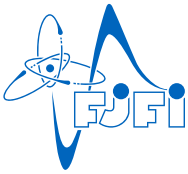
Czech Technical University in Prague
Faculty of Nuclear Sciences and
Physical Engineering

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

Klíčová slova: Keywords keywords keywords keywords keywords keywords keywords keywords keywords keywords keywords keywords keywords

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

$$\begin{aligned} \underset{\boldsymbol{w}, t}{\text{minimize}} \quad & C_1 \sum_{i \in \mathcal{I}_-} \mathbb{1}_{[s_i \geq t]} + C_2 \sum_{i \in \mathcal{I}_+} \mathbb{1}_{[s_i < t]} \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \end{aligned} \tag{1.1}$$

where $C_1, C_2 \in \mathbb{R}$, 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

$$\hat{y}_i = \begin{cases} 1 & \text{if } s_i \geq t, \\ 0 & \text{otherwise.} \end{cases} \tag{1.3}$$

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 $\hat{y}_i$ from (1.3) 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. $y_i = 1 \ \wedge \ \hat{y}_i = 1$.

If we assign each sample from dataset $\mathcal{D}$ to one of the categories above and count the number of samples in each of these four categories, we get the confusion matrix (sometimes also called contingency table) [9], see Figure 1.1. A confusion matrix consists of four fields that contains number of true-negative (**tn**), false-positive (**fp**), false-negative (**fn**), and true-positive (**tp**) samples in the whole dataset. More formally, using the prediction rule (1.3) we can compute all fields of the confusion matrix as follows

$$\begin{aligned} \mathrm{tp}(\boldsymbol{s}, t) &= \sum_{i \in \mathcal{I}_+} \mathbb{1}_{[s_i \geq t]}, & \mathrm{fn}(\boldsymbol{s}, t) &= \sum_{i \in \mathcal{I}_+} \mathbb{1}_{[s_i < t]}, \\ \mathrm{tn}(\boldsymbol{s}, t) &= \sum_{i \in \mathcal{I}_-} \mathbb{1}_{[s_i < t]}, & \mathrm{fp}(\boldsymbol{s}, t) &= \sum_{i \in \mathcal{I}_-} \mathbb{1}_{[s_i \geq t]}, \end{aligned} \tag{1.4}$$

where $\boldsymbol{s}$ is the vector of classification scores given by model f, and $\mathbb{1}_{[\cdot]}$ 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). 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

$$\mathrm{tpr} = \frac{\mathrm{tp}}{n_+}, \qquad \mathrm{fnr} = \frac{\mathrm{fn}}{n_+}, \qquad \mathrm{tnr} = \frac{\mathrm{tn}}{n_-}, \qquad \mathrm{fpr} = \frac{\mathrm{fp}}{n_-}. \tag{1.5}$$

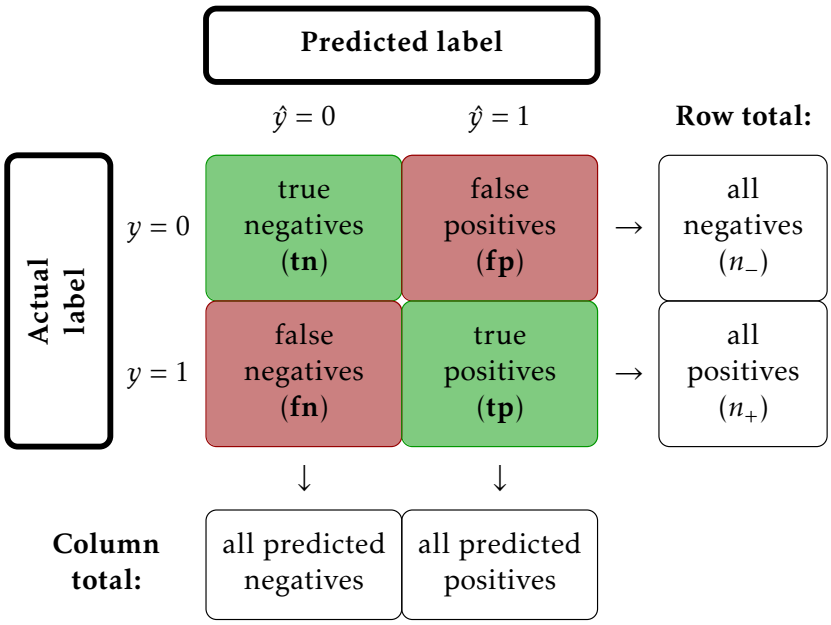


Figure 1.1: The confusion matrix for the binary classification problem, where the negative class has label 0 and the positive class has label 1. The true (target) label is denoted by y and predicted label is denoted by $\hat{y}$.

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. If we increase the value of the threshold t, we decrease the false-positive rate, but at the same time, we also increase the false-negative rate. On the other hand, if we decrease the value of t, we decrease the false-negative rate, but at the same time, we also increase the false-positive rate. In other words, it is not possible to decrease the false-positive rate only by moving the threshold t without increasing the false-negative rate and vice versa. Therefore, we always have to find some balance between these two types of errors.

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 (1.5) and rewrite the problem (1.1) to

$$\begin{aligned} \underset{\boldsymbol{w}, t}{\text{minimize}} \quad & C_1 \cdot \mathrm{fp}(\boldsymbol{s}, t) + C_2 \cdot \mathrm{fn}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}. \end{aligned} \tag{1.6}$$

The parameters C_1, $C_2 \in \mathbb{R}$ 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

$$\mathrm{acc} = \frac{1}{n}(\mathrm{tp} + \mathrm{tn}) \qquad\qquad \mathrm{bacc} = \frac{1}{2}(\mathrm{tpr} + \mathrm{tnr})$$

Note that the objective function in (1.6) is accuracy if $C_1 = C_2 = \frac{1}{n}$. Moreover, for $C_1 = \frac{1}{2n_-}$ and $C_2 = \frac{1}{2n_+}$, the objective function is balanced accuracy. This show the importance of these two performance matrices for standard binary classification. More performance metrics derived from the confusion matrix can be found in Table 1.1. Moreover, in the following section, we introduce a different approach for the performance evaluation of binary classifiers.

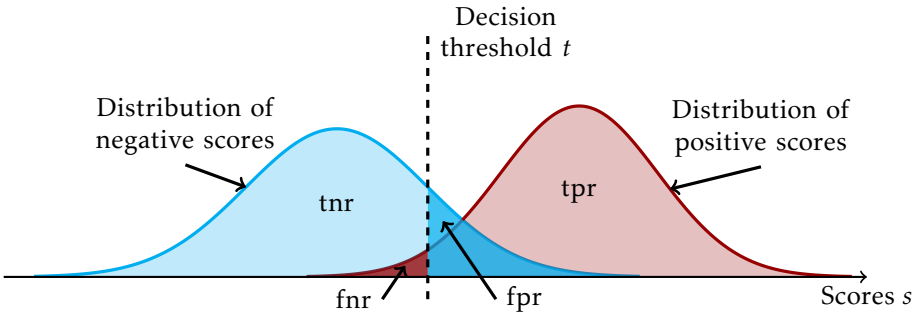


Figure 1.2: The relation between classification scores and rates. The blue/red curve is the theoretical distribution of the scores of negative/positive samples, respectively. The area between the blue line and the x-axis is divided by the decision threshold t. The left part represents a true-negative rate, while the right part represents a false-positive rate. The area between the red line and the x-axis is also divided by t. The left part represents a false-negative rate, and a right represents the true-positive rate.

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

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.

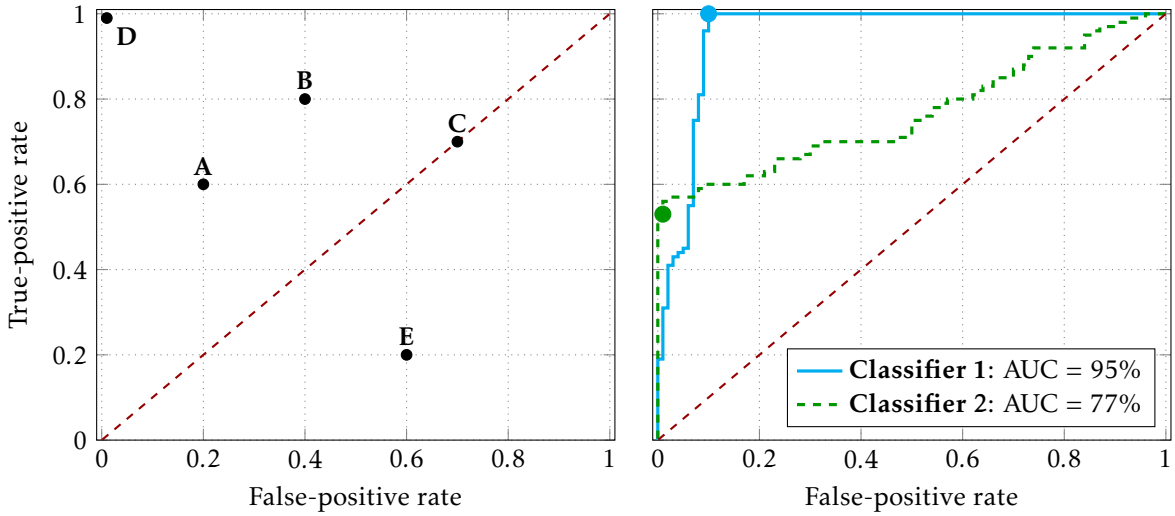


Figure 1.3: A basic representation of the ROC space with five different classifiers. (**left**) A comparison of ROC curves for two different classifiers. (**right**)

1.1.2 ROC Analysis

In the previous section, we defined a general binary classification formulation (1.6) that minimizes a weighted sum of false-positive and false-negative counts. Therefore, we always have to find some trade-off between the false-positive and false-negative counts and select the best hyperparameters C_1, C_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. The Receiver Operating Characteristic (ROC) space [13, 9] is one way to visualize the trade-off between false-positive and false-negative errors.

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 that classifies all samples correctly since fpr = 0 and tpr = 1.

ROC representation allows us to decide whether one classifier is better than another, only in some cases. For example, in Figure 1.3, classifier **B** is better than classifier **C** since **B** has a higher true-positive rate and at the same time a lower false-positive rate. 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 $(0.7, 0.7)$. Such 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 prediction 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**. 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**.

Many classifiers only predict whether 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. In this text, we consider only classifier from Notation 1.2, which predict a continuous score instead of a hard prediction. 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 shows 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 blue dot, while the blue line represents its ROC curve. **Classifier 2** represented by the green dot provides accuracy 76%, and the green dashed line represents its ROC curve. A standard method for comparing two classifiers is to compare the corresponding areas under the ROC curves (AUC) [14, 15]. Such an approach is a simple way to reduce the 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 blue classifier has AUC 95% while the green one has only 77%. Therefore, for most classification problems, the blue classifier is better. Even though we get almost the same values of accuracy and AUC for both classifiers, the accuracy is not equivalent to AUC. The similarity is only a consequence of the used example.

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. 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.

1.2 Classification at the Top

As discussed above, **Classifier 1** focuses on the overall performance, while the **Classifier 2** on the performance on low false-positive rates, see Figure 1.3. 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 the worst negative samples (negative samples with the largest classification scores). In this section, we use two different visual representations of the performance of classifiers from the right-hand side of Figure A to show the difference and emphasize the advantages of both of them.

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 or equal than the worst negative sample. In other words, **Classifier 2** maximizes true-positive rate at the smallest possible false-positive rate. If we go back to the example with search engines, the goal of **Classifier 2** is to push as many relevant results before the first

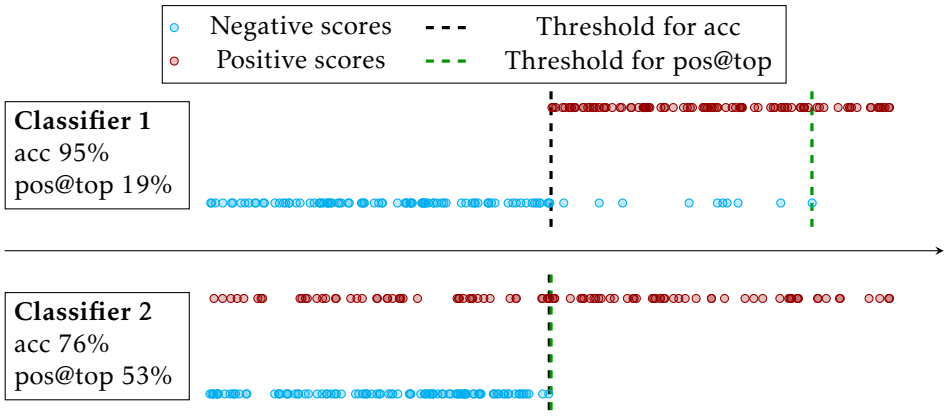


Figure 1.4: Difference between standard classifiers (**Classifier 1**) and classifiers maximizing pos@top metric (**Classifier 2**). While the former has a good total accuracy, the latter has a good pos@top metric.

irrelevant. Formally, **Classifier 2** maximizes the following metric

$$\text{pos@top}(\boldsymbol{s}) = \frac{1}{n_+} \sum_{i \in \mathcal{I}_+} \mathbb{1}_{[s_i \geq \max_{j \in \mathcal{I}_-} s_j]}. \tag{1.7}$$

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 (1.7) 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**, but 53% for **Classifier 2**.

The same behavior can also be demonstrated using ROC curves. Figure 1.5 show ROC curves for both classifier with (rihght) and without (left) logaritmic scaling of x-axis. The blue line represents ROC curve for **Classifier 1** and the green dashed one for **Classifier 2**. Moreover, There are two important points for **Classifier 2**. The blue filled circle corresponds to the black threshold and the blue filled square 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, since 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 allows us to concentrate on very low false-positive rates. If the false-positive rate is lower than $7 \cdot 10^{-1}$, then **Classifier 2** provides better true-positive rate than **Classifier 1**. Finally, the value of metric (1.7) is highlighted using squares for both classifiers and it is clear, that **Classifier 2** provides higher value of this metric.

The rest of the chapter presents three main categories of problems that focus only on a small number of the most relevant samples. Moreover, in Chapter 2, 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 the category of ranking problems. The ranking algorithms play a crucial role in many information retrieval problems:

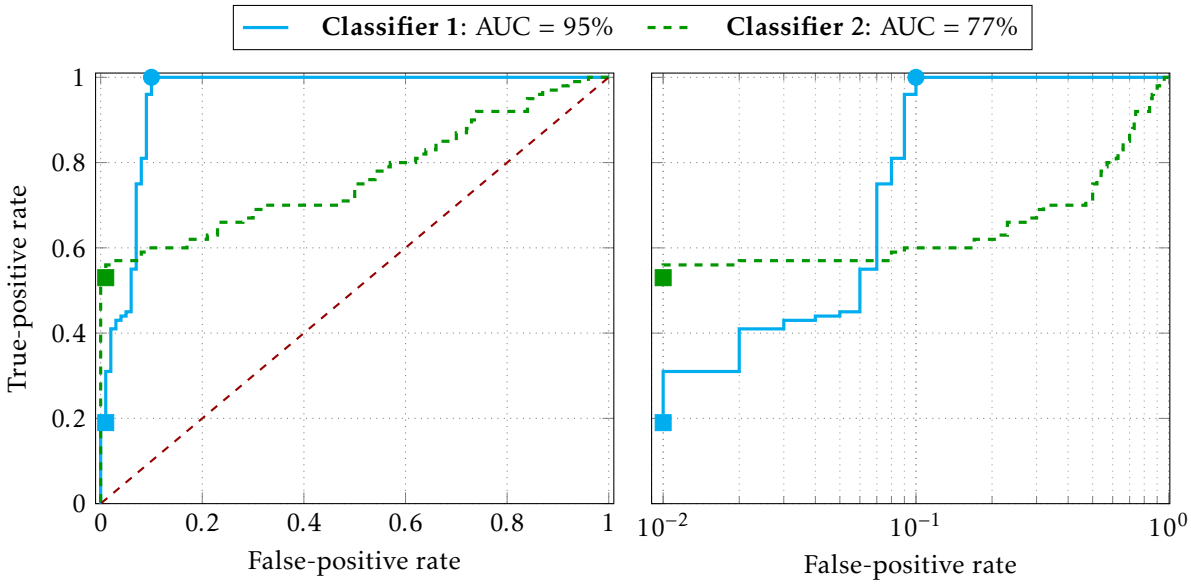


Figure 1.5: Difference between standard classifiers (**Classifier 1**) and classifiers maximizing pos@top metric (**Classifier 2**). While the former has a good total accuracy, the latter has a good pos@top metric.

- **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 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 $|\Phi(\boldsymbol{x}_i, \boldsymbol{x}_j)|$, the higher the preference. Many ranking algorithms try to find some ordering of all samples that minimizes the number of incorectly ordered pairs of samples. 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

$$\underset{r}{\text{minimize}} \quad \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} \mathbb{1}_{[r(\boldsymbol{x}_i) \leq r(\boldsymbol{x}_j)]} \cdot \max\Big\{0,\ \Phi(\boldsymbol{x}_i, \boldsymbol{x}_j)\Big\}. \tag{1.8}$$

This problem is computationally demanding 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. It is not important 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

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

The authors of [19] also proposed boosting algorithm called p-**Norm Push** to solve the formulation above. Note that for $p = 1$, the formulation (1.9) is very similar to the RankBoost (1.8). In such a case, the resulting ranking function maximizes the AUC and therefore focuses on optimizing overall rankig. On the other hand, for $p \to +\infty$, the formulation (1.9) minimizes the largest number of positive samples ranked below any negative sample

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

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. Moreover, the formulation above can be equivalently rewritten as follows

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

Authors of [20] focus on this formulation and introduces SVM (*Support Vector Machines* [21]) based algorithm called *Infinite Push* to solve it. Finally, authors of [22] proposed an even more efficient algorithm with the linear complexity in the number of samples called *TopPush*. Note that the objective function of the problem above is almost the same as the metric (1.7). 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 the following

formulation and it called **Accuracy at the Top**

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

where $f : \mathbb{R}^d \to \mathbb{R}$ is a model. This formulation focuses on the top τ-fraction of all samples and tries to maximize the number of positive samples and minimize the number of negative samples in it. 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, since the formulation is complicated due to the top τ-quantile in the constraint. 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 to reject it. 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.

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

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.

1.3 Summary

In this chapter, we introduced the general formulation (1.6) 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. More precisely, the rest of the text is organized as follows.

- Chapter 2 introduces a general optimization framework for classification at the top. Many problems fall into this framework even though they are usually considered separate problems. We describe Ranking problems, Accuracy at the Top, and the Neyman-Pearson problem in more detail and show that many formulations from these three categories fall into the framework. Moreover, we derive two new formulations closely related to the existing one. Finally, we discussed the basic properties and relations between introduced formulations. All formulations are introduced in a general form with arbitrary model f, even though many of them have been initially designed only for a linear model. Theoretical properties of the formulations with different models are discussed later.

- 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 guarantees for the optimal solution based on these properties.

- Chapter 4 is dedicated to the dual forms of formulations from Table 2.1. In this chapter, we again assume a linear model only and 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 is dedicated to all numerical experiments.
- Chapter 6 summarizes all results presented in this work.

Chapter 2 is crucial for the whole work since it introduces all formulations that are studied in the rest of the work. On the other hand, Chapters 3, 4, and 5 study the properties of these formulations in three different settings. Therefore, these three chapters can be read separately.

Note 1.3

To improve the readability of the main part of the work, we postpone many results into appendices. Main results are presented in the main part, but all auxiliary results and proofs are located in appendices.

2

Framework for Classification at the Top

In the previous chapter, we introduced the general formulation (1.6) 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 all 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 and show that all these problems fall into the following unified framework for binary classification at the top

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & C_1 \cdot \mathrm{fp}(\boldsymbol{s}, t) + C_2 \cdot \mathrm{fn}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = G(\boldsymbol{s}, \boldsymbol{y}), \end{aligned} \tag{2.1}$$

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 (1.6): 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 in (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 (1.4)), 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. The usual approach is to employ a surrogate function to replace the Iverson function [22, 3].

Notation 2.1: Surrogate function

To approximate the Iverson function (1.2), we use any surrogate function l that is convex, non-negative, and non-decreasing with $l(0) = 1$, and $l(s) \to 0$ as $s \to -\infty$. As examples of such function, we can mention the hinge loss or the quadratic hinge loss defined by

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

Figure 2.1 compares the Iverson function with the hinge and quadratic hinge loss with scaled inputs by $\vartheta = 2$ and without scaling. We use $\vartheta > 0$ to denote any scaling parameter.

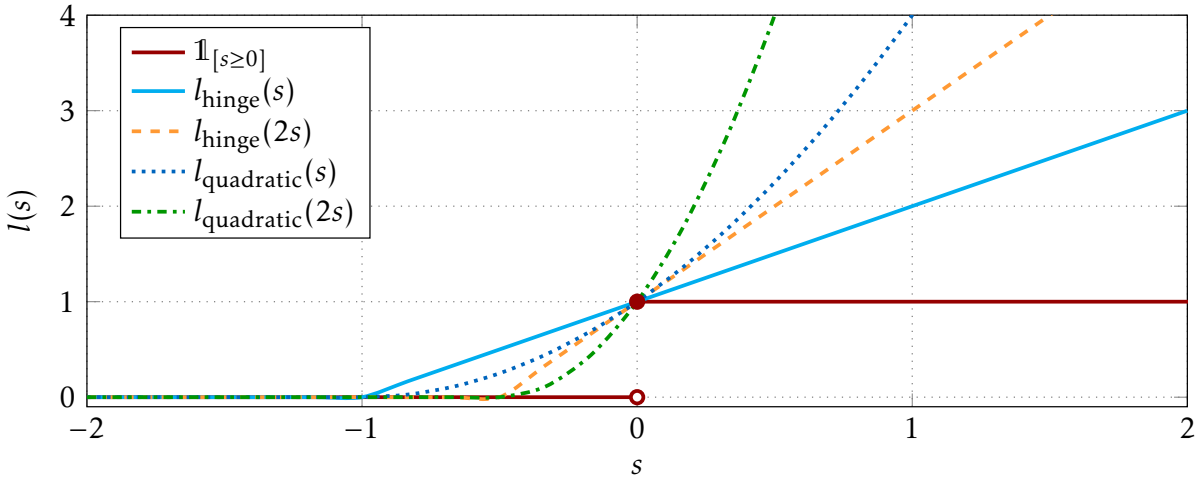


Figure 2.1: Comparison of the approximation quality of the Iverson function using different surrogate functions and scaling parameters.

