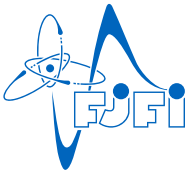
Czech Technical University in Prague
Faculty of Nuclear Sciences and
Physical Engineering

General Framework for Classification at the Top

Dissertation



Author: Ing. Václav Mácha
Academic year: 2021/2022

Poděkování:

Thanks thanks

Čestné prohlášení:

Prohlašuji na tomto místě, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze dne 1. prosince 2021

.....
Ing. Václav Mácha

Název: **Title title title title title title**

Autor: Ing. Václav Mácha

Obor: Matematické inženýrství

Druh práce: Disertační práce

Školitel: doc. Ing Václav Šmídl, Ph.D.

Školitel specialista: Mgr. Lukáš Adam, Ph.D.

Abstrakt: Abstract abstract abstract abstract abstract abstract abstract ab-
stract abstract abstract abstract abstract abstract abstract abstract
abstract abstract abstract abstract abstract abstract abstract ab-
stract abstract abstract abstract abstract abstract abstract abstract
abstract abstract abstract abstract abstract abstract abstract ab-
stract abstract abstract abstract abstract abstract abstract abstract
abstract abstract abstract abstract abstract abstract abstract ab-
stract abstract abstract abstract abstract abstract abstract abstract
abstract abstract abstract abstract abstract abstract abstract ab-
stract abstract abstract abstract abstract abstract abstract abstract
abstract

Klíčová slova: Keywords keywords keywords keywords keywords keywords key-
words keywords keywords keywords keywords keywords key-
words

Title: **Title title title title title title**

Abstract: Abstract abstract

Keywords: Keywords keywords keywords keywords keywords keywords keywords keywords keywords keywords keywords keywords keywords

Contents

1

Introduction to Binary Classification

The problem of data classification is very important in the modern world. The classification aims to find a relation between a set of objects and target variables based on some objects' properties. The properties of the objects are usually called features, and the target variables are usually called labels. Many real-world problems can be formulated as classification tasks:

- **Medical Diagnosis:** In medicine, the classification is often used to improve disease diagnosis. In such a case, the features are medical records such as the patient's blood tests, temperature, or roentgen images. The target variable is if the patient has some disease. For example, classification can be used to process mammogram images and detect cancer [1, 2].
- **Internet Security:** These days, the internet is a crucial part of our lives. With the increasing usage of the internet, the number of attacks increases as well. An essential part of the defense is intrusion detection systems [3, 4] that search for malicious activities (network attacks) in network traffic. Classification can be used to improve such systems as shown in [5, 6].
- **Marketing:** In marketing, the task can be to classify customers based on their buying interests. Such information can be used to build a personalized recommendation system for customers and therefore increase income [7, 8].

Besides these three examples, applications of classification can be found in almost every academic or even industrial field. Furthermore, a vast number of algorithms try to solve classifications problems. Typically these algorithms consist of three phases:

- **Training:** The classification problems usually fall into the category of supervised learning. It means that we assume the prior knowledge of the target classes in the training phase. The training data typically consists of pairs (sample, label) and can be described as follows

  $$\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n},$$

  where the sample $\boldsymbol{x}_i \in \mathbb{R}^d$ is a d-dimensional vector of features that describes the object of interes and the label $y_i \in \{1, 2, \ldots, k\}$ represents target class. Moreover $n \in \mathbb{N}$ is a number of training samples and $k \in \mathbb{N}$ is a number of target classes. In this phase, the algorithm uses the training data to learn a model, i.e., set model parameters according to some predefined criterion, to describe the training data as best as possible.
- **Validation:** All algorithms usually have some hyperparameters that can be changed to improve the resulting model. The validation phase is used to select the best hyperparameter settings that lead to the most performant and robust model.
- **Testing:** In the testing phase, the model is used to assign labels $\hat{y}_i \in \{1, 2, \ldots, k\}$ to the data from the testing set, which is not known during the training phase.

The previous definition of the training set is general for any classification problem with multiple classes. However, we focus on the special subclass of classification problems called binary classification in this work. The binary classification is a special case of classification in which the number of classes is $k = 2$. These two classes are usually referred to as negative and positive classes. Moreover, the positive class is usually the one we are more interested. Returning to example with cancer, the positive class would represent that the patient has cancer while the negative that the patient is healthy.

Notation 1.1: Dataset

In this work, we use label 0 to encode the negative class and label 1 to encode the positive class. Moreover, by a dataset of size $n \in \mathbb{N}$ we mean a set of pairs in the following form

$$\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n},$$

where $\boldsymbol{x}_i \in \mathbb{R}^d$ represents samples, $d \in \mathbb{N}$ its dimension and $y_i \in \{0, 1\}$ corresponding labels. To simplify future notation, we denote a set of all indices of dataset $\mathcal{D}$ as $\mathcal{I} = \mathcal{I}_- \cup \mathcal{I}_+$, where

$$\mathcal{I}_- = \{i \mid i \in \{1, 2, \ldots, n\} \ \wedge \ y_i = 0\},$$
$$\mathcal{I}_+ = \{i \mid i \in \{1, 2, \ldots, n\} \ \wedge \ y_i = 1\}.$$

We also denote the number of negative samples in $\mathcal{D}$ as $n_- = |\mathcal{I}_-|$ and the number of positive samples in $\mathcal{D}$ as $n_+ = |\mathcal{I}_+|$, i.e. total number of samples is $n = n_- + n_+$.

The goal of any classification problem is to classify given samples with the highest possible accuracy or, in other words, with the lowest possible error. In the case of binary classification, there are two types of error: positive sample is classified as negative and vice versa. Formally, using the Notation 1.1, the minimization of these two types of errors can be written as follows

$$\begin{aligned} \underset{\boldsymbol{w}, t}{\text{minimize}} \quad & \lambda_1 \sum_{i \in \mathcal{I}_-} \mathbb{1}_{[s_i \geq t]} + \lambda_2 \sum_{i \in \mathcal{I}_+} \mathbb{1}_{[s_i < t]} \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \end{aligned} \tag{1.1}$$

where $\lambda_1, \lambda_2 \in \mathbb{R}$, the function $f : \mathbb{R}^d \to \mathbb{R}$ is called model and $\mathbb{1}_{[\cdot]}$ is Iverson function that is used to counts misclassified samples and is defined as

$$\mathbb{1}_{[x]} = \begin{cases} 0 & \text{if } x \text{ is false,} \\ 1 & \text{if } x \text{ is true.} \end{cases} \tag{1.2}$$

Moreover, the vector $\boldsymbol{w} \in \mathbb{R}^d$ represents trainable parameters (weights) of the model f and $t \in R$ represents a decision threshold. The parameters $\boldsymbol{w}$ are determined from training data during the training phase of the algorithm. Although the decision threshold t can also be determined from the training data, in many cases, it is fixed. For example, many algorithms assume that the classification score $s_i = f(\boldsymbol{x}_i; \boldsymbol{w})$ given by the model f represents the probability that the sample $\boldsymbol{x}_i$ belongs to the positive class. Therefore, the decision threshold is set to $t = 0.5$, and the sample is classified as positive if its classification score is larger than this threshold. In Notation 1.2, we summarize the notation that is used in the rest of the work.

Notation 1.2: Classifier

By classifier, we always mean pair of model f and corresponding decision threshold t. By model, we mean a function $f : \mathbb{R}^d \to \mathbb{R}$ which maps samples $\boldsymbol{x}$ to its classification scores s, i.e. for all $i \in \mathcal{I}$ the classification score is defined as

$$s_i = f(\boldsymbol{x}_i; \boldsymbol{w}),$$

where $\boldsymbol{w}$ represents trainable parameters (weights) of the model. Predictions are defined for all $i \in \mathcal{I}$ in the following way

$$\hat{y}_i = \begin{cases} 1 & \text{if } s_i \geq t, \\ 0 & \text{otherwise.} \end{cases}$$

1.1 Performance Evaluation

In the previous section, we defined general binary classification problem (1.1). However, we did not discuss how to measure the performance of the resulting classifier. In this section, we introduce basic performance metrics that are used to measure the performance of binary classifiers.

1.1.1 Confusion Matrix

Based on the prediction $\hat{y}_i$ and an actual label y_i of the sample $\boldsymbol{x}_i$, each sample can be assigned to one of the four following categories:

- **True negative:** sample $\boldsymbol{x}_i$ is negative and is classified as negative, i.e. $y_i = 0 \ \wedge \ \hat{y}_i = 0$.
- **False positive:** sample $\boldsymbol{x}_i$ is negative and is classified as positive, i.e. $y_i = 0 \ \wedge \ \hat{y}_i = 1$.
- **False negative:** sample $\boldsymbol{x}_i$ is positive and is classified as negative, i.e. $y_i = 1 \ \wedge \ \hat{y}_i = 0$.
- **True positive:** sample $\boldsymbol{x}_i$ is positive and is classified as positive, i.e. $y_i = 1 \ \wedge \ \hat{y}_i = 1$.

Using these four categories, we can construct so-called confusion matrix (sometimes also called contingency table) [9] that predictions for all samples from the given dataset $\mathcal{D}$. An illustration of the confusion matrix is shown in Figure 1.1. If we denote the vector of all classification scores given by model f as $\boldsymbol{s} \in \mathbb{R}^n$, we can compute all fields of the confusion matrix as follows

$$\begin{aligned} \mathrm{tp}(\boldsymbol{s}, t) &= \sum_{i \in \mathcal{I}_+} \mathbb{1}_{[s_i \geq t]}, & \mathrm{fn}(\boldsymbol{s}, t) &= \sum_{i \in \mathcal{I}_+} \mathbb{1}_{[s_i < t]}, \\ \mathrm{tn}(\boldsymbol{s}, t) &= \sum_{i \in \mathcal{I}_-} \mathbb{1}_{[s_i < t]}, & \mathrm{fp}(\boldsymbol{s}, t) &= \sum_{i \in \mathcal{I}_-} \mathbb{1}_{[s_i \geq t]}, \end{aligned} \tag{1.3}$$

where $\mathbb{1}_{[\cdot]}$ is the Iverson function (1.2). In the following text, we sometimes use simplified notation $\mathrm{tp} = \mathrm{tp}(\boldsymbol{s}, t)$ (and similar notation for other counts) for example to define classification metrics. In such cases, the vector of classification scores and decision threshold is fixed and is known from the context. Using the simplified notation, we can define true-positive, false-positive, true-negative, and false-negative rates as follows

$$\mathrm{tpr} = \frac{\mathrm{tp}}{n_+}, \qquad \mathrm{fnr} = \frac{\mathrm{fn}}{n_+}, \qquad \mathrm{tnr} = \frac{\mathrm{tn}}{n_-}, \qquad \mathrm{fpr} = \frac{\mathrm{fp}}{n_-}. \tag{1.4}$$

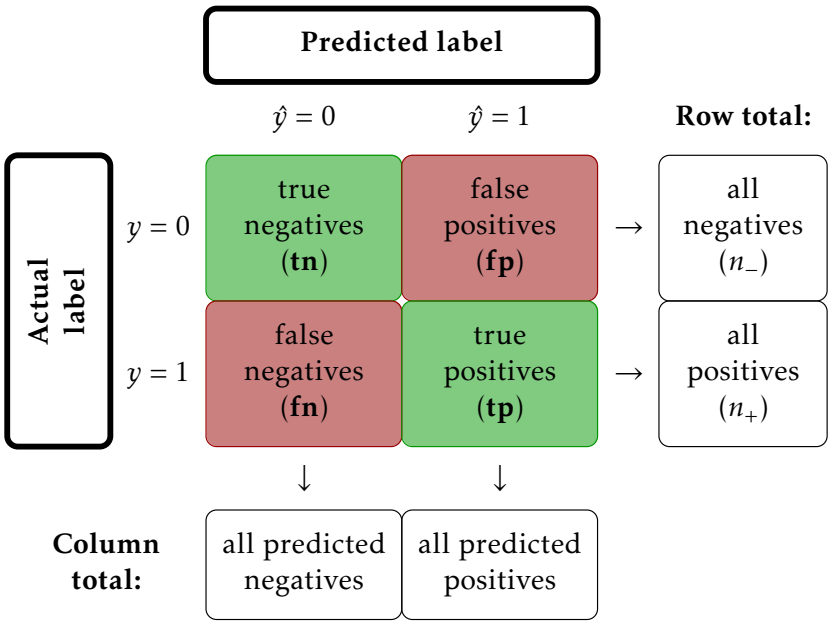


Figure 1.1: The confusion matrix for the binary classification problem, where the negative class has label 0 and the positive class has label 1. The true (target) label is denoted y and predicted label is denoted $\hat{y}$.

Figure 1.2 shows the relation between classification rates and the decision threshold. The blue and red curves represent the theoretical distribution of the scores of negative and positive samples, respectively. The position of the decision threshold determines the values of the classification rates. The higher the decision threshold, the lower the false-positive rate, but at the same time, the higher the false-negative rate. Similarly, the lower the decision threshold, the higher the false-positive rate and the lower the false-negative rate. Ideally, classification without errors is the goal, but it is not usually possible. If we look at the general definition of the binary classification problem (1.1), the objective function is just the weighted sum of false-positive and false-negative samples. Therefore, we can use the notation (1.4) and rewrite the problem (1.1) to

$$\begin{aligned} \underset{\boldsymbol{w},t}{\text{minimize}} \quad & \lambda_1 \cdot \mathrm{fp}(\boldsymbol{s},t) + \lambda_2 \cdot \mathrm{fn}(\boldsymbol{s},t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i;\boldsymbol{w}), \quad i \in \mathcal{I}. \end{aligned} \tag{1.5}$$

The parameters $\lambda_1, \lambda_2 \in \mathbb{R}$ are used to specify which error is more serious for the particular classification task.

The confusion matrix is not the only way to measure the performance of binary classifiers. For example, there are many different classification matrices, and many of them are derived directly from the confusion matrix [9, 10, 11, 12]. As an example, we can mention accuracy and balanced accuracy defined by

$$\mathrm{acc} = \frac{\mathrm{tp}+\mathrm{tn}}{n} \qquad\qquad \mathrm{bacc} = \frac{\mathrm{tpr}+\mathrm{tnr}}{2}$$

At the end of this chapter, we provide Table 1.1 that summarizes all classification metrics used in this work. Moreover, in the following section, we introduce a different approach for the performance evaluation of binary classifiers.

1.1.2 ROC Analysis

In the previous section, we defined a general binary classification formulation (1.5) that minimizes a weighted sum of false-positive and false-negative counts. Therefore, we always have

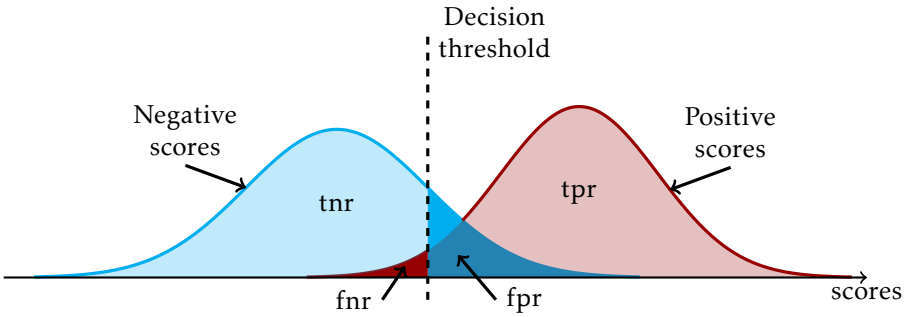


Figure 1.2: The relation between classification scores and rates. The blue curve is the theoretical distribution of the scores of negative samples, and the red curve the theoretical distribution of the scores of positive samples. Filled areas with light blue or light red represent true-negative and true-positive rates. Similarly, the filled areas with dark blue or dark red represent false-positive and false-negative rates.

to find some trade-off between the false-positive and false-negative counts and select the best hyperparameters λ_1, λ_2, for given tasks. There is no universal truth, which of these two errors is worse. For example, It is probably better to classify a healthy patient as sick and do additional tests than the other way around. On the other hand, in computer security, an antivirus program with a lot of false-positive alerts is useless since it is disruptive to the user. One way to visualize the trade-off between false-positive and false-negative errors is Receiver Operating Characteristic (ROC) space [13, 9].

ROC space is a two-dimensional space with the x-axis equal to the false-positive rate and the y-axis to the true-positive rate. The left-hand side of Figure 1.3 shows the ROC space with five highlighted points. Each point in the ROC space represents one fixed classifier, i.e., one pair of model f and decision threshold t. There are several important points in the ROC space. The point $(0,0)$ represents a classifier classifying all samples as negative, while $(1,1)$ is a classifier classifying all samples as positive. Both these classifiers are useless. On the other hand, the point $(0,1)$ represents the perfect classifier. Generally, we can say that one classifier is better than another if its representation in ROC space is to the northwest of the second one. In such a case, the classifier has a higher true-positive rate and lower false-positive rate than the second one. For example, in Figure 1.3, classifier **B** is better than classifier **C**. On the other hand, it is impossible to say which classifier is better if one has a higher true-positive rate and the other has a lower false-positive rate. We can see this situation for classifier **B** and **A**. In such a case, the preference depends on the given problem, as discussed at the beginning of this section.

Another important part of the ROC space is the diagonal line highlighted in red in Figure 1.3. Any classifier that appears on this diagonal provides the same performance as a random classifier. For example, classifier **C** is represented in ROC space by point $(0.7, 0.7)$. It means that this classifier randomly classifies 70% of samples as positive. Therefore, any classifier that appears in ROC space in the lower right triangle is worse than a random classifier. There are usually no classifiers in this area since any classifier from the lower right triangle can be easily improved. If we negate the decision of such a classifier for every sample, we get its negated version in the upper left triangle. Such a situation is in Figure 1.3 for classifier **E** and **B**. We negate every decision of classifier **E**. Therefore, all true-positive samples became false-negative and vice versa. Since classifier **E** has a false-negative rate of 0.8, we can deduce that negated classifier will have a true-positive rate of 0.8. Similarly, since classifier **E** has a true-negative rate of 0.4, its negated version will have a false-positive rate of 0.4. Therefore the negated version of classifier **E** is represented in ROC space by point $(0.4, 0.8)$, which is classifier **B**.

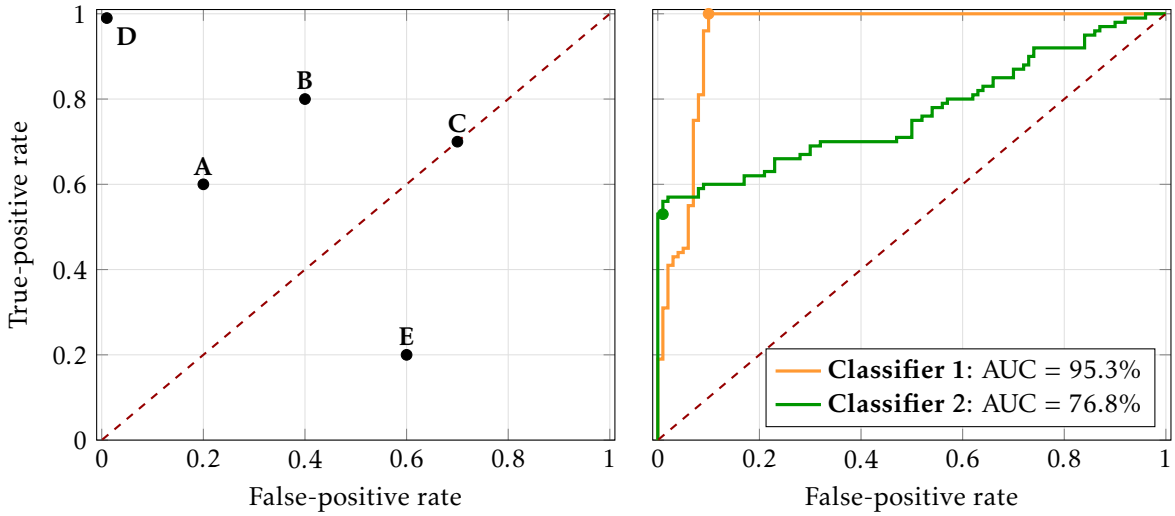


Figure 1.3: A basic representation of the ROC space with five different classifiers. (**left**) A comparison of ROC curves for two different classifiers. (**right**)

Many classifiers produce directly a decision that samples are positive or negative. As an example, we can mention decision trees. Such classifiers are always represented as a single point in the ROC space. Moreover, this class of classifiers does not fit into our description from Notation 1.2 since such classifiers do not provide any classification scores. In this text, we restrict to the classifiers as defined in Notation 1.2. We assume that the classifier consists of the model f that produces classification scores and the decision threshold t. Many standard classifiers such as neural networks or logistic regression fall into this setting. Even though the decision threshold is determined during the training process, it is possible to change it and obtain different predictions. This possibility is very often used to produce so-called ROC curves [9]. ROC curve represents how model f behaves for different thresholds t varying from $-\infty$ to $+\infty$. Right-hand side of Figure 1.3 provides an example of two ROC curves for two different classifiers. **Classifier 1** provides accuracy 95% and is represented by the orange dot, while the orange line represents its ROC curve. **Classifier 2** represented by the green dot provides accuracy 76%, and the green line represents its ROC curve. A standard method how for comparing two classifiers using ROC curves is to compare corresponding areas under the ROC curves (AUC) [14, 15]. Such an approach is a simple way to reduce the ROC curve to one number. In the case of standard binary classification, the larger the AUC, the better. In Figure 1.3 we can see that the orange classifier has AUC 95.3% while the green one has only 76.8%. Therefore, for most classification problems, the orange classifier is better.

Since both false-positive and true-positive rates are non-increasing functions of threshold t, we can efficiently compute the ROC curve from sorted classification scores $\boldsymbol{s}$ given by the model f. In Algorithm 1 we show an efficient algorithm for generating the ROC curve. This algorithm gives us an interesting insight into ROC curves: ROC curves represent the ability of the model to rank the positive samples higher than the negative ones [9]. Moreover, the AUC of a classifier is equivalent to the probability that the classifier will rank a randomly chosen positive sample higher than a randomly chosen negative sample [9]. By comparing the classifiers from the right-hand side of Figure 1.3, we can deduce that **Classifier 1** is generally better at a false-positive rate larger than 0.01. Otherwise, **Classifier 2** is the better one. Therefore, there is a specific region of the ROC space where **Classifier 2** outperforms **Classifier 1**. In the next section, we discuss multiple different problems which focus on the performance only at low false-positive rates.

Algorithm 1 Efficient algorithm [9] for generating ROC curve from sorted classification scores.

Require: Sorted classification scores $\boldsymbol{s}_{[\cdot]}$ in decreasing order $s_{[1]} \geq \cdots \geq s_{[n]}$

Set $\mathrm{tp} \leftarrow 0$, $\mathrm{fp} \leftarrow 0$, $s_{\mathrm{prev}} = +\infty$, and an empty set $ROC = \{\}$ of points in ROC space
for $l \in \{1, \ldots, n\}$ **do**
    if $s_{[i]} \neq s_{\mathrm{prev}}$ **then**
        push $(\mathrm{fp}/n_-, \mathrm{tp}/n_+)$ into ROC
    end if
    if $y_{[i]} = 1$ **then**
        $\mathrm{tp} \leftarrow \mathrm{tp} + 1$
    else
        $\mathrm{fp} \leftarrow \mathrm{fp} + 1$
    end if
end for
push $(\mathrm{fp}/n_-, \mathrm{tp}/n_+) = (1, 1)$ into ROC

1.2 Classification at the Top

As we discussed in the previous sections, standard binary classification aims to separate positive and negative samples with the lowest possible error on the whole dataset. The performance of a binary classifier can be measured using classification metrics such as accuracy or using ROC curves and their AUC. However, it is desirable to focus only on a small number of the most relevant samples in many applications. In such a case, the goal is to maximize the performance only on the relevant samples. Since the rest of the samples are irrelevant, their performance is unimportant. Figure 1.4 shows the difference between the standard classifier (**Classifier 1**) that maximizes the accuracy and the classifier that focuses only on the classification at the top (**Classifier 2**). In this particular case, **Classifier 2** maximizes the number of positive samples that are ranked higher than the worst negative sample. Formally, **Classifier 2** maximizes the following metric

$$\mathrm{pos@top}(\boldsymbol{s}) = \frac{1}{n_+} \sum_{i \in \mathcal{I}_+} \mathbb{1}_{[s_i \geq \max_{j \in \mathcal{I}_-} s_j]}. \tag{1.6}$$

For both classifiers, Figure 1.4 shows two different decision thresholds. The black threshold is the one for which the classifier was trained, while the green one represents the worst negative sample. For **Classifier 2** these two thresholds coincide. We can observe that **Classifier 1** provides a much better separation of positive and negative samples. Only a few samples above the black threshold ruin perfect separation. On the other hand, the separation provided by **Classifier 2** is much worse since half of the positive samples are mixed with negative ones. Therefore, the accuracy of **Classifier 1** is 95% while the accuracy of **Classifier 2** is only 76%. However, in terms of metric (1.6) the situation is quite different. Since there are few negative outliers, there is only 19% of positive samples above the worst negative for **Classifier 1**. **Classifier 2** achieves to push 53% of positive samples above the worst negative and therefore provides better performance for this case. The same behavior can also be demonstrated using ROC curves as shown in Figure 1.5. The orange line represents ROC curve for **Classifier 1** and the green one for **Classifier 2**. There are two important points for **Classifier 1** in the figure. The first one is highlighted by an orange circle and represents the point in the ROC space that corresponds to the actual threshold. In other words, it corresponds to the black threshold from Figure 1.4. The second point is highlighted by an orange square and corresponds to the green threshold from Figure 1.4. Since for **Classifier 2** both thresholds coincide, there is only one point in Figure 1.5 highlighted by a green square. The superiority of **Classifier 1** in the overall performance is evident from the left-hand side of the figure. The AUC for **Classifier 1** is

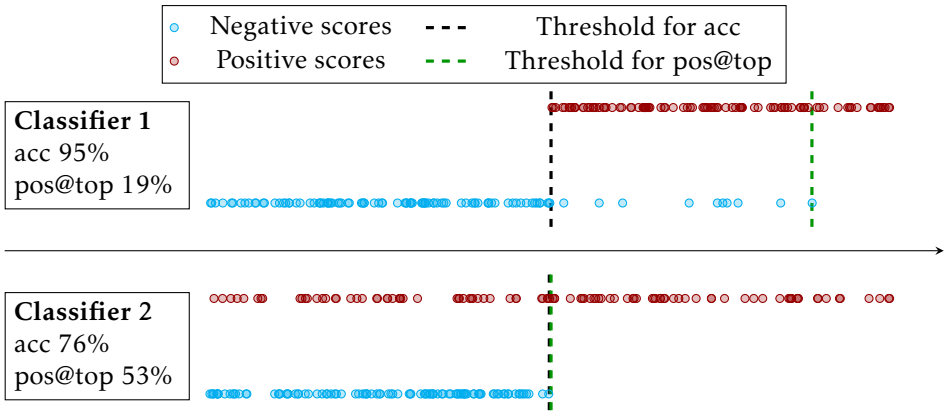


Figure 1.4: Difference between standard classifiers (**Classifier 1**) and classifiers maximizing pos@top metric (**Classifier 2**). While the former has a good total acc, the latter has a good pos@top metric.

95.3% and only 76.8% for **Classifier 2**. Moreover, we can see that there is only a small region of ROC space, where **Classifier 2** provides a higher true-positive rate. However, this region is very interesting. The right-hand side of Figure 1.5 provides the same ROC curves but with a logarithmic x-axis. The logarithmic scale allows us to concentrate on very low false-positive rates. We can see, that if the false-positive rate is lower than $7 \cdot 10^{-1}$, then **Classifier 2** provides better true-positive rate than **Classifier 1**. It means that in this region of the ROC space, **Classifier 2** provides better sorting of positive and negative samples, i.e., more positive samples are ranked higher than negative ones. Moreover, classification metric (1.6) is equivalent to the y-coordinate at the lowest non-zero false-positive rate. It is evident from the right-hand side of Figure 1.5 where the value of metric (1.6) is highlighted using squares for both classifiers.

As discussed above, **Classifier 1** focuses on the overall performance, while the **Classifier 2** on the performance on low false-positive rates. The latter classifier can be handy for search engines such as Google or DuckDuckGo, where the goal is to have all relevant results on the first few pages. The results on page 50 are usually of no interest to anyone, so it is crucial to move the most relevant results to the few first pages [16]. Therefore, it is essential to push as many positive samples above some small portion of negative samples. The rest of the chapter presents three main categories of problems that solve this kind of problem. Moreover, in the next chapter, we show that at least some formulations from these three categories are closely related to binary classification.

1.2.1 Ranking Problems

The first category is ranking problems. The ranking algorithms play a crucial role in many information retrieval problems:

- **Document (Text) retrieval systems** are used for obtaining relevant documents from the collection of documents based on the relevance to the user's query. Such systems are widely used for accessing books, journals, or any other documents. However, the most visible application is search engines such as Google or DuckDuckGo.
- **Collaborative filtering** is one of the techniques used to predict the user's rating of a new product based on the past ratings of users with similar rating patter. Such systems can be used to generate music or video playlist automatically. Therefore, such systems are widely used in services such as Youtube or Spotify.

The two examples above show that ranking problems usually depend on users' feedback or preferences. In binary classification, we have only the labels that represent if the samples are

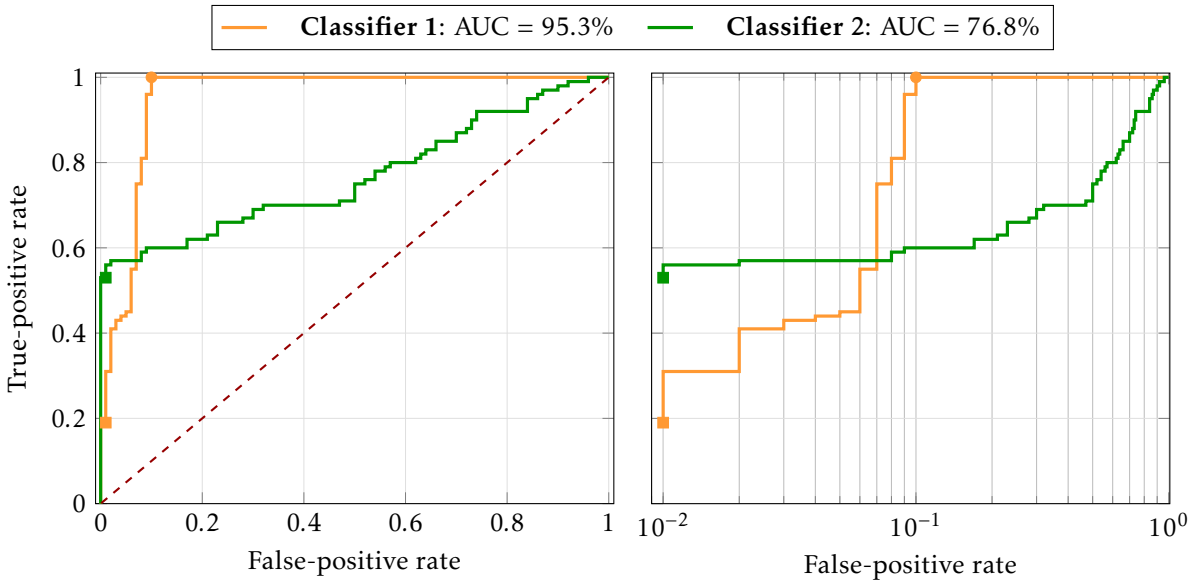


Figure 1.5: Difference between standard classifiers (**Classifier 1**) and classifiers maximizing pos@top metric (**Classifier 2**). While the former has a good total acc, the latter has a good pos@top metric.

positive or negative. On the other hand, ranking problems use multiple ways to describe the users' feedback. One approach uses the feedback function $\Phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ to represent the user's preferences [17]. In such a case, the feedback function can be defined for all pairs of samples $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ in the following way

$$\Phi(\boldsymbol{x}_i, \boldsymbol{x}_j) \begin{cases} > 0 & \boldsymbol{x}_i \text{ is prefered over } \boldsymbol{x}_j, \\ = 0 & \text{no preference}, \\ < 0 & \boldsymbol{x}_j \text{ is prefered over } \boldsymbol{x}_i. \end{cases}$$

We can see, that the feedback function specify if the user prefers $\boldsymbol{x}_i$ over $\boldsymbol{x}_j$ or not. Moreover, the feedback function also specifies how strong the preference is, i.e., the higher the volume $|\Phi(\boldsymbol{x}_i, \boldsymbol{x}_j)|$ is the more important the preference is. Many ranking algorithms try to find some ordering of all samples that minimizes the number of wrongly ordered pairs of samples concerning the given feedback function. Consider ranking function $r : \mathbb{R}^d \to \mathbb{R}$. The sample $\boldsymbol{x}_i$ is ranked higher than sample $\boldsymbol{x}_j$ if $r(\boldsymbol{x}_i) > r(\boldsymbol{x}_j)$. Then, the minimization of the number of misordered pairs can be formally written as follows

$$\underset{r}{\text{minimize}} \quad \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} \mathbb{1}_{[r(\boldsymbol{x}_i) \leq r(\boldsymbol{x}_j)]} \cdot \max\Big\{0,\ \Phi(\boldsymbol{x}_i, \boldsymbol{x}_j)\Big\}. \tag{1.7}$$

This problem is hard to solve since the objective function contains a pairwise comparison of all samples. Therefore, the problem is not suitable for large data. *RankBoost* [17] is an boosting algorithm based on the AdaBoost [18] that combines many weak ordering functions to obtain the final ranking. This approach leads to the maximization of the area under the ROC curve [19]. Therefore, RankBoost focuses on the overall performance. However, as we discussed at the beginning of the section, we want to focus only on the small portion of the most relevant samples in many applications. In such a case, this approach is not ideal.

Consider recommendation of movies. In such a case, we only care about if the movie is good or not. What is not important is if one bad movie is ranked higher than another bad movie. Both movies are still bad and therefore not relevant. Many ranking algorithms [19] use the so-called bipartite ranking to address this situation. In such a situation, each sample is

positive (good) or negative (bad), and the goal is to push positive samples above negative ones. The authors of [19] proposed the following formulation

$$\underset{r}{\text{minimize}} \quad \sum_{j\in\mathcal{I}_-}\left(\sum_{i\in\mathcal{I}_+}\mathbb{1}_{[r(\boldsymbol{x}_i)\leq r(\boldsymbol{x}_j)]}\right)^p. \tag{1.8}$$

The authors of [19] also proposed boosting algorithm called p-**Norm Push** to solve the formulation above. Note that for $p = 1$, the formulation (1.8) is very similar to the RankBoost (1.7). In such a case, the resulting ranking function maximizes the AUC and therefore optimizing overall rankig. On the other hand, for $p \to +\infty$, the formulation (1.8) minimizes the largest number of positive samples ranked below any negative sample

$$\underset{r}{\text{minimize}} \quad \max_{j\in\mathcal{I}_-}\sum_{i\in\mathcal{I}_+}\mathbb{1}_{[r(\boldsymbol{x}_i)\leq r(\boldsymbol{x}_j)]}. \tag{1.9}$$

In such a case, the resulting ranking function focus only on the absolute top, i.e., it aims to push as many positive samples above the negative sample with the highest rank. Authors of [20] focus on formulation (1.9) with $p \to +\infty$ and introduces SVM (*Support Vector Machines* [21]) based algorithm called *Infinite Push* to solve it. The major problem of formulation (1.9) is that the objective still contains the pairwise comparison of positive and negative samples. Therefore the formulation is not suitable for large data. To mitigate this issue, the authors of [22] proposed an equivalent formulation in the following form

$$\underset{r}{\text{minimize}} \quad \sum_{i\in\mathcal{I}_+}\mathbb{1}_{[r(\boldsymbol{x}_i)\leq \max_{j\in\mathcal{I}_-} r(\boldsymbol{x}_j)]}. \tag{1.10}$$

Moreover, authors of [22] proposed *TopPush* algorithm with the linear complexity in the number of samples. Note that the objective function of the problem above is almost the same as the metric (1.6). Therefore, this **Classifier 2** from Figures 1.4 and 1.5 corresponds to the ranking function given by *TopPush* algorithm. It shows a tied connection between binary classification and the bipartite ranking problems.

1.2.2 Accuracy at the Top

In the previous section, we introduced formulation (1.10), which focuses on maximizing the number of positive samples above the worst negative sample (the one with the highest rank or highest classification score). This formulation is very useful, as discussed at the beginning of this section. However, such a maximization problem can be unstable since the objective function does not allow false-positive errors. Therefore, if there is one negative outlier with a high score, the number of positive samples above this outlier can be tiny. The authors of [23] focus on similar problems as *TopPush*, but use a different approach. They proposed a formulation called **Accuracy at the Top** which aims to maximize the number of positive samples in the top τ-fraction of all samples. The top τ-fraction of all samples can be formally defined as samples with scores higher then the top τ-quantile of all scores

$$\begin{aligned} \underset{f}{\text{minimize}} \quad & \frac{1}{n_-}\operatorname{fp}(\boldsymbol{s},t)+\frac{1}{n_+}\operatorname{fn}(\boldsymbol{s},t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i), \quad i\in\mathcal{I}, \\ & t = \max\left\{t \;\middle|\; \frac{1}{n}\sum_{i\in\mathcal{I}}\mathbb{1}_{[s_i\geq t]}\geq\tau\right\}, \end{aligned} \tag{1.11}$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is a model. Even though the goal is to maximize the number of positive samples above the top τ-quantile, the objective function contains false-positive and false-negative

rates. It should be sufficient to include only one of them since the definition of the threshold implies the minimization of the other one as well. However, this form of objective function should be more robust [3]. The problem of Accuracy at the Top is useful, for example, in applications where identified samples undergo expensive post-processing, such as human evaluation. For example, potentially useful drugs need to be preselected and manually investigated in drug development. Since the manual investigation is costly, we have to select only a fraction of drugs with the highest potential. However, it is precisely what Accuracy at the Top does.

There are many methods on how to solve Accuracy at the Top. The early approaches aim at solving approximations. For example, the authors of [24] optimize a convex upper bound on the number of errors among the top samples. Due to exponentially many constraints, the method is computationally expensive. In [23] the authors presented an SVM-like formulation. They assume that the top τ-quantile is one of the samples, construct n unconstrained optimization problems with fixed thresholds, solve them and select the best solution. While this removes the necessity to handle the (difficult) quantile constraint, the algorithm is computationally infeasible for a large number of samples. The authors of [3] proposed the projected gradient descent method, where after each gradient step, the quantile is recomputed. In [25] authors suggested new formulations for various criteria and argued that they keep desired properties such as convexity. Finally, the authors of [26] showed that Accuracy at the Top is maximized by thresholding the posterior probability of the relevant class.

1.2.3 Hypothesis Testing

The hypothesis testing operates with null H_0 and alternative H_1 hypothesis. The goal is to either reject the null hypothesis in favor of the alternative or not. Since this problem is binary, two possible errors can occur. Type I occurs when H_0 is true but is rejected, and Type II error happens when H_0 is false but fails to be rejected. The Neyman-Pearson problem minimizes [27] Type II error while keeping Type I error smaller than some predefined bound. Using our notation for the Neyman-Pearson problem, the null hypothesis H_0 states that sample $\boldsymbol{x}$ has a negative label. Then Type I error occurs when the sample is false-positive, while Type II error occurs when the sample is false-negative. Therefore, the Neyman-Pearson problem minimizes the false-negative rate with the prescribed level τ of the false-positive rate. Such constraint can be written in the form of quantile, i.e., the threshold is the top τ-quantile of scores of all negative samples.

$$\begin{aligned} \underset{f}{\text{minimize}} \quad & \frac{1}{n_+}\operatorname{fn}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i), \quad i \in \mathcal{I}, \\ & t = \max\left\{ t \;\middle|\; \frac{1}{n_-}\sum_{i \in \mathcal{I}_-} \mathbb{1}_{[s_i \geq t]} \geq \tau \right\}, \end{aligned}$$

This formulation is very similar to the one for Accuracy at the Top (1.11). The main difference is that the quantile in the constraint is not computed from all but only from negative samples. Also, note one key difference in interpretation. The τ in Accuracy at the Top represents the total amount of samples we want to process with the smallest possible error. On the other hand, in the Neyman-Pearson problem τ represents the maximal acceptable false-positive rate. Therefore, the former approach is useful in situations where we can process only a certain number of samples. The latter is for situations where we have strict constraints on false-positive errors.

Name	Aliases	Formula
true negatives	correct rejection	tn
false positives	Type I error, false alarm	$\text{fp} = n_- - \text{tn}$
true positives	hit	tp
false negatives	Type II error	$\text{fn} = n_+ - \text{tp}$
true negative rate	specificity, selectivity	$\text{tnr} = \frac{\text{tn}}{n_-}$
false positive rate	fall-out	$\text{fpr} = \frac{\text{fp}}{n_-} = 1 - \text{tnr}$
true positive rate	sensitivity, recall, hit rate	$\text{tpr} = \frac{\text{tp}}{n_+}$
false negative rate	miss rate	$\text{fnr} = \frac{\text{fn}}{n_+} = 1 - \text{tpr}$
accuracy	—	$\text{acc} = \frac{\text{tp}+\text{tn}}{n}$
balanced accuracy	—	$\text{bacc} = \frac{\text{tpr}+\text{tnr}}{2}$
precision	positive predictive value	$\text{precision} = \frac{\text{tp}}{\text{tp}+\text{fp}}$

Table 1.1: Summary of classification metrics derived from confusion matrix. The first column shows the name used in this work, while the second column shows alternative names that can be found in the literature. The last column shows the formula based on the confusion matrix.

1.3 Summary

In this chapter, we introduced the general formulation (1.5) for binary classification and discussed how to measure the performance of binary classifiers. The first approach for performance evaluation is based on the confusion matrix. This approach is very straightforward. Moreover, it is possible to derive many different classification matrices from the confusion matrix. Table 1.1 summarizes classification matrices derived from the confusion matrix used in the upcoming chapters. The second approach introduced in this chapter uses the ROC space to visualize the ability of classifiers to rank positive samples above negative ones. Since standard binary classification focuses on optimizing the overall performance, we discussed that there are many problems closely tied to the binary classification that focuses on the performance only of the most relevant samples. Such problems occur in many applications, from search engines to drug development. We also introduced Ranking problems, the problem of Accuracy at the Top, and the Neyman-Pearson problem and discussed their relation to the binary classification. In the upcoming chapters, we focus on these three groups of problems and introduce a general framework to handle them.

2

Framework for Classification at the Top

In the previous chapter, we introduced the general formulation (1.5) and fundamental evaluation matrices for the binary classification problems. Furthermore, in Section 1.2, we introduced three problems closely related to binary classification but focused on specific performance criteria, namely: *Accuracy at the top* problem, *Ranking problems*, and the problem of *Hypothesis testing*. Even though these problems are usually considered separately, they have one crucial thing in common. All three problems aim to minimize the number of misclassified samples below (or above) a certain threshold. In the rest of the chapter, we focus on this common property. We show that all these problems fall into the following unified framework for binary classification at the top

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \lambda_1 \cdot \mathrm{fp}(\boldsymbol{s}, t) + \lambda_2 \cdot \mathrm{fn}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = G(\boldsymbol{s}, \boldsymbol{y}), \end{aligned} \tag{2.1}$$

where function $G\colon \mathbb{R}^n \times \{0,1\}^n \to \mathbb{R}$ takes the scores and labels of all samples and computes the decision threshold. The concrete form of the function G that defines the decision threshold depends on the used problem. As we show later in the chapter, all problems mentioned above differ only in the definition of the function G. Note the important distinction from the standard binary classification (1.5): the decision threshold is no longer fixed (as in the case of neural networks) or trained independently (as in SVM) but is a function of scores of all samples. Therefore, the minimization in problem (2.1) is performed only concerning the one variable $\boldsymbol{w}$.

