Efficient Learning Algorithms for Rankings and Other Combinatorial Concept Classes

末廣, 大貴

https://doi.org/10.15017/1441266

出版情報：九州大学, 2013, 博士（情報科学）, 課程博士
バージョン：

Efficient Learning Algorithms for Rankings and Other Combinatorial Concept Classes

Daiki Suehiro

February, 2014

Abstract

Recently, learning to rank techniques has been attracting increasing attention in various areas such as web search, product recommendation, financial risk analysis, scheduling, and so on. Informally, the task is to obtain an appropriate ranking among items from training data. The amount of data is typically very large in these areas and even rapidly increasing. Therefore, it is particularly important to develop learning algorithms that run efficiently, as well as producing rankings with high quality. In this thesis, we explore efficient algorithms in the following two theoretical learning models, the statistical learning model and the online prediction model.

In the statistical learning model, we consider the bipartite ranking problem, which is one of the most fundamental problems of learning ranking functions. In the bipartite ranking problem, we are given a randomly generated sample consisting of positive and negative instances and required to learn a real-valued function called a ranking function, so that it maps a positive instance to a higher value than a negative instance. It is well known that the problem is reduced to a binary classification problem and so we can apply any of standard learning algorithms such as the Support Vector Machines (SVMs). However, the sample is blown up quadratically in size through the reduction. This means that if we use the 1-norm or 2-norm regularized SVM to obtain a good ranking function, we need to solve a linear or quadratic programming problem of size $O(m^4)$, respectively, where m is the size of the original sample. In this thesis, we reformulate the SVMs for ranking functions as significantly simplified optimization problems of size $O(m^2)$ and give theoretical guarantees on the generalization ability of the ranking functions obtained by solving the optimization problems. In particular, the reformulation of the 1-norm regularized SVM yields the first practical algorithm that is competitive with the original 1-norm regularized SVM in performance.

As an application of our algorithm, we consider the problem of constructing an evalu-

ation function for shogi (Japanese chess) from game records of professional shogi players, where the positions appearing in the records are regarded as positive instances and those not appearing as negative instances. This kind of approach for constructing evaluation functions using machine learning techniques is called the Bonanza method, which is named after the first successful software that uses this approach. Most of the computer shogi softwares developed nowadays use the Bonanza method and its performance is now nearly as well as professional shogi players. But the feature extraction still remains a big problem, because it depends on the algorithm designers to determine what features are to be extracted to form the feature vector that represents a game position. In this thesis, we use the kernel method to generate the feature vector automatically that consists of all n-ary relations among the basic features. The resultant feature vector contains most features adopted by the state-of-the-art computer shogi softwares. We then combine the kernel method with the 2-norm regularized version of our algorithm for learning evaluation functions. Preliminary experiments show promising results.

An online prediction problem is a repeated game between the player and the adversarial environment, where in each trial the player first chooses a decision from a fixed decision space and then adversary returns a feedback. At this trial, the player incurs a loss defined by the decision and the feedback. The goal of the player is to minimize the total loss. Many problems such as web search and product recommendation are also nicely modeled as online prediction problems with the class of rankings (permutations) as the decision space. Similarly, the online versions of various combinatorial optimization problems are modeled as online prediction problems with corresponding combinatorial concept classes as the decision spaces. Examples of such classes are s-t paths in a given graph, spanning trees of a given graph, k-sets of a given set, and so on. Typically, these concept classes contain finite but exponentially many concepts. We have a general scheme called the "follow the regularized leader" (FTRL) for designing an online prediction algorithm with a good performance bound. But in the FTRL two external procedures called the projection and the decomposition are assumed to be implemented. In other words, it seems that we need to design efficient algorithms for the two procedures individually for each concept class. In this thesis, we give efficient algorithms that works uniformly for a wide family of concept classes including permutations, spanning trees, k-sets, based on the methods of submodular function minimization.

Acknowledge

I would like to show my gratitude for everyone's support while studying. First of all I am extremely grateful for Professor Eiji Takimoto. Professor Takimoto is my supervisor and the committee chair of this thesis. His purely theoretical point of view made me grow a lot. I am very thankful to Professor Masayuki Takeda who is a committee member of this thesis and was my supervisor when I was a master course student. He is something of a father figure. When I would start studying computer shogi, he accepted it as the theme of my master thesis with good grace. Professor Jun'ichi Takeuchi and Associate Professor Shuji Kijima gave me many valuable comments as committee members of this thesis. Assoc. Prof. Kijima also taught me the fun of discrete optimization theory and he supported some of the technical materials of this thesis. Assistant Professor Kohei Hatano taught me the culture of machine learning as well as the MATLAB programming. Associate Professor Shunsuke Inenaga advised me about improving presentation from a non-stereotypical angle. Associate Professor Hideo Bannai also gave me bright advice mainly on programming.

The results in this thesis were published in proceedings of GPW 2010 and IBIS 2010 (in Japanese) [71, 70], proceedings of ALT 2011 [74], and proceedings of ALT 2012 [73]. The journal version of GPW 2010 will be published in The IEICE Transactions on Information and Systems (Japanese Edition) [72]. Last but not least, I appreciate all editors, committees, anonymous referees, and publishers.

Contents

Chapter 1

Introduction

Ranking is to order items according to some criterion, and we find everywhere in our society such systems and services that produce rankings. For example, the web search engine produces a list of websites related to given keywords, so that the higher a website appears on the list, the more important it would be for the user. So the web search engine essentially produces rankings among websites. The recommendation system is an information filtering system that finds items such as books, music, or movies and shows them to a user, so that the user would be interested in them. The items recommended to the user are considered to be ranked higher than those not recommended. So the recommendation system produces rankings with ties possibly allowed. Another example is a job scheduling problem, where we are given a set of jobs and are required to find the order of the jobs to be processed by a single machine, so that the flow time is minimized. Here, the flow time is defined as the sum of the process time (including the wait time) over all jobs. The jobs processed earlier are considered to be ranked higher than the those processed later. So this problem is also a problem of producing rankings (i.e., permutations over the given set of jobs).

But how to find a good ranking? For a web search engine, the PageRank [12] is a seminal algorithm that has turned out to be very succesful. But originally, it uses only the link structures of the webpages, and so the ranking produced does not reflect the user's preferences. Nowadays, more refined web ranking algorithms are proposed by using *machine learning techniques* [63, 2], where the algorithms learn the user's preferences from the search history data. For a recommendation system, the method of collaborative filtering has been extensively studied in the machine learning community [36, 85]. Intuitively, the collaborative filtering system learns the similarity of preferences among users

from the users' ratings for items, and predicts a user's rating for an item, which the user has not rated yet.

As just described, *learning to rank*, that is, the problem of constructing good rankings from data has been one of the most active research areas in machine learning and information retrieval in the past decade [21, 44, 30, 23, 11, 64, 7, 54, 27, 65, 14, 39, 49, 86]. The amount of data is typically very large in these areas and even rapidly increasing. Therefore, it is particularly important to develop learning algorithms that run efficiently, as well as producing rankings with high quality. In this thesis, we explore efficient algorithms in the following two theoretical learning models, the statistical learning model and the online prediction model. In the statistical learning model, we consider the bipartite ranking problem, which is one of the most fundamental problems of learning ranking functions. In the online prediction model, we investigate efficient algorithms that work uniformly for various decision spaces including permutations (rankings).

1.1 Bipartite ranking problem

Background and motivation

In the bipartite ranking problem, we are given a randomly generated sample consisting of positive and negative instances and required to learn a real-valued function called a ranking function, so that it maps a positive instance to a higher value than a negative instance [44, 45, 30, 5]. The following recommendation task is modeled by this problem. Imagine that a user is given sample movies for free that are randomly chosen by a movie distribution site. The user is then required to return her evaluations by replying "positive" or "negative" for each sample movie. Based on the evaluations, the movie distribution site learns the preferences of the user as a ranking function over all movies, and recommend new movies of highest ranks according to the ranking function, in hope that the user would buy them.

It is well known that the problem is reduced to a binary classification problem over the new instance space that consists of all pairs of positive and negative instances. So we can apply any of the standard learning algorithms such as the Support Vector Machines (SVMs) over the new instance space. The methods obtained by combining the reduction and SVMs are called the Ranking SVMs. In particular, the algorithms that use the SVMs with the 1-norm or 2-norm of the weight vector regularized are called the 1-norm

Ranking SVM or 2-norm Ranking SVM, respectively. The 2-norm Ranking SVM is a more standard one and can be enhanced with the kernel trick. On the other hand, the 1-norm Ranking SVM is likely to produce linear functions with sparse weight vectors, and thus is suitable for the feature selection. But the complexity issue arises, when the sample is very large. This is caused by the fact that the sample is blown up quadratically in size through the reduction. Thus, naive implementations of the Ranking SVMs are impractical. More specifically, if we use the 1-norm or 2-norm Ranking SVMs, we need to solve linear or quadratic programming problems (LP or QP problems) of size $O(m^4)$, respectively, where m is the size of the original sample.

Surprisingly, for the 2-norm Ranking SVM, efficient algorithms are proposed that solve the $O(m^4)$ size QP problem in $O(m \ln m)$ time [45, 19]. However, it is unclear how the techniques used in these algorithms are applied for solving the LP problem of the 1-norm Ranking SVM. Another approach to solving the 1-norm Ranking SVM would be to use the boosting technique. The RankBoost [30] is a very efficient algorithm that simulates the AdaBoost over the new instance space in $O(m)$ time per iteration. But it does not have any theoretical guarantee on its performance when the sample is not linearly separable. The SoftRankBoost [56] is a modification of the RankBoost based on the smooth boosting framework. It also runs in $O(m)$ time and turns out to have the same guarantee as our algorithm. But the underlying optimization problem is not clear.

Our contributions

In this thesis, we take a different approach by simplifying the LP formulation for the 1-norm Ranking SVM, rather than devising a fast algorithm for it. More precisely, we reformulate the Ranking SVMs as significantly simplified optimization problems of size $O(m^2)$ and give theoretical guarantees on the generalization ability of the ranking functions obtained by solving the optimization problems. In particular, the reformulation of the 1-norm regularized SVM yields the first practical algorithm that is competitive with the original 1-norm regularized SVM in performance.

As an application of our algorithm, we consider the problem of constructing an evaluation function for shogi (Japanese chess) from game records of professional shogi players, where the positions appearing in the records are regarded as positive instances and those not appearing as negative instances. This kind of approach for constructing evaluation functions using machine learning techniques is called the Bonanza method, which is named

after the first successful software that uses this approach. Most of the computer shogi softwares developed nowadays use the Bonanza method and its performance is now nearly as well as professional shogi players. But the feature extraction still remains a big problem, because it depends on the algorithm designers to determine what features are to be extracted to form the feature vector that represents a game position. In this thesis, we use the kernel method to generate the feature vector automatically that consists of all n-ary relations among the basic features. The resultant feature vector contains most features adopted by the state-of-the-art computer shogi softwares. We then combine the kernel method with the 2-norm regularized version of our algorithm for learning evaluation functions. Preliminary experiments show promising results.

1.2 Online prediction for combinatorial concept classes

Background and motivation

Online prediction is a repeated game between the player and the adversarial environment, where in each trial the player first chooses a decision from a fixed decision space and then adversary returns a feedback. At this trial, the player incurs a loss defined by the decision and the feedback. The goal of the player is to minimize the total loss. Many problems such as the web search engine and the recommendation system are also nicely modeled as online prediction problems with the class of rankings (permutations) as the decision space. In particular, discussing these problems in the online prediction model is more suitable when we cannot put any (probabilistic) assumptions on the user's behaviors, whereas in the statistical learning model, we assume that the user's behaviors are statistically stationary. For example, consider the following list access problem. Imagine that a database has a fixed set of items. In each trial, the system chooses a ranking (i.e., a permutation) over the item set and sorts items in a list according to the ranking; Then the user searches for an item (called the target item) in the list; The loss of the system is k if the target item is in the k-th position in the list.

The most basic version of the online prediction problem is the expert problem [82, 53, 15, 40, 78], where the decision space is small enough for enumerating all elements (called the experts). In this case, we can employ a simple algorithm called the Hedge algorithm, which maintains a weight for each expert and makes predictions based on the weighted majority. But when the decision space contains all permutations over n items as in the

above example, then the size of the decision space is $n!$, which is too large to use the Hedge algorithm. Observe here that the decision space (i.e., the set of permutations over n items) is defined in a combinatorial way from a small seed, (i.e., the set of n items), and the seed is only explicitly given to the player.

Similarly, the online versions of various combinatorial optimization problems are modeled as online prediction problems with corresponding combinatorial concept classes as the decision spaces. Examples of such combinatorial concept classes are permutations over a fixed item set [1, 52, 80, 81, 39, 86] as just described, s-t paths in a given graph [75], k-sets of a given universal set [84], spanning trees of a given graph [49, 16], and so on. Typically, these concept classes contain finite but exponentially many concepts.

Most results obtained so far are based on the Follow the Regularized Leader (FTRL) framework [38], which is a general scheme of designing efficient online prediction algorithms with a good performance bound for combinatorial concept classes. In the FTRL framework, two external procedures called the projection and the decomposition are assumed to be implemented. In other words, it seems that we need to design efficient algorithms for the two procedures individually for each concept class.

Our contributions

In this thesis, we give polynomial time algorithms for the projection and the decomposition, provided that the concepts in the class are represented as the vertices of a base polyhedron defined by a submodular function. This implies that we have essentially a single algorithm that works uniformly and efficiently for a wide family of concept classes including permutations, spanning trees, k-sets, and more. The projection and the decomposition algorithms we propose are just simple applications of submodular function minimization. In particular, if the submodular function is cardinality-based, then the corresponding base polyhedron is highly symmetric and using the symmetry we give more efficient $O(n^2)$ time algorithms.

1.3 Organization

The rest of this thesis is organized as follows: In Chapter 2, we describe the statistical learning model and the online prediction model with basic learning algorithms and theory. In Chapter 3, we propose efficient algorithms for the bipartite ranking problem. In

Chapter 4, we describe a method for learning evaluation functions for shogi as an application of our method for bipartite ranking. In Chapter 5, we consider the online prediction problem over combinatorial concept classes and present efficient algorithms that work uniformly for a wide family of concept classes, based on the methods of submodular function minimization. In Chapter 6, we give conclusion of this thesis and discuss our future work.

Chapter 2

Preliminaries

In this chapter, we give the notations to be used throughout this thesis, and introduce two learning models, the statistical learning model and the online prediction model, with basic learning algorithms and theory covered in this thesis.

2.1 Notation

Let $\mathbb{R}$ and $\mathbb{R}_+$ be the set of real numbers and the set of non-negative real numbers, respectively. For an integer u, $[1, u]$ denotes the set $\{1, \ldots, u\}$. For an integer N, $\mathcal{P}_N$ denotes the N-dimensional probability simplex, i.e., $\mathcal{P}_N = \{\boldsymbol{p} \in [0,1]^N \mid \sum_i p_i = 1\}$. Let $I(\cdot)$ denote the indicator function. That is, for a proposition P, $I(P) = 1$ if P is true and $I(P) = 0$ otherwise.

2.2 Statistical learning model

The statistical learning model is one of the most important models for machine learning with ideas from statistics. The statistical learning model deals with the problem of finding a function (e.g. classification function or ranking function) called a hypothesis from randomly chosen data set, so that the hypothesis has a good generalization performance, by which we mean that the hypothesis predicts well the label or rank of an unseen data.

2.2.1 Problem settings

First we give the problem settings that are common to both the binary classification problem and the bipartite ranking problem.

We denote by $X \subseteq \mathbb{R}^N$ the set of *instances* and $Y = \{-1, +1\}$ the set of *labels*. We call a multiset of instance-label pairs $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\} \subseteq X \times Y$ a *sample*, where the number m of the pairs is called the size of S. We call $\boldsymbol{x}_i$ a *positive* instance if $y_i = +1$, and a *negative* instance otherwise. We assume that instances are independently and identically distributed (i.i.d.) according to some unknown distribution D over $X \times Y$. The learner is given a sample S and required to produce a function called a hypothesis h that maps X to $\mathbb{R}$. In particular, we mainly consider linear functions as hypotheses, which are described as

$$h(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b$$

for some weight vector $\boldsymbol{w} \in \mathbb{R}^N$ and real number $b \in \mathbb{R}$.

Binary classification problem

In the binary classification problem, the goal of the learner is to find a hypothesis h such that its *generalization error* with respect to the distribution D is small. The generalization error of h is defined as follows.

Definition 1 (Generalization error)
For a hypothesis $h : X \to \mathbb{R}$ and an underlying distribution D over $X \times Y$, the generalization error *of h is defined by*

$$R_D(h) = \Pr_{(\boldsymbol{x},y)\sim D}[h(\boldsymbol{x})y < 0] = \mathop{\mathrm{E}}_{(\boldsymbol{x},y)\sim D}[I(h(\boldsymbol{x})y < 0)].$$

Note that the generalization error of h is also defined as $\Pr_{(\boldsymbol{x},y)\sim D}[\mathrm{sign}(h(\boldsymbol{x})) \neq y]$, where $\mathrm{sign}(a) = 1$ if $a \geq 0$ and $\mathrm{sign}(a) = -1$ if $a < 0$. The learner cannot exactly compute the generalization error in general because the distribution D is unknown. However, the learner can estimate it by computing the *empirical error* of h on the given sample S, which is defined as follows.

Definition 2 (Empirical error)
For a hypothesis $h : X \to \mathbb{R}$ and a sample $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$, the empirical error *of h is defined by*

$$\hat{R}_S(h) = \frac{1}{m}\sum_{i=1}^{m}(I(h(\boldsymbol{x}_i)y_i < 0)).$$

In other words, the empirical error of h is the fraction of the instances in S that is misclassified by h. Note that for a fixed h, the expectation of the empirical error over

i.i.d. sample S of a fixed size m is equal to the generalization error:

$$\mathop{\mathrm{E}}_{S\sim D^m}[\hat{R}_S(h)] = R(h).$$

Bipartite ranking problem

In the bipartite ranking problem, a function $h : X \to \mathbb{R}$ is called a *ranking function*. The goal of the learner is to find a ranking function h that maximizes the *AUC score* given as follows.

Definition 3 (AUC score)
For a ranking function $h : X \to \mathbb{R}$ and an underlying distribution D over $X \times Y$, the AUC score *of h is defined by*

$$\mathrm{AUC}(h) = \Pr_{(\boldsymbol{x},y),(\boldsymbol{x}',y')\sim D}(h(\boldsymbol{x}) > h(\boldsymbol{x}') \mid y = +1, y' = -1).$$

In a word, the AUC score of h is the probability that h maps a positive instance to a higher value than a negative instance. For convenience, we renumber the indices of the instances in S, so that $\boldsymbol{x}_1^+, \ldots, \boldsymbol{x}_p^+$ and $\boldsymbol{x}_1^-, \ldots, \boldsymbol{x}_n^-$ are the positive and negative instances in S, respectively. Clearly, $m = p + n$ and

$$S = \{(\boldsymbol{x}_1^+, +1), \ldots, (\boldsymbol{x}_p^+, +1)\} \cup \{(\boldsymbol{x}_1^-, -1), \ldots, (\boldsymbol{x}_n^-, -1)\}$$

holds, where the sets appearing in the formula above are all multisets. Similar to the binary classification problem, we define the *empirical AUC score of h at ρ* for a real number $\rho > 0$ as follows.

Definition 4 (Empirical AUC score)
For a hypothesis h, a sample $S = \{(\boldsymbol{x}_1^+, +1), \ldots, (\boldsymbol{x}_p^+, +1)\} \cup \{(\boldsymbol{x}_1^-, -1), \ldots, (\boldsymbol{x}_n^-, -1)\}$, and a real number $\rho > 0$, the empirical AUC score of h at ρ *is defined as*

$$\mathrm{AUC}_{S,\rho}(h) = \frac{1}{pn}\sum_{i=1}^{p}\sum_{j=1}^{n} I\left(\frac{h(\boldsymbol{x}_i^+) - h(\boldsymbol{x}_j^-)}{2} \geq \rho\right).$$

For a linear ranking function $h(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b$ for some weight vector $\boldsymbol{w} \in \mathbb{R}^N$ and $b \in \mathbb{R}$, Rudin and Schapire give the following bounds on the AUC score in terms of the empirical AUC score [65]. Note that for the bipartite ranking problem, we may assume without loss of generality that $b = 0$.

Theorem 1

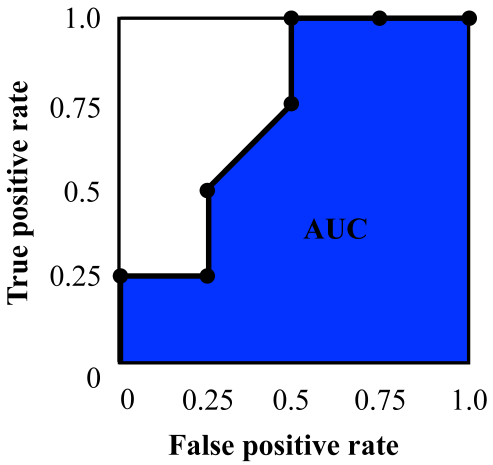
For any $\rho > 0$ and $\delta > 0$, with probability at least $1 - \delta$, the following bound holds for any linear ranking function $h(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x}$:

$$\mathrm{AUC}(h) \geq \mathrm{AUC}_{S,\rho}(h) - \sqrt{\frac{4}{mE^4}\left(\frac{8\ln N}{\rho^2}\ln\left(\frac{4mE^2\rho^2}{\ln N}\right) + 2\ln\left(\frac{2}{\delta}\right)\right)},$$

where

$$E = \mathop{\mathrm{E}}_{(x,y),(x',y')\sim D}[I(y = 1, y' = -1)].$$

The theorem says that a reasonable way of finding a linear function $h(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x}$ with high AUC score is to enlarge $\mathrm{AUC}_{S,\rho}(h)$ for as large ρ as possible.



threshold

	+	→			-			
True label	Pos	Neg	Pos	Pos	Neg	Pos	Neg	Neg
$h(\boldsymbol{x})$	15	7	4	1	1	1	- 3	- 5

Figure 2.1: An example of ROC curve (left) of the ranking function h (right).

As suggested by its name, the empirical AUC score at $\rho = 0$, which is simply called the empirical AUC, coincides with the area under the ROC curve as shown in the left panel of Figure 2.1. An ROC curve plots the *true positive rate* and the *false positive rate*, where the true positive rate is the fraction of positive instances with correctly predicted as positive, and the false positive rate is the fraction of negative instances with incorrectly predicted as positive. Each point on the curve is generated by the value of threshold θ from one extreme to another as in the right panel of Figure 2.1. The value of threshold θ determines the classification of any point $\boldsymbol{x}$ into positive and negative by $\mathrm{sgn}(h(\boldsymbol{x}) - \theta)$. At one extreme, all points are predicted as negative, and hence both the true positive rate and the false positive rate are zero. At the other extreme, all points are predicted as positive, and hence both the true positive rate and the false positive rate are one.

2.2.2 Classification SVM

Support Vector Machines (SVMs) are one of the most successful algorithms for the binary classification problem, since it is well-grounded in theory and effective in practice [24, 17, 43, 59]. It is known that the SVMs are useful not only for classification but also for regression [26, 69, 22] and ranking [45, 60]. In this and the next subsections, we show general frameworks for two types of SVMs: the SVM for the binary classification problem (Classification SVM) and the SVM for the bipartite ranking problem (Ranking SVM). Here we introduce the SVM for the binary classification problem. Given a sample $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$, the Classification SVM outputs a linear hypothesis

$$h(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b$$

for some weight vector $\boldsymbol{w} \in \mathbb{R}^N$ and real number $b \in \mathbb{R}$. We introduce four variants of the Classification SVM: $\{$*Hard, Soft*$\}$ *Margin SVM with* $\{1, 2\}$*-norm of the weight vector regularized*. Among these, the SVMs with 2-norm of the weight vector regularized (2-norm SVMs, for short) are standard ones and can be combined with the kernel method. On the other hand, SVMs with 1-norm of the weight vector regularized (1-norm SVMs, for short) does not seem to enjoy the kernel method, but, the resulting weight vector tends to be sparse and thus they are suitable for the feature selection.