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, 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 (1.4). 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

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + C_1 \cdot \overline{\text{fp}}(\boldsymbol{s}, t) + C_2 \cdot \overline{\text{fn}}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = G(\boldsymbol{s}, \boldsymbol{y}). \end{aligned} \tag{2.3}$$

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

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

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

Note 2.3

To improve the readability of the main part of the work, we present results only for hinge loss. Results for quadratic hinge loss are in appendix

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 $s^-_{[\cdot]}$, it is a function of all negative scores. Therefore, formulation (2.4) is just a special case of the general formulation (2.1) for $C_1 = 0$ and $C_2 = 1/n_+$. 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

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

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

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

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 nice 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. In statistics, the τ-quantile of all scores is defined as follows

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

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

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

This formulation already falls into our framework (2.1) for $C_1 = 1/n_-$ and $C_2 = 1/n_+$. 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

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

This formulation falls into our framework (2.3) for $C_1 = 1/n_-$ and $C_2 = 1/n_+$. 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*

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

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. Even though we start with a different optimization problem, we derived a very similar formulation to the *TopPushK* formulation (2.6) 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

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

where $\vartheta > 0$ is fixed scaling parameter. Due to the properties of the surrogate function l (see Notation2.1), the problem above has a unique solution, and we do not have to use the maximum as in the case of true threshold (2.7). 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

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

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).

2.3.1 Threshold Comparison

The previous section introduces three formulations *Grill*, *TopMeanK*, and *Pat&Mat*. While *Grill* formulations use true top τ-quantile (2.7) as a threshold, the remaining two formulations use only its approximations. The following lemma shows that the approximation used by *TopMeanK* is better than the one used by *Pat&Mat*.

Lemma 2.4: Thresholds relation [29]

Consider fixed vector of scores $\boldsymbol{s}$ and thresholds for *Grill*, *TopMeanK*, and *Pat&Mat* defined by

$$t_1(\boldsymbol{s}) = \max\left\{ t \;\middle|\; \frac{1}{n}\sum_{i\in\mathcal{I}} \mathbb{1}_{[s_i \geq t]} \geq \tau \right\}, \quad t_2(\boldsymbol{s}) = \frac{1}{n\tau}\sum_{i=1}^{n\tau} s_{[i]}, \quad t_3(\boldsymbol{s}) \ \text{ solves } \ \frac{1}{n}\sum_{i\in\mathcal{I}} l(\vartheta(s_i - t)) = \tau.$$

Then the following inequalities hold

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

The previous lemma shows that for fixed scores $\boldsymbol{s}$, the threshold (2.10) for *TopPushK* is always lower or equal to the threshold (2.12) for *Pat&Mat*. These formulations use a surrogate approximation of the false-negative rate as an objective function. Since this approximation is a non-decreasing function of t, a lower threshold t means a lower objective function value. Therefore, *TopPushK* seems to be a better formulation since it uses a better approximation of the true quantile. Besides that, it is also easier to compute the threshold for *TopMeanK* than for *Pat&Mat*. However, Chapter 3 shows that *Pat&Mat* formulation has better properties than *TopMeanK*. Moreover, the following section shows how to efficiently compute the threshold for *Pat&Mat* with hinge loss as a surrogate function.

2.3.2 Efficient Computing of the Threshold for *Pat&Mat*

In this section, we show how to efficiently find the threshold (2.12) for *Pat&Mat* when the hinge loss is used as a surrogate. Consider function

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

then finding threshold (2.12) for *Pat&Mat* 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. Moreover, $h(t) \to \infty$ 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. For better readability, the proof of the lemma is in Appendix A.

Lemma 2.5

Consider vector of scores $\boldsymbol{s}$ and its sorted version $\boldsymbol{s}_{[\cdot]}$ with non-increasing elements as defined in Notation 2.2, and threshold for *Pat&Mat* formulation

$$h(t) = \sum_{i \in \mathcal{I}} l(\vartheta(s_i - t)) - n\tau, \tag{2.14}$$

where $\vartheta > 0$ and l is the hinge loss from Notation 2.1. For all $i \in \mathcal{I}$ define $t_i = s_{[i]} + \frac{1}{\vartheta}$. Then for all $i = 2, 3, \ldots, n$ we have

$$h(t_i) = h(t_{i-1}) + (i-1)\vartheta(t_{i-1} - t_i), \tag{2.15}$$

with the initial condition $h(t_1) = -n\tau$.

The previous lemma shows how to compute the values of the function (2.14)from the sorted vector of classification scores. Therefore, to solve equation $h(t) = 0$, we can start from initial point $h(t_1) = -n\tau$ and use the recurrent relation (2.15) until we find the first threshold for which the function value of h is non-negative. Denote this threshold as t_i. Then, the threshold $\hat{t}$ can be found using linear interpolation as follows

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

where the secon equality follows from (2.15). The whole procedure of finding $\hat{t}$ is summarized in Algorithm 1.

Algorithm 1 An efficient algorithm for computing threshold (2.12) for *Pat&Mat* formulation.

Require: vector $\boldsymbol{s}$ sorted into $\boldsymbol{s}_{[\cdot]}$ and

Set $t_1 \leftarrow s_{[1]} + \frac{1}{\vartheta}$, $h(t_1) \leftarrow -n\tau$, and $i \leftarrow 2$
while $h(t_i) < 0$ **do**
    $t_i \leftarrow s_{[i]} + \frac{1}{\vartheta}$
    $h(t_i) \leftarrow h(t_{i-1}) + (i-1)\vartheta(s_{[i-1]} - s_{[i]})$
    $i \leftarrow i + 1$
end while
return $\hat{t} \leftarrow t_{i-1} + \frac{1}{(i-1)\vartheta}h(t_{i-1})$

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 to reject it. 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

$$t^{\text{NP}} = \max\left\{ t \;\middle|\; \frac{1}{n_-} \sum_{i \in \mathcal{I}_-} \mathbb{1}_{[s_i \geq t]} \geq \tau \right\}. \tag{2.16}$$

Note that we only count the false-positive samples in (2.16) instead of counting all positives in (2.7). Then, we may write the Neyman-Pearson problem as

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{n_+} \text{fn}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = \max\left\{ t \;\middle|\; \frac{1}{n_-} \sum_{i \in \mathcal{I}_-} \mathbb{1}_{[s_i \geq t]} \geq \tau \right\}. \end{aligned} \tag{2.17}$$

This problem falls within our framework for (2.1) for $C_1 = 0$ and $C_2 = 1/n_+$. Moreover, formulation (2.17) 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

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

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 (2.16) by the mean of $n_-\tau$ highest of scores corresponding to the negative samples

$$t^{\text{NP}} = \frac{1}{n_-\tau} \sum_{i=1}^{n_-\tau} s^-_{[i]}. \tag{2.19}$$

For simplicity, we again assume that $n_-\tau$ is an integer. Putting (2.19) into the constraint results in the Neyman-Pearson alternative to *TopMeanK* defined as

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

This problem already appeared in [29] under the name τ-*FPL*. Formulation (2.20) 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 (2.16) reads

$$t^{\text{NP}} \text{ solves } \frac{1}{n_-}\sum_{i\in\mathcal{I}_-} l(\vartheta(s_i - t)) = \tau. \tag{2.21}$$

Putting (2.21) into the constraint results in the Neyman-Pearson alternative to *Pat&Mat* in the following form

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n_+}\overline{\text{fn}}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t \text{ solves } \frac{1}{n_-}\sum_{i\in\mathcal{I}_-} l(\vartheta(s_i - t)) = \tau. \end{aligned} \tag{2.22}$$

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. For *Pat&Mat* we derived and efficient algorithm (Algorithm 1) for finding the threshold if the hinge loss is used as surrogate. Similar algorithm can be derived for *Pat&Mat-NP*.

2.4.1 Threshold Comparison

In Section 2.3.1, we show that the threshold for *TopMeanK* provides a better approximation of the true top τ-quantile of all scores than the threshold for *Pat&Mat*. The only difference between τ-*FPL* and *TopMeanK* is that the former computes the threshold only from negative samples while the latter uses all samples. The same holds for *Pat&Mat-NP* and *Pat&Mat*. Therefore, we can show the same for τ-*FPL* and *Pat&Mat-NP* relations as showed in Section 2.3.1 for *TopMeanK* and *Pat&Mat*. The following lemma shows that the approximation used by τ-*FPL* is better than the one used by *Pat&Mat-NP*.

Lemma 2.6: Thresholds relation [29]

Consider fixed vector of scores $\boldsymbol{s}$ and thresholds for *Grill-NP*, τ-*FPL*, and *Pat&Mat-NP* defined by

$$t_1^{\text{NP}}(\boldsymbol{s}) = \max\left\{ t \;\middle|\; \frac{1}{n_-}\sum_{i\in\mathcal{I}_-} \mathbb{1}_{[s_i \geq t]} \geq \tau \right\}, \qquad t_3^{\text{NP}}(\boldsymbol{s}) \text{ solves } \frac{1}{n_-}\sum_{i\in\mathcal{I}_-} l(\vartheta(s_i - t)) = \tau,$$

$$t_2^{\text{NP}}(\boldsymbol{s}) = \frac{1}{n\_\tau}\sum_{i=1}^{n\_\tau} s_{[i]}^{-}.$$

Then the following inequalities hold

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

Using the same arguments as at the end of Section 2.3.1, τ-*FPL* is a better formulation in the sense that for the fixed vector of scores, it provides a lower threshold and lower objective

function value. However, Chapter 3 shows that *Pat&Mat-NP* formulation has better properties than τ-*FPL*.

Finally, Lemma 2.7 shows that *Grill* provides a larger threshold under some conditions than *Grill-NP*. In the same way, *TopMeanK* under some conditions provides a larger threshold than τ-*FPL*. Since the goal of the presented formulations is to push s^+ above s^-, we may expect that the conditions in Lemma 2.7 are satisfied. The importance of these properties is discussed in Section 3.3, where we show that the formulations with larger thresholds are in some sense more unstable.

Lemma 2.7

Consider *Grill*, *Grill-NP*, *TopMeanK* and τ-*FPL* formulations and Notation 2.2. If the following inequality holds

$$s^+_{[n_+\tau]} > s^-_{[n_-\tau]},$$

then *Grill* has larger threshold than *Grill-NP*. In the same way, if the following inequality holds

$$\frac{1}{n_+\tau}\sum_{i=1}^{n_+\tau} s^+_{[i]} > \frac{1}{n_-\tau}\sum_{i=1}^{n_-\tau} s^-_{[i]}$$

then *TopMeanK* has larger threshold than τ-*FPL*.

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 designed 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.

In Section 2.3.1, we showed that the threshold for *TopMeanK* better approximates the true top τ-quantile than the threshold for *Pat&Mat*. Since both formulations use the same objective function, that is non-decreasing function of the threshold, *TopMeanK* provides a lower objective value for the fixed model. Furthermore, it is easier to compute the threshold for *TopMeanK* than for *Pat&Mat*. Therefore it seems that *TopMeanK* is a better formulation. To simplify the problem of finding the threshold for *Pat&Mat*, we derived an efficient algorithm for finding it when the hinge loss is used, see Section 2.3.2. Moreover, in the next chapter, we will show that *Pat&Mat* has better theoretical properties. A similar relation holds for the τ-*FPL* and *Pat&Mat-NP*, as discussed in Section 2.4.1.

Formulation	Label	Source	Ours	Hyper-parameters	C_1	C_2	Threshold
TopPush	(2.5)	[22]	✗	λ	0	$\frac{1}{n_+}$	$s^-_{[1]}$
TopPushK	(2.6)	[30]	✓	λ, K	0	$\frac{1}{n_+}$	$\frac{1}{K}\sum_{i=1}^{K} s^-_{[i]}$
Grill	(2.9)	[3]	✗	λ	$\frac{1}{n_-}$	$\frac{1}{n_+}$	$\max\left\{t \,\middle|\, \frac{1}{n}\sum_{i\in\mathcal{I}} \mathbb{1}_{[s_i\geq t]} \geq \tau\right\}$
TopMeanK	(2.11)	—	✗	λ	0	$\frac{1}{n_+}$	$\frac{1}{K}\sum_{i=1}^{K} s_{[i]}$
Pat&Mat	(2.13)	[30]	✓	λ, ϑ	0	$\frac{1}{n_+}$	$\frac{1}{n}\sum_{i\in\mathcal{I}} l(\vartheta(s_i - t)) = \tau$
Grill-NP	(2.18)	—	✗	λ	$\frac{1}{n_-}$	$\frac{1}{n_+}$	$\max\left\{t \,\middle|\, \frac{1}{n_-}\sum_{i\in\mathcal{I}_-} \mathbb{1}_{[s_i\geq t]} \geq \tau\right\}$
τ-*FPL*	(2.20)	[29]	✗	λ	0	$\frac{1}{n_+}$	$\frac{1}{n_-\tau}\sum_{i=1}^{n_-\tau} s^-_{[i]}$
Pat&Mat-NP	(2.22)	[30]	✓	λ, ϑ	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. Column **Hyperparameters** shows the hyperparameters available for each formulation. The last three columns show the values of parameters C_1, C_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

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + C_1 \cdot \overline{\text{fp}}(\boldsymbol{s}, t) + C_2 \cdot \overline{\text{fn}}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = \boldsymbol{w}^\top \boldsymbol{x}_i, \quad i \in \mathcal{I}, \\ & t = G(\boldsymbol{s}, \boldsymbol{y}). \end{aligned}$$

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 fixed vector of scores $\boldsymbol{s}$ with elements defined as $s_i = \boldsymbol{w}^\top \boldsymbol{x}_i$ for all $i \in \mathcal{I}$. Moreover, consider thresholds for *TopPush*, *Grill*, *TopMeanK* and *Pat&Mat* from Section 2.2 and 2.3 defined as

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

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

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

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

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.

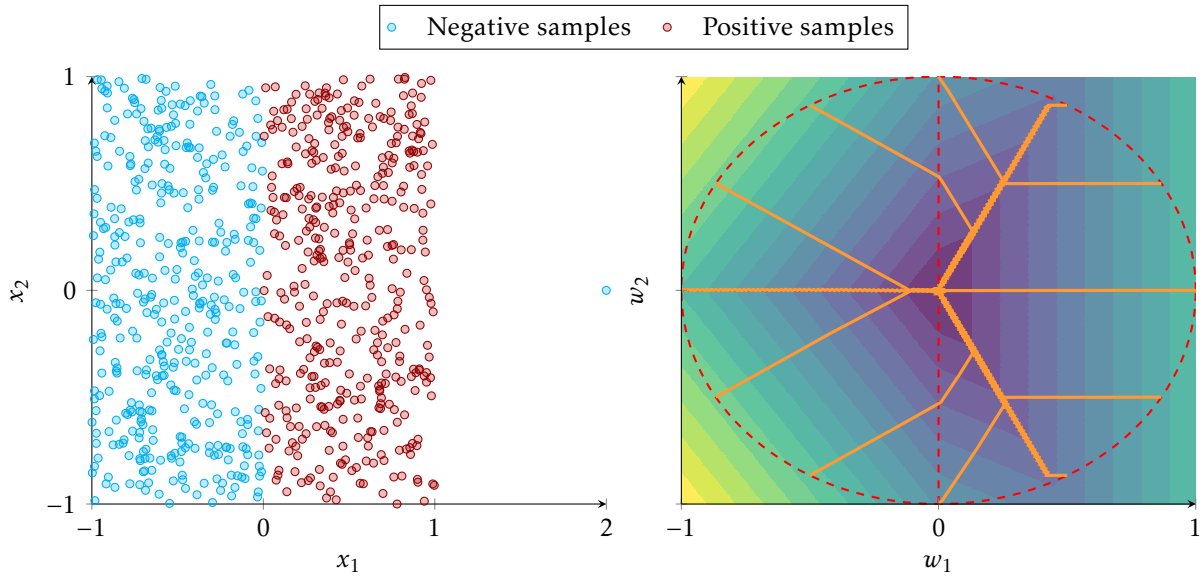


Figure 3.1: Distribution of positive (red circles) and negative samples (blue circles) for the example from Example 3.4. (**left**) Contour plot of the objective function value for *TopPush* with hinge loss as a surrogate and no regularization and its convergence (orange lines) to the zero vector from 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 B.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	$\frac{5}{2}$
TopPushK	(2.6)	0	1	$\frac{2}{K}$	$\frac{1}{2}+\frac{2}{K}$
Grill	(2.9)	0	2	$1-2\tau$	$\frac{3}{2}+2\tau(1-\tau)$
TopMeanK	(2.11)	0	1	$1-\tau$	$\frac{3}{2}-\tau$
Pat&Mat	(2.13)	$\frac{1}{\vartheta}(1-\tau)$	$1+\frac{1}{\vartheta}(1-\tau)$	$\frac{1}{\vartheta}(1-\tau)$	$\frac{1}{2}+\frac{1}{\vartheta}(1-\tau)$
Grill-NP	(2.18)	0	2	$-\tau$	$1+\frac{1}{2}\tau^2$
τ-FPL	(2.20)	0	1	$-\frac{1}{2}\tau$	$\frac{1}{2}-\frac{1}{8}\tau(4+\tau)$
Pat&Mat-NP	(2.22)	$\frac{1}{\vartheta}(1-\tau)$	$1+\frac{1}{\vartheta}(1-\tau)$	$\frac{1}{\vartheta}(1-\tau)-\frac{1}{2}$	$\frac{1}{\vartheta}(1-\tau)$

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 $\boldsymbol{w}_0 = (0,0)$. Two 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 (2.21) 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

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

Consequently, if τ-*FPL* is stable, then (3.3) is satisfied. 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 means that whenever (3.3) is satisfied, (3.4) is also satisfied. Thus, if τ-*FPL* is stable, *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.

In the same way, it can be shown that the stability of *TopMeanK* implies the stability of *Pat&Mat*.

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.

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. For simplicity, we show the proof only for *Pat&Mat*. The proof for *Pat&Mat-NP* is almost the same.

The decision in variable *Pat&Mat* formulation (2.13) is the normal vector of the separating hyperplane $\boldsymbol{w}$. Therefore, 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. Since the decision threshold t depends on $\boldsymbol{w}$, we need to use the chain rule to compute the gradient of the objective function. For each $\boldsymbol{w}$, the threshold t can be computed uniquely as discussed in Section 2.3.2. We stress this dependence by writing $t(\boldsymbol{w})$ instead of t. Note that the convexity is preserved. Then we can compute the derivative via the chain rule

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

where we assume $\frac{\lambda}{2} = \frac{1}{2}$ for simplicity. The only unknown part is the computation of $\nabla t(\boldsymbol{w})$. Theorem 3.3 shows the computation for *Pat&Mat* with efficient computation method presented in Section 2.3.2. 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.

Even though we focus only on the *Pat&Mat* formulation, the relations (3.5) can be used for almost all formulations from Table 2.1. The only two exceptions are *Grill* and *Grill-NP*, which use a slightly different objective function. However, the form on the gradient of the objective function for these two formulations is very similar. Nevertheless, the rest of the section that shows the convergence of the stochastic gradient is applicable only for *Pat&Mat*.

Consider piecewise disjoint minibatches $\mathcal{I}^1_{\text{mb}}$, $\mathcal{I}^2_{\text{mb}}$,..., $\mathcal{I}^m_{\text{mb}}$ which cycle periodically, i.e., for all k we have $\mathcal{I}^{k+m}_{\text{mb}} = \mathcal{I}^k_{\text{mb}}$. 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 Section 2.3.2 to compute the surrogate quantile t^k as the unique solution of

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

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 approximate the derivative $\nabla L(\boldsymbol{w}^k)$ from (3.5) by

$$g(\boldsymbol{w}^k) = \boldsymbol{w}^k + \frac{1}{n^k_{\text{mb},+}} \sum_{i \in \mathcal{I}^k_{\text{mb},+}} l'(t^k - s_i^k)(\nabla t^k - \boldsymbol{x}_i), \tag{3.8}$$

where ∇t^k is an approximation of $\nabla t(\boldsymbol{w}^k)$ from Theorem 3.3. To define the approximation ∇t^k, we first need to define and artificial variable

$$\boldsymbol{a}^k = \sum_{i \in \mathcal{I}^k_{\text{mb}}} l'\Big(\vartheta\Big(s_i^k - t^k\Big)\Big)\boldsymbol{x}_i. \tag{3.9}$$

Note that $\boldsymbol{a}^k$ is an approximation of the numerator of $\nabla t(\boldsymbol{w}^k)$ from Theorem 3.3, but it uses only the current minibatch. Since our minibatches cycle periodically, we sum the last m variables $\boldsymbol{a}^k$ and get the approximation of $\nabla t(\boldsymbol{w}^k)$ computed from all samples and delayed scores

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

It would be easier to consider only $\boldsymbol{a}^k$ in the numerator of (3.10). However, presented choice enables us to prove the convergence, and adds stability to the algorithm for small minibatches.

