\mathbf{x}_i) : i \in \mathcal{R}\}$, is a linear combination of the subset of points in feature space indexed by $\mathcal{R}$. Since the set $\mathcal{S}_\mathcal{R}$ defines all feasible values of $\mathbf{w}$, restricting the hypothesis to the $\mathcal{R}$-subset representation corresponds to solving the following *constrained* regularized loss minimization problem in feature space:

$$\begin{aligned} \min_{\mathbf{w} \in \mathbb{R}^{d'}} \quad & L\left(\mathbf{w}^T \phi(\mathbf{x}_1), ..., \mathbf{w}^T \phi(\mathbf{x}_m)\right) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 \;, \\ \text{subject to} \quad & \mathbf{w} \in \mathcal{S}_\mathcal{R} = \text{span}\{\phi(\mathbf{x}_i) : i \in \mathcal{R}\} \;, \quad \mathcal{R} \subseteq \{1, ..., m\} \;. \end{aligned} \tag{4.4}$$

Note, RankRC is a special case of (4.4) using a ranking loss for L and setting $\mathcal{R} = \{i : y_i = 1\}$.

In Theorem 5 we will establish that problem (4.4) is equivalent to problem (4.1) under a projected feature map. That is, problem (4.4) is equivalent to the following *unconstrained* loss minimization problem,

$$\min_{\mathbf{w} \in \mathbb{R}^{d'}} \quad L\left(\mathbf{w}^T \phi_\mathcal{R}(\mathbf{x}_1), ..., \mathbf{w}^T \phi_\mathcal{R}(\mathbf{x}_m)\right) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 \;, \tag{4.5}$$

with hypothesis, $f_{\phi_\mathcal{R}}(\mathbf{x}) = \mathbf{w}^T \phi_\mathcal{R}(\mathbf{x})$ and a feature map, $\phi_\mathcal{R} : \mathcal{X} \to \mathcal{F}_\mathcal{R} \subseteq \mathbb{R}^{d'}$, defined as the orthogonal projection of ϕ onto $\mathcal{S}_\mathcal{R}$, i.e.,

$$\phi_\mathcal{R}(\mathbf{x}) = \text{Proj}_{\mathcal{S}_\mathcal{R}}\left(\phi(\mathbf{x})\right) \;. \tag{4.6}$$

The feature map, $\phi_\mathcal{R}$, maps the input space to a feature space, $\mathcal{F}_\mathcal{R}$, which contains vectors of the same dimensionality, d', as the original feature space, $\mathcal{F}$. Before establishing the equivalence of (4.4) and (4.5), we first prove a technical lemma.

Lemma 4. *Consider a feature map, $\phi : \mathcal{X} \to \mathcal{F} \subseteq \mathbb{R}^{d'}$, and its projected map, $\phi_\mathcal{R} : \mathcal{X} \to$*

$\mathcal{F}_{\mathcal{R}} \subseteq \mathbb{R}^{d'}$, defined by (4.6) *for some index subset $\mathcal{R} \subseteq \{1,...,m\}$ and $\mathcal{S}_{\mathcal{R}} = span\{\phi(\mathbf{x}_i) : i \in \mathcal{R}\}$. Assume that $\mathbf{w} \in \mathbb{R}^{d'}$ is feasible for the constrained regularized loss minimization problem problem* (4.4)*. Let $\bar{f}_\phi(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x})$ and $f_{\phi_{\mathcal{R}}}(\mathbf{x}) = \mathbf{w}^T\phi_{\mathcal{R}}(\mathbf{x})$ be hypotheses associated with feature mapping ϕ and $\phi_{\mathcal{R}}$, respectively. Then*

$$\bar{f}_\phi(\mathbf{x}) = f_{\phi_{\mathcal{R}}}(\mathbf{x}), \quad \forall \mathbf{x} \in \mathcal{X}. \tag{4.7}$$

Proof. Given any $\phi(\mathbf{x})$, there exists a unique orthogonal decomposition

$$\phi(\mathbf{x}) = \phi_{\mathcal{R}}(\mathbf{x}) + \phi^{\perp}_{\mathcal{R}}(\mathbf{x}) \,, \tag{4.8}$$

where $\phi_{\mathcal{R}}(\mathbf{x}) \in \mathcal{S}_{\mathcal{R}} \subseteq \mathbb{R}^{d'}$ is a component in $\mathcal{S}_{\mathcal{R}}$ and $\phi^{\perp}_{\mathcal{R}}(\mathbf{x}) \in \mathbb{R}^{d'}$ is a component orthogonal to $\mathcal{S}_{\mathcal{R}}$. By definition, $\phi(\mathbf{x}_i) \in \mathcal{S}_{\mathcal{R}}, \forall i \in \mathcal{R}$. Hence

$$\phi(\mathbf{x}_i)^T \phi^{\perp}_{\mathcal{R}}(\mathbf{x}) = 0, \quad \forall i \in \mathcal{R}, \ \forall \mathbf{x} \in \mathcal{X}. \tag{4.9}$$

Since $\mathbf{w}$ is a feasible point for (4.4), we can write $\mathbf{w} = \sum_{i\in\mathcal{R}} \beta_i \phi(\mathbf{x}_i)$ for some $\boldsymbol{\beta} \in \mathbb{R}^{|\mathcal{R}|}$. Then using (4.9) we have

$$\begin{aligned}
\bar{f}_\phi(\mathbf{x}) &= \mathbf{w}^T\phi(\mathbf{x}) \\
&= \left(\sum_{i\in\mathcal{R}} \beta_i \phi(\mathbf{x}_i)\right)^T \left(\phi_{\mathcal{R}}(\mathbf{x}) + \phi^{\perp}_{\mathcal{R}}(\mathbf{x})\right) \\
&= \left(\sum_{i\in\mathcal{R}} \beta_i \phi(\mathbf{x}_i)\right)^T \phi_{\mathcal{R}}(\mathbf{x}) \\
&= \mathbf{w}^T\phi_{\mathcal{R}}(\mathbf{x}) \\
&= f_{\phi_{\mathcal{R}}}(\mathbf{x}).
\end{aligned}$$

This completes the proof. □

Lemma 4 shows that, for any feasible $\mathbf{w}$ of (4.4), the hypothesis $\bar{f}_\phi(\mathbf{x})$ corresponding to the map ϕ, is equivalent to the hypothesis $f_{\phi_{\mathcal{R}}}(\mathbf{x})$, corresponding to the projected map $\phi_{\mathcal{R}}$.

Theorem 5. *Consider a feature map, $\phi : \mathcal{X} \to \mathcal{F} \subseteq \mathbb{R}^{d'}$, and its projected map, $\phi_{\mathcal{R}} : \mathcal{X} \to \mathcal{F}_{\mathcal{R}} \subseteq \mathbb{R}^{d'}$, defined by* (4.6) *for some index subset $\mathcal{R} \subseteq \{1,...,m\}$ and $\mathcal{S}_{\mathcal{R}} = span\{\phi(\mathbf{x}_i) : i \in \mathcal{R}\}$. Then the constrained regularized loss minimization problem* (4.4)*, using map*

ϕ, is equivalent to the unconstrained regularized loss minimization problem (4.5)*, using the projected map $\phi_{\mathcal{R}}$, i.e., $\mathbf{w}^*$ solves* (4.4) *if and only if $\mathbf{w}^*$ solves* (4.5)*. In addition, assuming $\mathbf{w}^*$ solves either* (4.4) *or* (4.5)*, then the hypothesis $\bar{f}^*_{\phi}(\mathbf{x}) = (\mathbf{w}^*)^T\phi(\mathbf{x})$, with map ϕ, is equivalent to the hypothesis $f^*_{\phi_{\mathcal{R}}}(\mathbf{x}) = (\mathbf{w}^*)^T\phi_{\mathcal{R}}(\mathbf{x})$, with the projected map $\phi_{\mathcal{R}}$, i.e.,*

$$\bar{f}^*_{\phi}(\mathbf{x}) = f^*_{\phi_{\mathcal{R}}}(\mathbf{x}), \quad \forall \mathbf{x} \in \mathcal{X}. \tag{4.10}$$

Proof. Using the Representer Theorem, there exists a solution $\mathbf{w}^*_{\mathcal{R}}$ to problem (4.5), which can be expressed as

$$\mathbf{w}^*_{\mathcal{R}} = \sum_{i=1}^{m} \beta^*_i \phi_{\mathcal{R}}(\mathbf{x}_i). \tag{4.11}$$

Hence, for any $\mathbf{w} \in \mathbb{R}^{d'}$,

$$L\left((\mathbf{w}^*_{\mathcal{R}})^T\phi_{\mathcal{R}}(\mathbf{x}_1), ..., (\mathbf{w}^*_{\mathcal{R}})^T\phi_{\mathcal{R}}(\mathbf{x}_m)\right) + \frac{\lambda}{2}\|\mathbf{w}^*_{\mathcal{R}}\|_2^2 \leq L\left(\mathbf{w}^T\phi_{\mathcal{R}}(\mathbf{x}_1), ..., \mathbf{w}^T\phi_{\mathcal{R}}(\mathbf{x}_m)\right) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 \tag{4.12}$$

Since $\phi_{\mathcal{R}}(\mathbf{x}) \in \mathcal{S}_{\mathcal{R}}$, from (4.11), $\mathbf{w}^*_{\mathcal{R}} \in \mathcal{S}_{\mathcal{R}}$. Hence $\mathbf{w}^*_{\mathcal{R}}$ satisfies the constraint in (4.4). Following Lemma 4,

$$(\mathbf{w}^*_{\mathcal{R}})^T\phi_{\mathcal{R}}(\mathbf{x}) = (\mathbf{w}^*_{\mathcal{R}})^T\phi(\mathbf{x}), \quad \forall \mathbf{x} \in \mathcal{X}. \tag{4.13}$$

Now consider any feasible point $\mathbf{w}$ for (4.4). Following Lemma 4, we have

$$\mathbf{w}^T\phi(\mathbf{x}) = \mathbf{w}^T\phi_{\mathcal{R}}(\mathbf{x}), \quad \forall \mathbf{x} \in \mathcal{X}. \tag{4.14}$$

From (4.12) and (4.14),

$$L\left((\mathbf{w}^*_{\mathcal{R}})^T\phi(\mathbf{x}_1), ..., (\mathbf{w}^*_{\mathcal{R}})^T\phi(\mathbf{x}_m)\right) + \frac{\lambda}{2}\|\mathbf{w}^*_{\mathcal{R}}\|_2^2 \leq L\left(\mathbf{w}^T\phi(\mathbf{x}_1), ..., \mathbf{w}^T\phi(\mathbf{x}_m)\right) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 \tag{4.15}$$

Hence $\mathbf{w}^*_{\mathcal{R}}$ is a solution to (4.4).
Conversely let us assume that $\mathbf{w}^*$ is a solution to (4.4). Since $\mathbf{w}^*_{\mathcal{R}}$ is feasible for (4.4),

$$\begin{aligned} L\left((\mathbf{w}^*)^T\phi(\mathbf{x}_1),...,(\mathbf{w}^*)^T\phi(\mathbf{x}_m)\right)+\frac{\lambda}{2}\|\mathbf{w}^*\|_2^2 &\leq L\left((\mathbf{w}^*_{\mathcal{R}})^T\phi(\mathbf{x}_1),...,(\mathbf{w}^*_{\mathcal{R}})^T\phi(\mathbf{x}_m)\right)+\frac{\lambda}{2}\|\mathbf{w}^*_{\mathcal{R}}\|_2^2 \\ &= L\left((\mathbf{w}^*_{\mathcal{R}})^T\phi_{\mathcal{R}}(\mathbf{x}_1),...,(\mathbf{w}^*_{\mathcal{R}})^T\phi_{\mathcal{R}}(\mathbf{x}_m)\right)+\frac{\lambda}{2}\|\mathbf{w}^*_{\mathcal{R}}\|_2^2. \end{aligned}$$

where the equality follows from (4.13).
From Lemma 4,

$$(\mathbf{w}^*)^T\phi(\mathbf{x}) = (\mathbf{w}^*)^T\phi_{\mathcal{R}}(\mathbf{x}), \quad \forall \mathbf{x}\in\mathcal{X}.$$

Hence

$$L\left((\mathbf{w}^*)^T\phi_{\mathcal{R}}(\mathbf{x}_1),...,(\mathbf{w}^*)^T\phi_{\mathcal{R}}(\mathbf{x}_m)\right)+\frac{\lambda}{2}\|\mathbf{w}^*\|_2^2 \leq L\left((\mathbf{w}^*_{\mathcal{R}})^T\phi_{\mathcal{R}}(\mathbf{x}_1),...,(\mathbf{w}^*_{\mathcal{R}})^T\phi_{\mathcal{R}}(\mathbf{x}_m)\right)+\frac{\lambda}{2}\|\mathbf{w}^*_{\mathcal{R}}\|_2^2.$$

Following (4.12) , for any $\mathbf{w}\in\mathbb{R}^{d'}$,

$$L\left((\mathbf{w}^*)^T\phi_{\mathcal{R}}(\mathbf{x}_1),...,(\mathbf{w}^*)^T\phi_{\mathcal{R}}(\mathbf{x}_m)\right)+\lambda\|\mathbf{w}^*\|_2^2 \leq L\left(\mathbf{w}^T\phi_{\mathcal{R}}(\mathbf{x}_1),...,\mathbf{w}^T\phi_{\mathcal{R}}(\mathbf{x}_m)\right)+\frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

Hence the solution $\mathbf{w}^*$ is also a solution to (4.5). The result (4.10) immediately follows from Lemma 4 and the equivalence of (4.4) and (4.5). The proof is complete. □

If we assume the loss function is convex, we can also obtain the results of Lemma 4 and Theorem 5 directly from optimality conditions instead of using the Representer Theorem. The solution form (4.2) for the regularized loss minimization problem (4.1) can be obtained using the first order optimality conditions. Denote the subdifferential of L with respect to its ith argument by $\partial_i L(\cdot)$, then the optimality condition for (4.1) is

$$\sum_{i=1}^{m}\phi(x_i)(\partial_i L)+\lambda w = 0,$$

giving the required form. This method can be extended to the constrained regularized loss minimization (4.4) as well. If we use R to denote the matrix whose columns span $S_{\mathcal{R}}$, and N to be the matrix such that N^T spans the orthogonal complement of $S_{\mathcal{R}}$, then by the fundamental theorem of linear algebra, the constraint in (4.4) can be written as $Nw = 0$.

The optimality conditions of (4.4) are then

$$\sum_{i=1}^{m} \phi(x_i)(\partial_i L) + \lambda w - N^T \mu = 0, \quad Nw = 0.$$

We deduce from $Nw = 0$ that $w = Rw_{\mathcal{R}}$ for some vector $w_{\mathcal{R}}$, so from the first optimality condition, we have by multiplying through by R^T and solving that

$$w_{\mathcal{R}} = -\frac{1}{\lambda} \sum_{i=1}^{M} (\partial_i L)(R^T R)^{-1} R^T \phi(\mathbf{x}_i).$$

Since $R(R^T R)^{-1} R^T$ is the projection matrix onto the subspace $S_{\mathcal{R}}$, we obtain

$$w = -\frac{1}{\lambda} \sum_{i=1}^{M} (\partial_i L) \operatorname{Proj}_{S_{\mathcal{R}}} (\phi(\mathbf{x}_i)) = -\frac{1}{\lambda} \sum_{i=1}^{M} (\partial_i L) \phi_{\mathcal{R}}(\mathbf{x}_i).$$

The conclusions of Lemma 4 and Theorem 5 follow from this.

4.1.2 Projected Mapping Bound

Now consider the ranking loss problem. Define

$$R_\phi(\mathbf{w}) = \frac{1}{m_+ m_-} \sum_{\{i:y_i=+1\}} \sum_{\{j:y_j=-1\}} \ell_h \left(\mathbf{w}^T \phi(\mathbf{x}_i) - \mathbf{w}^T \phi(\mathbf{x}_j) \right), \tag{4.16}$$

where $\ell_h(z) = \max(0, 1-z)$ is the hinge loss. Setting

$$L\left(\mathbf{w}^T \phi(\mathbf{x}_1), \ldots, \mathbf{w}^T \phi(\mathbf{x}_m)\right) = R_\phi(\mathbf{w}) \tag{4.17}$$

in (4.1) gives:

$$\min_{\mathbf{w} \in \mathbb{R}^{d'}} \quad F_\phi(\mathbf{w}) = R_\phi(\mathbf{w}) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 . \tag{4.18}$$

Problem (4.18) corresponds to the RankSVM problem in feature space $\mathcal{F}$, defined by the feature map ϕ (or implicitly, by the kernel function, k). Similarly, setting (4.17) in problem (4.4) corresponds to a RankSVM problem in which the hypothesis is restricted to a $\mathcal{R}$-

subset representation:

$$\begin{aligned} \min_{\mathbf{w}\in\mathbb{R}^{d'}} \quad & F_\phi(\mathbf{w}) = R_\phi(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 \,, \\ \text{subject to} \quad & \mathbf{w} \in \mathcal{S}_\mathcal{R} = \text{span}\{\phi(\mathbf{x}_i) : i \in \mathcal{R}\} \,, \quad \mathcal{R} \subseteq \{1,...,m\} \,. \end{aligned} \tag{4.19}$$

Setting $L\left(\mathbf{w}^T\phi_\mathcal{R}(\mathbf{x}_1),...,\mathbf{w}^T\phi_\mathcal{R}(\mathbf{x}_m)\right) = R_{\phi_\mathcal{R}}(\mathbf{w})$ in problem (4.5), corresponds to the RankSVM problem with feature map $\phi_\mathcal{R}$:

$$\min_{\mathbf{w}\in\mathbb{R}^{d'}} \quad F_{\phi_\mathcal{R}}(\mathbf{w}) = R_{\phi_\mathcal{R}}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 \,. \tag{4.20}$$

Let f^*_ϕ, $\bar{f}^*_\phi$ and $f^*_{\phi_\mathcal{R}}$ denote the optimal hypotheses obtained by solving (4.18), (4.19) and (4.20), respectively. From Theorem 5, $\bar{f}^*_\phi(\mathbf{x}) = f^*_{\phi_\mathcal{R}}(\mathbf{x})$, and therefore $|f^*_\phi(\mathbf{x}) - \bar{f}^*_\phi(\mathbf{x})| = |f^*_\phi(\mathbf{x}) - f^*_{\phi_\mathcal{R}}(\mathbf{x})|$. In other words, we can bound the difference between a RankSVM classifier and a $\mathcal{R}$-subset classifier, by a stability analysis of the optimal RankSVM hypothesis under a perturbed (projected) feature map.

Stability analyses for a regular SVM have been conducted previously. In particular, Bousquet and Elisseeff [17] obtain a bound for a regular SVM under the effect of changing one training point. Cortes et al. [32] analyze stability of a regular SVM under the effect of changing the kernel matrix. Our stability analysis here differs from existing analyses in two aspects. Firstly, we obtain a bound under the effect of changing the feature mapping ϕ to $\phi_\mathcal{R}$. Secondly, we consider here the RankSVM problem instead of a regular SVM.

Theorem 6. *Consider a feature map, $\phi : \mathcal{X} \to \mathcal{F} \subseteq \mathbb{R}^{d'}$, and its projected map, $\phi_\mathcal{R} : \mathcal{X} \to \mathcal{F}_\mathcal{R} \subseteq \mathbb{R}^{d'}$, defined by* (4.6) *for some index subset $\mathcal{R} \subseteq \{1,...,m\}$ and $\mathcal{S}_\mathcal{R} = span\{\phi(\mathbf{x}_i) : i \in \mathcal{R}\}$. Assume that $f^*_\phi(\mathbf{x}) = (\mathbf{w}^*)^T\phi(\mathbf{x})$ is the optimal RankSVM hypothesis obtained by solving* (4.18) *with feature map ϕ, and $f^*_{\phi_\mathcal{R}}(\mathbf{x}) = (\mathbf{w}^*_\mathcal{R})^T\phi_\mathcal{R}(\mathbf{x})$ is the optimal RankSVM hypothesis obtained by solving* (4.20) *with feature map $\phi_\mathcal{R}$. Assume there exists $\kappa > 0$ such that $k(\mathbf{x},\mathbf{x}) \leq \kappa$, where $k : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ is the kernel map associated with ϕ. Then the following inequality holds,*

$$|f^*_{\phi_\mathcal{R}}(\mathbf{x}) - f^*_\phi(\mathbf{x})| \leq \frac{2\kappa}{\lambda}\left(\sum_{\{i:y_i=+1\}}\frac{\mathbb{I}[i\notin\mathcal{R}]}{m_+} + \sum_{\{j:y_j=-1\}}\frac{\mathbb{I}[j\notin\mathcal{R}]}{m_-}\right)^{\frac{1}{2}}, \quad \forall\mathbf{x}\in\mathcal{X}, \tag{4.21}$$

where $\mathbb{I}[p]$ denotes the indicator function and is equal to 1 if p is true, 0 if p is false.

Proof. Assume that $\mathbf{w}^*$ and $\mathbf{w}^*_{\mathcal{R}}$ are minimizers of (4.18) and (4.20), respectively. Let $\Delta\mathbf{w} = \mathbf{w}^*_{\mathcal{R}} - \mathbf{w}^*$.

Recall that a convex function g satisfies

$$g(\mathbf{u}+t(\mathbf{v}-\mathbf{u}))-g(\mathbf{u}) \leq t(g(\mathbf{v})-g(\mathbf{u}))$$

for all $\mathbf{u},\mathbf{v},t \in [0,1]$. Since ℓ_h is convex, R_ϕ and $R_{\phi_\mathcal{R}}$ are convex. Then

$$R_\phi(\mathbf{w}^*+t\Delta\mathbf{w})-R_\phi(\mathbf{w}^*) \leq t(R_\phi(\mathbf{w}^*_{\mathcal{R}})-R_\phi(\mathbf{w}^*)) \tag{4.22}$$

$$\text{and} \quad R_{\phi_\mathcal{R}}(\mathbf{w}^*_{\mathcal{R}}-t\Delta\mathbf{w})-R_{\phi_\mathcal{R}}(\mathbf{w}^*_{\mathcal{R}}) \leq t(R_{\phi_\mathcal{R}}(\mathbf{w}^*)-R_{\phi_\mathcal{R}}(\mathbf{w}^*_{\mathcal{R}}))\,, \tag{4.23}$$

for all $t \in [0,1]$.