Hard Margin SVMs

The hard margin SVMs find a linear function $h(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b$ that correctly classifies S (i.e., $\hat{R}_S(h) = 0$) if S is linearly separable. The linear functions $(\boldsymbol{w}, b)$ output by the Hard Margin 2-norm SVM and the Hard Margin 1-norm SVM can be expressed as the solutions to the following quadratic and linear programming problems, respectively.

OP 1: Hard Margin 2-norm SVM (primal)

$$\begin{aligned} &\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|_2^2 \\ &\text{sub.to} \\ &y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1, \quad i \in [1, m]. \end{aligned}$$

This formulation is motivated by the margin theory. The constraints imply that the hypothesis $(\boldsymbol{w}, b)$ should correctly classify S. Moreover, the constraints imply that the

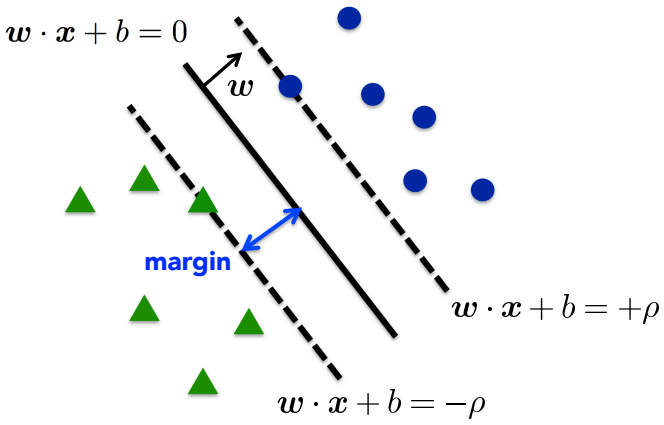


Figure 2.2: Illustration of the Hard Margin 2-norm SVM.

margin is at least $1/\|\boldsymbol{w}\|_2$, where the margin ρ of $(\boldsymbol{w}, b)$ for S is defined as the L_2 (Euclidean) distance between the hyperplane $(\boldsymbol{w}, b)$ and the instance $\boldsymbol{x}_i$ that is closest to $(\boldsymbol{w}, b)$, i.e.,

$$\rho = \min_{(\boldsymbol{x}_i, y_i) \in S} \frac{y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b)}{\|\boldsymbol{w}\|_2}.$$

Therefore, the formulation implies that the solution $(\boldsymbol{w}, b)$ is a hyperplane that maximizes the margin. Figure 2.2 illustrates the Hard Margin 2-norm SVM.

On the other hand, the Hard Margin 1-norm SVM finds a hyperplane $(\boldsymbol{w}, b)$ that maximizes L_∞-margin, where L_∞-margin ρ_∞ is the margin that is now measured with respect to the L_∞ distance, i.e.,

$$\rho_\infty = \min_{(\boldsymbol{x}_i, y_i) \in S} \frac{y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b)}{\|\boldsymbol{w}\|_1}.$$

In the following formulation, we assume without loss of generality that the weight vector $\boldsymbol{w}$ is nonnegative and normalized to 1. In other words, we assume that $\boldsymbol{w} \in \mathcal{P}_N$.

OP 2: Hard Margin 1-norm SVM (primal)

$$\begin{aligned} &\max_{\boldsymbol{w}, b} \rho \\ &\text{sub.to} \\ &y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq \rho, \quad i \in [1, m] \\ &\boldsymbol{w} \in \mathcal{P}_N. \end{aligned}$$

To see why we may assume the weight vectors to be non-negative, observe that any vector $\boldsymbol{w}$ can be written as $\boldsymbol{w} = \boldsymbol{w}^+ - \boldsymbol{w}^-$ for some two non-negative vectors $\boldsymbol{w}^+$ and $\boldsymbol{w}^-$ such

that $\|\boldsymbol{w}\|_1 = \|\boldsymbol{w}^+\|_1 + \|\boldsymbol{w}^-\|_1$. This implies that the reduction $\boldsymbol{x} \mapsto (\boldsymbol{x}, -\boldsymbol{x})$ enables us to learn the general class of hyperplanes $\{(\boldsymbol{w}, b) \mid \|\boldsymbol{w}\|_1 = 1\}$ by a learning algorithm for our class $\{(\boldsymbol{w}, b) \mid \boldsymbol{w} = (\boldsymbol{w}^+, \boldsymbol{w}^-) \in \mathcal{P}_{2N}\}$.

Soft Margin SVMs

The Soft Margin SVMs work well even when the sample S is not linearly separable.

In this thesis, we use the ν-SVM formulation [66].

OP 3: Soft Margin 2-norm SVM (primal)

$$\min_{\rho, \boldsymbol{w}, b, \boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \nu\rho + \frac{1}{m}\sum_{i=1}^{m} \xi_i$$

sub.to

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq \rho - \xi_i, \quad i \in [1, m]$$

$$\boldsymbol{\xi} \geq \boldsymbol{0},$$

$$\rho \geq 0.$$

Here, ν is a parameter that controls the tradeoffs between two objectives, maximizing ρ and minimizing ξ_i's, where ρ is a *target margin* and ξ_i is a penalty term that represents the amount of how the instance $(\boldsymbol{x}_i, y_i)$ violates the target margin ρ. See Figure 2.3 for geometric interpretation.

Next we give the formulation of the Soft Margin 1-norm SVM.

OP 4: Soft Margin 1-norm SVM (primal)

$$\max_{\rho, \boldsymbol{w}, b, \boldsymbol{\xi}} \rho - \frac{1}{\nu}\sum_{i=1}^{m} \xi_i$$

sub.to

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq \rho - \xi_i, \quad i \in [1, m]$$

$$\boldsymbol{\xi} \geq \boldsymbol{0},$$

$$\boldsymbol{w} \in \mathcal{P}_N,$$

$$\rho \geq 0.$$

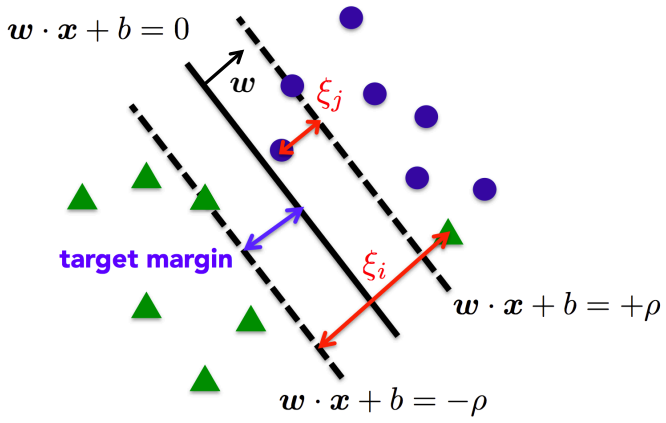


Figure 2.3: Illustration of the Soft Margin 2-norm SVM.

Dual form and kernel method

All the optimization problems given above have dual forms. In particular, the dual form of the Soft Margin 2-norm SVM is described as a quadratic programming program with m variables, $O(m)$ constraints, and a quadratic objective function of m^2 terms, whose coefficients are $-y_i y_j(\boldsymbol{x}_i \cdot \boldsymbol{x}_j)$, $1 \leq i, j \leq m$. Therefore, once we have the inner products $\boldsymbol{x}_i \cdot \boldsymbol{x}_j$ for all $1 \leq i, j \leq m$, the problem size becomes independent of the dimension N. By this property, the 2-norm SVM enjoys the kernel method.

Below let us derive the dual form of the Soft Margin 2-norm SVM. The Lagrangian of OP 3 is given as

$$\begin{aligned} L(\boldsymbol{w}, \boldsymbol{\xi}, b, \rho, \boldsymbol{\alpha}, \boldsymbol{\beta}, \delta) =& \frac{1}{2}\|\boldsymbol{w}\|^2 - \nu\rho + \frac{1}{m}\sum_i^m \xi_i \\ & - \sum_i^m (\alpha_i(y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - \rho + \xi_i) + \beta_i \xi_i) - \delta\rho, \end{aligned}$$

where $\boldsymbol{\alpha} \geq \mathbf{0}$, $\boldsymbol{\beta} \geq \mathbf{0}$ and $\delta \geq 0$ are Lagrangian multipliers that correspond to the first, second and third constraints of OP 3, respectively.

By KKT conditions [79] the solutions of OP 3 should satisfy

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i}^{m} \alpha_i y_i \boldsymbol{x}_i = \boldsymbol{0}, \tag{2.1}$$

$$\frac{\partial L}{\partial \xi_i} = \frac{1}{m} - \alpha_i - \beta_i = 0, \quad i \in [1, m] \tag{2.2}$$

$$\frac{\partial L}{\partial b} = -\sum_{i}^{m} \alpha_i y_i = 0, \tag{2.3}$$

$$\frac{\partial L}{\partial \rho} = -\nu + \sum_{i}^{m} \alpha_i - \delta = 0. \tag{2.4}$$

$$\alpha_i (y_i (\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - \rho + \xi_i) = 0, \quad i \in [1, m], \tag{2.5}$$

$$\beta_i \xi_i = 0, \quad i \in [1, m], \tag{2.6}$$

$$\delta \rho = 0. \tag{2.7}$$

Plugging these equations obtained above into OP 3, we get the following dual form.

OP 5: Soft Margin 2-norm SVM (dual)

$$\max_{\boldsymbol{\alpha}} -\frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\boldsymbol{x}_i \cdot \boldsymbol{x}_j)$$

sub.to

$$0 \leq \alpha_i \leq \frac{1}{m} \quad i \in [1, m],$$

$$\sum_{i}^{m} \alpha_i y_i = 0,$$

$$\sum_{i}^{m} \alpha_i \geq \nu.$$

Note that the last constraint can be made equality.

Now we are ready to explain the kernel method. We extend the framework of learning linear functions to the following scenario: Consider a *feature map* $\Phi : \mathbb{R}^N \to \mathbb{R}^{N_{\text{high}}}$ and let $\tilde{X} \subseteq \mathbb{R}^{N_{\text{high}}}$ be a new instance space (oftend called the *feature space*) defined as $\tilde{X} = \{\Psi(\boldsymbol{x}) \mid \boldsymbol{x} \in X\}$. Then, use the Soft Margin 2-norm SVM to learn a linear function over the new instance space $\tilde{X}$. Typically, we take Ψ to be a non-linear function that maps to a very high dimensional instance space (i.e., $N_{\text{high}} \gg N$), so that the resultant hypothesis $h(\boldsymbol{x}) = \boldsymbol{w} \cdot \Psi(\boldsymbol{x}) + b$ will perform much better than any linear function over the original instance space X. Here $\boldsymbol{w}$ is of dimension N_{high} and it may be impractical to

solve the primal form OP 3 with $\boldsymbol{x}_i$ replaced with $\Psi(\boldsymbol{x}_i)$. Instead, we use the dual form OP 5 with $(\boldsymbol{x}_i \cdot \boldsymbol{x}_j)$ replaced with $(\Psi(\boldsymbol{x}_i) \cdot \Psi(\boldsymbol{x}_j)) = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, where $K : X \times X \to \mathbb{R}$ is the kernel associated with Ψ, defined as

$$K(\boldsymbol{x}, \boldsymbol{x}') = \Psi(\boldsymbol{x}) \cdot \Psi(\boldsymbol{x}').$$

We give the kernelized version of OP 5 as follows.

OP 6: Soft Margin 2-norm SVM with Kernel (dual)

$$\begin{aligned}
&\max_{\boldsymbol{\alpha}} -\frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \\
&\text{sub.to} \\
&0 \leq \alpha_i \leq \frac{1}{m} \quad i \in [1, m], \\
&\sum_{i}^{m} \alpha_i y_i = 0, \\
&\sum_{i}^{m} \alpha_i \geq \nu.
\end{aligned}$$

Clearly, the time complexity of solving OP 5 depends on the time complexity of computing the kernel K. There are many kernels proposed in the literature that can be computed efficiently. A polynomial kernel $K(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x} \cdot \boldsymbol{x}' + 1)^d$ for some constant d is a widely used efficient kernel, which we will use in Chapter 4.

Note that we can also compute the hypothesis $h(\boldsymbol{x}) = \boldsymbol{w} \cdot \Psi(\boldsymbol{x}) + b$ using the kernel. By (2.1) we have

$$\boldsymbol{w} = \sum_{i}^{m} \alpha_i y_i \Psi(\boldsymbol{x}_i),$$

and hence the first term of h can be expressed by

$$\boldsymbol{w} \cdot \Psi(\boldsymbol{x}) = \sum_{i}^{m} \alpha_i y_i \Psi(\boldsymbol{x}_i) \cdot \Psi(\boldsymbol{x}) = \sum_{i}^{m} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}).$$

Let i_1 and i_2 be indices such that $y_{i_1} = +1$, $y_{i_2} = -1$ and $0 < \alpha_{i_1}, \alpha_{i_2} < 1/m$. By (2.2), we have $\beta_{i_1}, \beta_{i_2} > 0$ and hence by (2.6) we have $\xi_{i_1} = \xi_{i_2} = 0$. Therefore, by (2.5) we have

$$\begin{aligned}
\Psi(\boldsymbol{x}_{i_1}) \cdot \boldsymbol{w} + b - \rho = 0, \\
-\Psi(\boldsymbol{x}_{i_2}) \cdot \boldsymbol{w} - b - \rho = 0.
\end{aligned}$$

From these it follows that

$$b = \frac{\boldsymbol{w} \cdot (\Psi(\boldsymbol{x}_{i_1}) + \Psi(\boldsymbol{x}_{i_2}))}{2} = \frac{\sum_i^m \alpha_i y_i (K(\boldsymbol{x}_i, \boldsymbol{x}_{i_1}) + K(\boldsymbol{x}_i, \boldsymbol{x}_{i_2}))}{2}.$$

Note that the 1-norm SVMs cannot be kernelized because, unlike the 2-norm SVMs, the solution vectors $\boldsymbol{w}$ of OP 2 and OP 4 are not the forms of a linear combination of instances in general.

2.2.3 Ranking SVM

As we show below, the bipartite ranking problem can be reduced to the binary classification problem. So we can apply the SVMs to the reduced problem. The resultant algorithms are called the Ranking SVMs.

Assume that we are given a sample

$$S = \{(\boldsymbol{x}_1^+, +1), \ldots, (\boldsymbol{x}_p^+, +1)\} \cup \{(\boldsymbol{x}_1^-, -1), \ldots, (\boldsymbol{x}_n^-, -1)\}$$

with $m = p + n$. Note that the empirical AUC score of a linear ranking function $h(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x}$ at ρ can be rewritten as

$$\mathrm{AUC}_{S,\rho}(h) = \frac{1}{pn} \sum_{\boldsymbol{z} \in S'} I(\boldsymbol{w} \cdot \boldsymbol{z} \geq \rho),$$

where $S' = \{(\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)/2 \mid 1 \leq i \leq p, 1 \leq j \leq n\}$ is called the *pair-sample*.

Thus, the problem of finding h with large empirical AUC score can be seen as the standard binary classification problem of finding a large margin classifier $\boldsymbol{w}$ over the pair-sample. For convenience, we will use $\mathrm{AUC}(\boldsymbol{w})$ and $\mathrm{AUC}_{S,\rho}(\boldsymbol{w})$ to denote $\mathrm{AUC}(h)$ and $\mathrm{AUC}_{S,\rho}(h)$ with $h(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x}$, respectively.

In what follows, we denote by M the set of all indices (i, j) of the pair-sample. That is,

$$M = [1, p] \times [1, n].$$

2-norm Ranking SVMs

We give the formulation of 2-norm Ranking SVM as follows. First, we give hard margin optimization problems.

OP 7: Hard Margin 2-norm Ranking SVM (primal)

$$\begin{aligned}&\max_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|_2^2\\ &\text{sub.to}\\ &\boldsymbol{w}\cdot(\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)/2 \geq 1, \quad (i,j)\in M.\end{aligned}$$

The dual problem is given as

OP 8: Hard Margin 2-norm Ranking SVM (dual)

$$\begin{aligned}&\max_{\boldsymbol{\alpha}} \sum_{i=1}^{p}\sum_{j=1}^{n}\alpha_{ij} - \frac{1}{2}\left\|\sum_{i=1}^{p}\sum_{j=1}^{n}\alpha_{ij}(\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)/2\right\|^2\\ &\text{sub.to}\\ &\alpha_{ij}\geq 0, \quad (i,j)\in M.\end{aligned}$$

Next we give the standard soft margin 2-norm Ranking SVM, which is based on the ν-SVM formulation [66], where ν-SVM is an equivalent variant of SVM and useful for showing a lower bound on the empirical AUC score[1].

OP 9: Soft Margin 2-norm Ranking SVM (primal)

$$\begin{aligned}(\rho^*, \boldsymbol{w}^*, \boldsymbol{\xi}^*) = \min_{\rho,\boldsymbol{w},\boldsymbol{\xi}} &\frac{1}{2}\|\boldsymbol{w}\|_2^2 - \nu\rho + \frac{1}{pn}\sum_{i=1}^{p}\sum_{j=1}^{n}\xi_{ij}\\ &\text{sub.to}\\ &\boldsymbol{w}\cdot(\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)/2 \geq \rho - \xi_{ij}, \quad (i,j)\in M\\ &\xi_{ij}\geq 0, \quad (i,j)\in M.\end{aligned}$$

where $0\leq\nu\leq 1$ is a parameter. By the property of the ν-SVM (see, e.g., [66]), the optimal solution guarantees that the number of pairs $(\boldsymbol{x}_i^+, \boldsymbol{x}_j^-)$ for which $\boldsymbol{w}^*\cdot(\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)/2 \leq \rho^*$ is at most νpn. In other words, we have

$$\mathrm{AUC}_{S,\rho^*}(\boldsymbol{w}^*) \geq 1-\nu. \tag{2.8}$$

The dual problem of is given below:

[1] In the original formulation [66], there is another constraint that $\rho\geq 0$. In our analysis, we omit the constraint for simplicity.

OP 10: Soft Margin 2-norm Ranking SVM (dual)

$$\begin{aligned}\boldsymbol{\alpha}^* = \arg\max_{\boldsymbol{\alpha}} & -\frac{1}{2}\left\|\sum_{i=1}^{p}\sum_{j=1}^{n}\alpha_{ij}(\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)/2\right\|^2 \\ & \text{sub.to} \\ & 0 \le \alpha_{ij} \le \frac{1}{pn}, \quad (i,j) \in M \\ & \sum_{i=1}^{p}\sum_{j=1}^{n}\alpha_{ij} = \nu.\end{aligned}$$

Note that the dual form is of size $O((pn)^2)$, assuming that each inner product between instances is of unit size.

1-norm Ranking SVMs

Below we give the primal form of the hard margin 1-norm Ranking SVM.

OP 11: Hard Margin 1-norm Ranking SVM (primal)

$$\begin{aligned}&\max_{\rho,\boldsymbol{w}} \rho \\ &\text{sub.to} \\ &\boldsymbol{w}\cdot(\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)/2 \ge \rho, \quad (i,j) \in M, \\ &\boldsymbol{w} \in \mathcal{P}_N.\end{aligned}$$

This optimization problem is for finding $\boldsymbol{w}$ that maximizes the margin ρ such that $\mathrm{AUC}_{S,\rho}(\boldsymbol{w}) = 1$. So, if the sample is not linearly separable, then we have $\rho < 0$ and Theorem 1 says nothing about $\mathrm{AUC}(\boldsymbol{w})$. The dual form is given as follows.

OP 12: Hard Margin 1-norm Ranking SVM (dual)

$$\begin{aligned}&\min_{\gamma,\boldsymbol{d}} \gamma \\ &\text{sub.to} \\ &\sum_{i,j} d_{ij}(\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)/2 \le \gamma\mathbf{1}, \\ &\boldsymbol{d} \in \mathcal{P}_{pn}.\end{aligned}$$

We give the primal form of the soft margin 1-norm Ranking SVM.

OP 13: Soft Margin 1-norm Ranking SVM (primal)

$$
\begin{aligned}
(\rho^*, \boldsymbol{w}^*, \boldsymbol{\xi}^*) = \arg\max_{\rho, \boldsymbol{w}, \boldsymbol{\xi}} & \rho - \frac{1}{\nu} \sum_{i=1}^{p} \sum_{j=1}^{n} \xi_{ij} \\
& \text{sub.to} \\
& \boldsymbol{w} \cdot (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)/2 \geq \rho - \xi_{ij}, \quad (i,j) \in M \\
& \boldsymbol{w} \in \mathcal{P}_N, \\
& \xi_{ij} \geq 0, \quad (i,j) \in M.
\end{aligned}
$$

By using the KKT conditions, we can show that the optimal solution guarantees that the number of indices $(i,j) \in M$ for which $\boldsymbol{w}^* \cdot (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)/2 \leq \rho^*$ is at most ν [66, 83]. In other words, we have

$$\mathrm{AUC}_{S,\rho^*}(\boldsymbol{w}^*) \geq 1 - \frac{\nu}{pn}. \tag{2.9}$$

Note that ρ^* depends on ν and becomes positive when ν is large, even if the sample is not linearly separable. So, the soft margin 1-norm Ranking SVM is quite a robust approach for obtaining a linear function with high AUC score.

The dual problem is given as

OP 14: Soft Margin 1-norm Ranking SVM (dual)

$$
\begin{aligned}
(\gamma^*, \boldsymbol{d}^*) = \arg\min_{\gamma, \boldsymbol{d}} & \gamma \\
& \text{sub.to} \\
& \sum_{i,j} d_{ij} (\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)/2 \leq \gamma \cdot \mathbf{1}, \\
& 0 \leq d_{ij} \leq \frac{1}{\nu}, \quad (i,j) \in M \\
& \boldsymbol{d} \in \mathcal{P}_{pn}.
\end{aligned}
$$

Note that the size of the soft margin 1-norm Ranking SVM is $O(pn(pn+N))$.

2.3 Online prediction model

In this section, we introduce the online prediction model with a general scheme, called the Follow the Regularized Leader (FTRL), of designing online prediction algorithms.

Particularly in this thesis, we deal with the online prediction problem over *combinatorial concept classes*, where the decision space is defined by some combinatorial way from

a given simple concept (called a seed) such as a set and a graph. Examples of such classes are rankings (permutations) over a given set, s-t paths in a given graph, spanning trees of a given graph, k-sets of a given set, and so on. Typically, those concept classes contain finite but exponentially many concepts. Later in this section, we will give the formal definition of the online prediction problem over a family of combinatorial concept classes.

2.3.1 Problem settings

We assume a finite class $\mathcal{C}$ of concepts, where each concept in $\mathcal{C}$ is encoded as a non-negative vector in $\mathbb{R}^n$ for some integer n, i.e., $\mathcal{C} \subseteq \mathbb{R}^n_+$. *The online prediction problem over a concept class* $\mathcal{C}$ is described as a repeated game between the player and the adversary as follows: For each trial $t = 1, \ldots, T$,

1. the player predicts $\boldsymbol{c}_t \in \mathcal{C}$,
2. the adversary returns a loss vector $\boldsymbol{\ell}_t \in [0, 1]^n$,
3. the player incurs loss $\boldsymbol{c}_t \cdot \boldsymbol{\ell}_t$.

The goal of the player is to minimize the regret:

$$\sum_{t=1}^{T} \boldsymbol{c}_t \cdot \boldsymbol{\ell}_t - \min_{\boldsymbol{c} \in \mathcal{C}} \sum_{t=1}^{T} \boldsymbol{c} \cdot \boldsymbol{\ell}_t.$$

The first term represents the cumulative loss of the algorithm, and the second term the cumulative loss of the best concept in the class $\mathcal{C}$ in hindsight.

2.3.2 Bregman divergence

The FTRL is specified by a Bregman divergence. Here we give the definition of Bregman divergences and two examples. See, e.g. [38] for reference.