Theorem 3.9

Consider the *Pat&Mat* formulation, stepsizes $\alpha^k = \frac{1}{k+1}\alpha^0$, and piecewise disjoint minibatches $\mathcal{I}^1_{\text{mb}}$, $\mathcal{I}^2_{\text{mb}}$,..., $\mathcal{I}^m_{\text{mb}}$ which cycle periodically $\mathcal{I}^{k+m}_{\text{mb}} = \mathcal{I}^k_{\text{mb}}$. If l is the smoothened hinge function defined by

$$l(s) = \begin{cases} 0 & \text{for } s < -1 - \varepsilon, \\ \frac{1}{4\varepsilon}(1 + s + \varepsilon)^2 & \text{for } -1 - \varepsilon \le s < -1 + \varepsilon, \\ 1 + s & \text{otherwise,} \end{cases} \tag{3.11}$$

where $\varepsilon > 0$, then Algorithm 2 converges to the global minimum of (2.13).

The whole procedure of the stochastic gradient descent for *Pat&Mat* formulation 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.

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 (3.9)
    Compute ∇t^k according to (3.10)
    Compute $g(\boldsymbol{w}^k)$ according to (3.8)
    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 2.3.1. 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 Lemma 2.6, it is more stable due to the discussion after Theorem 3.8. Similarly, *Pat&Mat* has a worse approximation of the true threshold than *TopMeanK* due to Lemma 2.4, but is more stable.

Formulation	Label	Convex	Differentiable	Stable
TopPush	(2.5)	✓	✗	✗
TopPushK	(2.6)	✓	✗	✗
Grill	(2.9)	✗	✗	✓
TopMeanK	(2.11)	✓	✗	✗
Pat&Mat	(2.13)	✓	✓	✓
Grill-NP	(2.18)	✗	✗	✓
τ-FPL	(2.20)	✓	✗	✗
Pat&Mat-NP	(2.22)	✓	✓	✓

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

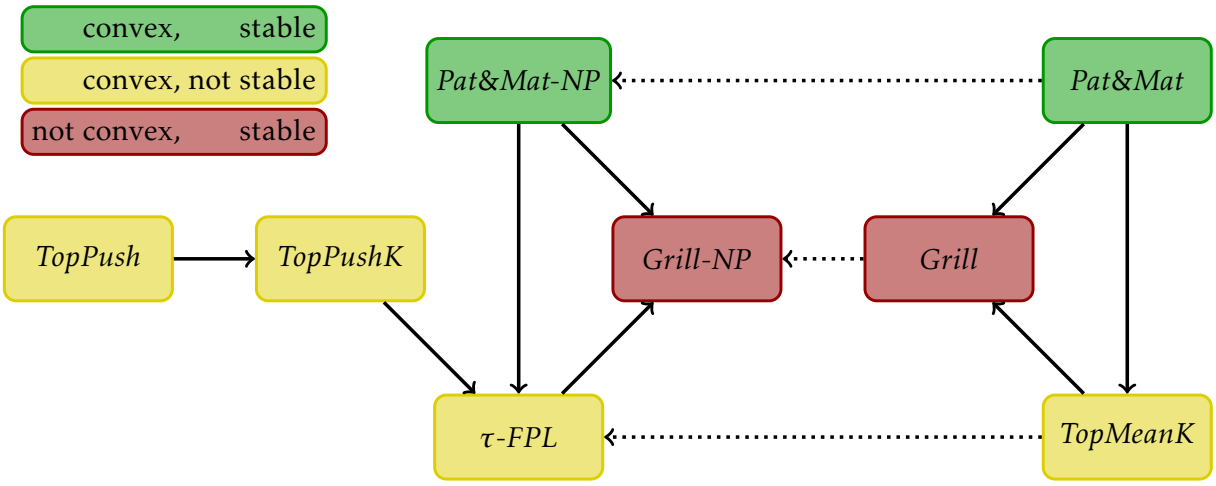


Figure 3.2: Summary of the formulations from Table 2.1. Methods in green and yellow are convex, while formulations in red are non-convex. Moreover, methods in yellow are vulnerable to having the global minimum at $\boldsymbol{w} = 0$. A full (dotted) arrow pointing from one formulation to another shows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

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + C_1 \cdot \overline{\text{fp}}(\boldsymbol{s}, t) + C_2 \cdot \overline{\text{fn}}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = \boldsymbol{w}^\top \boldsymbol{x}_i, \quad i \in \mathcal{I}, \\ & t = G(\boldsymbol{s}, \boldsymbol{y}). \end{aligned} \tag{4.1}$$

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 propose 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. We do not discuss *Grill* and *Grill-NP* formulations in the following text since both formulations are not convex, and therefore their primal and dual formulations are not equivalent. 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 the conjugate function of l. The domain of the conjugate function consists of $\boldsymbol{y} \in \mathbb{R}^n$ 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 was computed 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

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

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

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

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

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

where $C \in \mathbb{R}$. The set of indices $\tilde{\mathcal{I}}$ equals $\mathcal{I}$ for *TopMeanK* and $\mathcal{I}_-$ for other formulations. The parameter K equals 1 for *TopPush*, K for *TopPushK*, $n\tau$ for *TopMeanK*, and $n_-\tau$ for τ-*FPL*. Note that we use an alternative formulation with constant C, since it is more similar to the standard SVM, and we wanted to stress this similarity. For $C = 1/\lambda n_+$ the new formulation is identical to the original one.

The following theorem shows the dual form of formulation (4.4). The dual formulation for *TopPush* was originally derived in [22]. We only show, that our general dual formulation also covers this special case. To keep the readability as simple as possible, we postpone all proofs to Appendix C.

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

	K	$\mathbb{K}$	$\tilde{n}$	$\tilde{\boldsymbol{x}}_j$
TopPush	1	$\mathbb{K}^-$	n_-	$\boldsymbol{x}_j^-$
TopPushK	K	$\mathbb{K}^-$	n_-	$\boldsymbol{x}_j^-$
TopMeanK	$n\tau$	$\mathbb{K}^\pm$	n	$\boldsymbol{x}_j$
τ-*FPL*	$n_-\tau$	$\mathbb{K}^-$	n_-	$\boldsymbol{x}_j^-$

If $K = 1$, the upper bound in the second constraint (4.5c) vanishes due to the first constraint. Finally, the primal variables $\boldsymbol{w}$ can be computed from dual variables as follows

$$\boldsymbol{w} = \sum_{i=1}^{n_+}\alpha_i \boldsymbol{x}_i^+ - \sum_{j=1}^{\tilde{n}}\beta_j \tilde{\boldsymbol{x}}_j. \tag{4.6}$$

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

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i\in\mathcal{I}_+} l(t-\boldsymbol{w}^\top \boldsymbol{x}_i) \\ \text{subject to} \quad & t \text{ solves } \frac{1}{\tilde{n}}\sum_{i\in\tilde{\mathcal{I}}} l\Big(\vartheta(\boldsymbol{w}^\top \boldsymbol{x}_j - t)\Big) = \tau, \end{aligned} \tag{4.7}$$

where $C \in \mathbb{R}$. For *Pat&Mat* we have $\tilde{\mathcal{I}} = \mathcal{I}$ and $\tilde{n} = n$. For *Pat&Mat-NP* we have $\tilde{\mathcal{I}} = \mathcal{I}_-$ and $\tilde{n} = n_-$. Again, we use the alternative formulation with constant C. The following theorem shows the dual form of the formulation (4.7).

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

Consider Notation 4.2, surrogate function l, and formulation (4.7). Then the corresponding dual problem has the following form

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta},\delta}{\text{maximize}} \quad -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^\top \mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} - C\sum_{i=1}^{n_+} l^\star\left(\frac{\alpha_i}{C}\right) - \delta\sum_{j=1}^{\tilde{n}} l^\star\left(\frac{\beta_j}{\delta\vartheta}\right) - \delta\tilde{n}\tau \tag{4.8a}$$

$$\text{subject to} \quad \sum_{i=1}^{n_+}\alpha_i = \sum_{j=1}^{\tilde{n}}\beta_j, \tag{4.8b}$$

$$\delta \geq 0, \tag{4.8c}$$

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

	$\mathbb{K}$	$\tilde{n}$	$\tilde{\boldsymbol{x}}_j$
Pat&Mat	$\mathbb{K}^{\pm}$	n	$\boldsymbol{x}_j$
Pat&Mat-NP	$\mathbb{K}^{-}$	n_-	$\boldsymbol{x}_j^-$

Finally, the primal variables $\boldsymbol{w}$ can be computed from dual variables as follows

$$\boldsymbol{w} = \sum_{i=1}^{n_+}\alpha_i \boldsymbol{x}_i^+ - \sum_{j=1}^{\tilde{n}}\beta_j\tilde{\boldsymbol{x}}_j. \tag{4.9}$$

Note 4.5

For simplicity, the rest of the chapter covers only the *TopPushK* formulation with hinge loss. We use this formulation since it is the prototypical example for the *TopPushK* family of formulations. The results for the rest of the formulations from this family can be derived almost identically. Moreover, results for the *Pat&Mat* family of formulations can be derived similarly. Therefore, derivations for the *TopPushK* family with quadratic hinge loss and the *Pat&Mat* family with hinge and quadratic hinge loss are postponed to Appendix C.

4.2 Kernels

As we mentioned at the beginning of the chapter, our goal is to extend our framework to be usable for linearly inseparable problems. In two previous sections, we derived dual formulations for *TopPushK* and *Pat&Mat* families. 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 the *TopPushK* formulation that computes the decision threshold only from negative samples. As mentioned in Notation 4.2, *TopPushK* formulation uses kernel matrix $\mathbb{K} = \mathbb{K}^-$. The following derivation is the same for all other formulations.

To add kernels, we first realize that primal variables $\boldsymbol{w}$ can be computed from dual variables $\boldsymbol{\alpha}, \boldsymbol{\beta}$ using (4.6). Therefore, the classification score for any sample $\boldsymbol{x}$ can be calculated as follows

$$s = \boldsymbol{w}^\top \boldsymbol{x} = \sum_{i=1}^{n_+} \alpha_i \boldsymbol{x}^\top \boldsymbol{x}_i^+ - \sum_{i=1}^{n_-} \beta_i \boldsymbol{x}^\top \boldsymbol{x}_i^-. \tag{4.10}$$

Importantly, all samples $\boldsymbol{x}_i$ in the previous formula occur only in the dot product with $\boldsymbol{x}$ and not separately. This property allows us to use the standard kernel trick from SVMs [21]. The kernel trick replaces the dot product of the vectors from input space using the so-called kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. This function represents a dot product in the space of a higher dimension

$$k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^\top \phi(\boldsymbol{x}'),$$

where $\phi : \mathbb{R}^d \to \mathbb{R}^D$ is a mapping function. The idea is to transform the input vectors using ϕ into some feature space in which the classification problem is easier to solve. However, getting the explicit formula for the mapping function is usually very hard. The kernel trick allows us to avoid this explicit mapping to the feature space since we can only replace the dot product in (4.10) by the kernel function k

$$s = \sum_{i=1}^{n_+} \alpha_i k\left(\boldsymbol{x}, \boldsymbol{x}_i^+\right) - \sum_{i=1}^{n_-} \beta_i k\left(\boldsymbol{x}, \boldsymbol{x}_i^-\right). \tag{4.11}$$

The downside of this approach is, that we can not compute the primal variables using (4.6) if we do not know the mapping function ϕ. We always have to calculate the scores using the formula above, which is computationally expensive.

Now we must show how to modify the original dual problem (4.5) to incorporate kernels. Recall the form of the kernel matrix $\mathbb{K}$ for *TopPushK*

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

Since each component of the kernel matrix $\mathbb{K}$ is computed as a dot product of two training samples, we can replace $\mathbb{K}$ with a matrix in the following form

$$\mathbb{K} = \begin{pmatrix} k(\mathbb{X}^+, \mathbb{X}^+) & -k(\mathbb{X}^+, \mathbb{X}^-) \\ -k(\mathbb{X}^-, \mathbb{X}^+) & k(\mathbb{X}^-, \mathbb{X}^-) \end{pmatrix}. \tag{4.12}$$

The kernel function $k(\cdot, \cdot)$ is applied to all rows of both arguments. In other words, if we use the kernel trick, the original dual problem (4.5) remains almost the same. The only change is in the construction of the kernel matrix.

4.3 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, 4.8). 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].

Recall that we perform all derivations only for *TopPushK* with hinge loss. Classification scores can be computed directly from dual variables as shown in (4.11). Using the definition (4.12) of kernel matrix $\mathbb{K}$, we can define a vector of scores $\boldsymbol{s}$ by

$$\boldsymbol{s} = \mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}. \tag{4.13}$$

Note that dual scores are not identical to the primal ones (4.10) (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 in the input space, i.e., if $k(\boldsymbol{x},\boldsymbol{x}') = \boldsymbol{x}^\top\boldsymbol{x}'$. To simplify the indexing of the vector of scores (4.13) and kernel matrix $\mathbb{K}$, we introduce a new notation in Notation 4.6.

Notation 4.6

Consider any index l that satisfies $1 \le l \le n_+ + \tilde{n}$. Note that the length of dual variable $\boldsymbol{\alpha}$ is n_+ for both formulations (4.5) and (4.8). Therefore, we can define auxiliary index $\hat{l}$ as

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

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 $\boldsymbol{\beta}$.

4.3.1 Update Rules

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 $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$. There are only three update rules which modify two components of $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and satisfy constraints (4.5b). The first one updates two components of $\boldsymbol{\alpha}$

$$\alpha_{\hat{k}} \to \alpha_{\hat{k}} + \Delta, \qquad \alpha_{\hat{l}} \to \alpha_{\hat{l}} - \Delta, \qquad \boldsymbol{s} \to \boldsymbol{s} + (\mathbb{K}_{\bullet,k} - \mathbb{K}_{\bullet,l})\Delta, \tag{4.14a}$$

where $\mathbb{K}_{\bullet,i}$ denotes i-th column of $\mathbb{K}$ and indices $\hat{k}$, $\hat{l}$ are defined in Notation 4.6. 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 $\boldsymbol{\beta}$

$$\alpha_{\hat{k}} \to \alpha_{\hat{k}} + \Delta, \qquad \beta_{\hat{l}} \to \beta_{\hat{l}} + \Delta, \qquad \boldsymbol{s} \to \boldsymbol{s} + (\mathbb{K}_{\bullet,k} + \mathbb{K}_{\bullet,l})\Delta, \tag{4.14b}$$

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

$$\beta_{\hat{k}} \to \beta_{\hat{k}} + \Delta, \qquad \beta_{\hat{l}} \to \beta_{\hat{l}} - \Delta, \qquad \boldsymbol{s} \to \boldsymbol{s} + (\mathbb{K}_{\bullet,k} - \mathbb{K}_{\bullet,l})\Delta. \tag{4.14c}$$

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

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

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

$$\Delta^\star = \operatorname{clip}_{[\Delta_{lb},\,\Delta_{ub}]}(\gamma), \tag{4.15}$$

where $\gamma = -\frac{b}{a}$ and $\operatorname{clip}_{[a,\,b]}(x)$ amounts to clipping (projecting) x to interval $[a,b]$. Since we assume one of the update rules (4.14), the constraint (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 function from Notation 2.1 is used.

Plugging the conjugate (4.2) of the hinge loss into the dual formulation (4.5) yields

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{maximize}} \quad -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^\top \mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} + \sum_{i=1}^{n_+}\alpha_i \tag{4.16a}$$

$$\text{subject to} \quad \sum_{i=1}^{n_+}\alpha_i = \sum_{j=1}^{\tilde{n}}\beta_j, \tag{4.16b}$$

$$0 \le \alpha_i \le C, \qquad i = 1,2,\ldots,n_+, \tag{4.16c}$$

$$0 \le \beta_j \le \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i, \quad j = 1,2,\ldots,\tilde{n}. \tag{4.16d}$$

The form of $\mathbb{K}$ and $\tilde{n}$ depends on the used formulation as discussed in Theorem 4.3. Moreover, the upper bound in (4.16d) can be omitted for $K = 1$. Since we know the form of the optimal solution (4.15), we only need to show how to compute Δ_{lb}, Δ_{ub} and γ for all update rules (4.14). 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 C.2.1.

Proposition 4.7: Update rule (4.14a) for problem (4.16)

Consider problem (4.16), update rule (4.14a), indices $1 \le k \le n_+$ and $1 \le l \le n_+$ and Notation 4.6. Then the optimal solution $\Delta^\star$ is given by (4.15) 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 4.8: Update rule (4.14b) for problem (4.16)

Consider problem (4.16), update rule (4.14b), indices $1 \leq k \leq n_+$ and $n_+ + 1 \leq l \leq \tilde{n}$ and Notation 4.6. 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 (4.15) where

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

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

$$\gamma = -\frac{s_k + s_l - 1}{\mathbb{K}_{kk} + \mathbb{K}_{ll} + \mathbb{K}_{kl} + \mathbb{K}_{lk}}.$$

Proposition 4.9: Update rule (4.14c) for problem (4.16)

Consider problem (4.16), update rule (4.14c), indices $n_+ + 1 \leq k \leq \tilde{n}$ and $n_+ + 1 \leq l \leq \tilde{n}$ and Notation 4.6. Then the optimal solution $\Delta^\star$ is given by (4.15) where

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

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

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

4.3.2 Initialization

For all update rules (4.14) we assumed that the current solution $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ is feasible. So to create an iterative algorithm that solves problem (4.16) or (C.3), 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

$$\begin{aligned} \underset{\boldsymbol{\alpha}, \boldsymbol{\beta}}{\text{minimize}} \quad & \frac{1}{2}\left\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^0\right\|^2 + \frac{1}{2}\left\|\boldsymbol{\beta} - \boldsymbol{\beta}^0\right\|^2 \\ \text{subject to} \quad & \sum_{i=1}^{n_+} \alpha_i = \sum_{j=1}^{\tilde{n}} \beta_j, \\ & 0 \leq \alpha_i \leq C, \quad i = 1, 2, \ldots, n_+,, \\ & 0 \leq \beta_j \leq \frac{1}{K}\sum_{i=1}^{n_+} \alpha_i, \quad j = 1, 2, \ldots, \tilde{n}, \end{aligned} \tag{4.17}$$

where the upper bound in the second constraint depends on the used surrogate function. To solve problem (4.17), we follow the same approach as in [43]. In the following theorem, we show that problem (4.17) 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 4.10

Consider problem (4.17), 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

$$\sum_{j=1}^{K}\left(\beta^0_{[\tilde{n}-K+j]} + \max_{i=1,\dots,n_+} \alpha_i^0\right) \leq 0, \tag{4.18}$$

the optimal solution of (4.17) amounts to $\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0}$. In the opposite case, the following system of two equations

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

$$\sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu]}\left(\beta_j^0 + \lambda\right) - K\mu = 0, \tag{4.19b}$$

has a solution (λ, μ) with $\mu > 0$, and the optimal solution of (4.17) is equal to

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

Theorem 4.10 shows the optimal solution of (4.17) 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 (4.19) can be reduced to one. For any fixed μ, we denote the function on the left-hand side of (4.19b) 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 (4.19b) for a fixed μ. Denote $\boldsymbol{z}$ the sorted version of $-\boldsymbol{\beta}^0$. Then we have

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

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 (4.17) for fixed μ..