2.1 Surrogate Formulation

The objective function of problem (2.1) is a weighted sum of false-positive and false-negative counts. Since these counts are discontinuous due to the presence of the Iverson function (see (1.3)), the whole objective function is discontinuous too. Therefore, problem (2.1) is difficult to solve. One way how to simplify the problem is to derive its continuous approximation. Since the only discontinuous part of the objective function is the Iverson function, the usual approach is to employ a surrogate function to replace it [22, 3].

Notation 2.1: Surrogate function

In the text below, the symbol l denotes any convex non-negative non-decreasing function with $l(0) = 1$, and $\lim_{s\to-\infty} l(s) = 0$. As examples of such function we can mention the hinge loss function or the quadratic hinge loss functions defined as follows

$$l_{\text{hinge}}(s) = \max\{0, 1+s\}, \qquad l_{\text{quadratic}}(s) = (\max\{0, 1+s\})^2.$$

We use surrogate functions to approximate the Iverson function, as shown in Figure 2.1.

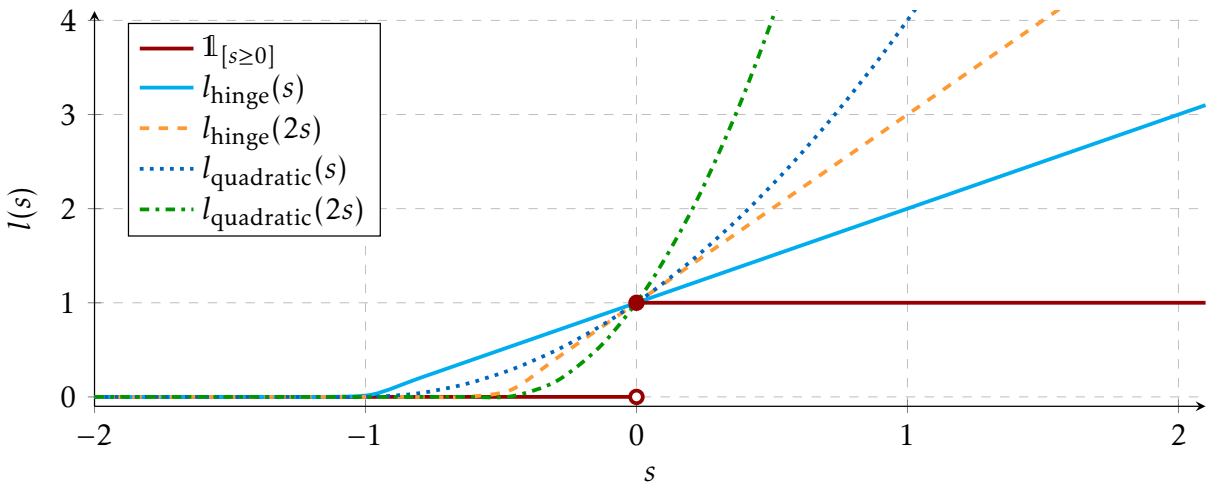


Figure 2.1: Comparison of the approximation quality of the Iverson function using different surrogate functions and scaling parameters.

Besides standard hinge and quadratic hinge function, Figure 2.1 also shows their version with scaled inputs $l_{\text{hinge}}(2s)$ and $l_{\text{quadratic}}(2s)$. We use $\vartheta > 0$ to denote any scaling parameter for surrogate functions in the text below.

Notation 2.1 summarizes all assumptions that a proper surrogate function must fulfill and introduces the two most often used surrogate functions: hinge and quadratic hinge loss functions. Moreover, Figure 2.1 compares these two surrogate functions with the Iverson function. It is clear that the surrogate function always provides an upper approximation of the Iverson function. In other words, if a surrogate function l satisfies assumptions from Notation 2.1, then $l(s) \geq \mathbb{1}_{[s\geq 0]}$ holds for any $s \in \mathbb{R}$. Besides that, Figure 2.1 shows how the scaling parameter ϑ affects the approximation quality of the surrogate function. If the scaling parameter is greater than 1, the surrogate function approximates the Iverson function better on interval $(-\infty, 0)$. In the opposite case, the approximation is better on interval $(0, \infty)$. The usual choice of scaling parameter is $\vartheta = 1$, and we used this choice for all surrogate functions used in the objective functions. However, we also use surrogate functions for approximation of the decision threshold. In such a case, the scaling parameter plays a crucial role for some theoretical guaranties, as shown in upcoming chapters.

With a properly defined surrogate function, we can define the surrogate approximation of the objective function of problem (2.1). To follow the notation from the previous chapter, we first replace the Iverson function in (2.1). Using any surrogate function l that satisfies assumptions from Notation 2.1, the true counts (2.1) may be approximated by their surrogate counterparts defined by

$$\begin{aligned} \overline{\text{tp}}(\boldsymbol{s}, t) &= \sum_{i \in \mathcal{I}_+} l(s_i - t), & \overline{\text{fn}}(\boldsymbol{s}, t) &= \sum_{i \in \mathcal{I}_+} l(t - s_i), \\ \overline{\text{tn}}(\boldsymbol{s}, t) &= \sum_{i \in \mathcal{I}_-} l(t - s_i), & \overline{\text{fp}}(\boldsymbol{s}, t) &= \sum_{i \in \mathcal{I}_-} l(s_i - t). \end{aligned} \tag{2.2}$$

Since the surrogate function provides upper approximation of the Iverson function, the surrogate counts (2.2) provide upper approximations of the true counts (1.3). By replacing the true counts in the objective function of (2.1) with their surrogate counterparts and adding a

regularization for better numerical stability, we get

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \lambda_1 \cdot \overline{\text{fp}}(\boldsymbol{s}, t) + \lambda_2 \cdot \overline{\text{fn}}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = G(\boldsymbol{s}, \boldsymbol{y}). \end{aligned} \tag{2.3}$$

The resulting objective function is continuous, and therefore the problem is easier to solve than the original problem (2.1). No additional theoretical properties can be derived without knowing the concrete form of model f and function G. Therefore, the rest of the chapter is dedicated to problems that fall into the general framework (2.3) and concrete form of G for such problems. More precisely, we focus on the three problems introduced in Section 1.2 and show how to rewrite them to our general formulation (2.3). Most of these problems are defined originally only for the linear model since this choice allows to derive nice theoretical properties and efficient solving algorithms. However, this chapter focuses on the problem formulation itself rather than on how to solve it. Therefore for all problems, we derive their version with general model f. The discussion of the theoretical properties for specific forms of f is provided in Chapter 3, 4, and 5.

Notation 2.2: Classification scores

In Notation 1.2, we defined vector $\boldsymbol{s} \in \mathbb{R}^n$ of scores of all samples with components defined for any $i \in \mathcal{I}$ as

$$s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I},$$

where $f : \mathbb{R}^d \to \mathbb{R}$ represents an arbitrary model. To simplify the upcoming sections, we define a sorted version of vector $\boldsymbol{s}$ with non-increasing components and denote it as $\boldsymbol{s}_{[\cdot]}$. It means that components of $\boldsymbol{s}_{[\cdot]}$ fulfill

$$s_{[1]} \geq s_{[2]} \geq \cdots \geq s_{[n-1]} \geq s_{[n]}.$$

Moreover, we denote negative samples as $\boldsymbol{x}^-$ and positive samples as $\boldsymbol{x}^+$. Finally, we define vectors $\boldsymbol{s}^- \in \mathbb{R}^{n_-}$, $\boldsymbol{s}^+ \in \mathbb{R}^{n_+}$ of scores of all positive and negative samples with components defined as

$$\begin{aligned} s_j^- &= f(\boldsymbol{x}_j^-; \boldsymbol{w}), \quad j = 1, 2, \ldots, n_-, \\ s_i^+ &= f(\boldsymbol{x}_i^+; \boldsymbol{w}), \quad i = 1, 2, \ldots, n_+, \end{aligned}$$

and their sorted versions $\boldsymbol{s}_{[\cdot]}^-$, $\boldsymbol{s}_{[\cdot]}^+$ with non-increasing components.

2.2 Ranking Problems

The first category of problems from Section 1.2 is a category of ranking problems. The general goal of problems from this category is to rank positive (relevant) samples higher than negative ones. That can be achieved in many different ways, but we focus only on the problems that concentrate on the high-ranked negative samples and try to push as many positive samples as possible above them. The simplest case is to maximize the number of positive samples above the worst negative. Since the worst negative sample is the negative sample with the highest classification score, the decision threshold for such a case is the highest score corresponding to the negative sample. Then the aim is to maximize the number of true-positive samples above this threshold or, equivalently, minimize the number of false-negative negative below it, which

may be written as

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{n_+}\operatorname{fn}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = s^-_{[1]}. \end{aligned} \tag{2.4}$$

Since the decision threshold t in the previous definition is computed from the sorted vector of negative scores $\boldsymbol{s}^-_{[\cdot]}$, it is a function of all negative scores. Therefore, formulation (2.4) is just a special case of the general formulation (2.1) for $\lambda_1 = 0$ and $\lambda_2 = 1/n_+$. The authors in [22] proposed an efficient method to solve formulation (2.4) and called it *TopPush*. They replaced the true counts in the objective function of (2.4) with its surrogate counterpart in the same way as we did in Section 2.1. The resulting formulation has the following form

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n_+}\overline{\operatorname{fn}}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = s^-_{[1]}, \end{aligned} \tag{2.5}$$

which again falls into our framework (2.3). To stress the origin of this formulation, we denote it as *TopPush*. Unfortunately, *TopPush* formulation can be very sensitive to outliers, especially when the linear model is used, as shown in Section 3.3. To robustify the formulation, we follow the idea presented in [28] and replace the highest negative score by the mean of K highest negative scores. The resulting formulation is as follows

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n_+}\overline{\operatorname{fn}}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = \frac{1}{K}\sum_{i=1}^{K} s^-_{[i]}. \end{aligned} \tag{2.6}$$

To emphasize the similarity with the *TopPush*, we call this formulation *TopPushK*. It is also possible to use the value of K-th highest negative score as the threshold. Such a choice may be advantageous in some cases, and we will discuss it in Chapter 5. For now, we will stick to the formulation that uses the mean since it will allow us to derive some theoretical properties in Section 3.1.

2.3 Accuracy at the Top

The second problem from Section 1.2 is the problem of Accuracy at the Top [23]. This problem aims to find an ordering of samples so that samples whose scores are among the top τ-quantile are as relevant as possible. The top τ-quantile of all scores is defined by

$$t = \max\left\{ t \;\middle|\; \frac{1}{n}\sum_{i \in \mathcal{I}} \mathbb{1}_{[s_i \geq t]} \geq \tau \right\}. \tag{2.7}$$

All relevant samples should be ranked above the quantile t and all irrelevant samples below the quantile t in an ideal case. Thus, the main difference to the ranking problems is that the problem of Accuracy at the Top considers both classification errors and does not focus only on false-negative samples. The original formulation [23] considers a balanced dataset with the

same number of positive and negative samples. Paper [3] reformulated the problem for the unbalanced dataset and derived the following formulation

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{n_-}\operatorname{fp}(\boldsymbol{s},t)+\frac{1}{n_+}\operatorname{fn}(\boldsymbol{s},t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i;\boldsymbol{w}), \quad i\in\mathcal{I}, \\ & t = \max\left\{t \;\middle|\; \frac{1}{n}\sum_{i\in\mathcal{I}} \mathbb{1}_{[s_i\geq t]} \geq \tau\right\}. \end{aligned} \tag{2.8}$$

This formulation already falls into our framework (2.1) for $\lambda_1 = 1/n_-$ and $\lambda_2 = 1/n_+$. Moreover, the authors of [23, 3] used the same surrogate trick to get rid of the discontinuous objective function, as we used in Section 2.1. Thus, by replacing false-positive and false-negative counts in the objective function with their surrogate counterparts we get

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2+\frac{1}{n_-}\overline{\operatorname{fp}}(\boldsymbol{s},t)+\frac{1}{n_+}\overline{\operatorname{fn}}(\boldsymbol{s},t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i;\boldsymbol{w}), \quad i\in\mathcal{I}, \\ & t = \max\left\{t \;\middle|\; \frac{1}{n}\sum_{i\in\mathcal{I}} \mathbb{1}_{[s_i\geq t]} \geq \tau\right\}. \end{aligned} \tag{2.9}$$

This formulation falls into our framework (2.3) for $\lambda_1 = 1/n_-$ and $\lambda_2 = 1/n_+$. Even though the original formulation is presented in [23], we denote the previous formulation as *Grill* based on the name of the first author of [3]. There are two reasons for that. The first one is that we used an unbalanced dataset as in [3]. The second one is that we use an algorithm proposed in [3] for numerical experiments since the one from [23] is suitable only for a small dataset.

The *Grill* formulation (2.9) is still challenging to solve due to the form of the decision threshold (2.7). The authors of [23] removed the necessity to handle the difficult quantile constraint by setting quantile as one of the samples and solving n independent problems. However, such an approach is infeasible for a large number of samples. The authors of [3] proposed the projected gradient descent method, where after each gradient step, the quantile is recomputed. This approach is suitable for large data but lacks theoretical guarantees. In the following text, we propose two approximations of the true quantile (2.7) that can be used to simplify formulation (2.9). The first one is a simple approximation by the mean of $n\tau$-th highest scores

$$t = \frac{1}{n\tau}\sum_{i=1}^{n\tau} s_{[i]}. \tag{2.10}$$

where for simplicity we assume, that $n\tau$ is an integer. The main purpose of (2.10) is to provide a convex approximation of the non-convex quantile (2.7). In fact, it is known that it is the tightest convex approximation of (2.7). Putting (2.10) into the constraint results in the following problem, which we call *TopMeanK*

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2+\frac{1}{n_+}\overline{\operatorname{fn}}(\boldsymbol{s},t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i;\boldsymbol{w}), \quad i\in\mathcal{I}, \\ & t = \frac{1}{K}\sum_{i=1}^{K} s_{[i]}, \end{aligned} \tag{2.11}$$

where $K = n\tau$. Besides changing the form of the decision threshold, we also simplified the objective function. This change allows preserving the convexity of the formulation for the

linear model as shown in Section 3.1. The resulting formulation is very similar to the *TopPushK* formulation from the previous section. The only difference is that the threshold for *TopMeanK* is computed from scores of all samples and not only from the negative ones.

The second option how to approximate the true quantile is to use surrogate counterparts to replace true counts in (2.7) and solve the following equality

$$t(\boldsymbol{s}) \quad \text{solves} \quad \frac{1}{n}\sum_{i\in\mathcal{I}} l(\vartheta(s_i - t)) = \tau, \tag{2.12}$$

where $\vartheta > 0$ is fixed scaling parameter. Since this threshold uses the surrogate approximation, we denote it as surrogate top τ-quantile. We get the following formulation by replacing the true quantile in the constrain and simplifying the objective function

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n_+}\overline{\text{fn}}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t \quad \text{solves} \quad \frac{1}{n}\sum_{i\in\mathcal{I}} l(\vartheta(s_i - t)) = \tau. \end{aligned} \tag{2.13}$$

This formulation also used only false negatives in the objective to preserve the convexity for the linear model. In such a case, the formulation is easily solvable due to the convexity and requires almost no tuning. Together with the fact that formulation (2.13) provides a good approximation to the Accuracy at the Top problem, we named it *Pat&Mat* (Precision At the Top & Mostly Automated Tuning).

2.4 Neyman-Pearson Problem

The last problem from Section 1.2 is the Neyman-Pearson problem, which is closely related to hypothesis testing. The hypothesis testing operates with null H_0 and alternative H_1 hypothesis. The goal is to either reject the null hypothesis in favor of the alternative or not. Since this problem is binary, two possible errors can occur. Type I occurs when H_0 is true but is rejected, and Type II error happens when H_0 is false but fails to be rejected. The Neyman-Pearson problem [27] minimizes Type II error while keeping Type I error smaller than some predefined bound. Using our notation for the Neyman-Pearson problem, the null hypothesis H_0 states that sample $\boldsymbol{x}$ has a negative label. Then Type I error occurs when the sample is false-positive, while Type II error when the sample is false-negative, see Table 1.1. In other words, Type II error corresponds to the false-negative rate, and Type I error to false-positive rate. Therefore, if the bound on the Type I error is τ, we may write this as

$$t^{\text{NP}} = \max\left\{ t \;\middle|\; \frac{1}{n_-}\sum_{i\in\mathcal{I}_-} \mathbb{1}_{[s_i \geq t]} \geq \tau \right\}. \tag{2.14}$$

Note that we only count the false-positive samples in (2.14) instead of counting all positives in (2.7). Then, we may write the Neyman-Pearson problem as

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{n_+}\text{fn}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = \max\left\{ t \;\middle|\; \frac{1}{n_-}\sum_{i\in\mathcal{I}_-} \mathbb{1}_{[s_i \geq t]} \geq \tau \right\}. \end{aligned} \tag{2.15}$$

This problem falls within our framework for (2.1) for $\lambda_1 = 0$ and $\lambda_2 = 1/n_+$. Moreover, formulation (2.15) differs from (2.8) by two things. The first one is the absence of a false-positive rate in the objective function. The second one is that the threshold is computed from negative samples only. Therefore, we can use the same techniques to approximate both objective function and the decision threshold.

To follow the previous section, we first derive the Neyman-Pearson alternative to the *Grill* formulation. We need to add false-positive counts in the objective function to do that. Moreover, we also need to replace true counts with their surrogate counterparts and add the regularization. The resulting formulation is as follows

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n_-}\overline{\text{fp}}(\boldsymbol{s}, t) + \frac{1}{n_+}\overline{\text{fn}}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = \max\left\{ t \;\middle|\; \frac{1}{n_-}\sum_{i \in \mathcal{I}_-} \mathbb{1}_{[s_i \geq t]} \geq \tau \right\}. \end{aligned} \tag{2.16}$$

We denote this formulation as *Grill-NP* to emphasize the relation with the original *Grill* formulation and the Neyman-Pearson problem.

The second formulation (2.11) from the previous section, uses mean of $n\tau$ highest scores to approximate true quantile (2.7). In the same way, we can approximate true quantile (2.14) by the mean of $n_-\tau$ highest of scores corresponding to the negative samples

$$t^{\text{NP}} = \frac{1}{n_-\tau}\sum_{i=1}^{n_-\tau} s^-_{[i]}. \tag{2.17}$$

For simplicity, we again assume that $n_-\tau$ is an integer. Putting (2.17) into the constraint results in the Neyman-Pearson alternative to *TopMeanK* defined as

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n_+}\overline{\text{fn}}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = \frac{1}{n_-\tau}\sum_{i=1}^{n_-\tau} s^-_{[i]}. \end{aligned} \tag{2.18}$$

This problem already appeared in [29] under the name τ-*FPL*. Formulation (2.18) has almost the same form as formulation (2.11). The only difference is that for τ-*FPL* we have $K = n_-\tau$ while for *TopPushK*, the value of K is small. Thus, even though we started from two different problems, we arrived at two approximations that differ only in the value of one parameter. This slight difference shows a close relationship between the ranking problems and the Neyman-Pearson problem and the need for a unified theory to handle these problems.

The last formulation (2.13) from the previous sections uses the surrogate approximation of the true quantile (2.7). The surrogate approximation of the true quantile (2.14) reads

$$t^{\text{NP}} \quad \text{solves} \quad \frac{1}{n_-}\sum_{i \in \mathcal{I}_-} l(\vartheta(s_i - t)) = \tau. \tag{2.19}$$

Putting (2.19) into the constraint results in the Neyman-Pearson alternative to *Pat&Mat* in the following form

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n_+}\overline{\text{fn}}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t \quad \text{solves} \quad \frac{1}{n_-}\sum_{i \in \mathcal{I}_-} l(\vartheta(s_i - t)) = \tau. \end{aligned} \tag{2.20}$$

We call this formulation *Pat&Mat-NP* to stress the similarity with *Pat&Mat*. The only difference between these two formulations is that only negative samples are involved in computing the decision threshold for *Pat&Mat-NP*, while *Pat&Mat* uses all samples.

2.5 Summary

In this chapter, we presented the general framework (2.1) and its surrogate approximation (2.3) to handle the problem of binary classification at the top. Moreover, we showed that many important problems might be formulated in a way that falls into the framework. In Table 2.1, we summarize all formulations introduced in this chapter and show their relation to the general framework (2.3). All these formulations were derived with general model f even though many of them have been initially proposed only for the linear model. The reason for that is simple. This chapter aims only to emphasize the similarities between these formulations. The theoretical properties that follow from the concrete form of the model are discussed in the upcoming chapters. More precisely, the rest of the text is organized as follows.

- Chapter 3 is dedicated to the linear model and formulations in their primal form, i.e., in the form presented in this chapter. This chapter shows that some formulations have nice properties such as convexity, differentiability, or stability. We derive some theoretical properties for the optimal solution based on these properties.
- Since many problems are not linearly separable, Chapter 4 is dedicated to the dual forms of formulations from Table 2.1. In this chapter, we again assume a linear model only. In the first part of the chapter, we show that all formulations can be split into two families based on their similarities. Then we derive dual formulations for these two families and show that these formulations are very similar to standard SVM. Using this observation, we use kernel trick to employ non-linearity into the formulations. Finally, we derive an efficient algorithm for solving the formulations.
- In Chapter 5, we assume a nonlinear model. A prototypical example of such a model can be a neural network. The resulting formulations are not decomposable since the decision threshold is always a function of all classification scores. Therefore, it is impossible to use the stochastic descent algorithm directly to solve them. In Chapter 5, we present two approaches to deal with this problem.
- Chapter 6 dedicated to all numerical experiments.
- Chapter 7 summarizes all results presented in this work.

Moreover, we postpone all proofs and additional results into the appendices for better readability.

Formulation	Label	Source	Ours	λ_1	λ_2	Threshold
TopPush	(2.5)	[22]	✗	0	$\frac{1}{n_+}$	$s^-_{[1]}$
TopPushK	(2.6)	[30]	✓	0	$\frac{1}{n_+}$	$\frac{1}{K}\sum_{i=1}^{K} s^-_{[i]}$
Grill	(2.9)	[3]	✗	$\frac{1}{n_-}$	$\frac{1}{n_+}$	$\max\left\{t \,\middle|\, \frac{1}{n}\sum_{i\in\mathcal{I}} \mathbb{1}_{[s_i\geq t]} \geq \tau\right\}$
TopMeanK	(2.11)	—	✗	0	$\frac{1}{n_+}$	$\frac{1}{K}\sum_{i=1}^{K} s_{[i]}$
Pat&Mat	(2.13)	[30]	✓	0	$\frac{1}{n_+}$	$\frac{1}{n}\sum_{i\in\mathcal{I}} l(\vartheta(s_i - t)) = \tau$
Grill-NP	(2.16)	—	✗	$\frac{1}{n_-}$	$\frac{1}{n_+}$	$\max\left\{t \,\middle|\, \frac{1}{n_-}\sum_{i\in\mathcal{I}_-} \mathbb{1}_{[s_i\geq t]} \geq \tau\right\}$
τ-FPL	(2.18)	[29]	✗	0	$\frac{1}{n_+}$	$\frac{1}{n_-\tau}\sum_{i=1}^{n_-\tau} s^-_{[i]}$
Pat&Mat-NP	(2.20)	[30]	✓	0	$\frac{1}{n_+}$	$\frac{1}{n_-}\sum_{i\in\mathcal{I}_-} l(\vartheta(s_i - t)) = \tau$

Table 2.1: Summary of problem fomrulations that fall in the framework (2.3). Column **Formulation** shows the name of the formulation that we use in this work. Column **Label** represents the label of the formulation in this text. Column **Source** is the citation of the work where the formulation was introduced. Column **Ours** shows whether the formulation was introduced in any of our previous papers. The last three columns show the values of parameters λ_1, λ_2 and the form of the decision threshold for given framework (2.3).

3

Primal Formulation: Linear Model

In the previous chapter, we introduced the general framework for binary classification at the top. Table 2.1 summarizes all formulations that fall into this framework. In this chapter, we focus on the particular case when the model f is linear, i.e., the model is in the following form

$$f(\boldsymbol{x};\boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{x},$$

where $\boldsymbol{w} \in \mathbb{R}^d$ is the normal vector to the separating hyperplane. In such a case, framework (2.3) simplifies into the form below

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \lambda_1 \cdot \overline{\text{fp}}(\boldsymbol{s},t) + \lambda_2 \cdot \overline{\text{fn}}(\boldsymbol{s},t) \\ \text{subject to} \quad & s_i = \boldsymbol{w}^\top \boldsymbol{x}_i, \quad i \in \mathcal{I}, \\ & t = G(\boldsymbol{s},\boldsymbol{y}). \end{aligned}$$

In the upcoming sections, we provide a theoretical analysis of this unified framework using linear model. We consider the problem formulations from Chapter 2 and not individual algorithms which specify how to solve these formulations. The theoretical properties we mainly focus on are as follows:

- *Convexity* implies a guaranteed convergence for many optimization algorithms or their better convergence rates [31].
- *Differentiability* increases the speed of convergence.
- *Stability* is a general term, by which we mean that the global minimum is not at $\boldsymbol{w} = \boldsymbol{0}$. This actually is the case for many formulations from Table 2.1 and results in the situation where the separating hyperplane is degenerate and does not actually exist.

We show the results only for formulations from Section 2.2 and 2.3 for better readability. Formulations in Section 2.3 are mostly identical to the ones from Section 2.4. The only difference is that all formulations in Section 2.3 compute the decision threshold from all samples, while formulations in Section 2.4 use only negative samples. Therefore, the results for both sections are identical, and we show only the ones for Section 2.3.

3.1 Convexity

Convexity is one of the most important properties in numerical optimization. It ensures that the optimization problem has neither stationary points nor local minima. All points of interest are global minima. Moreover, it allows for faster convergence rates. This section shows that some of the formulations from Table 2.1 are convex and, therefore, easier to solve. The first result is summarized in the following proposition. Note that we denote the thresholds as functions of weights $\boldsymbol{w}$. This dependence holds since the thresholds are defined in Section 2.3 as functions of scores $\boldsymbol{s}$.

Proposition 3.1

Consider vector of scores $\boldsymbol{s}$ with elements defined as $s_i = \boldsymbol{w}^\top \boldsymbol{x}_i$ for all $i \in \mathcal{I}$ and Notation 2.2. Recall the decision thresholds from Section 2.2 and 2.3

$$t_0(\boldsymbol{w}) = s_{[1]}^-, \qquad t_1(\boldsymbol{w}) = \max\left\{ t \;\middle|\; \frac{1}{n}\sum_{i\in\mathcal{I}} \mathbb{1}_{[s_i \geq t]} \geq \tau \right\},$$

$$t_2(\boldsymbol{w}) = \frac{1}{K}\sum_{i=1}^{K} s_{[i]}, \qquad t_3(\boldsymbol{w}) \quad \text{solves} \quad \frac{1}{n}\sum_{i\in\mathcal{I}} l(\vartheta(s_i - t)) = \tau,$$

Then thresholds t_0, t_2 and t_3 are convex functions of weights $\boldsymbol{w}$, while the threshold t_1 is non-convex.

The proposition says that *Grill* formulation uses non-convex threshold while *TopPush*, *TopMeanK*, and *Pat&Mat* use the convex ones. Moreover, the thresholds for τ-*FPL* and *TopPushK* are convex since both formulations use almost the same threshold as *TopMeanK*. The same holds for the thresholds of *Pat&Mat* and *Pat&Mat-NP* formulations. Notice that all formulations that have a convex threshold use the same objective function.

Theorem 3.2

If the threshold $t = t(\boldsymbol{w})$ is a convex function of weights $\boldsymbol{w}$, then function

$$L(\boldsymbol{w}) = \overline{\text{fn}}(\boldsymbol{s}, t)$$

is convex.

While the proof of Theorem 3.2 is simple, it points to the necessity of considering only false-negatives in the objective function. Due to this theorem, almost all formulations from Table 2.1 are convex optimization problems. There are only two exceptions: *Grill* and *Grill-NP* are not convex problems.

3.2 Differentiability

Similar to convexity, differentiability is crucial for improving the convergence rate. Moreover, differentiability can often be used to derive better termination criteria for numerical algorithms. The next theorem shows which formulations from Table 2.1 are differentiable.

Theorem 3.3

Consider thresholds from Proposition 3.1. Threshold t_0, t_1 and t_2 are non-differentiable functions of weights $\boldsymbol{w}$. Moreover, if the surrogate function l is differentiable, threshold t_3 is a differentiable function of weights $\boldsymbol{w}$, and its derivative equals

$$\nabla t_3(\boldsymbol{w}) = \frac{\sum_{i\in\mathcal{I}} l'(\vartheta(\boldsymbol{w}^\top \boldsymbol{x}_i - t_3(\boldsymbol{w})))\boldsymbol{x}_i}{\sum_{j\in\mathcal{I}} l'\Big(\vartheta(\boldsymbol{w}^\top \boldsymbol{x}_j - t_3(\boldsymbol{w}))\Big)}.$$

Due to the previous theorem and Theorem 3.2, only *Pat&Mat*, and *Pat&Mat-NP* are convex and differentiable optimization problems. These properties allow us to prove the convergence of the stochastic gradient descent for these two formulations, as shown in Section 3.4.

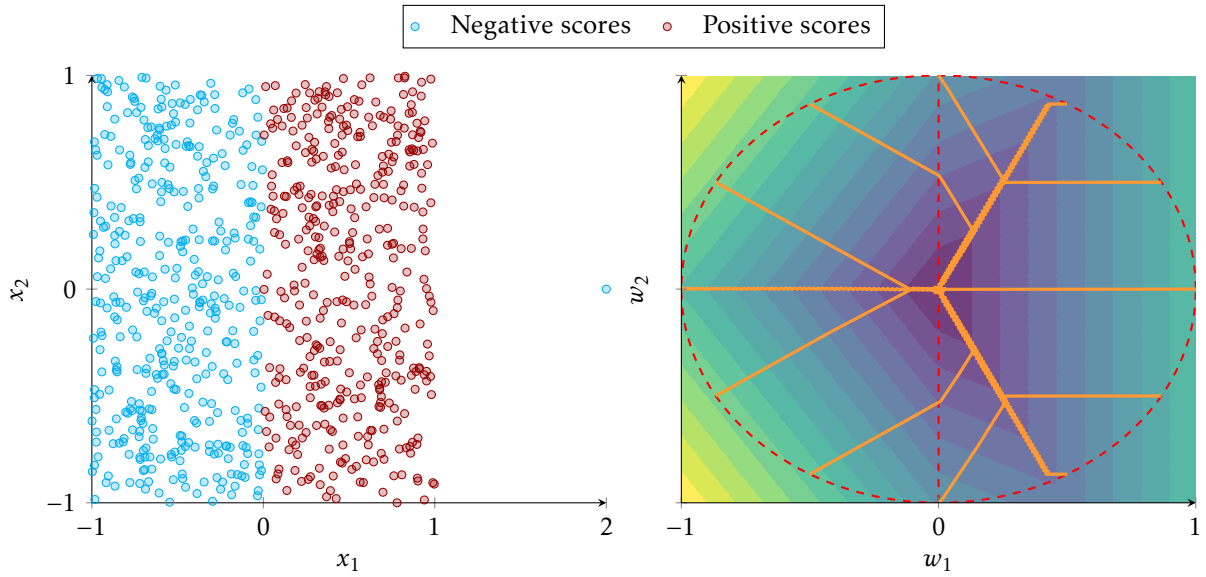


Figure 3.1: Distribution of positive (red circles) and negative samples (blue circles) for the example from Example 3.4. (**left**) Contour plot of the objective function value for *TopPush* with hinge loss as a surrogate and no regularization and its convergence (orange lines) to the zero vector from 12 different initial points. (**right**)

3.3 Stability

We first provide a simple example and show that many formulations from Table 2.1 are degenerate for it. Then we analyze general conditions under which this degenerate behavior happens.

Example 3.4: Degenerate Behaviour

Consider n negative samples uniformly distributed in $[-1,0]\times[-1,1]$, n positive samples uniformly distributed in $[0,1]\times[-1,1]$ and one negative sample at $(2,0)$. An illustration of such settings is provided in Figure 3.1 (**left**). If n is large enough, the point at $(2,0)$, is an outlier and the problem is (almost) perfectly separable using the separating hyperplane with normal vector $\boldsymbol{w}_1 = (1,0)$.

There are two important solutions for Example 3.4. The first is the optimal solution $\boldsymbol{w}_1 = (1,0)$, which generates the optimal separating hyperplane. The second is $\boldsymbol{w}_0 = (0,0)$, a degenerate solution that does not generate any separating hyperplane. Since the dataset is perfectly separable by $\boldsymbol{w}_1$, we expect that $\boldsymbol{w}_1$ provides a lower value of the objective function than $\boldsymbol{w}_0$ for all formulations from Table 2.1. However, it is not happening. Table 3.1 shows the threshold t and the value of the objective function L for $\boldsymbol{w}_0$ and $\boldsymbol{w}_1$. For the precise computation of the results, see Appendix A.3. By highlighting the better objective in Table 3.1 by green, we see that *TopPush* and *TopMeanK* has a better objective in $\boldsymbol{w}_0$. It can be shown that $\boldsymbol{w}_0$ is even the global minimum for these two formulations. This situation raises the question whether some tricks, such as early stopping or excluding a small ball around zero, cannot overcome this difficulty. The answer is negative, as shown in Figure 3.1 (**right**). Here, we run *TopPush* with hinge loss as a surrogate and no regularization from several starting points. In all cases, *TopPush* converges to zero from one of the three possible directions, and all these directions are far from the normal vector to the separating hyperplane.

The convexity derived in the previous section guarantees that there are no local minima. However, as we showed in the example above, the global minimum may be at $\boldsymbol{w} = \mathbf{0}$. Such

Formulation	Label	$\boldsymbol{w}_0 = (0,0)$		$\boldsymbol{w}_1 = (1,0)$	
		t	L	t	L
TopPush	(2.5)	0	1	2	2.5
TopPushK	(2.6)	0	1	$\frac{2}{K}$	$0.5 + \frac{2}{K}$
Grill	(2.9)	0	2	$1 - 2\tau$	$1.5 + 2\tau(1-\tau)$
TopMeanK	(2.11)	0	1	$1 - \tau$	$1.5 - \tau$
Pat&Mat	(2.13)	$\frac{1}{\vartheta}(1-\tau)$	$1 + \frac{1}{\vartheta}(1-\tau)$	$\frac{1}{\vartheta}(1-\tau)$	$0.5 + \frac{1}{\vartheta}(1-\tau)$
Grill-NP	(2.16)	0	2	$-\tau$	???
τ-FPL	(2.18)	0	1	$\frac{2}{\tau n_-}$	$0.5 + \frac{2}{\tau n_-}$
Pat&Mat-NP	(2.20)	$\frac{1}{\vartheta}(1-\tau)$	$1 + \frac{1}{\vartheta}(1-\tau)$	$\frac{1}{\vartheta}(1-\tau) - 0.5$	$\frac{1}{\vartheta}(1-\tau)$

Table 3.1: Comparison of formulations from Table 2.1 on the problem from Example 3.4. The table shows the threshold and the objective function value for two solutions: the optimal solution $\boldsymbol{w}_1 = (1,0)$ and degenerate solution $\boldsymbol{w}_0 = (0,0)$. Three formulations have the global minimum (denoted by green color) at $\boldsymbol{w}_0$, which does not generate any separating hyperplane.

a situation is highly undesirable since $\boldsymbol{w}$ is the normal vector to the separating hyperplane, and the zero vector provides no information. In the rest of the section, we analyze when this situation happens. Theorem 3.5 states that if the decision threshold $t = t(\boldsymbol{w})$ is above a certain value, then $\mathbf{0}$ has a better (lower) objective than $\boldsymbol{w}$. If this happens for all $\boldsymbol{w}$, then $\mathbf{0}$ is the global minimum.

Theorem 3.5

Consider any of these formulations: *TopPush*, *TopPushK*, *TopMeanK* or *τ-FPL*. Fix any $\boldsymbol{w}$ and denote the corresponding objective function $L(\boldsymbol{w})$ and threshold $t(\boldsymbol{w})$. If we have

$$t(\boldsymbol{w}) \geq \frac{1}{n_+} \sum_{i \in \mathcal{I}_+} \boldsymbol{w}^\top \boldsymbol{x}_i, \tag{3.1}$$

then $L(\mathbf{0}) \leq L(\boldsymbol{w})$. Specifically, using Notation 2.2 we get the following implications

$$\begin{aligned}
s^-_{[1]} \geq \frac{1}{n_+} \sum_{i=1}^{n_+} s^+_i \quad &\Longrightarrow \quad L(\mathbf{0}) \leq L(\boldsymbol{w}) \text{ for } \textit{TopPush}, \\
\frac{1}{K} \sum_{i=1}^{K} s^-_{[i]} \geq \frac{1}{n_+} \sum_{i=1}^{n_+} s^+_i \quad &\Longrightarrow \quad L(\mathbf{0}) \leq L(\boldsymbol{w}) \text{ for } \textit{TopPushK}, \\
\frac{1}{K} \sum_{i=1}^{K} s_{[i]} \geq \frac{1}{n_+} \sum_{i=1}^{n_+} s^+_i \quad &\Longrightarrow \quad L(\mathbf{0}) \leq L(\boldsymbol{w}) \text{ for } \textit{TopMeanK}, \\
\frac{1}{n_-\tau} \sum_{i=1}^{n_-\tau} s^-_{[i]} \geq \frac{1}{n_+} \sum_{i=1}^{n_+} s^+_i \quad &\Longrightarrow \quad L(\mathbf{0}) \leq L(\boldsymbol{w}) \text{ for } \tau\textit{-FPL}.
\end{aligned}$$

The proof of Theorem 3.5 employs the fact that all formulations in the theorem statement have only false-negatives in the objective. If we use the zero solution $\boldsymbol{w}_0 = \mathbf{0}$, all classification scores s_i are equal to zero, the threshold t equals zero, and the objective function L equals one. On the other hand, if the threshold t is large, many positive samples have scores below the threshold, and the false-negatives samples have the average surrogate value larger than one. In such a case, $\boldsymbol{w}_0 = \mathbf{0}$ becomes the global minimum for some formulations. More specifically, *TopPush* fails if there are outliers, and *TopMeanK* fails whenever there are many positive samples.

Corollary 3.6

Consider the *TopPush* formulation. If positive samples lie in the convex hull of negative samples, then $\boldsymbol{w} = \mathbf{0}$ is the global minimum.

Corollary 3.7

Consider the *TopMeanK* formulation. If $n_+ \geq n\tau$, then $\boldsymbol{w} = \mathbf{0}$ is the global minimum.

There are two fixes to the situation described above:

- Include false-positives to the objective. This approach is taken by *Grill* and *Grill-NP* and necessarily results in the loss of convexity as shown in Section 3.1.
- Move the threshold away from zero even when all scores $\boldsymbol{s}$ are zero. This approach is taken by our formulations *Pat&Mat* and *Pat&Mat-NP* and keeps convexity.

The following theorem shows the advantage of the second approach.

Theorem 3.8

Consider the *Pat&Mat* or *Pat&Mat-NP* formulation with the hinge loss as a surrogate and no regularization. Assume that for some $\boldsymbol{w}$ we have

$$\frac{1}{n_+} \sum_{i \in \mathcal{I}_+} \boldsymbol{w}^\top \boldsymbol{x}_i > \frac{1}{n_-} \sum_{j \in \mathcal{I}_-} \boldsymbol{w}^\top \boldsymbol{x}_j. \tag{3.2}$$

Then there exists a scaling parameter ϑ_0 for the surrogate top τ-quantile (2.12) or (2.19) such that $L(\boldsymbol{w}) < L(\mathbf{0})$ for all $\vartheta \in (0, \vartheta_0)$.

This theorem shed some light on the behavior of the formulations. Theorem 3.5 states that the stability of τ-*FPL* requires

$$\frac{1}{n_- \tau} \sum_{i=1}^{n_- \tau} s^-_{[i]} < \frac{1}{n_+} \sum_{i=1}^{n_+} s^+_i, \tag{3.3}$$

while Theorem 3.8 states that the stability of *Pat&Mat-NP* is ensured by

$$\frac{1}{n_-} \sum_{i=1}^{n_-} s^-_{[i]} < \frac{1}{n_+} \sum_{i=1}^{n_+} s^+_i. \tag{3.4}$$

The right-hand sides of (3.3) and (3.4) are the same, while the left-hand side of (3.4) is always smaller than the left-hand side of (3.3). This implies that if τ-*FPL* is stable, then *Pat&Mat-NP* is stable as well.

At the same time, there may be a considerable difference in the stability of both formulations. Since the scores of positive samples should be above the scores of negative samples, the scores $\boldsymbol{s}$ may be interpreted as performance. Then formula (3.3) states that if the mean performance of a small number of the worst negative samples is larger than the average performance of all positive samples, then τ-*FPL* fails. On the other hand, formula (3.4) states that if the average performance of all positive samples is better than the average performance of all negative samples, then *Pat&Mat-NP* is stable. The former may well happen as accuracy at the top is interested in a good performance of only a few positive samples.

3.4 Stochastic Gradient Descent

In the previous section, we analyzed the formulations from Table 2.1, but we did not consider any optimization algorithms. In this section we show a basic version of the stochastic gradient descent and then its convergent version. Due to considering the threshold, the gradient computed on a minibatch is a biased estimate of the true gradient. Therefore we need to use variance reduction techniques similar to SAG [32], and the proof is rather complex.

Many optimization algorithms for solving the formulations from Table 2.1 use primal-dual or purely dual formulations. Authors of [25] introduced dual variables and used alternating optimization to the resulting min-max problem. In [22, 29], authors dualized the problem and solved it with the steepest gradient ascent. Authors of [33] followed the same path but added kernels to handle non-linearity. We follow the ideas of [34] and [35] and solve the problems directly in their primal formulations. Therefore, even though we use the same formulation for *TopPush* as [22] or for τ-*FPL* as [29], our solution process is different. However, both algorithms should converge to the same point due to convexity.