Let $\Phi : \Gamma \to \mathbb{R}$ be a strictly convex function defined on a closed convex set $\Gamma \subseteq \mathbb{R}^n$. The Bregman divergence $\Delta_\Phi : \Gamma \times \Gamma \to \mathbb{R}_+$ with respect to Φ is defined as

$$\Delta_\Phi(\boldsymbol{p}, \boldsymbol{q}) = \Phi(\boldsymbol{p}) - \Phi(\boldsymbol{q}) - \nabla\Phi(\boldsymbol{q}) \cdot (\boldsymbol{p} - \boldsymbol{q}),$$

where $\nabla\Phi(\boldsymbol{q})$ is the gradient of Φ at $\boldsymbol{q}$. The function Φ is *separable* if there exists functions $\phi_i : \Gamma_i \to \mathbb{R}$ for $i = 1, 2, \ldots, n$ such that $\Gamma = \Gamma_1 \times \Gamma_2 \times \cdots \times \Gamma_n$ and for any $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \Gamma$, $\Phi(\boldsymbol{x}) = \sum_{i=1}^n \phi_i(x_i)$. In particular, if all ϕ_i's are the same, then the

function Φ is said to be *uniformly separable*. In this thesis, we will sometimes consider two particular uniformly separable convex functions, the *2-norm function*:

$$\Phi_{\mathrm{EUC}}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|_2^2 = \frac{1}{2}\sum_{i=1}^{n} x_i^2$$

defined on $\mathbb{R}^n$, and the *unnormalized negative entropy function*:

$$\Phi_{\mathrm{URE}}(\boldsymbol{x}) = \sum_{i=1}^{n}(x_i \ln x_i - x_i)$$

defined on $\mathbb{R}^n_+$. It is well known that these functions define the Euclidean distance and the unnormalized relative entropy, respectively. That is,

$$\Delta_{\mathrm{EUC}}(\boldsymbol{x}, \boldsymbol{z}) \stackrel{\mathrm{def}}{=} \Delta_{\Phi_{\mathrm{EUC}}}(\boldsymbol{x}, \boldsymbol{z}) = \frac{1}{2}\sum_{i=1}^{n}(x_i - z_i)^2,$$

and

$$\Delta_{\mathrm{URE}}(\boldsymbol{x}, \boldsymbol{z}) \stackrel{\mathrm{def}}{=} \Delta_{\Phi_{\mathrm{URE}}}(\boldsymbol{x}, \boldsymbol{z}) = \sum_{i=1}^{n} x_i \ln \frac{x_i}{z_i} + \sum_{i=1}^{n} z_i - \sum_{i=1}^{n} x_i.$$

2.3.3 Follow the Regularized Leader

Let us fix a Bregman divergence $\Delta_\Phi : \Gamma \times \Gamma \to \mathbb{R}_+$ by which the FTRL is specified. We assume that Γ is large enough so that $\mathcal{C} \subseteq \Gamma$. The FTRL uses two external procedures called the *Projection* and the *Decomposition*.

The procedure Projection is defined as follows, where $\mathrm{conv}(\mathcal{C})$ denotes the convex hull of $\mathcal{C}$.

Definition 5 (Projection)
The Projection *is a procedure that inputs a point $\boldsymbol{z} \in \Gamma$ and outputs a point in $\mathrm{conv}(\mathcal{C})$ that is closest to $\boldsymbol{z}$ with respect to Δ_Φ. That is,*

$$\mathrm{Projection}(\boldsymbol{z}) = \arg\inf_{\boldsymbol{x}\in\mathrm{conv}(\mathcal{C})} \Delta_\Phi(\boldsymbol{x}, \boldsymbol{z}).$$

The procedure Decomposition is defined as follows:

Definition 6 (Decomposition)
The Decomposition *is a randomized procedure that inputs a point $\boldsymbol{x}$ in $\mathrm{conv}(\mathcal{C})$ and outputs a point $\boldsymbol{c} \in \mathcal{C}$ randomly so that $\mathrm{E}[\boldsymbol{c}] = \boldsymbol{x}$.*

Algorithm 1 FTRL with Projection and Decomposition

1. Let $\boldsymbol{x}_1$ be any point in $\mathrm{conv}(\mathcal{C})$.
2. For $t = 1, \ldots, T$
   (a) Run Decomposition($\boldsymbol{x}_t$) and get $\boldsymbol{c}_t \in \mathcal{C}$ randomly so that $\mathrm{E}[\boldsymbol{c}_t] = \boldsymbol{x}_t$.
   (b) Predict $\boldsymbol{c}_t$ and incur loss $\boldsymbol{c}_t \cdot \boldsymbol{\ell}_t$.
   (c) Update $\boldsymbol{x}_{t+\frac{1}{2}}$ as $\boldsymbol{x}_{t+\frac{1}{2}} = \arg\min_{\boldsymbol{x}\in\Gamma}\big(\eta\boldsymbol{x}\cdot\boldsymbol{\ell}_t + \Delta_\Phi(\boldsymbol{x}, \boldsymbol{x}_t)\big)$.
   (d) Run Projection($\boldsymbol{x}_{t+\frac{1}{2}}$) and get $\boldsymbol{x}_{t+1}$, which is the projection of $\boldsymbol{x}_{t+\frac{1}{2}}$ onto $\mathrm{conv}(\mathcal{C})$. That is, $\boldsymbol{x}_{t+1} = \arg\inf_{\boldsymbol{x}\in\mathrm{conv}(\mathcal{C})} \Delta_\Phi(\boldsymbol{x}, \boldsymbol{x}_{t+\frac{1}{2}})$.

We give the FTRL in Algorithm 1.

The following regret bounds with respect to Euclidean distance and unnormalized relative entropy are known.

Theorem 2 ([31, 89, 38])

1. *Let $\Phi = \Phi_{\mathrm{EUC}}$. Then, for an appropriate choice of η, the expected regret of the FTRL is $O(D_{\mathrm{EUC}}\sqrt{nT})$, where $D_{\mathrm{EUC}} = \max_{\boldsymbol{c},\boldsymbol{c}'\in\mathcal{C}} \|\boldsymbol{c} - \boldsymbol{c}'\|_2$.*
2. *Let $\Phi = \Phi_{\mathrm{URE}}$. Then, for an appropriate choice of η, the expected regret of the FTRL is $O(\sqrt{L^* D_{\mathrm{URE}}} + D_{\mathrm{URE}})$, where $D_{\mathrm{URE}} = \max_{\boldsymbol{c}\in\mathcal{C}} \Delta_{\mathrm{URE}}(\boldsymbol{c}, \boldsymbol{x}_1)$ and $L^* = \min_{\boldsymbol{c}\in\mathcal{C}} \sum_{t=1}^{T} \boldsymbol{c}\cdot\boldsymbol{\ell}_t$.*

2.3.4 Combinatorial concept classes

Now, we give the formal definition of combinatorial concept classes. A family of combinatorial concept classes $\mathcal{F}$ is defined by a pair $(\mathcal{S}, \phi)$, where $\mathcal{S}$ is a language over a finite alphabet and $\phi : \mathcal{S} \to 2^{\mathbb{R}^n_+}$ is called a semantic function that maps $s \in \mathcal{S}$ to a concept class $\mathcal{C} = \phi(s) \subseteq \mathbb{R}^n_+$. We refer to $\mathcal{S}$ as a *seed set*. Thus, the family $\mathcal{F}$ defines the family of combinatorial concept classes $\{\phi(s) \mid s \in \mathcal{S}\}$.

In this thesis, we consider *an online prediction problem for a family $\mathcal{F} = (\mathcal{S}, \phi)$ of combinatorial concept classes*, where the algorithm is required to work efficiently and uniformly over all concept class $\mathcal{C}$ in the family $\mathcal{F}$. So, we assume that the algorithm first receives a seed $s \in \mathcal{S}$ before starting the protocol, and then makes predictions from the concept class $\mathcal{C} = \phi(s)$. Consequently, in order to employ the FTRL, we need to design

algorithms for the Projection and the Decomposition so that they receive a seed $s \in \mathcal{S}$ as well and work efficiently for the concept class $\mathcal{C} = \phi(s)$.

2.3.5 Examples

Below we show some examples of a family of combinatorial concept classes.

Experts The classical expert problem [31] is an example of online prediction problem over a family of combinatorial concept classes. In the expert problem, we are given n experts who give advice to the player, and the player would like to predict nearly as well as the best expert in hindsight. This problem can be seen as the online prediction problem for the following family $\mathcal{F} = (\mathcal{S}, \phi)$: The seed set $\mathcal{S}$ is the set of natural numbers, and for a seed n, the concept class $\phi(n)$ is the set $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\} \subseteq \{0,1\}^n$, where $\boldsymbol{e}_i$ is a unit vector whose i-th component is 1 and other components are 0.

k-sets The class of k-sets is a generalization of Experts, where each concept corresponds to a set of k experts among n experts. This problem was first considered by [84]. This problem can be described by the following family $\mathcal{F} = (\mathcal{S}, \phi)$: The seed set $\mathcal{S}$ is the set of pairs (n, k) of natural numbers with $k \leq n$, and for a seed $s = (n, k)$, the concept class $\phi(s)$ is defined by $\{\boldsymbol{c} \in \{0,1\}^n \mid \sum_i^n c_i = k\}$.

s-t paths The online shortest path problem is studied in many papers [49, 76]. Here the seed set $\mathcal{S}$ is the set of directed graphs $G = (V, E)$ with two special nodes s and t, and for a seed G, the concept class is defined by

$$\phi(G) = \{\boldsymbol{c} \in \{0,1\}^E \mid \text{the edge set } \{e \in E \mid c_e = 1\} \text{ forms an } s\text{-}t \text{ paths in } G\}.$$

Spanning trees Online prediction problems of undirected or directed spanning trees are studied in [16] and [49]. Here we consider undirected versions. Then the seed set $\mathcal{S}$ is the set of undirected graphs $G = (V, E)$ and for a seed G, the concept class is defined by

$$\phi(G) = \{\boldsymbol{c} \in \{0,1\}^E \mid \text{the edge set } \{e \in E \mid c_e = 1\} \text{ forms a spanning tree of } G\}.$$

Permutations The online prediction problem for this class models an online scheduling problem of n jobs with a single processor where the sum of flow time over all jobs is to be

minimized [86]. Here the seed set $\mathcal{S}$ contains natural numbers and the concept class $\phi(n)$ for seed n is the set of all permutations S_n over $\{1, \ldots, n\}$. A different representation of permutations and the related problem was also considered in [39].

Set covers The online set cover problem is studied in [33]. The problem is described by the following family $\mathcal{F} = (\mathcal{S}, \phi)$: The seed set $\mathcal{S}$ be a subset family U of some ground set X such that $\bigcup_{u \in U} u = X$, and for a seed U, the concept class is

$$\phi(U) = \{\boldsymbol{c} \in \{0,1\}^U \mid \{u \in U \mid c_u = 1\} \text{ is a cover of } X\}.$$

Chapter 3

Efficient Reformulation for 1-norm Ranking SVM

3.1 Introduction

Among the problems related to learning to rank, the bipartite ranking is a fundamental problem, which involves learning to obtain rankings over positive and negative instances. More precisely, for a given sample consisting of positive and negative instances, the goal of the bipartite ranking problem is to find a real-valued function h, which is referred to as a ranking function, with the following property: For a randomly chosen test pair of positive instance $\boldsymbol{x}^+$ and negative instance $\boldsymbol{x}^-$, the ranking function h maps $\boldsymbol{x}^+$ to a higher value than $\boldsymbol{x}^-$ with high probability. Thus, a natural measure for evaluating the goodness of ranking function h is the probability that $h(\boldsymbol{x}^+) > h(\boldsymbol{x}^-)$, which we call the AUC (the area under the receiver operating characteristic (ROC) curve) of h.

It is known that the bipartite ranking problem can be reduced to a binary classification problem over a new instance space, consisting of all pairs $(\boldsymbol{x}^+, \boldsymbol{x}^-)$ of positive and negative instances. More precisely, the problem of maximizing the AUC is equivalent to finding a binary classifier f of the form of $f(\boldsymbol{x}^+, \boldsymbol{x}^-) = h(\boldsymbol{x}^+) - h(\boldsymbol{x}^-)$ so that the probability that $f(\boldsymbol{x}^+, \boldsymbol{x}^-) > 0$ is maximized for a randomly chosen instance pair. Several studies including the Ranking SVM have taken this approach with a linear classifier $f(\boldsymbol{x}^+, \boldsymbol{x}^-) = \boldsymbol{w} \cdot (\boldsymbol{x}^+ - \boldsymbol{x}^-)$ for some weight vector $\boldsymbol{w}$ as the ranking function. The Ranking SVM is justified by generalization bounds [55] which say that a large margin over pairs of positive and negative instances in the sample implies a high AUC score under the standard assumption that instances are drawn i.i.d. under the underlying distribution.

Note that a naive implementation of the reduction approach for bipartite ranking problem is impractical since the sample constructed through the reduction (called the pair-sample) is of size pn when the original sample consists of p positive and n negative instances. This implies that we need to solve a quadratic problem (QP) problem of size $O(pn(pn+N))$ in the primal form or of size $O((pn)^2)$ in the dual form (with a kernel), where N is the dimension of the instance space. To overcome this inefficiency, Joachims proposed a Cutting-Plane based algorithm called the SVM-Perf [45], which simulates the Ranking SVM in $O(s(p+n)\log(p+n))$ time, where s is the maximum number of non-zero features in the given instances. Chapelle and Keerthi proposed another efficient implementation of the Ranking SVM, called the PRSVM [19], which runs in $O((p+n)\log(p+n)+N(p+n))$ time. Note that the Ranking SVM simulated in the above implementations is based on the standard SVM formulation. That is, they solve a soft margin optimization problem with 2-norm of the weight vector regularized.

In this thesis, we consider a 1-norm SVM formulation, which is originally proposed by Bradley and Mangasarian [10] not for the ranking but for the classification problem, where the 1-norm of the weight vector is regularized. This version of Ranking SVM is called the 1-norm Ranking SVM. It has some advantages over the standard Ranking SVM: It is formulated as a linear programming (LP) problem and thus can be solved much faster if its size is not too large. Moreover, note that again, the resulting weight vector tends to be sparse and thus is suitable for the feature selection. Unfortunately, the LP problem for the 1-norm Ranking SVM is naturally of size $O(pn(pn+N))$, the same as the QP problem for the standard Ranking SVM. But in this case, it is unclear how the techniques used in SVM-Perf and PRSVM are applied for solving the LP problem efficiently.

To avoid the difficulty, we take a different approach by simplifying the LP formulation for the 1-norm Ranking SVM, rather than devising a fast algorithm for it. More precisely, we reformulate the LP problems for hard as well as soft margin optimization, with additional constraints that the dual variables d_{ij} are restricted to the product form $d_{ij} = d_i^+ d_j^-$, where i and j range over the p positive and n negative instances, respectively. With these constraints, the number of variables is reduced from pn to $p+n$, which results in greatly simplified optimization problems of size $O((p+n)(p+n+N))$. We call the simplified problems $\mathrm{OPT}_{\mathrm{hard}}$ and $\mathrm{OPT}_{\mathrm{soft}}$ that correspond to the hard and soft margin optimization, respectively. Apparently, they would have worse solutions (i.e., weight vectors with less margins) than the original hard and soft margin 1-norm Ranking SVMs, because the

additional constraints significantly reduce the feasible solution spaces. But surprisingly, $\mathrm{OPT}_{\mathrm{hard}}$ turns out to be equivalent to the original hard margin 1-norm Ranking SVM. In other words, the optimal solution (d^*_{ij}) of the hard margin 1-norm Ranking SVM and the optimal solution $(\hat{d}^+_i, \hat{d}^-_j)$ of $\mathrm{OPT}_{\mathrm{hard}}$ actually have the relation $d^*_{ij} = \hat{d}^+_i \hat{d}^-_j$. This motivates the reformulation for the soft margin optimization, i.e., $\mathrm{OPT}_{\mathrm{soft}}$.

Unfortunately, unlike in the case of the hard margin optimization, the equivalence between $\mathrm{OPT}_{\mathrm{soft}}$ and the soft margin 1-norm Ranking SVM does not hold. To make matters worse, $\mathrm{OPT}_{\mathrm{soft}}$ is not a convex problem, let alone an LP problem. Nevertheless, we show that a feasible solution can be found by solving an LP problem that is obtained from $\mathrm{OPT}_{\mathrm{soft}}$ by fixing a parameter, and the solution has a certain amount of margin. Furthermore, if the given sample is close to be linearly separable, then our theoretical guarantee on the margin becomes close to that of the soft margin 1-norm Ranking SVM. We also give a practical heuristic to improve the feasible solution up to a local optimum.

Although, as mentioned above, several efficient algorithms for the 2-norm Ranking SVM have been proposed [45, 19], we show that our reformulation technique can be extended to the formulation of the 2-norm Ranking SVM, which yield simplified QP problems of size $O((p+n)N)$ in the primal form and of size $O(pn)$ in the dual form.

We conduct several experiments using artificial and real data sets. Surprisingly, the results show that our methods not only run much faster than most of the previously proposed algorithms as expected, they achieve relatively high AUC scores for many data sets.

Related works

Several related works have been done in the literature. Yu and Kim proposed a different notion of 1-norm Ranking SVM [87], where the weight vector is restricted to a linear combination of support vectors, so that the dual form of its LP formulation can be kernelized. But then its size is $O((pn)^2)$, which is an unacceptable blowup in size. Moreover, in their formulation the 1-norm of the dual variables (the coefficients of linear combination) is regularized, and so the resulting weight vector is not always sparse. Another approach to maximizing the margin with a sparse weight vector is to use the boosting technique such as the AdaBoost [31, 61, 62], although in the naive implementation we would again face with the same obstacle that the pair-sample constructed through the reduction is of size pn. Freund et al. proposed the RankBoost [30] that simulates the AdaBoost over the

pair-sample in time linear in the original sample size $(p + n)$. Later, Rudin and Schapire showed that under certain assumptions, the AdaBoost is equivalent to the RankBoost [65]. But the RankBoost has only a (weak) guarantee of hard margin and no theoretical justification is given when the sample is not linearly separable. Moribe et al. proposed the SoftRankBoost [56] based on the smooth boosting framework [25, 67, 35, 37, 9], so that it works well when the sample is not linearly separable. The SoftRankBoost runs in $O((p + n)N)$ time per iteration and is shown to have the same guarantee of soft margin as that of our algorithm. But the SoftRankBoost does not seem to solve a soft margin optimization problem in any sense.

We would like to mention that the notion of AUC consistency is recently proposed for a criterion of ranking algorithms. An algorithm is said to be AUC consistent if the algorithm outputs a ranking function that converges the Bayes optimal one, that is, the ranking function that maximizes the AUC with respect to the underlying probability distribution. Uematsu et al. [48] first propose this notion and investigate the relation between AUC maximization and a convex loss minimization, where the loss function is an upper bound on the ranking risk. In particular, they show that the algorithm for minimizing hinge loss is not AUC consistent. Kotolowski et al. [50] and Agarwal [3] show that the algorithms for minimizing exponential loss and logistic loss are both AUC consistent. These results would suggest that as for AUC maximization the AdaBoost (minimizing exponential loss) works well but the Ranking SVM (minimizing hinge loss) does not. However, in these results, the ranking functions are assumed to be chosen from the universal hypothesis class containing all functions (not only linear functions), and they do not discuss the performance of algorithms when the hypothesis class is restricted to, say, linear functions.

3.2 Reformulation of the hard margin 1-norm Ranking SVM

In this section, we reformulate the hard margin 1-norm Ranking SVM by restricting the dual variables $\boldsymbol{d} \in \mathcal{P}_{pn}$ to the product form $d_{ij} = d_i^+ d_j^-$, where $\boldsymbol{d}^+ \in \mathcal{P}_p$ and $\boldsymbol{d}^- \in \mathcal{P}_n$ are

new variables. Then, since

$$\begin{aligned}
&\sum_{i,j} d_{ij}(\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)/2 \\
=&\sum_{i,j} d_i^+ d_j^-(\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)/2 \\
=&\sum_i d_i^+ \left(\sum_j d_j^-\right)\boldsymbol{x}_i^+/2 - \sum_j d_j^- \left(\sum_i d_i^+\right)\boldsymbol{x}_j^-/2 \\
=&\sum_i d_i^+ \boldsymbol{x}_i^+/2 - \sum_j d_j^- \boldsymbol{x}_j^-/2,
\end{aligned}$$

we obtain from OP 12 the following simplified LP problem, called $\mathrm{OPT}_{\mathrm{hard}}$.

OP 15: $\mathrm{OPT}_{\mathrm{hard}}$ (dual)

$$\begin{aligned}
&\min_{\gamma, \boldsymbol{d}^+, \boldsymbol{d}^-} \gamma \\
&\text{sub.to} \\
&\sum_i d_i^+ \boldsymbol{x}_i^+/2 - \sum_j d_j^- \boldsymbol{x}_j^-/2 \le \gamma \mathbf{1}, \\
&\boldsymbol{d}^+ \in \mathcal{P}_p, \\
&\boldsymbol{d}^- \in \mathcal{P}_n.
\end{aligned}$$

It turns out that the primal form of the LP problem above is the standard classification version of hard margin 1-norm SVM over the original sample S, which now has the bias term:

OP 16: $\mathrm{OPT}_{\mathrm{hard}}$ (primal)

$$\begin{aligned}
&\max_{\rho, \boldsymbol{w}, b} \rho \\
&\text{sub.to} \\
&\boldsymbol{w} \cdot \boldsymbol{x}_i^+ + b \ge \rho, \quad i \in [1, p] \\
&\boldsymbol{w} \cdot \boldsymbol{x}_j^- + b \le -\rho, \quad j \in [1, n] \\
&\boldsymbol{w} \in \mathcal{P}_N.
\end{aligned}$$

Note that $\mathrm{OPT}_{\mathrm{hard}}$ is of size $O((p+n)N)$ in the both forms.

In the following, we show that $\mathrm{OPT}_{\mathrm{hard}}$ is equivalent to the original hard margin 1-norm Ranking SVM (OP 11), by showing that an optimal solution of OP 11 can be constructed from an optimal solution of OP 16.

Theorem 3

Let $(\rho_b, \boldsymbol{w}_b, b_b)$ *be an optimal solution of* OP 16. *Then,* $(\rho_b, \boldsymbol{w}_b)$ *is also an optimal solution of* OP 11.

Proof Let $(\rho_p, \boldsymbol{w}_p)$ be an optimal solution of OP 11. Clearly, $(\rho_b, \boldsymbol{w}_b)$ is a feasible solution of OP 11. So, $\rho_b \leq \rho_p$. Next, we show that the opposite is true. Let $\boldsymbol{x}^+$ and $\boldsymbol{x}^-$ be positive and negative *support vectors* of OP 11 for which $\boldsymbol{w}_p \cdot (\boldsymbol{x}^+ - \boldsymbol{x}^-)/2 = \rho_p$. Let

$$b_p = -\frac{\boldsymbol{w}_p \cdot (\boldsymbol{x}^+ + \boldsymbol{x}^-)}{2}.$$

Then, $(\rho_p, \boldsymbol{w}_p, b_p)$ is a feasible solution of OP 16. To see this, for any positive instance $\boldsymbol{x}_i^+$, observe that

$$\begin{aligned} \boldsymbol{w}_p \cdot \boldsymbol{x}_i^+ + b_p &= \frac{\boldsymbol{w}_p \cdot (\boldsymbol{x}_i^+ - \boldsymbol{x}^-)}{2} + \frac{\boldsymbol{w}_p \cdot (\boldsymbol{x}_i^+ - \boldsymbol{x}^+)}{2} \\ &\geq \rho_p + \frac{\boldsymbol{w}_p \cdot (\boldsymbol{x}_i^+ - \boldsymbol{x}^-) - \boldsymbol{w}_p \cdot (\boldsymbol{x}^+ - \boldsymbol{x}^-)}{2} \\ &\geq \rho_p + \rho_p - \rho_p = \rho_p. \end{aligned}$$

A similar inequality holds for negative instances as well. So we have $\rho_p \leq \rho_b$.