Require: vector $-\beta^0$ sorted into $\boldsymbol{z}$

```
1:  i ← 2, j ← 1, d ← 1
2:  λ ← z_1, g ← −Kμ
3:  while g < 0 do
4:      if z_i ≤ z_j + μ then
5:          g ← g + d(z_i − λ)
6:          λ ← z_i, d ← d + 1, i ← i + 1
7:      else
8:          g ← g + d(z_j + μ − λ)
9:          λ ← z_j + μ, d ← d − 1, j ← j + 1
10:     end if
11: end while
12: 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

$$h(\mu) = \sum_{i=1}^{n_+} \operatorname{clip}_{[0,\,C]}\left(\alpha_i^0 - \lambda(\mu) + \frac{1}{K}\sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,+\infty)}\left(\beta_j^0 + \lambda(\mu) - \mu\right)\right) - K\mu. \tag{4.20}$$

Then the system of equations (4.19) 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 4.11

Even though $\lambda(\mu)$ is not unique, function h from (4.20) 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 $(0, +\infty)$.

4.4 Summary

In this chapter, we derived dual formulation for *TopPushK* and *Pat&Mat* family of formulations. Moreover, we derived 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. For *TopPushK* family with hinge loss, we showed the derivation in the previous section, while the derivations for *Pat&Mat* family are in Appendix C.2.2. This section combines all these intermediate results into Algorithm 4 and discusses its computational complexity.

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 4.10 or Theorem C.17. Then, based on (4.13) we compute scores $\boldsymbol{s}$. Each `repeat` loop in step 3 updates two coordinates as shown in (4.14). 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 (4.14). 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 4.3.1 and C.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

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

```
Set α, β using Theorem 4.10
Set s based on (4.13)
repeat
    Pick random k from {1,...,n+ + ñ}
    for l ∈ {1,...,n+ + ñ} do
        Compute Δ_l
    end for
    Select the best Δ_l
    Update α, β, s according to (4.14)