Since $\mathbf{w}^*$ and $\mathbf{w}^*_{\mathcal{R}}$ are minimizers of F_ϕ and $F_{\phi_\mathcal{R}}$, for any $t \in [0,1]$, we have

$$F_\phi(\mathbf{w}^*) \leq F_\phi(\mathbf{w}^*+t\Delta\mathbf{w}) \tag{4.24}$$

$$\text{and} \quad F_{\phi_\mathcal{R}}(\mathbf{w}^*_{\mathcal{R}}) \leq F_{\phi_\mathcal{R}}(\mathbf{w}^*_{\mathcal{R}}-t\Delta\mathbf{w})\,. \tag{4.25}$$

Summing (4.24) and (4.25), using $F_\phi(\mathbf{w}) = R_\phi(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$ and the identity

$$\left(\|\mathbf{w}^*\|^2 - \|\mathbf{w}^*+t\Delta\mathbf{w}\|^2\right) + \left(\|\mathbf{w}^*_{\mathcal{R}}\|^2 - \|\mathbf{w}^*_{\mathcal{R}}-t\Delta\mathbf{w}\|^2\right) = 2t(1-t)\|\Delta\mathbf{w}\|^2,$$

we obtain

$$\lambda t(1-t)\|\Delta\mathbf{w}\|^2 \leq \left(R_\phi(\mathbf{w}^*+t\Delta\mathbf{w})-R_\phi(\mathbf{w}^*)\right) + \left(R_{\phi_\mathcal{R}}(\mathbf{w}^*_{\mathcal{R}}-t\Delta\mathbf{w})-R_{\phi_\mathcal{R}}(\mathbf{w}^*_{\mathcal{R}})\right) \tag{4.26}$$

Substituting (4.22) and (4.23) into (4.26), dividing by λt, and taking the limit $t \to 0$ gives

$$\begin{aligned}
\|\Delta\mathbf{w}\|^2 &\leq \frac{1}{\lambda}\left(R_\phi(\mathbf{w}^*_{\mathcal{R}})-R_{\phi_\mathcal{R}}(\mathbf{w}^*_{\mathcal{R}})+R_{\phi_\mathcal{R}}(\mathbf{w}^*)-R_\phi(\mathbf{w}^*)\right) \\
&= \frac{1}{\lambda m_+ m_-}\sum_{\{i:y_i=+1\}}\sum_{\{j:y_j=-1\}}\Big[\ell_h\left((\mathbf{w}^*_{\mathcal{R}})^T\phi(\mathbf{x}_i)-(\mathbf{w}^*_{\mathcal{R}})^T\phi(\mathbf{x}_j)\right)-\ell_h\left((\mathbf{w}^*_{\mathcal{R}})^T\phi_\mathcal{R}(\mathbf{x}_i)-(\mathbf{w}^*_{\mathcal{R}})^T\phi_\mathcal{R}(\mathbf{x}_j)\right) \\
&\qquad +\ell_h\left((\mathbf{w}^*)^T\phi_\mathcal{R}(\mathbf{x}_i)-(\mathbf{w}^*)^T\phi_\mathcal{R}(\mathbf{x}_j)\right)-\ell_h\left((\mathbf{w}^*)^T\phi(\mathbf{x}_i)-(\mathbf{w}^*)^T\phi(\mathbf{x}_j)\right)\Big]\,,
\end{aligned}$$

where the last inequality uses the definitions of R_ϕ and $R_{\phi_\mathcal{R}}$ respectively. Since $\ell_h(\cdot)$ is

1-Lipschitz, we obtain

$$\begin{aligned}\|\Delta \mathbf{w}\|^2 &\leq \frac{1}{\lambda m_+ m_-} \sum_{\{i:y_i=+1\}} \sum_{\{j:y_j=-1\}} (\|\mathbf{w}^*\| + \|\mathbf{w}^*_{\mathcal{R}}\|) \Big(\|\phi(\mathbf{x}_i) - \phi_{\mathcal{R}}(\mathbf{x}_i)\| + \|\phi(\mathbf{x}_j) - \phi_{\mathcal{R}}(\mathbf{x}_j)\| \Big) \\ &= \frac{\|\mathbf{w}^*\| + \|\mathbf{w}^*_{\mathcal{R}}\|}{\lambda} \left(\sum_{\{i:y_i=+1\}} \frac{\|\phi(\mathbf{x}_i) - \phi_{\mathcal{R}}(\mathbf{x}_i)\|}{m_+} + \sum_{\{j:y_j=-1\}} \frac{\|\phi(\mathbf{x}_j) - \phi_{\mathcal{R}}(\mathbf{x}_j)\|}{m_-} \right) .\end{aligned} \tag{4.27}$$

From $\phi(\mathbf{x}) = \phi_{\mathcal{R}}(\mathbf{x}) + \phi^{\perp}_{\mathcal{R}}(\mathbf{x})$, with $\phi_{\mathcal{R}}(\mathbf{x}) \in \mathcal{S}_{\mathcal{R}}$ and $\phi^{\perp}_{\mathcal{R}}(\mathbf{x})$ is in the space orthogonal to $\mathcal{S}_{\mathcal{R}}$, we have, for $i = 1, \ldots, m$,

$$\|\phi(\mathbf{x}_i) - \phi_{\mathcal{R}}(\mathbf{x}_i)\| = \|\phi^{\perp}_{\mathcal{R}}(\mathbf{x}_i)\| \leq \begin{cases} \|\phi(\mathbf{x}_i)\| = \sqrt{k(\mathbf{x}_i, \mathbf{x}_i)} \leq \sqrt{\kappa}, & \text{if } i \notin \mathcal{R} \\ 0, & \text{if } i \in \mathcal{R} . \end{cases} \tag{4.28}$$

In addition, recall that RankSVM is equivalent to a 1-class SVM on an enlarged dataset with the set of points $\mathcal{P} = \{\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j) : y_i > y_j, i, j = 1 \ldots, m\}$. Therefore $\mathbf{w}$ can be expressed in terms of the dual variables $0 \leq \alpha^*_{ij} \leq C$ of an SVM problem trained on $\mathcal{P}$ with $C = \frac{1}{\lambda m_+ m_-}$, as follows,

$$\begin{aligned}\mathbf{w}^* &= \sum_{\{i,j:y_i>y_j\}} \alpha^*_{ij} (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)) = \sum_{\{i:y_i=+1\}} \sum_{\{j:y_j=-1\}} \alpha^*_{ij} (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)) \\ &= \sum_{\{i:y_i=+1\}} \phi(\mathbf{x}_i) \left(\sum_{\{j:y_j=-1\}} \alpha^*_{ij} \right) - \sum_{\{j:y_j=-1\}} \phi(\mathbf{x}_j) \left(\sum_{\{i:y_i=+1\}} \alpha^*_{ij} \right) .\end{aligned}$$

Since $\|\phi(\mathbf{x})\| \leq \sqrt{\kappa}$ and $C = \frac{1}{\lambda m_+ m_-}$, we get $\|\mathbf{w}^*\| \leq \sqrt{\kappa} C m_- m_+ + \sqrt{\kappa} C m_+ m_- = \frac{2\sqrt{\kappa}}{\lambda}$. Similarly, $\|\phi_{\mathcal{R}}(\mathbf{x})\| \leq \|\phi(\mathbf{x})\| \leq \sqrt{\kappa}$ and $\|\mathbf{w}^*_{\mathcal{R}}\| \leq \frac{2\sqrt{\kappa}}{\lambda}$. Together with (4.28), we can then bound (4.27) by

$$\|\Delta \mathbf{w}\|^2 \leq \frac{4\kappa}{\lambda^2} \left(\sum_{\{i:y_i=+1\}} \frac{\mathbb{I}[i \notin \mathcal{R}]}{m_+} + \sum_{\{j:y_j=-1\}} \frac{\mathbb{I}[j \notin \mathcal{R}]}{m_-} \right) .$$

Therefore, we obtain

$$\begin{aligned}
|f_{\phi_{\mathcal{R}}}(\mathbf{x}) - f_{\phi}(\mathbf{x})| &= |\mathbf{w}_{\mathcal{R}}^T \phi_{\mathcal{R}}(\mathbf{x}) - \mathbf{w}^T \phi(\mathbf{x})| \\
&= |\mathbf{w}_{\mathcal{R}}^T \left(\phi(\mathbf{x}) - \phi_{\mathcal{R}}^{\perp}(\mathbf{x}) \right) - \mathbf{w}^T \phi(\mathbf{x})| \\
&= |\Delta \mathbf{w}^T \phi(\mathbf{x}) - \mathbf{w}_{\mathcal{R}}^T \phi_{\mathcal{R}}^{\perp}(\mathbf{x})| \\
&= |\Delta \mathbf{w}^T \phi(\mathbf{x})| \\
&\leq \|\Delta \mathbf{w}\| \|\phi(\mathbf{x})\| \\
&\leq \frac{2\kappa}{\lambda} \left(\sum_{\{i:y_i=+1\}} \frac{\mathbb{I}[i \notin \mathcal{R}]}{m_+} + \sum_{\{j:y_j=-1\}} \frac{\mathbb{I}[j \notin \mathcal{R}]}{m_-} \right)^{\frac{1}{2}},
\end{aligned}$$

where we have used $\mathbf{w}_{\mathcal{R}}^T \phi_{\mathcal{R}}^{\perp}(\mathbf{x}) = 0$ in the third equality since $\mathbf{w}_{\mathcal{R}} \in \mathcal{S}_{\mathcal{R}}$. This completes the proof. □

The following result is a direct consequence of Theorem 5 and Theorem 6.

Corollary 7. *For a feature map, $\phi : \mathcal{X} \to \mathcal{F} \subseteq \mathbb{R}^{d'}$ associated with kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, let $f_{\phi}^*(\mathbf{x})$ be the optimal RankSVM hypothesis obtained by solving* (4.18)*, and $\bar{f}_{\phi}^*(\mathbf{x})$ be the optimal hypothesis obtained by solving* (4.19)*, in which the hypothesis is restricted to an arbitrary subset of kernel functions indexed by $\mathcal{R} \subseteq \{1, ..., m\}$. Assume there exists $\kappa > 0$ such that $k(\mathbf{x}, \mathbf{x}) \leq \kappa$. Then the following inequality holds,*

$$|\bar{f}_{\phi}^*(\mathbf{x}) - f_{\phi}^*(\mathbf{x})| \leq \frac{2\kappa}{\lambda} \left(\sum_{\{i:y_i=+1\}} \frac{\mathbb{I}[i \notin \mathcal{R}]}{m_+} + \sum_{\{j:y_j=-1\}} \frac{\mathbb{I}[j \notin \mathcal{R}]}{m_-} \right)^{\frac{1}{2}}, \quad \forall \mathbf{x} \in \mathcal{X}. \tag{4.29}$$

□

Therefore, for the ranking loss, the bound (4.29) decreases asymmetrically depending on whether we include a point from the positive or negative class. In particular, if the dataset is unbalanced with $m_- \gg m_+$, or $\frac{1}{m_+} \gg \frac{1}{m_-}$, then the reduction obtained from including a positive class kernel function is much greater than including one from the negative class. Hence, for a fixed number of kernel functions, the bound is minimized by first including kernel functions corresponding to the positive or rare class.

4.2 Relation to Nyström Approximation

The Nyström method approximates a symmetric positive semi-definite matrix $Q \in \mathbb{R}^{m \times m}$ by a sample submatrix D of $n \ll m$ columns from Q [e.g. see 9, 105]. Without loss of generality, assume that the first n columns are the randomly chosen samples. Then D and Q can be written as

$$D = \begin{bmatrix} A \\ B \end{bmatrix} \quad \text{and} \quad Q = \begin{bmatrix} A & B^T \\ B & C \end{bmatrix},$$

with $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{(m-n) \times n}$, and $C \in \mathbb{R}^{(m-n) \times (m-n)}$. The Nyström method computes a rank-n approximation of Q as

$$\hat{Q} = DA^{\dagger}D^T = \begin{bmatrix} A & B^T \\ B & BA^{\dagger}B^T \end{bmatrix},$$

where $A^{\dagger}$ is the Moore-Penrose pseudoinverse of A. Thus, the Nyström method approximates Q with $\hat{Q}$ (or more specifically, C with $BA^{\dagger}B^T$) and can be seen as a method to complete matrix Q using information from only n columns.

Approximating a kernel matrix with a low-rank structured matrix to improve computational efficiency has been explored in the context of other kernel algorithms before. For instance, low-rank approximations have been used to speed up kernel PCA [6], multi-dimensional scaling [76], spectral clustering [43], manifold learning [87], Gaussian processes [105], and support vector machines [42, 85, 110].

4.2.1 Nyström Method Equivalence

In this section, we show that solving a regularized loss minimization with the $\mathcal{R}$-subset representation, is equivalent to solving the full unrestricted problem using a low-rank Nyström approximation of the kernel matrix. We show this is true for an arbitrary loss function.

Consider the regularized loss minimization problem (4.1) with a general loss function, $L : \mathbb{R}^m \to \mathbb{R}$. By substituting the solution (4.2) in (4.1), we can express the general regularized loss minimization problem (4.1) in terms of the kernel matrix, $K \in \mathbb{R}^{m \times m}$, and model

variables, $\boldsymbol{\beta} \in \mathbb{R}^m$,

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^m} \quad L(K\boldsymbol{\beta}) + \frac{\lambda}{2} \boldsymbol{\beta}^T K \boldsymbol{\beta} \,. \tag{4.30}$$

Here, $K\boldsymbol{\beta} = [f_\phi(\mathbf{x}_1), ..., f_\phi(\mathbf{x}_m)]^T \in \mathbb{R}^m$, where $f_\phi(\mathbf{x}) = \sum_{i=1}^m \beta_i k(\mathbf{x}_i, \mathbf{x})$.

Similarly, substituting the $\mathcal{R}$-subset hypothesis (4.3) in (4.1), results in the following problem with model variables, $\boldsymbol{\beta} \in \mathbb{R}^{|\mathcal{R}|}$,

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^{|\mathcal{R}|}} \quad L(K_{\bullet\mathcal{R}}\boldsymbol{\beta}) + \frac{\lambda}{2} \boldsymbol{\beta}^T K_{\mathcal{R}\mathcal{R}} \boldsymbol{\beta} \,. \tag{4.31}$$

Here $K_{\bullet\mathcal{R}} = [k_{ij}]_{i=1...m, j \in \mathcal{R}} \in \mathbb{R}^{m \times |\mathcal{R}|}$, $k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, is a subset of columns from K, and $K_{\mathcal{R}\mathcal{R}} = [k_{ij}]_{i,j \in \mathcal{R}} \in \mathbb{R}^{|\mathcal{R}| \times |\mathcal{R}|}$ is the square submatrix of K indexed by $\mathcal{R}$ along columns and rows. We have $K_{\bullet\mathcal{R}}\boldsymbol{\beta} = [\bar{f}_\phi(\mathbf{x}_1), ..., \bar{f}_\phi(\mathbf{x}_m)]^T \in \mathbb{R}^m$, where the hypothesis, $\bar{f}_\phi(\mathbf{x}) = \sum_{i \in \mathcal{R}} \beta_i k(\mathbf{x}_i, \mathbf{x})$, is restricted to use a subset of kernel functions indexed by $\mathcal{R} \subseteq \{1, ..., m\}$.

In the following proposition, we show that problem (4.31) is equivalent to problem (4.30), where the kernel matrix K is replaced by a Nyström approximation, $K' \in \mathbb{R}^{m \times m}$.

Proposition 8. *For any loss function $L : \mathbb{R}^m \to \mathbb{R}$ and kernel matrix $K \in \mathbb{R}^{m \times m}$, the regularized loss minimization problem* (4.31)*, in which the hypothesis is restricted to a subset of kernel functions indexed by $\mathcal{R} \subseteq \{1, ..., m\}$, is equivalent to the unrestricted problem* (4.30) *under a perturbed kernel matrix corresponding to the Nyström approximation, $K' = K_{\bullet\mathcal{R}} K_{\mathcal{R}\mathcal{R}}^\dagger K_{\bullet\mathcal{R}}^T \in \mathbb{R}^{m \times m}$, where $K_{\bullet\mathcal{R}} \in \mathbb{R}^{m \times |\mathcal{R}|}$ are columns of K indexed by $\mathcal{R}$, and $K_{\mathcal{R}\mathcal{R}} \in \mathbb{R}^{|\mathcal{R}| \times |\mathcal{R}|}$ are rows of $K_{\bullet\mathcal{R}}$ indexed by $\mathcal{R}$.*

Proof. Since $K_{\mathcal{R}\mathcal{R}}$ is positive semi-definite, using eigen-decomposition,

$$K_{\mathcal{R}\mathcal{R}} = U \Lambda U^T \,,$$

where U is an orthonormal matrix and Λ is a diagonal matrix of non-negative eigenvalues of $K_{\mathcal{R}\mathcal{R}}$.

Define $\mathbf{w} = \Lambda^{\frac{1}{2}} U^T \boldsymbol{\beta}$. Then $\boldsymbol{\beta} = U \Lambda^{\dagger\frac{1}{2}} \mathbf{w}$, and we can express (4.31) in terms of $\mathbf{w}$ as,

$$\min_{\mathbf{w} \in \mathbb{R}^{|\mathcal{R}|}} \quad L\left(K_{\bullet\mathcal{R}} U \Lambda^{\dagger\frac{1}{2}} \mathbf{w}\right) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 \,. \tag{4.32}$$

We recognize (4.32) as a problem in linear space with data points given by the rows of $K_{\bullet\mathcal{R}} U \Lambda^{*\frac{1}{2}} \in \mathbb{R}^{m \times |\mathcal{R}|}$.

Denote $[\phi_{\mathcal{N}}(\mathbf{x}_1), ..., \phi_{\mathcal{N}}(\mathbf{x}_m)]^T = K_{\bullet\mathcal{R}} U \Lambda^{\dagger\frac{1}{2}}$. Applying the Representer Theorem, the solution $f_{\phi_{\mathcal{N}}}(\mathbf{x}) = \mathbf{w}^T \phi_{\mathcal{N}}(\mathbf{x})$ can be expressed in the form,

$$f_{\phi_{\mathcal{N}}}(\mathbf{x}) = \left(\sum_{i=1}^{m} \beta_i \phi_{\mathcal{N}}(\mathbf{x}_i) \right)^T \phi_{\mathcal{N}}(\mathbf{x}), \text{ or } \mathbf{w} = \sum_{i=1}^{m} \beta_i \phi_{\mathcal{N}}(\mathbf{x}_i) \,, \tag{4.33}$$

Substituting (4.33) in problem (4.32) yields an equivalent problem,

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^m} \quad L\left(K'\boldsymbol{\beta}\right) + \frac{\lambda}{2} \boldsymbol{\beta}^T K' \boldsymbol{\beta} \,,$$

where $K' = \left(K_{\bullet\mathcal{R}} U \Lambda^{\dagger\frac{1}{2}}\right)\left(K_{\bullet\mathcal{R}} U \Lambda^{\dagger\frac{1}{2}}\right)^T = K_{\bullet\mathcal{R}} K_{\mathcal{R}\mathcal{R}}^{\dagger} K_{\bullet\mathcal{R}}^T \in \mathbb{R}^{m \times m}$, which we recognize as the Nyström approximation of K using the columns indexed by $\mathcal{R}$ as samples. The proof is completed. □

Proposition 8 formalizes the connection between selecting a set of points for the hypothesis representation, and using a low-rank Nyström approximation kernel for any regularized loss minimization problem which can be written in form (4.30). Conversely, it also shows that using a low-rank Nyström approximation kernel matrix can be viewed as selecting a $\mathcal{R}$-subset representable optimal hypothesis for problem (4.30). Thus, for problems which use a Nyström approximation kernel, Proposition 8 provides an efficient optimization formulation in the form (4.31), or in the linear space form (4.32), reducing problem dimension from m variables to $|\mathcal{R}|$ and space from $O(m^2)$ to $O(m|\mathcal{R}|)$.

4.2.2 Nyström Approximation Bound for SVM

Theoretically, Theorem 5 and Proposition 8 together imply that using a Nyström kernel approximation is equivalent to projecting the feature map ϕ onto the subspace spanned by the subset of samples in feature space. This relationship can potentially be used to analyze Nyström approximation algorithms based on a feature map projection, as in Theorem 5 & 6. For instance, Theorem 9 illustrates a stability bound that can be obtained for a regular SVM trained with a Nyström kernel approximation.

Theorem 9. *Let f_K^* denote the optimal hypothesis obtained by solving the SVM problem*

with a kernel matrix, $K = [k(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1}^m \in \mathbb{R}^{m\times m}$,

$$\min_{\beta\in\mathbb{R}^m} \quad \frac{1}{m}\sum_{i=1}^m \ell_h(y_i K_{i\bullet}\beta) + \frac{\lambda}{2}\beta^T K\beta \,, \tag{4.34}$$

where $K_{i\bullet} \in \mathbb{R}^{1\times m}$ *is the* i^{th} *row of* K. *Define a perturbed kernel matrix,* $K' \in \mathbb{R}^{m\times m}$, *as the Nyström approximation of* K *using the columns of* K *indexed by* $\mathcal{R} \subseteq \{1,...,m\}$. *Let* $f^*_{K'}$ *denote the optimal hypothesis obtained by solving the SVM problem* (4.34) *with the Nyström approximation* K'. *Then the following inequality holds,*

$$|f^*_{K'}(\mathbf{x}) - f^*_K(\mathbf{x})| \le \frac{\sqrt{2}\kappa}{\lambda}\sqrt{1-\frac{|\mathcal{R}|}{m}} \,, \quad \forall \mathbf{x}\in\mathcal{X}.$$

Proof. Following Proposition 8, using the Nyström approximation, K', in the SVM problem (4.34), is equivalent to

$$\min_{\beta\in\mathbb{R}^{\mathcal{R}}} \quad \frac{1}{m}\sum_{i=1}^m \ell_h(y_i K_{i\mathcal{R}}\beta) + \frac{\lambda}{2}\beta^T K_{\mathcal{R}\mathcal{R}}\beta \,, \tag{4.35}$$

where $K_{i\mathcal{R}}$ is the ith row of $K_{\bullet\mathcal{R}}$. Problem (4.35) solves SVM with a hypothesis restricted to a subset of kernel functions. Let ϕ denote the feature map corresponding to k. Then from Theorem 5, the optimal hypothesis $f^*_{K'}(\mathbf{x})$ equals the optimal SVM hypothesis under the projected mapping $\phi_{\mathcal{R}} = \text{Proj}_{\mathcal{S}_{\mathcal{R}}}(\phi)$. Consequently, the difference between the optimal hypotheses, f^*_K and $f^*_{K'}$, can be bounded following the proof of Theorem 6; the only difference is that, instead of the ranking loss function, R_ϕ, we have the SVM loss, $\frac{1}{m}\sum_{i=1}^m \ell_h\left(y_i \mathbf{w}^T\phi(\mathbf{x}_i)\right)$. This means that, instead of (4.27), we have

$$\|\Delta\mathbf{w}\|^2 \le \frac{\|\mathbf{w}^*\| + \|\mathbf{w}^*_{\mathcal{R}}\|}{\lambda}\sum_{i=1}^m \frac{\|\phi(\mathbf{x}_i) - \phi_{\mathcal{R}}(\mathbf{x}_i)\|}{m} \,.$$

Similarly, it can be shown that $\|\mathbf{w}^*\| \le \frac{\sqrt{\kappa}}{\lambda}$, $\|\mathbf{w}^*_{\mathcal{R}}\| \le \frac{\sqrt{\kappa}}{\lambda}$ and consequently

$$|f^*_{K'}(\mathbf{x}) - f^*_K(\mathbf{x})| \le \frac{\sqrt{2}\kappa}{\lambda}\left(\frac{\sum_{i=1}^m \mathbb{I}[i\notin\mathcal{R}]}{m}\right)^{\frac{1}{2}} \,, \quad \forall \mathbf{x}\in\mathcal{X}.$$

□

4.2.3 Comparison to Kernel Perturbation Bounds

Cortes et al. [32] obtain a stability bound for SVM assuming an arbitrary kernel matrix perturbation. They use the bound to analyze Nyström kernel approximations. The bound obtained in Cortes et al. [32] is a function of the spectral norm of the difference between the two kernel matrices, ie. $||K' - K||_2$. In comparison, the bound obtained in Theorem 9, based on the feature map projection for a Nyström approximation, is much simpler: it is proportional to the square root of the percentage of the points not in the hypothesis representation. In addition, since the Nyström approximation, K', is computed using the pseudo inverse of kernel submatrix, $K_{\mathcal{R}\mathcal{R}}$, it can become arbitrarily far away from K, depending on the condition number of $K_{\mathcal{R}\mathcal{R}}$. In contrast, the projected map approach offers a more stable, and often tighter, bound.

To demonstrate this, we compare bounds for the ranking loss problem obtained using the feature map projection and kernel matrix perturbation approaches. Below we state the stability bound for RankSVM under an arbitrary kernel perturbation following the approach in Cortes et al. [32].

Theorem 10. *Let f_K^* and $f_{K'}^*$ denote the optimal hypothesis obtained by RankSVM when using the kernel matrix $K \in \mathbb{R}^{m\times m}$ and $K' \in \mathbb{R}^{m\times m}$, respectively. Then the following inequality holds for all $\mathbf{x} \in \mathcal{X}$:*

$$|f_{K'}^*(\mathbf{x}) - f_K^*(\mathbf{x})| \leq \frac{2\sqrt{2}\kappa^{\frac{3}{4}}}{\lambda} \|K' - K\|_2^{\frac{1}{2}} \left[1 + \left(\frac{\|K' - K\|_2}{4\kappa}\right)^{\frac{1}{4}}\right] . \tag{4.36}$$

The proof is very similar to that in Cortes et al. [32]. The idea is to use an explicit $(m+1)$-dimension feature map ϕ and ϕ' associated with K and K' defined according to

$$\phi(\mathbf{x}_i) = K_{m+1}^{\frac{1}{2}} \mathbf{e}_i \quad \text{and} \quad \phi'(\mathbf{x}_i) = K_{m+1}'^{\frac{1}{2}} \mathbf{e}_i \, ,$$

where K_{m+1} and K'_{m+1} are augmented versions of K and K' with the $(m+1)$th point representing an arbitrary test point, and $\mathbf{e}_i \in \mathbb{R}^{m+1}$ is a unit vector, with the ith component equal to 1, and 0 everywhere else. Then using the fact the solution is at a minimizer and the objective is convex (as done in Theorem 6), $\|\Delta \mathbf{w}\|^2$ can be bounded in terms of $\|\phi(\mathbf{x}_i) - \phi'(\mathbf{x}_i)\|_2 \leq \|K_{m+1}^{\frac{1}{2}} - K_{m+1}'^{\frac{1}{2}}\|_2 \leq \|K_{m+1} - K'_{m+1}\|_2^{\frac{1}{2}} = \|K - K'\|_2^{\frac{1}{2}}$, which can then be used to obtain the final result. The bound obtained for RankSVM under a perturbed kernel matrix is simply twice that obtained in Cortes et al. [32] for a regular SVM. The factor of two

Name	Source	Subject	d	m	m_+	ρ
Page0	Keel	Computer	10	5472	559	10.2%
Satellite	UCI	Nature	36	6435	626	9.7%
Coil	KDD	Business	85	9822	586	6.0%
Mammograph	[106]	Life	6	11183	260	2.3%

Table 4.1: List of datasets and their characteristics used in Figure 4.1. d is the number of features, m is the total number of observations, m_+ is the number of rare class observations, and $\rho = \frac{m_+}{m}$ is the percentage of rare class examples.