3.3 Reformulation of the soft margin 1-norm Ranking SVM

Motivated by the equivalence result of the hard margin case, we now reformulate the soft margin 1-norm Ranking SVM with the same additional constraints $d_{ij} = d_i^+ d_j^-$. Then we obtain from OP 14 the following simplified optimization problem:

OP 17: $\mathrm{OPT}_{\mathrm{soft}}$

$$
\begin{aligned}
\hat{\gamma} = \min_{\gamma, \boldsymbol{d}^+, \boldsymbol{d}^-, \nu^+} \ & \gamma \\
\text{sub.to} \\
& \sum_i d_i^+ \boldsymbol{x}_i^+ / 2 - \sum_j d_j^- \boldsymbol{x}_j^- / 2 \leq \gamma \mathbf{1}, \\
& d_i^+ \leq \frac{1}{\nu^+}, \quad i \in [1, p] \\
& d_j^- \leq \frac{\nu^+}{\nu}, \quad j \in [1, n] \\
& \boldsymbol{d}^+ \in \mathcal{P}_p, \\
& \boldsymbol{d}^- \in \mathcal{P}_n.
\end{aligned}
$$

Note that we replace the constraint $\max_{(i,j)\in I} d_{ij} \leq 1/\nu$ of OP 14 by $\max_i d_i^+ \max_j d_j^- \leq 1/\nu$, which is further replaced by the two constraints $\max_i d_i^+ \leq 1/\nu^+$ and $\max_j d_j^- \leq \nu^+/\nu$ with a new variable ν^+ to be optimized.

$\mathrm{OPT}_{\mathrm{soft}}$ is of size $O((p+n)(p+n+N))$ and thus seems to be easier to solve. But it is not a convex optimization problem since the constraints $d_i^+ \leq 1/\nu^+$ are not convex. To overcome this difficulty, we first consider $\mathrm{OPT}_{\mathrm{soft}}$ as an LP problem with ν^+ to be fixed to a constant. The LP problem with parameter ν^+ is called $\mathrm{OPT}_{\mathrm{soft}}(\nu^+)$:

OP 18: $\mathrm{OPT}_{\mathrm{soft}}(\nu^+)$ (dual)

$$
\begin{aligned}
\hat{\gamma}(\nu^+) = \min_{\gamma, \boldsymbol{d}^+, \boldsymbol{d}^-} \ & \gamma \\
& \text{sub.to the same constraints as in } \mathrm{OPT}_{\mathrm{soft}}
\end{aligned}
$$

Clearly, $\min_{\nu^+} \hat{\gamma}(\nu^+) = \hat{\gamma}$. Unfortunately, the function $\hat{\gamma}(\nu^+)$ is not convex with respect to ν^+ (see Fig. 3.1 for example). So, it seems to be hard to obtain the optimum. In the next section we propose an iterative linearization-minimization method to find a local optimal solution of ν^+.

On the other hand, for any fixed choice of ν^+, we can guarantee that the solution of $\mathrm{OPT}_{\mathrm{soft}}(\nu^+)$ has a certain amount of empirical AUC score. To see this, we first give the primal form of $\mathrm{OPT}_{\mathrm{soft}}(\nu^+)$:

OP 19: $\mathrm{OPT}_{\mathrm{soft}}(\nu^+)$ (primal)

$$
\begin{aligned}
&(\hat{\rho}, \hat{\boldsymbol{w}}, \hat{b}, \hat{\boldsymbol{\xi}}^+, \hat{\boldsymbol{\xi}}^-) \\
&= \arg \max_{\rho, \boldsymbol{w}, b, \boldsymbol{\xi}^+, \boldsymbol{\xi}^-} \rho - \frac{1}{2\nu^+} \sum_i \xi_i^+ - \frac{\nu^+}{2\nu} \sum_j \xi_j^- \\
&\text{sub.to} \\
&\boldsymbol{w} \cdot \boldsymbol{x}_i^+ + b \geq \rho - \xi_i^+, \quad i \in [1, p] \\
&-\boldsymbol{w} \cdot \boldsymbol{x}_j^- - b \geq \rho - \xi_j^-, \quad j \in [1, n] \\
&\boldsymbol{w} \in \mathcal{P}_N, \\
&\boldsymbol{\xi}^+, \boldsymbol{\xi}^- \geq \mathbf{0}.
\end{aligned}
$$

Theorem 4

For a fixed ν^+, let $\hat{\rho}$ and $\hat{\boldsymbol{w}}$ be the solutions of $\mathrm{OPT}_{\mathrm{soft}}(\nu^+)$ (primal)*. Then,*

$$\mathrm{AUC}_{S,\hat{\rho}}(\hat{\boldsymbol{w}}) \geq 1 - \frac{\nu^+}{p} - \frac{\nu}{n\nu^+} + \frac{\nu}{pn}.$$

Proof By using the KKT conditions, we have $\hat{\xi}_i^+(\hat{d}_i^+ - 1/\nu^+) = 0$. So, if $\hat{\xi}_i^+ > 0$ then $\hat{d}_i^+ = 1/\nu^+$. Since there are at most ν^+ indices i such that $\hat{d}_i^+ = 1/\nu^+$, there are at most ν^+ indices i with $\hat{\xi}_i^+ > 0$. Similarly, there are at most ν/ν^+ indices j with $\hat{\xi}_j^- > 0$. Therefore, for at least $(p - \nu^+)(n - \nu/\nu^+)$ pairs of instances $\boldsymbol{z}_{ij}$ in the pair-sample, $\hat{\boldsymbol{w}} \cdot \boldsymbol{z} \geq \hat{\rho}$.

For particular choices of the parameters ν and ν^+, we obtain the following corollary.

Corollary 5

Let $\hat{\rho}$ and $\hat{\boldsymbol{w}}$ be the optimal solution of $\mathrm{OPT}_{\mathrm{soft}}(\nu^+)$ (primal) *for $\nu = \varepsilon pn$ and $\nu^+ = \sqrt{\varepsilon}p$. Then,*

$$\mathrm{AUC}_{S,\hat{\rho}}(\hat{\boldsymbol{w}}) \geq (1 - \sqrt{\varepsilon})^2.$$

For comparison, we show the guarantee (2.9) of the original soft margin 1-norm Ranking SVM for the same choice of ν as in the corollary above:

$$\mathrm{AUC}_{S,\rho^*}(\boldsymbol{w}^*) \geq 1 - \varepsilon.$$

Below we compare ρ^* and $\hat{\rho}$. Note that, by duality we have

$$\hat{\rho} - \frac{1}{2\nu^+} \sum_i \hat{\xi}_i^+ - \frac{\nu^+}{2\nu} \sum_j \hat{\xi}_j^- = \hat{\gamma}(\nu^+).$$

Combined this with the fact that

$$\hat{\gamma}(\nu^+) \geq \hat{\gamma} \geq \gamma^* = \rho^* - (1/\nu)\sum_{ij}\xi^*_{ij},$$

we have

$$\hat{\rho} \geq \rho^* - \frac{1}{\nu}\sum_{ij}\xi^*_{ij}.$$

Therefore, if $\sum_{ij}\xi^*_{ij}$ is small, i.e., the sample is close to be linearly separable, then we can say that the solution $\hat{\boldsymbol{w}}$ of $\mathrm{OPT}_{\mathrm{soft}}(\nu^+)$ has nearly as high AUC score as the original 1-norm soft margin Ranking SVM.

3.4 An iterative linearization-minimization method for optimizing ν^+

Now we give an iterative linearization-minimization method for finding a local optimal solution of ν^+, which attains a local minimum of $\mathrm{OPT}_{\mathrm{soft}}$. Recall that in $\mathrm{OPT}_{\mathrm{soft}}$, we have non-convex constraints $d_i^+ \leq 1/\nu^+$. In order to make them convex, we replace them by their linear approximations. To be more precise, we consider the constraint that every d_i^+ is bounded by the tangent line of $1/\nu^+$ at some point $\nu^+ = \nu_c^+$. That is,

$$d_i^+ \leq -\frac{1}{(\nu_c^+)^2} + \frac{2}{\nu_c^+}. \tag{3.1}$$

Thus we have the following LP problem called $\mathrm{LP}_{\mathrm{soft}}$, where ν_c^+ is a parameter:

OP 20: $\mathrm{LP}_{\mathrm{soft}}$

$$\begin{aligned}(\tilde{\gamma}, \tilde{\boldsymbol{d}}^+, \tilde{\boldsymbol{d}}^-, \tilde{\nu}^+) = \arg \min_{\boldsymbol{d}^+, \boldsymbol{d}^-, \gamma, \nu^+} \ & \gamma \\ \text{sub.to} \ & \\ & \sum_i d_i^+ \boldsymbol{x}_i^+/2 - \sum_j d_j^- \boldsymbol{x}_j^-/2 \leq \gamma \mathbf{1}, \\ & d_i^+ \leq -\frac{1}{(\nu_c^+)^2}\nu^+ + \frac{2}{\nu_c^+}, \quad i \in [1,p] \\ & d_j^- \leq \frac{\nu^+}{\nu}, \quad j \in [1,n] \\ & \boldsymbol{d}^+ \in \mathcal{P}_p, \\ & \boldsymbol{d}^- \in \mathcal{P}_n.\end{aligned}$$

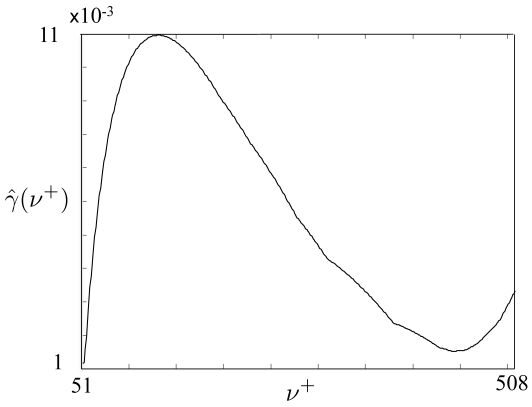


Figure 3.1: Non-convexity of the function $\hat{\gamma}(\nu^+)$ for an artificial data set.

Note that since any d_i^+ satisfying the new constraint (3.1) also satisfies the original constraints $d_i^+ \leq 1/\nu^+$, the optimal solution of $\mathrm{LP}_{\mathrm{soft}}$ is a feasible solution of $\mathrm{OPT}_{\mathrm{soft}}$.

Now we are ready to describe our algorithm:

1. Let ν_c^+ be an initial guess.
2. Solve $\mathrm{LP}_{\mathrm{soft}}$ and get a solution $(\tilde{\boldsymbol{d}}^+, \tilde{\boldsymbol{d}}^-, \tilde{\gamma}, \tilde{\nu}^+)$.
3. If the value of $\tilde{\gamma}$ decreases, then let $\nu_c^+ = \tilde{\nu}^+$ and go to 2.
4. Solve $\mathrm{OPT}_{\mathrm{soft}}(\nu^+)$(dual) with $\nu^+ = \tilde{\nu}^+$ and get a solution $(\hat{\boldsymbol{d}}^+, \hat{\boldsymbol{d}}^-, \hat{\gamma})$.

A reasonable choice of the initial guess in the first step would be $\nu_c^+ = \sqrt{\epsilon}p$ with $\nu = \epsilon pn$, as in Corollary 5. Observe that the solution $(\tilde{\boldsymbol{d}}^+, \tilde{\boldsymbol{d}}^-, \tilde{\gamma}, \tilde{\nu}^+)$ of $\mathrm{LP}_{\mathrm{soft}}$ is a feasible solution of $\mathrm{LP}_{\mathrm{soft}}(\tilde{\nu}^+)$. So, the minimum $\tilde{\gamma}'$ of $\mathrm{LP}_{\mathrm{soft}}(\tilde{\nu}^+)$ satisfies $\tilde{\gamma}' \leq \tilde{\gamma}$. Therefore, by repeating this procedure, we can obtain a monotonically decreasing sequence of $\tilde{\gamma}$, which will converge to a local minimum. Fig. 3.2 illustrates the algorithm. Note that the final step is redundant and thus can be skipped. We add this just for numerical stability.

3.5 Reformulation of 2-norm Ranking SVM

In this section, we employ a similar reformulation and simplification strategy to the standard 2-norm Ranking SVM, although as stated in Introduction, it has efficient algorithms under the original formulation. Here we no longer assume that the weight vector $\boldsymbol{w}$ is in $\mathcal{P}_N$.

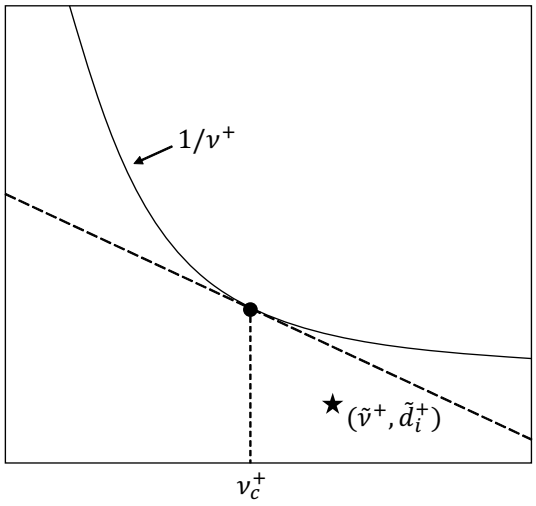


Figure 3.2: Illustration of our algorithm. The dotted line shows the tangent line of $1/\nu^+$ at $\nu^+ = \nu_c^+$. $\tilde{\nu}^+$ is the solution of $\mathrm{LP}_{\mathrm{soft}}$

3.5.1 Reformulation of the hard margin 2-norm Ranking SVM

We reformulate the hard margin 2-norm Ranking SVM by restricting the dual variables α_{ij} to $2\alpha_i^+\alpha_j^-/\sum_i^p \alpha_i^+$. If we add the restriction for OP 8, then we obtain the following QP problem which we call 2-norm $\mathrm{OPT}_{\mathrm{hard}}$.

OP 21: 2-norm $\mathrm{OPT}_{\mathrm{hard}}$ (dual)

$$\max_{\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-} \sum_{i=1}^{p} \alpha_i^+ + \sum_{j=1}^{n} \alpha_j^- - \frac{1}{2}\left\| \sum_{i=1}^{p} \alpha_i^+ \boldsymbol{x}_i^+ - \sum_{j=1}^{n} \alpha_j^- \boldsymbol{x}_j^- \right\|^2$$

sub.to

$$\sum_{i=1}^{p} \alpha_i^+ = \sum_{j=1}^{n} \alpha_j^-,$$

$$\alpha_i^+ \geq 0, \quad i \in [1,p]$$

$$\alpha_j^+ \geq 0, \quad j \in [1,n].$$

We can transform this QP problem into the primal form, which is the standard classification formulation of hard margin 2-norm SVM over the original sample S with bias term:

OP 22: 2-norm $\mathrm{OPT}_{\mathrm{hard}}$ (primal)

$$\begin{aligned}
&\max_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|_2^2 \\
&\text{sub.to} \\
&\boldsymbol{w}\cdot\boldsymbol{x}_i^+ + b \geq 1, \quad i \in [1,p] \\
&\boldsymbol{w}\cdot\boldsymbol{x}_j^- + b \leq -1, \quad j \in [1,n].
\end{aligned}$$

In hard margin 2-norm Ranking SVM, similar equivalence theorem holds between OP 7 and 2-norm OP 22.

Theorem 6

OP 7 and OP 22 are equivalent to each other. That is, each optimal solution can be constructed from the solution of the other one.

Proof We show the equivalence of their dual problems. Let $\theta_p(\boldsymbol{\alpha})$ and $\theta_b(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-)$ be the objective functions of the dual problems (8) and (21), respectively. Also, we denote $\widetilde{\boldsymbol{\alpha}}$ and $(\widehat{\boldsymbol{\alpha}}^+, \widehat{\boldsymbol{\alpha}}^-)$ as the optimal solution of the dual problems (8) and OP 21, respectively.

First, we construct a feasible solution of the problem (21) from the optimal solution of the problem (8). We define $(\widetilde{\boldsymbol{\alpha}}^+, \widetilde{\boldsymbol{\alpha}}^-)$ as $\widetilde{\alpha}_i^+ = \sum_{j=1}^n \widetilde{\alpha}_{ij}/2$ and $\widetilde{\alpha}_j^- = \sum_{i=1}^p \widetilde{\alpha}_{ij}/2$. Then, it holds that

$$\begin{aligned}
\theta_p(\widetilde{\boldsymbol{\alpha}}) &= \sum_{i=1}^p \sum_{j=1}^n \widetilde{\alpha}_{ij} - \frac{1}{2}\left(\sum_{i=1}^p \sum_{j=1}^n \widetilde{\alpha}_{ij}(\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)/2\right) \cdot \left(\sum_{i=1}^p \sum_{j=1}^n \widetilde{\alpha}_{ij}(\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)/2\right) \\
&= \sum_{i=1}^p \widetilde{\alpha}_i^+ + \sum_{j=1}^n \widetilde{\alpha}_j^- - \frac{1}{2}\left(\sum_{i=1}^p \widetilde{\alpha}_i^+ \boldsymbol{x}_i^+ - \sum_{j=1}^n \widetilde{\alpha}_j^- \boldsymbol{x}_j^-\right) \cdot \left(\sum_{i=1}^p \widetilde{\alpha}_i^+ \boldsymbol{x}_i^+ - \sum_{j=1}^n \widetilde{\alpha}_j^- \boldsymbol{x}_j^-\right) \\
&= \theta_b(\widetilde{\boldsymbol{\alpha}}^+, \widetilde{\boldsymbol{\alpha}}^-). \qquad (3.2)
\end{aligned}$$

Note that $\sum_{i=1}^p \widetilde{\alpha}_i^+ = \sum_{j=1}^n \widetilde{\alpha}_j^- = \sum_{i=1}^p \sum_{j=1}^n \alpha_{ij}$ and $\widetilde{\alpha}_i, \widetilde{\alpha}_j \geq 0$. So, $(\widetilde{\boldsymbol{\alpha}}^+, \widetilde{\boldsymbol{\alpha}}^-)$ is a feasible solution of the problem (21). This implies

$$\theta_b(\widetilde{\boldsymbol{\alpha}}^+, \widetilde{\boldsymbol{\alpha}}^-) \leq \theta_b(\widehat{\boldsymbol{\alpha}}^+, \widehat{\boldsymbol{\alpha}}^-). \qquad (3.3)$$

Next, we construct a feasible solution of OP 8 from the optimal solution of OP 21.

Let $\widehat{\boldsymbol{\alpha}}$ be such that $\widehat{\alpha}_{ij}/2 = \widehat{\alpha}_i^+ \frac{\widehat{\alpha}_j^-}{\sum_{j=1}^n \widehat{\alpha}_j^-}$. Then,

$$\begin{aligned}\theta_b(\widehat{\boldsymbol{\alpha}}^+, \widehat{\boldsymbol{\alpha}}^-) &= \sum_{i=1}^p \widehat{\alpha}_i^+ + \sum_{j=1}^n \widehat{\alpha}_j^- - \frac{1}{2}\left(\sum_{i=1}^p \widehat{\alpha}_i^+ \boldsymbol{x}_i^+ - \sum_{j=1}^n \widehat{\alpha}_j^- \boldsymbol{x}_j^-\right) \cdot \left(\sum_{i=1}^p \widehat{\alpha}_i^+ \boldsymbol{x}_i^+ - \sum_{j=1}^n \widehat{\alpha}_j^- \boldsymbol{x}_j^-\right) \\ &= \sum_{i=1}^p \sum_{j=1}^n \widehat{\alpha}_{ij} - \frac{1}{2}\left(\sum_{i=1}^p \sum_{j=1}^n \widehat{\alpha}_{ij}(\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)/2\right) \cdot \left(\sum_{i=1}^p \sum_{j=1}^n \widehat{\alpha}_{ij}(\boldsymbol{x}_i^+ - \boldsymbol{x}_j^-)/2\right) \\ &= \theta_p(\widehat{\boldsymbol{\alpha}}). \end{aligned} \tag{3.4}$$

Again, since $\widehat{\boldsymbol{\alpha}}$ is a feasible solution of OP 8,

$$\theta_p(\widehat{\boldsymbol{\alpha}}) \leq \theta_p(\widetilde{\boldsymbol{\alpha}}). \tag{3.5}$$

Combining (3.2), (3.3), (3.4), and (3.5), it follows that $\theta_p(\widetilde{\boldsymbol{\alpha}}) = \theta_p(\widehat{\boldsymbol{\alpha}}) = \theta_b(\widehat{\boldsymbol{\alpha}}^+, \widehat{\boldsymbol{\alpha}}^-) = \theta_b(\widetilde{\boldsymbol{\alpha}}^+, \widetilde{\boldsymbol{\alpha}}^-)$, which completes the proof.

3.5.2 Reformulation of the soft margin 1-norm Ranking SVM

Motivated by the equivalence result of the hard margin case, now we give our reformulation. Like the 1-norm case, we replace each dual variable α_{ij} with a (slightly modified) product form $4\alpha_i^+\alpha_j^-/\nu$ but now we put an additional constraint $\sum_i \alpha_i^+ = \sum_j \alpha_j^- (= \nu/2)$. Then, it is easy to see that we obtain from OP 10 the following simplified but non-convex optimization problem, called 2-norm OPT_{soft}:

OP 23: 2-norm OPT_{soft}

$$\begin{aligned} &(\widehat{\boldsymbol{\alpha}}^+, \widehat{\boldsymbol{\alpha}}^-, \widehat{\nu}^+) \\ &= \arg \max_{\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-, \nu^+} -\frac{1}{2}\left\|\sum_{i=1}^p \alpha_i^+ \boldsymbol{x}_i^+ - \sum_{j=1}^n \alpha_j^- \boldsymbol{x}_j^-\right\|^2 \\ &\text{sub.to} \\ &0 \leq \alpha_i^+ \leq \frac{\nu^+}{2p}, \quad i \in [1, p] \\ &0 \leq \alpha_j^- \leq \frac{\nu}{2n\nu^+}, \quad j \in [1, n] \\ &\sum_{i=1}^p \alpha_i^+ = \sum_{j=1}^n \alpha_j^- = \frac{\nu}{2}. \end{aligned}$$

Note that we replace the constraint $\max_{(i,j)\in I} \alpha_{ij} \leq 1/(pn)$ of OP 10 by $\max_i \alpha_i^+ \max_j \alpha_j^- \leq \nu/(4pn)$, which is further replaced by the two constraints $\max_i \alpha_i^+ \leq \nu^+/(2p)$ and $\max_j \alpha_j^- \leq \nu/(2n\nu^+)$ with the new variable ν^+ to be optimized.

When we fix ν^+ to a constant, then we have the QP problem, called 2-norm $\mathrm{OPT}_{\mathrm{soft}}(\nu^+)$, which has the following primal form:

OP 24: 2-norm $\mathrm{OPT}_{\mathrm{soft}}(\nu^+)$ (primal)

$$
\begin{aligned}
&(\hat{\rho}, \hat{\boldsymbol{w}}, \hat{b}, \hat{\boldsymbol{\xi}}^+, \hat{\boldsymbol{\xi}}^-) \\
&= \arg \min_{\rho, \boldsymbol{w}, b, \boldsymbol{\xi}^+, \boldsymbol{\xi}^-} \frac{||\boldsymbol{w}||^2}{2} - \nu\rho + \frac{\nu^+}{2p} \sum_{i=1}^{p} \xi_i^+ + \frac{\nu}{2n\nu^+} \sum_{j=1}^{n} \xi_j^- \\
&\qquad \text{sub.to} \\
&\qquad\qquad \boldsymbol{w} \cdot \boldsymbol{x}_i^+ + b \geq \rho - \xi_i, \quad i \in [1, p] \\
&\qquad\qquad -\boldsymbol{w} \cdot \boldsymbol{x}_j^- - b \geq \rho - \xi_j, \quad j \in [1, n] \\
&\qquad\qquad \boldsymbol{\xi}^+, \boldsymbol{\xi}^- \geq \mathbf{0}, \\
&\qquad\qquad \rho \geq 0.
\end{aligned}
$$

For any fixed choice of ν^+, we can guarantee that the solution of 2-norm $\mathrm{OPT}_{\mathrm{soft}}(\nu^+)$ has a certain amount of empirical AUC score.

Theorem 7

For a fixed ν^+, let $\hat{\rho}$ and $\hat{\boldsymbol{w}}$ be the solutions of 2-norm $\mathrm{OPT}_{\mathrm{soft}}(\nu^+)$ (primal). Then,

$$\mathrm{AUC}_{S,\hat{\rho}}(\hat{\boldsymbol{w}}) \geq 1 - \nu^+ - \frac{\nu}{\nu^+} + \nu.$$

Proof By the KKT conditions, $\xi_i^+ > 0$ implies $\alpha_i^+ = \nu^+/(2p)$. Since $\sum_i \alpha_i^+ = \nu/2$, there are at most $\nu p/\nu^+$ indices i such that $\xi_i > 0$. Similarly, there are at most $\nu^+ n$ indices j such that $\xi_j > 0$. Therefore, at least $(p - \nu p/\nu^+)(n - \nu^+ n)$ pairs of instances $\boldsymbol{z}_{ij}$ in the pair-sample, $\hat{\boldsymbol{w}} \cdot \boldsymbol{z} \geq \hat{\rho}$.