until stopping criterion is satisfied
```

```
Set α, β, δ using Theorem C.17
Set s based on (4.13)
repeat
    Pick random k from {1,...,n+ + ñ}
    for l ∈ {1,...,n+ + ñ} do
        Compute Δ_l and δ_l
    end for
    Select the best Δ_l and δ_l
    Update α, β, s according to (4.14)
    set δ ← δ_l
until stopping criterion is satisfied
```

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

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

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

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

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

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

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

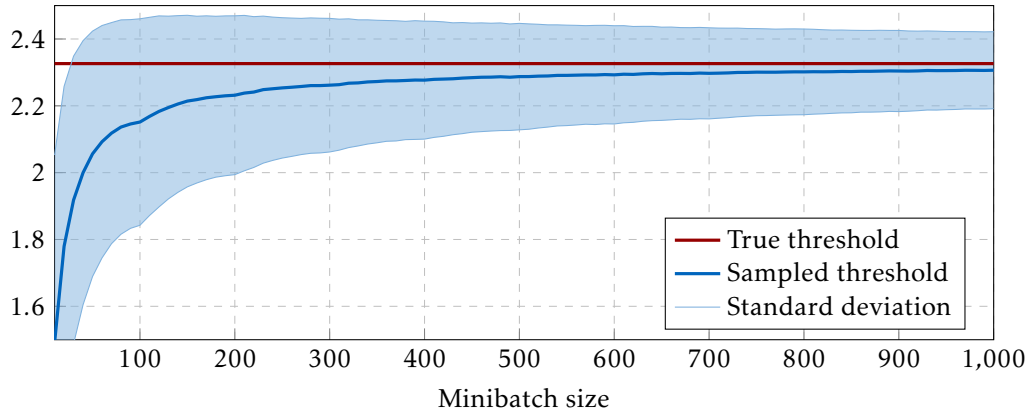


Figure 5.1: The bias between the sampled and true thresholds computed from scores following the standard normal distribution. The threshold separates the top 1% of samples with the highest scores.

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. One way how to reduce the bias is to use large minibatches, but in many cases, this is not possible. 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

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n_+}\operatorname{fn}(\boldsymbol{s}, t) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}, \\ & t = \max_{j \in \mathcal{I}_-} s_j. \end{aligned} \tag{5.5}$$

Since the threshold always equals one of the scores, 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 $\nabla t = \nabla f(\boldsymbol{x}_{j^\star}; \boldsymbol{w})$. Therefore, the gradient of the objective function of (5.5) reads

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

Using the same transition to the minibatches as in previous sections, we get the following formula for sampled gradient

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

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. 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 derived *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

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

The assumptions of Theorem 5.4 hold 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:

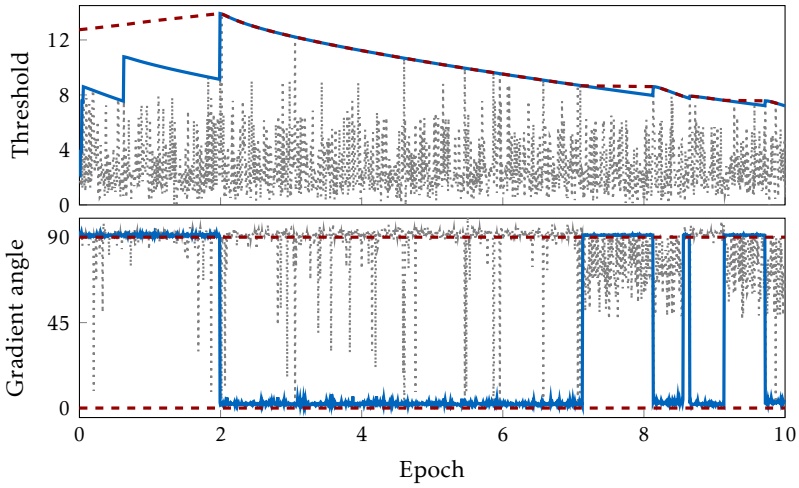


Figure 5.2: The **top** figure shows the comparison of the true threshold (red dashed line) and sampled threshold for *TopPush* (gray dotted line) and *DeepTopPush* (blue line). The **bottom** figure shows the angle between true and sampled gradients for *TopPush* (gray dotted line) and *DeepTopPush* (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.

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

In the previous sections, we derived general framework for classification at the top and showed that multiple well-known formulations fall into it. The summary of all formulations presented in this work is in Table 2.1. The goal of this chapter is to experimentally verify the properties of these formulations.

6.1 Settings

In this section we describe in detail all settings used for experiments. The section consists of five subsections. In the first one, we discuss which formulations from Table 2.1 we use for the experimental evaluation. In this subsection, we also introduce baseline formulations that are used for the comparison. In the second one, we introduce datasets used in experiments and describe their structure. A detailed description of the datasets is then provided in separate sections with results. The third and four subsections contain a detailed description of used performance metrics. The last subsection contains a description of tools used for implementation and the link to the GitHub repository with all relevant codes.

6.1.1 Formulations

Formulations from Table 2.1 can be divided into three categories:

- The first category contains *TopPush* and *TopPushK* formulations. Formulations from this category minimize the surrogate approximation of the false-negative rate. As a threshold, these formulations use the mean of a small fraction of the negative samples with the highest scores.
- The second category consists of *Grill*, *TopMeanK*, and *Pat&Mat* formulation. These three formulations again use the surrogate approximation of the false-negative rate as an objective function. The only exception is the *Grill* formulation also adds the surrogate approximation of the false-positive rate into the objective function for better stability. All three formulation uses some kind of approximation of the top τ-quantile of all scores as a threshold.
- The last category consists of *Grill-NP*, *τ-FPL*, and *Pat&Mat-NP*. These formulations use the same objectives as their corresponding formulations from the previous category and differ only in the definition of the decision threshold. All three formulation uses some kind of approximation of the top τ-quantile of negative scores as a threshold.

To simplify the setup of all experiments, we decided to focus on formulations that use only negative samples for the threshold computation, i.e. formulations from the first and third categories. The performance of these formulations can be easily compared using some basic performance metrics as we show later in Section 6.1.4.

In total, we use four different formulations from Table 2.1, namely *TopPush*, *TopPushK*, τ-*FPL*, and *Pat&Mat-NP*. We decided to omit the *Grill-NP* formulation in final experiments, because it provides very poor results in preliminary experiments. Moreover, for *TopPushK* we use two different values of $K = \{5, 10\}$ and consider the resulting formulations as separate formulations, i.e. we have *TopPushK* (5) and *TopPushK* (10). Similarly, for τ-*FPL* and *Pat&Mat* we use two different values of $\tau = \{0.01, 0.05\}$. For all formulations, we use the hinge loss defined in Notation 2.1 as a surrogate function.

We have a total of 7 different formulations, however, all these formulations fall into our general framework. To show that these formulations work properly, we have to compare them to some standard methods. In previous chapters, we showed how to solve presented formulations in their primal (Chapters 3 and 5) and dual form (Chapter 4). Whenever we use the primal form in the experiments, we use binary cross-entropy defined in the following way as a baseline formulation

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \frac{1}{n}\sum_{i\in\mathcal{I}}(-y_i \log(s_i) - (1-y_i)\log(1-s_i)) \\ \text{subject to} \quad & s_i = f(\boldsymbol{x}_i; \boldsymbol{w}), \quad i \in \mathcal{I}. \end{aligned} \tag{6.1}$$

We decided to use binary cross-entropy, since it is one of the most used objective functions for binary classification in machine learning applications. In the following text, we will denote binary cross-entropy as *BinCross* for simplicity. In experiments with dual forms, we use C-SVC variant of SVM [48, 21, 49] defined as follows

$$\begin{aligned} \underset{\boldsymbol{w}, b, \boldsymbol{\xi}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i\in\mathcal{I}}\xi_i \\ \text{subject to} \quad & y_i\left(\boldsymbol{w}^\top \phi(\boldsymbol{x}_i) + b\right) \geq 1 - \xi_i, \quad i \in \mathcal{I}, \\ & \xi_i \geq 0, \quad i \in \mathcal{I}, \end{aligned} \tag{6.2}$$

where $y_i \in \{-1, 1\}$ for all $i \in \mathcal{I}$ and $\phi(\boldsymbol{x}_i)$ maps $\boldsymbol{x}_i$ into a higher-dimensional space (see Section 4.2). The corresponding dual form is as follows

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{maximize}} \quad & -\frac{1}{2}\boldsymbol{\alpha}^\top \mathbb{K}\boldsymbol{\alpha} - \sum_{i=1}^{n}\alpha_i \\ \text{subject to} \quad & \sum_{i=1}^{n} y_i\alpha_i = 0, \\ & 0 \leq \alpha_i \leq C, \quad i = 1, 2, \ldots, n, \end{aligned} \tag{6.3}$$

where the kernel matrix $\mathbb{K}$ is defined as

$$\mathbb{K}_{i,j} = y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j),$$

for all $i, j = 1, 2, \ldots, n$. Note that the dual form of C-SVC is very similar to the dual forms of our formulations derived in Chapter 4. In the following text, we will denote C-SVC as *SVM* for simplicity.

In total, we have 9 different formulations. However, not all formulations are used for all experiments: *BinCross* formulation is also not used for experiments with dual forms, and *SVM* is used only for experiments with dual forms. The summary of all formulations used for experiments is in Table 6.1.

6.1.2 Hyperparameters

Since considered formulations differ in the number of available hyper-parameters, we decided to fix the number of hyper-parameters per formulation to six. For most of the considered

formulations, the remaining hyper-parameter is the regularization constant λ. The only exceptions are the formulations derived from *Pat&Mat-NP*, since they also have the scaling parameter ϑ. Therefore, we use the following six values of this hyper-parameter

$$\lambda \in \left\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\right\}$$

for all formulations except *Pat&Mat-NP*. Since we used a slightly different (but equivalent) primal formulation for the derivation of the dual forms, we use λ to compute hyper-parameter C used in dual forms

$$C = \frac{1}{\lambda \tilde{n}},$$

where $\tilde{n} = n$ for *SVM* and $\tilde{n} = n_+$ otherwise. For formulations derived from *Pat&Mat-NP*, we fixed λ to 10^{-3} and use the following six different values of the scaling parameter

$$\vartheta \in \left\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\right\}.$$

In all experiments, the best hyperparameter is selected based on the validation data and the appropriate performance metric.

Formulation	Fixed parameters	Hyper-parameter	Primal Form	Dual Form
BinCross	—	λ	✓	✗
SVM	—	λ	✗	✓
TopPush	—	λ	✓	✓
TopPushK (5)	$K = 5$	λ	✓	✓
TopPushK (10)	$K = 10$	λ	✓	✓
τ-*FPL* (0.01)	$\tau = 0.01$	λ	✓	✓
τ-*FPL* (0.05)	$\tau = 0.05$	λ	✓	✓
Pat&Mat-NP (0.01)	$\tau = 0.01$, $\lambda = 0.001$	ϑ	✓	✓
Pat&Mat-NP (0.05)	$\tau = 0.05$, $\lambda = 0.001$	ϑ	✓	✓

Table 6.1: Summary of all formulations used for experiments. The first column shows the aliases used for the formulations when describing the experiment results. The second column shows fixed hyperparameters used for each formulation, while the third column shows which hyper-parameters are tuned using validation data. The last two columns indicate whether the formulation is used in primal experiments, dual experiments, or both.

6.1.3 Datasets

For the numerical experiments, we consider variety of different datasets summarized in Table 6.2, that can be divided into three categories:

1. **Image Recognition:** In this category, we test formulations from Table 6.1 on datasets from the domain of image recognition. We use this domain since it is one of the most popular domains these days, and therefore, there are plenty of publicly available datasets.

2. **Steganalysis:** In this category, we use presented formulations in the domain of steganalysis. In this domain, the problem of maximizing the true-positive rate at the specific

level of the false-positive rate is well-known as we show at the beginning of Section 6.3. Therefore, formulations from Table 6.1 can be very useful in this domain.

3. **Malware Detection:** In this category, we use formulations for malware detection. Similar to steganalysis, the problem of maximizing the true-positive rate at the specific level of the false-positive rate is crucial for malware detection as discussed at the beginning of Section 6.4. Therefore, formulations from Table 6.1 can be very useful in this domain.

For each of these categories, there is a separate section with results and a more detailed description later in the text. It is worth mentioning, that not all datasets that are used in experiments are primarily designed for the classification at the top. In fact, all datasets from the first category are general image classification datasets. We use these datasets for two reasons. The first one is that they are publicly available. The second reason is, that all these datasets are well-known, and therefore it is easier to present the results on them.

Dataset	y^+	d	Train		Validation		Test	
			n	$\frac{n_+}{n}$	n	$\frac{n_+}{n}$	n	$\frac{n_+}{n}$
MNIST	1	$28 \times 28 \times 1$	45 000	11.3%	15 000	11.2%	10 000	11.4%
FashionMNIST	1	$28 \times 28 \times 1$	45 000	10.0%	15 000	9.9%	10 000	10.0%
CIFAR10	1	$32 \times 32 \times 3$	37 500	10.0%	12 500	9.9%	10 000	10.0%
CIFAR20	1	$32 \times 32 \times 3$	37 500	5.0%	12 500	5.1%	10 000	5.0%
CIFAR100	1	$32 \times 32 \times 3$	37 500	1.0%	12 500	1.0%	10 000	1.0%
SVHN2	1	$32 \times 32 \times 3$	54 944	18.9%	18 313	18.9%	26 032	19.6%
SVHN2-Extra	1	$32 \times 32 \times 3$	453 291	17.3%	151 097	17.1%	26 032	19.6%
Nsf5	—	22510×1	186 583	9.1%	62 194	9.1%	248 776	9.1%
JMiPOD	—	$256 \times 256 \times 3$	186 515	9.1%	62 172	9.1%	248 686	9.1%
Malware	—	variable	6 580 166	87.22%	—	—	800 346	91.8%

Table 6.2: Structure of the used datasets: The training, validation and testing sets show the positive label y^+, the number of features d, samples n and the fraction of positive samples $\frac{n_+}{n}$. Datasets depicted in red are not publicly available.

6.1.4 Performance Criteria

In the previous subsections, we described all formulations and datasets used for the experiments. In this section, we describe which performance criteria are used for evaluation and how these criteria are related to the tested formulations.

As we discussed at the beginning of Section 6.1, we decided to test only formulations that minimize the false-negative rate (or a combination of false-negative and false-positive rate) and use only negative samples for the threshold computation. This choice allows us to use simple metrics to compare used formulations. The first metric that we use in experiments is TPR@K defined as follows

$$\text{TPR@}K = \frac{1}{n_+} \sum_{i \in \mathcal{I}_+} \mathbb{1}_{[s_i \geq t]} \quad \text{where} \quad t = \sum_{j=1}^{K} s^-_{[j]}.$$

This metric computes the true-positive rate at threshold t which is the mean of K-largest negative scores. For $K = 1$ the threshold corresponds to the threshold used by *TopPush* formulation, and otherwise threshold t corresponds to the threshold used by *TopPushK*. Moreover, since minimizing the false-negative rate is equivalent to maximizing the true-positive rate, both *TopPush* and *TopPushK* should optimize the TPR@K metric. In the upcoming experiments, we use this metric with three different values of $K \in \{1, 5, 10\}$.

The second metric is defined in a similar way

$$\text{TPR@}\tau = \frac{1}{n_+} \sum_{i \in \mathcal{I}_+} \mathbb{1}_{[s_i \geq t]} \quad \text{where} \quad t = \max \left\{ t \;\middle|\; \frac{1}{n_-} \sum_{i \in \mathcal{I}_-} \mathbb{1}_{[s_i \geq t]} \geq \tau \right\}.$$

This metric computes the true-positive rate at a specific top τ-quantile of negative scores. This metric is ideal for testing the performance of τ-*FPL* and *Pat&Mat-NP*, since both formulations maximize true-positive rate and use some kind of approximation of the true top τ-quantile of negative scores as a threshold. In experiments, we use this metric with two different values of $\tau \in \{0.01, 0.05\}$.

The two previous metric are specific for the formulations from our framework. However, we should also test if the baseline formulations work properly. Since the baseline methods are designed to optimize overall perfomance, we use area under ROC curve to measure the overall performance. The summary of all used metrics is in Table 6.3.

Formulation	AUROC	TPR@K			TPR@τ	
		1	5	10	0.01	0.05
BinCross	✓	✗	✗	✗	✗	✗
SVM	✓	✗	✗	✗	✗	✗
TopPush	✗	✓	✗	✗	✗	✗
TopPushK (5)	✗	✗	✓	✗	✗	✗
TopPushK (10)	✗	✗	✗	✓	✗	✗
τ-*FPL* (0.01) and *Pat&Mat-NP* (0.01)	✗	✗	✗	✗	✓	✗
τ-*FPL* (0.05) and *Pat&Mat-NP* (0.05)	✗	✗	✗	✗	✗	✓

Table 6.3: The summary of all used perofmance metrics used for evaluation. In total we use six different metrics and eleven different formulations. For each formulation ✓denotes the metric in which the formulation should be the best.

6.1.5 Critical Difference Diagrams

All metrics from Section 6.1.4 can be used to compare different formulations on a single dataset. However, these metrics are not suitable for comparison of multiple formulations on multiple datasets. To address this issue, we follow the suggestion from [50] and use the Friedman test [51]. Consider that we have D, datasets and k formulations. Than for each dataset i, each formulation j is ranked by rank r_j^i according to some performance criterium (any performance metric from previous section), i.e. the formulation that provides the best result has rank 1, the second best has rank 2, etc.. If two formulations provides the same results, the average ranks are assigned. The average rank over all dataset for formulation j is then computed

as $R_j = \frac{1}{D}\sum_{i=1}^{D} r_j^i$. The Friedman test compares the average ranks of formulations under the null hypothesis, which states that all formulations are equivalent and therefore their average ranks should be equal. If the null hypothesis is rejected, we proceed with the post-hoc Nemenyi test [52] that compares all formulations to each other. The performance of two formulations is significantly different if the corresponding average ranks differ by at least the critical difference

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6D}},$$

where critical values q_α are based on the Studentized range statistic divided by $\sqrt{2}$, see Table 5(a) in [50]. The results of this post hoc test can be easily visualized using critical difference diagrams proposed in [50]. The x-axis of such diagram shows the average rank over all datasets for each formulation. Formulations that are not significatnly different according to the Nemenyi test are connected using green horizontal line. As an example of such diagrams see Figure 6.1.

6.1.6 Implementation

For the implementation of all formulations we use the Julia programming language [53]. The dual formulations are implemented from scratch in pure Julia, while the primal formulations are implemented using Flux.jl [54, 55] library. This library provides all the necessary tools for building neural networks. Moreover, the library allows the implementation of a custom gradient for any function, which allows us to implement all formulations from Table 2.1. For experiments with SVM, we use the Julia wrapper for the LIBSVM library [49]. All codes used for experiments as well as all configurations of all experiments are publicly available on GitHub:

`https://github.com/VaclavMacha/ClassificationAtTopExperiments.jl`

6.2 Image Recognition

In this section, we present experiments with six well-known image recognition datasets. All these datasets are publicly available. MNIST [56] and FashionMNIST [57] are grayscale datasets of digits and fashion items, respectively. CIFAR100 [58] is a dataset of coloured images of different items grouped into 100 classes. CIFAR10 and CIFAR20 merge these classes into 10 and 20 superclasses, respectively. Finally, SVHN2 [59] contains coloured images of house numbers. All these datasets are originally divided only into training and test sets. Therefore, we select 25% samples from the training set to obtain the validation set. For a more detailed description of the structure of datasets, see Table 6.2.

None of these datasets is primarily designed for the problem of classification at the top. We presented these datasets to show, that introduced formulations can be useful even for general datasets and may lead to improve performance on specific metrics. However, there are some drawbacks to using these datasets. However, there are also some drawbacks. For example, many state-of-the-art neural network architectures achieve almost the perfect classification on some of these datasets and therefore there is no room for improvement. For this reason, we use much simpler architectures in the experiments to show the behavior of formulations presented in this work.

6.2.1 Primal Formulation: Linear Model

In this subsection, we present results for a primal form of formulations from Table 6.1 with a linear model, i.e. resutls presented here are related to the Chapter 3. For training, we use stochastic gradient descent with balanced minibatches of size 512. As an optimizer, we use the

ADAM [60] with default settings and initial step length $\alpha = 0.01$ which we discount every 5 epochs by the factor of 0.8 using an exponential decay scheme. We also use a fixed number of epochs to 100 for all formulations, and we repeat each experiment ten times with ten different random seeds.

For comparison of all formulations, we use two different approaches. In the first one, we compare specific performance metrics on specific datasets. Since we have six hyperparameters for each formulation, we always select the best result for each formulation on the validation set based on the criterion for which the specific formulation is optimized. Then for each formulation, we select the median of the best results from ten independent runs. The results are presented in Table 6.4.

The second approach uses critical difference diagrams introduced in Section 6.1.5. One of the basic assumption of the critical difference diagrams to work properly is the large number of datasets. Since we perform all experiments for each formulation and each dataset ten times with different randomseed for train/valid/test split, we decided to consider each of this run as a separate dataset. It is important, that we use this settings only for the critical difference diagrams. The results are presented in Figure 6.1.