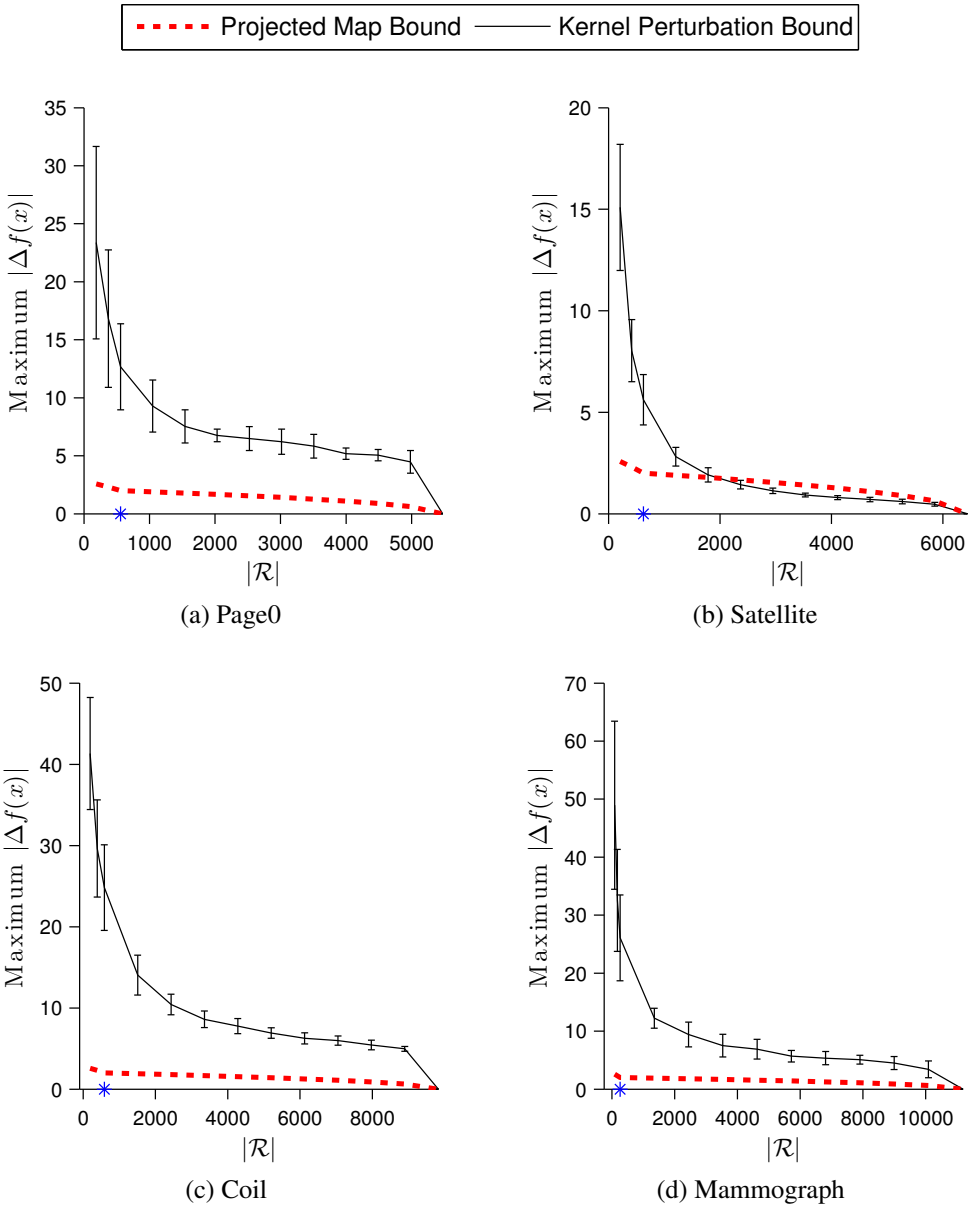
emerges due to the double summation in the ranking loss function.

From Proposition 8, we can bound the difference between the RankSVM classifier and a $\mathcal{R}$-subset RankSVM classifier by comparing the effect of perturbing the kernel matrix to its Nyström approximation. Thus, bound (4.36) also applies to the difference between the RankSVM classifier and a $\mathcal{R}$-subset RankSVM classifier by setting $K' = K_{\bullet\mathcal{R}} K_{\mathcal{R}\mathcal{R}}^{\dagger} K_{\bullet\mathcal{R}}^{T}$.

Figure 4.1 compares bound (4.36) with the projected map bound (4.21) obtained in Theorem 6 as $|\mathcal{R}|$ is increased on four unbalanced datasets described in Table 4.1. For the setup, we assume a Gaussian kernel, with width $\sigma^2 = \frac{1}{m^2}\sum_{i,j=1}^{m} ||\mathbf{x}_i - \mathbf{x}_j||_2^2$. Since we are using a Gaussian kernel, $\kappa = 1$. We choose $\lambda = 1$, it does not affect the comparison. For bound (4.21), we assume kernel functions corresponding to rare class points are included in the representation first. This leads to a deterministic trajectory as $|\mathcal{R}|$ is increased for each dataset. For bound (4.36), we randomly sample $|\mathcal{R}|$ columns 40 times. For each sample we compute K' to be used in (4.36). We show the mean and standard deviation of the bound for each value of $|\mathcal{R}|$. From Figure 4.1 it is clear that the projected map based bound can be significantly lower than the kernel perturbation bound, particularly when $|\mathcal{R}| \ll m$.

Note, the kernel perturbation bound (4.36) does not depend on class label information. To minimize (4.36), we need to minimize $||K' - K||_2$, where K' is a Nyström approximation. We can approach this using any one of the various strategies available in the literature for landmark selection in the Nyström method [e.g see 41, 85, 111]. However, better landmark selection is generally achieved at the expense of higher space and time costs. As a result uniform random sampling without replacement remains the method most commonly used in practice [63].

In contrast, the projected map bound (4.21) uses class label information and captures



(a) Page0 (b) Satellite

(c) Coil (d) Mammograph

Figure 4.1: Comparison of the projected map bound (4.21) (Theorem 6) and the kernel perturbation bound (4.36) (Theorem 10) for RankRC as $|\mathcal{R}|$ is increased. The projected map bound is computed assuming rare class points are used in the hypothesis first. The kernel perturbation bound is obtained by randomly sampling a set of basis functions 40 times. The mean value of the bound is plotted and standard deviation is shown as error bars. The blue '*' on the x-axis indicates the number of rare (positive) examples in the dataset.

the asymmetry associated with an unbalanced RankSVM problem. The result leads to a simple selection strategy: to include kernel functions corresponding to the rare class points first. This is compatible with the motivation presented in Section 3.4 for RankRC. Computational results for RankRC, presented in Chapter 5, confirm that the rare class representation performs better than an equal number of randomly selected points for unbalanced ranking problems.

Finally, one could consider selection methods that combine both insights. For example, for big datasets, we can select $|\mathcal{R}| < m_+$ points from the positive class using a more sophisticated landmark selection strategy than random sampling. In this case, the extra selection expense may be more acceptable, since we are restricting ourselves to a smaller set of columns, $m_+ \ll m$. On the other hand, if we are interested in selecting $|\mathcal{R}| > m_+$ kernel functions for the hypothesis representation, we can first select the rare class points, and then randomly sample the remaining $|\mathcal{R}| - m_+$ points from the majority class.

4.3 Summary

In this chapter, we analyze the solution of RankRC and compare it to the solution of RankSVM. More generally, we consider an arbitrary loss minimization problem, and examine the effect of restricting the hypothesis to any subset of kernel functions ($\mathcal{R}$-subset representation). We show that restricting the hypothesis to a $\mathcal{R}$-subset representation is equivalent to using a projected feature map while solving the unrestricted problem. We use this result to conduct a stability analysis of the $\mathcal{R}$-subset hypothesis for RankSVM. The resulting bound is proportional to $\sqrt{p_+ + p_-}$ where, p_+ is the percentage of points in the positive (rare) class and not in $\mathcal{R}$ and p_- is the percentage of the points in the negative (majority) class but not in $\mathcal{R}$. Therefore, for a fixed cardinality $|\mathcal{R}|$, this bound is minimized by including as many rare class points in the $\mathcal{R}$-subset representation as possible. This result provides further theoretical justification for the RankRC algorithm proposed in Chapter 3.

In addition, we show that using a $\mathcal{R}$-subset representation is equivalent to solving the original regularized loss minimization problem with a Nyström approximation of the kernel matrix. The Nyström approximation is formed using columns indexed by the set $\mathcal{R}$. This implies that RankRC can be considered as a special Nyström approximation method for RankSVM, with columns selected from the rare class only. Another implication is that we can obtain stability bounds for the $\mathcal{R}$-subset representation using a kernel perturbation approach. However, bounds obtained using the kernel perturbation approach for a Nyström

approximation can be arbitrary large. In contrast, the analysis using the projected feature map approach leads to more stable and tighter bounds. We illustrate this behavior computationally, by comparing bounds obtained for the RankSVM $\mathcal{R}$-subset classifier using the two different approaches.

Although our motivation has been to analyze RankRC, we note that the results we obtain on the equivalency of using a $\mathcal{R}$-subset classifier, a projected feature map, and a Nyström kernel approximation are quite general. These relationships can be used to analyze and devise algorithms for other approximate kernel problems as well.

Chapter 5

RankRC: Computational Results

In this chapter we empirically compare RankRC to other methods on several simulated and real unbalanced problems.

5.1 Methods and Experiment Setup

The following methods are compared:

1. KNN: k-Nearest-Neighbors algorithm. The posterior probability is used as the ranking function:

$$P(y|\mathbf{x}) = \frac{1}{k}\sum_{i\in\mathcal{K}} \mathbb{I}[y_i = 1] \,,$$

   where $\mathcal{K}$ is the set of k nearest neighbors in the training dataset.

2. SVM: This is the standard nonlinear SVM [97], in which the primal problem,

$$\min_{\mathbf{w}\in\mathbb{R}^d} \quad \frac{1}{m}\sum_{i=1}^{m} \max\left(0, 1 - y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b)\right) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2 \,,$$

   is solved (in the dual) to obtain the decision function, $f(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b = \sum_{i=1}^{m}\beta_i k(\mathbf{x}_i, \mathbf{x}) + b$, with $k(\mathbf{x}_i, \mathbf{x}) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x})$.

3. SVM-W: Weighted SVM [74, 97] in which

$$\min_{\mathbf{w}\in\mathbb{R}^d} \quad \frac{1}{m}\sum_{i=1}^{m}\omega_i \max\left(0, 1-y_i(\mathbf{w}^T\phi(\mathbf{x}_i)+b)\right)+\frac{\lambda}{2}\|\mathbf{w}\|_2^2\,,$$

is solved, with different weights associated with each class:

$$\omega_i = \begin{cases} \frac{m}{2m_+} & \text{if } y_i = +1 \\ \frac{m}{2m_-} & \text{if } y_i = -1\,. \end{cases}$$

The idea is to penalize misclassification error of minority examples more heavily in order to reduce the bias towards majority examples.

4. SVM-RUS: Randomly Under Sample the majority class examples ($y = -1$) to match the number of minority examples [62]. The resulting dataset, with $2m_+$ points, is used to train a standard SVM.

5. SVM-SMT: Uses a Synthetic Minority Oversampling TEchnique (SMOTE) [28] in which the rare class is over-sampled by creating new synthetic rare class samples according to each rare class sample and its k nearest neighbors. Each new sample is generated in the direction of some or all of the nearest neighbors. We oversample to match the number of majority examples. The resulting dataset, with $2m_-$ points, is used to train a standard SVM.

6. RANK-SVM: Nonlinear RankSVM problem (3.7).

7. RANK-RND: We solve the regularized ranking problem constrained to m_+ randomly selected set of basis function, i.e. Problem (**??**) with $|\mathcal{R}| = m_+$ and a randomly chosen index set $\mathcal{R}$.

8. RANK-RC: We solve the regularized ranking problem with a Rare Class representation, i.e. Problem (3.15).

We use LIBSVM [24] to solve the SVM problems (2-5). LIBSVM is a popular and efficient implementation of the sequential minimal optimization algorithm [75]. We set cache size to 10GB to minimize cache misses; termination criteria and shrinking heuristics are used in their default settings. The ranking methods (6-8) are solved using the subspace-trust-region method as outlined in Section 3.5. Termination tolerance is set at 1e-6. For

ranking methods, the memory available to store the kernel matrix is limited to 10GB. Experiments are performed on a Xeon E5620@2.4Ghz running Linux.

All datasets are standardized to zero mean and unit variance before training. Since our focus is on nonlinear kernels, for all SVM and ranking methods (2-8), we use a Gaussian kernel, $k(\mathbf{u},\mathbf{v}) = \exp(-\|\mathbf{u}-\mathbf{v}\|_2^2/\sigma^2)$ with $\sigma^2 = \frac{1}{m^2}\sum_{i,j=1}^{m}\|\mathbf{x}_i-\mathbf{x}_j\|_2^2$. The penalty parameter λ is determined by cross-validation over values $\log_2\lambda = [-20,-18,...,8,10]$. For KNN we cross-validate over $k = [1,2,...,\lceil\min(100,\sqrt{m})\rceil]$.

5.2 Simulated Data

Simulated unbalanced datasets are generated in the following manner. Rare class instances are sampled from six multivariate normal distributions with equal probability. Their centers, μ_i, $i = 1,...,6$, are randomly chosen within a unit cube. The majority class is sampled from $\binom{6}{2} = 15$ multivariate normal distributions with equal probability. Their centers are chosen along lines connecting all combinations of two rare class centers, i.e. $t\mu_i+(1-t)\mu_j$, $i > j$. The parameter $t \in [0,1]$ is used to roughly control the degree of class overlap. All

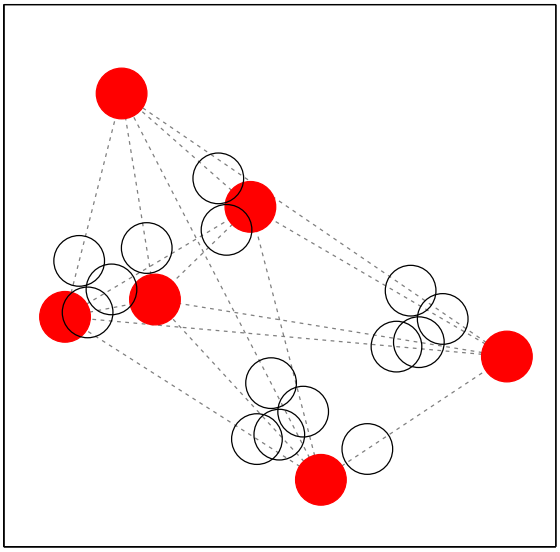


Figure 5.1: Example configuration of simulated dataset in 2-dimensions with $\sigma = 0.1$ and $t = 0.75$. The red, filled in circles show the locations of the six normal components for the rare class distribution. The black, empty circles show the location of the 15 normal components for the majority class distribution, whose centers lie along the dotted lines connecting all two rare class normal components.

covariances are chosen to be spherical, $\sigma^2 I$. Finally, the imbalance ratio, $\rho = \frac{m_+}{m}$, is used to determine the number of samples drawn from each of the class conditional distributions. An example configuration in 2-dimensional space is shown in Figure 5.1. The resulting dataset contains multiple rare-class subconcepts that vary in discriminative structure.

For our experiment we generate data in 5-dimensional space with $\sigma = 0.5$. We set t = 0.9, 0.75, and 0.6 to produce datasets with high, medium and low overlap, respectively. The imbalance ratio, ρ, is varied from 10% to 40% in 10% increments for each t value. Thus we have a total of 12 datasets. For each dataset we generate 1000 training points, 1000 validation points and 10000 testing points. Results are averaged over 10 trials.

Table 5.1 shows test AUC results using different methods. KNN does not perform as well as SVM and ranking methods. In general, ranking methods perform better than SVM based methods when there is greater overlap and higher imbalance (lower ρ). RANK-RND under performs in medium and low overlap datasets. In comparison, RANK-RC yields statistically similar performance as RANK-SVM across all datasets. Overall, both RANK-RC and RANK-SVM provide the best models.

Figure 5.2 compares the empirical ranking loss function,

$$\hat{R}_h = \frac{1}{m_+ m_-} \sum_{\{i:y_i=+1\}} \sum_{\{j:y_j=-1\}} \ell_h \left(f(\mathbf{x}_i) - f(\mathbf{x}_j) \right) ,$$

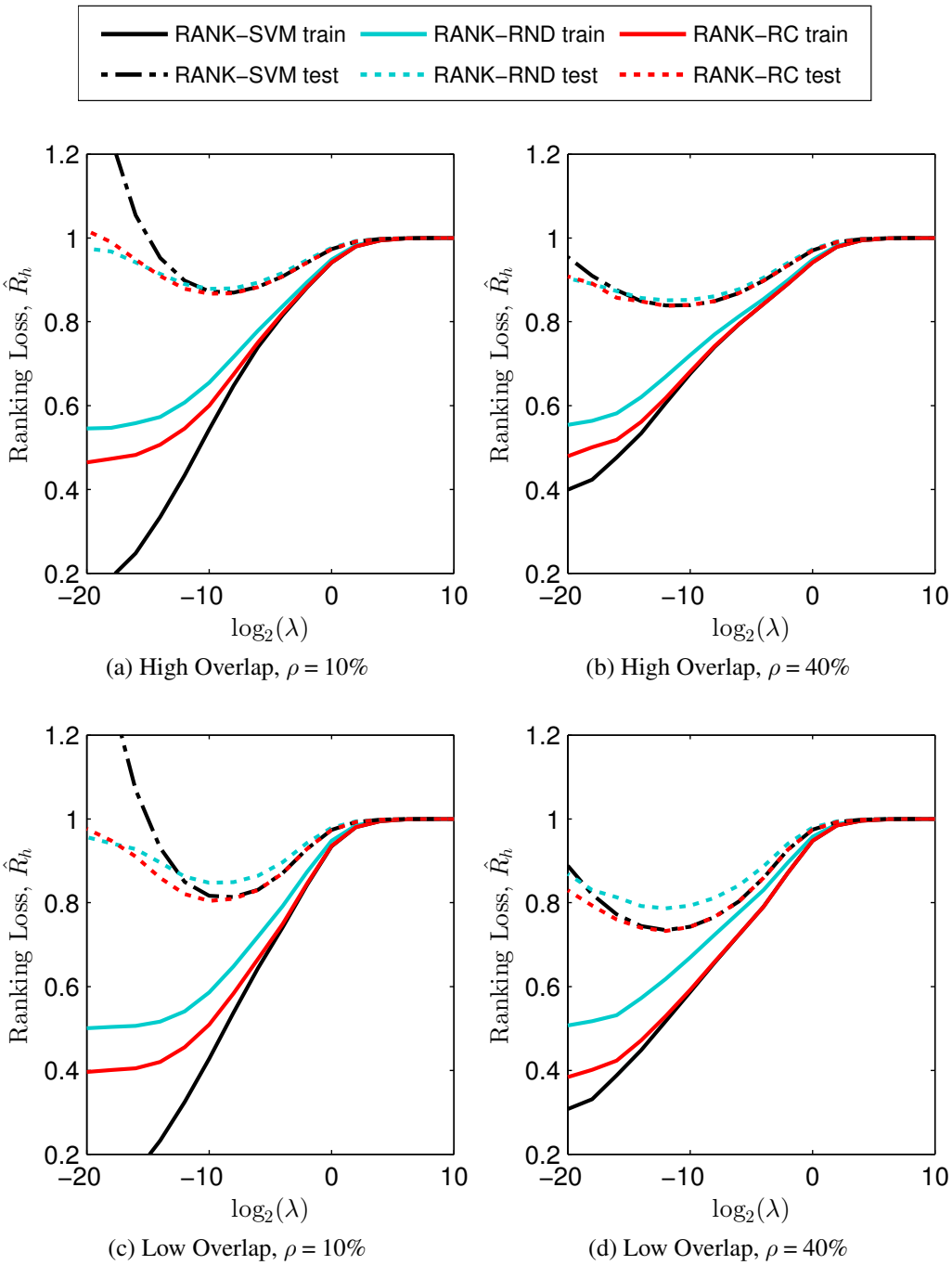
obtained by the ranking methods on four of the training and testing sets as λ is varied. We observe that the difference between RANK-SVM and the restricted basis models (RANK-RND and RANK-RC) decreases as λ is increased. Since restricting basis functions also limits the complexity of the model, the test loss of RANK-RND and RANK-RC is lower than that of RANK-SVM for small values of λ. However, RANK-RND is unable to achieve the optimal test loss levels at moderate values of λ (more noticeably in Figures 5.2c and 5.2d). In contrast, RANK-RC does not forfeit any test performance compared to RANK-SVM, while providing additional robustness as λ is reduced.

5.3 Real Datasets

In this section the methods are compared on several unbalanced real datasets obtained from various sources. Table 5.2 lists the datasets along with their characteristics. For each dataset, three-quarters of the observations are used for training and the remaining one-

Overlap	ρ	KNN	Classification Loss				Ranking Loss			True Bayes
			SVM	SVM-W	SVM-RUS	SVM-SMT	RANK-SVM	RANK-RND	RANK-RC	
High	10%	56.3±0.4	57.3±0.3	59.8±0.3	59.1±0.4	58.9±0.4	**61.5±0.3**	**61.4±0.3**	**61.4±0.2**	66.8
	20%	55.5±0.4	59.2±0.2	61.2±0.1	61.0±0.4	60.9±0.4	**61.6±0.3**	**61.3±0.4**	**62.3±0.3**	
	30%	57.3±1.0	61.1±0.2	**62.2±0.4**	62.2±0.1	61.8±0.3	**62.3±0.4**	**62.2±0.3**	**62.6±0.4**	
	40%	59.0±0.7	**62.8±0.2**	**63.2±0.3**	**63.3±0.2**	**62.8±0.2**	**62.7±0.3**	62.5±0.2	**62.7±0.3**	
Medium	10%	54.9±0.6	56.8±0.5	59.5±0.5	58.8±0.3	58.6±0.9	**61.4±0.5**	60.1±0.5	**61.4±0.4**	69.5
	20%	54.5±0.3	60.1±0.4	62.1±0.2	62.1±0.3	62.0±0.5	**62.8±0.4**	61.5±0.4	**63.4±0.2**	
	30%	57.6±0.8	63.4±0.1	**64.0±0.3**	63.3±0.1	63.4±0.4	**64.1±0.4**	62.4±0.5	**64.6±0.5**	
	40%	58.0±0.7	**65.3±0.2**	**65.5±0.1**	**65.4±0.1**	**65.3±0.2**	64.6±0.3	63.0±0.2	**64.9±0.3**	
Low	10%	55.9±1.0	61.1±0.6	64.0±0.4	63.5±0.2	63.0±0.8	**65.3±0.5**	62.7±0.6	**65.5±0.4**	74.4
	20%	57.2±0.4	64.2±0.3	66.6±0.2	66.2±0.2	66.4±0.2	**67.1±0.3**	63.5±0.4	**67.3±0.2**	
	30%	59.9±0.9	**69.1±0.4**	**69.4±0.2**	68.9±0.1	**69.2±0.2**	**69.2±0.4**	65.8±0.4	**69.8±0.4**	
	40%	61.6±1.5	**71.1±0.1**	**70.7±0.3**	70.5±0.1	**71.1±0.1**	69.8±0.3	65.8±0.3	69.9±0.3	

Table 5.1: Comparison of test AUC results for simulated datasets with high ($t = 0.9$), medium ($t = 0.75$) and low ($t = 0.6$) overlap, each with ρ =10%, 20%, 30% and 40% minority samples. Mean AUC score with standard error over 10 trials are shown. Bolded scores indicate the result is statistically not different than the best performing model using a two-tailed t-test with 99% confidence.



(a) High Overlap, $\rho = 10\%$

(b) High Overlap, $\rho = 40\%$

(c) Low Overlap, $\rho = 10\%$

(d) Low Overlap, $\rho = 40\%$

Figure 5.2: Comparison of empirical train and test ranking loss obtained by the ranking methods on four of the simulated datasets as λ is varied.

quarter for out-of-sample testing. Results are averaged over 20 stratified random splits of the data. The model parameter (λ or k) is tuned by running 10-fold cross-validation on the training set for each split.

Table 5.3 shows the mean test AUC score with standard error for each model. Overall, RANK-SVM and RANK-RC yield the best performing models with statistically similar results. RANK-RND, on the other hand, under performs on some datasets, indicating that a random basis set is not as effective as the rare class basis on unbalanced problems. SVM based methods generally do not perform as well as ranking methods, except when there appears to be more discriminative patterns in the data.

Table 5.4 compares the number of support vectors used by the SVM and ranking models. RANK-SVM uses more support vectors than SVM based models. It can use even more support vectors than SVM-SMT, which is trained on an enlarged dataset almost twice the size. This suggests that training RANK-SVM using a working-set type algorithm, which only tracks active support vectors (e.g. as proposed in [25] for standard SVMs), would still run costly in time and space. In comparison, RANK-RND and RANK-RC use significantly fewer support vectors. Moreover, with RANK-RC, test performance is also not compromised.