For a paticular choice of ν^+, we obtain the following corollary.

Corollary 8

Let $\hat{\rho}$ and $\hat{\boldsymbol{w}}$ be the optimal solutions of 2-norm $\mathrm{OPT}_{\mathrm{soft}}(\nu^+)$ (primal) for $\nu^+ = \sqrt{\nu}$. Then,

$$\mathrm{AUC}_{S,\hat{\rho}}(\hat{\boldsymbol{w}}) \geq (1 - \sqrt{\nu})^2.$$

Moreover, we can use the iterative linearization-minimization technique to find a local optimal value of ν^+, as we did for the 1-norm case. In this case, we replace the non-convex constraints

$$\alpha_j^- \leq \frac{\nu}{2n\nu^+}$$

of 2-norm $\mathrm{OPT}_{\mathrm{soft}}$ with the linear constraints

$$\alpha_j^- \leq -\frac{\nu}{2n(\nu_c^+)^2}\nu^+ + \frac{\nu}{n\nu_c^+},$$

where ν_c^+ is the current guess. Then, solve the QP problem and obtain $\hat{\nu}^+$. Repeat the procedure above with $\nu_c^+ = \hat{\nu}^+$ until convergence. Finally, solve 2-norm $\mathrm{OPT}_{\mathrm{soft}}(\nu^+)$ with $\nu^+ = \hat{\nu}^+$.

3.6 Experiments

In the following experiments, we verify effectiveness and efficiency of our method for maximizing AUCs. The data sets include artificial data sets, and some UCI data sets.

3.6.1 Artificial data

For the first experiment, we used artificial data sets with r-of-k threshold functions as target functions. An r-of-k threshold function f over N Boolean variables is associated with some set A of k Boolean variables and f outputs $+1$ if at least r of the k variables in A are positive and f outputs -1, otherwise. Assume that the instance space is $\{+1, -1\}^N$. That is, the r-of-k threshold function f is represented as

$$f(\boldsymbol{x}) = \mathrm{sign}\left(\sum_{x \in A} x + k - 2r + 1\right).$$

For $N = 100$, $k = 30$, and $r = 1, 8, 15$, we fix r-of-k threshold functions which determine labels. Then for each set of parameters, we generate $m = 1000$ random instances so that ratios of positive and negative instances are $5:5$, $7:3$, and $9:1$ respectively. Finally, we add random noise into labels by changing the label of each instance with probability 5%, 10%, and 15%. As hypotheses, we use N Boolean variables themselves and the constant hypothesis which always outputs $+1$.

We compare SVM-Perf [45], RankBoost [30], SoftRankBoost [56], naive 1-norm Ranking SVM (LP-Pair), and our methods for 1-norm and 2-norm. For SVM-Perf, we set the parameter $\epsilon = 0.001$, $C = 10, 20, \ldots, 100$. For RankBoost, we set the number of iterations as $T = 1000, 10000, 100000$. For the other methods, we set the parameter $\nu = \varepsilon pn$, where $\varepsilon \in \{0.05, 0.1, 0.15, 0.2\}$. We evaluate each method by 5-fold cross validation. Table 3.1 is the result that we change the noises, keep $p : n = 5 : 5$. Table 3.2 is the result that

Table 3.1: AUCs for artificial data sets with random noises 5%, 10%, and 15% so that ratios of positive and negative instances are 5:5.

data		SVM-Perf	Rank Boost	Soft RankBoost	LP-Pair	our method	
r	noise					1-norm	2-norm
1	5(%)	0.9363	0.9480	0.6991	0.9380	0.9445	**0.9566**
8		0.8967	0.9285	0.9441	0.9592	**0.9601**	0.9566
15		0.9325	0.9236	**0.9537**	0.9515	0.9526	0.9374
1	10(%)	0.8951	0.9030	0.6459	0.8909	0.9071	**0.9209**
8		0.8658	0.8928	**0.9227**	0.9186	0.9221	0.9169
15		0.8862	0.8786	**0.9158**	0.9002	0.9044	0.8914
1	15(%)	0.8480	0.8338	0.6343	0.6396	0.8343	**0.8650**
8		0.8337	0.8516	0.8566	0.8663	**0.8730**	0.8687
15		0.8436	0.8359	0.8598	0.8450	**0.8613**	0.8347

Table 3.2: AUCs for artificial data sets so that ratios of positive and negative instances are 7:3, 9:1.

data		SVM-Perf	Rank Boost	Soft RankBoost	LP-Pair	our method	
$p:n$	r					1-norm	2-norm
7:3	1	0.9411	0.9257	0.8440	0.9231	0.9046	**0.9458**
	8	0.9166	0.9078	0.9317	0.9333	**0.9336**	0.9177
	15	0.9179	0.9027	**0.9392**	0.9282	0.9333	0.9093
9:1	1	0.7950	0.8014	0.8095	0.8049	**0.8175**	0.7974
	8	0.7659	0.7734	**0.7773**	0.7678	0.7748	0.7645
	15	0.7269	0.7567	0.7573	0.7494	**0.7774**	0.7578

we change the ratios $p:n$, keep noises 5%. Surprisingly, our methods achieve high AUC scores comparing the other methods. We can observe that SoftRankBoost also often achieves high AUC scores, but our methods are better in stably.

3.6.2 Real data

For the next experiment we use data sets "hypothyroid", "ionosphere", "australian", "colon cancer" and "duke breast cancer" in LIBSVM data [18]. We set the parameter C

Table 3.3: AUCs for LIBSVM data sets

data	SVM-Perf	Rank Boost	Soft RankBoost	LP-Pair	our method	
					1-norm	2-norm
hypothyroid	**0.9374**	0.7883	0.8504	0.8504	0.9091	0.9318
ionosphere	0.9643	0.8961	0.9518	0.9778	**0.9790**	0.9568
australian	0.8882	0.9171	0.8896	0.9298	0.9257	**0.9325**
colon-cancer	0.8200	0.8500	**0.9350**	0.7900	0.8950	0.9000
duke	0.9520	0.6720	0.8720	**0.9600**	0.9520	0.9120

of SVM-Perf $100, 200, \ldots, 1000$. The parameters of the other algorithms are the same as in section 3.6.1. As can be seen in Table 3.3, It is not to say that our methods are clearly better than the other methods, but our methods stably archive high AUCs for all data sets.

3.6.3 Computation time

In this experiment, we will compare our method to LP-pair in 1-norm case and compare to SVM-Perf. The time complexity of SVM-Perf is guaranteed $O(sm\log(m))$, where s is the number of non-zero features. We use the machine with 6 cores of Intel Xeon 5560 2.80GHz and 32GByte memory, and use the artificial data sets, the size of each data set is $m = 250, 500, 1000, 1500, 3000$, respectively. The parameters of SVM-Perf are $\epsilon = 0.001$, $C = 100$ and 10000. For RankBoost, we set $T = 100$. We evaluate each execution time which is consumed to train for 5-fold cross validation and is averaged.

In 1-norm case, as is shown in Table 3.4, our method is clearly faster than LP-Pair. However, RankBoost and SoftRankBoost is much faster than our method. In 2-norm case, our method is faster than SVM-Perf set by $C = 10000$.

3.6.4 Sparsity

Finally, we show that the hyperplane obtained by 1-norm regularized methods has high sparsity using UCI data sets. As seen in Table 3.5, SoftRankBoost, LP-Pair and our method of 1-norm obtain sparse weight vectors for large feature data sets, and weight vectors of our method of 1-norm are as sparse as those of LP-Pair. Note that the each

Table 3.4: Computation time

m	SVM-Perf		Rank	Soft	LP-Pair	our method	
	$C = 100$	$C = 10000$	Boost	RankBoost		1-norm	2-norm
250	2.7	20.8	0.33	0.12	7.6	0.36	1.9
500	1.8	332.4	0.30	0.18	38.8	1.3	6.4
1000	1.9	527.1	0.22	0.13	153.8	3.0	27.1
1500	2.1	328.5	0.24	0.11	377.6	7.8	58.3
3000	3.9	683.6	0.41	0.16	1454.9	9.9	229.8

Table 3.5: The number of non-zero features of weight vectors obtained by each method using UCI data sets. N is the dimension each data set has.

data		SVM-Perf	Rank	Soft	LP-Pair	our method	
	N		Boost	RankBoost		1-norm	2-norm
hypothyroid	43	29	12	2	32	2	28
ionosphere	34	33	12	11	7	9	32
australian	14	8	6	6	9	8	14
colon-cancer	2000	1997	159	4	27	18	1991
duke	7129	7048	375	21	25	25	7066

norm of weight vector of SVM-Perf, RankBoost, and our method of 2-norm are normalized to 1.

3.7 Conclusion and future work

In this chapter, we have reformulated the Ranking SVMs for ranking functions as significantly simplified optimization problems of size $O(m^2)$, where m is the size of the original sample. We gave theoretical guarantees on the generalization ability of the ranking functions obtained by solving the optimization problems. In particular, the reformulation of the 1-norm Ranking SVM yields the first practical algorithm that is competitive with the original 1-norm Ranking SVM in performance.

As future work, we apply our practical method to optimizing other criteria biased to top elements [64].

Chapter 4

Learning Evaluation Functions for Shogi

4.1 Introduction

Game programming (for chess, shogi, Go) has been extensively investigated in artificial intelligence. Shogi (Japanese chess) is hard to solve by brute-force calculation, since it is said that the game tree from the start to final is very huge. Very recently, computer shogi has grown strong enough to defeat some professional shogi players. A key factor of this development is induced by *Bonanza Method* which is based on technique of machine learning. Since before the availability of Bonanza Method, machine learning of evaluation functions has been popular topic for game programming [34, 13, 77], but there had not been big result on computer shogi. Therefore the evaluation functions have had to been tuned by hand, of course it requires a lot of work and time. Furthermore, it causes a problem that the evaluation functions heavily depend on the knowledge and the sense of the designers. Bonanza Method allows us to tune the evaluation function automatically. More precisely, in Bonanza Method, the records of the professional players are used as a learning sample, and the optimization problem is formulated such that the evaluation function gives the maximum value to the professional players' moves within the legal moves in each position of the record. However, we still have an important problem that the designers do not know how to set the features of the evaluation function. That is, the evaluation functions still depend on the ability of the designers while computer shogi has grown strong. For example, it is considered that the positional relationships among multiple pieces or each piece's influence are important features to evaluate a position, and

such assorted features are made by the designers themselves. But it is unclear that the designers can cover all valuable features. Additionally, to set the combinations of some features increases the computation time to optimize the parameters.

In this chapter, we focus on this problem. For learning the evaluation functions, setting only basic and simple features, we apply polynomial kernel and Support Vector Machines (SVMs). d-polynomial kernel, which is one of kernel, allows us to cover all the features of position which can be expressed by at least d-monomial of the basic features. The computational complexity of kernels does not depend on d.

SVM is an efficient learning algorithm for a binary classification problem which is advantadgeous to combine kernels. To use SVM, we define positive examples as the positions appeared in professional records and negative examples as otherwise (they are consists of the positions after legal move at each position). Given such sample, an evaluation function f_{SVM} is obtained by SVM, then we can reasonably classify a position vector $\boldsymbol{x}$ into a positive position by the evaluation function if the value $f_{\mathrm{SVM}}(\boldsymbol{x}) \geq 0$. However, the function value absolutely does not estimate a goodness of a position. Thus, if the evaluation function values all of the (legal moved) positions set as negative, there is no foundation that we consider the position which has the largest value of them all is the best position. To obtain the evaluation function which estimates a goodness of a position, that is, we should rank the positions set. Therefore we see the learning evaluation functions problem as a bipartite ranking problem and we apply a ranking learning method.

As mentioned in Section 1.1, a naive implementation of classification reduction approach is impractical since the pair-sample constructed through the reduction is of size pn where the original sample consists of p positive and n negative instances. This is a quadratic blowup in size. In this thesis, by making pairs of only next positions set generated by the legal moves at a position, we can drastically reduce memory usage and computation time. In this thesis, we call this method RankSVM-Method.

Organization The rest of this chapter is organized as follows: In Section 4.2, we show a standard method with classification SVM to learn the evaluation function for shogi. In Section 4.3, we consider the learning of evaluation function as bipartite ranking problem. In Section 4.4, we examine our method using implementation of our computer shogi program. In Section 4.5, we conclude this chapter and mention future work.

4.2 Learning evaluation function using SVM

In this section, we show a method to learn the evaluation function for shogi using SVM.

4.2.1 Position vector

In this thesis, we represent position of shogi as a vector(we call *position vector*), and for which we use only basic and simple features (we call *low-level features*). We set the features of position vector as following:

(1) the number of each piece on the board,

(2) the number of each piece in hand,

(3) the kind of piece on each square,

(4) the kind and number of pieces influencing each square.

For example, at the initial array, the components of position vector include "the number of Black's pawn is nine on the board", "the number of Black's gold is two on the board", "a White's pawn is on 1c", and "a White's pawn influences 1d". All of the component of position vector values 0 or 1 if we construct the features redundantly as following: In case (1), for each piece

$$p \in P \equiv \{\text{Black's pawn}, \text{White's pawn}, \ldots, \\ \text{Black's promoted bishop}, \text{White's promoted bishop}\}$$

and for each natural number $i\,(0 \leq i \leq k_p)$, by converting "Can we find i of p on the board?" into binary variable. Thus, the component of the vector has $\sum_{p \in P}(k_p + 1)$ dimensions. For instance, if $p =$"Black's pawn", then "$k_p = 9$", if $p =$ "White's promoted pawn", then $k_p = 18$, that is, (1) is represented by 130 dimensions in total. Similarly, (2) is represented by 84 dimensions. Each of the features (3) mean "whether the kind p of piece exists or not on each square" (in the rules of playing), then the number of dimension of the features (3) is 2196. The features (4) can be expressed by 2358 dimensional features, each of which represents "the kind p and the number k_p of pieces influencing each square". Thus, totally the dimension number of position vector n is 4768.

4.2.2 Polynomial kernel

Most strong computer shogi programs use many high-level features which the designers consider useful but not only such low-level features. For example, the positional relationship between king and gold, and the number of influences around the king are considered as important features. In this chapter, we use polynomial kernel motivated by the fact that their complicated features can be expressed as d-degree polynomial of the low-level features. In particular, a polynomial kernel of degree d is defined by

$$K(\boldsymbol{a}, \boldsymbol{b}) = (\boldsymbol{a} \cdot \boldsymbol{b} + 1)^d.$$

Polynomial kernels map the input space to higher dimensional space $\mathbb{R}^N$ of dimension $N = \binom{n+d}{d}$, and the features associated to a polynomial of degree d are all the monomials of degree at most d based on the original features. Therefore this higher dimensional space consists of all the monomials of degree k of the low-level features. So, *high-level features* such as the positional relationship among multiple pieces and the number of influencing around the king are implicitly expressed as the monomials of degree k of low-level features. Furthermore, the computational complexity of polynomial kernel does not depend on a degree d. Thus, no matter how higher N dimensional space, the computational time and space do not increase.

4.2.3 Superiority of using polynomial kernel

The high-level features of the evaluation function of most recent computer shogi programs are technically or sensuously set by each designer. We introduce some features used in one of the most famous computer shogi program "Bonanza" [41] as an example:

(a) the number of each friendly piece,

(b) the positional relationship among king and a couple of other pieces,

(c) the positional relationship among the pairs of pieces neighboring each other,

(d) the kind of piece which is influenced by each of promoted rook, promoted bishop, rook, bishop, knight or lance, respectively,

(e) whether the number of square that each of promoted rook, promoted bishop, rook, bishop, lance can move, respectively,

(f) the kind of pinned piece, the direction of the pinning piece, and the distance from pinned piece to the king,

(g) the number of the pawns which occupies the same color square of friendly bishop,

(h) whether each of pawn, knight, silver on the board can move forward, respectively,

(i) the existence of the pawn which positions straight forward or backward of the positions of each promoted rook, rook, and lance,

(j) the influencing for each 25 square around the king.

All of these high-level features can be expressed by sum of products of low-level features that we showed in Subsection4.2.1. For instance, a feature "a gold positions next to a silver" as one of the features stated in (c) is given as follows: We identify each square on the board with an integer $l(1 \leq l \leq 81)$. For each piece p and each square l, let $x_{p,l}$ be a Boolean variable that indicates feature (3).

$$A = (x_{gold,1} \wedge x_{silver,2}) \vee \cdots \vee (x_{gold,81} \wedge x_{silver,80}).$$

Since each term is exclusive, A can also be expressed as the 2-degree polynomial

$$A = x_{gold,1}x_{silver,2} + \cdots + x_{gold,81}x_{silver,80}.$$

Similarity, the features (a)∼(j) can be expressed as d-degree polynomials of low-level features. The higher dimensional feature space contains these high-level features, thus, it contains all of the features used in "Bonanza". Moreover, polynomial kernel can express some more high-level features used in other strong computer shogi program such that "the kind of piece defensing against promoting of the opposing rook", "whether the pawn can be dropped forward in the opposing knight", and so on. Therefore, to procure the high-level features automatically using polynomial kernel is thought to be effective for the evaluation function of computer shogi.

4.2.4 Sampling professional records

We construct a learning sample S using some professional records following the modern strong computer shogi programs. First, for each position $\boldsymbol{z}$, we compose the next position set Next($\boldsymbol{z}$) obtained by enumerating all the legal moves. Second, for each $\boldsymbol{x} \in$ Next($\boldsymbol{z}$), if

a position $\boldsymbol{x}$ occurs in the professional records, we add to sample S as a positive instance $(\boldsymbol{x}, 1)$, otherwise we label as a negative instance $(\boldsymbol{x}, -1)$. We give this sample to a learning algorithm and get the evaluation function.

4.2.5 SVM-Method

We show a naive method using (classification) SVM, which we call SVM-Method. We denote by f_{SVM} a function obtained by giving the above sample to SVM. In game playing, we use f_{SVM} as an evaluation function. That is, for a position $\boldsymbol{x}$, $f_{\mathrm{SVM}}(\boldsymbol{x})$ is the evaluation value of $\boldsymbol{x}$. In this chapter, for a given position $\boldsymbol{z}$ we choose the next move which leads the next position $\boldsymbol{x}^* \in \mathrm{Next}(\boldsymbol{z})$ satisfying

$$f_{\mathrm{SVM}}(\boldsymbol{x}^*) = \max_{\boldsymbol{x} \in \mathrm{Next}(\boldsymbol{z})} f_{\mathrm{SVM}}(\boldsymbol{x}).$$

4.3 Learning evaluation function using Ranking SVM

In this chapter, we consider the bipartite ranking learning problem for learning the evaluation function, and we propose the method using Ranking SVM, which we call RankSVM-Method.

4.3.1 Sampling professional records

Ranking SVM solves the bipartite ranking problem by reducing to a binary classification problem over a pairwise instance space (see, 2.2.3). In general, a naive implementation of Ranking SVM is impractical since the pair-sample is of size pn, where the original sample consists of p positive and n negative instances. For example, if we construct a sample using 10000 records, the sample size blows up to more than four hundreds of millions.

However, in shogi, we do not need to rank all the positions but only need to rank the next positions set. Note that the number of optimal moves (professional moves) for a position is basically countable on one hand. Therefore, we construct a sample $\hat{S} \subset X \times X$ as following: For each position existing professional records, we construct $S_{\boldsymbol{z}}$ from the subset sample of S which consists of the next positions of $\boldsymbol{z}$,

$$S_{\boldsymbol{z}} = S \cap (\mathrm{Next}(\boldsymbol{z}) \times \{1, -1\}),$$

and we further construct the sample $\hat{S}$ which is the set of pair-sample $S^+_{\boldsymbol{z}} \times S^-_{\boldsymbol{z}}$,

$$\hat{S} = \bigcup_{\boldsymbol{z}} (S^+_{\boldsymbol{z}} \times S^-_{\boldsymbol{z}}).$$

We give the above sample to Ranking SVM and we obtain a ranking function, which we denoted by f_{RSVM}. When game playing, we use f_{RSVM} as the evaluation function. That is, for a position $\boldsymbol{x}$, $f_{\mathrm{RSVM}}(\boldsymbol{x})$ is the evaluation value of $\boldsymbol{x}$. In this chapter, for a given position $\boldsymbol{z}$, we choose the next move which leads the position $\boldsymbol{x}^* \in \mathrm{Next}(\boldsymbol{z})$ satisfying

$$f_{\mathrm{RSVM}}(\boldsymbol{x}^*) = \max_{\boldsymbol{x} \in \mathrm{Next}(\boldsymbol{z})} f_{\mathrm{RSVM}}(\boldsymbol{x}).$$

4.4 Experiments

In the following experiments, to examine the effectiveness of polynomial kernel for evaluation functions of shogi, we implement our methods and participate some tournament of computer shogi. Moreover, to compare SVM-Method and RankSVM-Method, we examine the agreement rate between professional players' moves and the moves ranked by our evaluation function.

4.4.1 Game playing

We construct a sample from one hundred professional records, and use SVM-light [43] to obtain an evaluation function f_{SVM} with 10-degree of polynomial kernel. We implement our computer shogi program "STR" using this evaluation function, and play games in some meets for computer shogi program. Note that we do not input any book and do not employ tree search but only checkmate search within seven moves. Figure 4.1 shows a position which is appeared when STR took on "ponanza" in the 20th Computer Shogi Championship. STR took a strategy of "Ibisha" with castle "Minogakoi" against Black's "Furibisha-Anaguma". Figure 4.2 shows one of the positions when playing with "Bonanza" in the 15th Computer Olympiad. STR took a strategy "Hachi-Go-Hisha" with castle "Nakahara-Gakoi" against Black's "Yoko-Fu-Dori". Both STR's strategies and both STR's castles are common forms in professional records, that is, we can say that STR plays games like humans. To take such strategies without book or tree search, many features such as the relationships among multiple pieces and opponent's strategy have to be considered. So, their STR's behaviors are thought to be results of the effectiveness of polynomial kernel.

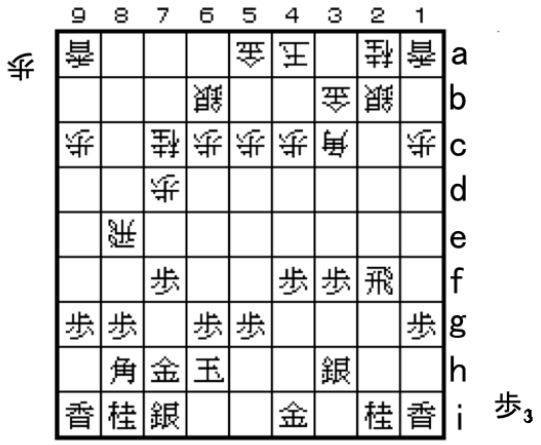


Figure 4.1: ponanza (Black) vs STR (White).

Figure 4.2: Bonanza (Black) vs STR (White).

4.4.2 Agreement rate with professional players

In the following, we compare SVM-Method and RankSVM-Method. We construct a sample from one thousand professional records, and use Pegasos [68] as an implementation of SVM to obtain f_{SVM} and f_{RSVM}[1]. Since Pegasos is based on random sampling, it learns from large sample faster than other general SVMs. For Pegasos, we set the normalization parameter $\lambda = 0.01, 0.05, 0.1$ and the parameter $\epsilon = 0.001, 0.01$, which adjusts the accuracy of the solutions. Then, we obtain six and six evaluation functions of f_{SVM} and f_{RSVM}. The number of iteration is set at $1/\lambda\epsilon$. We randomly choose one hundred records as the test data set. At each position of test data set without final position, we get rankings of the next positions set using f_{SVM} and f_{RSVM}. We do not employ tree search method, checkmate search, and do not input any book. If a ranking of a next position in the top r includes the position of the professional records, then we see the ranking agrees in the top r. We define the agreement rate with professional shogi players as (the number of agreements)/(the number of all positions in the test). As shown in Table 4.1, we can not observe much different for the agreement in the top 1, however, for the other r, the agreements rate of rankings output by the functions f_{RSVM} often higher than f_{SVM}.