The decision variables in (2.3) are the normal vector of the separating hyperplane $\boldsymbol{w}$ and the threshold t. Since the gradient descent algorithm uses the following rule

$$\boldsymbol{w}^{k+1} \leftarrow \boldsymbol{w}^k - \alpha^k \cdot \nabla L(\boldsymbol{w}^k),$$

where $k \in \mathbb{N}$ denotes iteration, α^k is a step size, and ∇L is a gradient of the objective function. Therefore, we need to compute gradients. The simplest idea [3] is to compute the gradient only with respect to $\boldsymbol{w}$ and then recompute t. A more sophisticated way is based on the chain rule. For each $\boldsymbol{w}$, the threshold t can be computed uniquely. We stress this dependence by writing $t(\boldsymbol{w})$ instead of t. By doing so, we effectively remove the threshold t from the decision variables and $\boldsymbol{w}$ remains the only decision variable. Note that the convexity is preserved. Then we can compute the derivative via the chain rule

$$\begin{aligned} L(\boldsymbol{w}) &= \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} l(t(\boldsymbol{w}) - \boldsymbol{w}^\top \boldsymbol{x}_i), \\ \nabla L(\boldsymbol{w}) &= \boldsymbol{w} + \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} l'(t(\boldsymbol{w}) - \boldsymbol{w}^\top \boldsymbol{x}_i)(\nabla t(\boldsymbol{w}) - \boldsymbol{x}_i). \end{aligned} \tag{3.5}$$

The last part is the computation of $\nabla t(\boldsymbol{w})$. It is simple for most of the formulations from Table 2.1. Moreover, Theorem 3.3 shows the computation for *Pat&Mat* with efficient computation method presented in Appendix A.5. Since we derive the gradient for the objective function in (3.5), it is easy to derive how to apply the stochastic gradient descent. We only have to partition the dataset into minibatches and provide an update of the weights $\boldsymbol{w}$ based only on a minibatch, namely by replacing the mean over the whole dataset in (3.5) by a mean over the minibatch.

For the convergence proof of stochastic gradient descent, we need differentiability. Due to Theorem 3.3, we have only two formulations that are differentiable: *Pat&Mat* and *Pat&Mat-NP*. Therefore, in the rest of the section, we consider only these two formulations. Moreover,

for simplicity, we show the proof only for *Pat&Mat*. At iteration k we have the decision variable $\boldsymbol{w}^k$ and the active minibatch $\mathcal{I}^k_{\text{mb}}$. First, we update the vector of scores $\boldsymbol{s}^k$ only on the active minibatch by setting

$$s_i^k = \begin{cases} \boldsymbol{x}_i^\top \boldsymbol{w}^k & \text{for all } i \in \mathcal{I}^k_{\text{mb}}, \\ s_i^{k-1} & \text{otherwise.} \end{cases} \tag{3.6}$$

We keep scores from previous minibatches intact. We use Appendix A.5 to compute the surrogate quantile t^k as the unique solution of

$$\sum_{i \in \mathcal{I}} l\Big(\vartheta(s_i^k - t^k)\Big) = n\tau. \tag{3.7}$$

This is an approximation of the surrogate quantile $t(\boldsymbol{w}^k)$ from (2.12). The only difference from the true quantile is that we use delayed scores. Then we introduce artificial variable

$$\boldsymbol{a}^k = \sum_{i \in \mathcal{I}^k_{\text{mb}}} l'\Big(\vartheta(s_i^k - t^k)\Big)\boldsymbol{x}_i. \tag{3.8}$$

Finally, we approximate the derivative $\nabla f(\boldsymbol{w}^k)$ from (3.5) by

$$g(\boldsymbol{w}^k) = \frac{1}{n^k_{\text{mb},+}} \sum_{i \in \mathcal{I}^k_{\text{mb},+}} l'(t^k - s_i^k)(\nabla t^k - \boldsymbol{x}_i), \tag{3.9}$$

where ∇t^k is an approximation of $\nabla t(\boldsymbol{w}^k)$ from Theorem 3.3 defined by

$$\nabla t^k = \frac{\boldsymbol{a}^k + \boldsymbol{a}^{k-1} + \cdots + \boldsymbol{a}^{k-m+1}}{\sum_{i \in \mathcal{I}} l'\Big(\vartheta(s_i^k - t^k)\Big)}. \tag{3.10}$$

A perhaps more straightforward possibility would be to consider only $\boldsymbol{a}^k$ in the numerator of (3.10). However, presented choice enables us to prove the convergence, and it adds stability to the algorithm for small minibatches. The whole procedure is summarized in Algorithm 2. Note that there are no vector operations outside of the current minibatch $\mathcal{I}^k_{\text{mb}}$. Moreover, note that a proper initialization for the first m iterations is needed.

Theorem 3.9

Consider the *Pat&Mat* or *Pat&Mat-NP* formulation, stepsizes $\alpha^k = \alpha^0/k+1$ and piecewise disjoint minibatches $\mathcal{I}^1_{\text{mb}}, \ldots, \mathcal{I}^m_{\text{mb}}$ which cycle periodically $\mathcal{I}^{k+m}_{\text{mb}} = \mathcal{I}^k_{\text{mb}}$. If l is the smoothened (Huberized) hinge function, then Algorithm 2 converges to the global minimum of (2.13).

Algorithm 2 Stochastic gradient descent for *Pat&Mat* formulation

Require: Dataset $\mathcal{D}$, minibatches $\mathcal{I}^1_{\text{mb}}$, $\mathcal{I}^2_{\text{mb}}$,..., $\mathcal{I}^m_{\text{mb}}$, and stepsize α^k

Initialize weights $\boldsymbol{w}^0$
for $k = 0, 1, \ldots$ **do**
    Select a minibatch $\mathcal{I}^k_{\text{mb}}$
    Compute s^k_i for all $i \in \mathcal{I}^k_{\text{mb}}$ according to (3.6)
    Compute t^k according to (3.7)
    Compute $\boldsymbol{a}^k$ according to (3.8)
    Compute ∇t^k according to (3.10)
    Compute $g(\boldsymbol{w}^k)$ according to (3.9)
    Set $\boldsymbol{w}^{k+1} \leftarrow \boldsymbol{w}^k - \alpha^k g(\boldsymbol{w}^k)$
end for

3.5 Summary

In this chapter, we derived theoretical properties for formulations from Table 2.1 with the linear model. We focused on the convexity, differentiability, and stability of formulations since these three properties are crucial for fast and proper convergence. All results are summarized in Table 3.2. We showed that *TopPush*, *TopPushK*, *TopMeanK*, and τ-*FPL* are convex, but all these formulations are vulnerable to having the global minimum at $\boldsymbol{w} = 0$. On the other hand, *Grill* and *Grill-NP* are stable, but they are not convex or differentiable. Finally, our formulations *Pat&Mat* and *Pat&Mat-NP* satisfy all three theoretical properties.

A similar comparison is performed in Figure 3.2. Methods in green and yellow are convex, while formulations in red are non-convex. Based on Theorem 3.5, four formulations in yellow are vulnerable to having the global minimum at $\boldsymbol{w} = 0$. This theorem states that the higher the threshold, the more vulnerable the formulation is. The full arrows depict this dependence. If it points from one formulation to another, the latter one has a smaller threshold and thus is less vulnerable to this undesired global minima. The dotted arrows indicate that this usually holds but not always. The precise formulation is provided in Appendix A.4. This complies with Corollaries 3.6 and 3.7 which state that *TopPush* and *TopMeanK* are most vulnerable. At the same time, it says that τ-*FPL* is the best one from the yellow formulations. Finally, even though *Pat&Mat-NP* has a worse approximation of the true threshold than τ-*FPL* due to Theorem 3.5, it is more stable due to the discussion after Theorem 3.8.

Formulation	Label	Hyperparameters	Convex	Differentiable	Stable
TopPush	(2.5)	—	✓	✗	✗
TopPushK	(2.6)	K	✓	✗	✗
Grill	(2.9)	τ	✗	✗	✓
TopMeanK	(2.11)	τ	✓	✗	✗
Pat&Mat	(2.13)	τ, ϑ	✓	✓	✓
Grill-NP	(2.16)	τ	✗	✗	✓
τ-FPL	(2.18)	τ	✓	✗	✗
Pat&Mat-NP	(2.20)	τ, ϑ	✓	✓	✓

Table 3.2: Summary of the formulations from Table 2.1. The second column shows the hyperparameters available for each formulation. The last three columns show whether the formulation is differentiable, convex, and stable (in the sense of having global minimum in $\boldsymbol{w} = \mathbf{0}$).

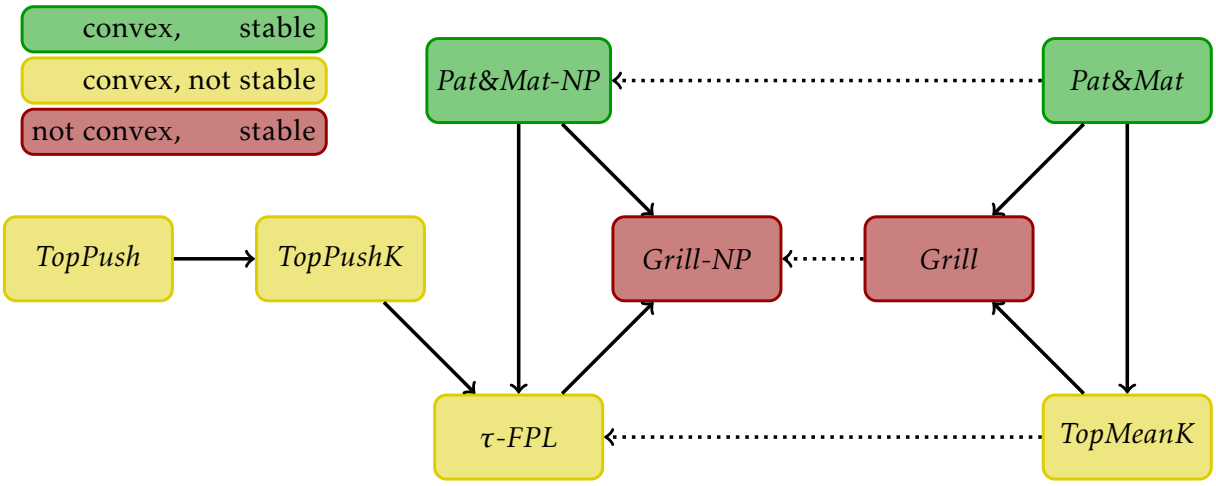


Figure 3.2: Summary of the formulations from Table 2.1. Methods in green and yellow are convex, while formulations in red are non-convex. Moreover, methods in yellow are vulnerable to having the global minimum at $\boldsymbol{w} = 0$. A full (dotted) arrow pointing from one formulation to another shows that the latter formulation has (usually) a smaller threshold.

4

Dual Formulation: Linear Model

In Chapter 2, we introduced a general framework for binary classification at the top. Moreover, we showed that several problem classes, considered separate problems so far, fit into this framework. The summary of all formulations is provided in Table 2.1. In Chapter 3 we discussed a special case when the linear model is used. Then formulation (2.3) reads

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \lambda_1 \cdot \overline{\text{fp}}(\boldsymbol{s}, t) + \lambda_2 \cdot \overline{\text{fn}}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = \boldsymbol{w}^\top \boldsymbol{x}_i, \quad i \in \mathcal{I}, \\ & t = G(\boldsymbol{s}, \boldsymbol{y}). \end{aligned} \tag{4.1}$$

Many formulations have nice theoretical properties such as convexity or differentiability in this specific case. However, many real-world problems are not linearly separable, and in such cases, the approach from the previous section is not sufficient. In this chapter, we use the similarity of (4.1) to primal formulation of SVM [21] and derive dual forms for all formulations from Table 2.1. Then we use the kernel method [36] to introduce nonlinearity into the dual formulations. Moreover, as dual problems are generally computationally expensive, we derive an efficient method to solve them.

4.1 Derivation of Dual Problems

As discussed in the introduction, this section is dedicated to deriving dual forms for all formulations from Table 2.1. *Grill* and *Grill-NP* formulations aren't used in the rest of the text since they both use true quantile as a threshold. Since many of the remaining formulations are very similar, we divide them into two families:

- ***TopPushK* family:** *TopPush*, *TopPushK*, *TopMeanK* and τ*-FPL*.
- ***Pat&Mat* family:** *Pat&Mat* and *Pat&Mat-NP*.

Both families use surrogate false-negative rate as an objective function. Moreover, all formulations from *TopPushK* family use the mean of K highest scores of all or negative samples as a threshold and differ only in the definition of K. Finally, both formulations from *Pat&Mat* family use a surrogate approximation of the top τ-quantile of scores of all or negative samples. In other words, we have two families of formulations that share the same objective function and the same form of the decision threshold. Therefore, we derive all results for the general form of these two families. Before we start, we need to introduce the concept of conjugate functions.

Definition 4.1: Conjugate function [31]

Let $l\colon \mathbb{R}^n \to \mathbb{R}$. The function $l^\star\colon \mathbb{R}^n \to \mathbb{R}$, defined as

$$l^\star(\boldsymbol{y}) = \sup_{\boldsymbol{x}\in \operatorname{dom} l} \{\boldsymbol{y}^\top \boldsymbol{x} - l(\boldsymbol{x})\}.$$

is called conjugate function of l. The domain of the conjugate function consists of $y \in \mathbb{R}$ for which the supremum is finite.

These functions will play a crucial role in the resulting form of dual problems. Recall the hinge loss and quadratic hinge loss function defined in Notation 2.1

$$l_{\text{hinge}}(s) = \max\{0, 1+s\}, \qquad l_{\text{quadratic}}(s) = (\max\{0, 1+s\})^2.$$

The conjugate function for the hinge loss can be found in [37] and has the following form

$$l^\star_{\text{hinge}}(y) = \begin{cases} -y & \text{if } y \in [0,1], \\ \infty & \text{otherwise.} \end{cases} \tag{4.2}$$

Similarly, the conjugate function for the quadratic hinge is defined in [38] as

$$l^\star_{\text{quadratic}}(y) = \begin{cases} \frac{y^2}{4} - y & \text{if } y \geq 0, \\ \infty & \text{otherwise.} \end{cases} \tag{4.3}$$

Notation 4.2: Kernel Matrix

To simplify the future notation, we introduce matrix $\mathbb{X}$ of all samples. Each row of $\mathbb{X}$ represents one sample and is defined for all $i \in \mathcal{I}$ as

$$\mathbb{X}_{i,\bullet} = \boldsymbol{x}_i^\top.$$

In the same way, we defined matrices $\mathbb{X}^+$, $\mathbb{X}^-$ of all negative and positive samples with rows defined as

$$\mathbb{X}^-_{i,\bullet} = \boldsymbol{x}_i^\top \quad i = 1, , 2, \ldots, n^-,$$
$$\mathbb{X}^+_{i,\bullet} = \boldsymbol{x}_i^\top \quad i = 1, , 2, \ldots, n^+.$$

Moreover, for all formulations that use only negative samples to compute the threshold t, we define kernel matrix $\mathbb{K}^-$ as

$$\mathbb{K}^- = \begin{pmatrix} \mathbb{X}^+ \\ -\mathbb{X}^- \end{pmatrix} \begin{pmatrix} \mathbb{X}^+ \\ -\mathbb{X}^- \end{pmatrix}^\top = \begin{pmatrix} \mathbb{X}^+\mathbb{X}^{+\top} & -\mathbb{X}^+\mathbb{X}^{-\top} \\ -\mathbb{X}^-\mathbb{X}^{+\top} & \mathbb{X}^-\mathbb{X}^{-\top} \end{pmatrix}.$$

and for all formulations that use only all samples to compute the threshold t, we define kernel matrix $\mathbb{K}^\pm$ as

$$\mathbb{K}^\pm = \begin{pmatrix} \mathbb{X}^+ \\ -\mathbb{X} \end{pmatrix} \begin{pmatrix} \mathbb{X}^+ \\ -\mathbb{X} \end{pmatrix}^\top = \begin{pmatrix} \mathbb{X}^+\mathbb{X}^{+\top} & -\mathbb{X}^+\mathbb{X}^{\top} \\ -\mathbb{X}\mathbb{X}^{+\top} & \mathbb{X}\mathbb{X}^{\top} \end{pmatrix}.$$

In the rest of the text, matrix $\mathbb{K}$ always refers to one of the kernel matrices defined above.

4.1.1 Family of *TopPushK* Formulations

In this section, we focus on the family of *TopPushK* formulations. The general optimization problem that covers all formulations from this family can be written in the following way

$$\underset{\boldsymbol{w}}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i\in\mathcal{I}_+} l(t - \boldsymbol{w}^\top \boldsymbol{x}_i) \tag{4.4a}$$

$$\text{subject to} \quad s_j = \boldsymbol{w}^\top \boldsymbol{x}_j, \quad j \in \tilde{\mathcal{I}}, \tag{4.4b}$$

$$t = \sum_{j=1}^{K} s_{[j]}, \tag{4.4c}$$

where $C \in \mathbb{R}$, and $\tilde{\mathcal{I}}$ and K is defined as follows

$$\tilde{\mathcal{I}} = \begin{cases} \mathcal{I} & \text{for } \textit{TopMeanK}, \\ \mathcal{I}_- & \text{otherwise}, \end{cases} \qquad K = \begin{cases} 1 & \text{for } \textit{TopPush}, \\ n\tau & \text{for } \textit{TopMeanK}, \\ n_-\tau & \text{for } \tau\textit{-FPL}. \end{cases}$$

Note that we use an alternative formulation with constant C. We use this formulation since it is more similar to the standard SVM, and we wanted to stress this similarity. For $C = 1/n_+$ the new formulation is identical to the original one. The following theorem shows the dual form of formulation (4.4). To keep the readability as simple as possible, we postpone all proofs to Appendix B.

Theorem 4.3: Dual formulation for *TopPushK* family

Consider Notation 4.2, surrogate function l, and formulation (4.4). Then the corresponding dual problem has the following form

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{maximize}} \quad -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^\top \mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} - C\sum_{i=1}^{n_+} l^\star\left(\frac{\alpha_i}{C}\right) \tag{4.5a}$$

$$\text{subject to} \quad \sum_{i=1}^{n_+}\alpha_i = \sum_{j=1}^{\tilde{n}}\beta_j, \tag{4.5b}$$

$$0 \le \beta_j \le \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i, \quad j = 1,2,\ldots,\tilde{n}, \tag{4.5c}$$

where $l^\star$ is conjugate function of l and

$$\mathbb{K} = \begin{cases} \mathbb{K}^{\pm} & \text{for } \textit{TopMeanK}, \\ \mathbb{K}^{-} & \text{otherwise}, \end{cases} \quad \tilde{n} = \begin{cases} n & \text{for } \textit{TopMeanK}, \\ n_- & \text{otherwise}, \end{cases} \quad K = \begin{cases} 1 & \text{for } \textit{TopPush}, \\ n\tau & \text{for } \textit{TopMeanK}, \\ n_-\tau & \text{for } \tau\textit{-FPL}. \end{cases}$$

If $K = 1$, the upper bound in the second constraint vanishes due to the first constraint.

4.1.2 Family of *Pat&Mat* Formulations

In the same way, as for *TopPushK* family, we introduce a general optimization problem that covers all formulations from *Pat&Mat* family and reads

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i\in\mathcal{I}_+} l(t-\boldsymbol{w}^\top \boldsymbol{x}_i) \\ \text{subject to} \quad & t \quad \text{solves} \quad \frac{1}{\tilde{n}}\sum_{i\in\tilde{\mathcal{I}}} l\Big(\vartheta(\boldsymbol{w}^\top \boldsymbol{x}_j - t)\Big) = \tau, \end{aligned} \tag{4.6}$$

where $C \in \mathbb{R}$, and $\tilde{\mathcal{I}}$ and $\tilde{n}$ is defined as follows

$$\tilde{\mathcal{I}} = \begin{cases} \mathcal{I} & \text{for } \textit{Pat\&Mat}, \\ \mathcal{I}_- & \text{otherwise.} \end{cases} \qquad \tilde{n} = \begin{cases} n & \text{for } \textit{Pat\&Mat}, \\ n_- & \text{otherwise.} \end{cases}$$

Again, we use the alternative formulation with constant C. The following theorem shows the dual form of the formulation (4.6).

Theorem 4.4: Dual formulation for *Pat&Mat* family

Consider Notation 4.2, surrogate function l, and formulation (4.6). Then the corresponding dual problem has the following form

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta},\delta}{\text{maximize}} \quad -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^\top \mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} - C\sum_{i=1}^{n_+} l^\star\Big(\frac{\alpha_i}{C}\Big) - \delta\sum_{j=1}^{\tilde{n}} l^\star\Big(\frac{\beta_j}{\delta\vartheta}\Big) - \delta\tilde{n}\tau \tag{4.7a}$$

$$\text{subject to} \quad \sum_{i=1}^{n_+}\alpha_i = \sum_{j=1}^{\tilde{n}}\beta_j, \tag{4.7b}$$

$$\delta \geq 0, \tag{4.7c}$$

where $l^\star$ is conjugate function of l, $\vartheta > 0$ is a scaling parameter and

$$\mathbb{K} = \begin{cases} \mathbb{K}^\pm & \text{for } \textit{Pat\&Mat}, \\ \mathbb{K}^- & \text{otherwise,} \end{cases} \qquad \tilde{n} = \begin{cases} n & \text{for } \textit{Pat\&Mat}, \\ n_- & \text{otherwise.} \end{cases}$$

4.1.3 Kernels

As we mentioned at the beginning of the chapter, our goal is to extend our framework to be usable for linearly inseparable problems. We derived dual formulations for *TopPushK* and *Pat&Mat* families in two previous sections. In this section, we show how to employ the kernels method [36] to introduce nonlinearity into these dual formulations. For simplicity, we focus only on formulations from *TopPushK* family that compute the decision threshold only from negative samples. As mentioned in Notation 4.2, all such formulations use kernel matrix $\mathbb{K}^-$. The derivation is the same for all other formulations.

To add kernels, we first realize that the classification score s_j for any sample $\boldsymbol{x}_j \in \mathbb{R}^d$ reads

$$s_j = \boldsymbol{w}^\top \boldsymbol{x}_j = \sum_{i=1}^{n_+}\alpha_i \boldsymbol{x}_j^\top \boldsymbol{x}_i^+ - \sum_{i=1}^{n_-}\beta_i \boldsymbol{x}_j^\top \boldsymbol{x}_i^-, \tag{4.8}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{n_+}$, $\boldsymbol{\beta} \in \mathbb{R}^{n_-}$ are dual variables. This relation yields from the proof of Theorem 4.3. Consider now any kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. Then the first part of the objective func-

tion (4.5a) amounts to

$$\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^{\top}\mathbb{K}^{-}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}=\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^{\top}\begin{pmatrix}\mathbb{X}^{+}\mathbb{X}^{+\top} & -\mathbb{X}^{+}\mathbb{X}^{-\top}\\ -\mathbb{X}^{-}\mathbb{X}^{+\top} & \mathbb{X}^{-}\mathbb{X}^{-\top}\end{pmatrix}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}=\begin{pmatrix}\boldsymbol{\alpha}\\ -\boldsymbol{\beta}\end{pmatrix}^{\top}\begin{pmatrix}\mathbb{X}^{+}\mathbb{X}^{+\top} & \mathbb{X}^{+}\mathbb{X}^{-\top}\\ \mathbb{X}^{-}\mathbb{X}^{+\top} & \mathbb{X}^{-}\mathbb{X}^{-\top}\end{pmatrix}\begin{pmatrix}\boldsymbol{\alpha}\\ -\boldsymbol{\beta}\end{pmatrix}.$$

Using the standard trick, we can replace the kernel matrix $\mathbb{K}^{-}$ with matrix in the following form

$$\mathbb{K}^{-}=\begin{pmatrix}k(\mathbb{X}^{+},\mathbb{X}^{+}) & -k(\mathbb{X}^{+},\mathbb{X}^{-})\\ -k(\mathbb{X}^{-},\mathbb{X}^{+}) & k(\mathbb{X}^{-},\mathbb{X}^{-})\end{pmatrix}, \tag{4.9}$$

where $k(\cdot,\cdot)$ is applied to all rows of both arguments. Then for any sample $\boldsymbol{x}_j$, the classification score (4.8) is replaced by

$$s_j=\sum_{i=1}^{n_+}\alpha_i k\left(\boldsymbol{x}_j,\boldsymbol{x}_i^{+}\right)-\sum_{i=1}^{n_-}\beta_i k\left(\boldsymbol{x}_j,\boldsymbol{x}_i^{-}\right). \tag{4.10}$$

4.2 Coordinate Descent Algorithm

In the previous sections, we derived dual formulations for *TopPushK* and *Pat&Mat* families of formulations. Moreover, we showed how to incorporate non-linear kernels into these formulations. As a result, we can use all presented formulations even for linearly non-separable problems. However, the dimension of the dual problems is at least equal to the number of all samples n, and therefore, it is computationally expensive to use standard techniques such as gradient descent. To handle this issue, the standard coordinate descent algorithm [39, 40] has been proposed in the context of SVMs. In this section, we derive a coordinate descent algorithm suitable for our dual problems (4.5, 4.7). We also show that we can reduce the whole optimization problem to a one-dimensional quadratic optimization problem with a closed-form solution in every iteration. Therefore, every iteration of our algorithm is cheap. For a review of other approaches see [41, 42].

Before we start, recall that the classification scores can be computed directly from dual variables as shown in (4.2). Using the definition of kernel matrix $\mathbb{K}$, we can define a vector of scores $\boldsymbol{s}$ by

$$\boldsymbol{s}=\mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}. \tag{4.11}$$

Note that dual scores are not identical to the primal ones (4.8) (even though we use the same notation). The main difference is that dual scores use kernel function k. Therefore, they are equivalent only if the kernel function is defined as a dot product, i.e., if $k(\boldsymbol{x},\boldsymbol{z})=\boldsymbol{x}^{\top}\boldsymbol{z}$. Moreover, the vector of dual scores (4.11) is not identical to the vector of primal scores from Notation 2.2, even with this choice of the kernel function. The reason is the construction of the kernel matrix $\mathbb{K}$ from Notation 4.2. For example, for $\mathbb{K}^{-}$, the resulting vector of dual scores is a permutation of the vector of primal scores. Similarly, for $\mathbb{K}^{\pm}$, the vector of primal scores corresponds to the elements of (4.11) with indices $n_+ + 1,\ldots,n_+ + n$. Therefore for both definitions of kernel matrices from Notation 2.2, we can quickly get the vector of primal scores by permuting vector (4.11) or selecting only some elements from vector (4.11). To simplify the indexing of the vector of scores (4.11) and kernel matrix $\mathbb{K}$, we introduce a new notation in Notation 4.5.

Notation 4.5

Consider any index l that satisfies $1 \leq l \leq n_+ + \tilde{n}$. Note that the length of dual variable $\boldsymbol{\alpha}$ is n_+ for both formulations (4.5) and (4.7). Therefore, we can define auxiliary index $\hat{l}$ as

$$\hat{l} = \begin{cases} l & \text{if } l \leq n_+, \\ l - n_+ & \text{otherwise.} \end{cases}$$

Then the index l can be safely used for kernel matrix $\mathbb{K}$ or vector of scores $\boldsymbol{s}$, while its corresponding version $\hat{l}$ can be used for dual variables $\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$.

4.2.1 Family of *TopPushK* Formulations

Consider dual formulation (4.5) from Theorem 4.3 and fixed feasible dual variables $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$. Our goal in this section is to derive an efficient iterative procedure for solving this problem. We follow the ideas presented in [39, 40] for solving SVMs using a coordinate descent algorithm. However, we must modify the approach since we have an additional constraint (4.5b). Due to this constraint, we always have to update (at least) two components of dual variables $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ so that we do not violate it. Moreover, there are only three update rules which modify two components of $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, satisfy constraints (4.5b) and also keep (4.11) satisfied. The first one updates two components of $\boldsymbol{\alpha}$

$$\alpha_{\hat{k}} \rightarrow \alpha_{\hat{k}} + \Delta, \qquad \alpha_{\hat{l}} \rightarrow \alpha_{\hat{l}} - \Delta, \qquad \boldsymbol{s} \rightarrow \boldsymbol{s} + (\mathbb{K}_{\bullet,k} - \mathbb{K}_{\bullet,l})\Delta, \tag{4.12a}$$

where $\mathbb{K}_{\bullet,i}$ denotes i-th column of $\mathbb{K}$ and indices $\hat{k}$, $\hat{l}$ are defined in Notation 4.5. Note that the update rule for $\boldsymbol{s}$ does not use matrix multiplication but only vector addition. The second rule updates one component of $\boldsymbol{\alpha}$ and one component of $\boldsymbol{\beta}$

$$\alpha_{\hat{k}} \rightarrow \alpha_{\hat{k}} + \Delta, \qquad \beta_{\hat{l}} \rightarrow \beta_{\hat{l}} + \Delta, \qquad \boldsymbol{s} \rightarrow \boldsymbol{s} + (\mathbb{K}_{\bullet,k} + \mathbb{K}_{\bullet,l})\Delta, \tag{4.12b}$$

and the last one updates two components of $\boldsymbol{\beta}$

$$\beta_{\hat{k}} \rightarrow \beta_{\hat{k}} + \Delta, \qquad \beta_{\hat{l}} \rightarrow \beta_{\hat{l}} - \Delta, \qquad \boldsymbol{s} \rightarrow \boldsymbol{s} + (\mathbb{K}_{\bullet,k} - \mathbb{K}_{\bullet,l})\Delta. \tag{4.12c}$$

Using any of the update rules above, the problem (4.5) can be written as a one-dimensional quadratic problem in the following form

$$\begin{aligned} \underset{\Delta}{\text{maximize}} \quad & -\frac{1}{2}a(\boldsymbol{\alpha},\boldsymbol{\beta})\Delta^2 - b(\boldsymbol{\alpha},\boldsymbol{\beta})\Delta - c(\boldsymbol{\alpha},\boldsymbol{\beta}) \\ \text{subject to} \quad & \Delta_{lb}(\boldsymbol{\alpha},\boldsymbol{\beta}) \leq \Delta \leq \Delta_{ub}(\boldsymbol{\alpha},\boldsymbol{\beta}) \end{aligned}$$

where a, b, c, Δ_{lb}, Δ_{ub} are constants with respect to Δ. The optimal solution to this problem is

$$\Delta^\star = \text{clip}_{[\Delta_{lb},\, \Delta_{ub}]}(\gamma), \tag{4.13}$$

where $-b/a$ and $\text{clip}_{[a,\,b]}(x)$ amounts to clipping (projecting) x to interval $[a, b]$. Since we assume one of the update rules (4.12), the constrain (4.5b) is always satisfied after the update. Even though all three update rules hold for any surrogate, the calculation of the optimal $\Delta^\star$ depends on the concrete form of surrogate function. In the following text, we show the closed-form formula for $\Delta^\star$, when the hinge loss or quadratic hinge loss is used as a surrogate function.

Hinge loss

We start with the hinge loss function from Notation 2.1. Plugging the conjugate (4.2) of the hinge loss into the dual formulation (4.5) yields

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{maximize}} \quad -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^{\top}\mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}+\sum_{i=1}^{n_+}\alpha_i \tag{4.14a}$$

$$\text{subject to} \quad \sum_{i=1}^{n_+}\alpha_i=\sum_{j=1}^{\tilde{n}}\beta_j, \tag{4.14b}$$

$$0\le\alpha_i\le C, \qquad i=1,2,\ldots,n_+, \tag{4.14c}$$

$$0\le\beta_j\le\frac{1}{K}\sum_{i=1}^{n_+}\alpha_i, \quad j=1,2,\ldots,\tilde{n}. \tag{4.14d}$$

The form of $\mathbb{K}$ and $\tilde{n}$ depends on the used formulation as discussed in Theorem 4.3. Moreover, the upper bound in (4.14d) can be omitted for $K=1$. Since we know the form of the optimal solution (4.13), we only need to show how to compute Δ_{lb}, Δ_{ub} and γ for all update rules (4.12). The following three propositions provide closed-form formulas for all three update rules. To keep the presentation as simple as possible, we postpone all proofs to Appendix B.2.1.

Proposition 4.6: Update rule (4.12a) for problem (4.14)

Consider problem (4.14), update rule (4.12a), indices $1\le k\le n_+$ and $1\le l\le n_+$ and Notation 4.5. Then the optimal solution $\Delta^\star$ is given by (4.13) where

$$\begin{aligned}\Delta_{lb}&=\max\{-\alpha_{\hat{k}},\ \alpha_{\hat{l}}-C\},\\ \Delta_{ub}&=\min\{C-\alpha_{\hat{k}},\ \alpha_{\hat{l}}\},\\ \gamma&=-\frac{s_k-s_l}{\mathbb{K}_{kk}+\mathbb{K}_{ll}-\mathbb{K}_{kl}-\mathbb{K}_{lk}}.\end{aligned}$$

Proposition 4.7: Update rule (4.12b) for problem (4.14)

Consider problem (4.14), update rule (4.12b), indices $1\le k\le n_+$ and $n_++1\le l\le\tilde{n}$ and Notation 4.5. Let us define

$$\beta_{\max}=\max_{j\in\{1,2,\ldots,\tilde{n}\}\setminus\{\hat{l}\}}\beta_j.$$

Then the optimal solution $\Delta^\star$ is given by (4.13) where

$$\begin{aligned}\Delta_{lb}&=\begin{cases}\max\left\{-\alpha_{\hat{k}},\ -\beta_{\hat{l}}\right\} & \text{K}=1,\\ \max\left\{-\alpha_{\hat{k}},\ -\beta_{\hat{l}},\ K\beta_{\max}-\sum_{i=1}^{n_+}\alpha_i\right\} & \text{otherwise},\end{cases}\\ \Delta_{ub}&=\begin{cases}C-\alpha_{\hat{k}} & \text{K}=1,\\ \min\left\{C-\alpha_{\hat{k}},\ \frac{1}{K-1}\left(\sum_{i=1}^{n_+}\alpha_i-K\beta_{\hat{l}}\right)\right\} & \text{otherwise}.\end{cases}\\ \gamma&=-\frac{s_k+s_l-1}{\mathbb{K}_{kk}+\mathbb{K}_{ll}+\mathbb{K}_{kl}+\mathbb{K}_{lk}}.\end{aligned}$$

Proposition 4.8: Update rule (4.12c) for problem (4.14)

Consider problem (4.14), update rule (4.12c), indices $n_+ + 1 \leq k \leq \tilde{n}$ and $n_+ + 1 \leq l \leq \tilde{n}$ and Notation 4.5. Then the optimal solution $\Delta^\star$ is given by (4.13) where

$$\Delta_{lb} = \begin{cases} -\beta_{\hat{k}} & \mathrm{K} = 1, \\ \max\left\{-\beta_{\hat{k}},\ \beta_{\hat{l}} - \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i\right\} & \text{otherwise}, \end{cases}$$

$$\Delta_{ub} = \begin{cases} \beta_{\hat{l}} & \mathrm{K} = 1, \\ \min\left\{\frac{1}{K}\sum_{i=1}^{n_+}\alpha_i - \beta_{\hat{k}},\ \beta_{\hat{l}}\right\} & \text{otherwise}. \end{cases}$$

$$\gamma = -\frac{s_k - s_l}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}}.$$

Quadratic hinge loss

The second considered surrogate function is the quadratic hinge loss from Notation 2.1. Plugging the conjugate (4.2) of the quadratic hinge loss into the dual formulation (4.5) yields

$$\begin{aligned}
\underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{maximize}} \quad & -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^\top \mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} + \sum_{i=1}^{n_+}\alpha_i - \frac{1}{4C}\sum_{i=1}^{n_+}\alpha_i^2 && \text{(4.15a)}\\
\text{subject to} \quad & \sum_{i=1}^{n_+}\alpha_i = \sum_{j=1}^{\tilde{n}}\beta_j, && \text{(4.15b)}\\
& 0 \leq \alpha_i, \qquad i = 1,2,\ldots,n_+, && \text{(4.15c)}\\
& 0 \leq \beta_j \leq \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i, \quad j = 1,2,\ldots,\tilde{n}, && \text{(4.15d)}
\end{aligned}$$

Similarly to the previous case, the form of $\mathbb{K}$ and $\tilde{n}$ depends on the used formulation and the upper bound in (4.15d) can be omitted for $K = 1$. For simplicity, we postpone formulas for update rules (4.12) and their proofs to Appendix B.2.1. More specifically, all update rules can be found in Proposition B.6, B.7, and B.8.

Initialization

For all update rules (4.12) we assumed that the current solution $\boldsymbol{\alpha}, \boldsymbol{\beta}$ is feasible. So to create an iterative algorithm that solves problem (4.14) or (4.15), we need to have a way how to obtain an initial feasible solution. Such a task can be formally written as a projection of random variables $\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0$ to the feasible set of solutions

$$\begin{aligned}
\underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{minimize}} \quad & \frac{1}{2}\left\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^0\right\|^2 + \frac{1}{2}\left\|\boldsymbol{\beta} - \boldsymbol{\beta}^0\right\|^2\\
\text{subject to} \quad & \sum_{i=1}^{n_+}\alpha_i = \sum_{j=1}^{\tilde{n}}\beta_j,\\
& 0 \leq \alpha_i \leq C_1, \quad i = 1,2,\ldots,n_+,,\\
& 0 \leq \beta_j \leq \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i, \quad j = 1,2,\ldots,\tilde{n},
\end{aligned} \qquad \text{(4.16)}$$

where the upper bound in the second constraint depends on the used surrogate function. For hinge loss function, the upper bound is $C_1 = C$, while for the quadratic hinge loss it is $C_1 = +\infty$.

To solve problem (4.18), we follow the same approach as in [43]. In the following theorem, we show that problem (4.18) can be written as a system of two equations of two variables λ and μ. Moreover, the theorem shows the concrete form of feasible solution $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ that depends only on λ and μ.

Theorem 4.9

Consider problem (4.18), some initial solution $\boldsymbol{\alpha}^0$, $\boldsymbol{\beta}^0$ and denote the sorted version (in non-decreasing order) of $\boldsymbol{\beta}^0$ as $\boldsymbol{\beta}^0_{[\cdot]}$. Then if the following condition holds

$$\sum_{j=1}^{K}\left(\beta^0_{[\tilde{n}-K+j]} + \max_{i=1,\dots,n_+} \alpha_i^0\right) \leq 0, \tag{4.17}$$

the optimal solution of (4.18) amounts to $\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0}$. In the opposite case, the following system of two equations

$$\sum_{i=1}^{n_+} \operatorname{clip}_{[0,\,C_1]}\left(\alpha_i^0 - \lambda + \frac{1}{K}\sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,+\infty)}\left(\beta_j^0 + \lambda - \mu\right)\right) - K\mu = 0, \tag{4.18a}$$

$$\sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu]}\left(\beta_j^0 + \lambda\right) - K\mu = 0, \tag{4.18b}$$

has a solution (λ, μ) with $\mu > 0$, and the optimal solution of (4.18) is equal to

$$\alpha_i = \operatorname{clip}_{[0,\,C_1]}\left(\alpha_i^0 - \lambda + \frac{1}{K}\sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,+\infty)}\left(\beta_j^0 + \lambda - \mu\right)\right),$$
$$\beta_j = \operatorname{clip}_{[0,\,\mu]}\left(\beta_j^0 + \lambda\right).$$

Theorem 4.9 shows the optimal solution of B that depends only on (λ, μ) but does not provide any way to find such a solution. In the following text, we show that the number of variables in the system of equations (4.18) can be reduced to one. For any fixed μ, we denote the function on the left-hand side of (4.18b) by

$$g(\lambda; \mu) := \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu]}\left(\beta_j^0 + \lambda\right) - K\mu.$$

Then g is non-decreasing in λ but not necessarily strictly increasing. We denote by $\lambda(\mu)$ any such λ solving (4.18b) for a fixed μ. Denote $\boldsymbol{z}$ the sorted version of $-\boldsymbol{\beta}^0$. Then we have

$$g(\lambda; \mu) = \sum_{\{j \,|\, \lambda - z_j \in [0,\mu)\}} (\lambda - z_j) + \sum_{\{j \,|\, \lambda - z_j \geq \mu\}} \mu - K\mu.$$

Now we can easily compute $\lambda(\mu)$ by solving $g(\lambda(\mu); \mu) = 0$ for fixed μ. To get the solution efficiently, we derive Algorithm 3, which can be described as follows: Index i will run over $\boldsymbol{z}$ while index j will run over $\boldsymbol{z} + \mu$. At every iteration, we know the values of $g(z_{i-1}; \mu)$ and $g(z_{j-1} + \mu; \mu)$ and we want to evaluate g at the next point. We denote the number of indices j such that $\lambda - z_j \in [0, \mu)$ by d. If $z_i \leq z_j + \mu$, then we consider $\lambda = z_i$ and since one index enters the set $\left\{j \,\middle|\, \lambda - z_j \in [0, \mu)\right\}$, we increase d by one. On the other hand, if $z_i > z_j + \mu$, then we consider $\lambda = z_j + \mu$ and since one index leaves the set $\left\{j \,\middle|\, \lambda - z_j \in [0, \mu)\right\}$, we decrease d by one. In both

cases, g is increased by d times the difference between the new λ and old λ. Once g exceeds 0, we stop the algorithm and linearly interpolate between the last two values. To prevent an overflow, we set $z_{m+1} = +\infty$. Concerning the initial values, since $z_1 \leq z_1 + \mu$, we set $i = 2$, $j = 1$ and $d = 1$.

Algorithm 3 An efficient algorithm for computing $\lambda(\mu)$ from (4.18) for fixed μ..

Require: vector $-\boldsymbol{\beta}^0$ sorted into $\boldsymbol{z}$
$i \leftarrow 2, j \leftarrow 1, d \leftarrow 1$
$\lambda \leftarrow z_1, g \leftarrow -K\mu$
while $g < 0$ **do**
  if $z_i \leq z_j + \mu$ **then**
    $g \leftarrow g + d(z_i - \lambda)$
    $\lambda \leftarrow z_i, d \leftarrow d + 1, i \leftarrow i + 1$
  else
    $g \leftarrow g + d(z_j + \mu - \lambda)$
    $\lambda \leftarrow z_j + \mu, d \leftarrow d - 1, j \leftarrow j + 1$
  end if
end while
return linear interpolation of the last two values of λ

Since $\lambda(\mu)$ can be computed for fixed μ using Algorithm 3, we can define auxiliary function H in the following form

$$h(\mu) = \sum_{i=1}^{n_+} \operatorname{clip}_{[0,\, C_1]}\left(\alpha_i^0 - \lambda(\mu) + \frac{1}{K}\sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,+\infty)}\left(\beta_j^0 + \lambda(\mu) - \mu\right)\right) - K\mu. \tag{4.19}$$

Then the system of equations (4.18) is equivalent to $h(\mu) = 0$. The following lemma describes properties of h. Since h is decreasing in μ on $(0, \infty)$, any root-finding algorithm such as Bisection can be used to find the optimal solution.

Lemma 4.10

Even though $\lambda(\mu)$ is not unique, function h from (4.19) is well-defined in the sense that it gives the same value for every choice of $\lambda(\mu)$. Moreover, h is decreasing in μ on $(0, +\infty)$.

4.2.2 Family of *Pat&Mat* Formulations

In this section, we derive a coordinate descent algorithm for solving dual formulation (4.7) for the family of *Pat&Mat* formulations. We follow the same approach as for *TopPushK* family in the previous section, i.e. we use update rules (4.12). In this case, we must also consider the third primary variable δ. Then the dual formulation (4.7) can be rewritten as a one-dimensional quadratic problem

$$\begin{aligned} \underset{\Delta}{\text{maximize}} \quad & -\frac{1}{2}a(\boldsymbol{\alpha}, \boldsymbol{\beta}, \delta)\Delta^2 - b(\boldsymbol{\alpha}, \boldsymbol{\beta}, \delta)\Delta - c(\boldsymbol{\alpha}, \boldsymbol{\beta}, \delta) \\ \text{subject to} \quad & \Delta_{lb}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \delta) \leq \Delta \leq \Delta_{ub}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \delta) \end{aligned}$$

where a, b, c, Δ_{lb}, Δ_{ub} are constants with respect to Δ. The form of the optimal solution is the same as for problem (4.5) and reads

$$\Delta^\star = \operatorname{clip}_{[\Delta_{lb},\, \Delta_{ub}]}(\gamma).$$

Since we assume one of the update rule (4.12), the constrain (4.7b) is always satisfied after the update. The exact form of the update rules depends on the surrogate function. Moreover, the form of optimal δ also depends on the surrogate function. The upcoming text follows the same order as in the previous section. Therefore, we introduce concrete forms of update rules for hinge and quadratic hinge loss function and then show how to find an initial feasible solution.