5.4 Intrusion Detection

In this section we use the KDD Cup 1999 dataset [4] to evaluate a large-scale unbalanced problem. The objective is to detect network intrusion by distinguishing between legitimate (normal) and illegitimate (attack) connections to a computer network. The dataset is a collection of simulated raw TCP dump data over a period of nine weeks on a local area network. The first seven weeks of data is used for training and the last two for test, providing a total of 4 898 431 training observations and 311 029 test cases. We processed the data to remove duplicate entries (as done in [89]) resulting in 1 074 975 training observations and 77 286 test cases. Each observation contains 41 features, three of which are categorical and the rest numerical. The three categorical features are protocol (3 categories), service (70 categories) and flag (11 categories). We represent protocol using three binary features, where each feature is an indicator for one of the three categories. Service and flag categories are replaced by the frequency in the training sample (i.e. probability) corresponding to the event of observing an attack given the category is present. Thus we obtain a total of 43 features. Finally, as done for all datasets, we standardize each feature to zero mean and

Name	Source	Subject	Features		Samples		
			Original	Derived (d)	m	m_+	ρ
Abalone19	UCI	Life	1N,7C	10	4177	32	0.8%
Mammograph	[106]	Life	6C	6	11183	260	2.3%
Ozone	UCI	Environment	72C	72	2536	73	2.9%
YeastME2	UCI	Life	8C	8	1484	51	3.4%
Wine4	UCI	Chemistry	11C	11	4898	183	3.7%
Oil	[61]	Environment	49C	49	937	41	4.4%
SolarM0	UCI	Nature	10N	32	1389	68	4.9%
Coil	KDD	Business	85C	85	9822	586	6.0%
Thyroid	UCI	Life	21N,7C	52	3772	231	6.1%
Libras	UCI	Physics	90C	90	360	24	6.7%
Scene	LibSVM	Nature	294C	294	2407	177	7.4%
YeastML8	LibSVM	Life	103C	103	2417	178	7.4%
Crime	UCI	Economics	122C	100	1994	150	7.5%
Vowel0	Keel	Computer	10C	10	989	90	9.1%
Euthyroid	UCI	Life	18N,7C	42	3163	293	9.3%
Abalone7	UCI	Life	1N,7C	10	4177	391	9.4%
Satellite	UCI	Nature	36C	36	6435	626	9.7%
Page0	Keel	Computer	10C	10	5472	559	10.2%
Ecoli	UCI	Life	7C	7	336	35	10.4%
Contra2	Keel	Life	9C	9	1473	333	22.6%

Table 5.2: List of datasets and their characteristics that we use to evaluate methods. Under original features, 'N' is used to denote number of nominal features, 'C', is used to denote number of continuous features. We derive d features by converting nominal features to an indicator representation and use continuous features as is. Under samples, m is the total number of observations, m_+ is the number of rare class observations, and $\rho = \frac{m_+}{m}$ is the percentage of rare class examples.

Dataset	KNN	Classification Loss				Ranking Loss		
		SVM	SVM-W	SVM-RUS	SVM-SMT	RANK-SVM	RANK-RND	RANK-RC
Abalone19	55.7±2.2	54.9±3.1	64.3±1.3	74.1±1.5	67.4±1.1	**81.0±1.2**	**79.1±1.1**	**81.4±1.1**
Mammograph	80.7±0.4	88.4±0.4	90.1±0.7	92.8±0.4	91.3±0.4	**93.7±0.4**	**93.9±0.4**	**94.4±0.3**
Ozone	66.4±2.2	85.0±1.1	85.5±0.7	86.4±0.9	85.9±0.8	**89.4±0.9**	**88.7±0.8**	**90.1±0.9**
YeastME2	69.3±1.7	81.8±0.5	85.5±0.7	**87.5±0.7**	**86.8±1.1**	**90.8±0.8**	**89.0±0.9**	**89.4±1.1**
Wine4	61.9±0.5	74.9±0.7	71.6±0.9	79.1±0.8	78.9±0.8	**83.5±0.6**	79.6±0.6	**82.7±0.7**
Oil	71.6±1.7	**91.1±0.9**	**88.0±1.4**	**90.6±0.8**	**90.6±1.0**	**92.5±0.9**	89.2±1.0	**91.7±0.8**
SolarM0	58.9±1.6	55.4±0.7	63.1±1.3	71.5±0.8	73.1±0.4	**78.5±0.5**	**77.2±0.8**	**77.5±0.8**
Coil	53.9±0.3	59.2±0.8	62.9±0.4	68.8±0.4	67.5±0.5	70.0±0.4	69.8±0.4	**72.3±0.2**
Thyroid	73.9±0.6	**94.8±0.4**	93.4±0.5	**94.8±0.3**	**94.4±0.4**	**95.7±0.4**	91.3±0.5	**95.7±0.3**
Libras	87.4±2.1	**96.8±0.9**	**96.7±0.9**	**96.4±0.8**	**96.8±0.9**	**97.6±0.8**	**95.4±1.0**	**94.8±1.1**
Scene	59.3±0.8	67.3±0.8	**75.4±0.9**	74.8±0.6	74.0±0.9	**77.1±0.7**	**76.4±0.8**	**77.5±0.6**
YeastML8	54.5±0.7	57.1±0.8	59.6±0.6	57.9±0.5	59.7±0.4	**61.5±0.5**	**60.2±0.7**	**62.0±0.5**
Crime	71.8±1.5	87.6±0.7	87.3±0.6	90.1±0.3	90.8±0.3	**92.3±0.3**	**91.2±0.3**	**91.6±0.3**
Vowel0	**100.0±0.0**	**100.0±0.0**	**100.0±0.0**	99.8±0.0	**100.0±0.0**	**100.0±0.0**	98.4±0.1	**100.0±0.0**
Euthyroid	75.8±0.8	**95.0±0.4**	**95.0±0.4**	**94.6±0.4**	**95.0±0.4**	**95.2±0.4**	90.7±0.4	**94.1±0.4**
Abalone7	78.2±2.1	56.1±3.2	76.3±0.5	77.4±0.3	74.4±0.2	**87.0±0.3**	**86.5±0.3**	**87.1±0.3**
Satellite	83.8±0.3	**94.8±0.1**	94.6±0.1	94.3±0.1	**95.1±0.1**	**95.3±0.1**	94.3±0.1	**95.1±0.1**
Page0	90.4±0.4	**98.4±0.1**	98.1±0.1	98.1±0.1	98.2±0.1	**98.6±0.1**	95.6±0.1	**98.4±0.1**
Ecoli	75.6±2.0	**94.6±0.7**	**93.7±0.7**	**94.1±0.6**	**93.2±0.6**	**94.1±0.6**	**93.4±0.9**	**94.5±0.7**
Contra2	60.5±0.9	66.9±0.8	70.2±0.5	70.5±0.6	70.6±0.8	**73.2±0.5**	**72.6±0.5**	**73.4±0.4**

Table 5.3: Comparison of test AUC results for real datasets (listed in Table 5.2). Mean AUC score with standard error over 20 trials are shown. Each trial uses one-quarter data for out-of-sample testing. Bolded scores indicate the result is statistically not different than the best performing model using a two-tailed t-test with 99% confidence.

Dataset	Classification Loss				Ranking Loss		
	SVM	SVM-W	SVM-RUS	SVM-SMT	RANK-SVM	RANK-RND	RANK-RC
Abalone19	117	1555	32	2644	2979	24	24
Mammograph	306	2152	119	2987	7252	195	193
Ozone	206	746	61	1165	995	55	55
YeastME2	105	429	41	594	1066	38	38
Wine4	458	2109	179	3166	3550	136	136
Oil	84	311	32	340	488	31	31
SolarM0	183	845	90	1114	1042	51	51
Coil	1754	5560	704	8178	7284	435	435
Thyroid	332	723	137	1020	2739	168	172
Libras	35	93	22	124	197	18	18
Scene	603	1171	213	1566	1748	132	133
YeastML8	833	1669	257	1562	1804	133	133
Crime	308	631	115	867	1326	112	112
Vowel0	35	37	25	45	730	68	67
Euthyroid	389	673	177	1002	2303	216	219
Abalone7	713	1391	274	2076	3079	291	292
Satellite	773	1158	301	1526	4734	466	469
Page0	322	570	145	924	4012	415	416
Ecoli	47	68	18	115	248	26	26
Contra2	560	843	396	912	1096	249	249

Table 5.4: Average number of support vectors used by the SVM and ranking models over 20 trials. For ranking models, support vectors are counted as the number of non-zero coefficients associated with kernel functions.

unit variance.

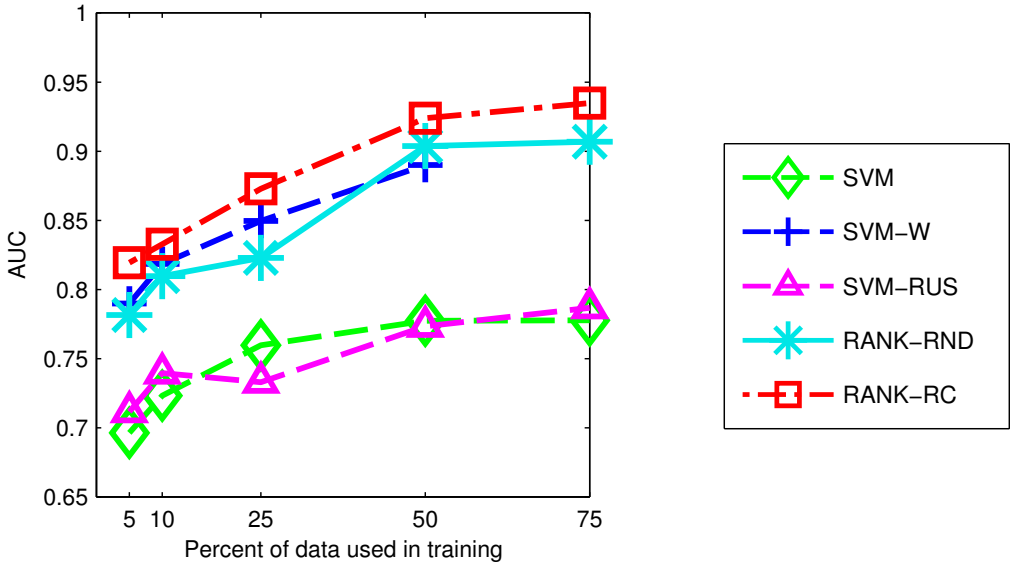
The attack types are grouped in four categories, DOS (Denial of Service), Probing (Surveillance, e.g. port scanning), U2R (user to root), R2L (remote to local). Table 5.5 shows the distribution of attack types in the training and test sets. Together, the U2R and R2L attacks constitute 4.0% of the test dataset, which is a substantial increase compared to the training set, but still a small fraction. Poor results have been reported in literature for identifying the U2R and R2L attacks [81]. In this experiment, we focus on identifying these attack types by forming a binary classification problem with the positive class representing either a U2R or R2L attack, and the negative class representing all other connection types. Thus the final training set is highly skewed with only 0.098% positive instances.

	Training		Test	
Normal	812808	75.6%	47913	62.0%
DOS	247266	23.0%	23568	30.5%
Probing	13850	1.3%	2677	3.5%
U2R	52	0.005%	215	0.278%
R2L	999	0.093%	2913	3.769%
Total	1074975	100%	77286	100%

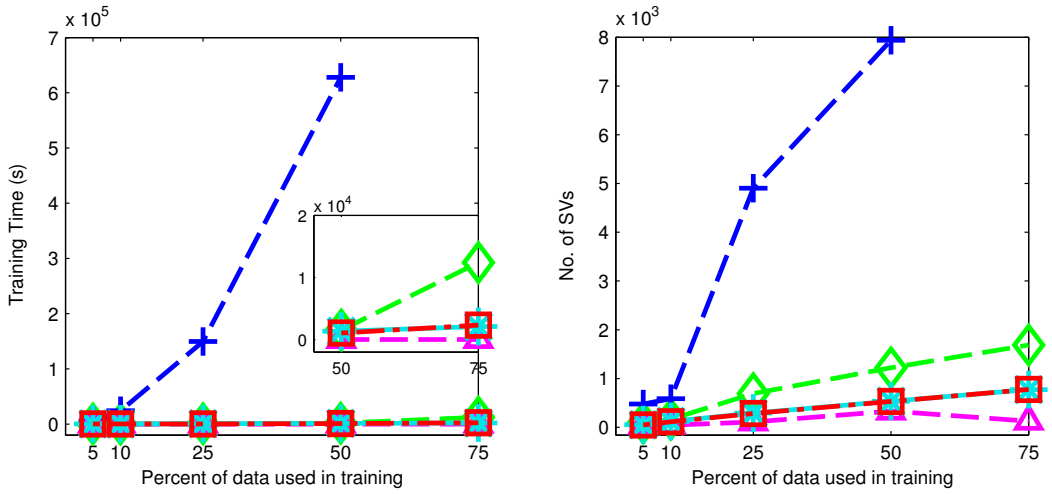
Table 5.5: Distribution of connection types in training and test sets for the intrusion detection problem.

We train using 5%, 10%, 25%, 50%, and 75% of the training data. The remaining training data is used for validation. We are unable to train RANK-SVM, even with just 5% of the data (53 749 samples), since the kernel matrix is too large to store in memory (>10GB). Clearly, this is an example where a large-scale solution is necessary to solve the ranking problem. We do not train SVM-SMT due to the large number of samples as well. We are able to train SVM-W using up to 50% of the data. With more samples SVM-W does not converge, due to the large number of support vectors which do not fit in the cache.

Figure 5.3a shows test AUC results obtained by different methods as training data is increased. We observe that SVM and SVM-RUS perform poorly. RANK-RC, RANK-RND and SVM-W produce better results, with RANK-RC performing the best. Figures 5.3b and 5.3c compare training time and number of support vectors, respectively, as training data is increased. SVM and SVM-RUS train in reasonable time, though they do not produce good models. On the other hand, SVM-W quickly becomes very expensive. RANK-RC and



(a)



(b) (c)

Figure 5.3: Comparison of (a) test AUC score, (b) training time in seconds, and (c) number of support vectors, for the intrusion detection problem as percent of data used for training is increased from 5% to 75%. In our experiment setup, we were unable to train RANK-SVM due to the large size of the dataset. Also, for more than 50% of data, SVM-W did not converge after more than 72 hours of training.

RANK-RND scale well, while able to produce effective models. RANK-RC and RANK-RND also use significantly fewer support vectors than SVM-W.

5.5 Summary

In this chapter we empirically evaluated RankRC on several unbalanced datasets and compared it to other methods, including RankSVM. RankRC perform similar to RankSVM, while outperforming other methods on rare class class problems. Compared to RankSVM, RankRC is scalable and can be used to efficiently solve much larger problems on regular machines.

Chapter 6

Multi-Level Rare Class Kernel Ranking

In this chapter, we extend the biclass RankRC formulation to multi-level ranking and apply it to a recent competition problem sponsored by the Heritage Health Provider Competition. The problem illustrates how RankRC can be used for ordinal regression where one ordinal level contains the vast majority of examples. We compare performance of RankRC with other methods and demonstrate computational and predictive advantages.

6.1 Predicting Days in Hospital

In many prediction problems, labels correspond to a set of more than two ordered categories or levels. This situation is referred to as ordinal regression [e.g. see 54]. If samples from one of the levels are plenty, while samples from the other levels are rare, then the problem can be considered a multi-level rare class problem.

The motivating application is based on a recent competition sponsored by the Heritage Health Provider Network [3].[1] The objective is to predict the number of days, $y_i \in \{0, 1, ..., 14, 15+\}$, member i will be hospitalized (inpatient or emergency room visit) in the following year using historical claims data. The number of days a member spends in the hospital is capped at 15 days to help protect the identiy of patients. The data provided consists of three years of historical member claims information. Claims data is anonymized to protect the identity of members [38]. The raw data contains basic member information,

[1] The competition ran for over two years ending in April 2013 and was highly publicized due to the potential impact on US healthcare and a US $3Mil prize. We participated in the competition placing 4th out of 1600+ teams. Our final submission used additional dataset variants and results from other methods as well, which were combined using a model stacking approach.

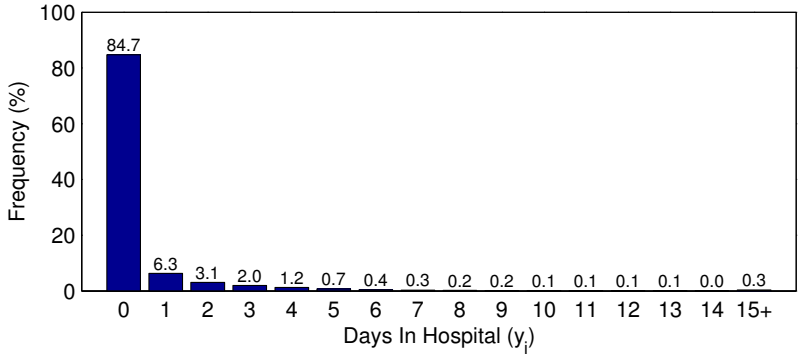


Figure 6.1: Distribution of the days in hospital for the Heritage Health Network problem.

claims data, drug counts, lab counts and outcome data in a set of relational tables. The training data consists of 147 473 patients over a two year period for which outcomes are given, with on average 12 claims per patient per year (1 764 561 total claims). The third year of data is used for testing, for which outcome information is not provided. We extracted 441 features from the relational data for each patient.

Figure 6.1 shows the outcome distribution for the two years of training data. We see that the distribution is highly skewed. In particular, examples corresponding to $y_i = 0$ constitute the majority of cases (85%), while other outcomes, $y_i = 1, ..., 15$, are significantly fewer. As in most rare class problems, we are more interested in identifying these rare outcomes.

6.2 Ordinal Regression with Multi-Level RankRC

To solve this problem, one may use traditional metric regression or multi-class classification approaches. However, neither of these approaches correctly capture the structure encoded in the labels. Traditional regression models assume the labels form an interval scale and errors of the same interval are penalized equally. But in the hospitalization prediction problem, errors are not all equal. For example, it is more important to distinguish between 0 and 1 days of hospitalization than between 14 and 15 days. Moreover, it is unclear what transformation would be most appropriate to represent the levels. Consequently, a regression approach may lead to a biased model with poor generalization ability [refer to 54, for further discussion]. On the other hand, these levels are also different from the labels of multiple classes in classification problems due to the existence of ordering information.

Therefore, this setting is best handled using an ordinal regression or ranking loss func-

tion, which attempts to rank the levels in the correct order, while not depending on the representation of the ranks [26, 54]. The biclass RankSVM problem (3.7) introduced in Section 3.3, can be generalized to multiple levels for this purpose, as follows:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^m} \quad \frac{1}{\sum_{r<s} m_r m_s} \sum_{r=1}^{R} \sum_{\{i:y_i=r\}} \sum_{\{j:y_i>y_j\}} \ell_h\left(f(\mathbf{x}_i) - f(\mathbf{x}_j)\right) + \frac{\lambda}{2} \sum_{i,j=1}^{m} \beta_i \beta_j k(\mathbf{x}_i, \mathbf{x}_j)\,, \tag{6.1}$$

where the hypothesis

$$f(\mathbf{x}) = \sum_{i=1}^{m} \beta_i k(\mathbf{x}_i, \mathbf{x})\,,$$

uses kernel instances at all data points following the Representer Theorem. Compared to (3.7), the loss function in (6.1), includes an additional summation over each rank level, $r > 0$, with a maximum rank of R. We assume the rank index starts at 0. For each r value, the objective in (6.1) reduces to a biclass ranking problem with r as the positive class label and all examples with label less than r as the negative class. Thus Problem (6.1) can be seen as combining R separate biclass ranking problems using the same hypothesis. The constant, m_r, denotes the number of observations which have output value r. For the hospitalization prediction problem, we set $R = 15$, and $r = 0, \ldots, 15$ represents the different ordinal levels (i.e. days in hospital).

We extend the notion of a rare-class representation to multiple levels as follows. In Figure 6.1 we observe that $r = 0$ correspond to the majority class, while all other outcomes represent rare cases. Therefore, we consider a representation which only uses kernel functions from examples corresponding to $r = 1, \ldots, 15$. If we decomposed the problem into R separate binary rare-class problems, this set would constitute the union of all the rare class points used in each of the problems. Thus we obtain the following multi-level RankRC problem:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^m} \quad \frac{1}{\sum_{r<s} m_r m_s} \sum_{r=1}^{R} \sum_{\{i:y_i=r\}} \sum_{\{j:y_i>y_j\}} \ell_h\left(\bar{f}(\mathbf{x}_i) - \bar{f}(\mathbf{x}_j)\right) + \frac{\lambda}{2} \sum_{\substack{\{i:y_i\neq 0\} \\ \{j:y_j\neq 0\}}} \beta_i \beta_j k(\mathbf{x}_i, \mathbf{x}_j)\,, \tag{6.2}$$

where the hypothesis

$$\bar{f}(\mathbf{x}) = \sum_{\{i:y_i \neq 0\}} \beta_i k(\mathbf{x}_i, \mathbf{x}) \ ,$$

is constrained to the set of rare class kernel functions. Problem (6.2) can be solved using similar method as described in Section 3.5 for the biclass RankRC, by noting the gradient and Hessian of the loss function is simply the sum of R biclass ranking loss functions. Thus, the complexity is $O\left(\sum_{r<s} m_r m_s\right)$ in both space and time.

To evaluate models we count the number of pairs that are correctly ranked among all possible pairs of data objects:

$$\text{MAUC} = \frac{1}{\sum_{r<s} m_r m_s} \sum_{r=1}^{R} \sum_{\{i:y_i=r\}} \sum_{\{j:y_i>y_j\}} \mathbb{I}\left(f(\mathbf{x}_i) > f(\mathbf{x}_j)\right) \ .$$

We call this measure MAUC to denote Multi-level AUC. An alternative measure is volume under the ROC surface, which generalizes ROC analysis to ordinal regression. However, computing volume under surface is prohibitive since it has exponential complexity in the number of ordinal levels. MAUC is an approximation of the volume under surface, which can be computed efficiently [101].

6.3 Comparison of Results

For our experiment, we compare the following methods: k-Neareast Neighbor (KNN) regression, Support Vector Regression (SVR) and three multi-level ranking methods, RANK-SVM (6.1), RANK-RC (6.2), and RANK-RND. RANK-RND is similar to RANK-RC, but with the hypothesis restricted to a randomly selected set of kernel functions, with the same cardinality used in RANK-RC.

We use LIBSVM [24] to solve the SVR problem. LIBSVM is a popular and efficient implementation of the sequential minimal optimization algorithm [75]. We set cache size to 10GB to minimize cache misses; termination criteria and shrinking heuristics are used in their default settings. The ranking methods (RANK-SVM, RANK-RC, RANK-RND) are solved using the subspace-trust-region method as described in Coleman and Li [30] and Branch et al. [21]. Termination tolerance is set at 1e-6. For ranking methods, the memory available to store the kernel matrix is limited to 10GB. Experiments are performed on a

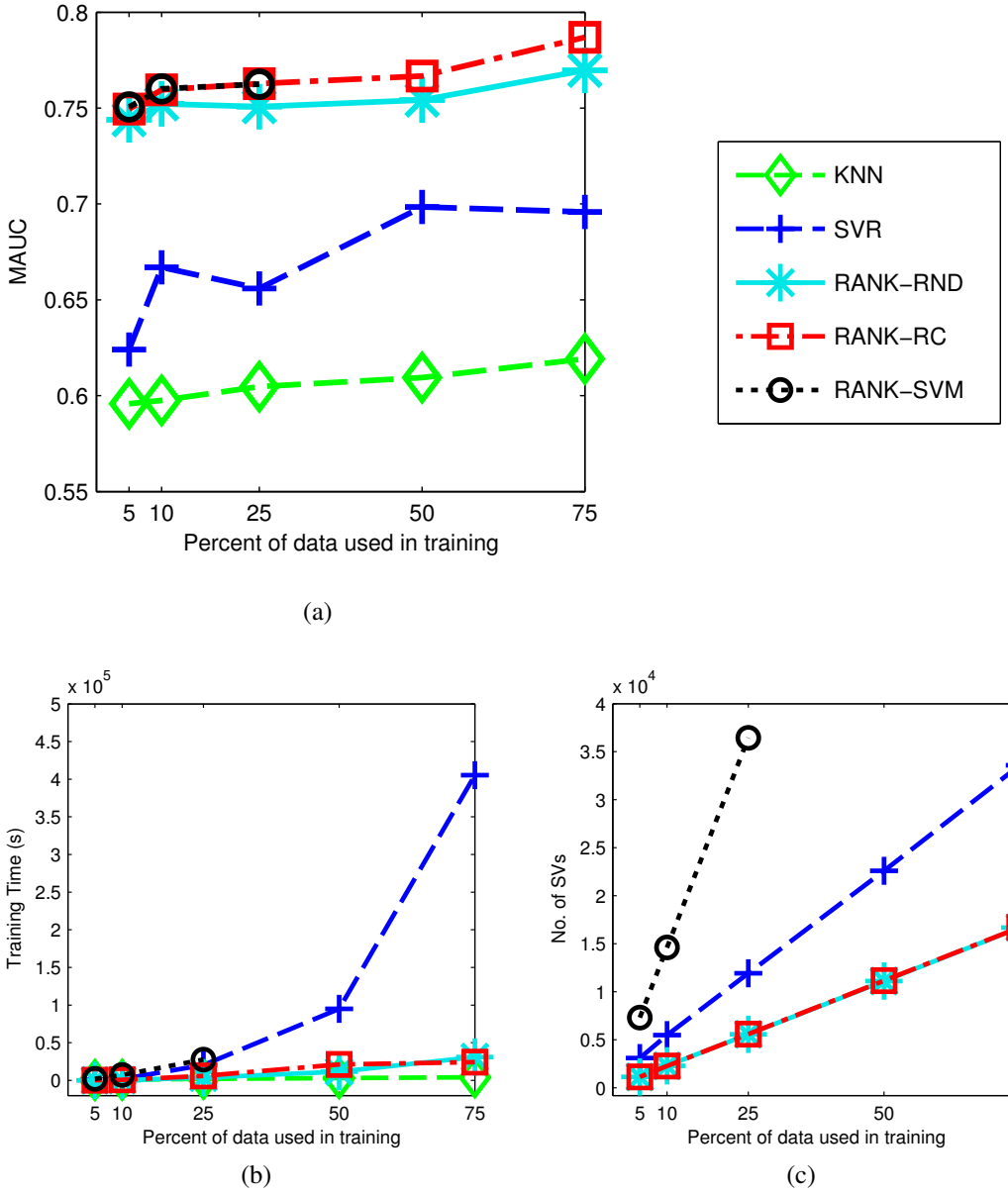


Figure 6.2: Comparison of (a) test MAUC (see text) score, (b) training time in seconds, and (c) number of support vectors, for the Heritage Health Network problem as percent of data used for training is increased from 5% to 75%. In our experiment setup, we were unable to train RANK-SVM with more than 25% of the data, due to the large size of the dataset.