Figure 4.3 shows that the cumulative frequency table for f_{SVM} and f_{RSVM}, the parameters of each are set at $\epsilon = 0.001$ and $\lambda = 0.01$. The agreements rate of f_{RSVM} get higher than f_{SVM} with r increasing. These results show that RankSVM-Method is more effective than SVM-Method for learning evaluation function of shogi.

4.5 Conclusion and future work

In this chapter, we showed effectiveness of polynomial kernel and bipartite ranking learning for a evaluation function of shogi. To evaluate a position more precisely, we may need to employ minimax-search. Therefore, as an idea for an improvement at the learning phase, we resample positions occurred in professional records to some leaf positions of minimax-tree rooted from them. Although minimax-search at this phase has to coincide with the calculation by professional players, there are no such data. So, we cannot seem to formulate the quadratic problem such as in Chapter 2 on learning an evaluation function

[1]When we employ RankSVM-Method, we take about one week to learn from one thousand professional records. Since we take much more time to learn several tens of thousands of professional records like the other strong programs, we use a sample of very smaller size in this thesis.

Table 4.1: Agreement rate with professional shogi players.

ϵ	λ	r	SVM	RSVM
0.001	0.01	1	0.056	0.058
		3	0.119	**0.146**
		5	0.163	**0.2014**
	0.05	1	0.057	**0.063**
		3	0.118	0.137
		5	0.163	0.194
	0.1	1	0.056	0.049
		3	0.119	0.119
		5	0.163	0.172
0.01	0.01	1	0.056	0.051
		3	0.118	0.116
		5	0.162	0.172
	0.05	1	0.056	0.038
		3	0.121	0.110
		5	0.164	0.156
	0.1	1	0.058	0.056
		3	0.120	0.131
		5	0.164	0.174

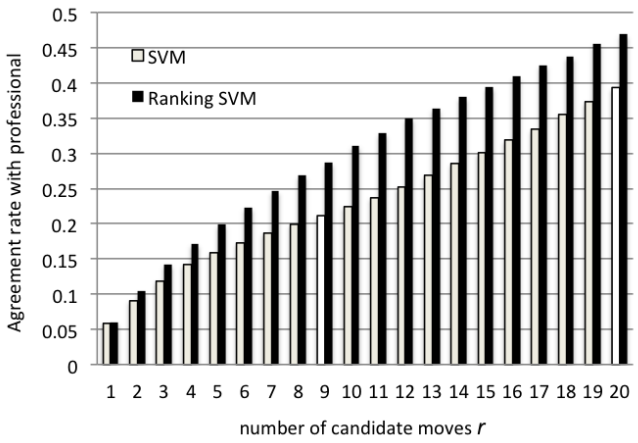


Figure 4.3: Agreement rate with professional shogi players.

combining minmax-search. Then, it seems to be effective that minimax-tree is decided based on the evaluation function at the phase. However, since the weight vector output by SVM increases with increasing the size of sample, the calculation of inner product for the evaluation function on a game needs much more time. These heuristics need a ingenuity such that we truncate the small weight vectors.

Chapter 5

Online Prediction under Submodular Constraints

5.1 Introduction

Online learning over combinatorial concept classes has gained much attention these days [39, 16, 49, 86]. Such combinatorial concept classes include shortest paths, k-sets, spanning trees, permutations, and so on. In typical settings, we assume a finite set $\mathcal{C}$ of combinatorial concept class where each concept can be represented as a vector in $\mathbb{R}^n$ for some fixed n, i.e., $\mathcal{C} \subseteq \mathbb{R}^n$. Then we consider the following protocol: For each trial $t = 1, \ldots, T$, (i) the player predicts $\boldsymbol{c}_t \in \mathcal{C}$, (ii) the adversary returns a loss vector $\boldsymbol{\ell}_t \in [0,1]^n$, and (iii) the player incurs loss $\boldsymbol{c}_t \cdot \boldsymbol{\ell}_t$. The goal of the player is to minimize the regret: $\sum_{t=1}^{T} \boldsymbol{c}_t \cdot \boldsymbol{\ell}_t - \min_{\boldsymbol{c} \in \mathcal{C}} \sum_{t=1}^{T} \boldsymbol{c} \cdot \boldsymbol{\ell}_t$.

There are some approaches to attack this type of problem. ranking or permutations prediction [1, 52, 80, 81, 39, 86], and there are many other online prediction problem over combinatorial or structured concept classes, shortest paths [75], k-sets [84], spanning trees [49, 16], and so on. A naive approach to minimize the regret in the above problem is to apply Hedge algorithm [31]. Hedge algorithm combines experts predictions, where each expert corresponds to each concept in $\mathcal{C}$. There are some efficient online prediction methods over combinatorial concept classes, for example, Follow the Perturbed Leader [47] PermELearn [39] and Component Hedge [49] and Comband in bandit setting [16]. These methods consist of external procedures.

Among the procedures, projection and decomposition are important and used in many online learning algorithms (see, e.g., [89, 39, 49]). Here, the projection is a procedure that,

when given a point, outputs its projection onto the convex hull of $\mathcal{C}$. The decomposition is a procedure that, when given a point in the convex hull, finds a convex combination representation of the point with some extreme points (i.e. concepts). So far, for particular combinatorial concept classes, we need to design projection and decomposition procedures individually. In this thesis, we investigate a unified and efficient projection and decomposition algorithms for a wide family of combinatorial concept classes. The concept class we consider coincides with the set of vertices (extreme points) of a polyhedron described by a submodular function f. In submodular function literature, the polyhedron is called (submodular) *base polyhedron* and denoted as $B(f)$ (we will give the definition later). That is, we consider the situation where $\mathcal{C}$ is the set of extreme points in $B(f)$. The base polyhedron $B(f)$ is defined using 2^n linear constraints and it is known that there are at most $n!$ vertices [32]. Examples of our problems include experts, k-sets [84], permutation [86], spanning trees [16, 49], k-truncated permutation, and k-forest. To the best of our knowledge, the last two problems are new for the online learning literature.

We propose projection and decomposition algorithms for the base polyhedron $B(f)$. The running times of the algorithms are both $O(n^6 + n^5 EO)$, where EO denotes the unit time to evaluate the submodular function. Furthermore, for cardinality-based submodular functions, we derive $O(n^2)$-time projection and decomposition algorithms. Such examples include k-sets and (k-truncated) permutation.

Our projection algorithms are designed for Euclidean distance and unnormalized relative entropy. So, we can combine them with Online Gradient Descent (OGD, [89]) or Hedge [31], respectively. Combined with our projection and decomposition algorithms for $B(f)$, their regret bounds become $O(D_{\mathrm{EUC}}\sqrt{nT})$ and $O(\sqrt{L^* f([n]) \ln n} + f([n]) \ln n)$, respectively, where $D_{\mathrm{EUC}} = \max_{\boldsymbol{c},\boldsymbol{c}' \in B(f)} \|\boldsymbol{c} - \boldsymbol{c}'\|_2$, and $L^* = \min_{\boldsymbol{c} \in B(f)} \sum_{t=1}^{T} \boldsymbol{c} \cdot \ell_t$.

Our contribution is to provide a unified view and efficient prediction strategies for an online prediction problem with exponentially many candidates by using rich theory of submodular function. Further, our $O(n^2)$-time algorithms for cardinality-based submodular functions are non-trivial for submodular optimization as well.

We discuss the relationship between previous and our results. First, we compare Follow the perturbed leader (FPL, [47]) with our algorithms. FPL uses an algorithm which solves "offline" linear optimization. It is well known that linear optimization over the base polyhedron is tractable and solved in $O(n \log n)$ time [28, 32]. So, the running time of FPL for our problem is $O(n \log n)$ at each trial. On the other hand, the regret of

FPL is $O(D_1\sqrt{nT})$, where $D_1 = \max_{\boldsymbol{c},\boldsymbol{c}'\in\mathcal{C}} \|\boldsymbol{c}-\boldsymbol{c}'\|_1$, which is worse than ours.

Next, we consider an algorithm proposed by [46] which converts an offline linear approximate optimization algorithm into the online one. This algorithm has an approximate projection procedure. But the running time of the projection procedure is $O(Tn\log n)$, which depends on T.

Component Hedge (CH, [49]) is also an efficient algorithm for predicting among exponentially many combinatorial concept classes. CH represents a combinatorial concept as a matrix and solves an entropy minimization problem with linear constraints at each trial. CH has more known applications such as directed spanning trees, paths and so on. However, the families of the concept classes with which CH can deal seem to be incomparable with ours. In an algorithmic sense, our algorithm has advantages for some concept classes. For example, for permutations and k-truncated permutations, it can be shown that CH requires $O(n^2)$ memory whereas ours uses $O(n)$ memory (see [86]) for related discussion.

Organization In Section 5.2 we refer to preliminary knowledge, submodular function (or base polyhedra) and their examples, and Bregman Divergence. In Section 5.3 we introduce Follow the Regularized Leader algorithm and its procedures, projection and decomposition. In Section 5.4 we propose efficient projection and decomposition algorithms when the underlying submodular function is cardinality-based. In Section 5.5 we conclude this Chapter.

5.2 Preliminaries

For any fixed positive integer n, we denote by $[n]$ the set $\{1,\dots,n\}$. A function $f : 2^{[n]} \to \mathbb{R}$ is *submodular* if for any $A, B \subset [n]$, $f(A\cup B)+f(A\cap B) \le f(A)+f(B)$. For simplicity, we assume that $f(\emptyset) = 0$. Given a submodular function f, *the base polyhedron* is defined as

$$B(f) = \left\{ \boldsymbol{x} \in \mathbb{R}^n \mid \sum_{i\in S} x_i \le f(S), \text{ for any } S \subset [n], \text{and } \sum_{i=1}^{n} x_i = f([n]) \right\}.$$

A point in $B(f)$ is an *extreme point* if it is not represented as a convex combination of other two points in $B(f)$. Let $\mathcal{C}$ be the set of extreme points in $B(f)$. In general, there can be exponentially many extreme points in $B(f)$. In this paper, for any submodular function f, we assume an oracle that returns the value $f(S)$ for any input S.

5.2.1 Examples

In this chapter, we consider a family $(\mathcal{S}, \phi)$ of concept classes such that for any seed $s \in \mathcal{S}$, the concept class $\phi(s)$ coincides with the set of extreme points of $B(f)$ for some submodular function $f : 2^{[n]} \to \mathbb{R}$. Moreover, we assume that the value $f(S)$ for any $S \subseteq [n]$ can be evaluated from the seed s in polynomial time. We show some examples of such families as below. In particular, the last two problems are new applications which are not previously studied.

Experts For a seed n, the concept class $\phi(s)$ coincides with the extreme points of $B(f)$ for the submodular function $f(S) = 1$.

k-sets For a seed $s = (n, k)$, the concept class $\phi(s)$ coincides with the extreme points of $B(f)$ for the submodular function $f : 2^{[n]} \to \mathbb{R}$ such that $f(S) = g(|S|)$, where $g(i) = i$, if $i \leq k$ and $g(i) = k$, if $i > k$ (see, e.g., [32]).

Spanning trees For a seed $G = (V, E)$, the concept class $\phi(G)$ coincides with the extreme points of $B(f)$ for the submodular function $f : 2^E \to \mathbb{R}$ such that $f(A) = |V(A)| - s(A)$, where $V(A)$ is the set of vertices of the subgraph induced by the set A of edges, and $s(A)$ is the number of the connected components of the subgraph [29, 20]. The base polyhedron $B(f)$ is called a spanning tree polyhedron.

Permutations For a seed n, the concept class $\phi(n)$ coincides with the extreme points of $B(f)$ for the submodular function $f : 2^{[n]} \to \mathbb{R}$ such that $f(S) = \sum_{i=1}^{|S|}(n + 1 - i)$. The base polyhedron $B(f)$ is called *Permutahedron* (see, e.g., [88, 32]). Figure 5.1 illustrates the fourth-order permutahedron.

k-truncated permutations We consider a generalized version of the online scheduling problem [86], where the flow time of the first k jobs are neglected. This problem can be seen as the online prediction problem for the following family $\mathcal{F} = (\mathcal{S}, \phi)$: The seed set $\mathcal{S}$ is the set of pairs (n, k) of natural numbers with $k < n$, and for a seed $s = (n, k)$, the concept class is defined by

$$\phi(s) = \{(i_1, \ldots, i_n) \mid \ (i_1, \ldots, i_n) \text{ is a permutation over the multiset } \{1, 2, \ldots, n-k-1, n-k, \ldots, n-k\}, \text{ in which } n-k \text{ appears } k+1 \text{ times}\}.$$

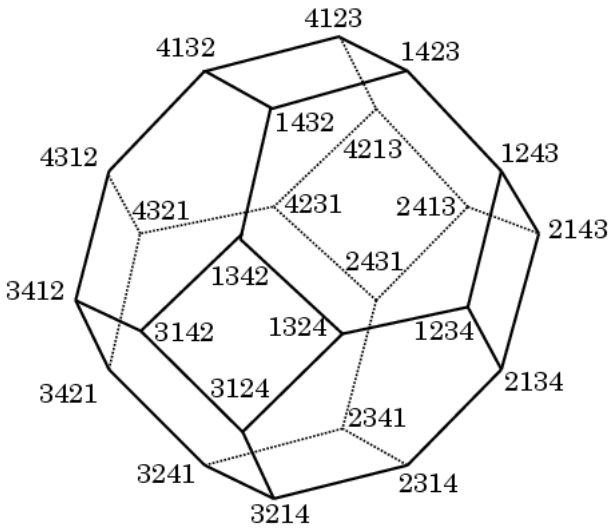


Figure 5.1: The fourth-order permutahedron.

For example, for the seed $s = (5, 2)$, $(3, 2, 3, 3, 1)$ is one of the concept in the class $\phi(s)$.

For a seed $s = (n, k)$, the concept class $\phi(s)$ coincides with the extreme points of $B(f)$ for the submodular function $f : 2^{[n]} \to \mathbb{R}$ such that $f(S) = (n-k)|S|$ if $|S| \leq k$ and $f(S) = (n-k)k + \sum_{j=k+1}^{|S|}(n+1-j)$, otherwise.

k-forests A k-forest of a graph $G = (V, E)$ is a subset $F \subseteq E$ of edges of size k such that F induces a forest (i.e., a subgraph with no cycles). It is known that the class of k-forests is a bases family of a truncation of a graphic matroid, which is known to be another matroid. Such classes can be described by the following family $\mathcal{F} = (\mathcal{S}, \phi)$: The seed set $\mathcal{S}$ is the set of pairs of a graph $G = (V, E)$ and a natural number $k(\leq |E|)$, and for a seed $s = (G, k)$, the concept class is defined by

$$\phi(s) = \left\{ \boldsymbol{c} \in \{0,1\}^E \mid \text{the edge set } \{e \in E \mid c_e = 1\} \text{ is a } k\text{-forest of } G \right\}.$$

The concept class $\phi(s)$ coincides with the extreme points of $B(f)$ for the submodular function $f : 2^E \to \mathbb{R}$ such that $f(S) = \min\{k, \max\{|F| \mid F \subseteq S \text{ is a forest}\}\}$.

5.2.2 Extreme points of the base polyhedron

In this subsection, we will see the correspondence between the permutations of $[n]$ and the extreme points of the base polyhedron $B(f)$.

Given a permutation $\boldsymbol{\sigma} = (i_1, \ldots, i_n)$ of $[n] = \{1, \ldots, n\}$, the greedy algorithm proposed by [28] generates a point $\boldsymbol{c}^{\boldsymbol{\sigma}} \in \mathbb{R}^n$ determined by

$$c^{\boldsymbol{\sigma}}_j = f(\{j' \in [n] : i_{j'} \leq i_j\}) - f(\{j' \in [n] : i_{j'} < i_j\}) \text{ for each } j \in [n].$$

Then $\boldsymbol{c}^{\boldsymbol{\sigma}}$ is an extreme point of $B(f)$. We will say that $\boldsymbol{c}^{\boldsymbol{\sigma}}$ is an extreme point of $B(f)$ generated by $\boldsymbol{\sigma}$. Conversely, for each extreme point $\boldsymbol{c}$ of $B(f)$, there is a permutation that generates $\boldsymbol{c}$. Figure 5.2 illustrates extreme points of base polyhedra.

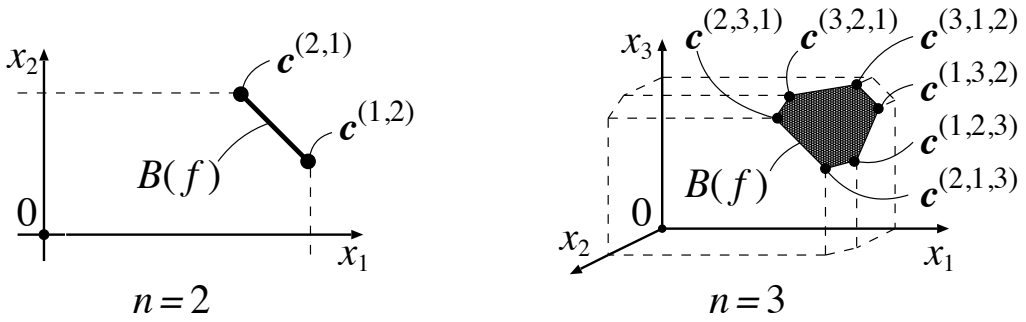


Figure 5.2: Extreme points of $B(f)$

5.3 Algorithm

In this section, we propose an algorithm that predicts extreme points of the base polyhedron $B(f)$ and prove its regret bounds.

5.3.1 Main structure

The main structure of the algorithm we use is shown in Algorithm 1 (see 2.3). Using FTRL itself is standard, but we need to design efficient procedures for projection and decomposition.

For particular combinatorial concept classes, we summarize their regret bounds in Table 5.1.

To complete our analysis, we specify the procedures Projection for separable strictly convex function Φ and Decomposition, respectively, in the following subsections. We will see that both of the two procedures are no harder than the submodular function minimization problem. For a submodular function $f : 2^{[n]} \to \mathbb{R}$ with $f(\emptyset) = 0$, the

problem	Hedge	OGD
Experts	$O(\sqrt{L^* \ln n})$	$O(\sqrt{T})$
k-sets	$O(\sqrt{L^* k \ln(n/k)} + k \ln(n/k))$	$O(k\sqrt{T})$
Spanning Trees	$O(\sqrt{L^* n \ln n} + n \ln n)$	$O(n\sqrt{T})$
Permutahedron	$O(n\sqrt{L^* \ln n} + n^2 \ln n)$	$O(n^2\sqrt{T})$
k-truncated Perm.	$O(\sqrt{L^*(n^2-k^2) \ln n} + (n^2-k^2) \ln n)$	$O((n-k)\sqrt{n(n+k)T})$
k-forest	$O(\sqrt{L^* k \ln(n/k)} + k \ln(n/k))$	$O(\sqrt{knT})$

Table 5.1: The regrets of combinatorial concept classes obtained using our projection and decomposition algorithms.

submodular function minimization (SFM) is a problem of finding a subset $S \subseteq [n]$ with $f(S)$ minimum. Many combinatorial SFM algorithms are known (see [42]), and the fastest known strongly polynomial algorithm proposed by [58] runs in $O(n^6 + n^5 EO)$ time, where EO is the unit time to evaluate the value of the submodular function. We will show that both of the procedures Projection and Decomposition can be implemented to run in $O(n^6 + n^5 EO)$ time.

5.3.2 Projection

For any given point $\boldsymbol{z} \in \mathbb{R}^n$, the procedure Projection in Algorithm 1 computes the projection of $\boldsymbol{z}$ onto the base polyhedron $B(f)$. We propose an efficient construction of this procedure. Formally, the projection problem is stated as follows:

OP 25: Projection over Base Polyhedra

$$\begin{aligned} \text{Projection}(\boldsymbol{z}) = \arg \inf_{\boldsymbol{x} \in B(f)} & \Delta_\Phi(\boldsymbol{x}, \boldsymbol{z}) \\ & \text{sub.to} \\ & \sum_{j \in S} x_j \leq f(S), \quad \forall S \subset [n], \\ & \sum_{j=1}^{n} x_j = f([n]), \end{aligned}$$

where $\Phi(\boldsymbol{x})$ is separable. This convex optimization problem with exponentially many constraints can be solved efficiently using the parametric submodular algorithm proposed by [57], which is a parametric extension of the SFM algorithm of [58].

Theorem 9 ([57])
There is an algorithm that solves OP 25 for separable strictly convex functions Φ in time $O(n^6 + n^5 EO)$.

5.3.3 Decomposition

For any given point $\boldsymbol{x}$ in the base polyhedron $B(f) \subseteq \mathbb{R}^n$, the procedure Decomposition in Algorithm 1 finds extreme points $\boldsymbol{c}^{\boldsymbol{\sigma}^1}, \ldots, \boldsymbol{c}^{\boldsymbol{\sigma}^K}$ in $B(f)$ and $\lambda_1, \ldots, \lambda_K \in \mathbb{R}_{>0}$ such that $\sum_{i=1}^K \lambda_i \boldsymbol{c}^{\boldsymbol{\sigma}^i} = \boldsymbol{x}$ and $\lambda_1 + \cdots + \lambda_K = 1$, where each $\boldsymbol{c}^{\boldsymbol{\sigma}^i}$ is an extreme point of $B(f)$ generated by a permutation $\boldsymbol{\sigma}^i$ of $[n]$ via the greedy algorithm proposed by [28]. In other words, this procedure represents $\boldsymbol{x}$ as a convex combination of extreme points of $B(f)$. Carathéodory's Theorem guarantees that $\boldsymbol{x} \in B(f)$ can be represented as a convex combination of at most n extreme points of $B(f)$.

To describe the procedure Decomposition, let us briefly review a common framework of algorithms for SFM. For a submodular function $f' : 2^{[n]} \to \mathbb{R}$ with $f'(\emptyset) = 0$, the result of [28] implies

$$\min_S \{f'(S) : S \subseteq [n]\} = \max_{\boldsymbol{z}} \left\{ \sum_{j=1}^n \min\{0, z_j\} : \boldsymbol{z} \in B(f') \right\}. \tag{5.1}$$

In many combinatorial SFM algorithms, including Orlin's algorithm [58], we finally obtain a minimizer $S^* \subseteq [n]$ and a maximizer $\boldsymbol{z}^* \in B(f')$ of (5.1). Moreover, we obtain $\boldsymbol{z}^* \in B(f')$ as a convex combination of at most n extreme points of $B(f')$. By the use of this fact, we can give an efficient construction of the procedure Decomposition.

For a given point $\boldsymbol{x} \in B(f)$, the function $f_{\boldsymbol{x}} : 2^{[n]} \to \mathbb{R}$ defined by $f_{\boldsymbol{x}}(S) = f(S) - \sum_{j \in S} x_j$ $(S \subseteq [n])$ is submodular and satisfies $f_{\boldsymbol{x}}(\emptyset) = 0$. For each permutation $\boldsymbol{\sigma}$ of $[n]$, let $\boldsymbol{c}^{\boldsymbol{\sigma}}$ be extreme points in $B(f)$ generated by $\boldsymbol{\sigma}$, and let $\boldsymbol{c}^{\boldsymbol{\sigma}}_{\boldsymbol{x}}$ be extreme points in $B(f_{\boldsymbol{x}})$ generated by $\boldsymbol{\sigma}$. Then it holds that $\boldsymbol{c}^{\boldsymbol{\sigma}}_{\boldsymbol{x}} = \boldsymbol{c}^{\boldsymbol{\sigma}} - \boldsymbol{x}$. In view of the definition of the base polyhedron, we have that $\min_{S \subseteq [n]} f_{\boldsymbol{x}}(S) = 0$ and the n-dimensional zero vector $\mathbf{0}_n$ is in $B(f_{\boldsymbol{x}})$. Therefore, $\boldsymbol{z} = \mathbf{0}_n$ is the unique optimal solution to the right hand side of (5.1) with $f' = f_{\boldsymbol{x}}$.