Hinge Loss

We again start with the hinge loss function from Notation 2.1. Plugging the conjugate (4.2) of the hinge loss into the dual formulation (4.7) yields

$$\begin{aligned}
\underset{\boldsymbol{\alpha},\boldsymbol{\beta},\delta}{\text{maximize}} \quad & -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^{\top}\mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}+\sum_{i=1}^{n_+}\alpha_i+\frac{1}{\vartheta}\sum_{j=1}^{\tilde{n}}\beta_j-\delta\tilde{n}\tau && \text{(4.20a)}\\
\text{subject to} \quad & \sum_{i=1}^{n_+}\alpha_i=\sum_{j=1}^{\tilde{n}}\beta_j, && \text{(4.20b)}\\
& 0\le\alpha_i\le C, \quad i=1,2,\dots,n_+, && \text{(4.20c)}\\
& 0\le\beta_j\le\delta\vartheta, \quad j=1,2,\dots,\tilde{n}, && \text{(4.20d)}\\
& \delta\ge 0. && \text{(4.20e)}
\end{aligned}$$

Since we know the form of the optimal solution (4.13), we only need to show how to compute Δ_{lb}, Δ_{ub} and γ for all update rules (4.12). However, in this case, constants Δ_{lb}, Δ_{ub} and γ also depend on the third dual variable δ. We do not perform a joint maximization in $(\alpha_{\hat{k}},\ \beta_{\hat{l}},\ \delta)$ but perform a maximization with respect to $(\alpha_{\hat{k}},\ \beta_{\hat{l}})$, update these two values and then optimize the objective with respect to δ. Then for fixed feasible solution $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, maximizing objective function (4.20a) with respect to δ yields

$$\begin{aligned}
\underset{\delta}{\text{maximize}} \quad & -\tilde{n}\tau\delta\\
\text{subject to} \quad & 0\le\beta_j\le\delta\vartheta, \quad j=1,2,\dots,\tilde{n},\\
& \delta\ge 0.
\end{aligned}$$

Since $\tilde{n}\tau\ge 0$, we have to find the smallest possible δ that satisfies constraints above. Such δ is in the following form

$$\delta^{*}=\frac{1}{\vartheta}\max_{j\in\{1,2,\dots,\tilde{n}\}}\beta_j. \tag{4.21}$$

Since the formulas for optimal update rules are relatively long, we postpone them to Appendix B.2.2. More specifically, all update rules can be found in Proposition B.11, B.12, and B.13.

Quadratic Hinge Loss

The second considered surrogate function is the quadratic hinge loss from Notation 2.1. Plugging the conjugate (4.3) of the quadratic hinge loss into the dual formulation (4.7) yields

$$\begin{aligned}
\underset{\boldsymbol{\alpha},\boldsymbol{\beta},\delta}{\text{maximize}} \quad & -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^{\top}\mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}+\sum_{i=1}^{n_+}\alpha_i-\frac{1}{4C}\sum_{i=1}^{n_+}\alpha_i^2 && \text{(4.22a)}\\
& +\frac{1}{\vartheta}\sum_{j=1}^{\tilde{n}}\beta_j-\frac{1}{4\delta\vartheta^2}\sum_{j=1}^{\tilde{n}}\beta_j^2-\delta\tilde{n}\tau && \text{(4.22b)}\\
\text{subject to} \quad & \sum_{i=1}^{n_+}\alpha_i=\sum_{j=1}^{\tilde{n}}\beta_j, && \text{(4.22c)}\\
& \alpha_i\geq 0, \qquad i=1,2,\ldots,n_+, && \text{(4.22d)}\\
& \beta_j\geq 0, \qquad j=1,2,\ldots,\tilde{n}, && \text{(4.22e)}\\
& \delta\geq 0, && \text{(4.22f)}
\end{aligned}$$

Similar to the previous case, we perform maximization only with respect to $(\alpha_{\hat{k}},\ \beta_{\hat{l}})$. Then for fixed feasible solution $\boldsymbol{\alpha},\ \boldsymbol{\beta}$, we need to maximize the objective function (4.22a-4.22b) with respect to δ, which leads to the following problem

$$\begin{aligned}
\underset{\delta}{\text{maximize}} \quad & -(\tilde{n}\tau)\delta-\left(\frac{1}{4\vartheta^2}\sum_{j=1}^{\tilde{n}}\beta_j^2\right)\frac{1}{\delta}\\
\text{subject to} \quad & \delta\geq 0,
\end{aligned}$$

with the optimal solution that equals to

$$\delta^*=\sqrt{\frac{1}{4\vartheta^2\tilde{n}\tau}\sum_{j=1}^{\tilde{n}}\beta_j^2}. \tag{4.23}$$

As in the previous section, we postpone the formulas for optimal update rules to Appendix B.2.2. More specifically, all update rules can be found in Propositions B.14, B.15, and B.16.

Initialization

As in the case of problem (4.5), all update rules (4.12) assume that the current solution $\boldsymbol{\alpha},\ \boldsymbol{\beta}\ \delta$ is feasible. So to create an iterative algorithm that solves problem (4.20) or (4.22), we need to have a way how to obtain an initial feasible solution. Such a task can be formally written as a projection of random variables $\boldsymbol{\alpha}^0,\ \boldsymbol{\beta}^0,\ \delta^0$ to the feasible set of solutions

$$\begin{aligned}
\underset{\boldsymbol{\alpha},\boldsymbol{\beta},\delta}{\text{minimize}} \quad & \frac{1}{2}\left\|\boldsymbol{\alpha}-\boldsymbol{\alpha}^0\right\|^2+\frac{1}{2}\left\|\boldsymbol{\beta}-\boldsymbol{\beta}^0\right\|^2+\frac{1}{2}(\delta-\delta^0)^2\\
\text{subject to} \quad & \sum_{i=1}^{n_+}\alpha_i=\sum_{j=1}^{\tilde{n}}\beta_j,\\
& 0\leq\alpha_i\leq C_1, \quad i=1,2,\ldots,n_+,\\
& 0\leq\beta_j\leq C_2\delta, \quad j=1,2,\ldots,\tilde{n},,\\
& \delta\geq 0,
\end{aligned} \tag{4.24}$$

where the upper bounds in the second and third constraints depend on the used surrogate function and are defined as follows

$$C_1 = \begin{cases} C & \text{for hinge loss,} \\ +\infty & \text{for quadratic hinge loss,} \end{cases} \qquad C_2 = \begin{cases} \delta\vartheta & \text{for hinge loss,} \\ +\infty & \text{for quadratic hinge loss.} \end{cases}$$

Again, we will follow the same approach as in [43] to solve this optimization problem. In the following theorem, we show that problem (4.26) can be written as a system of two equations of two variables λ and μ. The theorem also shows the concrete form of feasible solution $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ δ that depends only on λ and μ.

Theorem 4.11

Consider problem (4.26) and some initial solution $\boldsymbol{\alpha}^0$, $\boldsymbol{\beta}^0$ and δ^0. Then if the following condition holds

$$\delta^0 \leq -C_2 \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,+\infty)}\left(\beta_j^0 + \max_{i=1,\ldots,n_+} \alpha_i^0\right). \tag{4.25}$$

the optimal solution of (4.18) amounts to $\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0}$ and $\delta^0 = 0$. In the opposite case, the following system of two equations

$$0 = \sum_{i=1}^{n_+} \operatorname{clip}_{[0,\,C_1]}\left(\alpha_i^0 - \lambda\right) - \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\lambda+\mu]}\left(\beta_j^0 + \lambda\right), \tag{4.26a}$$

$$\lambda = C_2\delta^0 + C_2^2 \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,+\infty)}\left(\beta_j^0 - \mu\right) - \mu. \tag{4.26b}$$

has a solution (λ, μ) with $\lambda + \mu > 0$ and the optimal solution of (4.26) is equal to

$$\begin{aligned} \alpha_i &= \operatorname{clip}_{[0,\,C_1]}\left(\alpha_i^0 - \lambda\right), \\ \beta_j &= \operatorname{clip}_{[0,\,\lambda+\mu]}\left(\beta_j^0 + \lambda\right), \\ C_2\delta &= \lambda + \mu. \end{aligned}$$

System (4.26) is relatively simple to solve, since equation (4.26b) provides an explicit formula for λ. Let us denote it as $\lambda(\mu)$, then we denote the right-hand side of (4.26a) as

$$h(\mu) := \sum_{i=1}^{n_+} \operatorname{clip}_{[0,\,C_1]}\left(\alpha_i^0 - \lambda(\mu)\right) - \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\lambda(\mu)+\mu]}\left(\beta_j^0 + \mu\right). \tag{4.27}$$

Then the system of equations (4.26) is equivalent to solving $h(\mu) = 0$. The following lemma states that h is a non-decreasing function in μ on $(0, \infty)$ and thus the equation $h(\mu) = 0$ is simple to solve using any root-finding method. Note that if $\delta^0 < 0$, then it may happen that $\lambda + \mu < 0$ if the initial μ is chosen large. In such a case, it suffices to decrease μ until $\lambda + \mu$ is positive.

Lemma 4.12

Function h is non-decreasing in μ on $(0, \infty)$.

4.3 Summary

In this chapter, we derived dual formulation for *TopPushK* and *Pat&Mat* family of formulations. Moreover, we derive simple update rules that can be used to improve the current feasible solution. We also showed that these update rules have closed-form formulas, and therefore they are simple to compute. Finally, we showed how to find an initial feasible solution. This section combines all these intermediate results into Algorithm 4 and discusses its computational complexity.

Algorithm 4 Coordinate descent algorithm for *TopPushK* family of formulations (**left**) and *Pat&Mat* family of formulations (**right**).

1: Set $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ using Theorem 4.9	1: Set $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, δ using Theorem 4.11
2: Set $\boldsymbol{s}$ based on (4.11)	2: Set $\boldsymbol{s}$ based on (4.11)
3: **repeat**	3: **repeat**
4: Pick random k from $\{1,\ldots,n_+ + \tilde{n}\}$	4: Pick random k from $\{1,\ldots,n_+ + \tilde{n}\}$
5: **for** $l \in \{1,\ldots,n_+ + \tilde{n}\}$ **do**	5: **for** $l \in \{1,\ldots,n_+ + \tilde{n}\}$ **do**
6: Compute Δ_l	6: Compute Δ_l and δ_l
7: **end for**	7: **end for**
8: Select the best Δ_l	8: Select the best $Delta_l$ and δ_l
9: Update $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{s}$ according to (4.12)	9: Update $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{s}$ according to (4.12)
10:	10: set $\delta \leftarrow \delta_l$
11: **until** stopping criterion is satisfied	11: **until** stopping criterion is satisfied

The left column in Algorithm 4 describe the algorithm for *TopPushK* family while the right column for *Pat&Mat* family. In step 2 we initialize $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and δ to some feasible value using Theorem 4.9 or Theorem 4.11. Then, based on (4.11) we compute scores $\boldsymbol{s}$. Each `repeat` loop in step 3 updates two coordinates as shown in (4.12). In step 4 we select a random index k and in the `for` loop in step 5 we compute the optimal (Δ_l, δ_l) for all possible combinations (k, l) as in (4.12). In step 8 we select the best pair (Δ_l, δ_l) which maximizes the coresponding objective function. Finally, based on the selected update rule we update $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{s}$ and δ in steps 9 and 10.

Now we derive the computational complexity of each `repeat` loop from step 3. The computation of (Δ_l, δ_l) amounts to solving a quadratic optimization problem in one variable. As we showed in Sections 4.2.1 and 4.2.2, there is a closed-form solution and step 6 can be performed in $O(1)$. Since this is embedded in a `for` loop in step 5, the whole complexity of this loop is $O(n_+ + \tilde{n})$. Step 9 requires $O(1)$ for the update of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ while $O(n_+ + \tilde{n})$ for the update of $\boldsymbol{s}$. Since the other steps are $O(1)$, the total complexity of the `repeat` loop is $O(n_+ + \tilde{n})$. This holds only if the kernel matrix $\mathbb{K}$ is precomputed. In the opposite case, all complexities must be multiplied by the cost of computation of components of $\mathbb{K}$, which is $O(d)$. This complexity analysis is summarized in Table 4.1.

Operation	$\mathbb{K}$ precomputed	$\mathbb{K}$ not precomputed
Evaluation of Δ_l	$O(1)$	$O(d)$
Update of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$	$O(1)$	$O(1)$
Update of $\boldsymbol{s}$	$O(n_+ + \tilde{n})$	$O((n_+ + \tilde{n})d)$
Total per iteration	$O(n_+ + \tilde{n})$	$O((n_+ + \tilde{n})d)$

Table 4.1: Computational complexity of one `repeat` loop (which updates two coordinates of $\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$) from Algorithm 4.

5

Primal Formulation: Non-Linear Model

In Chapter 2 we introduced a general framework for binary classification at the top samples and showed multiple formulations that fall into it. All these formulations are summarized in Table 2.1. In Chapter 3, we discussed the theoretical properties of all formulations for the special case of a linear model. Since many real-world problems are not linearly separable, in Chpater 4, we derived dual forms of all formulations and employed non-linear kernels. Moreover, we derived an efficient algorithm to solve these dual formulations. However, it is still computationally expensive. Especially the evaluation of new samples is costly since the classification score for every new sample is computed from all training samples. More precisely, the classification score is calculated only from training samples whose corresponding dual variables are non-zero. Therefore, the dual formulations are unsuitable for large data. To overcome this issue, the following chapter is dedicated to our framework (2.3) with an arbitrary model model f. We can mention neural networks as a prototypical example of such a model. We will not discuss *Grill* and *Grill-NP* formulations from Table 2.1, since their authors introduced in [3] an algorithm to solve this formulation that is suitable even for non-linear models. Therefore, we focus only on the remaining formulations that use surrogate false-negative rate as an objective function, i.e., we assume the following general formulation

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} l(t(\boldsymbol{w}) - f(\boldsymbol{x}_i;\boldsymbol{w})) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i;\boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = G(\boldsymbol{s},\boldsymbol{y}), \end{aligned} \tag{5.1}$$

where f is an arbitrary model. Our goal is to show how to solve this formulation in a suitable way for large data. The standard approach is to use stochastic gradient descent. To do so, we need to know the gradient of the objective function, and therefore, we assume that the model f is differentiable. The optimization problem (5.1) depends on two decision variables $\boldsymbol{w}$ and t. However, for all formulations from Table 2.1 and each $\boldsymbol{w}$, the threshold t can be computed uniquely. We stress this dependence by writing $t(\boldsymbol{w})$ instead of t. By doing so, we effectively remove the threshold t from the decision variables and $\boldsymbol{w}$ remains the only decision variable. Denoting the objective of (5.1) by $L(\boldsymbol{w})$, the chain rule implies that the gradient is equal to

$$\nabla L(\boldsymbol{w}) = \boldsymbol{w} + \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} l'(t(\boldsymbol{w}) - f(\boldsymbol{x}_i;\boldsymbol{w}))(\nabla t(\boldsymbol{w}) - \nabla f(\boldsymbol{x}_i;\boldsymbol{w})). \tag{5.2}$$

The only remaining part is the computation of $\nabla t(\boldsymbol{w})$. It is simple for most of the formulations from Table 2.1. Moreover, Theorem 3.3 shows the computation for *Pat&Mat*.

The basic idea of the stochastic gradient descent is to split the dataset into several small minibatches $\mathcal{I}^1_{\text{mb}}, \mathcal{I}^2_{\text{mb}}, \ldots, \mathcal{I}^m_{\text{mb}}$ and at each iteration perform all operations only on one of them. This approach is easily applicable when for example, cross-entropy is used as an objective function since cross-entropy is additive and easily decomposable. However, in our case, the

decision threshold t depends on all scores $\boldsymbol{s}$, and consequently, the objective function is non-additive and non-decomposable. Therefore, we cannot compute the true threshold only from one minibatch, and we need to use some approximation $\hat{t}$ of it. The most straightforward way to approximate the true threshold is to use the same rule as for the whole dataset and compute the sampled threshold $\hat{t}$ only on data from one minibatch. By replacing all computations on the whole dataset $\mathcal{I}$ with their counterparts computed only on the minibatch $\mathcal{I}_{\text{mb}}$, we get the sampled gradient in the following form

$$\nabla \hat{L}(\boldsymbol{w}) = \boldsymbol{w} + \frac{1}{n_{\text{mb},+}} \sum_{i \in \mathcal{I}_{\text{mb},+}} l'\Big(\hat{t}(\boldsymbol{w}) - f(\boldsymbol{x}_i; \boldsymbol{w})\Big)\Big(\nabla \hat{t}(\boldsymbol{w}) - \nabla f(\boldsymbol{x}_i; \boldsymbol{w})\Big). \tag{5.3}$$

where $n_{\text{mb},+}$ denotes the number of positive samples in the minibatch and $\hat{t}$ denotes the sampled version of the true threshold t. The whole procedure of stochastic gradient descent for formulations from Table 2.1 is summarized in Algorithm 5.

Algorithm 5 Stochastic gradient descent for solving problem (5.1).

Require: Dataset $\mathcal{D}$, minibatches $\mathcal{I}^1_{\text{mb}}, \mathcal{I}^2_{\text{mb}}, \ldots, \mathcal{I}^m_{\text{mb}}$, and stepsize α^k

1: Initialize weights $\boldsymbol{w}^0$, $k \leftarrow 0$
2: **repeat**
3: Select a minibatch $\mathcal{I}^k_{\text{mb}}$
4: Compute scores $\boldsymbol{s}_{\text{mb}}$ for all $i \in \mathcal{I}^k_{\text{mb}}$ as $s_i \leftarrow f(\boldsymbol{x}_i; \boldsymbol{w})$
5: Compute sampled thershold $\hat{t} \leftarrow G(\boldsymbol{s}_{\text{mb}}, \boldsymbol{y}_{\text{mb}})$
6: Compute sampled gradient $\nabla \hat{L}$ based on $\mathcal{I}^k_{\text{mb}}$ according to (5.3)
7: Set $\boldsymbol{w}^{k+1} \leftarrow \boldsymbol{w}^k - \alpha^k \cdot \nabla \hat{L}$
8: Set $k \leftarrow k + 1$
9: **until** stopping criterion is satisfied

5.1 Bias of Sampled Gradient

In Algorithm 5, we summarized a basic form of the stochastic gradient descent algorithm that can be used for any formulation from Table 2.1 that uses surrogate false-negative rate as an objective function. The problem with this approach is that the sampled threshold $\hat{t}$ is computed only from data from one minibatch using the same formula as for the whole dataset. However, such a choice of sampled threshold $\hat{t}$ underestimates the true value. This issue is especially evident for *TopPush*, where the sampled maximum is always smaller or equal to the true maximum. The chances that the actual maximum is in the current minibatch are small for large data. Therefore, the sampled threshold is a biased estimate of the true threshold. Another example can be the threshold for *Pat&Mat* formulation. Consider the minibatch of size 32, a typical size used in many applications. How can we compute, for example, (0.01)-quantile from only 32 samples? Figure 5.1 illustrates the bias of sampled threshold for *Pat&Mat* with $\tau = 0.01$ and minibatches of different sizes. The bias between the true and sampled threshold is large, even for medium-sized minibatches. When the backpropagation is used, the sampling error is propagated through the whole gradient, and consequently, the sampled gradient is a biased estimate of the true gradient. This brings numerical issues as discussed in [44].

Convergence proofs of the stochastic gradient descent require that the sampled gradient is an unbiased estimate of the true gradient [44]. In other words, the bias defined as

$$\text{bias}(\boldsymbol{w}) := \nabla L(\boldsymbol{w}) - \mathbb{E}\nabla \hat{L}(\boldsymbol{w}) \tag{5.4}$$

must equal 0 for all $\boldsymbol{w}$. A comparison of (5.2) and (5.3) shows that a necessary condition is that the sampled threshold $\hat{t}$ is an unbiased estimate of the true threshold t. However, as

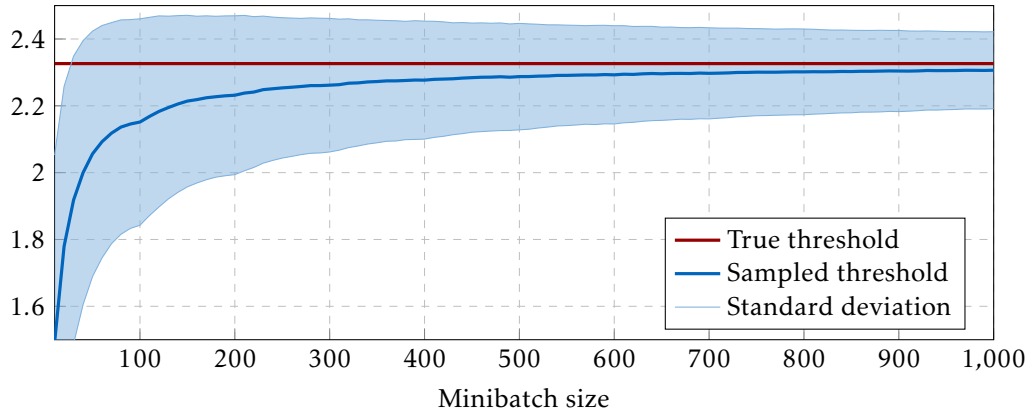


Figure 5.1: The bias between the sampled and true thresholds computed from scores following the standard normal distribution. The threshold separates the top 1% of samples with the highest scores.

we discussed above, it is not a case for Algorithm 5. The following proposition quantifies the difference between the sampled and true threshold for methods that use quantile as the threshold.

Proposition 5.1: [45]

Consider an absolutely continuous random variable X with distribution function F. Let $X_1, X_2, \ldots, X_n$ be i.i.d. samples from X and let $\tau \in (0,1)$. Denote the true quantile t and its sampled version as $\hat{t}$

$$t = F^{-1}(1-\tau), \qquad \hat{t} = F_n^{-1}(1-\tau),$$

where F_n is the empirical distribution function. If F is differentiable with a positive gradient at t, then

$$\sqrt{n}\left(t - \hat{t}\right) \to \mathcal{N}\left(0, \frac{\tau(1-\tau)}{F'(t)^2}\right),$$

where the convergence is in distribution and $\mathcal{N}$ denotes the normal distribution.

This proposition states that when the minibatch size increases to infinity, the variance of the sampled threshold is approximately

$$\frac{\tau(1-\tau)}{nF'(t)^2}.$$

Figure 5.1 shows this empirically for the case where the scores follow the standard normal distribution and $\tau = 0.01$ is the desired top fraction of all samples. The approximation is poor with both significant bias and standard deviation. The natural choice to mitigate the bias is to work with large minibatches. Even though this is not a standard way, some works suggest this route [46]. When the minibatch is large, it contains more samples, and the sampled threshold is more precise. However, such an approach is not applicable in many cases. For example, GPUs are often used to speed up the training process. But the usage of GPUs brings memory constraints, and therefore only small minibatches can be used in such a case.

5.2 *DeepTopPush*

In the previous sections, we derived Algorithm 5 that can be used for any formulation from Table 2.1 that uses surrogate false-negative rate as an objective function. However, this approach provides a biased sampled gradient, and the only way to reduce the bias is to use large minibatches. In this section, we derive a new method *DeepTopPush*, that mitigates this bias differently.

We start with *TopPush* formulation presented in [22]. Authors of [22] proposed the *TopPush* formulation with a linear model and solved it in its dual form. In Section 2.2 we generalized this formulation for general model f, and in Chapter 3 we solved the formulation directly in its primal form for a linear model. For general model f, we stay in the primal form to be able to employ stochastic gradient descent. It means that we have the following optimization problem

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{n_+}\operatorname{fn}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = \max_{j \in \mathcal{I}_-} s_j. \end{aligned} \tag{5.5}$$

Since the threshold always equals one of the scores [23], its computation has a simple local formula. In other words, if the highest negative score corresponds to sample $\boldsymbol{x}_{j^\star}$, then the gradient of the threshold equals $\nabla t = f(\boldsymbol{x}_{j^\star}; \boldsymbol{w})$. Therefore, the gradient of the objective function of (5.5) reads

$$\nabla L(\boldsymbol{w}) = \boldsymbol{w} + \frac{1}{n_+}\sum_{i \in \mathcal{I}_+} l'\Big(f(\boldsymbol{x}_{j^\star}; \boldsymbol{w}) - f(\boldsymbol{x}_i; \boldsymbol{w})\Big)\Big(\nabla f(\boldsymbol{x}_{j^\star}; \boldsymbol{w}) - \nabla f(\boldsymbol{x}_i; \boldsymbol{w})\Big), \tag{5.6}$$

where $j^\star$ represents the index of the negative sample with the highest score from the whole dataset. Using the same transition to the minibatches as in previous sections, we get the following formula for sampled gradient

$$\nabla \hat{L}(\boldsymbol{w}) = \boldsymbol{w} + \frac{1}{n_{\mathrm{mb},+}}\sum_{i \in \mathcal{I}_{\mathrm{mb},+}} l'\Big(f(\boldsymbol{x}_{j^\star_{\mathrm{mb}}}; \boldsymbol{w}) - f(\boldsymbol{x}_i; \boldsymbol{w})\Big)\Big(\nabla f(\boldsymbol{x}_{j^\star_{\mathrm{mb}}}; \boldsymbol{w}) - \nabla f(\boldsymbol{x}_i; \boldsymbol{w})\Big), \tag{5.7}$$

where $j^\star_{\mathrm{mb}}$ represents the index of the negative sample with the highest score from the current minibatch. As discussed in the previous section, such a choice of the decision threshold provides a lower estimate of the true threshold. To improve this approximation, we modify the idea presented in [35]. The authors of [35] suggest using delayed classification scores to compute the threshold. We used this approach in Section 3.4 to prove the convergence of stochastic gradient descent for *Pat&Mat* and *Pat&Mat-NP* formulation with a linear model. However, even in such a case, the resulting sampled gradient $\hat{t}$ is just an approximation of the true gradient t. To improve this approach, we use the fact that the threshold for *TopPush* is always equal to the negative sample with the highest score. To improve this approach, we use the fact that the threshold for *TopPush* is always equal to the negative sample with the highest score. When the weights $\boldsymbol{w}$ of model f are updated using stochastic gradient descent, the scores $\boldsymbol{s}$ usually do not change much. It is true, especially for small learning rates. Therefore, if some negative sample has the highest score, it will likely have the highest score even after the gradient step. Since we can easily track to which negative sample this highest score corresponds, we can enhance the next minibatch by this sample. This approach significantly increases the chance that the minibatch contains the negative sample with the highest score. In such a case, the sampled threshold $\hat{t}$ is not just an approximation but equals the true threshold t. The whole procedure is summarized in Algorithm 6.

Algorithm 6 *DeepTopPush* as an efficient method for maximizing accuracy at the top.

Require: Dataset $\mathcal{D}$, minibatches $\mathcal{I}_{\text{mb}}^1, \mathcal{I}_{\text{mb}}^2, \ldots, \mathcal{I}_{\text{mb}}^m$, and stepsize α^k
Initialize weights $\boldsymbol{w}^0$, $k \leftarrow 0$, and random index $j_{\text{mb}}^\star$
repeat
  Select a minibatch $\mathcal{I}_{\text{mb}}^k$
  Enhance minibatch $\mathcal{I}_{\text{mb}}^{\text{enh}} = \mathcal{I}_{\text{mb}}^k \cup \left\{j_{\text{mb}}^\star\right\}$
  Compute scores $s_i \leftarrow f(\boldsymbol{x}_i; \boldsymbol{w})$ for all $i \in \mathcal{I}_{\text{mb}}^{\text{enh}}$
  Find index of sampled thershold $j_{\text{mb}}^\star \leftarrow \arg\max\left\{s_j \;\middle|\; j \in \mathcal{I}_{\text{mb}}^{\text{enh}} \cap \mathcal{I}_-\right\}$
  Compute sampled gradient $\nabla\hat{L}$ based on $\mathcal{I}_{\text{mb}}^{\text{enh}}$ according to (5.7)
  Set $\boldsymbol{w}^{k+1} \leftarrow \boldsymbol{w}^k - \alpha^k \cdot \nabla\hat{L}$
  Set $k \leftarrow k + 1$
until stopping criterion is satisfied

Note 5.2: Other formulations

Algorithm 6 is applicable only on the *TopPush* formulation since all other formulations use more samples to compute the threshold. However, we can modify the algorithm to be usable, for example, with *TopPushK* formulation. Since *TopPushK* uses the mean of K highest negative scores as a threshold, we need to enhance the current minibatch by K indices corresponding to the K highest negative scores. However, since our goal was to use small minibatches, such an approach makes sense only for small K.

5.3 Theoretical Justification

In the previous section, we derive *DeepTopPush* method for solving the *TopPush* formulation. In Section 5.1, we discussed that the convergence proof of stochastic gradient descent requires that the sampled gradient is an unbiased estimate of the true gradient. Therefore, we are ultimately interested in the bias of the sampled gradient $\nabla\hat{L}(\boldsymbol{w})$ defined in (5.4). Recall that $j^\star$ is the index of true threshold on the whole dataset, while $j_{\text{mb}}^\star$ is the index of sampled threshold on the minibatch. We split the computation based on whether these two indices are identical or not.

Lemma 5.3

Let $j^\star$ be unique. Assume that the selection of positive and negative samples into the minibatch is independent and that the threshold is computed from negative samples while the objective is computed from positive samples. Then the conditional expectation of the sampled gradient satisfies

$$\mathbb{E}\left[\nabla\hat{L}(\boldsymbol{w}) \middle| j_{\text{mb}}^\star = j^\star\right] = \nabla L(\boldsymbol{w}).$$

Theorem 5.4

Under the assumptions of Lemma 5.3, the bias of the sampled gradient from (5.4) satisfies

$$\text{bias}(\boldsymbol{w}) = \mathbb{P}\left[j_{\text{mb}}^\star \neq j^\star\right]\left(\nabla L(\boldsymbol{w}) - \mathbb{E}\left[\nabla\hat{L}(\boldsymbol{w}) \middle| j_{\text{mb}}^\star \neq j^\star\right]\right). \tag{5.8}$$

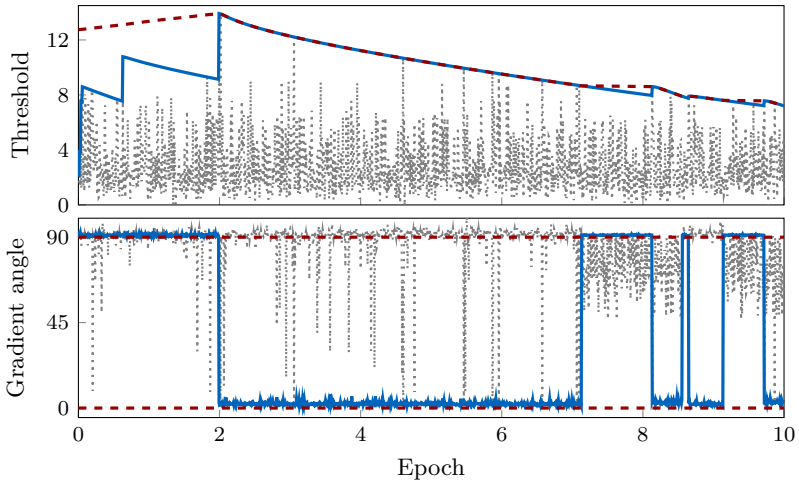


Figure 5.2: The **top** figure shows the comparison of the true threshold (red dashed line) and sampled threshold for *TopPush* (gray dotted line) and *DeepTopPush* (blue line). The **bottom** figure shows the angle between true and sampled gradients for *TopPush* (gray dotted line) and *DeepTopPush* (blue line). In this case, the red dashed lines represent 0 and 90 degrees angles. The experiment was performed on the CIFAR10 dataset with a minibatch of size 32 and 10 training epochs.

The assumptions of Theorem 5.4 holds only for *TopPush* formulation. The bias (5.8) consists of a multiplication of two terms. As we discussed in the previous sections, there are two strategies to reduce the bias:

1. **Large minibatches:** When the minibatch is large, it contains more samples, and the chance that $j^\star_{\text{mb}}$ differs from $j^\star$ decreases. This reduces the first term in (5.8). Moreover, Proposition 5.1 ensures that the difference between the sampled threshold $\hat{t}$ and the true threshold t is small. Then the difference between the true gradient (5.6) and the sampled gradient (5.7) decreases as well. This reduces the second term in (5.8).

2. **Enhanced minibatch:** The enhanced minibatch increases the chance that $j^\star_{\text{mb}}$ equals $j^\star$. This reduces the first term in (5.8).

The former strategy uses Algorithm 5 while the latter is described only for formulation (5.5) in Algorithm 6. For clarity, we use the name *DeepTopPush* for Algorithm 6 that solves formulation (5.5). Similarly, we use the name *TopPush* if Algorithm 5 is used to solve formulation (5.5).

Figure 5.2 shows the effect of the enhanced minibatch on the training. The top part of the figure compares the value of the true threshold t and its sampled version $\hat{t}$ during training. The red dashed line represents true threshold t. The full blue line shows the sampled threshold obtained by *DeepTopPush* (enhanced minibatch), while the dotted grey line the same for *TopPush*. While the sampled threshold for *TopPush* jumps wildly, for *DeepTopPush* is smooth and often equal to the true threshold. It shows the importance of the enhanced minibatch. Moreover, Theorem 5.4 implies that for *DeepTopPush* the sampled gradient is an unbiased estimate of the true gradient. It is even more pronounced in the bottom part of Figure 5.2, which shows the angle between the true gradient ∇L and the sampled gradient $\nabla \hat{L}$. This angle is essential because [47] showed that if this angle is in the interval $[0, 90)$, then gradient descent schemes converge. Which is precisely what happened for *DeepTopPush*. When the threshold is correct, the true and sampled gradients are parallel to each other, and the gradient descent moves in the correct direction.

6 Numerical Experiments

6.1 Linear Model

In this section, we present numerical results.

6.1.1 Implementational details and Hyperparameter choice

We recall that all methods fall into the framework of either (2.1) or (2.3). Since the threshold t depends on the weights $\boldsymbol{w}$, we can consider the decision variable to be only $\boldsymbol{w}$. Then to apply a method, we implemented the following iterative procedure. At iteration j, we have the weights $\boldsymbol{w}^j$ to which we compute the threshold $t^j = t(\boldsymbol{w}^j)$. Then according to (3.5), we compute the gradient of the objective and apply the ADAM descent scheme [48]. All methods were run for 10000 iterations using the stochastic gradient descent. The minibatch size was 512 except for the sigillito1989classification and Spambase datasets where the full gradient was used. All methods used the hinge surrogate (**??**). The initial point is generated randomly.

We run the methods for the following hyperparameters

$$\begin{aligned} &\beta \in \{0.0001,\ 0.001,\ 0.01,\ 0.1,\ 1,\ 10\}, \\ &\lambda \in \{0,\ 0.00001,\ 0.0001,\ 0.001,\ 0.01,\ 0.1\}, \\ &k \in \{1, 3, 5, 10, 15, 20\}. \end{aligned} \tag{6.1}$$

For *TopPushK*, *Pat&Mat* and *Pat&Mat-NP* we fixed $\lambda = 0.001$ to have six hyperparameters for all methods. For all datasets, we choose the hyperparameter which minimized the criterion on the validation set. The results are computed on the testing set which was not used during training the methods.

TopPush and τ-*FPL* were originally implemented in the dual. However, to allow for the same framework and the stochastic gradient descent, we implemented it in the primal. These two approaches are equivalent.

6.1.2 Dataset description and Performance criteria

For the numerical results, we considered 10 datasets summarized in Table 6.1. They can be downloaded from the UCI repository. sigillito1989classification [49] and Spambase are small, baldi2016parameterized [50] contains a large number of samples while guyon2005result [51] contains a large number of features. We also considered six visual recognition datasets: MNIST, FashionMNIST, CIFAR10, CIFAR20, CIFAR100 and SVHN2. MNIST and FashionMNIST are grayscale datasets of digits and fashion items, respectively. CIFAR100 is a dataset of coloured images of items grouped into 100 classes. CIFAR10 and CIFAR20 merge these classes into 10 and 20 superclasses, respectively. SVHN2 contains coloured images of house numbers. As Table 6.1 shows, these datasets are imbalanced.

Each of the visual recognition datasets was converted into ten binary datasets by considering one of the classes $\{0, \ldots, 9\}$ as the positive class and the rest as the negative class. The

experiments were repeated ten times for each dataset from different seeds, which influenced the starting point and minibatch creation. We use tpr@fpr as the evaluation criterion. This describes the true-positive rate at a prescribed true-negative rate, usually of 1% or 5%. For the linear classifier $\boldsymbol{w}^\top \boldsymbol{x} - t$, it selects the threshold t so that the desired true-negative rate is satisfied and then computes the true-positive rate for this threshold.

Dataset	m	Train		Validation		Test	
		n	$\frac{n_+}{n}$	n	$\frac{n_+}{n}$	n	$\frac{n_+}{n}$
Ionosphere	34	175	36.0%	88	36.4%	88	35.2%
Spambase	57	2 300	39.4%	1 150	39.4%	1 151	39.4%
Gisette	5 000	1 000	50.0%	1 500	50.0%	500	50.0%
Hepmass	28	5 250 000	50.0%	1 750 000	50.0%	3 500 000	50.0%
MNIST	$28 \times 28 \times 1$	44 999	11.2%	15 001	11.2%	10 000	11.4%
FashionMNIST	$28 \times 28 \times 1$	45 000	10.0%	15 000	10.0%	10 000	10.0%
CIFAR10	$32 \times 32 \times 3$	37 500	10.0%	12 500	10.0%	10 000	10.0%
CIFAR20	$32 \times 32 \times 3$	37 500	5.0%	12 500	5.0%	10 000	5.0%
CIFAR100	$32 \times 32 \times 3$	37 500	1.0%	12 500	1.0%	10 000	1.0%
SVHN2	$32 \times 32 \times 3$	54 944	18.9%	18 313	18.9%	26 032	19.6%

Table 6.1: Structure of the used datasets. The training, validation and testing sets show the number of features m, samples n and the fraction of positive samples $\frac{n_+}{n}$.

6.1.3 Numerical results

Figure 6.1 presents the standard ROC (receiver operating characteristic) curves on selected datasets. Since all methods from this paper are supposed to work at low false-positive rates, the x axis is logarithmic. Both figures depict averages over ten runs with different seeds. The left column depicts CIFAR100 while the right one Hepmass. These are the two more complicated datasets. We selected four representative methods: *Pat&Mat* and *Pat&Mat-NP* as our methods and *TopPush* and τ*-FPL* as state-of-the-art methods. Even though all methods work well, *Pat&Mat-NP* seems to outperform the remaining methods on most levels of false-positive rate.

While Figure 6.1 gave a glimpse of the behaviour of methods, Figures 6.2 and 6.3 provide a statistically more sound comparison. It employs the Nemenyi post hoc test for the Friedman test recommended in [52]. This test compares if the mean ranks of multiple methods are significantly different.

We consider 14 methods (we count different values of τ as different methods) as depicted in this table. For each dataset mentioned in Section 6.1.2 and each method, we evaluated the fpr@tpr metric and ranked all methods. Rank 1 refers to the best performance for given criteria, while rank 14 is the worst. The x-axis shows the average rank over all datasets. The Nemenyi test computes the critical difference. If two methods are within their critical difference, their performance is not deemed to be significantly different. Black wide horizontal lines group such methods.

From this figure and table, we make several observations:

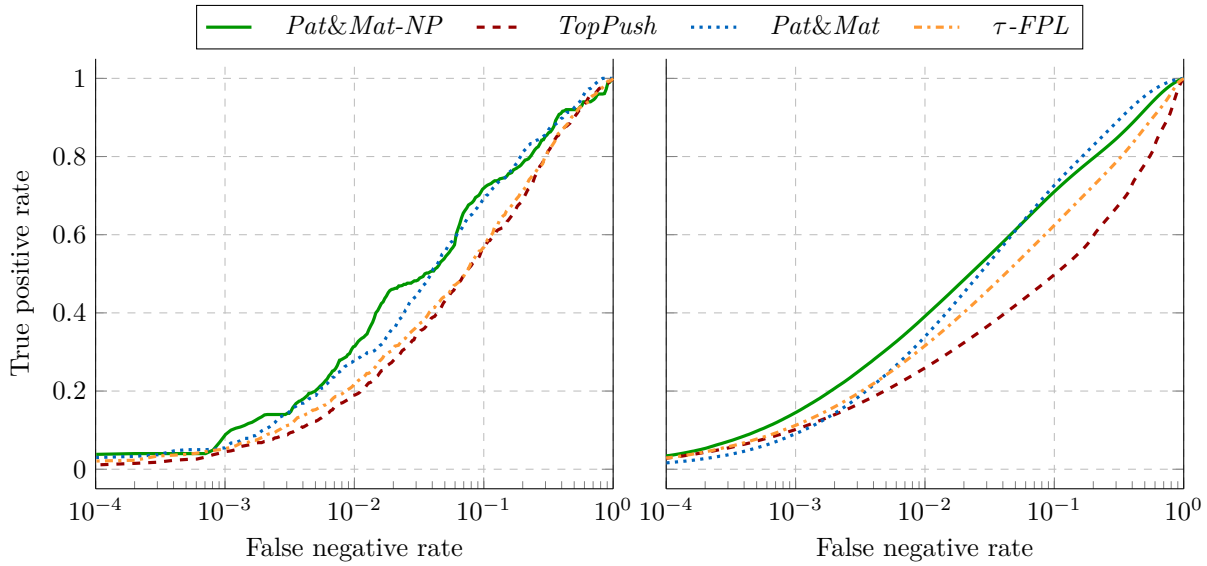


Figure 6.1: ROC curves (with logarithmic x axis) on CIFAR100 (left) and Hepmass (right).

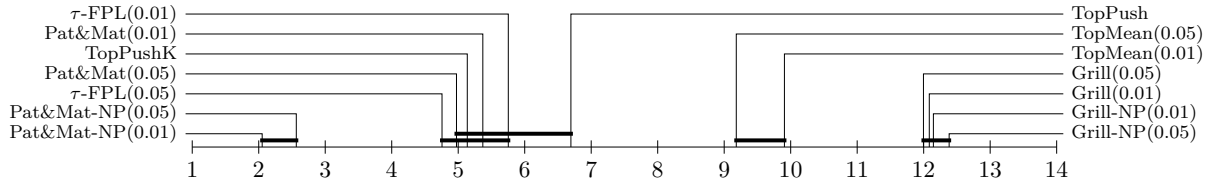


Figure 6.2: Critical difference (CD) diagrams (level of importance 0.05) of the Nemenyi post hoc test for the Friedman test. Each diagram shows the mean rank of each method, with rank 1 being the best. Black wide horizontal lines group together methods with the mean ranks that are not significantly different. The critical difference diagrams were computed for mean rank averages over all datasets of the tpr@fpr ($\tau = 0.01$) metric.