Xeon E5620@2.4Ghz running Linux.

We train using 5%, 10%, 25%, 50%, and 75% of the training data. Half of the remaining data is used for validation, the other half for test. Features are standardized to zero mean and unit variance before training. Since our focus is on nonlinear kernels, for SVR and the ranking methods, we use the Gaussian kernel, $k(\mathbf{u}, \mathbf{v}) = \exp(-\|\mathbf{u} - \mathbf{v}\|_2^2/\sigma^2)$ with $\sigma^2 = \frac{1}{m^2}\sum_{i,j=1}^{m} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2$. The penalty parameter λ (or $\frac{1}{C}$ for SVR) is determined by cross-validation over values $\log_2 \lambda = [-20, -18, ..., 8, 10]$. For KNN we cross-validate over $k = [1, 2, ..., 100]$, where k is the number of nearest neighbors.

Figure 6.2a shows test MAUC results as training data is increased. Note, we are unable to train RANK-SVM with more than 25% of the data as the kernel matrix no longer fits in memory. The ranking methods outperform KNN and SVR. Among the ranking methods, RANK-RC performs slightly better than RANK-RND, and produces almost identical results to RANK-SVM in the cases where RANK-SVM can be computed. Figures 6.2b and 6.2c compare training time and number of support vectors, respectively, as training data is increased. We observe RANK-RC and RANK-RND scale well and use fewer support vectors than SVR and RANK-SVM.

6.4 Summary

In this chapter, we extended the biclass RankRC problem to a ranking problem with more than two levels. Our motivating example is based on a competition problem proposed by the Heritage Health Provide Network to predict number of days a member will be hospitalized in the following year. Since the training data contains almost 150 000 samples, the kernel RankSVM problem is too large to solve on standard machines. However, since the outcome distribution is highly skewed, we are able to take advantage of the rare class representation to efficiently solve the problem, with no apparent degradation in performance.

Chapter 7

Feature Selection

In this chapter, we develop an embedded feature selection method for kernel support vector machines (SVMs) based on a primal formulation. Since the resulting problem is non-convex, second-order methods, such as trust-region method can provide significant advantage in avoiding suboptimal solutions. We devise an alternating optimization approach to tackle the problem efficiently, breaking it down into a convex subproblem, corresponding to standard SVM optimization, and a non-convex subproblem for feature selection. Importantly, we show that a straightforward alternating optimization approach can be susceptible to saddle point solutions. We propose a novel technique, which shares an explicit margin variable to overcome saddle point convergence and improve solution quality. Experiment results show our method outperforms the state-of-the-art embedded SVM feature selection method, as well as other leading filter and wrapper approaches.

7.1 Introduction

Feature selection has become a significant research focus in statistical machine learning and data mining communities. As increasingly more data is available, problems with hundreds and thousands of features have become common. Some examples include text processing of internet documents, gene micro-array analysis, combinatorial chemistry, economic forecasting and context based collaborative filtering. However, irrelevant and redundant features reduce the effectiveness of data mining and may detract from the quality and accuracy of the resulting model. The goal of feature selection is to identify the most relevant subset of input features for the learning task, improving generalization error and model interpretability.

We focus on feature selection for nonlinear Support Vector Machine (SVM) classification. SVM is based on the principle of maximum-margin separation, which achieves the goal of Structural Risk Minimization by minimizing a generalization bound on model complexity and training error concurrently [33, 97]. The model is obtained by solving a convex quadratic programming problem. Linear SVM models can be extended to nonlinear ones by transforming the input features using a set of nonlinear basis functions. An important advantage of the SVM is that the transformation can be done implicitly using the "kernel trick", thereby allowing even infinite-dimensional feature expansions [15]. Empirically, SVMs have performed extremely well in diverse domains [e.g. see 22, 84].

Determining the optimal set of input features is in general NP-hard, requiring an exhaustive search of all possible subsets. Practical alternatives can be grouped into filter, wrapper, and embedded techniques [49]. In addition, there are a class of Bayesian approaches which tackle the problem by incorporating sparsity inducing priors [13, 14, 64, 71, 73].

Filter methods operate independently of the SVM classifier to score features according to how useful they are in predicting the output. Relief [60, 100] is a popular multivariate nonlinear filter that has successfully been used as a preprocessing step for SVMs [67]. Wrapper methods, on the other hand, use the SVM classifier to guide the search in the space of all possible subsets. For instance the most common wrapper, recursive feature elimination, greedily removes the worst (or adds the best) feature according to the loss (or gain) of the SVM classifier at each iteration [50]. Finally, embedded approaches incorporate the feature selection criterion in the SVM objective itself. Embedded methods can offer significant advantages over filters and wrappers, since they tightly couple feature selection with SVM learning, simultaneously searching over the feature and model space.

For linear SVMs, several embedded feature selection methods have been proposed. The general idea is to incorporate sparse regularization of the primal weight vector [20, 23, 46, 88, 104, 113]. However, similar techniques cannot be readily applied to nonlinear SVM classifiers, since the weight vector is not explicitly formed. Sparse regularization of the dual variables (support vectors) lead to a reduction in the number of kernel functions needed to generate the nonlinear surface, but does not result in a reduction of input features [46].

Embedding feature selection in a nonlinear SVM requires optimizing over additional parameters in the kernel function. This can be viewed as an instance of Generalized Multiple Kernel Learning (GMKL) [99], which offers the state-of-the-art solution for embedded

nonlinear feature selection. In general, the resulting problem is non-convex. The algorithm proposed by Varma and Babu [99] to solve GMKL is based on gradient descent, i.e. line-search along the negative gradient. Hence, it uses a first-order convex approximation at each iterate, which can fail to find a minimizer when the problem is non-convex. In contrast, trust-region algorithms are better suited for non-convex optimization. At each iterate they solve non-convex second-order approximations with guaranteed convergence to a minimizer.

We develop an effective algorithm to solve the non-convex optimization problem that results from embedding feature selection in nonlinear SVMs. Specifically:

1. We invoke the Representer Theorem to formulate a primal embedded feature selection SVM problem and use a smoothed hinge loss function to obtain a simpler bound constrained problem. We solve the resulting non-convex problem using a generalized trust-region algorithm for bound constrained minimization.

2. To improve efficiency we propose a two-block alternating optimization scheme, in which we iteratively solve (a) the standard SVM problem and (b) a smaller non-convex feature selection problem. Importantly, we propose a novel alternate optimization method by sharing a single perspective variable. We establish mathematical conditions under which this perspective variable sharing AO method avoids saddle points. For SVM feature selection, the perspective variable explicitly represents the margin. We provide computational evidence to illustrate that this helps avoid suboptimal local solutions. Moreover, by focussing on maximizing margin in the feature selection problem—a critical quantity for generalization error—we are able to further improve solution quality.

3. We compare our methods to GMKL and other leading nonlinear feature selectors, and show that our approach improves results.

The rest of the chapter is organized as follows. Section 7.2 formulates the embedded feature selection problem. Section 7.3 describes the bound constrained trust-region approach to solve the problem in the full feature and model space. Section 7.4 develops the explicit margin alternating optimization approach. Section 7.5 compares our approach with other nonlinear feature selection methods on several datasets.

7.2 Feature Selection in Nonlinear SVMs

We start by describing the embedded feature selection problem for nonlinear SVMs. We motivate and explain the formulation with respect to margin-based generalization bounds.

Consider a set of n training points, $\mathbf{x}_i \in \mathbb{R}^d$, and corresponding class labels, $y_i \in \{+1, -1\}$, $i = 1, ..., n$. Each component of $\mathbf{x}_i$ is an input feature. In classical SVM, proposed by [33], a linear classifier $(\mathbf{w}, b)$ is learned by maximizing the geometric margin, defined as $\gamma = \min_i \; y_i(\mathbf{w}^T\mathbf{x}_i + b)/\|\mathbf{w}\|$, where $\|\cdot\|$ denotes 2-norm. Since the decision hyperplane associated with $(\mathbf{w}, b)$ does not change upon rescaling to $(\lambda\mathbf{w}, \lambda b)$, for $\lambda \in \mathbb{R}^+$, the function output at the margin (functional margin) is fixed to 1; geometric margin is given by $\gamma = 1/\|\mathbf{w}\|$, and the norm of the weight vector is minimized. Thus in the standard setting, SVM results in the following convex quadratic programming problem:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i, \\ \text{s.t.} \quad & y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \xi_i, \quad i = 1, ..., n, \\ & \xi_i \geq 0, \quad i = 1, ..., n\,. \end{aligned} \tag{7.1}$$

Here, ξ_i's are margin violations, and C is a penalty controlling the trade-off between empirical error and (implicitly computed) geometric margin.

To obtain a non-linear decision function, the kernel trick [15] is used by defining a kernel, $k(\mathbf{x}, \mathbf{x}') \equiv \phi(\mathbf{x})^T\phi(\mathbf{x}')$, where $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ and $\phi : \mathbb{R}^d \to \mathcal{F}$ is a non-linear map from input features to a (potentially infinite dimensional) derived feature space. A kernel function, satisfying Mercer's condition [34, 68], directly computes the inner product of two vectors in a feature space $\mathcal{F}$, without the need to explicitly determine the feature mapping. Conventionally, the kernel is used in the dual of problem (7.1), where all occurrences of data appear inside an inner product. However, we can also formulate the primal problem in the derived feature space by expressing the weight vector as a linear combination of mapped data points, $\mathbf{w} = \sum_{i=1}^{n}\beta_i\phi(\mathbf{x}_i)$, due to Representer theorem [83]. Substituting this

form in (7.1) leads to the following primal non-linear SVM problem,

$$\begin{aligned} \min_{\boldsymbol{\beta},b,\boldsymbol{\xi}} \ & \frac{1}{2}\sum_{i,j=1}^{n}\beta_i\beta_j k(\mathbf{x}_i,\mathbf{x}_j)+C\sum_{i=1}^{n}\xi_i, \\ \text{s.t.} \ & y_i\left(\sum_{j=1}^{n}\beta_j k(\mathbf{x}_i,\mathbf{x}_j)+b\right)\geq 1-\xi_i,\ i=1,...,n, \\ & \xi_i\geq 0,\ i=1,...,n\,. \end{aligned} \tag{7.2}$$

The geometric margin in the feature space $\mathcal{F}$, is given by

$$\gamma=\frac{1}{\sqrt{\sum_{i,j=1}^{n}\beta_i\beta_j k(\mathbf{x}_i,\mathbf{x}_j)}}\,.$$

The maximum margin classifier is motivated by theoretical bounds on the generalization error. Specifically, [97] shows that generalization error for n points is bounded by,

$$err\leq\frac{c}{n}\left[\left(\frac{R^2}{\gamma^2}+\|\boldsymbol{\xi}\|^2\right)\log^2 n+\log\frac{1}{\delta}\right], \tag{7.3}$$

for some constant c with probability $1-\delta$, where γ is the geometric margin of the classifier. The key expression, on which generalization depends, is $R^2/\gamma^2+\|\boldsymbol{\xi}\|^2$, where $\boldsymbol{\xi}$ is the margin slack vector (normalized by γ), and R is the radius of the ball that encloses the set of points in the derived feature space, $\{\phi(\mathbf{x}_i)\}_{i=1}^{n}$. For a fixed dataset and kernel choice, R is constant, and thus maximizing the margin while reducing margin violations minimizes the upper bound in (7.3). Although the generalization bound suggests using a 2-norm penalty on margin violations, a 1-norm penalty is preferred for classification tasks, since it is a better approximation to a step penalty [97].

Now consider learning such a classifier while allowing input features to be weighted according to their relevance. We introduce a feature weight vector, $\mathbf{z}\in\mathbb{R}^d$, where $z_l\geq 0$ is a weight applied to input feature l. For convenience we define a diagonal matrix, $Z\in\mathbb{R}^{d\times d}$ with $Z_{ll}=z_l$. Hence, weighted points are mapped to $\phi(Z\mathbf{x})$ and we can replace $k(\mathbf{x},\mathbf{x}')$ by $k(Z\mathbf{x},Z\mathbf{x}')$ in problem (7.2) to obtain the following embedded feature selection problem, in which we simultaneously search for optimal feature weights, $\mathbf{z}$, while solving for model

parameters, $(\boldsymbol{\beta}, b)$:

$$\begin{aligned} \min_{\boldsymbol{\beta},b,\boldsymbol{\xi},\mathbf{z}} \; & \frac{1}{2}\sum_{i,j=1}^{n}\beta_i\beta_j k(Z\mathbf{x}_i, Z\mathbf{x}_j) + C\sum_{i=1}^{n}\xi_i + \mu\|\mathbf{z}\|_1, \\ \text{s.t.} \; & y_i\left(\sum_{j=1}^{n}\beta_j k(Z\mathbf{x}_i, Z\mathbf{x}_j) + b\right) \geq 1-\xi_i, \; i=1,...,n, \\ & \xi_i \geq 0, \; i=1,...,n, \\ & z_l \geq 0, \; l=1,...,d\,. \end{aligned} \tag{7.4}$$

We include 1-norm regularization of feature weights, $\mathbf{z}$, with a penalty parameter $\mu > 0$. This serves two purposes. Firstly, the 1-norm regularizer has the beneficial effect of suppressing variables to produce a sparse set of non-zero feature weights [95]. This property is desirable for feature selection where we are interested in identifying the most useful subset of input features. Secondly, it acts to minimize the radius of the enclosing ball, R, for the generalization bound in (7.3). Given two feature weight vectors, $\mathbf{z}$ and $\mathbf{z}'$, if $z'_l \leq z_l$, for $l = 1,...,d$, then $\sum_l z'^2_l (x_{il} - x_{jl})^2 \leq \sum_l z_l^2 (x_{ik} - x_{jl})^2$, implying $\|Z'\mathbf{x}_i - Z'\mathbf{x}_j\| \leq \|Z\mathbf{x}_i - Z\mathbf{x}_j\|$. Thus suppressing feature weights reduces distances between points in input space, which in turn results in a smaller enclosing ball in feature space. To minimize the generalization bound, we solve (7.4) and calibrate margin, errors, and radius via parameters C and μ, which can be determined by cross-validation.

7.2.1 Relation to GMKL

We note that problem (7.4) can be viewed as an instance of generalized multiple kernel learning [99]. For example, if we consider a radial basis kernel, then weighting features is equivalent to considering a product of 1-dimensional radial basis kernels derived from individual features with different width parameters. To solve this optimization problem Varma and Babu [99] propose a method based on gradient descent. Their algorithm follows the approach used Chapelle et al. [27] by reformulating the problem as a nested two step optimization: in an outer loop, the width parameters (i.e. feature weights) are updated by a line search step along the negative gradient assuming fixed SVM model parameters, while in an inner loop, the kernel is held fixed and SVM model parameters are updated. Assuming

a 1-norm feature weight regularizer, in the outer loop GMKL solves

$$\min_{\mathbf{z}} \; F(\mathbf{z}) + \mu ||\mathbf{z}||_1 \;\text{ subject to }\; z_l \geq 0,\; l = 1, ..., d \tag{7.5}$$

where

$$\begin{aligned} F(\mathbf{z}) = \max_{\alpha} \; & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(Z\mathbf{x}_i, Z\mathbf{x}_j) \\ \text{s.t.} \; & \sum_{i=1}^{n} y_i \alpha_i = 0, \\ & 0 \leq \alpha_i \leq C,\; i = 1, ..., n, \end{aligned} \tag{7.6}$$

is the solution of the dual SVM problem for fixed feature weights, which is solved in an inner loop. If α^* solves (7.6) exactly, the gradient, $\nabla_z F$, can be determined as a function of optimal α^* due to Danskin's Theorem [36]. At each iteration, a projected Armijo-step (i.e. line search) is taken in the direction of negative gradient to minimize (7.5). $F(\mathbf{z})$ is a non-convex function. The gradient descent algorithm uses a local first-order convex approximation and does not guarantee convergence to a minimizer in the non-convex case. Moreover, only an approximation to a gradient is available since the gradient requires an exact solution to the SVM problem, which computationally cannot be achieved. In contrast, we use a trust region based algorithm to solve the problem, which is better suited for non-convex optimization (7.4) and guarantees convergence to a minimizer.

7.3 Solving the Full-Space Feature Selection Problem

In this section, we solve the embedded feature selection SVM problem using trust region algorithm for a bound constrained problem. Problem (7.4) can be written as:

$$\min_{\boldsymbol{\beta}, b, \mathbf{z}} \; \Omega(\boldsymbol{\beta}, b, \mathbf{z}) \;\text{ s.t. }\; z_l \geq 0, \quad l = 1, ..., d\,, \tag{7.7}$$

where the objective is expressed in exact-penalty form:

$$\Omega(\cdot) = \frac{1}{2} \sum_{i,j=1}^{n} \beta_i \beta_j k(Z\mathbf{x}_i, Z\mathbf{x}_j) + C \sum_{i=1}^{n} \ell_h\big(y_i f(\mathbf{x}_i)\big) + \mu \|\mathbf{z}\|_1\,.$$

Here, $f(\mathbf{x}_i) = \sum_{j=1}^{n} \beta_j k(Z\mathbf{x}_i, Z\mathbf{x}_j) + b$ is the decision function, and $\ell_h(t) = \max(0, 1-t)$ is a non-differentiable linear hinge loss function. Alternative differentiable loss functions can be used instead. We use the following ϵ-smoothed hinge loss function $\ell_\epsilon(t)$:

$$\ell_\epsilon(t)(t) \equiv \begin{cases} (1-t)-\epsilon & \text{if } t < 1-2\epsilon \\ \frac{1}{4\epsilon}(1-t)^2 & \text{if } 1-2\epsilon \leq t < 1 \\ 0 & \text{if } t \geq 1\ . \end{cases} \tag{7.8}$$

The loss function transitions from linear cost to zero cost using a quadratic segment and bears similarity to a (truncated) Huber loss (see Figure 2.2b). Thus problem (7.7) becomes a smooth minimization problem with simple bound constraints. In our experiments we set $\epsilon = 0.5$. From a classification perspective, the smoothed hinge loss function is asymptotically margin-maximizing [79] and Bayes-risk consistent [72], and offers similar benefits as a linear hinge loss.

7.3.1 Trust Region Algorithm

Trust region algorithms are a class of relatively new optimization algorithms compared to classical line search methods. The main difference can be explained as follows. In the trust region method, we choose a step size (the size of the trust region) first and then search for a step direction, while in line search methods, we first choose a descent direction and then a step size. The trust region is usually a spherical or elliptical neighborhood centered at the current iterate, in which a local second order Taylor expansion (i.e. quadratic approximation) of the objective can be trusted. One of the main advantages of trust region methods is that a global solution to the local quadratic model can be computed, even when the Hessian is indefinite (non-convex). As a result trust region algorithms are better suited for non-convex optimization and can guarantee convergence to a minimizer.

For unconstrained minimization, the trust region method solves the following subproblem to obtain the step-size, $\mathbf{s}$, given the current iterate $\mathbf{x}^{(p)}$:

$$\begin{aligned} \min_{\mathbf{s}\in\mathbb{R}^m} \quad & \mathbf{s}^T\mathbf{g}^{(p)} + \frac{1}{2}\mathbf{s}^T H^{(p)}\mathbf{s}, \\ \text{s.t.} \quad & \|\mathbf{s}\|_2 \leq \Delta^{(p)}., \end{aligned} \tag{7.9}$$

Here $\mathbf{g}^{(p)}$ and $H^{(p)}$ are the gradient and Hessian of the objective function at $\mathbf{x}^{(p)}$, and $\Delta^{(p)}$ is the current radius of the trust region. For a nonconvex minimization problem, the Hessian

$H^{(p)}$ can be indefinite and (7.9) is a nonconvex quadratic minimization problem with a ball constraint. A *global* minimizer of this subproblem can be computed since there is no duality gap for a trust region subproblem. For example, assuming $\Delta^{(p)} = 1$, the dual of (7.9) can be solved by first computing a solution to a convex 1-dimensional problem:

$$\begin{aligned} \max_{\upsilon \in \mathbb{R}} \quad & -\sum_{i=1}^{m} \frac{(\mathbf{q}_i^T \mathbf{g}^{(p)})^2}{\upsilon_i + \upsilon} - \upsilon, \\ \text{s.t.} \quad & \upsilon \geq -\upsilon_{\min}(H^{(p)}), \end{aligned}$$

where υ_i and $\mathbf{q}_i$ are the eigenvalues and corresponding orthonormal eigenvectors of $H^{(p)}$, respectively, and $\upsilon_{\min}(H^{(p)})$ denotes the minimum eigenvalue of $H^{(p)}$ [18].

For our implementation, we use the trust region method described by Coleman and Li [31], which generalizes the unconstrained case to bound constraints. Each iteration requires an eigen-decomposition of the Hessian matrix involving cubic complexity. Consequently, solving (7.7) requires $O\big((n+d)^3\big)$ operations at each iteration. In the next section, we propose an explicit margin alternating optimization approach, which improves computation efficiency by breaking the problem down into two smaller subproblems, with $O(n^3)$ complexity for the SVM subproblem and $O(d^3)$ for the feature selection subproblem, while able to further improve solution quality by avoiding suboptimal local solutions.

7.4 Explicit Margin Sharing

We develop a novel alternating optimization (AO) method with explicit margin. We devise the formulation in three successive stages. Firstly, we present a simple, but naive approach, which alternates between solving for SVM model parameters and feature weights. Secondly, we extend the problem with an explicit margin variable which is shared between AO subproblems. Finally, we relax the margin term so it is not tied to geometric margin when solving the feature selection subproblem.

7.4.1 Simple AO

For fixed feature weights, (7.7) reduces to a convex problem that corresponds to regular SVM optimization. The standard SVM problem can be solved efficiently [e.g. see 40, 75, 112]. To avail of this, we consider a two-block AO approach (also known as nonlinear block

coordinate descent or the Gauss-Seidel method), which iterates between 1) fixing feature weights and solving SVM for model parameters $(\boldsymbol{\beta}, b)$, and 2) fixing model parameters and solving a smaller non-convex problem for feature weights, $\mathbf{z}$. The procedure is outlined in Algorithm 1.

Algorithm 1 Simple AO

$\mathbf{z}^0 \leftarrow$ initial feature weights
$k \leftarrow 0$
repeat
  $(\boldsymbol{\beta}^k, b^k) \leftarrow \text{argmin}_{\boldsymbol{\beta},b}\ \Omega(\boldsymbol{\beta}, b, \mathbf{z}^k)$ (SVM)
  $\mathbf{z}^{k+1} \leftarrow \text{argmin}_{\mathbf{z}\geq\mathbf{0}}\ \Omega(\boldsymbol{\beta}^k, b^k, \mathbf{z})$ (FS)
  $k \leftarrow k+1$
until $\|\mathbf{z}^{k+1} - \mathbf{z}^k\|_\infty < tol$

We can use any convex solver for the SVM subproblem and use the bound-constrained trust-region algorithm described in Section 7.3.1 to solve the non-convex feature selection subproblem. The procedure generates a sequence $\{(\boldsymbol{\beta}^k, b^k, \mathbf{z}^k)\}_{k=1}^{\infty}$, which can be shown to converge to a stationary point of (7.7) [48]. We stop when successive changes in feature weights are less than a prespecified tolerance, tol.