From Table 6.4 and Figure 6.1, we make several observations:

- Most of the formulations perform well on the criteria for which they are optimized.
- *TopPushK* (5) and *TopPushK* (10) provides a slight improvement over *TopPush* in most of the experiments. However, that improvements are usually not statistically significant as can be seen in Figure 6.1, where all three formulations are connected by green line for almost all metrics. The only exception is TPR@$\tau = 0.01$ metric for which *TopPush* is significatnly worse than the remaining formulations.
- *BinCross* formulation works the best for AUROC metric for which is significatnly better than other formulations, see Figure 6.1. The same can be also seen from Table 6.4, where *BinCross* provides consistenly the best results for AUROC for all datasets.
- *BinCross* formulation is not suitable for any other metric than AUROC. It is especially evident for metrics tat operates at the absolute top such as TPR@$K = 1$, TPR@$K = 5$, or TPR@$K = 10$.
- *Pat&Mat-NP* formulations provides very good results for all metrics. Moreover, *Pat&Mat-NP* (0.01) is the best formulation for TPR@$\tau = 0.01$, and *Pat&Mat-NP* (0.05) is the best formulation for TPR@$\tau = 0.05$. It means that both methods are the best for the criterion for which they are optimised. This behaviour can be seen also in Table 6.4, where *Pat&Mat-NP* (0.05) is the best formulation for TPR@$\tau = 0.01$ almost for all datasets.
- τ*-FPL* (0.05) works very good for both TPR@$\tau = 0.01$ and TPR@$\tau = 0.05$. In fact, for both metrics the formulation achieves almost as good results as *Pat&Mat-NP* formulations.
- All formulations provide very poor results for CIFAR datastets and TPR@K metrics. This is caused by the use of simple linear model. It is even more evident for SVHN2 datasets, fow which the best AUROC is only 57.82% for SVHN2 and 56.10% for SVHN2Extra. However, the obtained results are still relevant, since we compare the relative performance of the formulations to each other and not the absolute performance.

TPR@$K = 10$

Formulation	MNIST	FashionMNIST	CIFAR10	CIFAR20	CIFAR100	SVHN2	SVHN2Extra
BinCross	68.54	85.65	1.25	1.40	2.00	0.04	0.02
TopPush	89.38	93.70	2.25	0.40	4.00	0.02	0.02
TopPushK (5)	89.60	93.30	3.30	1.10	3.00	0.04	0.06
TopPushK (10)	89.64	92.75	3.90	0.70	4.50	0.04	0.09
τ-*FPL* (0.01)	83.35	92.40	2.90	0.80	2.50	0.03	0.02
τ-*FPL* (0.05)	40.26	79.65	3.60	1.00	5.50	0.04	0.09
Pat&Mat-NP (0.01)	87.88	92.75	5.00	1.10	4.00	0.12	0.09
Pat&Mat-NP (0.05)	51.72	81.60	3.80	1.20	3.00	0.14	0.06

TPR@$\tau = 0.05$

Formulation	MNIST	FashionMNIST	CIFAR10	CIFAR20	CIFAR100	SVHN2	SVHN2Extra
BinCross	99.65	99.30	43.40	32.90	54.50	5.53	5.91
TopPush	99.12	98.30	31.10	16.30	43.50	5.22	6.40
TopPushK (5)	99.21	98.30	35.50	20.20	42.50	6.78	7.42
TopPushK (10)	99.30	98.30	37.90	20.80	46.50	6.15	8.07
τ-*FPL* (0.01)	99.47	98.75	35.85	24.00	44.00	6.90	7.76
τ-*FPL* (0.05)	99.56	99.20	39.50	25.70	50.50	8.16	9.69
Pat&Mat-NP (0.01)	99.52	98.85	45.35	33.70	56.00	9.28	12.47
Pat&Mat-NP (0.05)	99.65	99.40	46.80	34.80	58.50	9.34	12.25

AUROC

Formulation	MNIST	FashionMNIST	CIFAR10	CIFAR20	CIFAR100	SVHN2	SVHN2Extra
BinCross	99.86	99.83	84.00	76.26	88.48	57.82	56.10
TopPush	99.78	99.42	73.84	65.76	82.10	51.08	51.30
TopPushK (5)	99.8	99.42	76.67	65.70	81.52	50.98	50.69
TopPushK (10)	99.82	99.48	77.74	66.87	81.91	51.90	50.58
τ-*FPL* (0.01)	99.84	99.72	77.34	69.24	82.96	51.04	50.62
τ-*FPL* (0.05)	99.81	99.80	79.41	70.86	84.56	51.78	50.76
Pat&Mat-NP (0.01)	99.85	99.68	82.34	74.56	86.13	56.38	51.93
Pat&Mat-NP (0.05)	99.84	99.81	83.35	75.44	87.22	56.40	52.50

Table 6.4: **Primal formulation with linear model:** Each table corresponds to one performance metric and all presented results are medians of ten independent runs for each pair of datasets and formulation. The best result for each dataset is highlighted in green, while the worst result is highlighted in red.

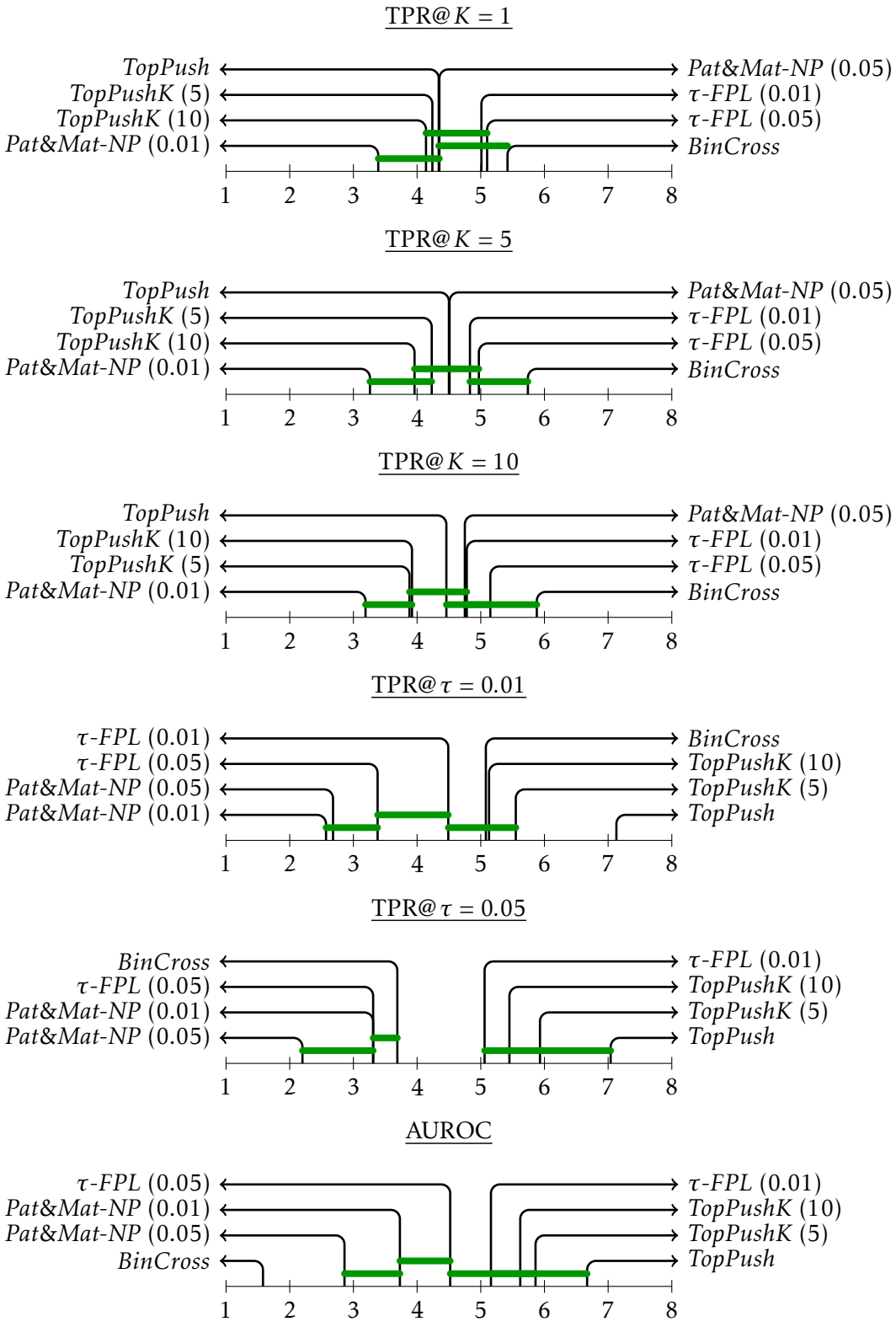


Figure 6.1: **Primal formulation with linear model:** Critical difference (CD) diagrams (level of importance 0.05) of the Nemenyi post hoc test for the Friedman test. Each diagram shows the mean rank of each method, with rank 1 being the best. Green wide horizontal lines group together methods with the mean ranks that are not significantly different. The critical difference diagrams were computed for mean rank averages over all datasets.

6.2.2 Dual Formulation: Linear Model

In this subsection, we present results for a dual form of formulations from Table 6.1 with a gaussian kernel model, i.e. resutls presented here are related to the Chapter 4. For training, we use coordinate descend algorithm introduced in Section 4.3 and we use the number of steps of this algorithm that corresponds to 20 epochs. For all experiments, we use precomputed kernel matrix with a gaussian kernel function defined as

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left\{-\frac{\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2}{d}\right\},$$

where d is the dimension of the primal problem. We used this setting since it is the default setting for radial basis kernel type in LIBSVM [49]. For computational reasons, we use only this one choice of the kernel function. Therefore, the presented results are just a comparison of formulations for this specific kernel function and not the best possible results that can be achieved.

In Figure 6.2, we investigate the convergence of coordinate descend algorithm introduced in Section 4.3 for three formulations, namely *TopPush*, *TopPushK* and *Pat&Mat-NP*. In each column, we show the convergence of primal and dual objective function for one formulation. To solve the primal problem, we used full gradient descent. Since we use full gradient to compute the optimal solution of primal form, we use very small Ionosphere dataset. This dataset is used only for this experiment. We can see that *TopPush* and *TopPushK* converge to the same objective for primal and dual form. This means that the problem was solved to optimality. However, there is a little gap between optimal solution of primal and dual form for *Pat&Mat-NP*.

For comparison of all formulations, we use the same two approaches as in Section 6.2.1. From Table 6.5 and Figure 6.3, we make several observations:

- We observed, that some formulations had problems with convergence an in some cases even diverge for some datasets. That can be caues by the inproper choice of the kernel function. As a results CD diagrams does not provide reliable results. If the formulation diverge in a few experiments, it immediately obtaind very high ranks for these experiments that totaly skwev the final diagram. It is especially evident for *Pat&Mat-NP* and *SVM* formulations.

- Figure 6.3 shows, that *Pat&Mat-NP* formulations provides the worst results for all metrics. This can be caues by the bad convergence of the coordinate descent algorithm as showed in Figure 6.2. However, it is important to say, that Figure 6.3 shows only relative results. From Table 6.5 is clear, that even thoug that *Pat&Mat-NP* provides usually worse results than other formulations, the results are in many cases only slightly worse.

- Similarly to *Pat&Mat-NP*, the *SVM* formulation does not perform well for most of the metrics, but as shown in Table 6.5, the results are usually only slihtly worse than the results of other formulations.

- τ-*FPL* formulations work very well for TPR@$\tau = 0.01$, TPR@$\tau = 0.05$ and AUROC metric.

- *TopPush*, *TopPushK* (5) and *TopPushK* (10) provides very good results for TPR@$K = 1$, TPR@$K = 5$ and TPR@$K = 10$.

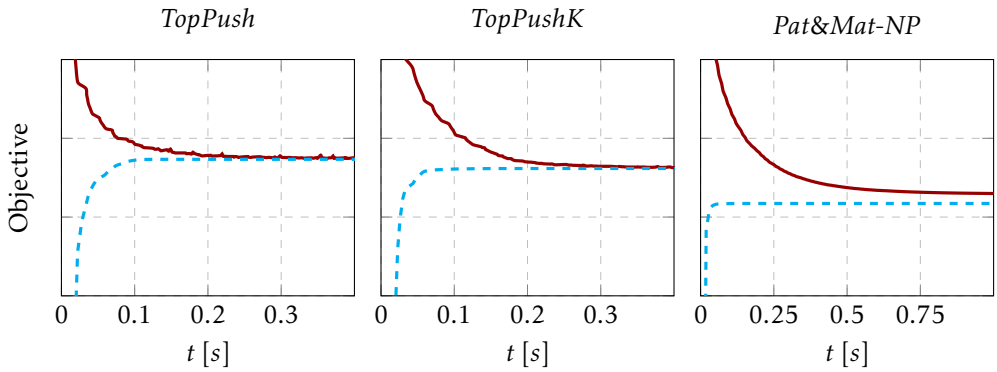


Figure 6.2: Convergence of the objectives for the primal (red line) and dual (blue dashed line) forms for with linear kernel.

6.2.3 Primal Formulation: Non-Linear Model

In this subsection, we present results for a primal form of formulations from Table 6.1 with a non-linear model, i.e. resutls presented here are related to the Chapter 5. For training, we use the same setting as in Section 6.2.1. For MNIST and FashionMNIST datasets, we use a neural network that consists of two convolution layers always followed by a max pooling layer, and one fully connected layer of the proper size for binary classification. The rest of the datasets use a similar architecture, but with three convolutional layers instead of just two. We purposely do not use state-of-the-art architectures, since they often lead to a perfect separation of used datasets. Our goal is to show, that formulations from Table 6.1 can improve specific metrics such as TPR@$\tau = 0.05$ (when compared to *BinCross*) even with these subpar architectures.

For comparison of all formulations, we use the same two approaches as in Section 6.2.1. From Table 6.6 and Figure 6.4, we make several observations:

- Most of the formulations perform well on the criteria for which they are optimized.
- Most of te formulations provides almost perfect separation on MNIST and FashionMNIST datasets.
- *DeepTopPush* does not provide good results. In fact, the formulation is the worst in five out of six metrics in Figure 6.4. However, it can be caused by the use of a relatively large minibatches with respect to the size ot the datasets. The true power of the formulation is shows in Section 6.3 and 6.4.
- *BinCross* formulation performs consistentnly very well for all metrics.
- *TopPushK* (10) fails for many times. This is especially evident from Table 6.6. We can see, that the formulation achieves 100% for TPR@$K = 10$ metric, which seems as a very good result. However, if take a look at the AUROC we can see that the formulation achieves 0%. The reason for that is simple, *TopPushK* (10) assigns the same score to all samples and therefore achieves 100% true-positive rate, but also 100% false-positive rate. This shows the disadvantage of TPR@K metrics.
- *Pat&Mat-NP* formulations provides very good results for all metrics. Moreover, *Pat&Mat-NP* (0.01) is the best formulation for TPR@$\tau = 0.01$, and *Pat&Mat-NP* (0.05) is the best formulation for TPR@$\tau = 0.05$. It means that both methods are the best for the criterion for which they are optimised. This behaviour can be seen also in Table 6.4, where *Pat&Mat-NP* (0.05) is the best formulation for TPR@$\tau = 0.01$ for all datasets.
- τ-*FPL* formulations does not work well.

TPR@$K = 10$

Formulation	MNIST	FashionMNIST	CIFAR10	CIFAR20	CIFAR100	SVHN2
SVM	97.89	95.40	9.10	4.90	11.50	4.52
TopPush	97.62	94.80	10.45	6.10	11.00	5.23
TopPushK (5)	97.97	94.90	10.05	6.00	11.0	5.07
TopPushK (10)	97.97	94.90	9.85	6.10	11.00	5.18
τ-*FPL* (0.01)	98.02	95.05	10.70	5.90	10.5	5.25
τ-*FPL* (0.05)	92.56	92.20	10.15	5.10	10.0	5.24
Pat&Mat-NP (0.01)	88.37	92.50	7.45	1.40	5.00	4.02
Pat&Mat-NP (0.05)	52.60	92.50	7.45	1.30	5.00	4.05

TPR@$\tau = 0.05$

Formulation	MNIST	FashionMNIST	CIFAR10	CIFAR20	CIFAR100	SVHN2
SVM	99.74	98.90	60.00	44.80	59.00	59.72
TopPush	99.74	98.80	57.10	37.70	59.50	72.54
TopPushK (5)	99.82	98.90	56.25	38.80	57.50	71.40
TopPushK (10)	99.82	98.90	56.90	38.70	58.00	71.61
τ-*FPL* (0.01)	99.82	98.90	58.10	39.10	59.00	73.52
τ-*FPL* (0.05)	99.74	99.10	60.80	44.40	61.00	74.26
Pat&Mat-NP (0.01)	99.30	98.10	54.70	44.60	62.50	63.47
Pat&Mat-NP (0.05)	99.38	98.10	54.70	44.50	63.50	63.48

AUROC

Formulation	MNIST	FashionMNIST	CIFAR10	CIFAR20	CIFAR100	SVHN2
SVM	99.94	99.66	90.02	79.75	87.80	90.14
TopPush	99.94	99.56	89.35	79.06	87.03	92.77
TopPushK (5)	99.95	99.64	89.05	79.13	87.21	92.60
TopPushK (10)	99.95	99.67	89.16	79.27	87.78	92.67
τ-*FPL* (0.01)	99.97	99.68	89.83	79.07	87.64	92.98
τ-*FPL* (0.05)	99.93	99.80	90.34	80.17	88.56	93.16
Pat&Mat-NP (0.01)	99.78	99.40	87.62	78.82	89.78	90.80
Pat&Mat-NP (0.05)	99.78	99.40	87.61	78.76	89.52	90.82

Table 6.5: **Dual formulations with gaussian kernel:** Each table corresponds to one performance metric and all presented results are medians of ten independent runs for each pair of datasets and formulation. The best result for each dataset is highlighted in green, while the worst result is highlighted in red.

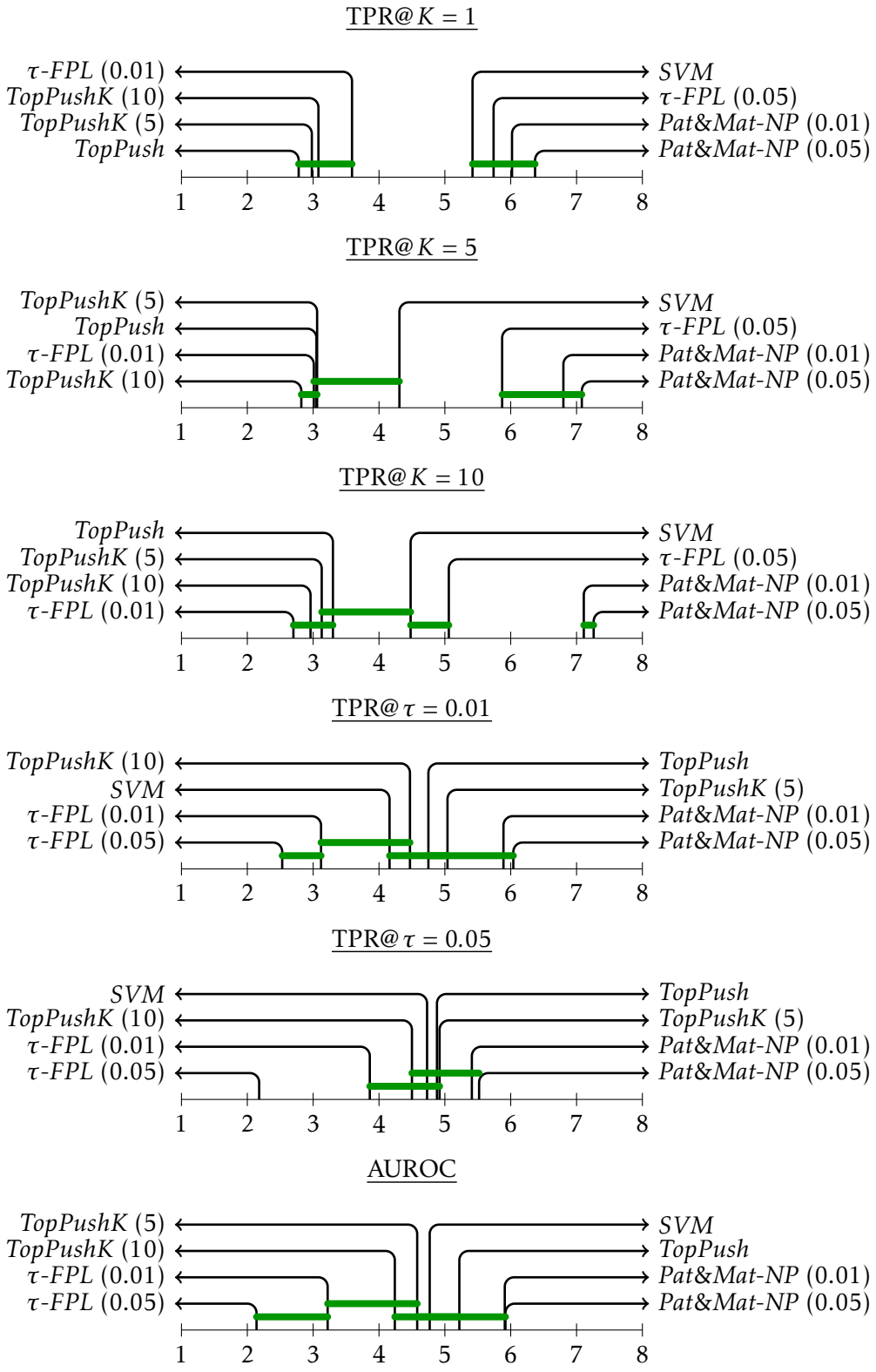


Figure 6.3: **Dual formulations with gaussian kernel:** Critical difference (CD) diagrams (level of importance 0.05) of the Nemenyi post hoc test for the Friedman test. Each diagram shows the mean rank of each method, with rank 1 being the best. Green wide horizontal lines group together methods with the mean ranks that are not significantly different. The critical difference diagrams were computed for mean rank averages over all datasets.

TPR@$K = 10$

Formulation	MNIST	FashionMNIST	CIFAR10	CIFAR20	CIFAR100	SVHN2	SVHN2Extra
BinCross	99.26	98.10	11.40	3.50	5.00	11.34	15.95
DeepTopPush	98.42	97.60	0.20	0.20	0.00	0.17	0.00
TopPushK (5)	98.54	97.50	2.00	0.00	8.50	10.58	16.12
TopPushK (10)	98.24	96.90	12.55	100.00	100.00	100.00	100.00
τ-*FPL* (0.01)	98.72	97.50	1.00	0.20	9.00	9.52	0.00
τ-*FPL* (0.05)	96.78	96.50	14.80	0.30	8.50	13.05	12.48
Pat&Mat-NP (0.01)	98.54	97.45	32.45	4.80	20.00	13.92	19.33
Pat&Mat-NP (0.05)	82.86	94.30	26.55	5.90	11.50	11.36	15.98

TPR@$\tau = 0.05$

Formulation	MNIST	FashionMNIST	CIFAR10	CIFAR20	CIFAR100	SVHN2	SVHN2Extra
BinCross	100.00	99.90	83.35	48.00	82.00	94.66	97.71
DeepTopPush	99.82	99.70	5.85	8.90	9.00	40.14	0.00
TopPushK (5)	100.00	99.85	34.30	7.70	53.50	86.54	93.96
TopPushK (10)	100.00	99.90	27.95	0.00	0.00	0.00	0.00
τ-*FPL* (0.01)	100.00	99.90	24.40	11.50	65.50	87.60	0.00
τ-*FPL* (0.05)	100.00	99.90	82.75	18.00	66.50	94.51	97.52
Pat&Mat-NP (0.01)	100.00	99.90	91.55	52.20	79.00	95.49	98.43
Pat&Mat-NP (0.05)	100.00	99.90	91.75	57.70	85.00	95.53	98.50

AUROC

Formulation	MNIST	FashionMNIST	CIFAR10	CIFAR20	CIFAR100	SVHN2	SVHN2Extra
BinCross	100.00	99.98	96.85	84.67	95.94	98.54	99.20
DeepTopPush	99.98	99.95	49.51	59.40	56.68	83.12	1.61
TopPushK (5)	100.0	99.97	77.10	55.50	84.24	96.57	98.32
TopPushK (10)	100.00	99.98	74.26	0.00	0.00	0.00	0.00
τ-*FPL* (0.01)	100.0	99.98	70.96	60.03	90.16	96.68	25.84
τ-*FPL* (0.05)	99.99	99.97	95.86	68.18	90.76	98.50	99.12
Pat&Mat-NP (0.01)	99.99	99.98	97.90	84.38	93.84	98.74	99.38
Pat&Mat-NP (0.05)	99.96	99.96	98.24	88.39	96.56	98.76	99.32

Table 6.6: **Primal formulations with non-linear model:** Each table corresponds to one performance metric and all presented results are medians of ten independent runs for each pair of datasets and formulation. The best result for each dataset is highlighted in green, while the worst result is highlighted in red.

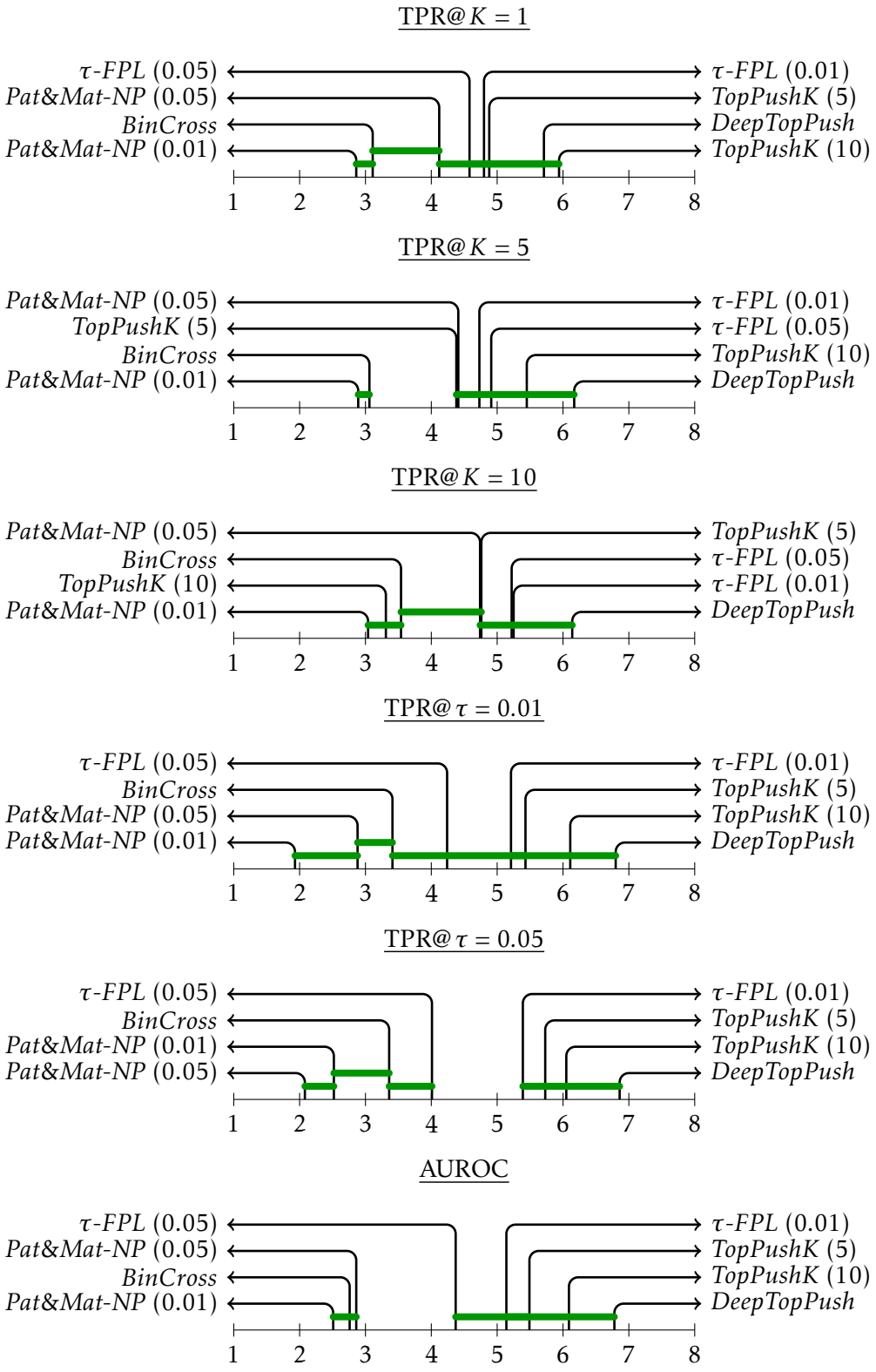


Figure 6.4: **Primal formulations with non-linear model:** Critical difference (CD) diagrams (level of importance 0.05) of the Nemenyi post hoc test for the Friedman test. Each diagram shows the mean rank of each method, with rank 1 being the best. Green wide horizontal lines group together methods with the mean ranks that are not significantly different. The critical difference diagrams were computed for mean rank averages over all datasets.

6.3 Steganalysis

In the previous section, we presented results on standard image recognition datasets. Even though the results are quite good even on these datasets, they did not fully show the importance of the problem of classification at the top. To show the importance of this problem properly, we need to find the field in which the maximizing true-positive rate at the low false-positive rate is an important task. Such a field can be for example steganography and steganalysis. The standard way how to share secret information these days is the use of encryption. However, in such a case, the presence of a secret message (even though encrypted) is obvious. Steganography aims to hide the fact that communication taking place, by hiding the secret message within an ordinary file (usually called cover file) in order to avoid detection. The secret message is then extracted at its destination. The secret data can be hidden in almost any type of digital content, however the most popular are images. There are two reasons for this. The first of them is the ubiquity of images on the Internet and therefore the ease to use them as cover files for secret messages. The second reason is their large potential payload, i.e. it is possible to hide a lot of information in the images with high resolution. With an appropriate cover image and steganography tools, it is possible to create an stego-image (image with a hidden message) that can not be recognized from the cover image by human perception. However, each tool leaves a fingerprint or signature in the image, that can be used to detect stego images. The field that tries to detect stego images and possibly decrypt messages from them is called steganalysis. In steganalysis, the goal is to achieve the best true-positive rate with the lowest possible false-positive rate. Therefore steganalysis is the branch suitable for the classification at the top, since many of the formulations derived in this work focus precisely on this task. [61, 62]

For the eperiments, we have a large dataset of cover-images that consists of approximately 450 000 images from Flickr. All these images are in the JPEG format with quality factor 80. Since the dataset does not contain any stego images, we use two different ways to generate them. For the purpose of this work, we named them **Nsf5** and **JMiPOD**.

6.3.1 Nsf5

In this case, we generate stego images using simulated F5 with matrix embedding turned off and we use payload 0.2. Since we are interested in low false-positive rates, we need a lot of negative (cover) samples to estimate it. This is evident in (2.22) where the threshold t is a surrogate approximation of false-positive rate, i.e., the threshold is computed only from negative samples. Positive (stego) samples occurs only in the objective function. Since generating of stego images is expensive and we do not need them to estimate false-positive rate, we decided to use 10% of all cover images to generate their stego counterparts. All images (both cover and stego images) are then described using 22 500 features and split into train / validation / test set in ratio 0.45 / 0.05 / 0.5. The resulting sizes of train / validation / test splits as well as the number of stego images in them, are in Table 6.2.

Since the resulting classification task is relatively easier to solve, we decided to use a simple linear model. The number o training samples and their size is not too big, therefore we can load the whole dataset into memory. It allows us to use full gradient descent instead of its stochastic version. As an optimizer, we use the ADAM [60] with default settings and fixed step length $\alpha = 0.01$. We also use a fixed number of epochs to 1000 for all formulations. Finally, we repeat each experiment ten times with ten different random seeds.

Figure 6.5 shows ROC curves for the test set of **Nsf5** dataset. For simplicity, we show ROC curves only for one run of the experiment. Moreover, Table 6.7 shows seven different performance metrics computed for each formulation. Each shown result in this table is a median of ten independent runs. It is evident, that *BinCross* provides very poor results for all metrics except the AUROC. Surprisingly, *BinCross* is the worst even for the AUROC. On the other hand, *DeepTopPush* excels at very low false-positive rates, as can be seen from both the table and the

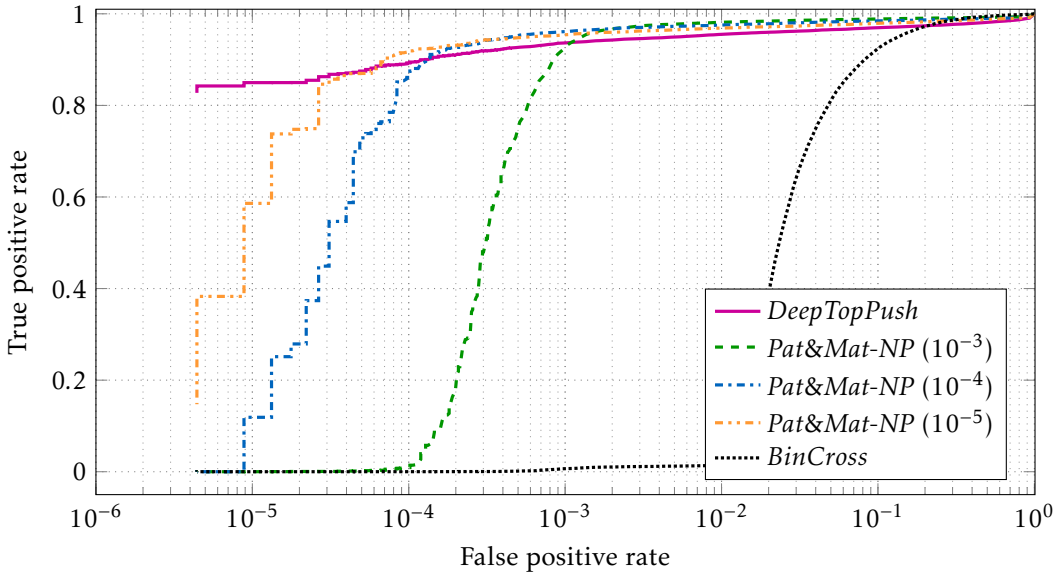


Figure 6.5: **Nsf5 dataset:** ROC curves with logaithmic x-axis.

Formulation	AUROC	TPR@K			TPR@τ		
		1	10	5	10^{-5}	10^{-4}	10^{-3}
BinCross	95.84	0.0	0.0	0.0	0.0	0.02	0.7
DeepTopPush	98.29	5.07	35.48	57.66	48.65	89.56	93.67
Pat&Mat-NP (10^{-5})	98.81	2.55	23.02	47.24	35.28	91.9	95.84
Pat&Mat-NP (10^{-4})	98.98	0.0	0.05	4.34	1.78	79.76	96.18
Pat&Mat-NP (10^{-3})	99.26	0.0	0.0	0.01	0.0	0.29	91.98

Table 6.7: **NSF5 dataset:** All presented results are medians of ten independent runs with different random seeds. Each column of the table corresponds to one performance metric and every row to one formulation. The best result for each metric is highlighted in green, while the worst result is highlighted in red.

figure. In fact, *DeepTopPush* provides the best results for four out of seven performance metrics (the best results are highlighted in green). Note that all these four metrics operate at extremely small false-positive rates. We can also see, that *Pat&Mat-NP* (10^{-5}) is the best at false-positive rate 10^{-4}. This unexpected result is probably caused by the approximation of the true top τ-quantile of all scores of negative samples in *Pat&Mat-NP* formulation. Therefore, *Pat&Mat-NP* (10^{-5}) is optimized for a false-positive rate slightly higher than 10^{-5} and as a consequence outperforms *Pat&Mat-NP* (10^{-5}) at false-positive rate 10^{-4}. Similar behavior can be seen for *Pat&Mat-NP* (10^{-4}) and *Pat&Mat-NP* (10^{-3}) at false-positive rate 10^{-4}.

6.3.2 JMiPOD

In this case, we first select all images that can be cropped to to size $256 \times 256 \times 3$ and than cropped them looselesly using *jpegtran* library. Than, we use JMiPOD [63] algorithm to generate stego images with payload 0.1. We use the same approach as in the case of Nsf5 dataset and use only 10% of cover images to generate their stego counterparts. We split the data into train