Now we describe the procedure Decomposition. Initially, we apply some combinatorial SFM algorithm, (e.g., Orlin's algorithm, [58]), to the submodular function $f_{\boldsymbol{x}}$. Then we obtain permutations $\boldsymbol{\sigma}^1, \ldots, \boldsymbol{\sigma}^K$ of $[n]$ and $\lambda_1, \ldots, \lambda_K \in \mathbb{R}_{>0}$ such that $\sum_{i=1}^K \lambda_i \boldsymbol{c}^{\boldsymbol{\sigma}^i}_{\boldsymbol{x}} = \mathbf{0}$, $\lambda_1 + \cdots + \lambda_K = 1$, and $K \leq n$. As for the function f, these permutations $\boldsymbol{\sigma}^1, \ldots, \boldsymbol{\sigma}^K$

and positive coefficients $\lambda_1, \ldots, \lambda_K$ generate another point $\sum_{i=1}^{K} \lambda_i \boldsymbol{c}^{\boldsymbol{\sigma}^i}$. For this point, we obtain

$$\sum_{i=1}^{K} \lambda_i \boldsymbol{c}^{\boldsymbol{\sigma}^i} = \sum_{i=1}^{K} \lambda_i (\boldsymbol{c}_{\boldsymbol{x}}^{\boldsymbol{\sigma}^i} + \boldsymbol{x}) = \sum_{i=1}^{K} \lambda_i \boldsymbol{c}_{\boldsymbol{x}}^{\boldsymbol{\sigma}^i} + \sum_{i=1}^{K} \lambda_i \boldsymbol{x} = \boldsymbol{x}.$$

Thus we have a required representation of $\boldsymbol{x}$. This gives the following.

Theorem 10

For any $\boldsymbol{x} \in B(f)$, there is an algorithm that gives a convex combination representation of $\boldsymbol{x}$ using at most n extreme points of $B(f)$ in $O(n^6 + n^5 EO)$ time.

5.4 Algorithm for cardinality-based submodular functions

In this section, we propose more efficient projection and decomposition algorithms when the underlying submodular function f is cardinality-based, i.e., $f(S) = g(|S|)$ for some $g : \mathbb{N} \to \mathbb{R}$. For projection, however, we only consider the Euclidean distance and the unnormalized relative entropy, rather than any Bregman divergence Δ_Φ for a separable function Φ as in the previous section.

A cardinality-based submodular function f has the following nice property: For any point $\boldsymbol{x} \in B(f)$ and any $i, j \in [n]$, the vector $\boldsymbol{x}'$ obtained by exchanging x_i and x_j in $\boldsymbol{x}$ is also contained in $B(f)$. A submodular function having this property is said to be *exchangeable*.

The following lemma says that for any exchangeable submodular function f, the projection onto $B(f)$ preserves the order of indices of vector with respect to the inequality relation.

Lemma 1

Let $\boldsymbol{x}^*$ be the projection of $\boldsymbol{z}$ in OP 25 under the Bregman divergence Δ_Φ for a strictly convex and uniformly separable function Φ. Assume that the submodular function f is exchangeable and $z_1 \geq \cdots \geq z_n$. Then, it holds that $x_1^* \geq x_2^* \geq \cdots \geq x_n^*$.

Proof Suppose on the contrary that $x_i^* < x_j^*$ for some $i < j$. Let $\widehat{\boldsymbol{x}}$ be the point obtained by exchanging x_i^* and x_j^* in $\boldsymbol{x}^*$. Then, by definition, we have $\widehat{\boldsymbol{x}} \in B(f)$. Furthermore,

observe that

$$\begin{aligned}\Delta_\Phi(\boldsymbol{x}^*, \boldsymbol{z}) - \Delta_\Phi(\widehat{\boldsymbol{x}}, \boldsymbol{z}) =& \Phi(\boldsymbol{x}^*) - \Phi(\widehat{\boldsymbol{x}}) - \nabla\Phi(\boldsymbol{z}) \cdot (\boldsymbol{x}^* - \widehat{\boldsymbol{x}}) \\ =& \phi(x_i^*) + \phi(x_j^*) - \phi(x_i^*) - \phi(x_j^*) \\ & - (\phi'(z_i)(x_i^* - x_j^*) - \phi'(z_j)(x_j^* - x_i^*)) \\ =& (x_j^* - x_i^*) \cdot (\phi'(z_i) - \phi'(z_j)) \geq 0,\end{aligned}$$

which contradicts the assumption that $\boldsymbol{x}^*$ is the projection. □

In the following, we assume that $z_1 \geq \cdots \geq z_n$ without loss of generality (this can be achieved by sorting). Lemma 1 implies that for any $S \subseteq [n]$, $\sum_{i \in S} x_i^* \leq \sum_{j=1}^{|S|} x_j^*$, which means that, if the right hand side is bounded by $f(S) = g(|S|)$, the left hand side is also bounded by $g(|S|)$. Therefore, OP 25 is equivalent to the following problem with only n constraints:

OP 26: Projection for Cardinality-based Submodular Functions

$$\begin{aligned}&\min_{\boldsymbol{x}} \Delta_\Phi(\boldsymbol{x}, \boldsymbol{z}) \\ &\text{sub.to} \\ &\sum_{i=1}^{j} x_i \leq g(j), \quad j = [1, n-1] \\ &\sum_{i=1}^{n} x_i = g(n).\end{aligned}$$

Now we propose an efficient implementation of Projection that solves OP 26.

5.4.1 Projection under Euclidean distance

First we give an algorithm which computes Projection under Euclidean distance. We show the algorithm in Algorithm 2. Then we prove the following.

Theorem 11

(i) Given $\boldsymbol{z}$, Algorithm 2 outputs the projection of $\boldsymbol{x}$ onto the base polyhedron $B(f)$. (ii) The time complexity of Algorithm 2 is $O(n^2)$.

Algorithm 2 Projection under Euclidean distance

Input: $\boldsymbol{z} \in \mathbb{R}^n$ satisfying that $z_1 \geq z_2 \geq \cdots \geq z_n$.

Output: projection $\boldsymbol{x}$ of $\boldsymbol{z}$ onto $B(f)$.

1. Let $i_0 = 0$.
2. **For** $t = 1, \ldots,$
   (a) Let $C^t(i) = \frac{g(i) - g(i_{t-1}) - \sum_{j=i_{t-1}+1}^{i} z_j}{i - i_{t-1}}$, for $i = 1, \ldots, n$
   and $i_t = \arg\min_{i: i_{t-1}+1 \leq i \leq n} C^t(i)$,
   if there are multiple minimizers, choose the largest one as i_t.
   (b) Set $x_i = z_i + C^t(i_t)$, for $i_{t-1} + 1 \leq i \leq i_t$.
   (c) **If** $i_t = n$, **then** break.
3. **Output** $\boldsymbol{x}$.

Proof By KKT condition (see, e.g., [8]), $\boldsymbol{x}^*$ is the solution of OP 26 if and only if there exists $\alpha_1, \ldots, \alpha_{n-1}$ and η such that

$$
\begin{aligned}
& x_i^* = z_i - \sum_{j=1}^{i} \alpha_j - \eta, \quad i = [1, n-1], \\
& x_n^* = z_n - \eta, \\
& \sum_{i=1}^{n} x_i^* = g(n), \\
& \alpha_i \left(\sum_{j=1}^{i} x_j^* - g(i) \right) = 0, \\
& \alpha_i \geq 0, \\
& \sum_{j=1}^{i} x_j^* \leq g(i), \quad i = [1, n-1].
\end{aligned} \tag{5.2}
$$

Now we show that there indeed exists $\alpha_1, \ldots, \alpha_{n-1}$ such that the output $\boldsymbol{x}$ of Projection($\boldsymbol{z}$) satisfies the optimality conditions (5.2), which suffices to prove the first statement. To do so, first we show that $C^{t-1}(i_{t-1}) \leq C^t(i_t)$ for each iteration t. By the definition of

$C^{t-1}(i_{t-1})$, we have $C^{t-1}(i_{t-1}) \leq C^{t-1}(i_t)$. Observe that

$$\begin{aligned} C^{t-1}(i_t) &= \frac{g(i_t) - g(i_{t-2}) - \sum_{j=i_{t-2}+1}^{i_t} z_j}{i_t - i_{t-2}} \\ &= \frac{g(i_t) - g(i_{t-1}) - \sum_{j=i_{t-1}+1}^{i_t} z_j + g(i_{t-1}) - g(i_{t-2}) - \sum_{j=i_{t-2}+1}^{i_{t-1}} z_j}{(i_t - i_{t-1}) + (i_{t-1} - i_{t-2})} \\ &= \frac{(i_t - i_{t-1})(C^t(i_t)) + (i_{t-1} - i_{t-2})(C^{t-1}(i_{t-1}))}{(i_t - i_{t-1}) + (i_{t-1} - i_{t-2})}. \end{aligned}$$

Since $C^{t-1}(i_{t-1}) \leq C^{t-1}(i_t)$,

$$\frac{(i_t - i_{t-1})(C^{t-1}(i_{t-1}))}{(i_t - i_{t-1}) + (i_{t-1} - i_{t-2})} \leq \frac{(i_t - i_{t-1})(C^t(i_t))}{(i_t - i_{t-1}) + (i_{t-1} - i_{t-2})}.$$

By simplifying this, we get $C^{t-1}(i_{t-1}) \leq C^t(i_t)$, as desired.

Then we determine each α_{i_t} so that $-\alpha_{i_t} + C^{t+1}(i_{t+1}) = C^t(i_t)$, i.e., $\alpha_{i_t} = C^{t+1}(i_{t+1}) - C^t(i_t)$ and fix η to be $C^T(n)$, where T satisfies $i_T = n$. Note that since $C^t(i_t) \leq C^{t+1}(i_{t+1})$, each α_{i_t} is strictly positive. For other $i \notin \{i_1, \ldots, i_T\}$, we set $\alpha_i = 0$. Then, each x_{i_t} can be expressed as

$$x_{i_t} = z_i + C^t(i_t) = z_i - (\alpha_{i_t} + \alpha_{i_{t+1}} + \cdots + \alpha_{i_T}) - \eta = z_i - (\alpha_{i_t} + \alpha_{i_t+1} + \cdots + \alpha_{i_{n-1}}) - \eta.$$

Similarity, for other i such that $i_{t-1} < i < i_t$, we have

$$x_i = z_i + C^t(i_t) = z_i - (\alpha_{i_t} + \alpha_{i_t+1} + \cdots + \alpha_{n-1}) - \eta = z_i - (\alpha_i + \alpha_{i+1} + \cdots + \alpha_{n-1}) - \eta.$$

To check if the specified α_is and η satisfies the optimality conditions (5.2), observe that (i) for each i_t,

$$\sum_{j=1}^{i_t} x_j = \sum_{j=1}^{i_{t-1}} x_j + \sum_{j=i_{t-1}+1}^{i_t} z_j + \sum_{j=i_{t-1}+1}^{i_t} C^t(i_t) = g(i_{t-1}) + (g(i_t) - g(i_{t-1})) = g(i_t)$$

and $\alpha_{i_t} > 0$, and (ii) for each i such that $i_{t-1} < i < i_t$,

$$\begin{aligned} \sum_{j=1}^{i} x_j = \sum_{j=1}^{i_{t-1}} x_j + \sum_{j=i_{t-1}+1}^{i} z_j + \sum_{j=i_{t-1}+1}^{i} C^t(i) &\leq \sum_{j=1}^{i_{t-1}} x_j + \sum_{j=i_{t-1}+1}^{i} (z_j + C^t(i)) \\ &= g(i_{t-1}) + (g(i) - g(i_{t-1})) = g(i) \end{aligned}$$

and $\alpha_i = 0$.

Finally, the algorithm terminates in time $O(n^2)$ since the number of iteration is at most n and each iteration takes $O(n)$ time, which proves the second statement of the lemma. □

5.4.2 Projection under unnormalized relative entropy

Next we propose an algorithm for Projection under unnormalized relative entropy. We construct the projection algorithm by generalizing the one used for permutahedron [86]. Note that the algorithm is also a generalization of the capping algorithm [84]. By KKT conditions, $\boldsymbol{x}^*$ satisfies the following conditions:

$$
\begin{aligned}
& x_i^* = z_i e^{-\sum_{j=1}^{n-1} \alpha_j}/\eta, \quad i = [1, n-1], \\
& x_n^* = z_n/\eta, \\
& \sum_{i=1}^{n} x_i^* = g(n), \\
& \alpha_i \left(\sum_{j=1}^{i} x_j^* - g(i) \right) = 0, \\
& \alpha_i \geq 0, \\
& \sum_{j=1}^{i} x_j^* \leq g(i), \quad i = [1, n-1].
\end{aligned} \tag{5.3}
$$

The algorithm shown in Algorithm 3 outputs the solution which satisfies the optimality

Algorithm 3 Projection under unnormalized relative entropy

Input: $\boldsymbol{z} \in \mathbb{R}^n$ satisfying that $z_1 \geq z_2 \geq \cdots \geq z_n$.

Output: projection $\boldsymbol{x}$ of $\boldsymbol{z}$ onto $B(f)$.

1. Let $i_0 = 0$.
2. **For** $t = 1, \ldots,$
   (a) Let $C^t(i) = \frac{g(i) - g(i_{t-1})}{\sum_{j=i_{t-1}+1}^{i} z_j}$, for $i = 1, \ldots, n$
   and $i_t = \arg\min_{i: i_{t-1}+1 \leq i \leq n} C^t(i)$,
   if there are multiple minimizers, choose the largest one as i_t.
   (b) Set $x_i = z_i C^t(i_t)$, for $i_{t-1} + 1 \leq i \leq i_t$.
   (c) **If** $i_t = n$, **then** break.
3. **Output** $\boldsymbol{x}$.

conditions, and following theorem holds.

Theorem 12

(i) Given $\boldsymbol{z}$, the Algorithm 3 outputs the projection of $\boldsymbol{x}$ onto the base polyhedron $B(f)$.

(ii) The time complexity of Algorithm 3 is $O(n^2)$.

The proof is also a generalization of the proof [86] and omitted due to the space constraints.

5.4.3 Decomposition

In this subsection, we describe how to represent a point $\boldsymbol{x} \in B(f)$ by a convex combination of extreme points of $B(f)$. More precisely, we are concerned with the following randomized rounding problem; given a point $\boldsymbol{x} \in B(f)$, output an extreme point X of $B(f)$ with a probability such that

$$\mathrm{E}[X] \stackrel{\mathrm{def}}{=} \sum_{j=1}^{k} \Pr\left[X = \boldsymbol{c}^j\right] \cdot \boldsymbol{c}^j = \boldsymbol{x}$$

for an appropriate $k > 0$.

As a preliminary step, we explain the following Propositions 13, 14, and 15, which are well-known facts (see e.g., [32]) Let $a \in \mathbb{R}_{>0}$ be a constant satisfying $a > g(n-1) - g(n)$, and we define $\tilde{f}\colon 2^{[n]} \to \mathbb{R}$ by $\tilde{f}(S) \stackrel{\mathrm{def}}{=} f(S) + a|S|$ for any $S \subseteq [n]$. Notice that $\tilde{f}$ is clearly a cardinality based function; let $\tilde{g}(z) \stackrel{\mathrm{def}}{=} g(z) + a \cdot z$ then $\tilde{f}(S) = \tilde{g}(|S|)$ holds.

Proposition 13

The function $\tilde{f}$ is cardinality based submodular*, as well as* monotone increasing*, i.e., $\tilde{g}(i) < \tilde{g}(i+1)$ for each $i \in [n-1]$.*

Note that $\tilde{f}(\emptyset) = 0$, and $\tilde{f}(S) > 0$ hold for any S ($\emptyset \subset S \subseteq [n]$).

Proposition 14

A point $\boldsymbol{x}$ is in $B(f)$ if and only if $\tilde{\boldsymbol{x}} \stackrel{\mathrm{def}}{=} \boldsymbol{x} + a\mathbf{1}$ is in $B(\tilde{f})$. A point $\boldsymbol{c}$ is an extreme point of $B(f)$ if and only if $\tilde{\boldsymbol{c}} \stackrel{\mathrm{def}}{=} \boldsymbol{c} + a\mathbf{1}$ is an extreme point of $B(\tilde{f})$.

Proposition 15

Suppose $\boldsymbol{x} \in B(f)$ satisfies $\boldsymbol{x} = \sum_{j=1}^{k} \lambda_j \boldsymbol{c}^j$ for $\lambda_j > 0$ ($j \in [k]$) satisfying $\sum_{j=1}^{k} \lambda_j = 1$ and $\boldsymbol{c}^j \in B(f)$ ($j \in [k]$). Then, $\tilde{\boldsymbol{x}} \stackrel{\mathrm{def}}{=} \boldsymbol{x} + a\mathbf{1} \in B(\tilde{f})$ satisfies $\tilde{\boldsymbol{x}} = \sum_{j=1}^{k} \lambda_j \tilde{\boldsymbol{c}}^j$ where $\tilde{\boldsymbol{c}}^j \stackrel{\mathrm{def}}{=} \boldsymbol{c}^j + a\mathbf{1} \in B(\widetilde{f})$.

Now, let $\tilde{f}\colon 2^{[n]} \to \mathbb{R}_{\geq 0}$ be a cardinality based submodular function which is *monotone increasing*, then we consider the randomized rounding problem; given a point $\tilde{\boldsymbol{x}} \in B(\tilde{f})$,

output an extreme point X of $B(\tilde{f})$ with a probability such that

$$\mathrm{E}[X] \stackrel{\mathrm{def}}{=} \sum_{j=1}^{k} \Pr\left[X = \tilde{\boldsymbol{c}}^j\right] \cdot \tilde{\boldsymbol{c}}^j = \tilde{\boldsymbol{x}}$$

for an appropriate $k > 0$. By Proposition 15, it is easily transformed into the case from a general cardinality based submodular function. Without loss of generality, we may assume that $\tilde{x}_1 \geq \cdots \geq \tilde{x}_n$ in the following. We remark that our randomized rounding algorithm generalizes the rounding algorithm for the permutahedron [86] in a sense.

To begin with, we define special points in $B(\tilde{f})$, which we call *partially averaged points*. Suppose $\tilde{\boldsymbol{q}} \in B(\tilde{f})$ satisfies that $\tilde{q}_1 \geq \tilde{q}_2 \geq \cdots \geq \tilde{q}_n$, then, $\tilde{\boldsymbol{q}}$ is a partially averaged point if $\sum_{j=1}^{i} \tilde{q}_j = \tilde{g}(i)$ holds for each $i \in [n]$ satisfying $\tilde{q}_i > \tilde{q}_{i+1}$. Notice that if $\tilde{q}_i > \tilde{q}_{i+1} = \cdots = \tilde{q}_j > \tilde{q}_{j+1}$ hold for $i, j \in [n]$ then $q_{i+1} = \cdots = q_j = (\tilde{g}(j) - \tilde{g}(i))/(j-i)$. This means that the partially averaged point is uniquely determined only by a sequence of equalities(=)/inequalities(>). We simply say "a partially averaged point of $\tilde{\boldsymbol{x}}$" ($\tilde{\boldsymbol{x}} \in B(\tilde{f})$) as a partially averaged point determined by a sequence of equalities/inequalities derived from $\tilde{x}_1 \geq \tilde{x}_2 \geq \cdots \geq \tilde{x}_n$ of $\tilde{\boldsymbol{x}}$.

Proposition 16
Suppose $\tilde{\boldsymbol{q}} \in B(\tilde{f})$ is a partially averaged point satisfying $\tilde{q}_1 \geq \tilde{q}_2 \geq \cdots \geq \tilde{q}_n$. Let $\Pi \stackrel{\mathrm{def}}{=} \{\boldsymbol{\sigma} \in \mathrm{Sym}(n) \mid \tilde{q}_{\sigma(1)} \geq \tilde{q}_{\sigma(2)} \geq \cdots \geq \tilde{q}_{\sigma(n)}\}$, where $\mathrm{Sym}(n)$ is the set of all permutations over $\{1, \ldots, n\}$. Let $\tilde{\boldsymbol{c}}^{\boldsymbol{\sigma}} = (\tilde{c}_1^{\boldsymbol{\sigma}}, \ldots, \tilde{c}_n^{\boldsymbol{\sigma}})$ for $\boldsymbol{\sigma} \in \Pi$ denote the extreme point defined by hyperplanes $\sum_{j=1}^{i} \tilde{c}_{\sigma(j)}^{\boldsymbol{\sigma}} = \tilde{g}(i)$ for all $i \in [n]$. Note that $\boldsymbol{\sigma} \neq \boldsymbol{\sigma}'$ does not imply $\tilde{\boldsymbol{c}}^{\boldsymbol{\sigma}} \neq \tilde{\boldsymbol{c}}^{\boldsymbol{\sigma}'}$ in general. Then, $\tilde{\boldsymbol{q}} = \frac{1}{|\Pi|} \sum_{\boldsymbol{\sigma} \in \Pi} \tilde{\boldsymbol{c}}^{\boldsymbol{\sigma}}$.

Proof Suppose $i \in [n-1]$ satisfies $\tilde{q}_i > \tilde{q}_{i+1}$. Since any $\boldsymbol{\sigma} \in \Pi$ satisfies $\tilde{q}_{\sigma(1)} \geq \tilde{q}_{\sigma(2)} \geq \cdots \geq \tilde{q}_{\sigma(n)}$, we see that $\{\sigma(1), \ldots, \sigma(i)\} = [i]$ holds for any $\boldsymbol{\sigma} \in \Pi$. This implies that $\sum_{j=1}^{i} \tilde{c}_j^{\boldsymbol{\sigma}} = \sum_{j=1}^{i} \tilde{c}_{\sigma(j)}^{\boldsymbol{\sigma}} = \tilde{g}(i)$. Since $\tilde{\boldsymbol{q}}$ is a partially averaged point, remember that $\sum_{j=1}^{i} \tilde{q}_j = \tilde{g}(i)$ holds, too.

Next, suppose $\tilde{q}_i > \tilde{q}_{i+1} = \cdots = \tilde{q}_j > \tilde{q}_{j+1}$ hold for $i, j \in [n]$. From the above arguments, we see that $\sum_{k=i+1}^{j} \tilde{c}_k^{\boldsymbol{\sigma}} = \tilde{g}(j) - \tilde{g}(i)$ holds for any $\boldsymbol{\sigma} \in \Pi$. For an arbitrary $\boldsymbol{\sigma} \in \Pi$, let $\boldsymbol{\sigma}' \in \mathrm{Sym}(n)$ satisfy $\sigma'(k) = \sigma(k)$ for each k ($k \leq i$ or $k > j$), then $\boldsymbol{\sigma}'$ is also in Π. Thus, let $\tilde{\boldsymbol{r}} \stackrel{\mathrm{def}}{=} \frac{1}{|\Pi|} \sum_{\boldsymbol{\sigma} \in \Pi} \tilde{\boldsymbol{c}}^{\boldsymbol{\sigma}}$ for convenience, then we see that $\tilde{r}_{i+1} = \cdots = \tilde{r}_j = (\tilde{g}(j) - \tilde{g}(i))/(j-i)$ hold. Since $\tilde{\boldsymbol{q}}$ is a partially averaged point, remember that $\tilde{q}_{i+1} = \cdots = \tilde{q}_j = (\tilde{g}(j) - \tilde{g}(i))/(j-i)$ hold, too. □

Algorithm 4 Decomposition by partially average points

Input: $\tilde{\boldsymbol{x}} \in B(\tilde{f})$ satisfying that $\tilde{x}_1 \geq \tilde{x}_2 \geq \cdots \geq \tilde{x}_n$.

Output: Partially average points $\tilde{\boldsymbol{q}}^1, \ldots, \tilde{\boldsymbol{q}}^K$ and $\lambda_1, \ldots, \lambda_K \in \mathbb{R}_{>0}$ s.t. $\sum_{i=1}^K \lambda_i \tilde{\boldsymbol{q}}^i = \tilde{\boldsymbol{x}}$, $\sum_{i=1}^K \lambda_i = 1$.