Although this simple alternating optimization scheme improves computational efficiency by breaking the problem down into two smaller subproblems, it detracts from an important advantage of using the trust-region algorithm—convergence to a minimizer. Even though each subproblem converges to a minimizer when viewed along their restricted subspaces, the solution may not converge to a minimizer in the full variable space [12]. A simple example can be illustrative. Consider minimizing the three-variable quadratic function,

$$f(x_1, x_2, x_3) = (x_1 + x_2 - 2)^2 - 3(x_1 + x_2 - 2)(x_3 - 1) + (x_3 - 1)^2,$$

using AO on variable subsets $\{x_1, x_2\}$ and $\{x_3\}$. For fixed $x_3 = 1$, the point $(x_1, x_2) = (1, 1)$ is a global minimizer of $f(x_1, x_2, 1) = (x_1 + x_2 - 2)^2$, and for the fixed point $(x_1, x_2) = (1, 1)$, $x_3 = 1$ is the global minimizer of $f(1, 1, x_3) = (x_3 - 1)^2$. Consequently, AO can converge to $(x_1, x_2, x_3) = (1, 1, 1)$, which is a stationary point, but not a minimizer of the full variable space (i.e. it is a saddle point).

We illustrate that saddle point convergence in Algorithm 1 can be problematic. Table 7.1 shows test error on simulated NDCC dataset as the number of irrelevant features

(probes) are increased. Refer to Section 7.5.1 for description of the data. We setup the problem as described in Section 7.5.1. The Simple AO method is unable to identify the correct set of features and test error quickly deteriorates as the number of probes increase. Moreover, the projected Hessian is indefinite at most solutions, indicating the algorithm is terminating at saddle points.

	Simple AO		Margin AO-I		Margin AO-II	
Probes	$\upsilon_{\min}$	Test error	$\upsilon_{\min}$	Test error	$\upsilon_{\min}$	Test error
0	0.0242	**8.3 (M)**	0.0889	**9.4 (M)**	0.0288	**8.9 (M)**
2	-0.9567	8.9 (S)	0.0201	**10.7 (M)**	0.0307	**9.8 (M)**
5	-0.4114	19.1 (S)	0.0232	**17.5 (M)**	0.0463	**8.4 (M)**
10	-20.1816	44.9 (S)	0.0209	**11.2 (M)**	0.0254	**9.9 (M)**
20	-4.5354	52.0 (S)	0.0221	**10.4 (M)**	0.0256	**9.9 (M)**
50	-3.0313	50.0 (S)	0.0000	48.0 (M)	0.0199	**9.8 (M)**
100	-2.3565	47.5 (S)	0.0000	45.9 (M)	0.0194	**10.5 (M)**

Table 7.1: Comparison of solutions using Simple AO, Margin AO-I and Margin AO-II as additional probe features are included in the NDCC dataset. Minimum eigenvalue of projected Hessian ($\upsilon_{\min}$) in $(\mathbf{u}, b, \mathbf{z})$ space and test error in percent is shown for each solution. Label (M) indicates that, at termination, the projected Hessian is positive semi-definite, while label (S) indicates that the projected Hessian is indefinite. Bolded values indicate the solution identified the correct set of features.

7.4.2 Shared Margin AO-I

To overcome the issues with naive AO, we propose to share a dummy variable between AO iterates. We illustrate the proposed technique by returning to the simple three-variable example. By introducing a perspective transformation, $x_1 = \bar{x}_1/y$ and $x_2 = \bar{x}_2/y$, we obtain,

$$\bar{f}(\bar{x}_1, \bar{x}_2, x_3, y) = (\bar{x}_1/y + \bar{x}_2/y - 2)^2 - 3(\bar{x}_1/y + \bar{x}_2/y - 2)(x_3 - 1) + (x_3 - 1)^2 .$$

Instead of alternating between two disjoint sets of variables, we share the y variable between AO iterates (also known as overlapping domain decomposition). Thus we minimize $\bar{f}(\bar{x}_1, \bar{x}_2, x_3, y)$ over variable subsets $\{\bar{x}_1, \bar{x}_2, y\}$ and $\{x_3, y\}$. For fixed $x_3 = 1$, we minimize $\bar{f}(\bar{x}_1, \bar{x}_2, 1, y) = (\bar{x}_1/y + \bar{x}_2/y - 2)^2$, to obtain a global minimizer $(\bar{x}_1, \bar{x}_2, y) = (1, 1, 1)$ which corresponds to $(x_1, y_1) = (1, 1)$ as before. However, for fixed $(\bar{x}_1, \bar{x}_2) = (1, 1)$, we now minimize $\bar{f}(1, 1, x_3, y) = 4(1/y - 1)^2 - 6(1/y - 1)(x_3 - 1) + (x_3 - 1)^2$ to find that the Hessian with

respect to (x_3, y) is indefinite at $(x_3, y) = (1, 1)$. Thus by extending the subspace with an auxiliary perspective variable, which is shared between AO subproblems, convergence to saddle points is avoided in this case.

Although sharing a perspective variable in AO optimization does not guarantee a local minimum in general, it can reduce the possibility of convergence to saddle points with negligible additional computational cost. We demonstrate this both analytically and computationally.

To illustrate it analytically, we establish Theorem 11 below, which provides conditions under which local minima in the overlapping subspaces lead to a local minimum in the full space. In addition, assuming that one is at a saddle point, we provide a sufficient condition under which this is detected and a negative curvature direction in the overlapping subspace can be found to move away from the saddle point. We note that all gradient and Hessian expressions of f and $\bar{f}$ in Theorem 11 are assumed to be evaluated at $\left(\frac{\bar{\mathbf{x}}_1^*}{\kappa^*}, \mathbf{x}_2^*\right)$ and $(\bar{\mathbf{x}}_1^*, \mathbf{x}_2^*, \kappa^*)$, respectively.

Theorem 11. *Let $f(\mathbf{x}_1, \mathbf{x}_2) : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}$ be a twice continuously differentiable function. Let $\kappa \neq 0$, $\kappa \in \mathbb{R}$, and define $\bar{f}(\bar{\mathbf{x}}_1, \mathbf{x}_2, \kappa) = f\left(\frac{\bar{\mathbf{x}}_1}{\kappa}, \mathbf{x}_2\right)$. Assume that $\bar{f}(\bar{\mathbf{x}}_1, \mathbf{x}_2, \kappa)$ attains local minima at $(\bar{\mathbf{x}}_1^*, \kappa^*)$ and $(\mathbf{x}_2^*, \kappa^*)$, where $\kappa^* \neq 0$, in the subspace $(\bar{\mathbf{x}}_1, \kappa)$ and $(\mathbf{x}_2, \kappa)$ respectively. Then the first order necessary conditions for $(\mathbf{x}_1^*, \mathbf{x}_2^*)$ to be at a local minimum for $f(\mathbf{x}_1, \mathbf{x}_2)$ are satisfied. In addition, if the second order sufficient condition is satisfied at $(\mathbf{x}_2^*, \kappa^*)$ in the subspace $(\mathbf{x}_2, \kappa)$, then $\nabla^2_{\mathbf{x}_2\mathbf{x}_2} f \succ 0$. Moreover*

(a) At $\left(\frac{\bar{\mathbf{x}}_1^*}{\kappa^*}, \mathbf{x}_2^*\right)$, the second order necessary (sufficient) condition for minimizing $f(\mathbf{x}_1, \mathbf{x}_2)$ is satisfied if and only if

$$H_{schur} \triangleq \left(\nabla^2_{\mathbf{x}_1\mathbf{x}_1} f\right) - \left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2} f\right)\left(\nabla^2_{\mathbf{x}_2\mathbf{x}_2} f\right)^{-1}\left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2} f\right)^T \succeq (\succ) 0\,. \tag{7.10}$$

(b) Assume that the minimum eigenvalue of

$$H = \begin{bmatrix} \nabla^2_{\mathbf{x}_1\mathbf{x}_1} f & \nabla^2_{\mathbf{x}_1\mathbf{x}_2} f \\ \left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2} f\right)^T & \nabla^2_{\mathbf{x}_2\mathbf{x}_2} f \end{bmatrix}$$

is negative. Then the minimum eigenvalue of H_{schur} defined in (7.10) is negative. Define the region of negative curvature of H_{schur} as

$$\mathcal{N} = \left\{v : \; v^T H_{schur} v < 0\right\}$$

If $\mathbf{x}_1^* \in \mathcal{N}$, then the minimum eigenvalue of the Hessian of $\bar{f}(\bar{\mathbf{x}}_1, \mathbf{x}_2, \kappa)$ at $(\mathbf{x}_2^*, \kappa^*)$ in the subspace $(\mathbf{x}_2, \kappa)$ is negative.

Proof. Assume that $\bar{f}(\bar{\mathbf{x}}_1, \mathbf{x}_2, \kappa)$ attains local minima at $(\bar{\mathbf{x}}_1^*, \kappa^*)$ and $(\mathbf{x}_2^*, \kappa^*)$, where $\kappa^* \neq 0$, in the subspace $(\bar{\mathbf{x}}_1, \kappa)$ and $(\mathbf{x}_2, \kappa)$ respectively. Then, at $(\bar{\mathbf{x}}_1^*, \kappa^*)$,

$$\nabla_{\bar{\mathbf{x}}_1} \bar{f} = 0, \quad \nabla_{\kappa} \bar{f} = 0$$

and, at $(\mathbf{x}_2^*, \kappa^*)$,

$$\nabla_{\mathbf{x}_2} \bar{f} = 0.$$

From

$$\nabla_{\bar{\mathbf{x}}_1} \bar{f} = \frac{1}{\kappa} \nabla_{\mathbf{x}_1} f \ \text{ and } \ \nabla_{\mathbf{x}_2} \bar{f} = \nabla_{\mathbf{x}_2} f$$

the first order necessary condition for minimizing $f(\mathbf{x}_1, \mathbf{x}_2)$ is satisfied at $(\frac{1}{\kappa^*}\bar{\mathbf{x}}_1^*, \mathbf{x}_2^*)$.

We note that the Hessian of $\bar{f}(\bar{\mathbf{x}}_1, \mathbf{x}_2, \kappa)$ at $(\mathbf{x}_2^*, \kappa^*)$ in the subspace $(\mathbf{x}_2, \kappa)$ is

$$\begin{aligned}
Q(\mathbf{x}_2^*, \kappa^*) &\triangleq \begin{bmatrix} \nabla^2_{\mathbf{x}_2\mathbf{x}_2} \bar{f} & \nabla^2_{\mathbf{x}_2\kappa} \bar{f} \\ \left(\nabla^2_{\mathbf{x}_2\kappa} \bar{f}\right)^T & \nabla^2_{\kappa\kappa} \bar{f} \end{bmatrix} \\
&= \begin{bmatrix} \nabla^2_{\mathbf{x}_2\mathbf{x}_2} f & -\frac{1}{\kappa^{*2}} \left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2} f\right)^T \bar{\mathbf{x}}_1^* \\ -\frac{1}{\kappa^{*2}} \bar{\mathbf{x}}_1^{*T} \left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2} f\right) & \frac{1}{\kappa^{*4}} \bar{\mathbf{x}}_1^{*T} \left(\nabla^2_{\mathbf{x}_1\mathbf{x}_1} f\right) \bar{\mathbf{x}}_1^* + \frac{2}{\kappa^{*3}} \left(\nabla_{\mathbf{x}_1} f\right)^T \bar{\mathbf{x}}_1^* \end{bmatrix} \\
&= \begin{bmatrix} \nabla^2_{\mathbf{x}_2\mathbf{x}_2} f & -\frac{1}{\kappa^*} \left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2} f\right)^T \mathbf{x}_1^* \\ -\frac{1}{\kappa^*} \mathbf{x}_1^{*T} \left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2} f\right) & \frac{1}{\kappa^{*2}} \mathbf{x}_1^{*T} \left(\nabla^2_{\mathbf{x}_1\mathbf{x}_1} f\right) \mathbf{x}_1^* \end{bmatrix}
\end{aligned}$$

where we use $\nabla_{\mathbf{x}_1} f = 0$ and $\mathbf{x}_1^* = \frac{\bar{\mathbf{x}}_1^*}{\kappa^*}$ in the last step. If the second order sufficient condition is satisfied at $(\mathbf{x}_2^*, \kappa^*)$ in the subspace $(\mathbf{x}_2, \kappa)$, then we immediately conclude that $\nabla^2_{\mathbf{x}_2\mathbf{x}_2} f \succ 0$.

(a) Since $\nabla^2_{\mathbf{x}_2\mathbf{x}_2} f \succ 0$, results follow from recognizing H_{schur} as the Schur Complement of H.

(b) From (a) we conclude that the minimum eigenvalue of H_{schur} is negative.

If $\mathbf{x}_1^* \in \mathcal{N}$, then

$$\mathbf{x}_1^{*T} \left(\left(\nabla^2_{\mathbf{x}_1\mathbf{x}_1} f\right) - \left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2} f\right) \left(\nabla^2_{\mathbf{x}_2\mathbf{x}_2} f\right)^{-1} \left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2} f\right)^T \right) \mathbf{x}_1^* < 0.$$

Consequently

$$\frac{1}{\kappa^{*2}}\mathbf{x}_1^{*T}\left(\nabla^2_{\mathbf{x}_1\mathbf{x}_1}f\right)\mathbf{x}_1^* - \left(-\frac{1}{\kappa^*}\mathbf{x}_1^{*T}\left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2}f\right)\right)\left(\nabla^2_{\mathbf{x}_2\mathbf{x}_2}f\right)^{-1}\left(-\frac{1}{\kappa^*}\left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2}f\right)^T\mathbf{x}_1^*\right) < 0 \tag{7.11}$$

Recognizing the LHS of (7.11) as the Schur complement of $Q(\mathbf{x}_2^*, \kappa^*)$, we conclude that the minimum eigenvalue of $Q(\mathbf{x}_2^*, \kappa^*)$ is negative. This completes the proof.

□

It is also possible to establish a sufficiency condition, based on the minimum and maximum eigenvalues of $\nabla^2_{\mathbf{x}_1\mathbf{x}_1}f$, $\nabla^2_{\mathbf{x}_2\mathbf{x}_2}f$, and $(\nabla^2_{\mathbf{x}_1\mathbf{x}_2}f)(\nabla^2_{\mathbf{x}_1\mathbf{x}_2}f)^T$ respectively, under which local minima in the overlapping subspaces leads to a local minimum in the full space. This is presented in Theorem 12 below.

Theorem 12. *Let $\upsilon_{\min}(A)$ and $\upsilon_{\max}(A)$ denote the minimum and maximum eigenvalue of matrix A, respectively. Assume that $\nabla^2_{\mathbf{x}_1\mathbf{x}_1}f \succ 0$. In addition, assume that*

$$\upsilon_{\min}\left(\nabla^2_{\mathbf{x}_2\mathbf{x}_2}f\right) > \frac{\upsilon_{\max}\left(\left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2}f\right)\left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2}f\right)^T\right)}{\upsilon_{\min}\left(\nabla^2_{\mathbf{x}_1\mathbf{x}_1}f\right)}, \tag{7.12}$$

then

$$\left(\nabla^2_{\mathbf{x}_1\mathbf{x}_1}f\right) - \left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2}f\right)\left(\nabla^2_{\mathbf{x}_2\mathbf{x}_2}f\right)^{-1}\left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2}f\right)^T \succ 0.$$

Proof. Assume that I_{n_2} is the n_2-by-n_2 identity matrix. We have

$$\nabla^2_{\mathbf{x}_2\mathbf{x}_2}f \succeq \upsilon_{\min}\left(\nabla^2_{\mathbf{x}_2\mathbf{x}_2}f\right)I_{n_2}.$$

Since $\nabla^2_{\mathbf{x}_1\mathbf{x}_1}f \succ 0$, using assumption (7.12)

$$\nabla^2_{\mathbf{x}_2\mathbf{x}_2}f \succ \frac{\upsilon_{\max}\left(\left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2}f\right)\left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2}f\right)^T\right)}{\upsilon_{\min}\left(\nabla^2_{\mathbf{x}_1\mathbf{x}_1}f\right)}I_{n_2}.$$

For any $\mathbf{d} \in \mathbb{R}^{n_1}$, $\mathbf{d} \neq 0$,

$$\frac{\upsilon_{\max}\left(\left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2}f\right)\left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2}f\right)^T\right)}{\upsilon_{\min}\left(\nabla^2_{\mathbf{x}_1\mathbf{x}_1}f\right)}I_{n_2} \succeq \frac{\mathbf{d}^T\left(\left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2}f\right)\left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2}f\right)^T\right)\mathbf{d}}{\mathbf{d}^T\left(\nabla^2_{\mathbf{x}_1\mathbf{x}_1}f\right)\mathbf{d}}I_{n_2} \succeq \frac{\left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2}f\right)^T\mathbf{d}\mathbf{d}^T\left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2}f\right)}{\mathbf{d}^T\left(\nabla^2_{\mathbf{x}_1\mathbf{x}_1}f\right)\mathbf{d}}.$$

From the assumption $\nabla^2_{\mathbf{x}_1\mathbf{x}_1} f \succ 0$, by Schur's complement, we conclude

$$\begin{bmatrix} \nabla^2_{\mathbf{x}_2\mathbf{x}_2} f & \left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2} f\right)^T \mathbf{d} \\ \mathbf{d}^T \left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2} f\right) & \mathbf{d}^T \left(\nabla^2_{\mathbf{x}_1\mathbf{x}_1} f\right) \mathbf{d} \end{bmatrix} \succ 0\,.$$

Applying Schur's complement again, using $\nabla^2_{\mathbf{x}_1\mathbf{x}_1} f \succ 0$, we have

$$\mathbf{d}^T \left(\left(\nabla^2_{\mathbf{x}_1\mathbf{x}_1} f\right) - \left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2} f\right) \left(\nabla^2_{\mathbf{x}_2\mathbf{x}_2} f\right)^{-1} \left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2} f\right)^T \right) \mathbf{d} > 0.$$

Thus

$$\left(\left(\nabla^2_{\mathbf{x}_1\mathbf{x}_1} f\right) - \left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2} f\right) \left(\nabla^2_{\mathbf{x}_2\mathbf{x}_2} f\right)^{-1} \left(\nabla^2_{\mathbf{x}_1\mathbf{x}_2} f\right)^T \right) \succ 0\,.$$

□

Theorem 11 and 12 suggest that sharing a scalar perspective variable between AO iterates can reduce the possibility of saddle point convergence compared to the naive AO technique.

Therefore, we consider a perspective transformation of the SVM model parameters in the AO approach for the feature selection problem. We substitute $\boldsymbol{\beta} = \bar{\boldsymbol{\beta}}/\lambda$ and $b = \bar{b}/\lambda$ in (7.4) to obtain,

$$\begin{aligned} \min_{\bar{\boldsymbol{\beta}},\bar{b},\boldsymbol{\xi},\mathbf{z},\lambda} \;& \frac{1}{2\lambda^2} \sum_{i,j=1}^{n} \bar{\beta}_i \bar{\beta}_j k(Z\mathbf{x}_i, Z\mathbf{x}_j) + C \sum_{i=1}^{n} \xi_i + \mu \|\mathbf{z}\|_1, \\ \text{s.t. } & y_i \left(\sum_{j=1}^{n} \bar{\beta}_j k(Z\mathbf{x}_i, Z\mathbf{x}_j) + \bar{b} \right) \geq \lambda - \lambda\xi_i,\ i = 1,...,n, \\ & \xi_i \geq 0, \quad i = 1,...,n, \\ & z_l \geq 0, \quad l = 1,...,d, \\ & \lambda \geq 0\,. \end{aligned} \tag{7.13}$$

It is important to note that here the auxiliary perspective variable, λ, is the functional margin (see Section 7.2), which is a component of the generalization bound. By sharing λ and performing AO with respect to (7.13), maximizing the functional margin becomes part of the objective in each iteration of the subspace (z, λ) optimization. Consequently margin sharing AO ensures that the feature z is selected according to generalization bound,

in addition to reducing the possibility of becoming trapped at a saddle point in the AO optimization.

We perform functional margin sharing AO as follows. For fixed fixture weights, (7.13) is equivalent to a regular SVM, as before (since we fix the functional margin, λ, to 1 to make the problem well-posed). However, in the feature selection subproblem, when model parameters $(\bar{\boldsymbol{\beta}}, \bar{b})$ are fixed, λ provides an additional view of the margin component. This allows us to move along a direction in the SVM model space while solving the feature selection subproblem. The procedure is shown in Algorithm 2 using the following exact-penalty expression for the objective:

$$\bar{\Omega}(\bar{\boldsymbol{\beta}}, \bar{b}, \mathbf{z}, \lambda) = \frac{1}{2\lambda^2} \sum_{i,j=1}^{n} \bar{\beta}_i \bar{\beta}_j k(Z\mathbf{x}_i, Z\mathbf{x}_j) + C \sum_{i=1}^{n} \ell\left(\frac{y_i f(\mathbf{x}_i)}{\lambda}\right) + \mu \|\mathbf{z}\|_1. \tag{7.14}$$

We solve the feature selection subproblem (FS-I) using the bound constrained trust region algorithm described in Section 7.3.1. We use $\mathbf{z} = \mathbf{z}^k$ and $\lambda = 1$ as initial points in step 6.

Algorithm 2 Margin AO-I

1: $\mathbf{z}^0 \leftarrow$ initial feature weights
2: $k \leftarrow 0$
3: **repeat**
4: $\quad \lambda^k \leftarrow 1$
5: $\quad (\bar{\boldsymbol{\beta}}^k, \bar{b}^k) \leftarrow \operatorname{argmin}_{\bar{\boldsymbol{\beta}}, \bar{b}} \bar{\Omega}(\bar{\boldsymbol{\beta}}, \bar{b}, \mathbf{z}^k, \lambda^k)$ (SVM)
6: $\quad (\mathbf{z}^{k+1}, \lambda^{k+1}) \leftarrow \operatorname{argmin}_{\mathbf{z} \geq \mathbf{0}, \lambda \geq 0} \bar{\Omega}(\bar{\boldsymbol{\beta}}^k, \bar{b}^k, \mathbf{z}, \lambda)$ (FS-I)
7: $\quad k \leftarrow k+1$
8: **until** $\|\mathbf{z}^{k+1} - \mathbf{z}^k\|_\infty < tol$

Table 7.1 shows test error and minimum eigenvalue of full Hessian using Margin AO-I (Algorithm 2) on the NDCC dataset. We observe that Margin AO-I is able to avoid saddle point solutions and obtain better test error than Simple AO. In our experiments, we generally observed that margin sharing AO yields similar test performance to the full-space solution discussed in Section 7.3—with the added benefit of lower complexity.

7.4.3 Explicit (Functional) Margin AO-II

We can further improve the solution by observing that for fixed support vectors, it is more relevant to maximize functional margin than geometric margin. Specifically, we propose to

minimize the following objective (over $\mathbf{z}, \lambda$) in place of the subproblem (FS-I).

$$\Psi(\cdot) = \frac{1}{2\lambda^2} + C\sum_{i=1}^{n} \ell\left(\frac{y_i f(\mathbf{x}_i)}{\lambda}\right) + \mu\|\mathbf{z}\|_1. \tag{7.15}$$

The first term in (7.15) represents the (inverse) functional margin. In comparison, the first term of (7.14) represents the (inverse) geometric margin. Recall, in SVM the norm is minimized as the functional margin is held constant at 1 to fix the scale of support vector coefficients. That is, in a standard SVM the following equivalent problem is solved

$$\begin{aligned} \min_{f\in\mathcal{H},\boldsymbol{\xi},\lambda} \quad & \frac{1}{2\lambda^2} + C\sum_{i=1}^{n}\xi_i, \\ \text{s.t.} \quad & y_i f(\mathbf{x}_i) \geq \lambda - \lambda\xi_i, \quad \xi_i \geq 0, \quad i = 1,...,n, \\ & \|f\|_{\mathcal{H}}^2 = 1\,, \end{aligned} \tag{7.16}$$

where $\|\cdot\|_{\mathcal{H}}$ is the norm in the reproducing kernel Hilbert space $\mathcal{H}$. The final constraint can be seen as fixing the scale of coefficients to make the solution unique, while the functional margin λ is maximized. Since, in the feature selection subproblem support vector coefficients are assumed fixed, this constraint becomes unnecessary. Minimizing (7.15) is equivalent to problem (7.16) without the constraint $\|f\|_{\mathcal{H}}^2 = 1$. Thus we directly optimize feature weights to maximize (functional) margin in the feature subproblem. Since we are solving the same problem over a larger feasible set, the optimal objective value of (7.15) is always less than or equal to (7.16). Consequently, (7.15) allows greater flexibility as we search for optimal features and can further avoid suboptimal solutions.

The procedure is summarized in Algorithm 3. We use the bound constrained trust-region algorithm to solve the feature selection subproblem (FS-II). To make the SVM subproblem solution unique we fix λ in step 5, as in Margin AO-I. Similarly, in step 4, we always reset $\lambda = 1$, since the specific value of λ only controls the scale of $(\bar{\boldsymbol{\beta}}, \bar{b})$. Note, AO iterates no longer share a common objective. Therefore, we set $\mathbf{z} = \mathbf{z}^k$ and $\lambda = 1/\sqrt{\sum_{i,j=1}^{n} \bar{\beta}_i^k \bar{\beta}_j^k k(Z^k\mathbf{x}_i, Z^k\mathbf{x}_j)}$ as initial points in step 6 to match the initial objective value obtained from step 5. Since we are mainly interested in feature selection, we use a weaker stopping criteria based on the zero norm[1] of the weight vector, which in practice terminates after a few AO iterates.