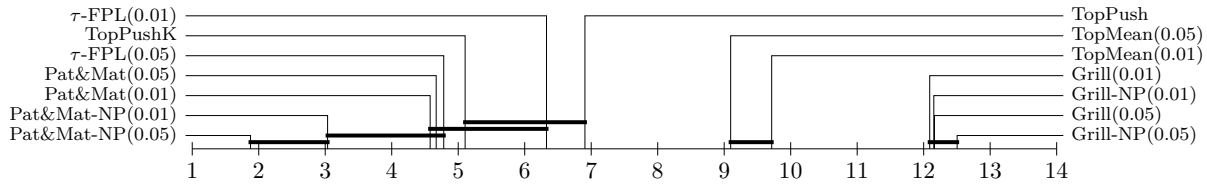


Figure 6.3: Critical difference (CD) diagrams (level of importance 0.05) of the Nemenyi post hoc test for the Friedman test. Each diagram shows the mean rank of each method, with rank 1 being the best. Black wide horizontal lines group together methods with the mean ranks that are not significantly different. The critical difference diagrams were computed for mean rank averages over all datasets of the tpr@fpr ($\tau = 0.05$) metric.

- *TopPushK* (rank 5.1) provides a slight improvement over *TopPush* (rank 6.7) even though this improvement is not statistically significant as both methods are connected by the black line in both Figures 6.2 and 6.3.
- Neither *Grill* (ranks 12.0 and 12.1) nor *Grill-NP* (ranks 12.1 and 12.4) perform well. We believe this happened due to the lack of convexity as indicated in Theorem 3.2 and the discussion after that.
- *TopMeanK* (ranks 9.2 and 9.9) does not perform well either. Since the thresholds τ are small, then $\boldsymbol{w} = 0$ is the global minimum as proved in Corollary 3.7.

- *Pat&Mat-NP* (rank 2.1 and 2.6) seems to outperform other methods.
- *Pat&Mat* (ranks 5.0 and 5.4), τ-*FPL* (ranks 4.8 and 5.8) and *TopPushK* (rank 5.1) perform similarly. Since they are connected, there is no statistical difference between their behaviours.
- *Pat&Mat-NP* at level 0.01 (rank 2.1) outperforms *Pat&Mat-NP* at level 0.05 (rank 2.6) for $\tau = 0.01$. *Pat&Mat-NP* at level 0.05 (rank 1.9 in Figure 6.3) outperforms *Pat&Mat-NP* at level 0.01 (rank 3.0 in Figure 6.3) for $\tau = 0.05$. This should be because these methods are optimized for the corresponding threshold. For τ-*FPL* we observed this behaviour for Figure 6.3 but not for Figure 6.2.

Figure 6.4 provides a similar comparison. Both axes are sorted from the best (left) to the worst (right) average ranks. The numbers in the graph show the p-value for the pairwise Wilcoxon signed-rank test, where the null hypothesis is that the mean tpr@fpr of both methods is the same. Even though Figure 6.2 employs a comparison of mean ranks and Figure 6.4 a pairwise comparison of fpr@tpr, the results are almost similar. Methods grouped by the black line in the former figure usually show a large p-value in the latter figure.

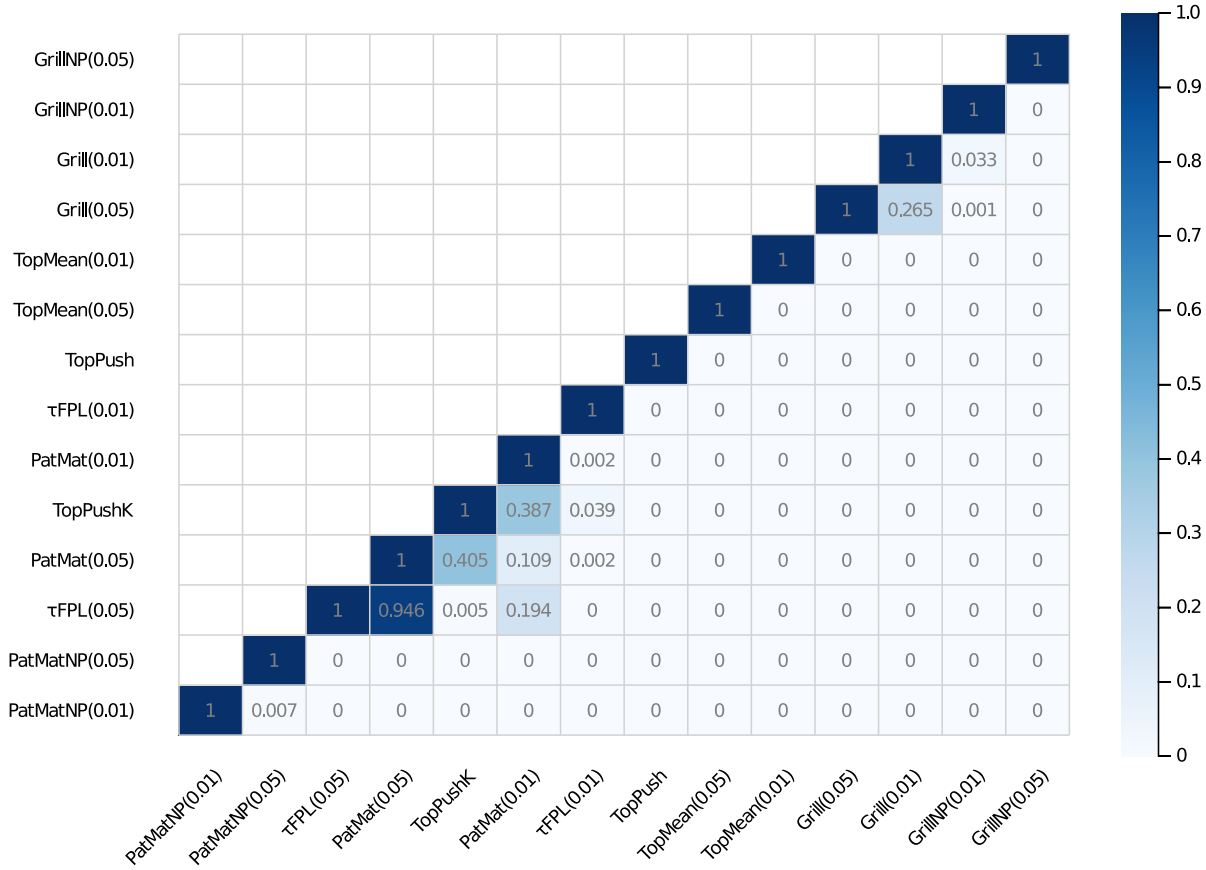


Figure 6.4: The p-value for the parwise Wilcoxon signed-rank test, where the null hypothesis is that the mean tpr@fpr(0.01) of both methods is the same. The methods are sorted by mean rank (left = better).

Table 6.2 investigates the impact of $\boldsymbol{w} = 0$ as a potential global minimum. Each method was optimized for six different values of hyperparameters. The table depicts the condition under which the final value has a lower objective than $\boldsymbol{w} = 0$. Thus, ✓ means that it is always better while ✗ means that the algorithm made no progress from the starting point $\boldsymbol{w} = 0$. The latter case implies that $\boldsymbol{w} = 0$ seems to be the global minimum. We make the following observations:

- *Pat&Mat* and *Pat&Mat-NP* are the only methods which succeeded at every dataset for some hyperparameter. Moreover, for each dataset, there was some β_0 such that these methods were successful if and only if $\beta \in (0, \beta_0)$. This is in agreement with Theorem 3.8.
- *TopMeanK* fails everywhere which agrees with Corollary 3.7.
- Figure 3.2 states that the methods from Section 2.3 has a higher threshold than their Neyman-Pearson variants from Section 2.4. This is documented in the table as the latter have a higher number of successes.

Method		Ionosphere	Hepmass	FashionMNIST	CIFAR100
TopPush		✓	✗	✓	✗
TopPushK		✓	✗	✓	✗
Grill	$\tau = 0.01$	✗	✗	✗	✗
	$\tau = 0.05$	✗	✗	✗	✗
Pat&Mat	$\tau = 0.01$	✓	$\beta \leq 0.1$	$\beta \leq 1$	$\beta \leq 1$
	$\tau = 0.05$	✓	$\beta \leq 1$	✓	✓
TopMeanK	$\tau = 0.01$	✗	✗	✗	✗
	$\tau = 0.05$	✗	✗	✗	✗
Grill-NP	$\tau = 0.01$	✗	✗	✗	✗
	$\tau = 0.05$	✗	✗	✗	✗
Pat&Mat-NP	$\tau = 0.01$	✓	$\beta \leq 1$	✓	$\beta \leq 1$
	$\tau = 0.05$	✓	✓	✓	$\beta \leq 1$
τ-FPL	$\tau = 0.01$	✓	✗	✓	✗
	$\tau = 0.05$	✓	✓	✓	$\lambda \leq 0.001$

Table 6.2: Necessary hyperparameter choice for the solution to have a better objective than zero. ✓ means that the solution was better than zero for all hyperparameters while ✗ means that it was worse for all hyperparameters.

6.2 Dual

In this section, we present numerical results. All codes were implemented in the Julia language [53] and are available online.[1]

6.2.1 Performance criteria

For the evaluation of numerical experiments, we use precision and recall. For a threshold t they are defined by

$$\text{precision} = \frac{\sum_{i=1}^{n_+} [w^\top x_i^+ - t]}{\sum_{i=1}^{n} [w^\top x_i - t]}, \qquad \text{recall} = \frac{1}{n_+} \sum_{i=1}^{n_+} [w^\top x_i^+ - t]. \tag{6.2}$$

[1] All codes are available at https://github.com/VaclavMacha/ClassificationOnTop_new.jl

We also use the Precision-Recall (PR) curve that are commonly used for unbalanced data [54] and precision at a certain level of recall which we denote by Precision@Recall.

6.2.2 Hyperparameter choice

In Section 4.2 we introduced Algorithm 4 for solving dual problems (**??**, **??**). We let it run for 20000 `repeat` loops, which corresponds to 40000 updates of coordinates of (α, β). We use the linear and Gaussian kernels defined by

$$k_{\text{lin}}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^\top \boldsymbol{y}, \tag{6.3}$$

$$k_{\text{gauss}}(\boldsymbol{x}, \boldsymbol{y}) = \exp\{-\sigma \left\| \boldsymbol{x} - \boldsymbol{y} \right\|_2^2\} \tag{6.4}$$

and the truncated quadratic loss (**??**) with $\vartheta = 1$ as a surrogate.

The classifiers were trained on the training set. We selected the optimal hyperparameter from

$$\tau \in \{0.01,\ 0.05,\ 0.1\}, \quad K \in \{5,\ 10\}, \quad C \in \{0.1,\ 1,\ 10\}, \quad \sigma \in \{0.01,\ 0.05\}$$

which gave the best performance on the validation set. All presented result are shown on the testing set which was not part of the training process.

Dataset	y^+	d	Train		Validation		Test	
			n	$\frac{n_+}{n}$	n	$\frac{n_+}{n}$	n	$\frac{n_+}{n}$
Ionosphere	–	34	176	64.2%	87	64.4%	88	63.6%
Spambase	–	57	2 301	39.4%	1 150	39.4%	1 150	39.4%
WhiteWineQuality	7, 8, 9	11	2 449	21.6%	1 224	21.7%	1 225	21.6%
RedWineQuality	7, 8	11	800	13.5%	400	13.8%	399	13.5%
Fashion-MNIST	0	784	50 000	10.0%	10 000	10.0%	10 000	10.0%

Table 6.3: Summary of the used datasets. It shows which original labels y^+ were selected as the positive class, the number of features d, samples n, and the fraction of positive samples $\frac{n_+}{n}$.

6.2.3 Dataset description

For numerical experiments, we consider the FashionMNIST dataset [55] and four smaller datasets from the UCI repository [56]: sigillito1989classification [49], Spambase, WhiteWineQuality [57] and RedWineQuality [57]. Datasets that do not contain testing set were randomly divided into a training (50%), validation (25%) and testing (25%) sets. For datasets that contain a testing set, the training set was randomly divided into a training and a validation set, where the validation set has the same size as the testing set. FashionMNIST dataset was converted to binary classification tasks by selecting class with label 0 as the positive class and the rest as the negative class. All datasets are summarized in Table 6.3.

6.2.4 Experiments

In Figure 6.5 we present the PR curves for all methods with two different kernels evaluated on the FashionMNIST dataset. The left column corresponds to the linear kernel (6.3) while the right one to the Gaussian kernel (6.4) with $\sigma = 0.01$. The nonlinear Gaussian kernel significantly outperforms the linear kernel. This will be confirmed later in Table 6.4 where we present a comparison from multiple datasets.

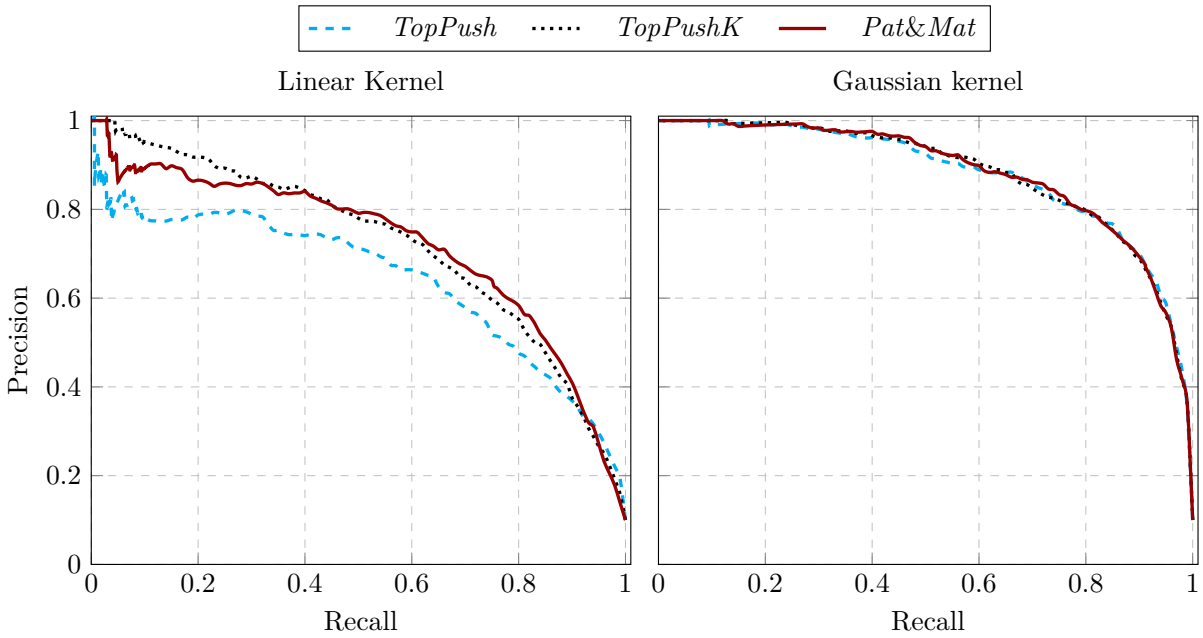


Figure 6.5: PR curves for all methods and FashionMNIST dataset. The left column corresponds to the linear kernel (6.3) and the right column corresponds to the Gaussian kernel (6.4).

For a better illustration of how the methods from Figure 6.5 work, we present density estimates of scores $\boldsymbol{s}$ from (**??**). High scores predict positive labels while low scores predict negative labels. The rows of Figure 6.6 depict the linear (6.3) and the Gaussian kernels (6.4) with $\sigma = 0.01$ while each column corresponds to one method. The black vertical lines depict the top 5%-quantile of all scores (on the testing set). Since a smaller overlap of scores of samples with positive and negative labels implies a better separation, we deduce the benefit of the Gaussian over the linear kernel.

In Table 6.4 we present the precision of all methods across all datasets from Table 6.3. For each dataset, we trained each method and computed precision at certain levels of recall. The depicted values are averages over all datasets. For each kernel and each level of recall, the best precision is highlighted in light green. Moreover, the best overall precision for each level of recall is depicted in dark green. We can make several observations from Table 6.4:

- All methods perform better with the Gaussian kernels than with the linear kernel.
- *TopPush* and *TopPushK* perform better for sufficiently small recall. This happened because they consider the threshold to be the maximal K negative scores and small recall corresponds to high threshold. However, for the same reason, *TopPush* is not robust.
- *Pat&Mat* is the best for all kernels if the recall is sufficiently large. The reason is again the form of the decision threshold.

In Figure 6.7, we investigate the convergence of methods. In each column, we show the convergence of primal and dual problems for one method. To solve the primal problem, we use the gradient method proposed in [30]. For the dual problem, we use our Algorithm 4. Since [30] considers only linear kernels, we present them. Moreover, since the computation of the objective is expensive, the results are presented for the sigillito1989classification dataset. We can see that *TopPush* and *TopPushK* converge to the same objective for primal and dual problems. This means that the problem was solved to optimality. However, there is a little gap between optimal solution of primal and dual problems for *Pat&Mat*.

Finally, Table 6.5 depicts the time comparison for all methods and all datasets. It shows the average time in milliseconds needed for one `repeat` loop in Algorithm 4. The time is relatively

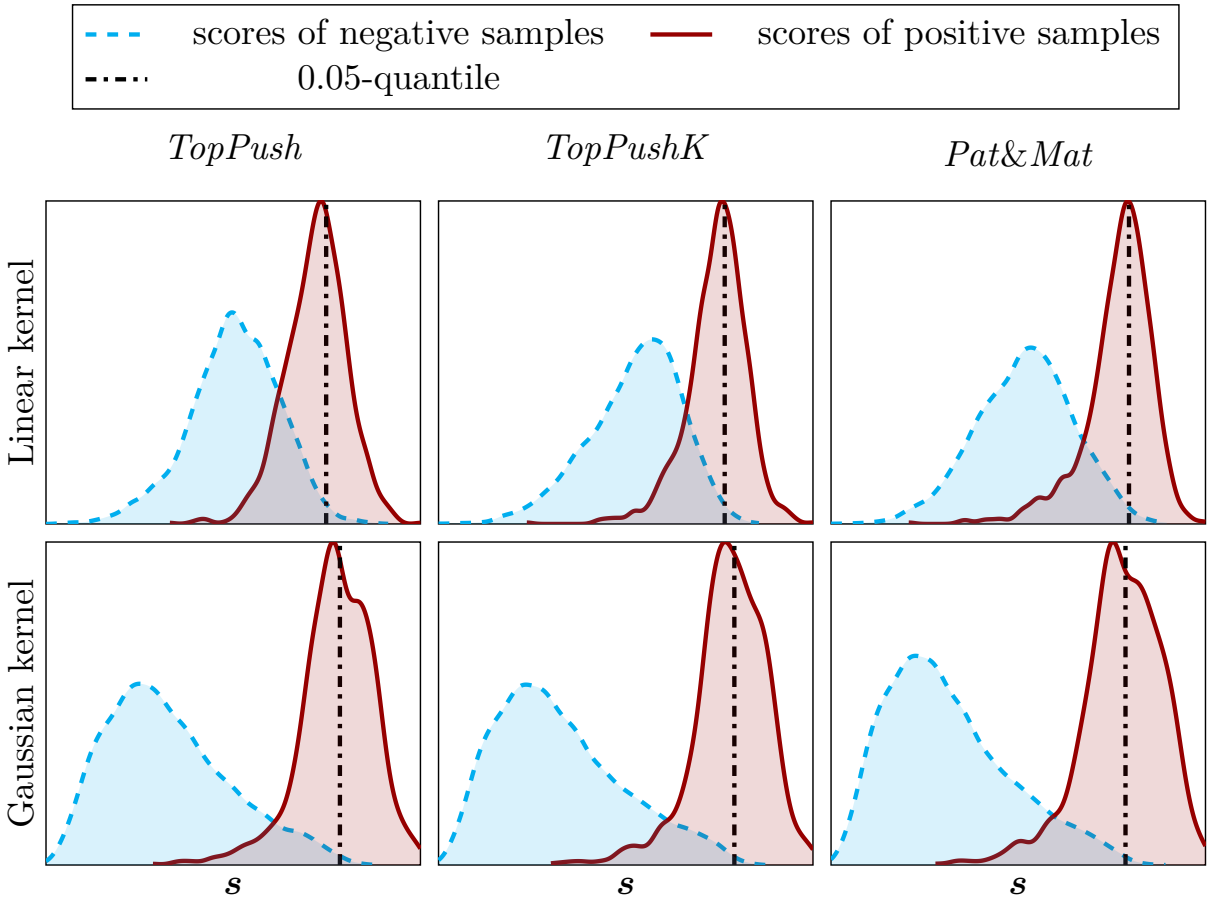


Figure 6.6: Density estimates for scores corresponding to samples with positive and negative labels for the FashionMNIST dataset.

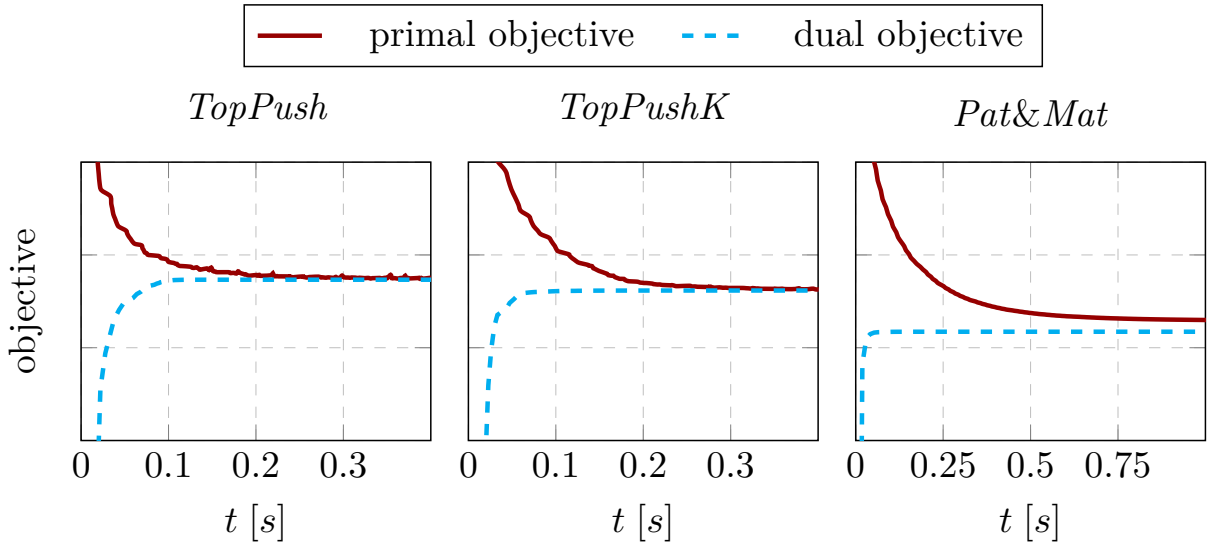


Figure 6.7: Convergence of the objectives for the primal (red line) and dual (blue line) problems for the sigillito1989classification dataset with linear kernel.

stable and for most of the datasets it is below one millisecond. Since we run all experiments for 20000 `repeat` loops, the evaluation of one method with one hyperparameter setting takes a few seconds for smaller datasets and approximately 7 minutes for FashionMNIST. The average

	Method		Precision@Recall					
			0.05	0.1	0.2	0.4	0.6	0.8
Linear kernel	*TopPush*		79.83	64.27	65.55	61.85	57.89	51.83
	TopPushK	$K = 5$	73.96	65.41	64.82	60.28	56.94	50.52
		$K = 10$	60.63	61.97	59.69	56.89	54.40	49.83
	Pat&Mat	$\tau = 0.01$	63.67	60.30	58.74	57.75	53.32	48.42
		$\tau = 0.05$	54.05	60.91	63.32	55.24	52.55	48.30
		$\tau = 0.1$	57.02	61.24	62.49	63.11	59.91	52.14
Gaussian kernel	*TopPush*		97.50	86.06	81.28	76.15	71.13	60.17
	TopPushK	$K = 5$	92.50	87.56	85.31	78.47	70.77	57.10
		$K = 10$	89.50	87.56	83.15	79.09	71.88	59.27
	Pat&Mat	$\tau = 0.01$	89.65	89.11	86.75	80.77	75.44	65.95
		$\tau = 0.05$	80.77	81.28	85.74	82.92	74.91	65.04
		$\tau = 0.1$	81.30	84.14	82.58	83.12	77.82	66.50

Table 6.4: The precision of all methods averaged across all datasets from Table 6.3. Each column represents precision at a certain level of recall. Light green depicts the best method for the given kernel and dark green depicts the best overall method.

time for one Δ_l in step 6 in Algorithm 4 took between $1.7 \cdot 10^{-7}$ and $3.1 \cdot 10^{-7}$ seconds for each methods. It is almost the same for all datasets, which corresponds to the fact that the complexity of step 6 is independent of the size of the dataset. Note that in all experiments we used precomputed kernel matrix $\mathbb{K}$ saved on the hard drive and not in memory.

6.3 Neural Networks

This section presents numerical results for *DeepTopPush*. Table **??** shows that it is similar to *Pat&Mat-NP*. While the former maximizes the number of positives above the largest negative, while the latter maximizes the number of positives above the n_τ-largest negative. The former may be understood as requiring no false-positives, while the latter allows for false positive rate τ.

Section **??** showed that we can use large minibatches to obtain good results for *Pat&Mat-NP* for small fractions of top samples τ. Section **??** showed that *DeepTopPush* works well even with small minibatches if we track the threshold by enhancing the minibatch by one sample. We present numerical comparisons in several sections, each with a different purpose. Comparison with the prior art *TFCO* and *Ap-Perf* is performed on several visual recognition datasets and shows that *DeepTopPush* outperforms other methods. Then we present two real-world applications. The first one shows that *DeepTopPush* can handle ranking problems. The second one presents results on a complex malware detection problem. Finally, we show similarities between *DeepTopPush* and *Pat&Mat-NP* and explain why enhancing the minibatch in Algorithm **??** works.

	Dataset	*TopPush*	*TopPushK*	*Pat&Mat*
One repeat loop [ms]	Ionosphere	0.04 ± 0.00	0.03 ± 0.00	0.03 ± 0.00
	Spambase	0.56 ± 0.02	0.49 ± 0.01	0.50 ± 0.01
	WhiteWineQuality	0.62 ± 0.03	0.53 ± 0.01	0.54 ± 0.01
	RedWineQuality	0.17 ± 0.01	0.14 ± 0.01	0.15 ± 0.01
	Fashion-MNIST	17.16 ± 0.74	15.95 ± 0.14	15.54 ± 0.80

Table 6.5: The average time with standard deviation (in milliseconds) for one `repeat` loop in Algorithm 4. The average time for one Δ_l in step 6 in Algorithm 4 took between $1.7 \cdot 10^{-7}$ and $3.1 \cdot 10^{-7}$ seconds for each methods.

6.3.1 related work deep

Two approaches for solving (**??**) exist. The first approach considers the threshold constraint as it is, while the second approach uses heuristics to approximate it. In the first approach, Acc@Top [23] argues that the threshold equals one of the scores. They fix the index of a sample and solve as many optimization problems as there are samples. [25, 30, 58] write the threshold as a constraint and replace both the objective and the constraint via surrogates. [25] uses Lagrange multipliers to obtain a minimax problem, [34] implicitly removes the threshold as an optimization variable and uses the chain rule to compute the gradient while [33] solves an SVM-like dual formulation with kernels. [3] uses the same formulation but applies surrogates only to the objective and recomputes the threshold after each gradient step. *TFCO* [59] solves a general class of constrained problems via a minimax reformulation. In the second approach, SoDeep [60] or SmoothI [61] use the fact that the threshold may be easily computed from sorted scores. They approximate the sorting operator by a network trained on artificial data. *Ap-Perf* [62] considers a general metric and hedges against the worst-case perturbation of scores. The authors argue that the problem is bilinear in scores and use duality arguments. However, the bilinearity is lost when optimizing with respect to the weights of the original network.

6.3.2 Dataset description and Computational setting

We consider the following image recognition datasets: FashionMNIST [55], CIFAR100 [63], SVHN2 [64] and ImageNet [65]. These datasets were converted to binary classification tasks by selecting one class as the positive class and the rest as the negative class. ImageNet merged turtles and non-turtles. We also consider the 3A4 dataset [66] with molecules and their activity levels. Finally, malware analysis reports of executable files were provided by a cybersecurity company. This is an extremely tough dataset as individual samples are JSON files whose size ranges from 1kB to 2.5MB. Moreover, they contain different features, and their features may have variable lengths. Table 6.6 summarizes the used datasets. The Malware Detection dataset was represented by JSONs, which contain varying number of features. Moreover, many features are not scalar but have some hierarchical structure as well.

We use truncated quadratic loss $l(z) = (\max\{0, 1+z\})^2$ as the surrogate function and $\tau = \frac{1}{n_-}$ and $\tau = 0.01$. This first one computes the true positive rate above the second highest-ranked negative, while the latter allows for the false positive rate of 1%. All algorithms were run for 200 epochs on an NVIDIA P100 GPU card with balanced minibatches of 32 samples. The only exception was Malware Detection, which was run on a cluster in a distributed manner, and where the minibatch size was 20000. For the evaluation of numerical experiments, we use

Dataset	d	Train		Test		Licence
		n	$\frac{n_+}{n}$	n	$\frac{n_+}{n}$	
FashionMNIST	$28 \times 28 \times 1$	60 000	10.00%	10 000	10.00%	MIT
CIFAR100	$32 \times 32 \times 3$	50 000	1.00%	10 000	1.00%	not specified
SVHN2 extra	$32 \times 32 \times 3$	604 388	17.28%	26 032	19.59%	not specified
ImageNet	62720×1	1 281 167	0.51%	50000	0.50%	registration
3A4	9491×1	37 241	0.98%	37 241	1.07%	CC BY 4.0
Malware Detection	variable	6 580 166	87.22%	800 346	91.80%	proprietary

Table 6.6: Summary of the used datasets with the number of features d, number of samples n and the fraction of positive samples $\frac{n_+}{n}$ in the training set.

the standard receiver operating characteristic (ROC) curve. All results are computed from the test set. All codes were implemented in the Julia language [53]. The network structure was the same for all methods; we describe them in the online appendix.

6.3.3 Used network architecture

For 3A4, we preprocessed the input with 9491 into a 100-dimensional input by PCA. Then we used two dense layers of size 100×50 and 50×25 with batch-normalization after these layers. The last layer was dense.

For FashionMNIST, we used a network alternating two hidden convolutional layers with two max-pooling layers finished with a dense layer. The convolutional layers used kernels 5×5 and had 20 and 50 channels, respectively. For CIFAR100 and SVHN2, we increased the number of hidden and max-pooling layers from two to three. The convolutional layers used kernels 3×3 and had 64, 128, and 128 channels, respectively. A more detailed description can be found in our codes online. We are fully aware that these architectures are suboptimal. Since the accuracy at the top needs to select only a few relevant samples and the rest of the dataset's performance is irrelevant, such a network can be used. Moreover, using a simpler network has the advantage of faster experiments.

For ImageNet, we merged all turtles into the positive class and all non-turtles into the negative class. Then we used the pre-trained EfficientNet B0, where we replaced the last dense layer with 1000 outputs by a dense layer into a scalar output.

6.3.4 Comparison with prior art

We compare our methods with *BaseLine*, which uses the weighted cross-entropy. Moreover, we use two prior art methods which have codes available online, namely *TFCO* [59, 67] and *Ap-Perf* [62]. We did not implement the original TopPush because its duality arguments restrict the classifiers to only linear ones. Table 6.7 shows the time requirement per epoch. All methods besides *Ap-Perf* have similar time requirements, while *Ap-Perf* is much slower. This difference increases drastically when the minibatch size increases, as noted in [62]. We do not present the results for SVHN for *Ap-Perf* because it was too slow and for *TFCO* because we encountered a TensorFlow memory error. All these methods are designed to maximize true-positives when the false positive rate is at most τ. This is the same as for *Pat&Mat-NP*.

Table 6.8 shows the true positive rate (tpr) above the second-largest negative and at the prescribed false positive rate (fpr) $\tau = 0.01$. Using the second-largest negative, which corresponds

Method	FashionMNIST	CIFAR100	SVHN
BaseLine	4.4s	5.1s	62.8s
DeepTopPush	4.8s	5.6s	66.6s
Pat&Mat-NP	4.8s	5.6s	66.6s
TFCO	7.2s	6.5s	-
Ap-Perf	95.3s	81.2s	-

Table 6.7: Time requirements per epoch for investigated methods for minibatches of size $n_{\text{mb}} = 32$.

	Dataset	BaseLine	*DeepTopPush*	*Pat&Mat-NP*	*TFCO*	*Ap-Perf*
tpr@fpr $\tau = 1/n_-$	FashionMNIST	5.06 ± 1.41	27.30 ± 5.91	22.21 ± 5.62	11.30 ± 3.44	9.90
	CIFAR100	1.70 ± 0.46	14.40 ± 5.44	8.10 ± 3.45	7.70 ± 2.28	5.00
	3A4	2.58 ± 0.61	5.61 ± 1.70	3.79 ± 0.90	3.03 ± 1.52	3.03
	SVHN	6.51 ± 1.37	12.21 ± 5.39	12.07 ± 4.41	—	—
tpr@fpr $\tau = 0.01$	FashionMNIST	63.14 ± 1.39	75.37 ± 1.18	74.11 ± 1.00	73.27 ± 2.92	64.60
	CIFAR100	49.40 ± 4.90	70.20 ± 2.14	66.30 ± 2.33	67.30 ± 1.79	65.00
	3A4	57.80 ± 0.35	60.08 ± 3.35	65.91 ± 0.59	54.55 ± 10.22	63.64
	SVHN	84.72 ± 0.84	91.05 ± 1.45	91.07 ± 0.30	—	—

Table 6.8: The true positive rates (in %) at two levels of false positive rates averaged across ten indepenedent runs with standard deviation. The best methods are highlighted.

to $\tau = \frac{1}{n_-}$, allows for one outlier. The results are averaged over ten independent runs except for *Ap-Perf*, which is too slow. The best result for each metric (in columns) is highlighted. All methods are better than *BaseLine*. This is not surprising as all these methods are designed to work well for low false positive rates. *DeepTopPush* outperforms all other methods at the top, while it performs well at the low fpr of $\tau = 0.01$. There *Pat&Mat-NP*, which also falls into our framework, performs well. Both these methods outperform the state of the art methods.

Figure 6.9 **A)** shows the ROC curves on CIFAR100 averaged over ten independent runs. We use the logarithmic x axis to highlight low fpr modes. *DeepTopPush* performs significantly the best again whenever the false positive rate is smaller than 0.01.

As a further test, we performed a simple experiment on ImageNet. We modified the pre-trained EfficientNet B0 [68] by removing the last dense layer and adding another dense layer with one output. Then we retrained the newly added layer to perform well at the top. The original EfficientNet achieved 68.0% at the top, while *DeepTopPush* achieved 70.0% for the same metric. This shows that *DeepTopPush* can provide better accuracy at the top than pre-trained networks.

6.3.5 Application to ranking

The 3A4 dataset contains information about activity levels of approximately 50000 molecules, each with about 10000 descriptors. The activity level corresponds to the usefulness of the molecule for creating new drugs. Since medical scientists can focus on properly investigating only a small number of molecules, it is important to select a small number of molecules with high activity.

We converted the continuous activity level into binary by considering a threshold on the activity. Since the input is large-dimensional, and there is no spatial structure to use convolutional neural networks, we used PCA to reduce the dimension to 100. Then we created a network with two hidden layers and applied *DeepTopPush* to it. The test activity was evaluated at the continuous (and not binary level). Table 6.8 shows again the results at the top. *DeepTopPush* outperforms other methods. Figure 6.8 shows that high scores (output of the network) indeed correspond to high activity. Thus, even though the problem was "binarized" and its dimension reduced, our algorithm was able to select a small number of molecules with high activity levels. These molecules can be used for further manual (expensive) investigation.

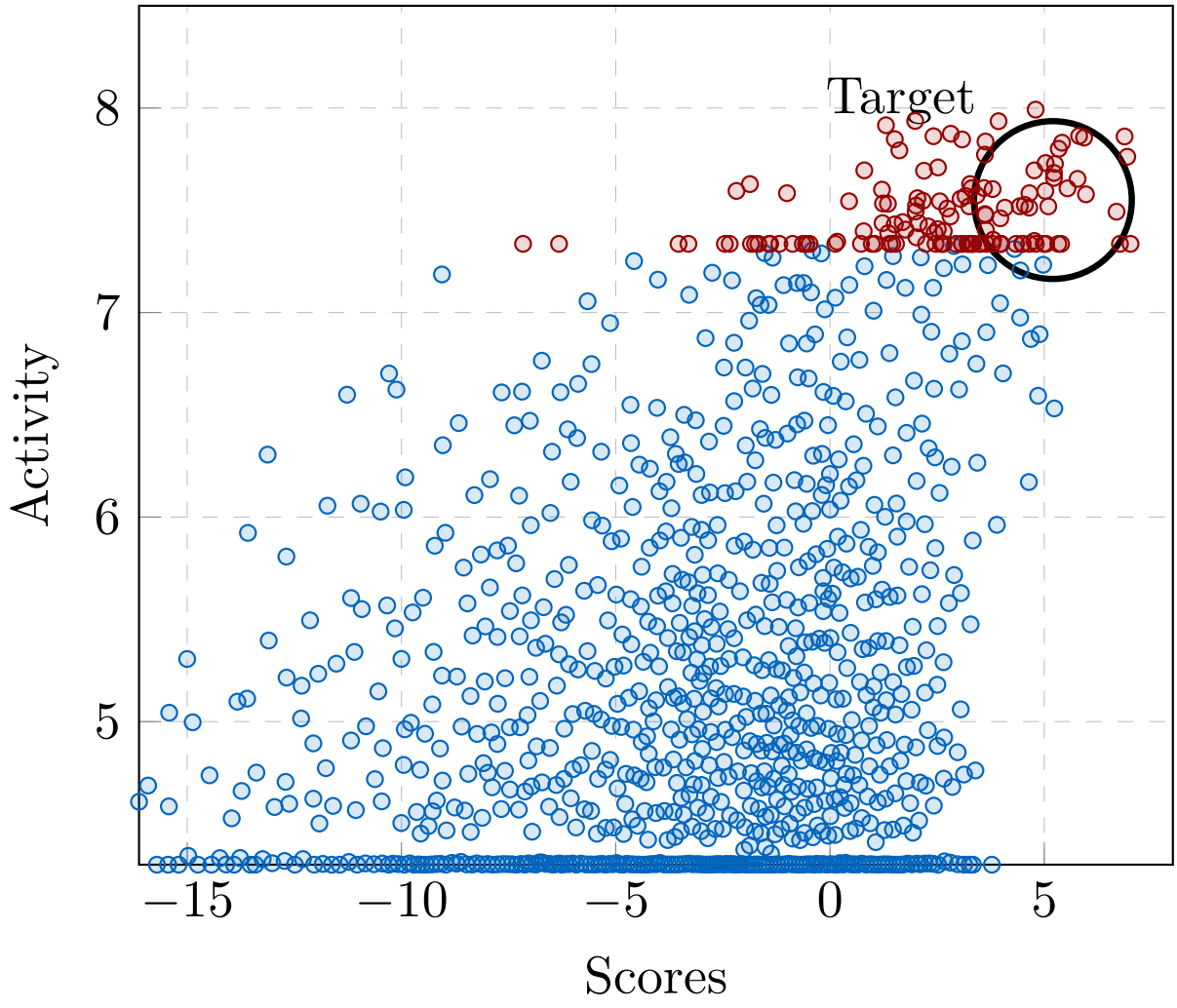


Figure 6.8: Results for the 3A4 dataset. The goal was to assign large scores to a few molecules with high activity (scores on top-right are preferred).

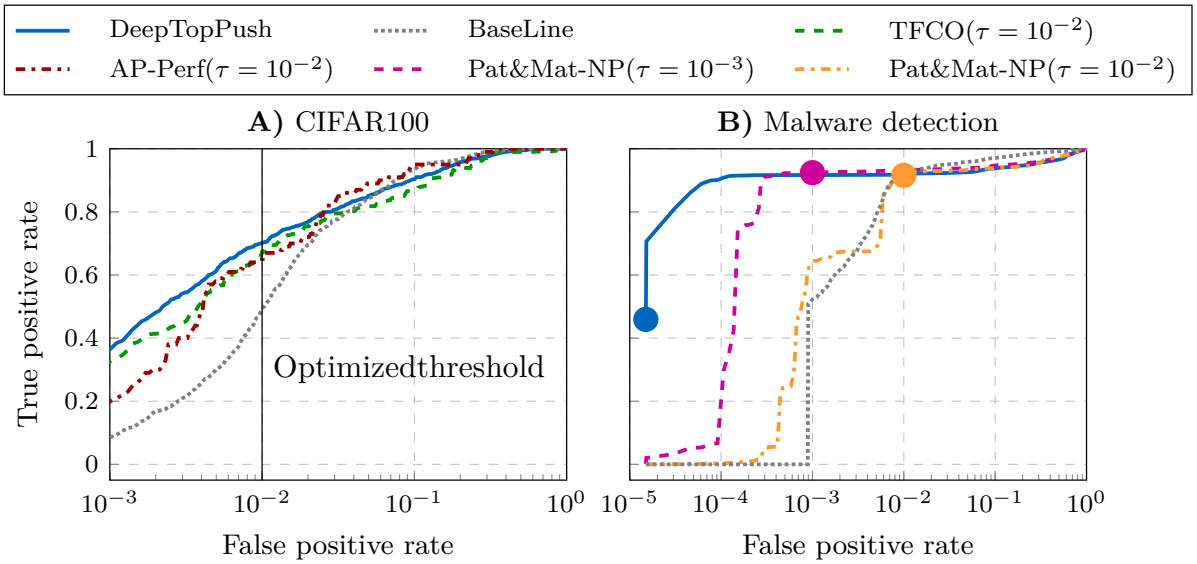


Figure 6.9: **A)** ROC curves averaged over ten runs on the CIFAR100 dataset. **B)** ROC curve for Malware Detection dataset. The circles show the thresholds the methods were optimized for.

6.3.6 Real-world application

This section shows a real-world application of the accuracy at the top. A renowned cybersecurity company provided malware analysis reports of executable files. Its structure is highly complicated because each sample has a different number of features, and features may have a complicated structure, such as a list of ports to which the file connects. This is in sharp contrast with standard datasets, where each sample has the same number of features, and each feature is a real number. We processed the data by a public implementation of hierarchical multi-instance learning (HMIL) [69]. Then we applied *DeepTopPush* and *Pat&Mat-NP* at $\tau = 10^{-3}$ and $\tau = 10^{-2}$. The latter maximizes the true positives rate when the false positive rate is at most τ. The minibatch size was 20000, which allowed us to obtain precise threshold estimates and unbiased sampled gradients due to Section **??**.

Figure 6.9 **B)** shows the performance on the test set. *DeepTopPush* is again the best at low false positive rates. This is extremely important in cybersecurity as it prevents false alarms for malware. Even at the extremely low false positive rate $\tau = 10^{-5}$, our algorithm correctly identified 46% of malware. The circles denote the thresholds for which the methods were optimized. *DeepTopPush* should have the best performance at the leftmost point, *Pat&Mat-NP* ($\tau = 10^{-3}$) at $\tau = 10^{-3}$ and similarly *Pat&Mat-NP* ($\tau = 10^{-2}$).