1. Let $\tilde{\boldsymbol{x}}^1 = \tilde{\boldsymbol{x}}$ and $\lambda = 1$.
2. **For** $t = 1, \ldots,$
   (a) Find a partially averaged point $\tilde{\boldsymbol{q}}^t$ for $\tilde{\boldsymbol{x}}^t$.
   (b) **if** $\tilde{q}_i^t = \tilde{q}_{i+1}^t$ for any i, **then** $\lambda_t = \lambda$
   else $\lambda_t = \min_{i \in [n-1]} \left\{ \frac{\tilde{x}_i^t - \tilde{x}_{i+1}^t}{\tilde{q}_i^t - \tilde{q}_{i+1}^t} \mid \tilde{q}_i^t \neq \tilde{q}_{i+1}^t \right\}$.
   (c) Let $\tilde{\boldsymbol{x}}^{t+1} = \tilde{\boldsymbol{x}}^t - \lambda_t \tilde{\boldsymbol{q}}^t$ and let $\lambda = \lambda - \lambda_t$.
   (d) **If** $\lambda = 0$ **then** let $K = t$ and break.
3. **Output** $\tilde{\boldsymbol{q}}^1, \ldots \tilde{\boldsymbol{q}}^K$ and $\lambda_1, \ldots, \lambda_K$.

Proposition 16 and its proof immediately suggest an algorithm for randomized rounding of a partially averaged point; generate $\boldsymbol{\sigma} \in \Pi$ uniformly at random, and output $\tilde{\boldsymbol{c}}^{\boldsymbol{\sigma}}$. It's running time is $O(n)$, clearly.

Now, we explain our Algorithm 4, which provides a convex combination of partially average points representing $\tilde{\boldsymbol{x}} \in B(\tilde{f})$, i.e., given $\tilde{\boldsymbol{x}} \in B(\tilde{f})$, find partially average points $\tilde{\boldsymbol{q}}^1, \ldots, \tilde{\boldsymbol{q}}^K$ and $\lambda_1, \ldots, \lambda_K \in \mathbb{R}_{>0}$ such that $\sum_{i=1}^K \lambda_i \tilde{\boldsymbol{q}}^i = \tilde{\boldsymbol{x}}$ and $\sum_{i=1}^K \lambda_i = 1$. Once we obtain such a convex combination, it is clear to obtain an algorithm for randomized rounding into partially average points. Combining the above arguments concerning Proposition 16, we obtain a desired algorithm. We will prove the following lemma on Algorithm 4.

Theorem 17

Algorithm 4 provides a convex combination of at most n partially averaged points representing an arbitrarily given $\tilde{\boldsymbol{x}} \in B(\tilde{f})$. Its running time is $O(n^2)$.

To show Theorem 17, we show the following lemmas.

Lemma 2

At any iteration t in Algorithm 4, $\tilde{\boldsymbol{x}}^t$ satisfies that $\tilde{x}_i^t \geq \tilde{x}_{i+1}^t$ for any $i \in [n-1]$.

Proof We give an inductive proof with respect to t. In case of $t = 1$, it is clear. In case of $t > 1$, we assume $\tilde{x}_i^{t-1} \geq \tilde{x}_{i+1}^{t-1}$ holds for any $i \in [n-1]$. If $\tilde{x}_i^{t-1} = \tilde{x}_{i+1}^{t-1}$, then $\tilde{q}_i^{t-1} = \tilde{q}_{i+1}^{t-1}$

holds, from the definition of $\tilde{\boldsymbol{q}}^{t-1}$. Thus

$$\tilde{x}_i^t = \tilde{x}_i^{t-1} - \lambda_{t-1}\tilde{q}_i^{t-1} = \tilde{x}_{i+1}^{t-1} - \lambda_{t-1}\tilde{q}_{i+1}^{t-1} = \tilde{x}_{i+1}^t$$

and we obtain the claim. If $\tilde{x}_i^{t-1} > \tilde{x}_{i+1}^{t-1}$, then $\tilde{q}_i^{t-1} > \tilde{q}_{i+1}^{t-1}$ holds, and

$$\tilde{x}_i^{t+1} - \tilde{x}_{i+1}^{t+1} \;=\; \tilde{x}_i^t - \tilde{x}_{i+1}^t - \lambda_t(\tilde{q}_i^t - \tilde{q}_{i+1}^t) = (\tilde{q}_i^t - \tilde{q}_{i+1}^t)\left(\frac{\tilde{x}_i^t - \tilde{x}_{i+1}^t}{\tilde{q}_i^t - \tilde{q}_{i+1}^t} - \lambda_t\right) \geq 0,$$

where the last inequality comes from the definition of λ_t, followed by

$$\lambda_t \leq \min_{i\in[n-1]} \left\{ \left(\tilde{x}_{i+1}^t - \tilde{x}_i^t\right) / \left(\tilde{q}_{i+1}^t - \tilde{q}_i^t\right) \mid \tilde{q}_{i+1}^t \neq \tilde{q}_i^t \right\}.$$

□

Lemma 3

In Algorithm 4, $\tilde{\boldsymbol{x}}^{K+1}$ $(= \tilde{\boldsymbol{x}}^K - \lambda^K \tilde{\boldsymbol{q}}^K) = 0$ holds.

Proof Without loss of generality, we may assume that $\tilde{x}_1 \geq \tilde{x}_2 \geq \cdots \geq \tilde{x}_n$, for simplicity of notations. First we show $\tilde{\boldsymbol{x}}^{K+1} \geq 0$. Since Lemma 2, if there exists $j \in [n]$ satisfying that $\tilde{x}_j^{K+1} < 0$, then $\tilde{x}_n^{K+1} < 0$ holds. Thus it is enough to show $\tilde{x}_n^{K+1} \geq 0$. Let $i^* = \min\{j \in [n] \mid \tilde{x}_j^K = \tilde{x}_n^K\}$. Then we have $\tilde{x}_{i^*}^K = \tilde{x}_{i^*+1}^K = \cdots = \tilde{x}_n^K$ and $\tilde{q}_{i^*}^K = \tilde{q}_{i^*+1}^K = \cdots = \tilde{q}_n^K$. Hence, we get $\tilde{x}_{i^*}^{K+1} = \tilde{x}_{i^*+1}^{K+1} = \cdots = \tilde{x}_n^{K+1}$. In case of $i^* \geq 2$, $\tilde{x}_{i^*-1}^t > \tilde{x}_{i^*}^t$ holds for any $t \in [K]$, meaning that $\tilde{q}_{i^*-1}^t > \tilde{q}_{i^*}^t$ holds for any $t \in [K]$. Thus we can see that $\sum_{j=i^*}^n \tilde{q}_j^t = \tilde{g}(n) - \tilde{g}(i^*-1)$ holds for any $t \in [K]$, from the definition of $\tilde{\boldsymbol{q}}^t$. Then we obtain

$$\sum_{j=i^*}^{n}\sum_{t=1}^{K} \lambda_t \tilde{q}_j^t = \sum_{t=1}^{K} \lambda_t \left(\tilde{g}(n) - \tilde{g}(i^*-1)\right) = \tilde{g}(n) - \tilde{g}(i^*-1) \leq \sum_{j=i^*}^{n} \tilde{x}_j$$

where the last inequality is due to constraints of $B(\tilde{f})$, $\sum_{j=1}^{i^*-1} \tilde{x}_j \leq \tilde{g}(i^*-1)$ and $\sum_{j=1}^n \tilde{x}_j = \tilde{g}(n)$. Thus we obtain that $\sum_{j=i^*}^n \tilde{x}_j^{K+1} = \sum_{j=i^*}^n \left(\tilde{x}_j - \sum_{t=1}^K \lambda_t \tilde{q}_j^t\right) \geq 0$. As discussed above, $\tilde{x}_{i^*}^{K+1} = \tilde{x}_{i^*+1}^{K+1} = \cdots = \tilde{x}_n^{K+1}$ holds, and we obtain $\tilde{x}_n^{K+1} \geq 0$. In case of $i^* = 1$, the proof is done in a similar way.

Now we show $\tilde{\boldsymbol{x}}^{K+1} = 0$. Since $\tilde{\boldsymbol{x}} \in B(\tilde{f})$, $\sum_{j=1}^n \tilde{x}_j^{K+1} = \tilde{g}(n)$ holds. In a similar way as the proof of $\tilde{\boldsymbol{x}}^{K+1} \geq 0$,

$$\sum_{j=1}^{n}\sum_{t=1}^{K} \lambda_t \tilde{q}_j^t = \sum_{t=1}^{K} \lambda_t \sum_{j=1}^{n} \tilde{q}_j^t = \sum_{t=1}^{K} \lambda_t \tilde{g}(n) = \tilde{g}(n).$$

Since $\boldsymbol{x}^{K+1} \geq 0$, $\tilde{\boldsymbol{x}}^{K+1} = \tilde{\boldsymbol{x}} - \sum_{t=1}^K \lambda_t \tilde{\boldsymbol{q}}^t = 0$. □

Lemma 4

The number of iterations K is at most n.

Proof From the definition of λ_t, there is at least one $i \in [n]$ satisfying that $\tilde{x}_i^t > \tilde{x}_{i+1}^t$ and $\tilde{x}_i^{t+1} = \tilde{x}_{i+1}^{t+1}$. If $\tilde{x}_i^t = \tilde{x}_{i+1}^t$, then $\tilde{x}_i^{t+1} = \tilde{x}_{i+1}^{t+1}$ as discussed in the proof of Lemma 2. Now the claim is clear. □

Proof of Theorem 17. Since Lemma 3, it is clear that the output $\sum_{t=0}^{K} \lambda_t \tilde{\boldsymbol{q}}^t$ by Algorithm 4 is equal to an arbitrarily given $\tilde{\boldsymbol{x}} \in B(\tilde{f})$. It is not difficult to see that every lines in Algorithm 4 is done in $O(n)$. Hence, the running time of Algorithm 4 is $O(n^2)$ by Lemma 4. □

Note that, by modifying Algorithm 4, we can design an algorithm for randomized rounding of $\boldsymbol{x} \in B(f)$ using only $O(n)$ space, with the same time complexity of $O(n^2)$. We can also improve the algorithm with a time complexity of $O(n \log n)$ using a heap, with $O(n)$ space.

5.5 Conclusion

In this chapter, we consider a prediction problem over the base polyhedron defined by a submodular function and propose efficient prediction algorithms. An open problem is to design a projection algorithm for cardinality based submodular functions under all of Bregman divergence but not only Euclidean distance and unnormalized relative entropy.

Chapter 6

Conclusions

In this thesis, we considered learning rankings and other combinatorial concept classes.

In Chapter 3, we considered bipartite ranking problem and proposed efficient reformulation for 1-norm Ranking SVM. While the LP problem for the 1-norm Ranking SVM is naturally of size $O((pn)^2)$ we achieved to reformulate the LP problem of size $O((p+n)^2)$ where p and n is number of positive and negative instances, respectively. We theoretically guaranteed the bound of AUC score and we show that it is practical in experiments using artificial data sets and real data sets. Furthermore, we extend our technique to 2-norm Ranking SVM.

In Chapter 4, we proposed some learning schemes for evaluation functions for shogi as an application of Ranking SVM. Conventionally, the features of position are made by hand, however, using kernel method combining with Ranking SVM, we could generate such features automatically. Additionally, we could solve efficiently bipartite ranking learning problem by sampling on each position. In experiments, we show the effectiveness of kernel method and Ranking SVM for evaluation function of shogi.

In Chapter 5, we considered online prediction problem for combinatorial concept classes including rankings. We proposed first general method for uniformly solving the concept classes which are represented by submodular function, and we gave regret bound. Moreover, we proposed more efficient method for in case the submodular functions are cardinality-based.

Bibliography

[1] D. L. Adolphson. Single machine job sequencing with precedence constraints. *SIAM Journal on Computing*, 6(1):40–54, 1977.

[2] A. Agarwal and S. Chakrabarti. Learning random walks to rank nodes in graphs. In *Proceedings of the 24th International Conference on Machine Learning*, pages 9–16, New York, NY, USA, 2007. ACM.

[3] S. Agarwal. Surrogate regret bounds for bipartite ranking via strongly proper losses. *CoRR*, abs/1207.0268, 2012.

[4] R. Agrawal, D. Teneketzis, V. Anantharam, N. V. Anantharam, and M. R. Asymptotically efficient adaptive allocation schemes for controlled i.i.d. processes: Finite parameter space. *IEEE Trans. Automat. Contr*, 34:258–267, 1988.

[5] M. R. Amini, T. V. Truong, and C. Goutte. A boosting algorithm for learning bipartite ranking functions with partially labeled data. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 99–106, New York, NY, USA, 2008. ACM.

[6] V. Anantharam, P. Varaiya, and J. Walrand. Asymptotically efficient allocation rules for the multiarmed bandit problem with multiple plays part i: I.i.d. rewards, part ii: Markovian rewards. Technical Report UCB/ERL M86/62, EECS Department, University of California, Berkeley, 1986.

[7] N. Balcan, N. Bansal, A. Beygelzimer, D. Coppersmith, J. Langford, and G. B. Sorkin. Robust reductions from ranking to classification. In *Proceedings of the 20th Annual Conference on Learning Theory*, pages 604–619, 2007.

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

[9] J. K. Bradley and R. Shapire. FilterBoost: Regression and classification on large datasets. In *Advances in Neural Information Processing Systems 20*, pages 185–192, 2008.

[10] P. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In *Machine Learning Proceedings of the 15th International Conference*, pages 82–90. Morgan Kaufmann, 1998.

[11] U. Brefeld and T. Scheffer. AUC maximizing support vector learning. In *Proceedings of the ICML Workshop on ROC Analysis in Machine Learning*, 2005.

[12] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998. Proceedings of the Seventh International World Wide Web Conference.

[13] M. Buro. Improving heuristic mini-max search by supervised learning. *Artificial Intelligence*, pages 85–99, 2002.

[14] B. B. Cambazoglu, H. Zaragoza, O. Chapelle, J. Chen, C. Liao, Z. Zheng, and J. Degenhardt. Early exit optimizations for additive machine learned ranking systems. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, pages 411–420, New York, NY, USA, 2010. ACM.

[15] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[16] N. Cesa-Bianchi and G. Lugosi. Combinatorial bandits. In *Proceedings of the 22nd Conference on Learning Theory*, 2009.

[17] C. C. Chang and C. J. Lin. LIBSVM: a library for support vector machines, 2001.

[18] C.-C. Chang and C.-J. Lin. LIBSVM:a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1– 27:27, 2011.

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

[20] S. Chopra. On the spanning tree polyhedron. *Operations Research Letters*, 8(1):25–29, 1989.

[21] W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–279, 1999.

[22] R. Collobert, S. Bengio, and C. Williamson. SVMTorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001.

[23] C. Cortes and M. Mohri. AUC optimization vs. error rate minimization. In *Advances in Neural Information Processing Systems 16*, 2004.

[24] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[25] C. Domingo and O. Watanabe. MadaBoost: A Modification of AdaBoost. In *Proceedings of 13th Annual Conference on Computational Learning Theory*, pages 180–189, 2000.

[26] H. Drucker, C. J. Burges, L. Kaufman, C. J. C, B. L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines, 1996.

[27] K. Duh and K. Kirchhoff. Learning to rank with partially-labeled data. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 251–258, New York, NY, USA, 2008. ACM.

[28] J. Edmonds. Submodular functions, matroids, and certain polyhedra. In *Combinatorial Structures and Their Applications*, pages 69–87, 1970.

[29] J. Edmonds. Matroids and the greedy algorithm. *Mathematical Programming*, 1(1):127–136, 1971.

[30] Y. Freund, R. Iyer, R. E. Shapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

[31] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[32] S. Fujishige. *Submodular functions and optimization.* Elsevier Science, 2nd edition, 2005.

[33] T. Fujita, K. Hatano, and E. Takimoto. Combinatorial online prediction via metarounding. In *Proceedings of the 24th International Conference on Algorithmic Learning Theory*, pages 68–82, 2013.

[34] J. Fürnkranz. Machine learning in games: A survey. In *Machines That Learn to Play Games, chapter 2*, pages 11–59. Nova Science Publishers, 2000.

[35] D. Gavinsky. Optimally-smooth adaptive boosting and application to agnostic learning. *Journal of Machine Learning Research*, 2003.

[36] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, Dec. 1992.

[37] K. Hatano. Smooth boosting using an information-based criterion. In *Proceedings of the 17 th International Conference on Algorithmic Learning Theory*, pages 304–319, 2006.

[38] E. Hazan. The convex optimization approach to regret minimization. In Suvrit Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for Machine Learning*, chapter 10, pages 287–304. MIT Press, 2011.

[39] D. P. Helmbold and M. K. Warmuth. Learning permutations with exponential weights. *Journal of Machine Learning Research*, 10:1705–1736, 2009.

[40] M. Herbster and M. Warmuth. Tracking the best linear predictor. *Journal of Machine Learning Research*, 1:281–309, 2001.

[41] K. Hoki. Optimal control of minimax search results to learn positional evaluation. In *The 11th Game Programming Workshop*, pages 78–83, 2006. (In Japanese).

[42] S. Iwata. Submodular function minimization. *Mathematical Programming, Ser. B*, 112:45–64, 2008.

[43] T. Joachims. Making large-scale support vector machine learning practical. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in kernel methods*, pages 169–184. MIT Press, Cambridge, MA, USA, 1999.

[44] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002.

[45] T. Joachims. A support vector method for multivariate performance measures. In *Proceedings of the 22nd international conference on Machine learning*, pages 377–384, New York, NY, USA, 2005. ACM.

[46] S. Kakade, A. T. Kalai, and L. Ligett. Playing games with approximation algorithms. *SIAM Journal on Computing*, 39(3):1018–1106, 2009.

[47] A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

[48] Y. L. Kazuki Uematsu. On theoretically optimal ranking functions in bipartite ranking. Technical report, Technical Report 863, Department of Statistics, The Ohio State University, Dec. 2011.

[49] W. M. Koolen, M. K. Warmuth, and J. Kivinen. Hedging structured concepts. In *Proceedings of the 23rd Conference on Learning Theory*, pages 93–105, 2010.

[50] W. Kotlowski, K. J. Dembczynski, and E. Hllermeier. Bipartite ranking through minimization of univariate loss. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning*, pages 1113–1120, New York, NY, USA, 2011. ACM.

[51] T. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.

[52] E. L. Lawler. On Sequencing jobs to minimize weighted completion time subject to precedence constraints. *Annals of Discrete Mathematics 2*, 2:75–90, 1978.

[53] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[54] P. M. Long and R. A. Servedio. Boosting the area under the ROC curve. In *Advances in Neural Information Processing Systems 20*, 2008.

[55] A. T. Mehryar Mohri, Afshin Rostamizadeh. *Foundations of Machine Learning*. The MIT Press, 2012.

[56] J. Moribe, K. Hatano, E. Takimoto, and M. Takeda. Smooth boosting for margin-based ranking. In *Proceedings of the 19th International Conference on Algorithmic Learning Theory*, pages 227–239, 2008.

[57] K. Nagano. A faster parametric submodular function minimization algorithm and applications. Technical Report METR 2007–43, Department of Mathematical Informatics, Graduate School of Information Science and Technology, University of Tokyo, 2007.

[58] J. B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. In *Proceedings of the 12th International Conference on Integer Programming and Combinatorial Optimization*, pages 240–251, 2007.

[59] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods*, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

[60] A. Rakotomamonjy. Optimizing area under roc curve with svms. In J. Hernndez-Orallo, C. Ferri, N. Lachiche, and P. A. Flach, editors, *ROCAI*, pages 71–80, 2004.

[61] G. Rätsch. *Robust Boosting via Convex Optimization: Theory and Applications*. PhD thesis, University of Potsdam, 2001.

[62] G. Rätsch and M. K. Warmuth. Efficient margin maximizing with boosting. *Journal of Machine Learning Research*, 6:2131–2152, 2005.

[63] M. Richardson, A. Prakash, and E. Brill. Beyond PageRank: Machine learning for static ranking. In *Proceedings of the 15th International Conference on World Wide Web*, pages 707–715, New York, NY, USA, 2006. ACM.

[64] C. Rudin. Ranking with a P-Norm Push. In *Proceedings of 19th Annual Conference on Learning Theory*, pages 589–604, 2006.

[65] C. Rudin and R. E. Schapire. Margin-based Ranking and an Equivalence between AdaBoost and RankBoost. *Journal of Machine Learning Research*, 10:2193–2232, 2009.

[66] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector Algorithms. *Neural Computation*, 12(5):1207–1245, 2000.

[67] R. A. Servedio. Smooth boosting and learning with malicious noise. *Journal of Machine Learning Research*, 4:473–489, 2003.

[68] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal Estimated sub-GrAdient SOlver for Svm. In *Proceedings of the 24th International Conference on Machine Learning*, pages 807–814, New York, NY, USA, 2007. ACM.

[69] M. Smola, A. J. Smola, G. Ratsch, B. Scholkopf, J. Kohlmorgen, and V. Vapnik. Predicting time series with support vector machines, 1997.

[70] D. Suehiro, K. Hatano, H. Bannai, E. Takimoto, and M. Takeda. Learning evaluation functions for shogi using SVM-based bipartite ranking learning. *IEICE technical report*, 110(265):113–118, Oct. 2010.

[71] D. Suehiro, K. Hatano, H. Bannai, E. Takimoto, and M. Takeda. Learning shogi evaluation functions using kernel methods. In *The 15th Game Programming Workshop*, pages 23–27, 2010.

[72] D. Suehiro, K. Hatano, H. Bannai, E. Takimoto, and M. Takeda. Learning evaluation functions for shogi using SVM-based bipartite ranking learning. *IEICE Transaction*, J97-D(3), March 2012. to be appeared.

[73] D. Suehiro, K. Hatano, S. Kijima, E. Takimoto, and K. Nagano. Online prediction under submodular constraints. In *Proceedings of the 23rd International Conference on Algorithmic Learning Theory*, pages 260–274, 2012.

[74] D. Suehiro, K. Hatano, and E. Takimoto. Approximate reduction from AUC maximization to 1-norm soft margin optimization. In *Proceedings of the 22nd International Conference on Algorithmic Learning Theory*, pages 324–337, 2011.

[75] E. Takimoto and M. K. Warmuth. Predicting nearly as well as the best pruning of a planar decision graph. *Theoretical Computer Science*, 288(2):217–235, Oct. 2002.

[76] E. Takimoto and M. K. Warmuth. Path kernels and multiplicative updates. *Journal of Machine Learning Research*, 4:773–818, Dec. 2003.

[77] G. Tesauro. Programming backgammon using self-teaching neural nets. *Artificial Intelligence*, 134:181–199, 2002.

[78] W. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Bulletin of the American Mathematics Society*, 25:285–294, 1933.

[79] V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[80] A. von Arnim, U. Faigle, and R. Schrader. The permutahedron of series-parallel posets. *Discrete Applied Mathematics*, 28(1):3–9, 1990.

[81] A. von Arnim and A. S. Schulz. Facets of the generalized permutahedron of a poset. *Discrete Applied Mathematics*, 72:179–192, 1997.

[82] V. Vovk. Aggregating strategies. In *Proceedings of the 3rd Annual Workshop on Computational Learning Theory*, pages 371–386, 1990.

[83] M. Warmuth, K. Glocer, and G. Rätsch. Boosting algorithms for maximizing the soft margin. In *Advances in Neural Information Processing Systems 20*, pages 1585–1592, 2008.

[84] M. K. Warmuth and D. Kuzmin. Randomized online PCA algorithms with regret bounds that are logarithmic in the dimension. *Journal of Machine Learning Research*, 9:2287–2320, 2008.

[85] M. Weimer, A. Karatzoglou, Q. V. Le, and A. J. Smola. Cofi rank - maximum margin matrix factorization for collaborative ranking. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1593–1600, Cambridge, MA, 2008. MIT Press.

[86] S. Yasutake, K. Hatano, S. Kijima, E. Takimoto, and M. Takeda. Online linear optimization over permutations. In *Proceedings of the 22nd International Symposium on Algorithms and Computation (ISAAC 2011)*, pages 534–543, 2011.

[87] H. Yu, J. Kim, Y. Kim, S. Hwang, and Y. H. Lee. An efficient method for learning nonlinear Ranking SVM functions. *Information Sciences*, 209(0):37–48, 2012.

[88] G. M. Ziegler. *Lectures on Polytopes*. Graduate texts in mathematics 152. Springer-Verlag, 1995.

[89] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning*, pages 928–936, 2003.