[1] In our computation a component is considered zero if its absolute value is less than $0.01 \times \max_k |z_k|$.

Algorithm 3 Explicit (functional) Margin AO-II

$\mathbf{z}^0 \leftarrow$ initial feature weights
$k \leftarrow 0$
repeat
  $\lambda^k \leftarrow 1$
  $(\bar{\boldsymbol{\beta}}^k, \bar{b}^k) \leftarrow \operatorname{argmin}_{\bar{\boldsymbol{\beta}}, \bar{b}} \; \bar{\Omega}(\bar{\boldsymbol{\beta}}, \bar{b}, \mathbf{z}^k, \lambda^k)$ (SVM)
  $(\mathbf{z}^{k+1}, \lambda^{k+1}) \leftarrow \operatorname{argmin}_{\mathbf{z} \geq \mathbf{0}, \lambda \geq 0} \; \Psi(\bar{\boldsymbol{\beta}}^k, \bar{b}^k, \mathbf{z}, \lambda)$ (FS-II)
  $k \leftarrow k+1$
until $\|\mathbf{z}^{k+1} - \mathbf{z}^k\|_0 = 0$

Table 7.1 shows results on NDCC dataset using Explicit Margin AO-II (Algorithm 3) method. Similar to Margin AO-I, the algorithm is able to avoid saddle points. In addition, Explicit Margin AO-II can effectively identify the correct set of features and recover a competitive test error even as the number of probes become substantial. Compared to the full-space approach (Section 7.3), the explicit margin AO-II approach is more efficient, and in addition, by focussing on improving the margin directly—a critical quantity for generalization—it further improves solution quality.

7.5 Experiments

In this section we evaluate our Full-Space (FULL-FS, Section 7.3) and Margin AO-II (Section 7.4.3) methods on various datasets. We compare results with the state-of-the-art embedded feature selection algorithm, GMKL [99]. On a simulated dataset [94] we show that GMKL can fail to find the correct subset of features, while FULL-FS recovers a better solution and AO-II recovers the correct solution. On several other real datasets we show that our methods perform better than GMKL by 8-14% on average in terms of test error and with a reduction of 16-28% of features. We also demonstrate that FULL-FS and AO-II improve upon other leading filter and wrapper approaches in ranking relevant features.

7.5.1 Comparison to GMKL

For FULL-FS, AO-II and GMKL, we use the following radial-basis kernel,

$$k(Z\mathbf{x}_i, Z\mathbf{x}_j) = \exp\left(-\sum_{k=1}^{d}(z_k x_{ik} - z_k x_{jk})^2\right). \tag{7.17}$$

and a 1-norm penalty on feature weights, $\mu||\mathbf{z}||_1$. The implementation for GMKL is publicly available from Varma [98]. All datasets are standardized to zero mean and unit variance and we always start with an initial feature weight vector of ones. The two parameters, C and μ, are determined by cross-validation over $(\log_2 C, \log_2 \mu)$ space at grid points $[-5,-4,...,14,15] \times [-10,-8,...,8,10]$. We also compare results with regular SVM using the entire set of features. For SVM we use a radial basis kernel with width σ,

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\sum_{k=1}^{d}(x_{ik}-x_{jk})^2}{\sigma^2}\right),$$

and cross-validate over $(\log_2 C, \log_2 \sigma)$ at $[-5,-4,...,14,15] \times [-10,-8,...,8,10]$.

Normally Distributed Clusters on Cubes

In this example we evaluate feature selection using a simulated dataset. Normally distributed clusters on cubes (NDCC) generates nonlinearly separable data by sampling from multivariate normal distributions with centers at the vertices of three concentric 1-norm cubes [94]. An example with 2-dimensional cubes is shown in Figure 7.1. The distribution

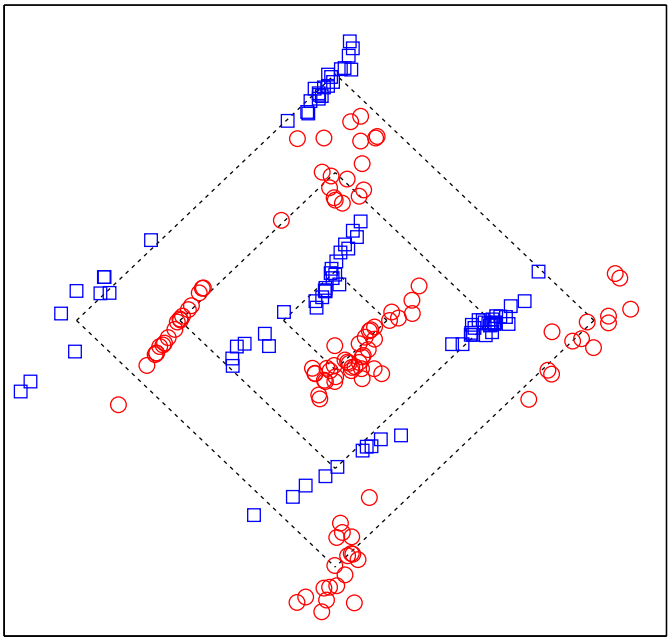


Figure 7.1: NDCC dataset example in 2-dimensions shown with the underlying 1-norm cubes.

at each vertex uses a different (randomly generated) covariance matrix. Some centers generate a relatively small number of points, while others generate a relatively large number of points. Points around opposing vertices of each cube are assigned to opposite classes preventing linear separation.

In our experiment we generate data at vertices of 20-dimensional cubes and add 100 noisy features by sampling from a normal distribution. Thus the data contains a total of 120 features of which 20 are informative. This is a challenging dataset for feature selection because of the high degree of nonlinear interaction among informative features. Methods which rely on marginal contributions of features will perform poorly since projection to any single dimension will not reveal class separation.

We generate 200 training points, 200 validation points and 1000 testing points. Table 7.2 shows test error results using SVM, GMKL, FULL-FS and AO-II, along with the number of correct and incorrect features identified by each method. Note standard SVM is unable to detect a useful model since noisy features drown out any signal. This is clearly an example where feature selection is necessary in order to recover a meaningful model. The best parameter choice for GMKL, determined by cross-validation, yields 15 correct and 6 incorrect features resulting in a test error of 32.0%. In comparison, FULL-FS is able to recover 17 correct features with 1 incorrect one and obtains 16.5% test error. Finally, AO-II is able to identify all 20 features with no incorrect ones and obtains the lowest test error of 10.5%. We also observe that AO-II achieves a lower objective value and does not get stuck at suboptimal solutions.

			Number of Features	
	Objective	Test Error	Correct	Incorrect
SVM	106.1	44.3%	20	100
GMKL	86.1	32.0%	15	6
FULL-FS	75.2	16.5%	17	1
AO-II	62.4	10.5%	20	0

Table 7.2: 20-dimensional NDCC dataset feature selection results. The objective value, test error and the number of correct and incorrect features are shown.

Gender Identification

In this example, we try to identify gender from face images in the FEI database [93]. The database consists of 200 different individuals collected from students and staff at FEI between the ages of 19 and 40. There are 100 male and 100 female subjects. Each image in the database has been aligned to a common template so that pixel-wise features correspond roughly to the same location across all subjects. Images are normalized, equalized, cropped and have been scaled down to have dimensions 18×15. Thus each image consists of 270 pixels of grey scale intensity. Figure 7.2 shows a few examples from the dataset.

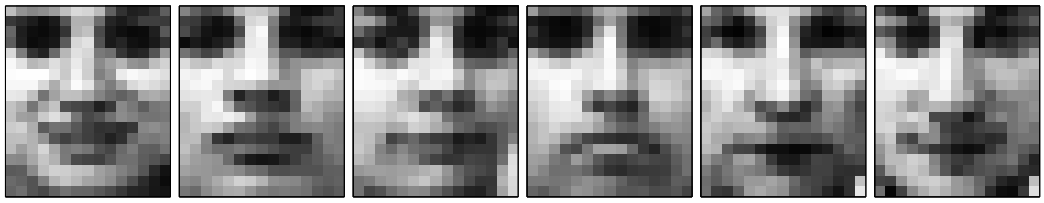


Figure 7.2: Example of a few processed images in the FEI faces dataset.

We follow standard experimental setup and use 167 images for training and 33 for out-of-sample testing. Results are averaged over 5 random splits of the data to reduce variance. Parameters are tuned by running 10-fold cross-validation on the training set for each split. Table 7.3 shows the feature selection results.

	Test Error(%)	Av. # of Features
SVM	10.8 ± 0.8	270.0
GMKL	12.6 ± 1.4	31.4
FULL-FS	11.9 ± 0.9	19.1
AO-II	11.0 ± 0.6	16.2

Table 7.3: Test error and average number of features obtained on FEI faces dataset.

AO-II achieves an error of 11.0% using on average 16 features. In comparison, FULL-FS achieves an error of 11.9% using 19 features and GMKL performs comparatively worse with an error of 12.6% using 31 features. Regular SVM obtains an error of 10.8%. SVM results are obtained using all 270 features. AO-II can obtain similar generalization error with approximately 17 times compression factor. Figure 7.3 shows the average male and female faces superimposed with the features identified by GMKL, FULL-FS and AO-II.

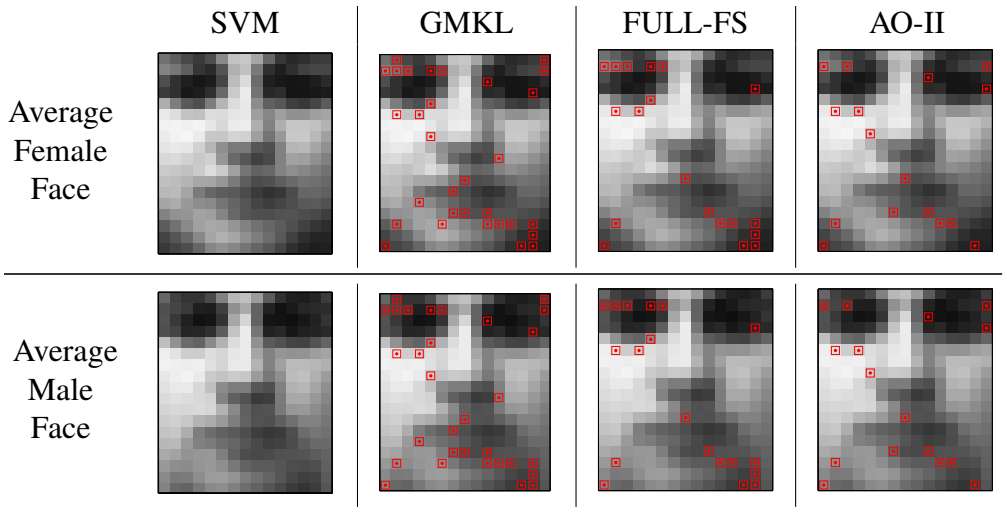


Figure 7.3: The average male and female faces in the FEI dataset superimposed with the features identified by GMKL, FULL-FS and AO-II. Note that SVM uses all 270 features.

Other Datasets

We compare results on several other datasets obtained from UCI repository [44]. Two-thirds of the observations are used for training and the remaining one-third for out-of-sample testing. Results are averaged over 5 (stratified) random splits of the data. Parameters are tuned by running 10-fold cross-validation on the training set for each split. This methodology is used for all datasets, except Madelon. Madelon was used in the NIPS 2003 Feature Selection Challenge [1] and comes with separate training, validation and testing sets.

Table 7.4 summarizes the feature selection results. Average test errors and corresponding average number of features are shown for each dataset. FULL-FS and AO-II improve test error on average by 8% and 14% compared to GMKL, while using 16% and 28% fewer features, respectively. AO-II performs slightly better than FULL-FS in terms of test error and number of features used. A regular SVM using uniform feature weights generally yields similar performance, though FULL-FS and AO-II use significantly fewer features. The exception is the Madelon dataset. Madelon is constructed specifically to evaluate multivariate feature selection and by design contains many noisy features, which lead to poor SVM performance.

		Test Error				Average Number of Features			
	n	SVM	GMKL	FULL-FS	AO-II	SVM	GMKL	FULL-FS	AO-II
Sonar	208	21.1 ± 1.0	16.8 ± 1.1	16.0 ± 1.1	14.9 ± 0.8	60.0	12.9	13.9	12.3
Ion	351	5.1 ± 0.2	6.0 ± 0.4	5.3 ± 0.8	5.3 ± 0.4	33.0	10.6	10.7	7.4
S.A. Heart	462	27.8 ± 0.9	30.6 ± 1.1	29.9 ± 0.7	28.7 ± 0.8	9.0	5.4	3.9	3.8
Musk	476	7.6 ± 0.7	9.3 ± 0.8	9.7 ± 0.7	7.0 ± 0.6	166.0	29.9	32.4	27.5
Wdbc	569	3.7 ± 0.5	5.3 ± 0.4	3.8 ± 0.4	3.8 ± 0.3	30.0	6.3	6.6	4.5
Aust. Credit	690	13.5 ± 0.7	14.3 ± 0.5	13.4 ± 0.4	13.0 ± 0.3	14.0	9.6	5.7	7.0
German Credit	1000	23.2 ± 0.5	23.6 ± 0.8	23.1 ± 0.4	23.4 ± 0.4	24.0	12.0	10.4	9.8
Madelon	2000	40.7	7.7	7.2	7.0	500.0	14.0	10.0	8.0
Avg. improvement rel. to GMKL				7.7%	13.6%			15.8%	27.8%

Table 7.4: Feature selection results on UCI datasets comparing test error and average number of features used. Note, SVM uses all the features in the dataset. n is the number of examples in the dataset. Refer to text for experiment methodology. The last row shows the average percentage improvement compared to GMKL.

7.5.2 Feature Ranking Comparison

In this section we evaluate the ability of FULL-FS and AO-II to rank features. We compare with GMKL as well as three other popular feature selection methods, described below. For the following algorithms, we use the implementations provided in the Spider machine learning toolbox [2].

- **Mutual Information (MI):** A filter method, which uses the mutual information score between candidate features and the output class as a basis to rank features [108]. For discrete random variables, mutual information is given by

$$I(\pi) = \sum_i \sum_j \pi_{ij} \log \frac{\pi_{ij}}{\pi_i \pi_j} ,$$

  where π_{ij} is the probability (frequency) of jointly observing events i and j, and $\pi_i = \sum_j \pi_{ij}$ and $\pi_j = \sum_i \pi_{ij}$ are the marginal probability of events. Continuous features are binned to a discrete set corresponding to index i, while j indexes the binary class output. Higher values of $I(\pi)$ imply greater feature relevance.

- **Relief:** A multivariate filter method, which estimates relevance by determining how well features distinguish classes between nearby points [60]. At each iteration a point is chosen and the weight for each feature is updated according to the distance of the point to its nearest neighbor from the same class (hit) and nearest neighbor from the other class (miss). The final score of a feature is the ratio between the average distance to the nearest miss and nearest hit over all examples.

- **Recursive Feature Elimination (RFE):** A wrapper method that uses a greedy approach to eliminate features, one at a time, that decrease the margin the least [50]. An SVM is trained at each iteration, and the (inverse) margin is computed: $W^2(\boldsymbol{\beta}) = \sum u_i u_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$. For each feature l, $W^2_{(-l)}(\boldsymbol{\beta}) = \sum u_i u_j y_i y_j k(\mathbf{x}_i^{-l}, \mathbf{x}_j^{-l})$ is computed, where $\mathbf{x}_i^{-l}$ means training point i with feature l removed. The feature with the smallest value of $|W^2(\boldsymbol{\beta}) - W^2_{(-l)}(\boldsymbol{\beta})|$ is removed. Repeated application of the procedure results in a ranking of features.

For embedded feature selection methods (GMKL, FULL-FS, AO-II), instead of varying parameter μ to select the required number of features, we obtain rankings by taking the top ranked components of $\mathbf{z}$ at a fixed C and μ. C and μ are chosen by cross-validation to minimize classification error. Similar methodology was used by Varma and Babu [99].

We show test error results versus the number of selected features in Figures 7.4 to 7.13. Each figure corresponds to a dataset used in Section 7.5.1. For a given number of features, we select the top ranked features and relearn an SVM classifier using only the selected features. We use a radial basis kernel and cross-validate to determine optimal SVM parameters, C and σ, for the reduced feature set. Apart from the NDCC and Madelon datasets, each test error point is obtained by averaging results over five trials. In each trial, two-thirds of the data is used for training and one-third for testing. Parameters C and σ are tuned by 10-fold cross-validation on the training set. NDCC and Madelon use separate training, validation and testing sets.

The results show that embedded methods generally perform better than filter and wrapper methods. This is more prominent in the NDCC and Madelon datasets, where there is a complex multivariate relationship among informative features. RELIEF performs well on Madelon, since it is able to capture multivariate relationships, but is not as effective on other datasets. MI performs well when single features are independently significant, for example in German Credit data, but is unable to identify multivariate relationships. Among the embedded methods, AO-II performs the best, followed closely by FULL-FS, and then GMKL. In particular, we see GMKL is not as effective on some datasets, namely NDCC, S.A. Heart, Wdbc, and Aust. Credit, compared to FULL-FS and AO-II.

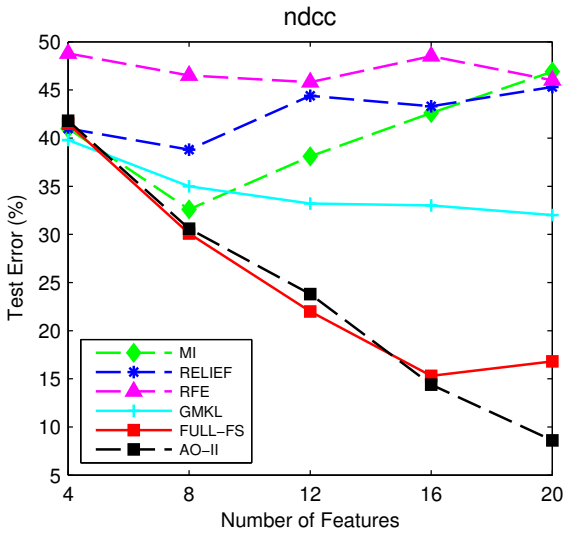


Figure 7.4: Test error as a function of number of features selected for NDCC.

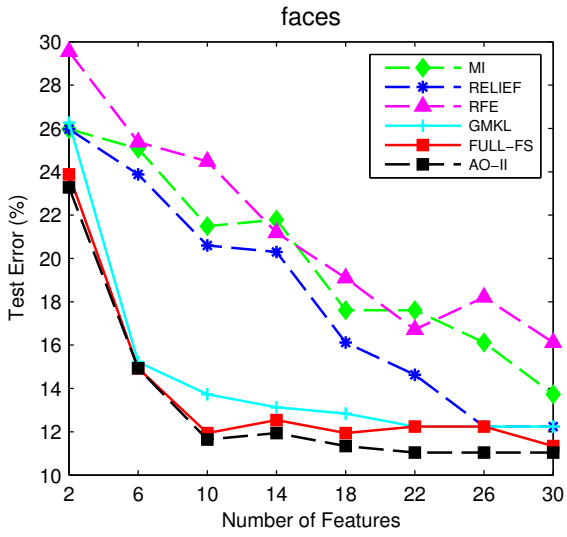


Figure 7.5: Test error as a function of number of features selected for FEI Faces.

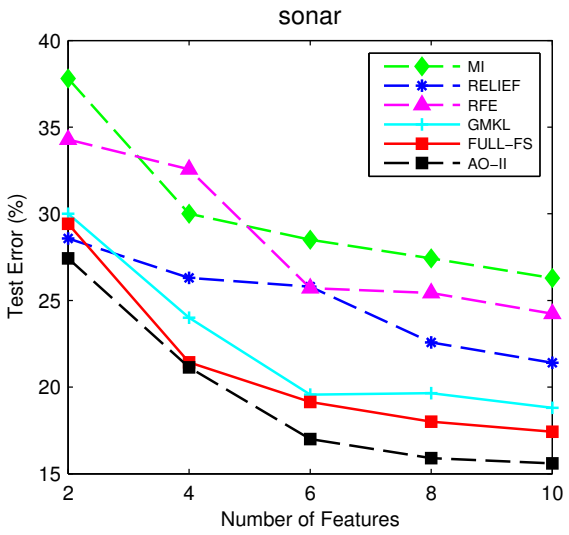


Figure 7.6: Test error as a function of number of features selected for Sonar.

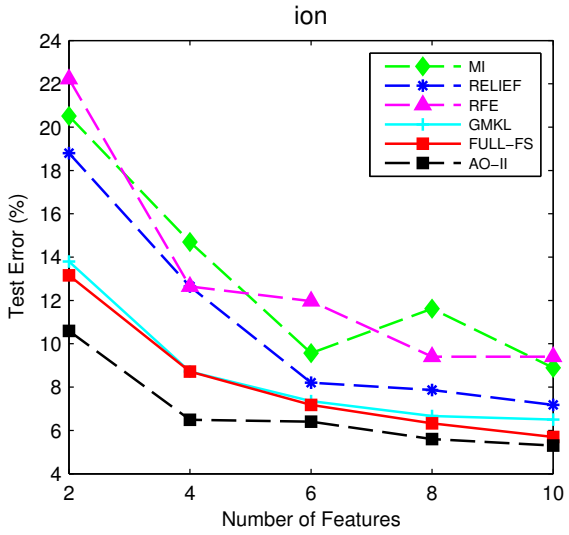


Figure 7.7: Test error as a function of number of features selected for Ion.

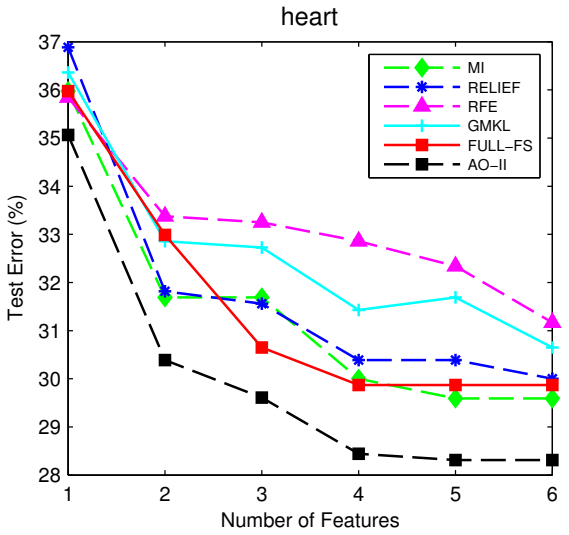


Figure 7.8: Test error as a function of number of features selected for S.A. Heart.

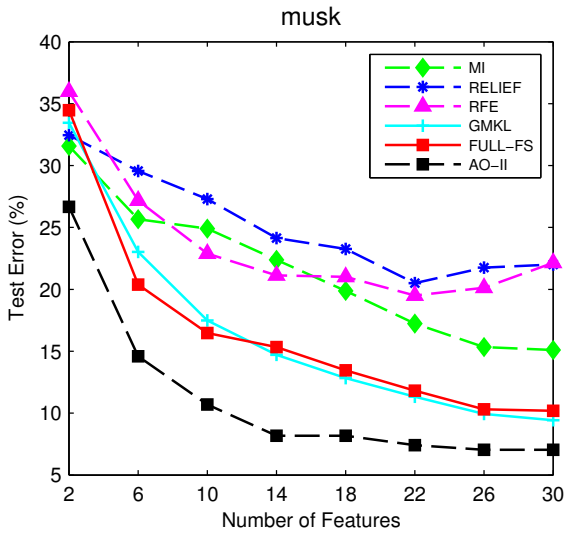


Figure 7.9: Test error as a function of number of features selected for Musk.

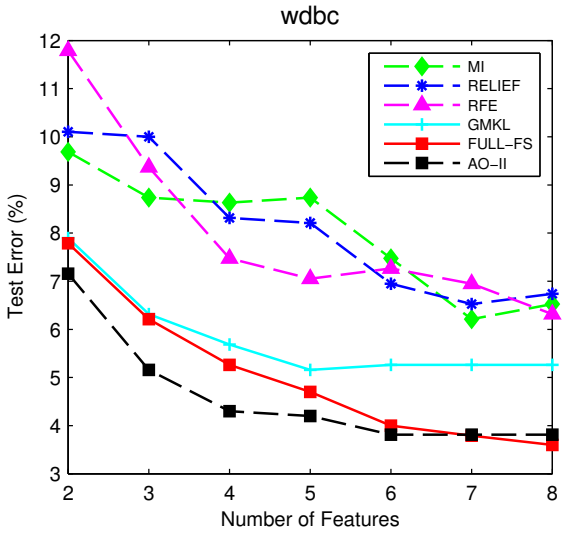


Figure 7.10: Test error as a function of number of features selected for Wdbc.