7 Conclusion

7.1 Linear Model

In this paper, we achieved the following results:

- We presented a unified framework for the three criteria from Chapter 2. These criteria include ranking, accuracy at the top and hypothesis testing.
- We showed that several known methods (*TopPush*, *Grill*, τ-*FPL*) fall into our framework and derived some completely new methods (*Pat&Mat*, *Pat&Mat-NP*).
- We performed a theoretical analysis of the methods. We showed that known methods suffer from certain disadvantages. While *TopPush* and τ-*FPL* are sensitive to outliers, *Grill* is non-convex. We proved the global convergence of the stochastic gradient descent for *Pat&Mat* and *Pat&Mat-NP*.
- We performed a numerical comparison and we showed a good performance of our method *Pat&Mat-NP*.

7.2 Dual

In this paper, we analyzed and extended the general framework for binary classification on top samples from [30] to nonlinear problems. Achieved results can be summarized as follows:

- We derived the dual formulations for *TopPush*, *TopPushK* and *Pat&Mat*.
- We proposed a new method for solving the dual problems. We performed its complexity analysis. For selected surrogate functions we also derived the exact formulas needed in the method.
- We performed a numerical analysis of the proposed method. We showed its good convergence as well as improved performance of nonlinear kernels over the linear one.

Based on the numerical analysis from Section 6.2, we recommend using *TopPush* or *TopPushK* for problems where the resulting recall should be small. Otherwise, we recommend using *Pat&Mat* with an appropriately selected τ parameter.

7.3 Neural Networks

We proposed *DeepTopPush* as an efficient method for solving the constrained non-decomposable problem of accuracy at the top, which focuses on the performance only above a threshold. We implicitly removed some optimization variables, created an unconstrained end-to-end network

and used the stochastic gradient descent to train it. We modified the minibatch so that the sampled threshold (computed on a minibatch) is a good estimate of the true threshold (computed on all samples). We showed both theoretically and numerically that this procedure reduces the bias of the sampled gradient. The time increase over the standard method with no threshold is small. We demonstrated the usefulness of *DeepTopPush* both on visual recognition datasets, a ranking problem and on a real-world application of malware detection.

Apendices

A

Appendix for Chapter 3

A.1 Convexity

Proposition 3.1

Consider vector of scores $\boldsymbol{s}$ with elements defined as $s_i = \boldsymbol{w}^\top \boldsymbol{x}_i$ for all $i \in \mathcal{I}$ and Notation 2.2. Recall the decision thresholds from Section 2.2 and 2.3

$$t_0(\boldsymbol{w}) = s^-_{[1]}, \qquad t_1(\boldsymbol{w}) = \max\left\{ t \;\middle|\; \frac{1}{n}\sum_{i\in\mathcal{I}} \mathbb{1}_{[s_i \geq t]} \geq \tau \right\},$$

$$t_2(\boldsymbol{w}) = \frac{1}{K}\sum_{i=1}^{K} s_{[i]}, \qquad t_3(\boldsymbol{w}) \quad \text{solves} \quad \frac{1}{n}\sum_{i\in\mathcal{I}} l(\vartheta(s_i - t)) = \tau,$$

Then thresholds t_0, t_2 and t_3 are convex functions of weights $\boldsymbol{w}$, while the threshold t_1 is non-convex.

Proof of Proposition 3.1 on page 24:

From Notation 2.2, threshold t_0 is just a maximum from vector $\boldsymbol{s}^-$ of scores of all negative samples. Since maximum is a convex function, threshold t is a convex function of weights $\boldsymbol{w}$. Moreover, it is easy to show that the quantile t_1 is not convex. Due to [28], the mean of the K highest values of a vector is a convex function. Therefore, threshold t_2 is a convex function of weights $\boldsymbol{w}$. It remains to analyze threshold t_3. Let us define function g as follows

$$g(\boldsymbol{w}, t) := \frac{1}{n}\sum_{i\in\mathcal{I}} l(\boldsymbol{w}^\top \boldsymbol{x}_i - t) - \tau.$$

where we set $\vartheta = 1$ for simplicity. Then t_3 is defined via an implicit equation $g(\boldsymbol{w}, t) = 0$. Since l is convex, we immediately obtain that g is jointly convex in both variables.

To show the convexity, consider $\boldsymbol{w}$, $\tilde{\boldsymbol{w}} \in \mathbb{R}^d$ and the corresponding thresholds $t = t_3(\boldsymbol{w})$, $\tilde{t} = t_3(\tilde{\boldsymbol{w}})$. Then for any $\lambda \in [0, 1]$, we have

$$g(\lambda\boldsymbol{w} + (1-\lambda)\tilde{\boldsymbol{w}},\ \lambda t + (1-\lambda)\tilde{t}) \leq \lambda g(\boldsymbol{w}, t) + (1-\lambda) g(\tilde{\boldsymbol{w}}, \tilde{t}) = 0. \tag{A.1}$$

The inequality follows from the convexity of g and the equality from $g(\boldsymbol{w}, t) = g(\tilde{\boldsymbol{w}}, \tilde{t}) = 0$, which holds due to the definition of t_3. From the definition of t_3, we also have

$$g(\lambda\boldsymbol{w} + (1-\lambda)\tilde{\boldsymbol{w}},\ t_3(\lambda\boldsymbol{w} + (1-\lambda)\tilde{\boldsymbol{w}})) = 0. \tag{A.2}$$

Since g is non-increasing in the second variable, from (A.1) and (A.2) we deduce

$$t_3(\lambda \boldsymbol{w} + (1-\lambda)\tilde{\boldsymbol{w}}) \leq \lambda t + (1-\lambda)\tilde{t} = \lambda t_3(\boldsymbol{w}) + (1-\lambda) t_3(\tilde{\boldsymbol{w}}),$$

which implies that function $\boldsymbol{w} \mapsto t_3(\boldsymbol{w})$ is convex. ■

Theorem 3.2

If the threshold $t = t(\boldsymbol{w})$ is a convex function of weights $\boldsymbol{w}$, then function

$$L(\boldsymbol{w}) = \overline{\mathrm{fn}}(\boldsymbol{s}, t)$$

is convex.

Proof of Theorem 3.2 on page 24:
Due to the definition (2.2), the objective function L equals to

$$L(\boldsymbol{w}) = \overline{\mathrm{fn}}(\boldsymbol{s}, t(\boldsymbol{w})) = \sum_{i \in \mathcal{I}_+} l\Big(t(\boldsymbol{w}) - \boldsymbol{w}^\top \boldsymbol{x}_i\Big).$$

Here we write $t(\boldsymbol{w})$ to stress the dependence of t on $\boldsymbol{w}$. Since $\boldsymbol{w} \mapsto t(\boldsymbol{w})$ is a convex function, we also have that $\boldsymbol{w} \mapsto t(\boldsymbol{w}) - \boldsymbol{w}^\top \boldsymbol{x}$ is a convex function. From its definition, the surrogate function l is convex and non-decreasing. Since the composition of a convex function with a non-decreasing convex function is a convex function, this finishes the proof. ■

A.2 Differentiability

Theorem 3.3

Consider thresholds from Proposition 3.1. Threshold t_0, t_1 and t_2 are non-differentiable functions of weights $\boldsymbol{w}$. Moreover, if the surrogate function l is differentiable, threshold t_3 is a differentiable function of weights $\boldsymbol{w}$, and its derivative equals

$$\nabla t_3(\boldsymbol{w}) = \frac{\sum_{i \in \mathcal{I}} l'(\vartheta(\boldsymbol{w}^\top \boldsymbol{x}_i - t_3(\boldsymbol{w})))\boldsymbol{x}_i}{\sum_{j \in \mathcal{I}} l'\Big(\vartheta(\boldsymbol{w}^\top \boldsymbol{x}_j - t_3(\boldsymbol{w}))\Big)}.$$

Proof of Theorem 3.3 on page 24:
The non-differentiability of t_0, t_1 and t_2 happens whenever the threshold value is achieved at two different scores. The result for t_3 follows directly from the implicit function theorem. ■

A.3 Stability

Example 3.4: Degenerate Behaviour

Consider n negative samples uniformly distributed in $[-1,0]\times[-1,1]$, n positive samples uniformly distributed in $[0,1]\times[-1,1]$ and one negative sample at $(2,0)$. An illustration of such settings is provided in Figure 3.1 (**left**). If n is large enough, the point at $(2,0)$, is an outlier and the problem is (almost) perfectly separable using the separating hyperplane with normal vector $\boldsymbol{w}_1 = (1,0)$.

Additionally to the assumptions from Example 3.4, we consider the hinge loss function and no regularization for all formulations from Table 2.1. We also assume that n is large, and the outlier may be ignored for the computation of thresholds that require a large number of points. Since the computation is simple for other formulations, we show it only for *Pat&Mat*.

- For $\boldsymbol{w}_0 = (0,0)$, we have

$$\tau = \frac{1}{n}\sum_{i\in\mathcal{I}} l\Big(\vartheta(\boldsymbol{w}_0^\top \boldsymbol{x}_i - t)\Big) = l(0-\vartheta t) = 1-\vartheta t,$$

  which implies that threshold t equals

$$t = \frac{1-\tau}{\vartheta}. \tag{A.3}$$

  Consequently, the value of the objective function is

$$L(\boldsymbol{w}_0) = \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} l(t-\boldsymbol{w}_0^\top \boldsymbol{x}_i) = l(t-0) = 1+t, \tag{A.4}$$

  where the last equality follows from the definition of the hinge loss function and the fact that $t \geq 0$. This finishes the computation for $\boldsymbol{w}_0$.

- For $\boldsymbol{w}_1 = (1,0)$, the computation is similar. Since all samples are uniformly distributed in $[-1,1]\times[-1,1]$, scores $\boldsymbol{w}_1^\top \boldsymbol{x}_i$ for $i \in \mathcal{I}$ are uniformly distributed in $[-1,1]$. Then, if the scaling parameter ϑ satisfies $\vartheta \leq \tau$, we have

$$\begin{aligned}\tau &= \frac{1}{n}\sum_{i\in\mathcal{I}} l\Big(\vartheta(\boldsymbol{w}_1^\top \boldsymbol{x}_i - t)\Big) \approx \frac{1}{2}\int_{-1}^{1} l(\vartheta(s-t))\,\mathrm{d}s = \frac{1}{2}\int_{-1}^{1} \max\{0, 1+\vartheta(s-t)\}\,\mathrm{d}s \\ &= \frac{1}{2}\int_{-1}^{1} (1+\vartheta(s-t))\,\mathrm{d}s = 1-\vartheta t + \frac{\vartheta}{2}\int_{-1}^{1} s\,\mathrm{d}s = 1-\vartheta t.\end{aligned}$$

  which again implies that threshold t equals

$$t = \frac{1-\tau}{\vartheta}.$$

  Note that we could ignore the max operator in the relation above, since

$$1+\vartheta(s-t) \geq 1+\vartheta(-1-t) = 1+\vartheta\left(-1-\frac{1-\tau}{\vartheta}\right) = \tau-\vartheta \geq 0.$$

  The last inequality follows from the assumption $\vartheta \leq \tau$. Finally, since positive samples are uniformly distributed in $[0,1]\times[-1,1]$, corresponding scores $\boldsymbol{w}_1^\top \boldsymbol{x}_i$ for $i \in \mathcal{I}_+$ are uniformly distributed in $[0,1]$. Therefore, for the objective function, we have

$$L(\boldsymbol{w}_1) = \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} l(t-\boldsymbol{w}_1^\top \boldsymbol{x}_i) \approx \int_0^1 l(t-s)\,\mathrm{d}s = \int_0^1 (1+t-s)\,\mathrm{d}s = 0.5+t.$$

Results for *Pat&Mat-NP* can be obtained in a similar way.

Theorem 3.5

Consider any of these formulations: *TopPush*, *TopPushK*, *TopMeanK* or τ-*FPL*. Fix any $\boldsymbol{w}$ and denote the corresponding objective function $L(\boldsymbol{w})$ and threshold $t(\boldsymbol{w})$. If we have

$$t(\boldsymbol{w}) \geq \frac{1}{n_+} \sum_{i \in \mathcal{I}_+} \boldsymbol{w}^\top \boldsymbol{x}_i, \tag{3.1}$$

then $L(\mathbf{0}) \leq L(\boldsymbol{w})$. Specifically, using Notation 2.2 we get the following implications

$$\begin{aligned}
s^-_{[1]} \geq \frac{1}{n_+} \sum_{i=1}^{n_+} s^+_i \quad &\Longrightarrow \quad L(\mathbf{0}) \leq L(\boldsymbol{w}) \text{ for } \textit{TopPush}, \\
\frac{1}{K} \sum_{i=1}^{K} s^-_{[i]} \geq \frac{1}{n_+} \sum_{i=1}^{n_+} s^+_i \quad &\Longrightarrow \quad L(\mathbf{0}) \leq L(\boldsymbol{w}) \text{ for } \textit{TopPushK}, \\
\frac{1}{K} \sum_{i=1}^{K} s_{[i]} \geq \frac{1}{n_+} \sum_{i=1}^{n_+} s^+_i \quad &\Longrightarrow \quad L(\mathbf{0}) \leq L(\boldsymbol{w}) \text{ for } \textit{TopMeanK}, \\
\frac{1}{n_\tau} \sum_{i=1}^{n_\tau} s^-_{[i]} \geq \frac{1}{n_+} \sum_{i=1}^{n_+} s^+_i \quad &\Longrightarrow \quad L(\mathbf{0}) \leq L(\boldsymbol{w}) \text{ for } \tau\text{-}\textit{FPL}.
\end{aligned}$$

Proof of Theorem 3.5 on page 26:

All mentioned formulations use a surrogate approximation of the false-negative rate as the objective function L. The objective function has the following form

$$L(\boldsymbol{w}) = \frac{1}{n_+} \sum_{i \in \mathcal{I}_+} l(t - \boldsymbol{w}^\top \boldsymbol{x}_i)$$

Due to $l(0) = 1$ and the convexity of l, we have $l(s) \geq 1 + cs$, where c equals to the derivative of l at 0. Then we have

$$L(\boldsymbol{w}) \geq \frac{1}{n_+} \sum_{i \in \mathcal{I}_+} (1 + c(t - \boldsymbol{w}^\top \boldsymbol{x}_i)) = 1 + c \left(t - \frac{1}{n_+} \sum_{i \in \mathcal{I}_+} \boldsymbol{w}^\top \boldsymbol{x} \right) \geq 1,$$

where the last inequality follows from assumption (3.1). Now we realize that for any formulation from the statement, the corresponding threshold for $\boldsymbol{w} = 0$ equals to $t = 0$, and thus $L(\mathbf{0}) = 1$. But it implies that $L(\mathbf{0}) \leq L(\boldsymbol{w})$. The second part of the result follows from the form of thresholds $t(\boldsymbol{w})$. ■

Theorem 3.8

Consider the *Pat&Mat* or *Pat&Mat-NP* formulation with the hinge loss as a surrogate and no regularization. Assume that for some $\boldsymbol{w}$ we have

$$\frac{1}{n_+} \sum_{i \in \mathcal{I}_+} \boldsymbol{w}^\top \boldsymbol{x}_i > \frac{1}{n_-} \sum_{j \in \mathcal{I}_-} \boldsymbol{w}^\top \boldsymbol{x}_j. \tag{3.2}$$

Then there exists a scaling parameter ϑ_0 for the surrogate top τ-quantile (2.12) or (2.19) such that $L(\boldsymbol{w}) < L(\mathbf{0})$ for all $\vartheta \in (0, \vartheta_0)$.

Proof of Theorem 3.8 on page 27:
Recall that we use linear model and Notation 2.2 and let us define the following auxiliary variables

$$s_{\min} = \min_{i \in I} s_i, \qquad s_{\max} = \max_{i \in I} s_i, \qquad \bar{s} = \frac{1}{n} \sum_{i \in \mathcal{I}} s_i.$$

Using the definition of $\bar{s}$ we get the following relation

$$\bar{s} = \frac{1}{n} \sum_{i \in \mathcal{I}_+} s_i + \frac{1}{n} \sum_{i \in \mathcal{I}_-} s_i < \frac{1}{n} \sum_{i \in \mathcal{I}_+} s_i + \frac{n_-}{n n_+} \sum_{i \in \mathcal{I}_+} s_i = \frac{1}{n_+} \sum_{i \in \mathcal{I}_+} s_i, \tag{A.5}$$

where the inequality follows from (3.2), and the last equality follows from

$$\frac{1}{n} + \frac{n_-}{n n_+} = \frac{1}{n}\left(1 + \frac{n_-}{n_+}\right) = \frac{1}{n} \frac{n_+ + n_-}{n_+} = \frac{1}{n} \frac{n}{n_+} = \frac{1}{n_+}.$$

Since the average of all elements of the vector is smaller or equal to its maximum B, we get the following relation

$$\bar{s} < \frac{1}{n_+} \sum_{i \in \mathcal{I}_+} s_i \le \max_{i \in \mathcal{I}_+} s_i \le \max_{i \in \mathcal{I}} s_i = s_{\max}$$

where the first inequality follows from (A.5). The lower bound for $\bar{s}$ can be computed in a similar way.

Combining all results above, we have $s_{\min} < \bar{s} < s_{\max}$. Then we can define

$$\vartheta_0 = \min\left\{ \frac{\tau}{\bar{s} - s_{\min}}, \frac{1 - \tau}{s_{\max} - \bar{s}}, \tau \right\}.$$

Note that $\vartheta_0 > 0$. Now we fix any $\vartheta \in (0, \vartheta_0)$ and define

$$t = \frac{1 - \tau}{\vartheta} + \bar{s}.$$

Then for any $i \in \mathcal{I}$, we obtain

$$1 + \vartheta(s_i - t) \ge 1 + \vartheta(s_{\min} - t) = 1 + \vartheta\left(s_{\min} - \frac{1 - \tau}{\vartheta} - \bar{s}\right) = \tau - \vartheta(\bar{s} - s_{\min}),$$

where the first equality follows from the definition of t. From the definition ϑ_0 we deduce

$$0 < \vartheta \le \vartheta_0 \le \frac{\tau}{\bar{s} - s_{\min}}.$$

Since $\bar{s} - s_{\min} > 0$, we get the following inequality

$$1 + \vartheta(s_i - t) \ge \tau - \vartheta(\bar{s} - s_{\min}) \ge \tau - \frac{\tau}{\bar{s} - s_{\min}}(\bar{s} - s_{\min}) = 0 \tag{A.6}$$

Combining the definition of the hinge loss function from Notation 2.1 and the inequality above, we have

$$l(\vartheta(s_i - t)) = \max\{0, 1 + \vartheta(s_i - t), 0\} = 1 + \vartheta(s_i - t).$$

Finally, replacing the hinge loss in the left-hand side of (2.12) leads to

$$\frac{1}{n} \sum_{i \in \mathcal{I}} l(\vartheta(s_i - t)) = \frac{1}{n} \sum_{i \in \mathcal{I}} (1 + \vartheta(s_i - t)) = 1 - \vartheta t + \frac{\vartheta}{n} \sum_{i \in \mathcal{I}} s_i = 1 - \vartheta\left(\frac{1 - \tau}{\vartheta} + \bar{s}\right) + \vartheta \bar{s} = \tau,$$

where the third equality employs the definition of $\bar{s}$ and t. But this means that t is the threshold corresponding to $\boldsymbol{w}$, i.e. it solves (2.12). In the same way, as we derived (A.6), we get

$$1 + t - s_i \geq 1 + t - s_{\max} = 1 + \frac{1-\tau}{\vartheta} + \bar{s} - s_{\max} \geq \frac{1-\tau}{\vartheta} + \bar{s} - s_{\max} \geq 0, \tag{A.7}$$

where the last inequality follows from the definition of ϑ_0. Then for the objective, we have

$$L(\boldsymbol{w}) = \frac{1}{n_+} \sum_{i \in \mathcal{I}_+} l(t - s_i) = \frac{1}{n_+} \sum_{i \in \mathcal{I}_+} (1 + t - s_i) = 1 + t - \frac{1}{n_+} \sum_{i \in \mathcal{I}_+} s_i < 1 + \left(\frac{1-\tau}{\vartheta} + \bar{s} \right) - \bar{s} = 1 + \frac{1-\tau}{\vartheta},$$

where the second equality follows from (A.7), the only inequality from (A.5). Using (A.3) and (A.4), we finally get

$$L(\boldsymbol{w}) < 1 + \frac{1-\tau}{\vartheta} = L(\boldsymbol{0}).$$

Thus, we finished the proof for *Pat&Mat*. The proof for *Pat&Mat-NP* can be performed identically. ■

A.4 Threshold Comparison

A lower threshold t means a lower objective function value when the objective contains only false-negative samples. Therefore, a lower threshold is preferred. The two following lemmas compare thresholds defined in Chapter 2.

Lemma A.7: Thresholds relation [29]

Consider thresholds from Proposition 3.1. Then the following inequalities hold

$$t_1(\boldsymbol{s}) \leq t_2(\boldsymbol{s}) \leq t_3(\boldsymbol{s}).$$

Lemma A.8

Consider the *Grill*, *Grill-NP*, *TopMeanK* and τ-*FPL* formulations and Notation 2.2. Then we have the following statements:

$$s^+_{[n_+\tau]} > s^-_{[n_-\tau]} \implies \textit{Grill} \text{ has larger threshold than } \textit{Grill-NP},$$

$$\frac{1}{n_+\tau} \sum_{i=1}^{n_+\tau} s^+_{[i]} > \frac{1}{n_-\tau} \sum_{i=1}^{n_-\tau} s^-_{[i]} \implies \textit{TopMeanK} \text{ has larger threshold than } \tau\textit{-FPL}.$$

Proof:
Since $\boldsymbol{s}^+$ and $\boldsymbol{s}^-$ are computed on disjunctive indices, we have

$$s_{[n\tau]} \geq \min\{s^+_{[n_+\tau]}, s^-_{[n_-\tau]}\}.$$

Since $s_{[n\tau]}$ is the threshold for *Grill* and $s^-_{[n_-\tau]}$ is the threshold for *Grill-NP*, the first statement follows. The second part can be shown in a similar way. ■

Since the goal of the presented formulations is to push s^+ above s^-, we may expect that the conditions in Lemma A.8 are satisfied.

A.5 Efficient Computing of the Threshold for *Pat&Mat*

In this section, we show how to efficiently compute the threshold (2.12) for *Pat&Mat* with the linear model and the hinge loss as a surrogate. Consider function

$$h(t) = \sum_{i \in \mathcal{I}} l(\vartheta(s_i - t)) - n\tau. \tag{A.8}$$

Then solving (3.7) is equivalent to looking for $\hat{t}$ such that $h(\hat{t}) = 0$. Function h is continuous and strictly decreasing (until it hits the global minimum) with $h(t) \to \infty$ as $t \to -\infty$ and $h(t) \to -n\tau$ as $t \to \infty$. Thus, there is a unique solution to the equation $h(t) = 0$. For sorted data, the following lemma advises how to solve this equation.

Lemma A.9

Consider vector of scores $\boldsymbol{s}$ and its sorted version $\boldsymbol{s}_{[\cdot]}$ with decreasing elements as defined in Notation 2.2. Define $\gamma = 1/\vartheta$. Then for all $i = 2, 3, \ldots, n$ we have

$$h(s_{[j]} + \gamma) = h(s_{[j-1]} + \gamma) + (j-1)\vartheta(s_{[j-1]} - s_{[j]}), \tag{A.9}$$

with the initial condition $h(s_{[1]} + \gamma) = -n\tau$.

Proof:
Observe that

$$\begin{aligned} h(s_{[j]} + \gamma) &= \sum_{i \in \mathcal{I}} l\Big(\vartheta(s_i - (s_{[j]} + \gamma))\Big) - n\tau = \sum_{i \in \mathcal{I}} \max\Big\{0,\ 1 + \vartheta\Big(s_i - s_{[j]} - \frac{1}{\vartheta}\Big)\Big\} - n\tau \\ &= \sum_{i \in \mathcal{I}} \max\Big\{0,\ \vartheta\Big(s_i - s_{[j]}\Big)\Big\} - n\tau = \sum_{i=1}^{j-1} \vartheta(s_{[i]} - s_{[j]}) - n\tau, \end{aligned}$$

where the last equality holds since $\vartheta > 0$ and $s_{[i]} - s_{[j]} \le 0$ for all $i \ge j$. From here, we obtain $h(s_{[1]} + \gamma) = -n\tau$. Moreover, we have

$$\begin{aligned} h(s_{[j]} + \gamma) &= \sum_{i=1}^{j-1} \vartheta(s_{[i]} - s_{[j]}) - n\tau \\ &= \sum_{i=1}^{j-2} \vartheta(s_{[i]} - s_{[j]}) + \vartheta(s_{[j-1]} - s_{[j]}) - n\tau \\ &= \sum_{i=1}^{j-2} \vartheta(s_{[i]} - s_{[j]} + s_{[j-1]} - s_{[j-1]}) + \vartheta(s_{[j-1]} - s_{[j]}) - n\tau \\ &= \sum_{i=1}^{j-2} \vartheta(s_{[i]} - s_{[j-1]}) + \sum_{i=1}^{j-2} \vartheta(s_{[j-1]} - s_{[j]}) + \vartheta(s_{[j-1]} - s_{[j]}) - n\tau \\ &= h(s_{[j-1]} + \gamma) + (j-1)\vartheta(s_{[j-1]} - s_{[j]}), \end{aligned}$$

which finishes the proof. ∎

Thus, to solve $h(t) = 0$, we start with $t_1 = s_{[1]} + \gamma$ and $h(t_1) = -n\tau$. Then we start decreasing t according to (A.9) until we find some $t_i = s_{[i]} + \gamma$ such that $h(t_i) > 0$. The desired threshold $\hat{t}$ then lies between t_i and t_{i-1}. Since h is a piecewise linear function with

$$h(t) = h(t_{i-1}) + \frac{t - t_{i-1}}{t_i - t_{i-1}}(h(t_i) - h(t_{i-1}))$$

for all $t \in [t_{i-1}, t_i]$, the precise value of $\hat{t}$ can be computed by a simple interpolation

$$\hat{t} = t_{i-1} - h(t_{i-1})\frac{t_i - t_{i-1}}{h(t_i) - h(t_{i-1})} = t_{i-1} - h(t_{i-1})\frac{t_i - t_{i-1}}{-(i-1)\vartheta(t_i - t_{i-1})} = t_{i-1} + \frac{h(t_{i-1})}{\vartheta(i-1)}.$$

A.6 Stochastic Gradient Descent

The proof of convergence of stochastic gradient descent for *Pat&Mat* and *Pat&Mat-NP* is divided into three parts. In Section A.6.1, we prove a general statement for convergence of stochastic gradient descent with a convex objective function. In Section A.6.2 we apply it to Theorem 3.9. Finally, in Section A.6.3, we provide auxiliary results.

A.6.1 General Results

Consider a differentiable objective function L and the optimization method

$$\boldsymbol{w}^{k+1} = \boldsymbol{w}^k - \alpha^k g(\boldsymbol{w}^k), \tag{A.10}$$

where $\alpha^k > 0$ is a stepsize and $g(\boldsymbol{w}^k)$ is an approximation of the gradient $\nabla L(\boldsymbol{w}^k)$. Assume the following:

(**A1**) L is differentiable, convex, and attains a global minimum;

(**A2**) $\left\|g(\boldsymbol{w}^k)\right\| \leq B$ for all k;

(**A3**) the stepsize is non-increasing and satisfies $\sum_{k=0}^{\infty} \alpha^k = \infty$;

(**A4**) the stepsize satisfies $\sum_{k=0}^{\infty} (\alpha^k)^2 < \infty$;

(**A5**) the stepsize satisfies $\sum_{k=0}^{\infty} \left\|\alpha^{k+1} - \alpha^k\right\| < \infty$.

Assumptions (**A3**)-(**A5**) are satisfied for example for

$$\alpha^k = \frac{\alpha^0}{k+1}.$$

Theorem A.10

Assume that the assumptions (**A1**)-(**A4**) are satisfied. If there exists some C such that for the global minimizer $\boldsymbol{w}^*$ of L we have

$$\sum_{k=0}^{\infty} \alpha^k \left\langle g(\boldsymbol{w}^k) - \nabla L(\boldsymbol{w}^k),\, \boldsymbol{w}^* - \boldsymbol{w}^k \right\rangle \leq C, \tag{A.11}$$

then the sequence $\{\boldsymbol{w}^k\}$ generated by (A.10) is bounded and $L(\boldsymbol{w}^k) \to L(\boldsymbol{w}^*)$. Thus, all its convergent subsequences converge to some global minimum of L.

Proof:
Note first that the convexity of L from (**A1**) implies

$$\left\langle \nabla L(\boldsymbol{w}^k),\, \boldsymbol{w}^* - \boldsymbol{w}^k \right\rangle \leq L(\boldsymbol{w}^*) - L(\boldsymbol{w}^k). \tag{A.12}$$

Then we have

$$\begin{aligned}
\left\|w^{k+1}-w^*\right\|^2 &= \left\|w^k-\alpha^k g(w^k)-w^*\right\|^2 \\
&= \left\|w^k-w^*\right\|^2 + 2\alpha^k\left\langle g(w^k),\, w^*-w^k\right\rangle + (\alpha^k)^2\left\|g(w^k)\right\|^2 \\
&\le \left\|w^k-w^*\right\|^2 + 2\alpha^k\left\langle g(w^k),\, w^*-w^k\right\rangle + (\alpha^k)^2 B^2 \\
&= \left\|w^k-w^*\right\|^2 + 2\alpha^k\left\langle g(w^k)+\nabla L(w^k)-\nabla L(w^k),\, w^*-w^k\right\rangle + (\alpha^k)^2 B^2 \\
&\le \left\|w^k-w^*\right\|^2 + 2\alpha^k\left\langle g(w^k)-\nabla L(w^k),\, w^*-w^k\right\rangle + 2\alpha^k\Big(L(w^*)-L(w^k)\Big) + (\alpha^k)^2 B^2,
\end{aligned}$$

where the first inequality follows from assumption (A2) and the second one from the properties of inner product and (A.12). Summing this expression for all k and using (A.11) leads to

$$\limsup_{k\to\infty}\left\|w^k-w^*\right\|^2 \le \left\|w^0-w^*\right\|^2 + 2C + 2\sum_{k=0}^{\infty}\alpha^k(L(w^*)-L(w^k)) + \sum_{k=0}^{\infty}(\alpha^k)^2B^2.$$

Using assumption (A4) results in the existence of some $\hat{C}$ such that

$$\limsup_{k\to\infty}\left\|w^k-w^*\right\|^2 + 2\sum_{k=0}^{\infty}\alpha^k\Big(L(w^k)-L(w^*)\Big) \le 2\hat{C}. \tag{A.13}$$

Since $\alpha^k > 0$ and $L(w^k) \ge L(w^*)$ as w^* is a global minimizer of L, we infer that sequence $\{w^k\}$ is bounded and (A.13) implies

$$\sum_{k=0}^{\infty}\alpha^k\Big(L(w^k)-L(w^*)\Big) \le \hat{C}.$$

Since $L(w^k)-L(w^*) \ge 0$, due to assumption (A3) we obtain

$$\lim_{k\to\infty} L(w^k) = L(w^*),$$

which implies the theorem statement. ■

A.6.2 Proof of Theorem 3.9

For the proof of Theorem 3.9, we consider a general surrogate function l that satisfies:

- (**S1**) $l(s) \ge 0$ for all $s \in \mathbb{R}$, $l(0) = 1$ and $l(s) \to 0$ as $s \to -\infty$;
- (**S2**) l is convex and strictly increasing function on (s_0, ∞), where $s_0 := \sup\{s \mid l(s) = 0\}$;
- (**S3**) $\frac{l'}{l}$ is a decreasing function on (s_0, ∞);
- (**S4**) l' is a bounded function;
- (**S5**) l' is a Lipschitz continuous function with Lipschitz constant D.

All these requirements are satisfied for the logistic loss or the Huber loss function. Huber loss is the hinge surrogate smoothened on an ε-neighborhood of zero.

Theorem 3.9

Consider the *Pat&Mat* or *Pat&Mat-NP* formulation, stepsizes $\alpha^k = {}^{\alpha^0}/_{k+1}$ and piecewise disjoint minibatches $\mathcal{I}^1_{\text{mb}},\ldots,\mathcal{I}^m_{\text{mb}}$ which cycle periodically $\mathcal{I}^{k+m}_{\text{mb}} = \mathcal{I}^k_{\text{mb}}$. If l is the smoothened

(Huberized) hinge function, then Algorithm 2 converges to the global minimum of (2.13).

Proof of Theorem 3.9 on page 29:
We intend to apply Theorem A.10 and thus, we need to verify its assumptions. Assumption (**A1**) is satisfied as L is convex due to Theorem 3.2. Assumption (**A2**) follows directly from Lemma A.13, and assumptions (**A3**)-(**A5**) are imposed directly in the statement of this theorem. It remains to verify (A.11).

For simplicity, we will do so only for $\vartheta = 1$ and for 2 minibatches of the same size. However, the proof would be identical for different ϑ and more minibatches. From the assumptions, we have two minibatches $\mathcal{I}^k_{\text{mb}}$ and $\mathcal{I}^{k+1}_{\text{mb}}$, which are pairwise disjoint and cover all samples. Moreover, for all k, we have $\mathcal{I}^k_{\text{mb}} = \mathcal{I}^{k+2}_{\text{mb}}$. Furthermore, the assumptions imply that the number of positive samples in each minibatch is equal to $n^k_{\text{mb},+} = \frac{1}{2}n_+$, where n_+ is the total number of positive samples.

First we estimate the difference between s^k_i defined in (3.6) and $\boldsymbol{x}_i^\top \boldsymbol{w}^k$. For any $i \in \mathcal{I}^k_{\text{mb}}$ we have $s^k_i = \boldsymbol{x}_i^\top \boldsymbol{w}^k$. Since we have two disjoint minibatches, due to the construction (3.6) we get

$$\begin{aligned} s^{k-1}_i = s^{k-2}_i = \boldsymbol{x}_i^\top \boldsymbol{w}^{k-2} &= \boldsymbol{x}_i^\top \left(\boldsymbol{w}^k + \alpha^{k-2} g(\boldsymbol{w}^{k-2}) + \alpha^{k-1} g(\boldsymbol{w}^{k-1})\right) \\ &= \boldsymbol{x}_i^\top \boldsymbol{w}^k + \alpha^{k-2} \boldsymbol{x}_i^\top g(\boldsymbol{w}^{k-2}) + \alpha^{k-1} \boldsymbol{x}_i^\top g(\boldsymbol{w}^{k-1}). \end{aligned} \tag{A.14}$$

Similarly, due to the construction of s^k_i from (3.6), we have for $i \notin \mathcal{I}^k_{\text{mb}}$

$$s^k_i = s^{k-1}_i = \boldsymbol{x}_i^\top \boldsymbol{w}^{k-1} = \boldsymbol{x}_i^\top (\boldsymbol{w}^k + \alpha^{k-1} g(\boldsymbol{w}^{k-1})) = \boldsymbol{x}_i^\top \boldsymbol{w}^k + \alpha^{k-1} \boldsymbol{x}_i^\top g(\boldsymbol{w}^{k-1}). \tag{A.15}$$

Recall that we already verified (**A1**)-(**A5**). Combining (**A2**) with (A.14) and (A.15) yields the existence of some C_2 such that for all $i \in \mathcal{I}$ we have

$$\left\| s^k_i - \boldsymbol{x}_i^\top \boldsymbol{w}^k \right\| \leq C_2 \alpha^{k-1}, \qquad \left\| s^{k-1}_i - \boldsymbol{x}_i^\top \boldsymbol{w}^k \right\| \leq C_2 \left(\alpha^{k-1} + \alpha^{k-2}\right). \tag{A.16}$$

This also immediately implies

$$\left\| t^k - t(\boldsymbol{w}^k) \right\| \leq C_2 \alpha^{k-1}, \qquad \left\| t^{k-1} - t(\boldsymbol{w}^k) \right\| \leq C_2 \left(\alpha^{k-1} + \alpha^{k-2}\right). \tag{A.17}$$

Moreover, we know that l' is Lipschitz continuous with Lipschitz constant D according to (**S5**). Then due to (A.16) and (A.17) we get

$$\left\| l'(t^k - s^k_i) - l'(t(\boldsymbol{w}^k) - \boldsymbol{x}_i^\top \boldsymbol{w}^k) \right\| \leq D \left\| t^k - s^k_i - t(\boldsymbol{w}^k) + \boldsymbol{x}_i^\top \boldsymbol{w}^k \right\| \leq 2 C_2 D \alpha^{k-1}. \tag{A.18}$$

In an identical way, we can derive the following relations

$$\begin{aligned} \left\| l'(t^{k-1} - s^{k-1}_i) - l'(t(\boldsymbol{w}^k) - \boldsymbol{x}_i^\top \boldsymbol{w}^k) \right\| &\leq 2 C_2 D \left(\alpha^{k-1} + \alpha^{k-2}\right), \\ \left\| l'(s^k_i - t^k) - l'(\boldsymbol{x}_i^\top \boldsymbol{w}^k - t(\boldsymbol{w}^k)) \right\| &\leq 2 C_2 D \alpha^{k-1}, \\ \left\| l'(s^{k-1}_i - t^{k-1}) - l'(\boldsymbol{x}_i^\top \boldsymbol{w}^k - t(\boldsymbol{w}^k)) \right\| &\leq 2 C_2 D \left(\alpha^{k-1} + \alpha^{k-2}\right). \end{aligned} \tag{A.19}$$

Now we need to estimate the distance between $\nabla t(\boldsymbol{w}^k)$ and ∇t^k. By plugging (3.8) into (3.10), we get

$$\nabla t^k = \frac{\sum_{i \in \mathcal{I}^k_{\text{mb}}} l'(s^k_i - t^k) \boldsymbol{x}_i + \sum_{i \in \mathcal{I}^{k-1}_{\text{mb}}} l'(s^{k-1}_i - t^{k-1}) \boldsymbol{x}_i}{\sum_{i \in \mathcal{I}} l'(s^k_i - t^k)}.$$

Moreover, using Theorem 3.3 and the fact that we have only two minibatches and therefore for any k we have $\mathcal{I} = \mathcal{I}^k_{\text{mb}} \cup \mathcal{I}^{k-1}_{\text{mb}}$, we get

$$\nabla t(\boldsymbol{w}^k) = \frac{\sum_{i\in\mathcal{I}^k_{\text{mb}}} l'(\boldsymbol{x}_i^\top \boldsymbol{w}^k - t(\boldsymbol{w}^k))\boldsymbol{x}_i + \sum_{i\in\mathcal{I}^{k-1}_{\text{mb}}} l'(\boldsymbol{x}_i^\top \boldsymbol{w}^k - t(\boldsymbol{w}^k))\boldsymbol{x}_i}{\sum_{i\in\mathcal{I}} l'(\boldsymbol{x}_i^\top \boldsymbol{w}^k - t(\boldsymbol{w}^k))}.$$

From Lemma A.12 we deduce that the denominators in the relations above are bounded away from zero uniformly in k. Assumption (**A4**) implies $\alpha^k \to 0$. This allows us to use Lemma A.14 which together with (A.19) implies that there is some C_3 such that for all sufficiently large k we have

$$\left\|\nabla t^k - \nabla t(\boldsymbol{w}^k)\right\| \le C_3\left(\alpha^{k-1} + \alpha^{k-2}\right). \tag{A.20}$$

Using the assumptions above, we can simplify the terms for $g(\boldsymbol{w}^k)$ and $\nabla L(\boldsymbol{w}^k)$ to

$$\begin{aligned}
g(\boldsymbol{w}^k) &= \frac{2}{n_+}\sum_{i\in\mathcal{I}^k_{\text{mb},+}} l'(t^k - s_i^k)(\nabla t^k - \boldsymbol{x}_i),\\
g(\boldsymbol{w}^{k+1}) &= \frac{2}{n_+}\sum_{i\in\mathcal{I}^{k+1}_{\text{mb},+}} l'(t^{k+1} - s_i^{k+1})(\nabla t^{k+1} - \boldsymbol{x}_i),\\
\nabla L(\boldsymbol{w}^k) &= \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} l'(t(\boldsymbol{w}^k) - \boldsymbol{x}_i^\top \boldsymbol{w}^k)(\nabla t(\boldsymbol{w}^k) - \boldsymbol{x}_i),\\
\nabla L(\boldsymbol{w}^{k+1}) &= \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} l'(t(\boldsymbol{w}^{k+1}) - \boldsymbol{x}_i^\top \boldsymbol{w}^{k+1})(\nabla t(\boldsymbol{w}^{k+1}) - \boldsymbol{x}_i).
\end{aligned}$$

Due to the assumptions, we have $\mathcal{I}_+ = \mathcal{I}^k_{\text{mb},+} \cup \mathcal{I}^{k+1}_{\text{mb},+}$ and $\emptyset = \mathcal{I}^k_{\text{mb},+} \cap \mathcal{I}^{k+1}_{\text{mb},+}$, which allows us to write

$$\begin{aligned}
&n_+\left(g(\boldsymbol{w}^k) + g(\boldsymbol{w}^{k+1}) - \nabla L(\boldsymbol{w}^k) - \nabla L(\boldsymbol{w}^{k+1})\right) && \text{(A.21a)}\\
&\quad= \sum_{i\in\mathcal{I}^k_{\text{mb},+}} l'(t^k - s_i^k)(\nabla t^k - \boldsymbol{x}_i) - \sum_{i\in\mathcal{I}^k_{\text{mb},+}} l'(t(\boldsymbol{w}^k) - \boldsymbol{x}_i^\top \boldsymbol{w}^k)(\nabla t(\boldsymbol{w}^k) - \boldsymbol{x}_i) && \text{(A.21b)}\\
&\quad+ \sum_{i\in\mathcal{I}^k_{\text{mb},+}} l'(t^k - s_i^k)(\nabla t^k - \boldsymbol{x}_i) - \sum_{i\in\mathcal{I}^k_{\text{mb},+}} l'(t(\boldsymbol{w}^{k+1}) - \boldsymbol{x}_i^\top \boldsymbol{w}^{k+1})(\nabla t(\boldsymbol{w}^{k+1}) - \boldsymbol{x}_i) && \text{(A.21c)}\\
&\quad+ \sum_{i\in\mathcal{I}^{k+1}_{\text{mb},+}} l'(t^{k+1} - s_i^{k+1})(\nabla t^{k+1} - \boldsymbol{x}_i) - \sum_{i\in\mathcal{I}^{k+1}_{\text{mb},+}} l'(t(\boldsymbol{w}^k) - \boldsymbol{x}_i^\top \boldsymbol{w}^k)(\nabla t(\boldsymbol{w}^k) - \boldsymbol{x}_i) && \text{(A.21d)}\\
&\quad+ \sum_{i\in\mathcal{I}^{k+1}_{\text{mb},+}} l'(t^{k+1} - s_i^{k+1})(\nabla t^{k+1} - \boldsymbol{x}_i) - \sum_{i\in\mathcal{I}^{k+1}_{\text{mb},+}} l'(t(\boldsymbol{w}^{k+1}) - \boldsymbol{x}_i^\top \boldsymbol{w}^{k+1})(\nabla t(\boldsymbol{w}^{k+1}) - \boldsymbol{x}_i). && \text{(A.21e)}
\end{aligned}$$

Then relations (A.20) and (A.18) applied to Lemma A.15 imply

$$\left\|\sum_{i\in\mathcal{I}^k_{\text{mb},+}} l'(t^k - s_i^k)(\nabla t^k - \boldsymbol{x}_i) - \sum_{i\in\mathcal{I}^k_{\text{mb},+}} l'(t(\boldsymbol{w}^k) - \boldsymbol{x}_i^\top \boldsymbol{w}^k)(\nabla t(\boldsymbol{w}^k) - \boldsymbol{x}_i)\right\| \le C_4\left(\alpha^{k-1} + \alpha^{k-2}\right)$$

for some C_4, which gives a bound for (A.21b). Bound for (A.21e) is obtained by increasing k by one. Bounds for (A.21c) and (A.21d) can be find similarly using (A.19). Altogether, we showed

$$\left\|g(\boldsymbol{w}^k) + g(\boldsymbol{w}^{k+1}) - \nabla L(\boldsymbol{w}^k) - \nabla L(\boldsymbol{w}^{k+1})\right\| \le C_1(\alpha^{k-2} + \alpha^{k-1} + \alpha^k + \alpha^{k+1}) \tag{A.22}$$

for some C_1.