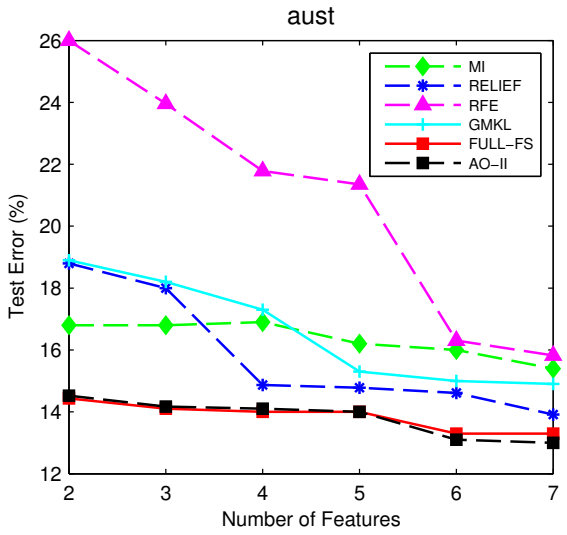


Figure 7.11: Test error as a function of number of features selected for Aust. Credit.

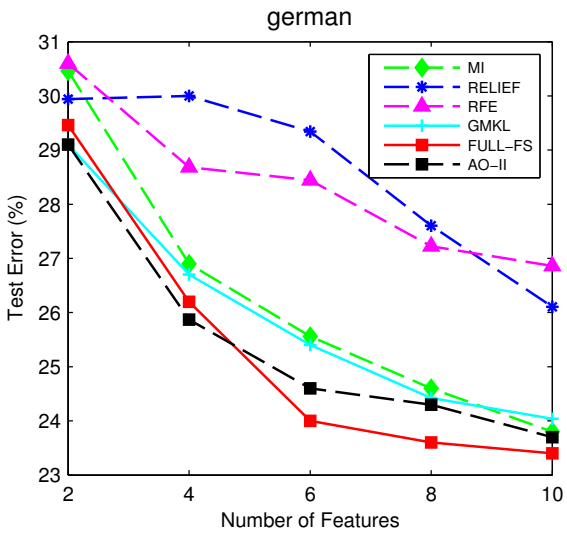


Figure 7.12: Test error as a function of number of features selected for German Credit.

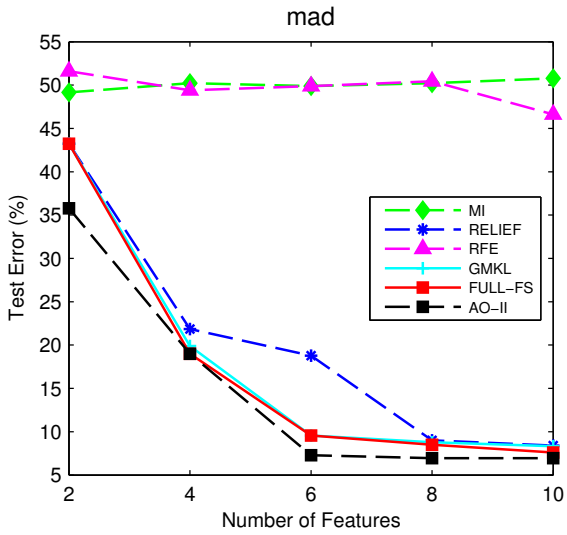


Figure 7.13: Test error as a function of number of features selected for Madelon.

7.6 Summary

In this chapter, we developed an effective algorithm to solve the non-convex optimization problem that results from embedding feature selection in nonlinear SVMs. We solve the primal embedded SVM problem using a trust region method for a bound constrained problem, which is more suitable for non-convex optimization than line-search methods. The trust region algorithm uses local second order approximate models and can guarantee convergence to a minimizer. For computational efficiency, we apply an alternating optimization (AO) framework. We show a naive application of AO can lead to iterates being trapped at saddle points. We extend the space in which AO is performed with an auxiliary variable corresponding to the margin. Sharing the margin variable between AO subproblems reduces saddle point convergence. We further improve solution quality by directly maximizing the functional margin, instead of the geometric margin, in the feature selection subproblem. This focusses on maximizing margin, while permitting greater flexibility, as we optimize over the feature space.

We compare the proposed methods to GMKL, the state-of-the-art embedded SVM feature selection method. GMKL uses a gradient descent algorithm, which does not guarantee convergence to a minimizer for a non-convex problem and can be susceptible to suboptimal solutions. On a simulated dataset we show that GMKL can get stuck at poor solutions and is unable to recover the correct feature subset. On several other real datasets we show that our methods improve upon GMKL by 8-14% in test error while further reducing features by 16-28%. We also show how our methods outperform other leading filter and wrapper approaches in ranking features.

Chapter 8

Conclusion

Kernels are powerful tools to solve statistical machine learning problems. They provide an elegant framework to learn nonlinear models from algorithms designed to learn linear models. A key benefit of kernel methods is separation of the training algorithm from data representation, encoded in the kernel specification. Once the kernel is chosen, the learning algorithm can usually be formulated as a convex optimization problem.

In this thesis, we focus on the quintessential classification problem, which identifies the category of a new observation on the basis of a training set. In spite of the significant success of kernel based classification, as popularized by the support vector machine, there are often several challenges faced in practical problems. In this thesis, we developed effective and efficient solutions to two such challenges associated with kernel based classification: 1) large-scale rare class learning and 2) input feature selection.

Rare class problems are common in many real-world applications. Standard classification algorithms are known to perform poorly in these cases, since they minimize overall classification accuracy, and therefore can lead to biased solutions. Computational scalability is also crucial with the ever increasing amounts of data at our fingertips. In the context of kernel methods, we address these issues by optimizing area under curve (AUC) of the receiver operator characteristic, while limiting the hypothesis space to a linear combination of rare class kernel functions. We argue that optimizing the AUC measure is more appropriate than classification accuracy for rare class problems, as it can avoid biased models due to misspecification of class costs. Maximizing AUC results in a kernel biclass ranking problem. Limiting the hypothesis representation to a linear combination of rare class kernel functions allows us to obtain a computationally efficient algorithm requiring $O(mm_+)$ time and $O(mm_+)$ space, while not sacrificing predictive performance. We call this method

RankRC. The formulation is motivated by using kernel density estimation and probability arguments for rare class datasets. We analyze the solution of RankRC by bounding the difference between RankSVM and a classifier based on a subset of kernel functions. The result indicates that for unbalanced datasets, it is optimal to include rare class kernel functions in the hypothesis representation first, which further justifies the RankRC formulation. Moreover, we establish results for a general regularized loss minimization problem, proving the equivalency of a subset kernel function representation, a projected feature map and the Nyström method. These relationships can be useful to analyze and devise algorithms for other approximate kernel problems as well. Finally, we extend the biclass RankRC problem to multiple levels, and illustrate its benefit on a recent competition problem sponsored by the Heritage Health Provider network.

Another challenge with kernel classification is identifying the most relevant subset of inputs for the learning task. This is known as the input feature selection problem. Most applications contain irrelevant or noisy inputs, which can detract from the quality and accuracy of the resulting model. By identifying the most relevant subset of inputs we can improve generalization error and model interpretability. We develop a primal embedded approach for feature selection in kernel support vector machines. In contrast to using the dual formulation, the primal formulation enables us to use second order methods to solve the resulting non-convex optimization problem. We improve efficiency by considering an alternating optimization scheme. In particular, we are able to improve both effectiveness and efficiency, by sharing an explicit margin variable between alternating optimization iterates. The resulting method is shown to outperform state-of-the-art feature selection methods.

8.1 Future Work

Below we list directions for future work for the rare class representation:

1. Regularization: In problem (3.15) we can use an ℓ_1-regularizer, $\|\boldsymbol{\beta}\|_1$, instead of $\boldsymbol{\beta}^T K_{++}\boldsymbol{\beta}$. This would lead to sparser solutions [95] and could be solved using coordinate descent methods [45].

2. Loss function: We can replace the loss function with other variants of ranking loss. The AUC concentrates uniformly across all threshold levels. We can use weighted AUC [103] or the p-norm push [80] to emphasize specific portions of the AUC curve. Also, we can use list based ranking methods to optimize other criteria such as F_1-

score or Precision/Recall breakeven point [57]. The rare-class representation can enable learning nonlinear models with more complex loss functions in reasonable time and space when the dataset is unbalanced.

3. Stochastic Learning: For very large datasets, the $m \times m_+$ kernel submatrix may be too large to fit in memory. In this case, we can store $K_{++} \in \mathbb{R}^{m_+ \times m_+}$ and cycle (randomly) through majority class examples updating the $\beta \in \mathbb{R}^{m_+}$ vector via gradient descent using an adaptive learning rate [16]. Unlike standard stochastic gradient descent, in each iteration we use the full set of minority examples and a single (or small subset) of majority samples to perform the update. This may lead to faster convergence while using only $O(m_+ m_+)$ space.

4. Fixed Cardinality Representation: More generally, we may consider the problem of finding the optimal subset representation given a cardinality constraint, i.e. $\|\beta\|_0 \leq c$, where c is the maximum number of kernel functions to be used. Since this problem is NP-hard, approximate (e.g. greedy) algorithms could be devised.

Below we list directions for future work for the feature selection problem:

1. Multiple kernel learning: Use second-order optimization methods for more general non-convex multiple kernel learning.

2. Computation scalability: Solving the feature selection problem is not scalable to very large datasets due to solving the trust region subproblem using a full eigen-decomposition. We can explore large-scale approaches by incorporating an iterative method to find a trust region step, using low rank Hessian approximations and updates, and/or considering stochastic methods.

References

[1] (2003). NIPS Workshop on Feature Extraction.

[2] (2013 (Last Accessed)). Spider Machine Learning Toolbox.

[3] (Accessed: 2013-08-31). Heritage Provider Network Health Prize. `http://www.heritagehealthprize.com/c/hhp`.

[4] (Accessed: 2013-08-31). KDD Cup 1999. `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`.

[5] Ablavsky, V. and Sclaroff, S. (2011). Learning parameterized histogram kernels on the simplex manifold for image and action classification. In *ICCV*, pages 1473–1480. IEEE.

[6] Achlioptas, D., McSherry, F., and Schölkopf, B. (2001). Sampling Techniques for Kernel Methods. In *NIPS*, volume 14, pages 335–342.

[7] Aizerman, M. A., Braverman, E. A., and Rozonoer, L. (1964). Theoretical foundations of the potential function method in pattern recognition learning. In *Automation and Remote Control,*, number 25 in Automation and Remote Control,, pages 821–837.

[8] Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404.

[9] Baker, C. T. H. (1977). *The numerical treatment of integral equations*. Clarendon Press.

[10] Bartlett, P. L., Jordan, M. I., and McAuliffe, J. D. (2006). Convexity, Classification, and Risk Bounds. *Journal of the American Statistical Association*, 101(473):138–156.

[11] Batista, G. E. A. P. A., Prati, R. C., and Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.*, 6(1):20–29.

[12] Bezdek, J. C. and Hathaway, R. J. (2002). Some notes on alternating optimization. In *Proceedings of the 2002 AFSS International Conference on Fuzzy Systems (AFSS '02)*, pages 288–300.

[13] Bickel, P., Ritov, Y., and Tsybakov, A. (2008). Hierarchical selection of variables in sparse high-dimensional regression.

[14] Bishop, C. M. (1998). Bayesian pca. In *NIPS*, pages 382–388. The MIT Press.

[15] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, COLT '92, pages 144–152. ACM.

[16] Bottou, L. and Bousquet, O. (2008). The tradeoffs of large scale learning. In *NIPS*, pages 161–168.

[17] Bousquet, O. and Elisseeff, A. (2002). Stability and generalization. *J. Mach. Learn. Res.*, 2:499–526.

[18] Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

[19] Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30:1145–1159.

[20] Bradley, P. S. and Mangasarian, O. L. (1998). Feature selection via concave minimization and support vector machines. In *Machine Learning Proceedings of the Fifteenth International Conference(ICML 98)*, pages 82–90.

[21] Branch, M. A., Coleman, T. F., and Li, Y. (1999). A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM J. Scientific Computing*, 21(1):1–23.

[22] Byun, H. and Lee, S.-W. (2002). Applications of support vector machines for pattern recognition: A survey. In *Pattern Recognition with Support Vector Machines*, pages 213–236.

[23] Chan, A. B., Vasconcelos, N., and Lanckriet, G. R. G. (2007). Direct convex relaxations of sparse svm. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 145–153. ACM.

[24] Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[25] Chapelle, O. (2007). Training a support vector machine in the primal. *Neural Comput.*, 19(5):1155–1178.

[26] Chapelle, O. and Keerthi, S. S. (2010). Efficient algorithms for ranking with svms. *Inf. Retr.*, 13(3):201–215.

[27] Chapelle, O., Vapnik, V., Bousquet, O., and Mukherjee, S. (2002). Choosing multiple parameters for support vector machines. *Mach. Learn.*, 46(1-3):131–159.

[28] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357.

[29] Chawla, N. V., Japkowicz, N., and Kotcz, A. (2004). Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explor. Newsl.*, 6(1):1–6.

[30] Coleman, T. F. and Li, Y. (1994). On the convergence of interior-reflective newton methods for nonlinear minimization subject to bounds. *Mathematical programming*, 67(1-3):189–224.

[31] Coleman, T. F. and Li, Y. (1996). An interior trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on Optimization*, 6(2):415–425.

[32] Cortes, C., Mohri, M., and Talwalkar, A. (2010). On the impact of kernel approximation on learning accuracy. In *Conference on Artificial Intelligence and Statistics*.

[33] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

[34] Courant, R. and Hilbert, D. (1953). *Methods of Mathematical Physics*. Interscience.

[35] Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

[36] Danskin, J. (1967). *The Theory of Max-Min and its Application to Weapons Allocation Problems*.

[37] DeLong, E. R., DeLong, D. M., and Clarke-Pearson, D. L. (1988). Comparing the Areas under Two or More Correlated Receiver Operating Characteristic Curves: A Nonparametric Approach. *Biometrics*, 44(3):837–845.

[38] El Emam, K., Arbuckle, L., Koru, G., Eze, B., Gaudette, L., Neri, E., Rose, S., Howard, J., and Gluck, J. (2012). De-identification methods for open health data: the case of the heritage health prize claims dataset. *J Med Internet Res*, 14(1).

[39] Ezawa, K., Singh, M., and Norton, S. W. (1996). Learning goal oriented bayesian networks for telecommunications risk management. In *ICML*, pages 139–147.

[40] Fan, R.-E., Chen, P.-H., and Lin, C.-J. (2005). Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, 6:1889–1918.

[41] Farahat, A. K., Ghodsi, A., and Kamel, M. S. (2011). A novel greedy algorithm for Nyström approximation. In *AISTATS*, pages 269–277.

[42] Fine, S. and Scheinberg, K. (2002). Efficient svm training using low-rank kernel representations. *J. Mach. Learn. Res.*, 2:243–264.

[43] Fowlkes, C., Belongie, S., Chung, F., and Malik, J. (2004). Spectral grouping using the Nyström method. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2):214–225.

[44] Frank, A. and Asuncion, A. (2010). UCI machine learning repository.

[45] Friedman, J., Hastie, T., and Tibshirani, R. (2009). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*.

[46] Fung, G. M. and Mangasarian, O. L. (2004). A feature selection newton method for support vector machine classification. *Computational Optimization and Applications*, 28(2):185–202.

[47] Gärtner, T. (2003). A survey of kernels for structured data. *SIGKDD Explor. Newsl.*, 5(1):49–58.

[48] Grippo, L. and Sciandrone, M. (2000). On the convergence of the block nonlinear Gauss-Seidel method under convex constraints. *Operations Research Letters*, 26:127–136.

[49] Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182.

[50] Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1-3):389–422.

[51] Hanley, J. A. and Mcneil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36.

[52] Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer New York Inc., New York, NY, USA.

[53] He, H. and Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Trans. on Knowl. and Data Eng.*, 21(9):1263–1284.

[54] Herbrich, R., Graepel, T., and Obermayer, K. (2000). *Large Margin Rank Boundaries for Ordinal Regression*. MIT Press.

[55] Japkowicz, N. and Stephen, S. (2002). The class imbalance problem: A systematic study. *Intell. Data Anal.*, 6(5):429–449.

[56] Joachims, T. (2002). Optimizing search engines using clickthrough data. In *KDD*, pages 133–142.

[57] Joachims, T. (2005). A support vector method for multivariate performance measures. In *ICML*, pages 377–384.

[58] Karakoulas, G. and Shawe-Taylor, J. (1999). Optimizing classifiers for imbalanced training sets. In *NIPS*, pages 253–259.

[59] Kimeldorf, G. and Wahba, G. (1970). A correspondence between Bayesian estimation of stochastic processes and smoothing by splines. *Ann. Math. Statist.*, 41:495–502.

[60] Kira, K. and Rendell, L. (1992). A practical approach to feature selection. In *ML92: Proceedings of the ninth international workshop on Machine learning*, pages 249–256.

[61] Kubat, M., Holte, R., and Matwin, S. (1998). Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, 30:195–215.

[62] Kubat, M. and Matwin, S. (1997). Addressing the curse of imbalanced training sets: One-sided selection. In *ICML*, pages 179–186.

[63] Kumar, S., Mohri, M., and Talwalkar, A. (2009). Sampling techniques for the Nyström method. In *AISTATS*, pages 304–311.

[64] Li, J. and Tao, D. (2012). On preserving original variables in bayesian pca with application to image analysis. *IEEE Transactions on Image Processing*, 21(12):4830–4843.

[65] Lin, Y., Lee, Y., and Wahba, G. (2000). Support vector machines for classification in nonstandard situations. *Machine Learning*, pages 191–202.

[66] Maloof, M. A. (2003). Learning when data sets are imbalanced and when costs are unequal and unknown. In *ICML*.

[67] Marchiori, E. (2005). Feature selection for classification with proteomic data of mixed quality. In *In Proceedings of the 2005 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 385–391.

[68] Mercer, J. (1909). Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Royal Soc. (A)*, 83(559):69–70.

[69] Metz, C. E. (1978). Basic principles of ROC analysis. *Seminars in nuclear medicine*, 8(4):283–298.

[70] Micchelli, C. A., Xu, Y., and Zhang, H. (2006). Universal kernels. *Journal of Machine Learning Research*, 6:2651–2667.

[71] Nakajima, S., Sugiyama, M., and Babacan, S. D. (2011). On bayesian pca: Automatic dimensionality selection and analytic solution. In *ICML*, pages 497–504. Omnipress.

[72] Nguyen, X., Wainwright, M. J., and Jordan, M. I. (2009). On surrogate loss functions and f-divergences. *Annals of Statistics*, 37(2):876–904.

[73] O'Hara, R. and Sillanpaa, M. (2009). A review of Bayesian variable selection methods: What, how, and which. *Bayesian Analysis*, 4:85–118.

[74] Osuna, E. E., Freund, R., and Girosi, F. (1997). Support vector machines: Training and applications. Technical report, MIT.

[75] Platt, J. C. (1999). Advances in kernel methods. chapter Fast training of support vector machines using sequential minimal optimization, pages 185–208. MIT Press.

[76] Platt, J. C. (2005). Fastmap, metricmap, and landmark mds are all Nyström algorithms. In *In Proceedings of 10th International Workshop on Artificial Intelligence and Statistics*, pages 261–268.

[77] Provost, F., Fawcett, T., and Kohavi, R. (1997). The case against accuracy estimation for comparing induction algorithms. In *In Proceedings of the Fifteenth International Conference on Machine Learning*, pages 445–453.

[78] Raskutti, B. and Kowalczyk, A. (2004). Extreme re-balancing for svms: a case study. *SIGKDD Explor. Newsl.*, 6(1):60–69.

[79] Rosset, S., Zhu, J., and Hastie, T. (2003). Margin maximizing loss functions. In *Advances in Neural Information Processing Systems (NIPS 15)*. MIT Press.

[80] Rudin, C. (2009). The p-norm push: A simple convex ranking algorithm that concentrates at the top of the list. *J. Mach. Learn. Res.*, 10:2233–2271.

[81] Sabhnani, M. (2003). Application of machine learning algorithms to kdd intrusion detection dataset within misuse detection context. In *ICML*, pages 209–215.

[82] Schölkopf, B., Herbrich, R., and Smola, A. J. (2001). A generalized representer theorem. In *Proceedings of the 14th Annual Conference on Computational Learning Theory and and 5th European Conference on Computational Learning Theory*, pages 416–426.

[83] Schölkopf, B. and Smola, A. J. (2002). *Learning with Kernels*. MIT Press, Cambridge, MA, USA.

[84] Schölkopf, B., Tsuda, K., and Vert, J. P., editors (2004). *Kernel Methods in Computational Biology*. MIT Press.

[85] Smola, A. J. and Schölkopf, B. (2000). Sparse greedy matrix approximation for machine learning. In *ICML*, pages 911–918.

[86] Sun, Y., Kamel, M. S., Wong, A. K. C., and Wang, Y. (2007). Cost-sensitive boosting for classification of imbalanced data. *Pattern Recogn.*, 40(12):3358–3378.

[87] Talwalkar, A. (2010). *Matrix Approximation for Large-scale Learning*. PhD thesis, Courant Institute of Mathematical Sciences, New York University, New York, NY.

[88] Tan, M., Wang, L., and Tsang, I. W. (2010). Learning sparse svm for feature selection on very high dimensional datasets. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 1047–1054.

[89] Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. A. (2009). A detailed analysis of the kdd cup 99 data set. In *Proceedings of the Second IEEE international conference on Computational intelligence for security and defense applications*, CISDA'09, pages 53–58.

[90] Tayal, A., Coleman, T. F., and Li, Y. (2013a). Bounding the difference between RankRC and RankSVM and application to multi-level rare class kernel ranking. *Submitted to Journal of Machine Learning Research.*

[91] Tayal, A., Coleman, T. F., and Li, Y. (2013b). RankRC: Large-scale nonlinear rare class ranking. *To Be Submitted.*

[92] Tayal, A., Coleman, T. F., and Li, Y. (2014). Primal explicit max margin feature selection for nonlinear support vector machines. *Accepted. To appear in J. Pattern Recognition*.

[93] Thomaz, C. E. and Giraldi, G. A. (2010). A new ranking method for principal components analysis and its application to face image analysis. *Image and Vision Computing*, 28(6):902 – 913.

[94] Thompson, M. E. (2006). NDCC: normally distributed clustered datasets on cubes. www.cs.wisc.edu/dmi/svm/ndcc/.

[95] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *J. Roy. Statist. Soc. Ser. B*, 58(1):267–288.

[96] Turney, P. D. (2000). Types of cost in inductive concept learning. In *ICML*.

[97] Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley, 1st edition.

[98] Varma, M. (2013 (Last Accessed)).

[99] Varma, M. and Babu, B. R. (2009). More generality in efficient multiple kernel learning. In *Proceedings of the 26th International Conference on Machine Learning (ICML '09)*, pages 1065–1072.

[100] Šikonja, M. R. and Kononenko, I. (2003). Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning*, 53(1-2):23–69.

[101] Waegeman, W., Baets, B. D., and Boullart, L. (2006). A comparison of different roc measures for ordinal regression. In *ICML*.

[102] Weiss, G. M. (2004). Mining with rarity: a unifying framework. *SIGKDD Explor. Newsl.*, 6(1):7–19.

[103] Weng, C. G. and Poon, J. (2008). A new evaluation measure for imbalanced datasets. In *Seventh Australasian Data Mining Conference (AusDM 2008)*, volume 87, pages 27–32.

[104] Weston, J., Elisseeff, A., Schölkopf, B., and Tipping, M. (2003). Use of the zero norm with linear models and kernel methods. *Journal of Machine Learning Research*, 3:1439–1461.

[105] Williams, C. and Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press.

[106] Woods, K., Doss, C., Bowyer, K., Solka, J., Preibe, C., and Keglmyer, P. (1993). Comparative evaluation of pattern recognition techniques for detection of microcalcifications in mammography. *International Journal of Pattern Recognition and Artificial Intelligence*, 7:1417–1436.

[107] Wu, G. and Chang, E. Y. (2003). Class-boundary alignment for imbalanced dataset learning. In *ICML*, pages 49–56.

[108] Zaffalon, M. and Hutter, M. (2002). Robust feature selection by mutual information distributions. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, UAI'02, pages 577–584.

[109] Zhang, J. and Mani, I. (2003). KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction. In *ICML*.

[110] Zhang, K., Lan, L., Wang, Z., and Moerchen, F. (2012). Scaling up kernel svm on limited resources: A low-rank linearization approach. pages 1425–1434.

[111] Zhang, K., Tsang, I. W., and Kwok, J. T. (2008). Improved Nyström low-rank approximation and error analysis. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 1232–1239.

[112] Zhou, T., Tao, D., and Wu, X. (2010). Nesvm: A fast gradient method for support vector machines. In *ICDM*, pages 679–688. IEEE Computer Society.

[113] Zhu, J., Rosset, S., Hastie, T., and Tibshirani, R. (2003). 1-norm support vector machines. In *Neural Information Processing Systems*, volume 16.

[114] Zhu, M., Su, W., and Chipman, H. A. (2006). LAGO: A Computationally Efficient Approach for Statistical Detection. *Technometrics*, 48.