We now estimate

$$\begin{aligned}
&\alpha^k\left\langle g(w^k) - \nabla L(w^k),\, w^* - w^k\right\rangle + \alpha^{k+1}\left\langle g(w^{k+1}) - \nabla L(w^{k+1}),\, w^* - w^{k+1}\right\rangle \\
&\quad = \left\langle g(w^k) - \nabla L(w^k),\, \alpha^k(w^* - w^k)\right\rangle + \left\langle g(w^{k+1}) - \nabla L(w^{k+1}),\, \alpha^{k+1}(w^* - w^{k+1})\right\rangle \\
&\quad = \left\langle g(w^k) - \nabla L(w^k) + g(w^{k+1}) - \nabla L(w^{k+1}),\, \alpha^k(w^* - w^k)\right\rangle \\
&\quad + \left\langle g(w^{k+1}) - \nabla L(w^{k+1}),\, \alpha^{k+1}(w^* - w^{k+1}) - \alpha^k(w^* - w^k)\right\rangle.
\end{aligned} \tag{A.23}$$

To estimate the second part of the right hand side of (A.23), we make use of Lemma A.13 to obtain the existence of some C_5 such that

$$\begin{aligned}
&\left\langle g(w^{k+1}) - \nabla L(w^{k+1}),\, \alpha^{k+1}(w^* - w^{k+1}) - \alpha^k(w^* - w^k)\right\rangle \\
&\le 2B\left\|\alpha^{k+1}(w^* - w^{k+1}) - \alpha^k(w^* - w^k)\right\| \\
&= 2B\left\|\alpha^{k+1}(w^* - w^k + \alpha^k g(w^k)) - \alpha^k(w^* - w^k)\right\| \\
&= 2B\left\|(\alpha^{k+1} - \alpha^k)w^* + (\alpha^k - \alpha^{k+1})w^k + \alpha^k\alpha^{k+1}g(w^k)\right\| \\
&\le C_5\left\|\alpha^{k+1} - \alpha^k\right\| + C_5(\alpha^k)^2 + C_5(\alpha^{k+1})^2.
\end{aligned} \tag{A.24}$$

In the last inequality we used the inequality $2ab \le a^2 + b^2$. To estimate the first part of the right hand side of (A.23), we can apply (A.22) together with the boundedness of $\{w^k\}$ to obtain the existence of some C_6 such that

$$\begin{aligned}
&\left\langle g(w^k) - \nabla L(w^k) + g(w^{k+1}) - \nabla L(w^{k+1}),\, \alpha^k(w^* - w^k)\right\rangle \\
&\qquad \le C_6(\alpha^{k-2})^2 + C_6(\alpha^{k-1})^2 + C_6(\alpha^k)^2 + C_6(\alpha^{k+1})^2.
\end{aligned} \tag{A.25}$$

Plugging (A.24) and (A.25) into (A.23) and summing the terms yields (A.11). Then the assumptions of Theorem A.10 are verified and the theorem statement follows. ■

A.6.3 Auxiliary Results

Lemma A.12

Let l satisfy (S1)-(S3). Then there exists some $\hat{C} > 0$ such that for all k we have

$$\hat{C} \le \sum_{i\in\mathcal{I}} l'(s_i^k - t^k), \qquad \hat{C} \le \sum_{i\in\mathcal{I}} l'(x_i^\top w^k - t(w^k)).$$

Proof:

First, we will find an upper bound of $s_i^k - t^k$. Fix any index i_0. Since l is nonnegative due to (S1), equation (3.7) implies

$$n\tau = \sum_{i\in\mathcal{I}} l(s_i^k - t^k) \ge l(s_{i_0}^k - t^k).$$

Since l is a strictly increasing function due to (S2) and $n\tau > 0$, we get

$$l^{-1}(n\tau) \ge s_{i_0}^k - t^k. \tag{A.26}$$

Since i_0 was an arbitrary index, it holds true for all indices. Then (S3) which leads to a

further estimate

$$\sum_{i\in\mathcal{I}} l'(s_i^k - t^k) = \sum_{i\in\mathcal{I}} l(s_i^k - t^k)\frac{l'(s_i^k - t^k)}{l(s_i^k - t^k)} \geq \sum_{i\in\mathcal{I}} l(s_i^k - t^k)\frac{l'(l^{-1}(n\tau))}{l(l^{-1}(n\tau))} = n\tau\frac{l'(l^{-1}(n\tau))}{l(l^{-1}(n\tau))} = l'(l^{-1}(n\tau)),$$

where the inequality follows from (A.26) and the following equality from (3.7). Due to (S2) we obtain that $l'(l^{-1}(n\tau))$ is a positive number, which finishes the proof of the first part. The second part can be obtained in an identical way. ■

Lemma A.13

Let l satisfy (S1)-(S4). Then there exists some B such that for all k we have

$$\left\|\nabla L(\boldsymbol{w}^k)\right\| \leq B, \qquad \left\|g(\boldsymbol{w}^k)\right\| \leq B.$$

Proof:
Due to (S4) the derivative l' is bounded by some $\hat{B}$. Then Theorem 3.3 and Lemma A.12 imply

$$\left\|\nabla t(\boldsymbol{w}^k)\right\| \leq \frac{\hat{B}\sum_{i\in\mathcal{I}}\|\boldsymbol{x}_i\|}{\sum_{i\in\mathcal{I}} l'(\boldsymbol{x}_i^\top \boldsymbol{w}^k - t(\boldsymbol{w}^k))} \leq \frac{\hat{B}}{\hat{C}}\sum_{i\in\mathcal{I}}\|\boldsymbol{x}_i\|,$$

which is independent of k. Then (3.5) and again the boundedness of l' imply the existence of some B such that $\left\|\nabla L(\boldsymbol{w}^k)\right\| \leq B$ for all k. The proof for $g(\boldsymbol{w}^k)$ can be performed identically. ■

Lemma A.14

Consider uniformly bounded positive sequences c_1^k, c_2^k, d_1^k, d_2^k, α^k and positive constants C_1, C_2 such that for all k we have

$$\left\|c_1^k - c_2^k\right\| \leq C_1\alpha^k, \qquad \left\|d_1^k - d_2^k\right\| \leq C_1\alpha^k, \qquad d_1^k \geq C_2, \qquad d_2^k \geq C_2.$$

If $\alpha^k \to 0$, then there exists a constant C_3 such that for all sufficiently large k we have

$$\left\|\frac{c_1^k}{d_1^k} - \frac{c_2^k}{d_2^k}\right\| \leq C_3\alpha^k.$$

Proof:
Since d_1^k and d_2^k are bounded away from zero and since $\alpha^k \to 0$, we have

$$\left\|\frac{c_1^k}{d_1^k} - \frac{c_2^k}{d_2^k}\right\| \leq \max\left\{\frac{c_1^k}{d_1^k} - \frac{c_1^k + C_1\alpha^k}{d_1^k - C_1\alpha^k}, \frac{c_1^k}{d_1^k} - \frac{c_1^k - C_1\alpha^k}{d_1^k + C_1\alpha^k}\right\}.$$

The first term can be estimated as

$$\left\|\frac{c_1^k}{d_1^k} - \frac{c_1^k + C_1\alpha^k}{d_1^k - C_1\alpha^k}\right\| = \left\|\frac{(c_1^k + d_1^k)C_1\alpha^k}{d_1^k(d_1^k - C_1\alpha^k)}\right\| \leq \frac{(c_1^k + d_1^k)C_1\alpha^k}{C_2|d_1^k - C_1\alpha^k|}.$$

Since $\alpha^k \to 0$ by assumption, for large k we have $\left\|d_1^k - C_1\alpha^k\right\| \geq \frac{1}{2}C_2$. Since the sequences are uniformly bounded, the statement follows. ■

Lemma A.15

Consider scalars a_i, c_i and vectors b_i, d_i. If there is some $\hat{C}$ such that $|a_i| \leq \hat{C}$ and $\|d_i\| \leq \hat{C}$, then

$$\left\| \sum_{i=1}^{n} a_i b_i - \sum_{i=1}^{n} c_i d_i \right\| \leq \hat{C} \sum_{i=1}^{n} (|a_i - c_i| + \|b_i - d_i\|).$$

Proof:
It is simple to verify

$$\left\| \sum_{i=1}^{n} a_i b_i - \sum_{i=1}^{n} c_i d_i \right\| \leq \sum_{i=1}^{n} \|d_i\| |a_i - c_i| + \sum_{i=1}^{n} |a_i| \|b_i - d_i\|,$$

from which the statement follows. ■

B

Appendix for Chapter 4

B.1 Derivation of Dual Problems

B.1.1 Family of *TopPushK* Formulations

Theorem 4.3: Dual formulation for *TopPushK* family

Consider Notation 4.2, surrogate function l, and formulation (4.4). Then the corresponding dual problem has the following form

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{maximize}} \quad -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^{\top}\mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} - C\sum_{i=1}^{n_+} l^{\star}\left(\frac{\alpha_i}{C}\right) \tag{4.5a}$$

$$\text{subject to} \quad \sum_{i=1}^{n_+}\alpha_i = \sum_{j=1}^{\tilde{n}}\beta_j, \tag{4.5b}$$

$$0 \le \beta_j \le \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i, \quad j = 1,2,\ldots,\tilde{n}, \tag{4.5c}$$

where $l^{\star}$ is conjugate function of l and

$$\mathbb{K} = \begin{cases}\mathbb{K}^{\pm} & \text{for } \textit{TopMeanK},\\ \mathbb{K}^{-} & \text{otherwise},\end{cases} \quad \tilde{n} = \begin{cases}n & \text{for } \textit{TopMeanK},\\ n_- & \text{otherwise},\end{cases} \quad K = \begin{cases}1 & \text{for } \textit{TopPush},\\ n\tau & \text{for } \textit{TopMeanK},\\ n_-\tau & \text{for } \tau\textit{-FPL}.\end{cases}$$

If $K = 1$, the upper bound in the second constraint vanishes due to the first constraint.

Proof:
We show the proof only for *TopPushK* formulation, i.e., the decision threshold is computed only from negative samples. The proof for the remaining formulations is identical. Firstly, we derive an alternative formulation to formulation (4.4). Using Lemma 1 from [70], we can rewrite the formula for the decision threshold to the following form

$$\sum_{j=1}^{K} s^{-}_{[j]} = \min_{t}\left\{Kt + \sum_{j=1}^{n_-}\max\{0,\, s^{-}_{j} - t\}\right\}.$$

By substituting this formula into the objective function of (4.4), we get

$$\sum_{i=1}^{n_+} l\left(\frac{1}{K}\sum_{j=1}^{K} s_{[j]}^- - s_i^+\right) = \sum_{i=1}^{n_+} l\left(\frac{1}{K}\min_t\left\{Kt + \sum_{j=1}^{n_-}\max\left\{0,\, s_j^- - t\right\}\right\} - s_i^+\right)$$
$$= \min_t \sum_{i=1}^{n_+} l\left(t + \frac{1}{K}\sum_{j=1}^{n_-}\max\left\{0,\, s_j^- - t\right\} - s_i^+\right).$$

where the last equality follows from the fact that the surrogate function l is non-decreasing. The max operator can be replaced using an auxiliary variable $\boldsymbol{z} \in \mathbb{R}^{n_-}$ that fulfills $z_j \geq s_j^- - t$ and $z_j \geq 0$ for all $j = 1,\ldots, n_-$. Furthermore, we use auxilliary variable $\boldsymbol{y} \in \mathbb{R}^{n_-}$ defined for all $i = 1,\ldots, n_+$ as

$$y_i = t + \frac{1}{K}\sum_{j=1}^{n_-} z_j - s_i^+.$$

Since we use a linear model, the combination of all relations above leads to the following formulation

$$\begin{aligned}
\underset{\boldsymbol{w},t,\boldsymbol{y},\boldsymbol{z}}{\text{maximize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n_+} l(y_i) \\
\text{subject to} \quad & y_i = t + \frac{1}{K}\sum_{j=1}^{n_-} z_j - \boldsymbol{w}^\top \boldsymbol{x}_i^+, && i = 1,\, 2,\ldots,\, n_+, \\
& z_j \geq \boldsymbol{w}^\top \boldsymbol{x}_j^- - t, && j = 1,\, 2,\ldots,\, n_-, \\
& z_j \geq 0, && j = 1,\, 2,\ldots,\, n_-,
\end{aligned}$$

The Lagrangian of this formulation is defined as

$$\mathcal{L}(\boldsymbol{w},t,\boldsymbol{y},\boldsymbol{z};\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n_+} l(y_i) + \sum_{i=1}^{n_+}\alpha_i\left(t + \frac{1}{K}\sum_{j=1}^{n_-} z_j - \boldsymbol{w}^\top \boldsymbol{x}_i^+ - y_i\right)$$
$$+ \sum_{j=1}^{n_-}\beta_j\left(\boldsymbol{w}^\top \boldsymbol{x}_j^- - t - z_j\right) + \sum_{j=1}^{n_-}\gamma_j z_j,$$

with feasibility conditions $\beta_j \geq 0$ and $\gamma_j \geq 0$ for all $j = 1,\ldots, n_-$. Since the Lagrangian $\mathcal{L}$ is separable in primal variables, it can be minimized with respect to each variable separately. Then the dual function reads

$$g(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) = \min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \boldsymbol{w}^\top\left(\sum_{i=1}^{n_+}\alpha_i \boldsymbol{x}_i^+ - \sum_{j=1}^{n_-}\beta_j \boldsymbol{x}_j^-\right) \tag{B.1a}$$
$$+ \min_t\, t\left(\sum_{i=1}^{n_+}\alpha_i - \sum_{j=1}^{n_-}\beta_j\right) \tag{B.1b}$$
$$+ \min_{\boldsymbol{y}}\, C\sum_{i=1}^{n_+}\left(l(y_i) - \frac{\alpha_i}{C}y_i\right) \tag{B.1c}$$
$$+ \min_{\boldsymbol{z}} \sum_{j=1}^{n_-}\left(\sum_{i=1}^{n_+}\alpha_i - \beta_j - \gamma_j\right)z_j \tag{B.1d}$$

From optimality conditions with respect to $\boldsymbol{w}$, we deduce

$$\boldsymbol{w} = \sum_{i=1}^{n_+} \alpha_i \boldsymbol{x}_i^+ - \sum_{j=1}^{n_-} \beta_j \boldsymbol{x}_j^- = \begin{pmatrix} \mathbb{X}^+ \\ -\mathbb{X}^- \end{pmatrix}^\top \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} \implies \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \boldsymbol{w}^\top \left(\sum_{i=1}^{n_+} \alpha_i \boldsymbol{x}_i^+ - \sum_{j=1}^{n_-} \beta_j \boldsymbol{x}_j^- \right) = -\frac{1}{2} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix}^\top \mathbb{K}^- \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix},$$

where we use Notation 4.2. It mean, that we get the first part of the objective function (4.4a). Optimality condition with respect to t reads

$$\sum_{i=1}^{n_+} \alpha_i - \sum_{j=1}^{n_-} \beta_j = 0,$$

and implies constrain (4.4b). Similarly, optimality condition of (B.1d) with respect to $\boldsymbol{z}$ reads for all $j = 1, \ldots, n_-$ as

$$\frac{1}{K} \sum_{i=1}^{n_+} \alpha_i - \beta_j - \gamma_j = 0.$$

Plugging the feasibility condition $\gamma_j \geq 0$ into this equality and combining it with the feasibility conditions $\beta_j \geq 0$, yields constraint (4.4c). Finally, minimization of (B.1c) with respect to $\boldsymbol{y}$ yields for all $i = 1, \ldots, n_+$

$$C \min_{y_i} \left(l(y_i) - \frac{\alpha_i}{C} y_i \right) = -C l^\star \left(\frac{\alpha_i}{C} \right).$$

where the equality follows from Definition 4.1 of the conjugate function. Plugging this back into (B.1c) yields the second part of the objective function (4.4a). For *TopPush*, we have $K = 1$. From (4.4b) and non-negativity of β_j we deduce that the upper bound in (4.4c) is always fulfilled and can be omitted. ■

B.1.2 Family of *Pat&Mat* Formulations

Theorem 4.4: Dual formulation for *Pat&Mat* family

Consider Notation 4.2, surrogate function l, and formulation (4.6). Then the corresponding dual problem has the following form

$$\underset{\boldsymbol{\alpha}, \boldsymbol{\beta}, \delta}{\text{maximize}} \quad -\frac{1}{2} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix}^\top \mathbb{K} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} - C \sum_{i=1}^{n_+} l^\star \left(\frac{\alpha_i}{C} \right) - \delta \sum_{j=1}^{\tilde{n}} l^\star \left(\frac{\beta_j}{\delta \vartheta} \right) - \delta \tilde{n} \tau \tag{4.7a}$$

$$\text{subject to} \quad \sum_{i=1}^{n_+} \alpha_i = \sum_{j=1}^{\tilde{n}} \beta_j, \tag{4.7b}$$

$$\delta \geq 0, \tag{4.7c}$$

where $l^\star$ is conjugate function of l, $\vartheta > 0$ is a scaling parameter and

$$\mathbb{K} = \begin{cases} \mathbb{K}^\pm & \text{for } \textit{Pat\&Mat}, \\ \mathbb{K}^- & \text{otherwise}, \end{cases} \qquad \tilde{n} = \begin{cases} n & \text{for } \textit{Pat\&Mat}, \\ n_- & \text{otherwise}. \end{cases}$$

Proof:
For simplicity, we again show the proof only for *Pat&Mat-NP*, i.e. the threshold is computed only from negative samples. Let us first realize that formulation (4.6) is equivalent to the following formulation

$$\begin{aligned} \underset{\boldsymbol{w},t,\boldsymbol{y},\boldsymbol{z}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n_+} l(y_i) \\ \text{subject to} \quad & \sum_{j=1}^{n_-} l(\vartheta z_i) \leq n_-\tau, \\ & y_i = t - \boldsymbol{w}^\top \boldsymbol{x}_i^+, \quad i = 1,\, 2,\ldots,\, n_+., \\ & z_j = \boldsymbol{w}^\top \boldsymbol{x}_j^- - t, \quad j = 1,\, 2,\ldots,\, n_- \end{aligned}$$

The corresponding Lagrangian then reads

$$\begin{aligned} \mathcal{L}(\boldsymbol{w},t,\boldsymbol{y},\boldsymbol{z};\boldsymbol{\alpha},\boldsymbol{\beta},\delta) = \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n_+} l(y_i) + \sum_{i=1}^{n_+} \alpha_i(t - \boldsymbol{w}^\top \boldsymbol{x}_i^+ - y_i) \\ + \sum_{j=1}^{n_-} \beta_j(\boldsymbol{w}^\top \boldsymbol{x}_j^- - t - z_j) + \delta\left(\sum_{j=1}^{n_-} l(\vartheta z_j) - n_-\tau\right). \end{aligned}$$

with feasibility condition $\delta \geq 0$. Since the Lagrangian $\mathcal{L}$ is separable in primal variables, it can be minimized with respect to each variable separately. Then the dual function can be rewritten as follows

$$g(\boldsymbol{\alpha},\boldsymbol{\beta},\delta) = \min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \boldsymbol{w}^\top\left(\sum_{i=1}^{n_+} \alpha_i \boldsymbol{x}_i^+ - \sum_{j=1}^{n_-} \beta_j \boldsymbol{x}_j^-\right) \tag{B.2a}$$

$$+ \min_t t\left(\sum_{i=1}^{n_+} \alpha_i - \sum_{j=1}^{n_-} \beta_j\right) \tag{B.2b}$$

$$+ \min_{\boldsymbol{y}} C\sum_{i=1}^{n_+}\left(l(y_i) - \frac{\alpha_i}{C} y_i\right) \tag{B.2c}$$

$$+ \min_{\boldsymbol{z}} \delta\sum_{j=1}^{n_-}\left(l(\vartheta z_j) - \frac{\beta_j}{\delta} z_j\right) \tag{B.2d}$$

$$- \delta n_-\tau. \tag{B.2e}$$

Note that the resulting dual function is very similar to one (B.1) for *TopPushK*. In fact, the first three parts of (B.1) and (B.2) are identical. Therefore, we only have to show how to minimize (B.2) with respect to $\boldsymbol{z}$. For that, we can use the conjugate function as in the case of minimization with respect to $\boldsymbol{y}$. Then, for all $j = 1,\ldots,\, n_-$, we get

$$\delta \min_{\boldsymbol{z}}\left(l(\vartheta z_j) - \frac{\beta_j}{\delta\vartheta}\vartheta z_j\right) = -\delta l^\star\left(\frac{\beta_i}{\delta\vartheta}\right),$$

where the equality follows from Definition 4.1 of a conjugate function. Plugging this back into (B.2d) yields the third part of the objective function (4.7a), which finishes the proof. ■

B.2 Coordinate Descent Algorithm

B.2.1 Family of *TopPushK* Formulations

Hinge Loss

Proposition 4.6: Update rule (4.12a) for problem (4.14)

Consider problem (4.14), update rule (4.12a), indices $1 \leq k \leq n_+$ and $1 \leq l \leq n_+$ and Notation 4.5. Then the optimal solution $\Delta^\star$ is given by (4.13) where

$$\begin{aligned}\Delta_{lb} &= \max\{-\alpha_{\hat{k}},\ \alpha_{\hat{l}} - C\},\\ \Delta_{ub} &= \min\{C - \alpha_{\hat{k}},\ \alpha_{\hat{l}}\},\\ \gamma &= -\frac{s_k - s_l}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}}.\end{aligned}$$

Proof of Proposition 4.6 on page 39:
Constraint (4.14b) is always satisfied from the definition of the update rule (4.12a). Constraint (4.14d) is always satisfied since no β_j was updated and the sum of all α_i did not change. Constraint (4.14c) reads

$$\begin{aligned}0 \leq \alpha_{\hat{k}} + \Delta \leq C \quad &\Longrightarrow \quad -\alpha_{\hat{k}} \leq \Delta \leq C - \alpha_{\hat{k}},\\ 0 \leq \alpha_{\hat{l}} - \Delta \leq C \quad &\Longrightarrow \quad \alpha_{\hat{l}} - C \leq \Delta \leq \alpha_{\hat{l}},\end{aligned}$$

which gives the lower and upper bound of Δ. Using the update rule (4.12a), objective function (4.14a) can be rewritten as a quadratic function with respect to Δ as

$$-\frac{1}{2}[\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}]\Delta^2 - [s_k - s_l]\Delta - c(\boldsymbol{\alpha}, \boldsymbol{\beta}).$$

Finally, the optimal solution $\Delta^\star$ is given by (4.13). ■

Proposition 4.7: Update rule (4.12b) for problem (4.14)

Consider problem (4.14), update rule (4.12b), indices $1 \leq k \leq n_+$ and $n_+ + 1 \leq l \leq \tilde{n}$ and Notation 4.5. Let us define

$$\beta_{\max} = \max_{j \in \{1,2,\ldots,\tilde{n}\} \setminus \{\hat{l}\}} \beta_j.$$

Then the optimal solution $\Delta^\star$ is given by (4.13) where

$$\begin{aligned}\Delta_{lb} &= \begin{cases}\max\left\{-\alpha_{\hat{k}},\ -\beta_{\hat{l}}\right\} & \mathrm{K} = 1,\\ \max\left\{-\alpha_{\hat{k}},\ -\beta_{\hat{l}},\ K\beta_{\max} - \sum_{i=1}^{n_+}\alpha_i\right\} & \text{otherwise},\end{cases}\\ \Delta_{ub} &= \begin{cases}C - \alpha_{\hat{k}} & \mathrm{K} = 1,\\ \min\left\{C - \alpha_{\hat{k}},\ \frac{1}{K-1}\left(\sum_{i=1}^{n_+}\alpha_i - K\beta_{\hat{l}}\right)\right\} & \text{otherwise}.\end{cases}\\ \gamma &= -\frac{s_k + s_l - 1}{\mathbb{K}_{kk} + \mathbb{K}_{ll} + \mathbb{K}_{kl} + \mathbb{K}_{lk}}.\end{aligned}$$

Proof of Proposition 4.7 on page 39:
Constraint (4.14b) is always satisfied from the definition of the update rule (4.12b). Constraint (4.14c) reads

$$0 \leq \alpha_{\hat{k}} + \Delta \leq C \quad \Longrightarrow \quad -\alpha_{\hat{k}} \leq \Delta \leq C - \alpha_{\hat{k}}.$$

Using the definition of $\beta_{\max}$, constraint (4.14d) for any $K \geq 2$ reads

$$0 \leq \beta_{\max} \leq \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i + \frac{\Delta}{K} \quad \Longrightarrow \quad K\beta_{\max} - \sum_{i=1}^{n_+}\alpha_i \leq \Delta,$$
$$0 \leq \beta_{\hat{l}} + \Delta \leq \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i + \frac{\Delta}{K} \quad \Longrightarrow \quad -\beta_{\hat{l}} \leq \Delta \quad \wedge \quad \Delta \leq \frac{1}{K-1}\left(\sum_{i=1}^{n_+}\alpha_i - K\beta_{\hat{l}}\right).$$

The combination of these bounds yields the lower bound Δ_{lb} and upper bound Δ_{ub}. If $K = 1$, the upper bound in (4.14d) is always satisfied due to (4.14b) and the lower and upper bound of Δ can be simplified. Using the update rule (4.12b), objective function (4.14a) can be rewritten as a quadratic function with respect to Δ as

$$-\frac{1}{2}[\mathbb{K}_{kk} + \mathbb{K}_{ll} + \mathbb{K}_{kl} + \mathbb{K}_{lk}]\Delta^2 - [s_k + s_l - 1]\Delta - c(\boldsymbol{\alpha}, \boldsymbol{\beta}).$$

Finally, the optimal solution $\Delta^\star$ is given by (4.13). ■

Proposition 4.8: Update rule (4.12c) for problem (4.14)

Consider problem (4.14), update rule (4.12c), indices $n_+ + 1 \leq k \leq \tilde{n}$ and $n_+ + 1 \leq l \leq \tilde{n}$ and Notation 4.5. Then the optimal solution $\Delta^\star$ is given by (4.13) where

$$\Delta_{lb} = \begin{cases} -\beta_{\hat{k}} & \text{K} = 1, \\ \max\left\{-\beta_{\hat{k}},\ \beta_{\hat{l}} - \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i\right\} & \text{otherwise}, \end{cases}$$
$$\Delta_{ub} = \begin{cases} \beta_{\hat{l}} & \text{K} = 1, \\ \min\left\{\frac{1}{K}\sum_{i=1}^{n_+}\alpha_i - \beta_{\hat{k}},\ \beta_{\hat{l}}\right\} & \text{otherwise}. \end{cases}$$
$$\gamma = -\frac{s_k - s_l}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}}.$$

Proof of Proposition 4.8 on page 40:
Constraint (4.14b) is always satisfied from the definition of the update rule (4.12c). Constraint (4.14c) is also always satisfied since no α_i is updated. Constraint (4.14d) for any $K \geq 2$ reads

$$0 \leq \beta_{\hat{k}} + \Delta \leq \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i \quad \Longrightarrow \quad -\beta_{\hat{k}} \leq \Delta \leq \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i - \beta_{\hat{k}},$$
$$0 \leq \beta_{\hat{l}} - \Delta \leq \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i \quad \Longrightarrow \quad \beta_{\hat{l}} - \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i \leq \Delta \leq \beta_{\hat{l}},$$

which gives the lower and upper bound of Δ. If $K = 1$, the upper bound in (4.14d) is always satisfied due to (4.14b) and the lower and upper bound of Δ can be simplified. Using the update rule (4.12c), objective function (4.14a) can be rewritten as a quadratic function with respect to Δ as

$$-\frac{1}{2}[\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}]\Delta^2 - [s_k - s_l]\Delta - c(\boldsymbol{\alpha}, \boldsymbol{\beta}).$$

Finally, the optimal solution $\Delta^\star$ is given by (4.13). ■

Quadratic Hinge Loss

Proposition B.6: Update rule (4.12a) for problem (4.15)

Consider problem (4.15), update rule (4.12a), indeices $1 \leq k \leq n_+$ and $1 \leq l \leq n_+$ and Notation 4.5. Then the optimal solution $\Delta^\star$ is given by (4.13) where

$$\Delta_{lb} = -\alpha_{\hat{k}}, \qquad \Delta_{ub} = \alpha_{\hat{l}}, \qquad \gamma = -\frac{s_k - s_l + \frac{1}{2C}(\alpha_{\hat{k}} - \alpha_{\hat{l}})}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk} + \frac{1}{C}}.$$

Proof:
Constraint (4.15b) is always satisfied from the definition of the update rule (4.12a). Constraint (4.15d) is also always satisfied since no β_j was updated and the sum of all α_i did not change. Constraint (4.15c) reads

$$\begin{aligned} 0 \leq \alpha_{\hat{k}} + \Delta \quad &\Longrightarrow \quad -\alpha_{\hat{k}} \leq \Delta, \\ 0 \leq \alpha_{\hat{l}} - \Delta \quad &\Longrightarrow \quad \Delta \leq \alpha_{\hat{l}}, \end{aligned}$$

which gives the lower and upper bound of Δ. Using the update rule (4.12a), objective function (4.15a) can be rewritten as a quadratic function with respect to Δ as

$$-\frac{1}{2}\left[\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk} + \frac{1}{C}\right]\Delta^2 - \left[s_k - s_l + \frac{1}{2C}(\alpha_{\hat{k}} - \alpha_{\hat{l}})\right]\Delta - c(\boldsymbol{\alpha}, \boldsymbol{\beta}).$$

Finally, the optimal solution $\Delta^\star$ is given by (4.13). ■

Proposition B.7: Update rule (4.12b) for problem (4.15)

Consider problem (4.15), update rule (4.12b), indeices $1 \leq k \leq n_+$ and $n_+ + 1 \leq l \leq \tilde{n}$ and Notation 4.5. Let us define

$$\beta_{\max} = \max_{j \in \{1,2,\ldots,\tilde{n}\} \setminus \{\hat{l}\}} \beta_j.$$

Then the optimal solution $\Delta^\star$ is given by (4.13) where

$$\Delta_{lb} = \begin{cases} \max\left\{-\alpha_{\hat{k}}, -\beta_{\hat{l}}\right\} & \mathrm{K} = 1, \\ \max\left\{-\alpha_{\hat{k}}, -\beta_{\hat{l}}, K\beta_{\max} - \sum_{i=1}^{n_+} \alpha_i\right\} & \text{otherwise}, \end{cases}$$

$$\Delta_{ub} = \begin{cases} +\infty & \mathrm{K} = 1, \\ \frac{1}{K-1}\left(\sum_{i=1}^{n_+} \alpha_i - K\beta_{\hat{l}}\right) & \text{otherwise}, \end{cases}$$

$$\gamma = -\frac{s_k + s_l - 1 + \frac{1}{2C}\alpha_{\hat{k}}}{\mathbb{K}_{kk} + \mathbb{K}_{ll} + \mathbb{K}_{kl} + \mathbb{K}_{lk} + \frac{1}{2C}}.$$

Proof:
Constraint (4.15b) is always satisfied from the definition of the update rule (4.12b). Constraint (4.15c) reads

$$0 \leq \alpha_{\hat{k}} + \Delta \quad \Longrightarrow \quad -\alpha_{\hat{k}} \leq \Delta.$$

Using the definition of $\beta_{\max}$, constraint (4.15d) for any $K \geq 2$ reads

$$0 \leq \beta_{\max} \leq \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i + \frac{\Delta}{K} \quad \Longrightarrow \quad K\beta_{\max} - \sum_{i=1}^{n_+}\alpha_i \leq \Delta,$$
$$0 \leq \beta_{\hat{l}} + \Delta \leq \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i + \frac{\Delta}{K} \quad \Longrightarrow \quad -\beta_{\hat{l}} \leq \Delta \quad \wedge \quad \Delta \leq \frac{1}{K-1}\left(\sum_{i=1}^{n_+}\alpha_i - K\beta_{\hat{l}}\right).$$

The combination of these bounds yields the lower bound Δ_{lb} and upper bound Δ_{ub}. If $K = 1$, the upper bound in (4.15d) is always satisfied due to (4.15b) and the lower and upper bound of Δ can be simplified. Using the update rule (4.12b), objective function (4.15a) can be rewritten as a quadratic function with respect to Δ as

$$-\frac{1}{2}\left[\mathbb{K}_{kk} + \mathbb{K}_{ll} + \mathbb{K}_{kl} + \mathbb{K}_{lk} + \frac{1}{2C}\right]\Delta^2 - \left[s_k + s_l - 1 + \frac{1}{2C}\alpha_{\hat{k}}\right]\Delta - c(\boldsymbol{\alpha}, \boldsymbol{\beta}).$$

Finally, the optimal solution $\Delta^\star$ is given by (4.13). ■

Proposition B.8: Update rule (4.12c) for problem (4.15)

Consider problem (4.15), update rule (4.12c), indices $n_+ + 1 \leq k \leq \tilde{n}$ and $n_+ + 1 \leq l \leq \tilde{n}$ and Notation 4.5. Then the optimal solution $\Delta^\star$ is given by (4.13) where

$$\Delta_{lb} = \begin{cases} -\beta_{\hat{k}} & \mathrm{K} = 1, \\ \max\left\{-\beta_{\hat{k}},\ \beta_{\hat{l}} - \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i\right\} & \text{otherwise}, \end{cases}$$
$$\Delta_{ub} = \begin{cases} \beta_{\hat{l}} & \mathrm{K} = 1, \\ \min\left\{\beta_{\hat{l}},\ \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i - \beta_{\hat{k}}\right\} & \text{otherwise}, \end{cases}$$
$$\gamma = -\frac{s_k - s_l}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}}.$$

Proof:
Constraint (4.15b) is always satisfied from the definition of the update rule (4.12c). Constraint (4.15c) is also always satisfied since no α_i is updated. Constraint (4.15d) for any $K \geq 2$ reads

$$0 \leq \beta_{\hat{k}} + \Delta \leq \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i \quad \Longrightarrow \quad -\beta_{\hat{k}} \leq \Delta \leq \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i - \beta_{\hat{k}},$$
$$0 \leq \beta_{\hat{l}} - \Delta \leq \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i \quad \Longrightarrow \quad \beta_{\hat{l}} - \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i \leq \Delta \leq \beta_{\hat{l}},$$

which gives the lower and upper bound of Δ. If $K = 1$, the upper bound in (4.15d) is always satisfied due to (4.15b) and the lower and upper bound of Δ can be simplified. Using the update rule (4.12c), objective function (4.15a) can be rewritten as a quadratic function with respect to Δ as

$$-\frac{1}{2}[\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}]\Delta^2 - [s_k - s_l]\Delta - c(\boldsymbol{\alpha}, \boldsymbol{\beta}).$$

Finally, the optimal solution $\Delta^\star$ is given by (4.13). ■

Initialization

Theorem 4.9

Consider problem (4.18), some initial solution $\boldsymbol{\alpha}^0$, $\boldsymbol{\beta}^0$ and denote the sorted version (in non-decreasing order) of $\boldsymbol{\beta}^0$ as $\boldsymbol{\beta}^0_{[\cdot]}$. Then if the following condition holds

$$\sum_{j=1}^{K}\left(\beta^0_{[\tilde{n}-K+j]} + \max_{i=1,\dots,n_+} \alpha_i^0\right) \leq 0, \tag{4.17}$$

the optimal solution of (4.18) amounts to $\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0}$. In the opposite case, the following system of two equations

$$\sum_{i=1}^{n_+} \operatorname{clip}_{[0,\,C_1]}\left(\alpha_i^0 - \lambda + \frac{1}{K}\sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,+\infty)}\left(\beta_j^0 + \lambda - \mu\right)\right) - K\mu = 0, \tag{4.18a}$$

$$\sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu]}\left(\beta_j^0 + \lambda\right) - K\mu = 0, \tag{4.18b}$$

has a solution (λ, μ) with $\mu > 0$, and the optimal solution of (4.18) is equal to

$$\alpha_i = \operatorname{clip}_{[0,\,C_1]}\left(\alpha_i^0 - \lambda + \frac{1}{K}\sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,+\infty)}\left(\beta_j^0 + \lambda - \mu\right)\right),$$
$$\beta_j = \operatorname{clip}_{[0,\,\mu]}\left(\beta_j^0 + \lambda\right).$$

Proof of Theorem 4.9 on page 41:
The Lagrangian of (4.18) reads

$$\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\beta}; \lambda, \boldsymbol{p}, \boldsymbol{q}, \boldsymbol{u}, \boldsymbol{v}) = \frac{1}{2}\left\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^0\right\|^2 + \frac{1}{2}\left\|\boldsymbol{\beta} - \boldsymbol{\beta}^0\right\|^2 + \lambda\left(\sum_{i=1}^{n_+}\alpha_i - \sum_{j=1}^{\tilde{n}}\beta_j\right)$$
$$- \sum_{i=1}^{n_+} p_i\alpha_i + \sum_{i=1}^{n_+} q_j(\alpha_i - C_1) - \sum_{j=1}^{\tilde{n}} u_j\beta_j + \sum_{j=1}^{\tilde{n}} v_j\left(\beta_j - \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i\right).$$

The KKT conditions then amount to the optimality conditions

$$\frac{\partial \mathcal{L}}{\partial \alpha_i} = \alpha_i - \alpha_i^0 + \lambda - p_i + q_i - \frac{1}{K}\sum_{j=1}^{\tilde{n}} v_j = 0, \qquad i = 1, 2, \dots, n_+, \tag{B.3a}$$

$$\frac{\partial \mathcal{L}(\cdot)}{\partial \beta_j} = \beta_j - \beta_j^0 - \lambda - u_j + v_j = 0, \qquad j = 1, 2, \dots, \tilde{n}, \tag{B.3b}$$

the primal feasibility conditions (4.18), the dual feasibility conditions $\lambda \in \mathbb{R}$, $p_i \geq 0$, $q_i \geq 0$,

$u_j \geq 0$, $v_j \geq 0$ and finally the complementarity conditions

$$p_i \alpha_i = 0, \qquad i = 1, 2, \ldots, n_+, \tag{B.3c}$$
$$q_i(\alpha_i - C_1) = 0, \qquad i = 1, 2, \ldots, n_+, \tag{B.3d}$$
$$u_j \beta_j = 0, \qquad j = 1, 2, \ldots, \tilde{n}, \tag{B.3e}$$
$$v_j \left(\beta_j - \frac{1}{K} \sum_{i=1}^{n_+} \alpha_i \right) = 0, \qquad j = 1, 2, \ldots, \tilde{n}. \tag{B.3f}$$

Case 1: The first case concerns when the optimal solution satisfies $\sum_i \alpha_i = 0$. From the primal feasibility conditions, we immediately get $\alpha_i = 0$ for all i and $\beta_j = 0$ for all j. Then (B.3d) implies $q_i = 0$ for all i and all complementarity conditions are satisfied. Moreover, optimality condition (B.3a) implies

$$\lambda = \alpha_i^0 + p_i + \frac{1}{K} \sum_{j=1}^{\tilde{n}} v_j.$$

Since the only condition on p_i is the non-negativity, this implies

$$\lambda \geq \max_{i=1,\ldots,n_+} \alpha_i^0 + \frac{1}{K} \sum_{j=1}^{\tilde{n}} v_j.$$

Similarly, from optimality condition (B.3b) we deduce

$$v_j = \beta_j^0 + \lambda + u_j \geq \beta_j^0 + \lambda \geq \beta_j^0 + \max_{i=1,\ldots,n_+} \alpha_i^0 + \frac{1}{K} \sum_{i=1}^{\tilde{n}} v_i.$$

Since we need to fulfill $v_j \geq 0$, this amounts to

$$v_j \geq \operatorname{clip}_{[0,\,+\infty)} \left(\beta_j^0 + \max_{i=1,\ldots,n_+} \alpha_i^0 + \frac{1}{K} \sum_{i=1}^{\tilde{n}} v_i \right).$$

Summing this with respect to j and using the substitution $\bar{v} = \frac{1}{K} \sum_i v_i$ results in

$$K\bar{v} - \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,+\infty)} \left(\beta_j^0 + \max_{i=1,\ldots,n_+} \alpha_i^0 + \bar{v} \right) = 0. \tag{B.4}$$

Denote by $\beta_{[j]}^0$ the sorted version of β_j^0. Then the function on the left-hand side of (B.4) as a function of $\bar{v}$ is increasing on $\left(-\infty, -\beta_{[n_+-K+1]}^0 - \max_i \alpha_i^0\right]$ and non-decreasing otherwise. Thus, (B.4) can be satisfied if and only if its function value at $-\beta_{[n_+-K+1]}^0 - \max_i \alpha_i^0$ is non-negative. But this is precisely the violation of (4.17).

Case 2: If (4.17) holds true, then from the discussion above we obtain that the optimal solution satisfies $\sum_i \alpha_i > 0$. For simplicity, we define

$$\bar{\alpha} = \frac{1}{K} \sum_{i=1}^{n_+} \alpha_i, \qquad \bar{\beta} = \frac{1}{K} \sum_{j=1}^{\tilde{n}} \beta_j, \qquad \bar{v} = \frac{1}{K} \sum_{j=1}^{\tilde{n}} v_j.$$

For any fixed i, the standard trick is to combine the optimality condition (B.3a) with the primal feasibility condition $0 \le \alpha_i \le C_1$, the dual feasibility conditions $p_i \ge 0$, $q_i \ge 0$ and the complementarity conditions (B.3c, B.3d) to obtain

$$\alpha_i = \operatorname{clip}_{[0,\, C_1]}\left(\alpha_i^0 - \lambda + \bar{v}\right). \tag{B.5}$$

Similarly for any fixed j, we combine the optimality condition (B.3b) with the primal feasibility condition $0 \le \beta_j \le \bar{\alpha}$, the dual feasibility conditions $u_j \ge 0$, $v_j \ge 0$ and the complementarity conditions (B.3e, B.3f) to obtain

$$\beta_j = \operatorname{clip}_{[0,\, \bar{\alpha}]}\left(\beta_j^0 + \lambda\right), \tag{B.6}$$

$$v_j = \operatorname{clip}_{[0,\, +\infty)}\left(\beta_j^0 + \lambda - \bar{\alpha}\right). \tag{B.7}$$

Summing equations (B.5), (B.6) and (B.7) respectively with respect to i and j results in

$$K\bar{\alpha} = \sum_{i=1}^{n_+} \operatorname{clip}_{[0,\, C_1]}\left(\alpha_i^0 - \lambda + \bar{v}\right), \tag{B.8a}$$

$$K\bar{\beta} = \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\, \bar{\alpha}]}\left(\beta_j^0 + \lambda\right), \tag{B.8b}$$

$$K\bar{v} = \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\, +\infty)}\left(\beta_j^0 + \lambda - \bar{\alpha}\right). \tag{B.8c}$$

We denote $\mu = \bar{\alpha}$. Then (4.18a) results by plugging (B.8c) into (B.8a) while (4.18b) follows from (B.8b) and $\sum_i \alpha_i = \sum_j \beta_j$. ■

Lemma 4.10

Even though $\lambda(\mu)$ is not unique, function h from (4.19) is well-defined in the sense that it gives the same value for every choice of $\lambda(\mu)$. Moreover, h is decreasing in μ on $(0, +\infty)$.

Proof of Lemma 4.10 on page 42:
Recall that based on (4.18b) we defined

$$g(\lambda; \mu) := \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\, \mu]}\left(\beta_j^0 + \lambda\right) - K\mu,$$

and solutions of $g(\lambda; \mu) = 0$ for a fixed μ are denoted by $\lambda(\mu)$. Function $g(\cdot; \mu)$ is non-decreasing and since K is an integer, the only case when the solution to $g(\lambda) = 0$ is not unique happens when the optimal solution $\lambda(\mu)$ satisfies

$$\beta_{[j]}^0 + \lambda(\mu) \begin{cases} \ge \mu & \text{for } j = \tilde{n} - K + 1, \ldots, \tilde{n}, \\ \le 0 & \text{otherwise.} \end{cases}$$

Here, we again denote $\beta^0_{[\cdot]}$ to be the sorted version of β^0_j. Then h defined in (4.19) equals to

$$\begin{aligned} h(\mu) &= \sum_{i=1}^{n_+} \operatorname{clip}_{[0,\,C_1]}\left(\alpha_i^0 - \lambda(\mu) + \frac{1}{K}\sum_{j=\tilde{n}-K+1}^{\tilde{n}} \left(\beta_j^0 + \lambda(\mu) - \mu\right)\right) - K\mu \\ &= \sum_{i=1}^{n_+} \operatorname{clip}_{[0,\,C_1]}\left(\alpha_i^0 - \mu + \frac{1}{K}\sum_{j=\tilde{n}-K+1}^{\tilde{n}} \beta_j^0\right) - K\mu. \end{aligned}$$

This implies the first statement of the lemma stating that h is independent of the choice of $\lambda(\mu)$. Now we need to show that h is a decreasing function. Fix any $\mu_2 > \mu_1 > 0$. From (4.18b) we have

$$\sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu_1]}\left(\beta_j^0 + \lambda(\mu_1)\right) - K\mu_1 = 0, \tag{B.9}$$

$$\sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu_2]}\left(\beta_j^0 + \lambda(\mu_2)\right) - K\mu_2 = 0. \tag{B.10}$$

Equation (B.9) implies that at most K values of $\beta_j^0 + \lambda(\mu_1)$ are greater or equal than μ_1. If we increase the upper bound in the projection, at most K values can increase, which results in

$$\sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu_2]}\left(\beta_j^0 + \lambda(\mu_1)\right) \le \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu_1]}\left(\beta_j^0 + \lambda(\mu_1)\right) + K(\mu_2 - \mu_1) = K\mu_2, \tag{B.11}$$

where the equality follows from (B.9). Comparing (B.10) and (B.11) yields $\lambda(\mu_2) \ge \lambda(\mu_1)$. Now define

$$J = \left\{ j \;\middle|\; \beta_j^0 + \lambda(\mu_1) \ge 0 \right\}$$

and observe that due to (B.9) we have $|J| \ge K$. Moreover, the definition of J and equation (B.9) yields

$$\sum_{j\in J} \operatorname{clip}_{[0,\,\mu_1]}\left(\beta_j^0 + \lambda(\mu_1)\right) - K\mu_1 = \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu_1]}\left(\beta_j^0 + \lambda(\mu_1)\right) - K\mu_1 = 0. \tag{B.12}$$

Then we have

$$\begin{aligned} \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu_2]}\left(\beta_j^0 + \lambda(\mu_1) + \mu_2 - \mu_1\right) &\ge \sum_{j\in J} \operatorname{clip}_{[0,\,\mu_2]}\left(\beta_j^0 + \lambda(\mu_1) + \mu_2 - \mu_1\right) \\ = \sum_{j\in J} \operatorname{clip}_{[\mu_2-\mu_1,\,\mu_2]}\left(\beta_j^0 + \lambda(\mu_1) + \mu_2 - \mu_1\right) &= \sum_{j\in J} \operatorname{clip}_{[0,\,\mu_1]}\left(\beta_j^0 + \lambda(\mu_1)\right) + |J|(\mu_2 - \mu_1) \\ &= K\mu_1 + |J|(\mu_2 - \mu_1) \ge K\mu_1 + K(\mu_2 - \mu_1) = K\mu_2, \end{aligned}$$

where the first equality follows from the definition of J and the second equality is a shift by a $\mu_2 - \mu_1$. The third equality follows from (B.12) and finally, the last inequality follows from $|J| \ge K$. The chain above together with (B.10) implies $\lambda(\mu_2) - \mu_2 \le \lambda(\mu_1) - \mu_1$. Combining this with $\mu_2 > \mu_1$ and $\lambda(\mu_2) \ge \lambda(\mu_1)$, this implies that h from (4.19) is non-increasing which is precisely the lemma statement. ∎

B.2.2 Family of *Pat&Mat* Formulations

Hinge Loss

Lemma B.11: Update rule (4.12a) for problem (4.20)

Consider problem (4.20), update rule (4.12a), indices $1 \le k \le n_+$ and $1 \le l \le n_+$ and Notation 4.5. Then the optimal solution $\Delta^\star$ is given by (4.13) where

$$\Delta_{lb} = \min\{-\alpha_{\hat{k}},\ \alpha_{\hat{l}} - C\}, \qquad \Delta_{ub} = \max\{C - \alpha_{\hat{k}},\ \alpha_{\hat{l}}\},$$
$$\gamma = -\frac{s_k - s_l}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}}, \qquad \delta^\star = \delta.$$

Proof:
Constraint (4.20b) is always satisfied from the definition of the update rule (4.12a). Constraint (4.20d) is also always satisfied since no β_j was updated and the sum of all α_i did not change. Constraint (4.20c) reads

$$0 \le \alpha_{\hat{k}} + \Delta \le C \quad \Longrightarrow \quad -\alpha_{\hat{k}} \le \Delta \le C - \alpha_{\hat{k}}$$
$$0 \le \alpha_{\hat{l}} - \Delta \le C \quad \Longrightarrow \quad \alpha_{\hat{l}} - C \le \Delta \le \alpha_{\hat{l}}$$

which gives the lower and upper bound of Δ. Using the update rule (4.12a), objective function (4.20a) can be rewritten as a quadratic function with respect to Δ as

$$-\frac{1}{2}[\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}]\Delta^2 - [s_k - s_l]\Delta - c(\boldsymbol{\alpha}, \boldsymbol{\beta}).$$

The optimal solution $\Delta^\star$ is given by (4.13). Finally, since optimal δ is given by (4.21) and no β_j was updated, the optimal δ does not change. ■

Lemma B.12: Update rule (4.12b) for problem (4.20)

Consider problem (4.20), update rule (4.12b), indices $1 \le k \le n_+$ and $n_+ + 1 \le l \le \tilde{n}$ and Notation 4.5. Let us define

$$\beta_{\max} = \max_{j \in \{1,2,\ldots,\tilde{n}\} \setminus \{\hat{l}\}} \beta_j.$$

Then the bounds from (4.13) are defined as $\Delta_{lb} = \max\{-\alpha_{\hat{k}},\ -\beta_{\hat{l}}\}$ and $\Delta_{ub} = C - \alpha_{\hat{k}}$ and there are two possible solutions

1. $\Delta_1^\star$ is feasible if $\beta_{\hat{l}} + \Delta_1^\star \le \beta_{\max}$ and is given by (4.13) where

$$\gamma = -\frac{s_k + s_l - 1 - \frac{1}{\vartheta}}{\mathbb{K}_{kk} + \mathbb{K}_{ll} + \mathbb{K}_{kl} + \mathbb{K}_{lk}}, \qquad \delta_1^* = \frac{\beta_{\max}}{\vartheta}.$$

2. $\Delta_2^\star$ is feasible if $\beta_{\hat{l}} + \Delta_2^\star \ge \beta_{\max}$ and is given by (4.13) where

$$\gamma = -\frac{s_k + s_l - 1 - \frac{1-\tilde{n}\tau}{\vartheta}}{\mathbb{K}_{kk} + \mathbb{K}_{ll} + \mathbb{K}_{kl} + \mathbb{K}_{lk}}, \qquad \delta_2^* = \frac{\beta_{\hat{l}} + \Delta_2^\star}{\vartheta}.$$

The optimal solution $\Delta^\star$ is equal to one of them, which maximizes the original objective and is feasible.

Proof:
Constraint (4.20b) is always satisfied from the definition of the update rule (4.12b). Constraint (4.20c) reads $-\alpha_{\hat{k}} \leq \Delta \leq C - \alpha_{\hat{k}}$. Using the definition of $\beta_{\max}$, constraint (4.20d) reads $\beta_{\max} \leq \delta\vartheta$ and $0 \leq \beta_{\hat{l}} + \Delta \leq \delta\vartheta$. Since the optimal δ is given by (4.21), there are only two possible choices: $\delta_1^\star = \beta_{\max}/\vartheta$ and $\delta_2^\star = \beta_{\hat{l}}+\Delta/\vartheta$. If δ is feasible, all upper bounds in constraint (4.20d) hold. Therefore, we can simplify the constraints to $-\beta_{\hat{l}} \leq \Delta$, which in combination with bounds for $\alpha_{\hat{k}}$ gives the lower and upper bound of Δ. Now let us discuss how to select optimal δ :

1. Using $\delta_1^\star$ and the update rule (4.12a), objective function (4.20a) can be rewritten as a quadratic function with respect to Δ as

$$-\frac{1}{2}[\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}]\Delta^2 - \left[s_k - s_l - 1 - \frac{1}{\vartheta}\right]\Delta - c(\boldsymbol{\alpha}, \boldsymbol{\beta}).$$

   The optimal solution $\Delta_1^\star$ is given by (4.13) and is feasible if $\beta_{\hat{l}} + \Delta_1^\star \leq \beta_{\max}$.

2. Using $\delta_2^\star$ and the update rule (4.12a), objective function (4.20a) can be rewritten as a quadratic function with respect to Δ as

$$-\frac{1}{2}[\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}]\Delta^2 - \left[s_k - s_l - 1 - \frac{1-\tilde{n}\tau}{\vartheta}\right]\Delta - c(\boldsymbol{\alpha}, \boldsymbol{\beta}).$$

   The optimal solution $\Delta_2^\star$ is given by (4.13) and is feasible if $\beta_{\hat{l}} + \Delta_2^\star \geq \beta_{\max}$.

The optimal solution is the one, which maximizes the objective (4.20a) and is feasible. ■

Lemma B.13: Update rule (4.12c) for problem (4.20)

Consider problem (4.20), update rule (4.12c), indices $n_+ + 1 \leq k \leq \tilde{n}$ and $n_+ + 1 \leq l \leq \tilde{n}$ and Notation 4.5. Let us define

$$\beta_{\max} = \max_{j \in \{1,2,\ldots,\tilde{n}\} \setminus \{\hat{k},\hat{l}\}} \beta_j.$$

Then the bounds from (4.13) are defined as $\Delta_{lb} = -\beta_{\hat{k}}$ and $\Delta_{ub} = \beta_{\hat{l}}$ and there are three possible solutions

1. $\Delta_1^\star$ is feasible if $\beta_{\max} \geq \max\{\beta_{\hat{k}} + \Delta_1^\star, \beta_{\hat{l}} - \Delta_1^\star\}$ and is given by (4.13) where

$$\gamma = -\frac{s_k - s_l}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}}, \qquad \delta_1^* = \frac{\beta_{\max}}{\vartheta}.$$

2. $\Delta_2^\star$ is feasible if $\beta_{\hat{k}} + \Delta_2^\star \geq \max\{\beta_{\max}, \beta_{\hat{l}} - \Delta_2^\star\}$ and is given by (4.13) where

$$\gamma = -\frac{s_k - s_l + \frac{\tilde{n}\tau}{\vartheta}}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}}, \qquad \delta_2^* = \frac{\beta_{\hat{k}} + \Delta_2^\star}{\vartheta}.$$

3. $\Delta_3^\star$ is feasible if $\beta_{\hat{l}} - \Delta_3^\star \geq \max\{\beta_{\hat{k}} + \Delta_3^\star, \beta_{\max}\}$ and is given by (4.13) where

$$\gamma = -\frac{s_k - s_l - \frac{\tilde{n}\tau}{\vartheta}}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}}, \qquad \delta_3^* = \frac{\beta_{\hat{k}} - \Delta_3^\star}{\vartheta}.$$

The optimal solution $\Delta^\star$ is equal to one of them, which maximizes the original objective and is feasible.

Proof:
Constraint (4.20b) is always satisfied from the definition of the update rule (4.12c). Constraint (4.20c) is also always satisfied since no α_i is updated. Using the definition of $\beta_{\max}$, constraint (4.20d) reads

$$\begin{aligned} \beta_{\max} &\leq \delta\vartheta, \\ 0 \leq \beta_{\hat{k}} + \Delta &\leq \delta\vartheta, \\ 0 \leq \beta_{\hat{l}} - \Delta &\leq \delta\vartheta. \end{aligned}$$

Since the optimal δ is given by (4.21), there are only two possible choices

$$\delta_1^\star = \frac{\beta_{\max}}{\vartheta}, \qquad \delta_2^\star = \frac{\beta_{\hat{k}} + \Delta}{\vartheta}, \qquad \delta_3^\star = \frac{\beta_{\hat{l}} - \Delta}{\vartheta}. \tag{B.13}$$

If we use any of these choices which is feasible, all upper bounds in constraint (4.20d) hold, i.e. we can simplify the constraints to

$$\begin{aligned} 0 \leq \beta_{\hat{k}} + \Delta \quad &\Longrightarrow \quad -\beta_{\hat{k}} \leq \Delta, \\ 0 \leq \beta_{\hat{l}} - \Delta \quad &\Longrightarrow \quad \Delta \leq \beta_{\hat{l}}, \end{aligned}$$

which gives the lower and upper bound of Δ. Now let us discuss how to select optimal δ :

1. Using $\delta_1^\star$ from (B.13) and the update rule (4.12a), objective function (4.20a) can be rewritten as a quadratic function with respect to Δ as

   $$-\frac{1}{2}[\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}]\Delta^2 - [s_k - s_l]\Delta - c(\boldsymbol{\alpha}, \boldsymbol{\beta}).$$

   The optimal solution $\Delta_1^\star$ is given by (4.13) and is feasible if

   $$\beta_{\max} \geq \max\{\beta_{\hat{k}} + \Delta_1^\star,\ \beta_{\hat{l}} - \Delta_1^\star\}.$$

2. Using $\delta_2^\star$ from (B.13) and the update rule (4.12a), objective function (4.20a) can be rewritten as a quadratic function with respect to Δ as

   $$-\frac{1}{2}[\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}]\Delta^2 - \left[s_k - s_l + \frac{\tilde{n}\tau}{\vartheta}\right]\Delta - c(\boldsymbol{\alpha}, \boldsymbol{\beta}).$$

   The optimal solution $\Delta_2^\star$ is given by (4.13) and is feasible if

   $$\beta_{\hat{k}} + \Delta_2^\star \geq \max\{\beta_{\max}, \beta_{\hat{l}} - \Delta_2^\star\}.$$

3. Using $\delta_3^\star$ from (B.13) and the update rule (4.12a), objective function (4.20a) can be rewritten as a quadratic function with respect to Δ as

   $$-\frac{1}{2}[\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}]\Delta^2 - \left[s_k - s_l - \frac{\tilde{n}\tau}{\vartheta}\right]\Delta - c(\boldsymbol{\alpha}, \boldsymbol{\beta}).$$

   The optimal solution $\Delta_3^\star$ is given by (4.13) and is feasible if

   $$\beta_{\hat{l}} - \Delta_3^\star \geq \max\{\beta_{\max}, \beta_{\hat{k}} + \Delta_3^\star\}.$$

The optimal solution is the one, which maximizes the objective (4.20a) and is feasible. ■

Quadratic Hinge Loss

Lemma B.14: Update rule (4.12a) for problem (4.22)

Consider problem (4.22), update rule (4.12a), indices $1 \leq k \leq n_+$ and $1 \leq l \leq n_+$ and Notation 4.5. Then the optimal solution $\Delta^\star$ is given by (4.13) where

$$\begin{aligned}
\Delta_{lb} &= -\alpha_{\hat{k}}, \\
\Delta_{ub} &= \alpha_{\hat{l}}, \\
\gamma &= -\frac{s_k - s_l + \frac{1}{2C}(\alpha_{\hat{k}} - \alpha_{\hat{l}})}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk} + \frac{1}{C}}, \\
\delta^\star &= \delta.
\end{aligned}$$

Proof of Lemma B.14 on page 100:
Constraint (4.22c) is always satisfied from the definition of the update rule (4.12a). Constraint (4.22e) is also always satisfied since no β_j was updated and the sum of all α_i did not change. Constraint (4.22d) reads

$$\begin{aligned}
0 \leq \alpha_{\hat{k}} + \Delta \quad &\Longrightarrow \quad -\alpha_{\hat{k}} \leq \Delta, \\
0 \leq \alpha_{\hat{l}} - \Delta \quad &\Longrightarrow \quad \Delta \leq \alpha_{\hat{l}},
\end{aligned}$$

which gives the lower and upper bound of Δ. Using the update rule (4.12a), objective function (4.22a-4.22b) can be rewritten as a quadratic function with respect to Δ as

$$-\frac{1}{2}\left[\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk} + \frac{1}{C}\right]\Delta^2 - \left[s_k - s_l + \frac{1}{2C}(\alpha_{\hat{k}} - \alpha_{\hat{l}})\right]\Delta - c(\boldsymbol{\alpha}, \boldsymbol{\beta}).$$

The optimal solution $\Delta^\star$ is given by (4.13). Finally, since optimal δ is given by (4.23) and no β_j was updated, the optimal δ does not change. ■

Lemma B.15: Update rule (4.12b) for problem (4.22)

Consider problem (4.22), update rule (4.12b), indices $1 \leq k \leq n_+$ and $n_+ + 1 \leq l \leq \tilde{n}$ and Notation 4.5. Then the optimal solution $\Delta^\star$ is given by (4.13) where

$$\begin{aligned}
\Delta_{lb} &= \max\{-\alpha_{\hat{k}}, -\beta_{\hat{l}}\}, \\
\Delta_{ub} &= +\infty, \\
\gamma &= -\frac{s_k + s_l - 1 + \frac{\alpha_{\hat{k}}}{2C} - \frac{1}{\vartheta} + \frac{\beta_{\hat{l}}}{2\delta\vartheta^2}}{\mathbb{K}_{kk} + \mathbb{K}_{ll} + \mathbb{K}_{kl} + \mathbb{K}_{lk} + \frac{1}{2C} + \frac{1}{2\delta\vartheta^2}}, \\
\delta^\star &= \sqrt{\delta^2 + \frac{1}{4\vartheta\tilde{n}\tau}(\Delta^{\star 2} + 2\Delta^\star\beta_{\hat{l}})}.
\end{aligned}$$

Proof of Lemma B.15 on page 100:
Constraint (4.22c) is always satisfied from the definition of the update rule (4.12b). Constraints (4.22d) and (4.22e) reads

$$\begin{aligned}
0 \leq \alpha_{\hat{k}} + \Delta \quad &\Longrightarrow \quad -\alpha_{\hat{k}} \leq \Delta, \\
0 \leq \beta_{\hat{l}} + \Delta \quad &\Longrightarrow \quad -\beta_{\hat{l}} \leq \Delta,
\end{aligned}$$

which gives the lower bound of Δ. In this case, Δ has no upper bound. Using the update rule (4.12b), objective function (4.22a-4.22b) can be rewritten as a quadratic function with respect to Δ as

$$-\frac{1}{2}\left[\mathbb{K}_{kk}+\mathbb{K}_{ll}+\mathbb{K}_{kl}+\mathbb{K}_{lk}+\frac{1}{2C}+\frac{1}{2\delta\vartheta^2}\right]\Delta^2 - \left[s_k+s_l-1+\frac{\alpha_{\hat{k}}}{2C}-\frac{1}{\vartheta}+\frac{\beta_{\hat{l}}}{2\delta\vartheta^2}\right]\Delta - c(\boldsymbol{\alpha},\boldsymbol{\beta}).$$

The optimal solution $\Delta^\star$ is given by (4.13). We know that the optimal δ^* is given by (4.23), then

$$\delta^* = \sqrt{\frac{1}{4\vartheta^2\tilde{n}\tau}\left(\sum_{j\neq\hat{l}}\beta_j^2+(\beta_{\hat{l}}+\Delta^\star)^2\right)} = \sqrt{\delta^2+\frac{1}{4\vartheta^2\tilde{n}\tau}(\Delta^{\star 2}+2\Delta^\star\beta_{\hat{l}})}.$$

■

Lemma B.16: Update rule (4.12c) for problem (4.22)

Consider problem (4.22), update rule (4.12c) indices $n_+ + 1 \leq k \leq \tilde{n}$ and $n_+ + 1 \leq l \leq \tilde{n}$ and Notation 4.5. Then the optimal solution $\Delta^\star$ is given by (4.13) where

$$\begin{aligned}
\Delta_{lb} &= -\beta_{\hat{k}},\\
\Delta_{ub} &= \beta_{\hat{l}},\\
\gamma &= -\frac{s_k - s_l + \frac{1}{2\delta\vartheta^2}(\beta_{\hat{k}}-\beta_{\hat{l}})}{\mathbb{K}_{kk}+\mathbb{K}_{ll}-\mathbb{K}_{kl}-\mathbb{K}_{lk}+\frac{1}{\delta\vartheta^2}},\\
\delta^\star &= \sqrt{\delta^2+\frac{1}{2\vartheta\tilde{n}\tau}(\Delta^{\star 2}+\Delta^\star(\beta_{\hat{k}}-\beta_{\hat{l}}))}.
\end{aligned}$$

Proof of Lemma B.16 on page 101:
Constraint (4.22c) is always satisfied from the definition of the update rule (4.12c). Constraint (4.22d) is also always satisfied since no α_i is updated. Constraint (4.22e) reads

$$\begin{aligned}
0 \leq \beta_{\hat{k}} + \Delta \quad &\Longrightarrow \quad -\beta_{\hat{k}} \leq \Delta,\\
0 \leq \beta_{\hat{l}} + \Delta \quad &\Longrightarrow \quad -\beta_{\hat{l}} \leq \Delta,
\end{aligned}$$

which gives the lower and upper bound of Δ. Using the update rule (4.12c), objective function (4.22a-4.22b) can be rewritten as a quadratic function with respect to Δ as

$$-\frac{1}{2}\left[\mathbb{K}_{kk}+\mathbb{K}_{ll}-\mathbb{K}_{kl}-\mathbb{K}_{lk}+\frac{1}{2\delta\vartheta^2}\right]\Delta^2 - \left[s_k-s_l+\frac{1}{\delta\vartheta^2}(\beta_{\hat{k}}-\beta_{\hat{l}})\right]\Delta - c(\boldsymbol{\alpha},\boldsymbol{\beta}).$$

The optimal solution $\Delta^\star$ is given by (4.13). We know that the optimal δ^* is given by (4.23), then

$$\delta^* = \sqrt{\frac{1}{4\vartheta^2\tilde{n}\tau}\left(\sum_{j\notin\{\hat{l},\hat{k}\}}\beta_j^2+(\beta_{\hat{k}}+\Delta^\star)^2+(\beta_{\hat{l}}-\Delta^\star)^2\right)} = \sqrt{\delta+\frac{1}{2\vartheta^2\tilde{n}\tau}(\Delta^{\star 2}+\Delta^\star(\beta_{\hat{k}}-\beta_{\hat{l}}))}.$$

■

Initialization

Theorem 4.11

Consider problem (4.26) and some initial solution $\boldsymbol{\alpha}^0$, $\boldsymbol{\beta}^0$ and δ^0. Then if the following condition holds

$$\delta^0 \leq -C_2 \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,+\infty)}\left(\beta_j^0 + \max_{i=1,\dots,n_+} \alpha_i^0\right). \tag{4.25}$$

the optimal solution of (4.18) amounts to $\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0}$ and $\delta^0 = 0$. In the opposite case, the following system of two equations

$$0 = \sum_{i=1}^{n_+} \operatorname{clip}_{[0,\,C_1]}\left(\alpha_i^0 - \lambda\right) - \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\lambda+\mu]}\left(\beta_j^0 + \lambda\right), \tag{4.26a}$$

$$\lambda = C_2\delta^0 + C_2^2 \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,+\infty)}\left(\beta_j^0 - \mu\right) - \mu. \tag{4.26b}$$

has a solution (λ, μ) with $\lambda + \mu > 0$ and the optimal solution of (4.26) is equal to

$$\begin{aligned} \alpha_i &= \operatorname{clip}_{[0,\,C_1]}\left(\alpha_i^0 - \lambda\right), \\ \beta_j &= \operatorname{clip}_{[0,\,\lambda+\mu]}\left(\beta_j^0 + \lambda\right), \\ C_2\delta &= \lambda + \mu. \end{aligned}$$

Proof of Theorem 4.11 on page 45:
The Lagrangian of (4.26) reads

$$\begin{aligned} \mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\beta}; \lambda, \boldsymbol{p}, \boldsymbol{q}, \boldsymbol{u}, \boldsymbol{v}) = \frac{1}{2}\left\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^0\right\|^2 + \frac{1}{2}\left\|\boldsymbol{\beta} - \boldsymbol{\beta}^0\right\|^2 + \frac{1}{2}(\delta - \delta^0)^2 + \lambda\left(\sum_{i=1}^{n_+}\alpha_i - \sum_{j=1}^{\tilde{n}}\beta_j\right) \\ - \sum_{i=1}^{n_+} p_i\alpha_i + \sum_{i=1}^{n_+} q_j(\alpha_i - C_1) - \sum_{j=1}^{\tilde{n}} u_j\beta_j + \sum_{j=1}^{\tilde{n}} v_j(\beta_j - C_2\delta). \end{aligned}$$

The KKT conditions then amount to the optimality conditions

$$\frac{\partial \mathcal{L}}{\partial \alpha_i} = \alpha_i - \alpha_i^0 + \lambda - p_i + q_i = 0, \qquad i = 1, 2, \dots, n_+, \tag{B.14a}$$

$$\frac{\partial \mathcal{L}(\cdot)}{\partial \beta_j} = \beta_j - \beta_j^0 - \lambda - u_j + v_j = 0, \qquad j = 1, 2, \dots, \tilde{n}, \tag{B.14b}$$

$$\frac{\partial \mathcal{L}(\cdot)}{\partial \delta} = \delta - \delta^0 - C_2 \sum_{j=1}^{\tilde{n}} v_j = 0, \tag{B.14c}$$

the primal feasibility conditions (4.26), the dual feasibility conditions $\lambda \in \mathbb{R}$, $p_i \geq 0$, $q_i \geq 0$,

$u_j \geq 0$, $v_j \geq 0$ and finally the complementarity conditions

$$p_i \alpha_i = 0, \qquad i = 1, 2, \ldots, n_+, \tag{B.14d}$$
$$q_i(\alpha_i - C_1) = 0, \qquad i = 1, 2, \ldots, n_+, \tag{B.14e}$$
$$u_j \beta_j = 0, \qquad j = 1, 2, \ldots, \tilde{n}, \tag{B.14f}$$
$$v_j\big(\beta_j - C_2\delta\big) = 0, \qquad j = 1, 2, \ldots, \tilde{n}. \tag{B.14g}$$

Case 1: The first case concerns when the optimal solution satisfies $\delta = 0$. From the primal feasibility conditions, we immediately get $\alpha_i = 0$ for all i and $\beta_j = 0$ for all j. Then (B.14e) implies $q_i = 0$ and all complementarity conditions are satisfied. Moreover, (B.14a) implies for all i

$$\lambda = \alpha_i^0 + p_i.$$

Since the only condition on p_i is the non-negativity, this implies $\lambda \geq \max_i \alpha_i^0$. Similarly, from (B.14b) we deduce

$$v_j = \beta_j^0 + \lambda + u_j \geq \beta_j^0 + \lambda \geq \beta_j^0 + \max_{i=1,\ldots,n_+} \alpha_i^0.$$

Since we also have the non-negativity constraint on u_j, this implies

$$v_j \geq \operatorname{clip}_{[0,\,+\infty)}\Big(\beta_j^0 + \max_{i=1,\ldots,n_+} \alpha_i^0\Big).$$

Condition (B.14c) implies

$$\delta^0 = -C_2 \sum_{j=1}^{\tilde{n}} v_j \leq -C_2 \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,+\infty)}\Big(\beta_j^0 + \max_{i=1,\ldots,n_+} \alpha_i^0\Big).$$

This corresponds to the first case in the theorem statement and the violation of condition (4.25).

Case 2: If (4.25) holds true, then from the discussion above we obtain that the optimal solution satisfies $\delta > 0$. For any fixed i, the standard trick is to combine the optimality condition (B.14a) with the primal feasibility condition $0 \leq \alpha_i \leq C_1$, the dual feasibility conditions $p_i \geq 0$, $q_i \geq 0$ and the complementarity conditions (B.14d, B.14e) to obtain

$$\alpha_i = \operatorname{clip}_{[0,\,C_1]}\big(\alpha_i^0 - \lambda\big). \tag{B.15}$$

Similarly for any fixed j, we combine the optimality condition (B.14b) with the primal feasibility condition $0 \leq \beta_j \leq C_2\delta$, the dual feasibility conditions $u_j \geq 0$, $v_j \geq 0$ and the complementarity conditions (B.14f, B.14g) to obtain

$$\beta_j = \operatorname{clip}_{[0,\,C_2\delta]}\big(\beta_j^0 + \lambda\big), \tag{B.16}$$
$$v_j = \operatorname{clip}_{[0,\,+\infty)}\big(\beta_j^0 + \lambda - C_2\delta\big). \tag{B.17}$$

Note that we now obtain the following system

$$\sum_{i=1}^{n_+} \operatorname{clip}_{[0,\, C_1]}\left(\alpha_i^0 - \lambda\right) - \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\, C_2\delta]}\left(\beta_j^0 + \lambda\right) = 0,$$

$$\delta - \delta^0 - C_2 \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\, +\infty)}\left(\beta_j^0 + \lambda - C_2\delta\right) = 0.$$

Here, the first equation follows from plugging (B.15) and (B.16) into the feasibility condition $\sum_i \alpha_i = \sum_j \beta_j$ while the second equation follows from plugging (B.17) into (B.14c). Finally, system (4.26) follows after making the substitution $C_2\delta = \lambda + \mu$. ■

Lemma 4.12

Function h is non-decreasing in μ on $(0, \infty)$.

Proof of Lemma 4.12 on page 45:
Consider any $\mu_1 < \mu_2$. Then from (4.26b) we obtain both $\lambda(\mu_1) \geq \lambda(\mu_2)$ and $\mu_1 + \lambda(\mu_1) \geq \mu_2 + \lambda(\mu_2)$. The statement then follows from the definition of h in (4.27). ■

C

Appendix for Chapter 5

Lemma 5.3

Let $j^\star$ be unique. Assume that the selection of positive and negative samples into the minibatch is independent and that the threshold is computed from negative samples while the objective is computed from positive samples. Then the conditional expectation of the sampled gradient satisfies

$$\mathbb{E}\Big[\nabla \hat{L}(\boldsymbol{w}) \Big| j^\star_{\text{mb}} = j^\star\Big] = \nabla L(\boldsymbol{w}).$$

Proof of Lemma 5.3 on page 51:
If $j^\star$ is unique, then the true threshold t is a differentiable function of weights $\boldsymbol{w}$. The differentiability of L and $\hat{L}$ follows from the chain rule. If $j^\star_{\text{mb}} = j^\star$ holds, then the sampled gradient equals to

$$\nabla \hat{L}(\boldsymbol{w}) = \frac{1}{n_{\text{mb},+}} \sum_{i \in \mathcal{I}_{\text{mb},+}} l'(t - s_i)\Big(\nabla f(\boldsymbol{x}_{j^\star}; \boldsymbol{w}) - \nabla f(\boldsymbol{x}_i; \boldsymbol{w})\Big). \tag{C.1}$$

The summands are identical to the ones in (5.2). Since the sum is performed with respect to positive samples, the threshold is computed from negative samples, the lemma statement follows. ■

Theorem 5.4

Under the assumptions of Lemma 5.3, the bias of the sampled gradient from (5.4) satisfies

$$\text{bias}(\boldsymbol{w}) = \mathbb{P}\Big[j^\star_{\text{mb}} \neq j^\star\Big]\Big(\nabla L(\boldsymbol{w}) - \mathbb{E}\Big[\nabla \hat{L}(\boldsymbol{w}) \Big| j^\star_{\text{mb}} \neq j^\star\Big]\Big). \tag{5.8}$$

Proof of Theorem 5.4 on page 51:
The law of total expectation implies

$$\mathbb{E}\nabla \hat{L}(\boldsymbol{w}) = \mathbb{P}\Big[j^\star_{\text{mb}} = j^\star\Big]\mathbb{E}\Big[\nabla \hat{L}(\boldsymbol{w}) \Big| j^\star_{\text{mb}} = j^\star\Big] + \mathbb{P}\Big[j^\star_{\text{mb}} \neq j^\star\Big]\mathbb{E}\Big[\nabla \hat{L}(\boldsymbol{w}) \Big| j^\star_{\text{mb}} \neq j^\star\Big],$$

from where the statement follows due to definiton (5.4) and Lemma 5.3. ■

Bibliography

[1] Giuseppe Viale. The current state of breast cancer classification. *Annals of Oncology*, 23:207–210, 2012.

[2] Daniel Lévy and Arzav Jain. Breast mass classification from mammograms using deep convolutional neural networks. *arXiv preprint arXiv:1612.00542*, 2016.

[3] Martin Grill and Tomáš Pevný. Learning combination of anomaly detectors for security domain. *Computer Networks*, 107:55–63, 2016.

[4] Karen Scarfone, Peter Mell, et al. Guide to intrusion detection and prevention systems (idps). *NIST special publication*, 800(2007):94, 2007.

[5] Giorgio Giacinto and Fabio Roli. Intrusion detection in computer networks by multiple classifier systems. In *Object recognition supported by user interaction for service robots*, volume 2, pages 390–393. IEEE, 2002.

[6] Shashank Shanbhag and Tilman Wolf. Accurate anomaly detection through parallelism. *IEEE network*, 23(1):22–28, 2009.

[7] Frederick Kaefer, Carrie M Heilman, and Samuel D Ramenofsky. A neural network application to consumer classification to improve the timing of direct marketing activities. *Computers & Operations Research*, 32(10):2595–2615, 2005.

[8] Xi-Zheng Zhang. Building personalized recommendation system in e-commerce using association rule-based mining and classification. In *2007 International Conference on Machine Learning and Cybernetics*, volume 7, pages 4113–4118. IEEE, 2007.

[9] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[10] Charles E Metz. Basic principles of roc analysis. In *Seminars in nuclear medicine*, volume 8, pages 283–298. Elsevier, 1978.

[11] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann. The balanced accuracy and its posterior distribution. In *2010 20th international conference on pattern recognition*, pages 3121–3124. IEEE, 2010.

[12] Mohammad Hossin and Md Nasir Sulaiman. A review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process*, 5(2):1, 2015.

[13] James P Egan and James Pendleton Egan. *Signal detection theory and ROC-analysis*. Academic press, 1975.

[14] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[15] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

[16] Corinna Cortes and Mehryar Mohri. Auc optimization vs. error rate minimization. *Advances in neural information processing systems*, 16, 2003.

[17] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *The Journal of machine learning research*, 4:933–969, 2003.

[18] Yoav Freund and Robert 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.

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

[20] Shivani Agarwal. The infinite push: A new support vector ranking algorithm that directly optimizes accuracy at the absolute top of the list. In *Proceedings of the 2011 SIAM International Conference on Data Mining*, pages 839–850. SIAM, 2011.

[21] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[22] Nan Li, Rong Jin, and Zhi-Hua Zhou. Top rank optimization in linear time. In *Advances in neural information processing systems*, NIPS'14, pages 1502–1510, Cambridge, MA, USA, 2014. MIT Press.

[23] Stephen Boyd, Corinna Cortes, Mehryar Mohri, and Ana Radovanovic. Accuracy at the top. In *Advances in neural information processing systems*, pages 953–961, 2012.

[24] Thorsten Joachims. A support vector method for multivariate performance measures. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pages 377–384, New York, NY, USA, 2005. ACM.

[25] Elad ET Eban, Mariano Schain, Alan Mackey, Ariel Gordon, Rif A Saurous, and Gal Elidan. Scalable learning of non-decomposable objectives. In *Artificial Intelligence and Statistics*, pages 832–840, 2017.

[26] Dirk Tasche. A plug-in approach to maximising precision at the top and recall at the top. *arXiv preprint arXiv:1804.03077*, 2018.

[27] Jerzy Neyman and Egon Sharpe Pearson. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231(694-706):289–337, 1933.

[28] Maksim Lapin, Matthias Hein, and Bernt Schiele. Top-k multiclass svm. In *Advances in Neural Information Processing Systems*, pages 325–333, 2015.

[29] Ao Zhang, Nan Li, Jian Pu, Jun Wang, Junchi Yan, and Hongyuan Zha. *tau*-fpl: Tolerance-constrained learning in linear time. *arXiv preprint arXiv:1801.04701*, 2018.

[30] Lukáš Adam, Václav Mácha, Václav Šmídl, and Tomáš Pevný. General framework for binary classification on top samples. *Optimization Methods and Software*, pages 1–32, 2021.

[31] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[32] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.

[33] Václav Mácha, Lukáš Adam, and Václav Šmídl. Nonlinear classifiers for ranking problems based on kernelized svm. *arXiv preprint arXiv:2002.11436*, 2020.

[34] Alan Mackey, Xiyang Luo, and Elad Eban. Constrained classification and ranking via quantiles. *arXiv preprint arXiv:1803.00067*, 2018.

[35] Lukáš Adam and Martin Branda. Machine learning approach to chance-constrained problems: An algorithm based on the stochastic gradient descent. *arXiv preprint arXiv:1905.10986*, 2019.

[36] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[37] Shai Shnlev-Shwartz and Tong Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. In *31st International Conference on Machine Learning, ICML 2014*, volume 1, page 111, 2014.

[38] Takafumi Kanamori, Akiko Takeda, and Taiji Suzuki. Conjugate relation between loss functions and uncertainty sets in classification problems. *The Journal of Machine Learning Research*, 14(1):1461–1504, 2013.

[39] Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. Coordinate descent method for large-scale l2-loss linear support vector machines. *Journal of Machine Learning Research*, 9(Jul):1369–1398, 2008.

[40] Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S Sathiya Keerthi, and Sellamanickam Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th international conference on Machine learning*, pages 408–415. ACM, 2008.

[41] Zeynep Batmaz, Ali Yurekli, Alper Bilge, and Cihan Kaleli. A review on deep learning for recommender systems: challenges and remedies. *Artificial Intelligence Review*, 52(1):1–37, 2019.

[42] Tino Werner. A review on ranking problems in statistical learning. *arXiv preprint arXiv:1909.02998*, 2019.

[43] Lukáš Adam and V Mácha. Projections onto the canonical simplex with additional linear inequalities. *Optimization Methods and Software*, pages 1–29, 2020.

[44] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[45] Peter W. Glynn. Importance sampling for monte carlo estimation of quantiles. In *Proc. 2nd St. Petersburg Workshop on Simulation*, pages 180–185, St Petersburg, Russia, 1996. Publishing House of Saint Petersburg University.

[46] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019.

[47] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[48] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[49] Vincent G Sigillito, Simon P Wing, Larrie V Hutton, and Kile B Baker. Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest*, 10(3):262–266, 1989.

[50] Pierre Baldi, Kyle Cranmer, Taylor Faucett, Peter Sadowski, and Daniel Whiteson. Parameterized neural networks for high-energy physics. *The European Physical Journal C*, 76(5):235, 2016.

[51] Isabelle Guyon, Steve Gunn, Asa Ben-Hur, and Gideon Dror. Result analysis of the nips 2003 feature selection challenge. In *Advances in neural information processing systems*, pages 545–552, 2005.

[52] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7:1–30, 2006.

[53] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[54] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[55] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[56] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[57] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009.

[58] Abhishek Kumar, Harikrishna Narasimhan, and Andrew Cotter. Implicit rate-constrained optimization of non-decomposable objectives. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5861–5871. PMLR, 2021.

[59] Andrew Cotter, Heinrich Jiang, Maya R Gupta, Serena Wang, Taman Narayan, Seungil You, and Karthik Sridharan. Optimization with non-differentiable constraints with applications to fairness, recall, churn, and other goals. *Journal of Machine Learning Research*, 20(172):1–59, 2019.

[60] Martin Engilberge, Louis Chevallier, Patrick Pérez, and Matthieu Cord. Sodeep: a sorting deep net to learn ranking loss surrogates. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10792–10801, 2019.

[61] Thibaut Thonet, Yagmur Gizem Cinar, Eric Gaussier, Minghan Li, and Jean-Michel Renders. Smoothi: Smooth rank indicators for differentiable ir metrics. *arXiv preprint arXiv:2105.00942*, 2021.

[62] Rizal Fathony and J Zico Kolter. Ap-perf: Incorporating generic performance metrics in differentiable learning. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

[63] Alex Krizhevsky, Geoffrey Hinton, et al. *Learning multiple layers of features from tiny images*. Citeseer, 2009.

[64] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *Advances in neural information processing systems*, NIPS'11, pages 1502–1510, 2011.

[65] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision (IJCV)*, 115(3):211–252, 2015.

[66] Junshui Ma, Robert P Sheridan, Andy Liaw, George E Dahl, and Vladimir Svetnik. Deep neural nets as a method for quantitative structure–activity relationships. *Journal of chemical information and modeling*, 55(2):263–274, 2015.

[67] Harikrishna Narasimhan, Andrew Cotter, and Maya Gupta. Optimizing generalized rate metrics with three players. In *Advances in Neural Information Processing Systems*, pages 10747–10758, 2019.

[68] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[69] Tomáš Pevný and Petr Somol. Using neural network formalism to solve multiple-instance problems. In *International Symposium on Neural Networks*, pages 135–142. Springer, 2017.

[70] Wlodzimierz Ogryczak and Arie Tamir. Minimizing the sum of the k largest functions in linear time. *Information Processing Letters*, 85(3):117–122, 2003.