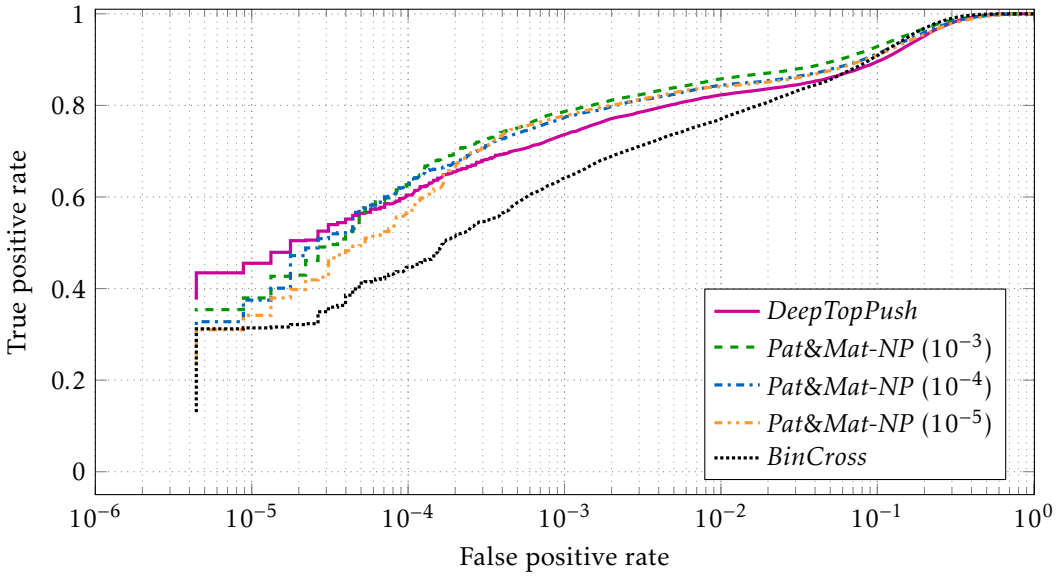


Figure 6.6: **JMiPOD dataset:** ROC curves with logaithmic x-axis.

Formulation	AUROC	TPR@K			TPR@τ		
		1	10	5	10^{-5}	10^{-4}	10^{-3}
BinCross	97.5	13.52	24.65	29.54	27.28	44.58	63.84
DeepTopPush	97.26	34.25	42.57	47.3	43.59	60.42	73.67
Pat&Mat-NP (10^{-5})	97.66	21.24	31.6	39.38	33.54	60.35	78.04
Pat&Mat-NP (10^{-4})	97.49	25.66	36.76	45.19	38.28	63.49	77.43
Pat&Mat-NP (10^{-3})	98.0	26.99	38.76	44.83	42.17	64.5	78.11

Table 6.8: **JMiPOD dataset:** All presented results are medians of ten independent runs with different random seeds. Each column of the table corresponds to one performance metric and every row to one formulation. The best result for each metric is highlighted in green, while the worst result is highlighted in red.

/ validation / test set in ratio 0.375 / 0.125 / 0.5. The resulting sizes of train / validation / test splits as well as the number of stego images in them, are in Table 6.2.

In this case, the resulting classification task is quite complicated, therefore we decided to use pre-trained EfficientNet-B0 [64] as a model. Originally the model is trained for 1000 classes, therefore, we removed the last fully-connected layer and replaced it with a randomly initialized fully-connected layer of appropriate size for binary classification. The resulting model is large and therefore it is not possible to use a full gradient. For this reason, we use stochastic gradient descent with balanced mini-batches of size 256. As an optimizer, we use the ADAM [60] with default settings and fixed step length $\alpha = 0.01$. Finally, we use a fixed number of epochs to 30 for all formulations and we repeat each experiment ten times with ten different random seeds.

As in the previous subsection, Figure 6.6 shows ROC curves for the test set of **JMiPOD** dataset, and Table 6.8 shows seven performance metrics. Each shown result in this table is a median of ten independent runs. Since trained models use stochastic gradient descent, the results are not as evident as in the case of Nsf5 dataset. *BinCross* still provides the worst results

for most of the metrics, but the differences are much smaller than for the Nsf5 dataset. We can see, that *DeepTopPush* again provides the best performance for 4 of 7 metrics. It shows that the enhanced minibatch used in *DeepTopPush* Algorithm 6 improves the approximation quality of the true threshold and therefore it reduces the bias of sampled gradient (as we already showed in Figure 5.2). Even though *Pat&Mat-NP* (10^{-3}), *Pat&Mat-NP* (10^{-4}) and *Pat&Mat-NP* (10^{-5}) were trained for different levels of false-positive rate, they all perform similarly. As we said before, the decision threshold t of *Pat&Mat-NP* model is the approximation of true top τ-quantile of all scores of negative samples. Since we use minibatches with 128 negative samples, the smallest quantile that can be found on this minibatch is $\tau = \frac{1}{128} = 0.0078125$. If we try to approximate smaller quantiles, we always get the same results. Therefore, *Pat&Mat-NP* (10^{-3}), *Pat&Mat-NP* (10^{-4}), *Pat&Mat-NP* (10^{-5}) should work almost identically, and we can see from both the figure and the table, that these three formulations provide very similar results.

6.4 Malware Detection

In the previous section, we presented results from the domain of steganalysis. Another domain in which formulations from presented framework can be very useful, is the domain of malware detection. As an example, consider standard antivirus software on a personal computer. Every user wants to be protected, so the goal of antivirus software is to detect as much malware as possible. However, if the antivirus software is too restrictive, it can easily happen, that clean software is marked as malware, i.e. the antivirus software can easily produce false alarms. If the antivirus software produces false alarms too often, it can be very annoying to the user and may lead to uninstalling the antivirus software. Therefore, the goal of every antivirus software is to maximize a true-positive rate at a very low false-positive rate, which is precisely what the formulations from the framework do.

In this section, we presented results on a real-world dataset provided by a renowned cyber-security company. The dataset consists of malware analysis reports of executable files. This is an extremely tough dataset as individual samples are JSON files whose size ranges from 1kB to 2.5MB. The structure of the sample 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. The usual approach how to process such complicated data is to create manually feature vectors and use them for training instead of the original data. However, such an approach is extremely time-demanding and requires expert knowledge of the original data. For this reason, we decided to use a different approach that is called Hierarchical Multiple Instance Learning (HMIL) [65]. For the training, we use a publicly available implementation of HMIL [66], which allows train models directly from JSON files without the necessity of complicated feature extraction.

Since the dataset is very large (see Table 6.2), we train each formulation only one time. Moreover, we use only the formulations that worked the best in the previous experiments, i.e. we use only the *BinCross*, *Pat&Mat-NP* (10^{-2}), *Pat&Mat-NP* (10^{-3}) and *BinCross*. As an optimizer, we use the ADAM [60] with default settings and fixed step length $\alpha = 0.01$. We also use balanced mini-batches of size 2000, which allows us to obtain a very good estimate of the true thresholds as discussed in Section 5.3. Finally, we use a fixed number of epochs to 100 for all formulations.

Figure 6.7 shows the performance of all formulations on the test set. For the comparison, we use ROC curves and we use filled circles to highlight the thresholds for which the formulation were optimized. It is clear, that *DeepTopPush* is the best at low false-positive rates. Even at the extremely low false positive rate $\tau = 10^{-5}$, *DeepTopPush* correctly identified 46% of malware. We can also see, that *Pat&Mat-NP* (10^{-3}) is the best at false-positive rate 10^{-3}, which is exactly the point for which the formulation should be optimized. However, *DeepTopPush* performs

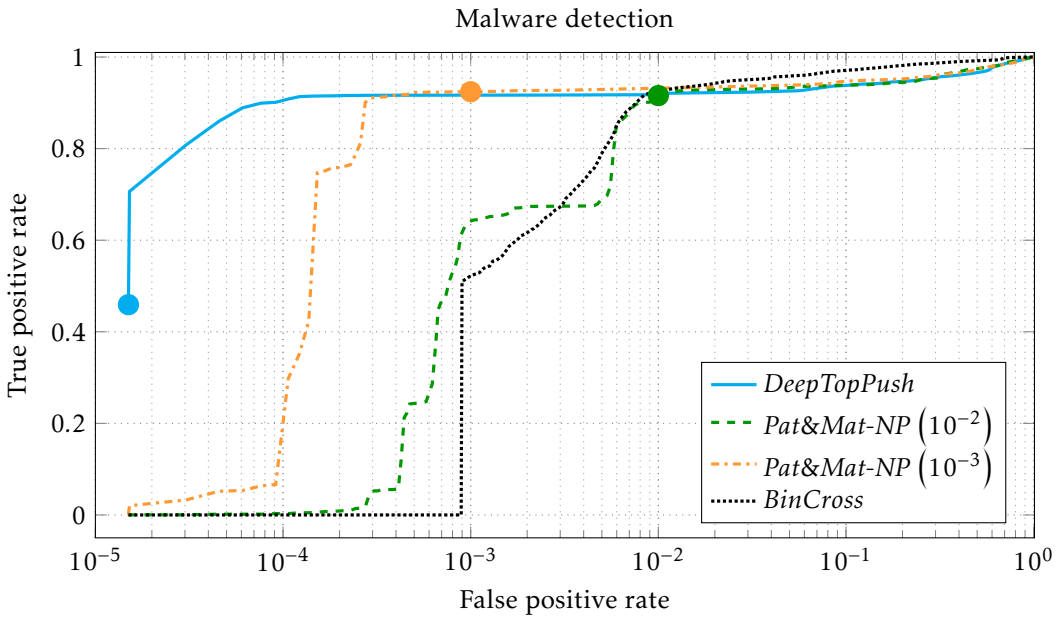


Figure 6.7: **Malware detection:** ROC curves with logaithmic x-axis. The circles show the thresholds the formulations were optimized for.

almost as well as *Pat&Mat-NP* (10^{-3}) at this false-positive rate. Finally, at the false-positive rate 10^{-2} all formulations perform equally well.

Conclusion

In this work, we studied the problem of classification at the top that is closely related to the binary classification. We showed that many of well known categories of problems such as ranking, accuracy at the top or hypothesis testing are closely related classification at the top. In Chapter 2, we study these three categories in detail and showed, that they can be all formulated in a similar way. This lead us to introduce unified framework for classification at the top (2.3). We showed that several known formulation (*TopPush*, *Grill*, τ-*FPL*) fall into our framework and derived some completely new formulations (*Pat&Mat*, *Pat&Mat-NP*). The summary of all presented formulations is in Table 2.1.

In Chapter 3, we performed a theoretical analysis of the presented formulations when the linear model is used. We showed that known formulations suffer from certain disadvantages. While *TopPush* and τ-*FPL* are sensitive to outliers, *Grill* is non-convex. On the other hand, we shoed that newly introduce *Pat&Mat* and *Pat&Mat-NP* formulations are robust and convex. We also proved the global convergence of the stochastic gradient descent for *Pat&Mat* and *Pat&Mat-NP*.

In Chapter 4, we extended the framework (2.3) to nonlinear problems. We showed, that all presented formulations (with the exception of *Grill* and *Grill-NP*) can be divided into two families based on the form of the constrains, namely *TopPushK* and *Pat&Mat* family of formulations. We derived dual formulation for *TopPushK* and *Pat&Mat* family of formulations. Moreover, we proposed a new coordinate descent algorithm for solving the resulting dual problems. For selected surrogate functions we also derived the closed-form formulas needed in the coordinate descent algorithm. Since the coordinate decent algorithm has to be initialized wit the feasible solution, we also showed how to find an initial feasible solution.

In Chapter 5, we study the primal formulations with non-linear models. We showed, that when we use non-linear model, the resulting formulations are non-decomposable. This property is caused by the special threshold constraint in (2.3), and prevents us from using of stochastic gradient descent in standard way. We introduce modified stochastic gradient descend for our formulations. Unfortunatelly, we showed that using of stochastic gradient descent peads to the biased eastimate of the true gradient. We suggested that this can migitiate by using large minibatche, however, such an approach is often not possible. For such cases, we proposed *DeepTopPush* as an efficient alternative to *TopPush* formulation, that does not suffer from this issue. For *DeepTopPush*, 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.

In Chapter 6, we performed a numerical comparison of presented formulations. We showed a good performance of our newly introduce formulation *Pat&Mat-NP*, when used in its primal form. We also showed, that *DeepTopPush* formulation can be very useful, especially for very large real-world dataset. On steganalysis datasets and malware detection dataset, we demonstrated, that standard formulations provide poor results at very low false-positive rates, while formulations proposed in this work, work very well on them.

Apendices

A

Appendix for Chapter 2

Lemma 2.7

Consider *Grill*, *Grill-NP*, *TopMeanK* and τ-*FPL* formulations and Notation 2.2. If the following inequality holds

$$s^+_{[n_+\tau]} > s^-_{[n_-\tau]},$$

then *Grill* has larger threshold than *Grill-NP*. In the same way, if the following inequality holds

$$\frac{1}{n_+\tau}\sum_{i=1}^{n_+\tau} s^+_{[i]} > \frac{1}{n_-\tau}\sum_{i=1}^{n_-\tau} s^-_{[i]}$$

then *TopMeanK* has larger threshold than τ-*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. ■

Lemma 2.5

Consider vector of scores $\boldsymbol{s}$ and its sorted version $\boldsymbol{s}_{[\cdot]}$ with non-increasing elements as defined in Notation 2.2, and threshold for *Pat&Mat* formulation

$$h(t) = \sum_{i\in\mathcal{I}} l(\vartheta(s_i - t)) - n\tau, \tag{2.14}$$

where $\vartheta > 0$ and l is the hinge loss from Notation 2.1. For all $i \in \mathcal{I}$ define $t_i = s_{[i]} + \frac{1}{\vartheta}$. Then for all $i = 2,\ 3, \ldots, n$ we have

$$h(t_i) = h(t_{i-1}) + (i-1)\vartheta(t_{i-1} - t_i), \tag{2.15}$$

with the initial condition $h(t_1) = -n\tau$.

Proof:
Observe that

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

where the last equality holds since $\vartheta > 0$ and $s_{[i]} - s_{[j]} \leq 0$ for all $i \geq j$. From here, we obtain $h(t_1) = -n\tau$. Moreover, we have

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

which finishes the proof. ■

B

Appendix for Chapter 3

B.1 Convexity

Proposition 3.1

Consider fixed vector of scores $\boldsymbol{s}$ with elements defined as $s_i = \boldsymbol{w}^\top \boldsymbol{x}_i$ for all $i \in \mathcal{I}$. Moreover, consider thresholds for *TopPush*, *Grill*, *TopMeanK* and *Pat&Mat* from Section 2.2 and 2.3 defined as

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

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

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 28:

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

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

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

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

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

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

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

Theorem 3.2

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

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

is convex.

Proof of Theorem 3.2 on page 28:
Due to the definition (2.2), the objective function L equals to

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

Here we write $t(w)$ to stress the dependence of t on w. Since $w \mapsto t(w)$ is a convex function, we also have that $w \mapsto t(w) - w^\top x$ 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. ■

B.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 w. Moreover, if the surrogate function l is differentiable, threshold t_3 is a differentiable function of weights w, and its derivative equals

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

Proof of Theorem 3.3 on page 28:
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. ■

B.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.

***TopPush* formulation (2.5):**

- For $\boldsymbol{w}_0 = (0,0)^\top$, all scores are equal to 0. Since the threshold t is the largest negative score, it also equals 0. Consequently, the value of the objective function is

$$L(\boldsymbol{w}_0) = \frac{1}{n_+}\sum_{i\in\mathcal{I}_+}\max\{0,\ 1+(0-0)\} = 1.$$

- For $\boldsymbol{w}_1 = (1,0)^\top$, the largest negative score equals 2; therefore $t = 2$. Then, the value of the objective function is

$$L(\boldsymbol{w}_1) \approx \int_0^1 \max\{0,\ 1+(2-s_i)\} = \int_0^1 (3-s)\,\mathrm{d}s = \frac{5}{2},$$

  where we can remove the max operator since all positive samples are uniformly distributed in $[0,1]\times[-1,1]$, and their scores are uniformly distributed in $[0,1]$.

***TopPushK* formulation (2.6):**

- For $\boldsymbol{w}_0 = (0,0)^\top$, the threshold and the objective function is the same as for *TopPush*.
- For $\boldsymbol{w}_1 = (1,0)^\top$, the threshold is the mean of K largest negative scores. The largest negative score equals 2 and for sufficiently large n, the rest of K largest negative scores equal 0. Therefore, the threshold is $t = \frac{2}{K}$. Then, the value of the objective function is

$$L(\boldsymbol{w}_1) \approx \int_0^1 \max\left\{0,\ 1+\left(\frac{2}{K}-s\right)\right\}\mathrm{d}s = \int_0^1 \left(1+\frac{2}{K}-s\right)\mathrm{d}s = \frac{1}{2}+\frac{2}{K},$$

  where we can remove the max operator since all positive scores are uniformly distributed in $[0,1]$.

***Grill* formulation (2.9):**

- For $\boldsymbol{w}_0 = (0,0)^\top$, all scores equal 0 and the threshold (top τ-quantile) is also 0. The objective reads

$$L(\boldsymbol{w}_0) = \frac{1}{n_+}\sum_{i\in\mathcal{I}_+}\max\{0,\ 1+(0-0)\} + \frac{1}{n_-}\sum_{i\in\mathcal{I}_-}\max\{0,\ 1+(0-0)\} = 1+1 = 2.$$

- For $\boldsymbol{w}_1 = (1,0)^\top$, all scores are uniformly distributed in $[-1,1]$ and the top τ-quantile of all score equals $t = 1-2\tau$. Then for $\tau \le \frac{1}{2}$, the value of the objective function is

$$\begin{aligned} L(\boldsymbol{w}_1) &\approx \int_0^1 \max\{0,\ 1+(1-2\tau-s)\}\,\mathrm{d}s + \int_{-1}^0 \max\{0,\ 1+(s-1+2\tau)\}\,\mathrm{d}s \\ &= \int_0^1 \max\{0, 2(1-\tau)-s\}\,\mathrm{d}s + \int_{-1}^0 \max\{0, s+2\tau\}\,\mathrm{d}s \\ &= \int_0^1 (2(1-\tau)-s)\,\mathrm{d}s + \int_{-2\tau}^0 (s+2\tau)\,\mathrm{d}s \\ &= \frac{3}{2} - 2\tau(1-\tau) \end{aligned}$$

***TopMeanK* formulation (2.11):**

- For $\boldsymbol{w}_0 = (0,0)^\top$, the threshold and objective function is the same as for *TopPushK*.
- For $\boldsymbol{w}_1 = (1,0)^\top$, the threshold is the mean of $K = n\tau$ largest scores. Since all scores are uniformly distributed in $[-1,1]$, the top $n\tau$ fraction of all scores is uniformely distributed in $[1-2\tau, 1]$. Therefore the threshold is $t = 1-\tau$. Then, the value of the objective function is

$$L(\boldsymbol{w}_1) \approx \int_0^1 \max\{0,\ 1+(1-\tau-s)\}\,\mathrm{d}s = \int_0^1 (2-\tau-s)\,\mathrm{d}s = \frac{3}{2} - \tau,$$

***Pat&Mat* formulation (2.13):**

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

$$\tau = \frac{1}{n}\sum_{i\in\mathcal{I}} \max\{0,\ 1+\vartheta(0-t)\} = \frac{1}{n}\sum_{i\in\mathcal{I}} (1-\vartheta t) = 1-\vartheta t,$$

  which implies that threshold t equals

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

  Consequently, the value of the objective function is

$$L(\boldsymbol{w}_0) = \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} \max\{0,\ 1+(t-0)\} = \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} (1+t) = 1+t, \tag{B.4}$$

  where the last equality follows the fact that $t \ge 0$.

- For $\boldsymbol{w}_1 = (1,0)^\top$, the computation is similar. All scores are uniformly distributed in $[-1,1]$. Then, if the scaling parameter ϑ satisfies $\vartheta \le \tau$, we have

$$\tau = \frac{1}{n}\sum_{i\in\mathcal{I}} \max\{0,\ 1+\vartheta(s_i-t)\} \approx \frac{1}{2}\int_{-1}^1 \max\{0,\ 1+\vartheta(s-t)\}\,\mathrm{d}s = \frac{1}{2}\int_{-1}^1 (1+\vartheta(s-t))\,\mathrm{d}s = 1-\vartheta t.$$

  which again implies that threshold t is $t = \frac{1}{\vartheta}(1-\tau)$. Note that we could ignore the max operator in the relation above, since

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

  where the last inequality follows from the assumption $\vartheta \le \tau$. Finally, since positive scores are uniformly distributed in $[0,1]$, the value of the objective function is

$$L(\boldsymbol{w}_1) = \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} \max\{0,\ 1+(t-s_i)\} \approx \int_0^1 \max\{0,\ 1+t-s\}\,\mathrm{d}s = \int_0^1 (1+t-s)\,\mathrm{d}s = \frac{1}{2}+t$$

***Grill-NP* formulation (2.18):**

- For $\boldsymbol{w}_0 = (0,0)^\top$, the threshold and objective function is the same as for *Grill*.
- For $\boldsymbol{w}_1 = (1,0)^\top$, negative scores are uniformly distributed in $[-1,0]$ and the top τ-quantile of negative score equals $t = -\tau$. Then, the value of the objective function is

$$\begin{aligned} L(\boldsymbol{w}_1) &\approx \int_0^1 \max\{0,\ 1 + (-\tau - s)\}\,\mathrm{d}s + \int_{-1}^0 \max\{0,\ 1 + (s+\tau)\}\,\mathrm{d}s \\ &= \int_0^{1-\tau} (1 - \tau - s)\,\mathrm{d}s + \int_{-1}^0 (1 + \tau + s)\,\mathrm{d}s = 1 + \frac{1}{2}\tau^2 \end{aligned}$$

***τ-FPL* formulation (2.20):**

- For $\boldsymbol{w}_0 = (0,0)^\top$, the threshold and objective function is the same as for *TopPushK*.
- For $\boldsymbol{w}_1 = (1,0)^\top$, the threshold is the mean of n_τ largest negative scores. Since negative scores are uniformly distributed in $[-1,0]$, top n_τ fraction of negative scores is uniformely distributed in $[-\tau,0]$. Therefore the threshold is $t = -\frac{1}{2}\tau$. Then, the value of the objective function i

$$L(\boldsymbol{w}_1) \approx \int_0^1 \max\left\{0,\ 1 + \left(-\frac{1}{2}\tau - s\right)\right\}\mathrm{d}s = \int_0^{1-\frac{1}{2}\tau} \left(1 - \frac{1}{2}\tau - s\right)\mathrm{d}s = \frac{1}{2} - \frac{1}{8}\tau(4+\tau)$$

***Pat&Mat-NP* formulation (2.22):**

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

$$\tau = \frac{1}{n_-}\sum_{i\in\mathcal{I}_-} \max\{0,\ 1 + \vartheta(0 - t)\} = \frac{1}{n_-}\sum_{i\in\mathcal{I}_-} (1 - \vartheta t) = 1 - \vartheta t,$$

  which implies that threshold is $t = \frac{1-\tau}{\vartheta}$. Consequently, the value of the objective function is

$$L(\boldsymbol{w}_0) = \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} \max\{0,\ 1 + (t - 0)\} = \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} (1 + t) = 1 + t,$$

  where the last equality follows the fact that $t \geq 0$.
- For $\boldsymbol{w}_1 = (1,0)^\top$, the computation is similar. Negative scores are uniformly distributed in $[-1,0]$. Then, if the scaling parameter ϑ satisfies $\vartheta \leq 2\tau$, we have

$$\begin{aligned} \tau &= \frac{1}{n_-}\sum_{i\in\mathcal{I}_-} \max\{0,\ 1 + \vartheta(s_i - t)\} \approx \int_{-1}^0 \max\{0,\ 1 + \vartheta(s - t)\}\,\mathrm{d}s \\ &= \int_{-1}^0 (1 + \vartheta(s - t))\,\mathrm{d}s = 1 - \vartheta t - \frac{1}{2}\vartheta. \end{aligned}$$

  which implies that threshold is $t = \frac{1}{\vartheta}(1-\tau) - \frac{1}{2}$. Note that we could ignore the max operator in the relation above, since

$$1 + \vartheta(s - t) \geq 1 + \vartheta(-1 - t) = 1 + \vartheta\left(-1 - \frac{1-\tau}{\vartheta} + \frac{1}{2}\right) = \tau - \frac{1}{2}\vartheta \geq 0,$$

  where the last inequality follows from the assumption $\vartheta \leq 2\tau$. Finally, since positive scores are uniformly distributed in $[0,1]$, the value of the objective function is

$$L(\boldsymbol{w}_1) = \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} \max\{0,\ 1 + (t - s_i)\} \approx \int_0^1 \max\{0,\ 1 + t - s\}\,\mathrm{d}s = \int_0^1 (1 + t - s)\,\mathrm{d}s = \frac{1}{2} + t.$$

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

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

Proof of Theorem 3.5 on page 30:

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 (2.21) such that $L(\boldsymbol{w}) < L(\mathbf{0})$ for all $\vartheta \in (0, \vartheta_0)$.

Proof of Theorem 3.8 on page 31:
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

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

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 a vector is smaller or equal to its maximum, we get the following relation

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

where the first inequality follows from (B.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

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

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

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

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

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

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 (B.6), we get

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

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 (B.7), the only inequality from (B.5). Using (B.3) and (B.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. ■

B.4 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 B.4.1, we prove a general statement for convergence of stochastic gradient descent with a convex objective function. In Section B.4.2 we apply it to Theorem 3.9. Finally, in Section B.4.3, we provide auxiliary results.

B.4.1 General Results

Consider a differentiable objective function L and the optimization method

$$\boldsymbol{w}^{k+1} = \boldsymbol{w}^k - \alpha^k g(\boldsymbol{w}^k), \tag{B.8}$$

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 a stepsize defined by

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

Theorem B.7

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

$$\sum_{k=0}^{\infty} \alpha^k \left\langle g(\boldsymbol{w}^k) - \nabla L(\boldsymbol{w}^k),\ \boldsymbol{w}^* - \boldsymbol{w}^k \right\rangle \leq C, \tag{B.9}$$

then the sequence $\{\boldsymbol{w}^k\}$ generated by (B.8) 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

$$\left\langle \nabla L(\boldsymbol{w}^k),\ \boldsymbol{w}^* - \boldsymbol{w}^k \right\rangle \leq L(\boldsymbol{w}^*) - L(\boldsymbol{w}^k). \tag{B.10}$$

Then we have

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

where the first inequality follows from assumption (A2) and the second one from the properties of inner product and (B.10). Summing this expression for all k and using (B.9) leads to

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

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

$$\limsup_{k\to\infty} \left\|\boldsymbol{w}^k - \boldsymbol{w}^*\right\|^2 + 2\sum_{k=0}^{\infty} \alpha^k \left(L(\boldsymbol{w}^k) - L(\boldsymbol{w}^*)\right) \leq 2\hat{C}. \tag{B.11}$$

Since $\alpha^k > 0$ and $L(\boldsymbol{w}^k) \geq L(\boldsymbol{w}^*)$ as $\boldsymbol{w}^*$ is a global minimizer of L, we infer that sequence $\{\boldsymbol{w}^k\}$ is bounded and (B.11) implies

$$\sum_{k=0}^{\infty} \alpha^k \left(L(\boldsymbol{w}^k) - L(\boldsymbol{w}^*)\right) \leq \hat{C}.$$

Since $L(\boldsymbol{w}^k) - L(\boldsymbol{w}^*) \geq 0$, due to assumption (A3) we obtain

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

which implies the theorem statement. ■

B.4.2 Proof of Theorem 3.9

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

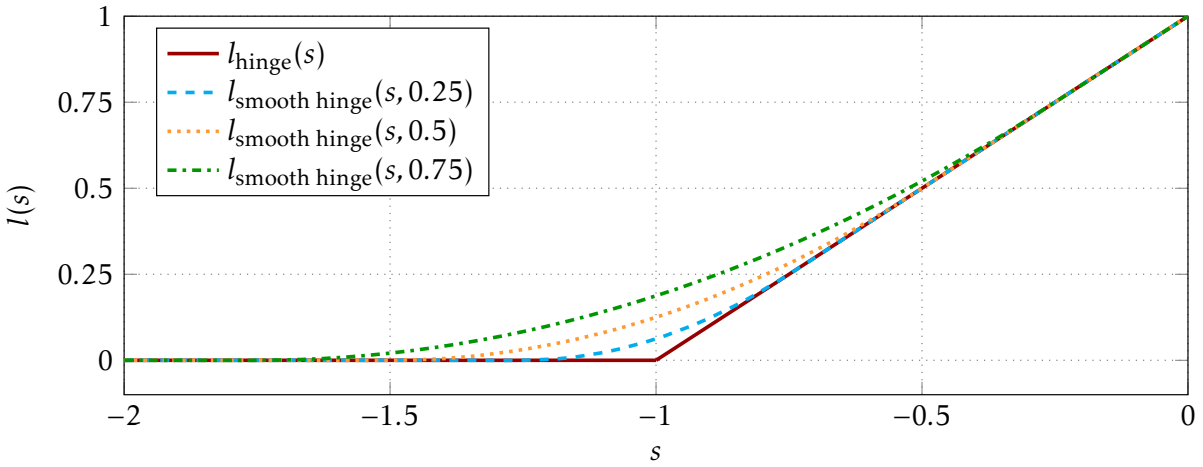


Figure B.1: Comparison of the hinge loss and its smoothened version with $\varepsilon = 0.25$, $\varepsilon = 0.5$, and $\varepsilon = 0.75$.

(**S1**) $l(s) \geq 0$ 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 $(s_0, +\infty)$, where $s_0 := \sup\{s \mid l(s) = 0\}$;

(**S3**) $\frac{l'}{l}$ is a decreasing function on $(s_0, +\infty)$;

(**S4**) l' is a bounded function;

(**S5**) l' is a Lipschitz continuous function with Lipschitz constant D.

For simplicity, we also assume that the scaling parameter for the threshold in the *Pat&Mat* formulation is $\vartheta = 1$, and regularization parameter is $\lambda = 1$. All requirements above are satisfied for the hinge loss smoothened on an ε-neighborhood of -1

$$l(s) = \begin{cases} 0 & \text{for } s < -1 - \varepsilon, \\ \frac{1}{4\varepsilon}(1 + s + \varepsilon)^2 & \text{for } -1 - \varepsilon \leq s < -1 + \varepsilon, \\ 1 + s & \text{otherwise.} \end{cases}$$

Figure B.1 shows the comparison of the hinge loss and its smoothened version with different ε.

Theorem 3.9

Consider the *Pat&Mat* formulation, stepsizes $\alpha^k = \frac{1}{k+1}\alpha^0$, and piecewise disjoint mini-batches $\mathcal{I}^1_{\text{mb}}$, $\mathcal{I}^2_{\text{mb}}$,..., $\mathcal{I}^m_{\text{mb}}$ which cycle periodically $\mathcal{I}^{k+m}_{\text{mb}} = \mathcal{I}^k_{\text{mb}}$. If l is the smoothened hinge function defined by

$$l(s) = \begin{cases} 0 & \text{for } s < -1 - \varepsilon, \\ \frac{1}{4\varepsilon}(1 + s + \varepsilon)^2 & \text{for } -1 - \varepsilon \leq s < -1 + \varepsilon, \\ 1 + s & \text{otherwise,} \end{cases} \tag{3.11}$$

where $\varepsilon > 0$, then Algorithm 2 converges to the global minimum of (2.13).

Proof of Theorem 3.9 on page 33:
We intend to apply Theorem B.7 and thus, we need to verify its assumptions. Recall the form of the objective function

$$L(\boldsymbol{w}^k) = \frac{1}{2}\left\|\boldsymbol{w}^k\right\|^2 + \frac{1}{n_+}\sum_{i\in\mathcal{I}_+} l(t(\boldsymbol{w}^k) - \boldsymbol{x}_i^\top \boldsymbol{w}^k).$$

The objective is a combination of squared norm of weights, which is strictly convex function, and a surrogate approximation of false-negative rate, which is convex function due to Theorem 3.2. Therefore, the objective L is a strictly convex function. Moreover, it is also differentiable due to Theorem 3.3 and differentiability of smoothened hinge loss l. As a result, Assumption **(A1)** is satisfied.

Lemma B.10 says that $\left\|g(\boldsymbol{w}^k)\right\| \leq \left\|\boldsymbol{w}^k\right\| + \hat{B}$ for all k. To show that Assumption **(A2)** is satisfied, we have to show, that $\left\|\boldsymbol{w}^k\right\|$ is uniformly bounded. Consider sufficiently large k such that $\alpha^k < 1$. Then

$$\begin{aligned}\left\|\boldsymbol{w}^{k+1}\right\| &= \left\|\boldsymbol{w}^k - \alpha^k g(\boldsymbol{w}^k)\right\| = \left\|(1-\alpha^k)\boldsymbol{w}^k - \alpha^k \frac{1}{n^k_{\mathrm{mb},+}}\sum_{i\in\mathcal{I}^k_{\mathrm{mb},+}} l'(t^k - s_i^k)(\nabla t^k - \boldsymbol{x}_i)\right\| \\ &\leq (1-\alpha^k)\left\|\boldsymbol{w}^k\right\| + \alpha^k \left\|\frac{1}{n^k_{\mathrm{mb},+}}\sum_{i\in\mathcal{I}^k_{\mathrm{mb},+}} l'(t^k - s_i^k)(\nabla t^k - \boldsymbol{x}_i)\right\| \leq (1-\alpha^k)\left\|\boldsymbol{w}^k\right\| + \alpha^k \hat{B},\end{aligned}$$

where the first inequality follows from the triangle inequality and the last from the proof of Lemma B.10. Now we have two possible cases

- If $\left\|\boldsymbol{w}^k\right\| \leq \hat{B}$, then we get

$$\left\|\boldsymbol{w}^{k+1}\right\| \leq (1-\alpha^k)\left\|\boldsymbol{w}^k\right\| + \alpha^k \hat{B} \leq (1-\alpha^k)\hat{B} + \alpha^k \hat{B} = \hat{B}.$$

- If $\left\|\boldsymbol{w}^k\right\| > \hat{B}$, then we get

$$\left\|\boldsymbol{w}^{k+1}\right\| \leq (1-\alpha^k)\left\|\boldsymbol{w}^k\right\| + \alpha^k B < (1-\alpha^k)\left\|\boldsymbol{w}^k\right\| + \alpha^k \left\|\boldsymbol{w}^k\right\| = \left\|\boldsymbol{w}^k\right\|.$$

Therefore, for sufficiently large k, we have $\left\|\boldsymbol{w}^k\right\| \leq \max\left\{\hat{B}, \left\|\boldsymbol{w}^0\right\|\right\}$. Combining this with Lemma B.10, we get $\left\|g(\boldsymbol{w}^k)\right\| \leq B$, for some B, and Assumption **(A2)** is satisfied.

Assumptions **(A3)**-**(A5)** are imposed directly in the statement of this theorem.

It remains to verify (B.9). 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 $\mathcal{I}^k_{\mathrm{mb}}$ and $\mathcal{I}^{k+1}_{\mathrm{mb}}$, which are pairwise disjoint and cover all samples. Moreover, for all k, we have $\mathcal{I}^k_{\mathrm{mb}} = \mathcal{I}^{k+2}_{\mathrm{mb}}$. Furthermore, the assumptions imply that the number of positive samples in each minibatch is equal to $n^k_{\mathrm{mb},+} = \frac{1}{2}n_+$, where n_+ is the total number of positive samples.

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

$$\begin{aligned}s_i^{k-1} = s_i^{k-2} &= \boldsymbol{x}_i^\top \boldsymbol{w}^{k-2} = \boldsymbol{x}_i^\top\left(\boldsymbol{w}^k + \alpha^{k-2} g(\boldsymbol{w}^{k-2}) + \alpha^{k-1} g(\boldsymbol{w}^{k-1})\right) \\ &= \boldsymbol{x}_i^\top \boldsymbol{w}^k + \alpha^{k-2}\boldsymbol{x}_i^\top g(\boldsymbol{w}^{k-2}) + \alpha^{k-1}\boldsymbol{x}_i^\top g(\boldsymbol{w}^{k-1}).\end{aligned} \tag{B.12}$$

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

$$s_i^k = s_i^{k-1} = \boldsymbol{x}_i^\top \boldsymbol{w}^{k-1} = \boldsymbol{x}_i^\top (\boldsymbol{w}^k + \alpha^{k-1} g(\boldsymbol{w}^{k-1})) = \boldsymbol{x}_i^\top \boldsymbol{w}^k + \alpha^{k-1} \boldsymbol{x}_i^\top g(\boldsymbol{w}^{k-1}). \tag{B.13}$$

Recall that we already verified (A1)-(A5). Combining (A2) with (B.12) and (B.13) yields the existence of some C_2 such that for all $i \in \mathcal{I}$ we have

$$\left\| s_i^k - \boldsymbol{x}_i^\top \boldsymbol{w}^k \right\| \leq C_2 \alpha^{k-1}, \qquad \left\| s_i^{k-1} - \boldsymbol{x}_i^\top \boldsymbol{w}^k \right\| \leq C_2 \left(\alpha^{k-1} + \alpha^{k-2} \right). \tag{B.14}$$

This also immediately implies

$$\left\| t^k - t(\boldsymbol{w}^k) \right\| \leq C_2 \alpha^{k-1}, \qquad \left\| t^{k-1} - t(\boldsymbol{w}^k) \right\| \leq C_2 \left(\alpha^{k-1} + \alpha^{k-2} \right). \tag{B.15}$$

Moreover, we know that l' is Lipschitz continuous with Lipschitz constant D according to (S5). Then due to (B.14) and (B.15) we get

$$\left\| l'(t^k - s_i^k) - l'(t(\boldsymbol{w}^k) - \boldsymbol{x}_i^\top \boldsymbol{w}^k) \right\| \leq D \left\| t^k - s_i^k - t(\boldsymbol{w}^k) + \boldsymbol{x}_i^\top \boldsymbol{w}^k \right\| \leq 2 C_2 D \alpha^{k-1}. \tag{B.16}$$

In an identical way, we can derive the following relations

$$\begin{aligned} \left\| l'(t^{k-1} - s_i^{k-1}) - l'(t(\boldsymbol{w}^k) - \boldsymbol{x}_i^\top \boldsymbol{w}^k) \right\| &\leq 2 C_2 D \left(\alpha^{k-1} + \alpha^{k-2} \right), \\ \left\| l'(s_i^k - t^k) - l'(\boldsymbol{x}_i^\top \boldsymbol{w}^k - t(\boldsymbol{w}^k)) \right\| &\leq 2 C_2 D \alpha^{k-1}, \\ \left\| l'(s_i^{k-1} - t^{k-1}) - l'(\boldsymbol{x}_i^\top \boldsymbol{w}^k - t(\boldsymbol{w}^k)) \right\| &\leq 2 C_2 D \left(\alpha^{k-1} + \alpha^{k-2} \right). \end{aligned} \tag{B.17}$$

Now we need to estimate the distance between $\nabla t(\boldsymbol{w}^k)$ and ∇t^k. By plugging (3.9) into (3.10), we get

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

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

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

From Lemma B.9 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 B.11 which together with (B.17) implies that there is some C_3 such that for all sufficiently large k we have

$$\left\| \nabla t^k - \nabla t(\boldsymbol{w}^k) \right\| \leq C_3 \left(\alpha^{k-1} + \alpha^{k-2} \right). \tag{B.18}$$

Using the assumptions above, we can simplify the terms for $g(\boldsymbol{w}^k)$ from (3.8) and $\nabla L(\boldsymbol{w}^k)$

from (3.5) to

$$g(w^k) = w^k + \frac{2}{n_+} \sum_{i \in \mathcal{I}^k_{\mathrm{mb},+}} l'(t^k - s_i^k)(\nabla t^k - x_i),$$

$$g(w^{k+1}) = w^{k+1} + \frac{2}{n_+} \sum_{i \in \mathcal{I}^{k+1}_{\mathrm{mb},+}} l'(t^{k+1} - s_i^{k+1})(\nabla t^{k+1} - x_i),$$

$$\nabla L(w^k) = w^k + \frac{1}{n_+} \sum_{i \in \mathcal{I}_+} l'(t(w^k) - x_i^\top w^k)(\nabla t(w^k) - x_i),$$

$$\nabla L(w^{k+1}) = w^{k+1} + \frac{1}{n_+} \sum_{i \in \mathcal{I}_+} l'(t(w^{k+1}) - x_i^\top w^{k+1})(\nabla t(w^{k+1}) - x_i).$$

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

$$\begin{aligned}
&n_+\Big(g(w^k) + g(w^{k+1}) - \nabla L(w^k) - \nabla L(w^{k+1})\Big) && \text{(B.19a)}\\
&= \sum_{i \in \mathcal{I}^k_{\mathrm{mb},+}} l'(t^k - s_i^k)(\nabla t^k - x_i) - \sum_{i \in \mathcal{I}^k_{\mathrm{mb},+}} l'(t(w^k) - x_i^\top w^k)(\nabla t(w^k) - x_i) && \text{(B.19b)}\\
&+ \sum_{i \in \mathcal{I}^k_{\mathrm{mb},+}} l'(t^k - s_i^k)(\nabla t^k - x_i) - \sum_{i \in \mathcal{I}^k_{\mathrm{mb},+}} l'(t(w^{k+1}) - x_i^\top w^{k+1})(\nabla t(w^{k+1}) - x_i) && \text{(B.19c)}\\
&+ \sum_{i \in \mathcal{I}^{k+1}_{\mathrm{mb},+}} l'(t^{k+1} - s_i^{k+1})(\nabla t^{k+1} - x_i) - \sum_{i \in \mathcal{I}^{k+1}_{\mathrm{mb},+}} l'(t(w^k) - x_i^\top w^k)(\nabla t(w^k) - x_i) && \text{(B.19d)}\\
&+ \sum_{i \in \mathcal{I}^{k+1}_{\mathrm{mb},+}} l'(t^{k+1} - s_i^{k+1})(\nabla t^{k+1} - x_i) - \sum_{i \in \mathcal{I}^{k+1}_{\mathrm{mb},+}} l'(t(w^{k+1}) - x_i^\top w^{k+1})(\nabla t(w^{k+1}) - x_i). && \text{(B.19e)}
\end{aligned}$$

Then relations (B.16) and (B.18) applied to Lemma B.12 imply

$$\left\| \sum_{i \in \mathcal{I}^k_{\mathrm{mb},+}} l'(t^k - s_i^k)(\nabla t^k - x_i) - \sum_{i \in \mathcal{I}^k_{\mathrm{mb},+}} l'(t(w^k) - x_i^\top w^k)(\nabla t(w^k) - x_i) \right\| \leq C_4\Big(\alpha^{k-1} + \alpha^{k-2}\Big)$$

for some C_4, which gives a bound for (B.19b). Bound for (B.19e) is obtained by increasing k by one. Bounds for (B.19c) and (B.19d) can be find similarly using (B.17). Altogether, we showed

$$\left\|g(w^k) + g(w^{k+1}) - \nabla L(w^k) - \nabla L(w^{k+1})\right\| \leq C_1(\alpha^{k-2} + \alpha^{k-1} + \alpha^k + \alpha^{k+1}) \qquad \text{(B.20)}$$

for some C_1.

We now estimate

$$\begin{aligned}
&\alpha^k\Big\langle g(w^k) - \nabla L(w^k),\, w^* - w^k\Big\rangle + \alpha^{k+1}\Big\langle g(w^{k+1}) - \nabla L(w^{k+1}),\, w^* - w^{k+1}\Big\rangle\\
&= \Big\langle g(w^k) - \nabla L(w^k),\, \alpha^k(w^* - w^k)\Big\rangle + \Big\langle g(w^{k+1}) - \nabla L(w^{k+1}),\, \alpha^{k+1}(w^* - w^{k+1})\Big\rangle\\
&= \Big\langle g(w^k) - \nabla L(w^k) + g(w^{k+1}) - \nabla L(w^{k+1}),\, \alpha^k(w^* - w^k)\Big\rangle\\
&+ \Big\langle g(w^{k+1}) - \nabla L(w^{k+1}),\, \alpha^{k+1}(w^* - w^{k+1}) - \alpha^k(w^* - w^k)\Big\rangle.
\end{aligned} \qquad \text{(B.21)}$$

To estimate the second part of the right hand side of (B.21), we make use of Lemma B.10 to

obtain the existence of some C_5 such that

$$\begin{aligned}
&\left\langle g(\boldsymbol{w}^{k+1}) - \nabla L(\boldsymbol{w}^{k+1}),\ \alpha^{k+1}(\boldsymbol{w}^* - \boldsymbol{w}^{k+1}) - \alpha^k(\boldsymbol{w}^* - \boldsymbol{w}^k)\right\rangle \\
&\leq 2B\left\|\alpha^{k+1}(\boldsymbol{w}^* - \boldsymbol{w}^{k+1}) - \alpha^k(\boldsymbol{w}^* - \boldsymbol{w}^k)\right\| \\
&= 2B\left\|\alpha^{k+1}(\boldsymbol{w}^* - \boldsymbol{w}^k + \alpha^k g(\boldsymbol{w}^k)) - \alpha^k(\boldsymbol{w}^* - \boldsymbol{w}^k)\right\| \\
&= 2B\left\|(\alpha^{k+1} - \alpha^k)\boldsymbol{w}^* + (\alpha^k - \alpha^{k+1})\boldsymbol{w}^k + \alpha^k\alpha^{k+1} g(\boldsymbol{w}^k)\right\| \\
&\leq C_5\left\|\alpha^{k+1} - \alpha^k\right\| + C_5(\alpha^k)^2 + C_5(\alpha^{k+1})^2.
\end{aligned} \tag{B.22}$$

In the last inequality we used the inequality $2ab \leq a^2 + b^2$. To estimate the first part of the right hand side of (B.21), we can apply (B.20) together with the boundedness of $\{\boldsymbol{w}^k\}$ to obtain the existence of some C_6 such that

$$\begin{aligned}
&\left\langle g(\boldsymbol{w}^k) - \nabla L(\boldsymbol{w}^k) + g(\boldsymbol{w}^{k+1}) - \nabla L(\boldsymbol{w}^{k+1}),\ \alpha^k(\boldsymbol{w}^* - \boldsymbol{w}^k)\right\rangle \\
&\qquad \leq C_6(\alpha^{k-2})^2 + C_6(\alpha^{k-1})^2 + C_6(\alpha^k)^2 + C_6(\alpha^{k+1})^2.
\end{aligned} \tag{B.23}$$

Plugging (B.22) and (B.23) into (B.21) and summing the terms yields (B.9). Then the assumptions of Theorem B.7 are verified and the theorem statement follows. ■

B.4.3 Auxiliary Results

Lemma B.9

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

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

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

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

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

$$l^{-1}(n\tau) \geq s_{i_0}^k - t^k. \tag{B.24}$$

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 (B.24) 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 B.10

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

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

Proof:
By applying norm to the gradient (3.5) of the objective function L, we get

$$\left\|\nabla L(\boldsymbol{w}^k)\right\| \leq \left\|\boldsymbol{w}^k\right\| + \left\|\frac{1}{n_+}\sum_{i\in\mathcal{I}_+} l'(t(\boldsymbol{w}^k) - \boldsymbol{x}_i^\top \boldsymbol{w}^k)(\nabla t(\boldsymbol{w}^k) - \boldsymbol{x}_i)\right\|,$$

where the inequality follows from the triangle inequality. Due to (S4) the derivative l' is bounded by some $\hat{C}_1$. Then Theorem 3.3 and Lemma B.9 imply

$$\left\|\nabla t(\boldsymbol{w}^k)\right\| \leq \frac{\hat{C}_1 \sum_{i\in\mathcal{I}} \|\boldsymbol{x}_i\|}{\sum_{i\in\mathcal{I}} l'(\boldsymbol{x}_i^\top \boldsymbol{w}^k - t(\boldsymbol{w}^k))} \leq \frac{\hat{C}_1}{\hat{C}_2}\sum_{i\in\mathcal{I}} \|\boldsymbol{x}_i\|,$$

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

Lemma B.11

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 B.12

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. ■

C Appendix for Chapter 4

C.1 Derivation of Dual Problems

C.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

	K	$\mathbb{K}$	$\tilde{n}$	$\tilde{\boldsymbol{x}}_j$
TopPush	1	$\mathbb{K}^{-}$	n_-	$\boldsymbol{x}_j^-$
TopPushK	K	$\mathbb{K}^{-}$	n_-	$\boldsymbol{x}_j^-$
TopMeanK	$n\tau$	$\mathbb{K}^{\pm}$	n	$\boldsymbol{x}_j$
τ-*FPL*	$n_-\tau$	$\mathbb{K}^{-}$	n_-	$\boldsymbol{x}_j^-$

If $K = 1$, the upper bound in the second constraint (4.5c) vanishes due to the first constraint. Finally, the primal variables $\boldsymbol{w}$ can be computed from dual variables as follows

$$\boldsymbol{w} = \sum_{i=1}^{n_+}\alpha_i\boldsymbol{x}_i^+ - \sum_{j=1}^{\tilde{n}}\beta_j\tilde{\boldsymbol{x}}_j. \tag{4.6}$$

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 [67], 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

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

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

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

The combination of all the above relations and the use of a linear model yields to

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

The Lagrangian of this formulation is defined as

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

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 objective function (to be maximized) reads

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

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

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

$$+\min_{\boldsymbol{z}}\, \sum_{j=1}^{n_-}\left(\frac{1}{K}\sum_{i=1}^{n_+}\alpha_i - \beta_j - \gamma_j\right)z_j \tag{C.1d}$$

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 (4.5a), ane we also get the relation (4.6) between primal and dual variables.

Optimality condition with respect to t reads

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

and implies constrain (4.5b).

Similarly, optimality condition of (C.1d) with respect to $\boldsymbol{z}$ reads for all $j = 1,\dots, n_-$ 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 (4.5c).

Finally, the second part of the objective function (4.5a) follows from Definition 4.1 of the conjugate function. Using the definition, minimization of (C.1c) with respect to $\boldsymbol{y}$ yields

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

for all $i = 1,\dots, n_+$, which finishes the proof for *TopPushK*. For *TopPush*, we have $K = 1$. From (4.5b) and non-negativity of β_j we deduce that the upper bound in (4.5c) is always fulfilled and can be omitted. ■

C.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 (4.7). Then the corresponding dual problem has the following form

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta},\delta}{\text{maximize}} \quad -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^{\top}\mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} - C\sum_{i=1}^{n_+} l^{\star}\left(\frac{\alpha_i}{C}\right) - \delta\sum_{j=1}^{\tilde{n}} l^{\star}\left(\frac{\beta_j}{\delta\vartheta}\right) - \delta\tilde{n}\tau \tag{4.8a}$$

$$\text{subject to} \quad \sum_{i=1}^{n_+}\alpha_i = \sum_{j=1}^{\tilde{n}}\beta_j, \tag{4.8b}$$

$$\delta \geq 0, \tag{4.8c}$$

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

	$\mathbb{K}$	$\tilde{n}$	$\tilde{\boldsymbol{x}}_j$
Pat&Mat	$\mathbb{K}^{\pm}$	n	$\boldsymbol{x}_j$
Pat&Mat-NP	$\mathbb{K}^{-}$	n_-	$\boldsymbol{x}_j^-$

Finally, the primal variables $\boldsymbol{w}$ can be computed from dual variables as follows

$$\boldsymbol{w} = \sum_{i=1}^{n_+}\alpha_i\boldsymbol{x}_i^+ - \sum_{j=1}^{\tilde{n}}\beta_j\tilde{\boldsymbol{x}}_j. \tag{4.9}$$

Proof:
For simplicity, we show the proof only for *Pat&Mat-NP*, i.e. the threshold is computed only from negative samples. Let us first realize that formulation (4.7) 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 objective function

(to be maximized) can be rewritten as follows

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

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

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

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

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

Note that the resulting dual function is very similar to one (C.1) for *TopPushK*. In fact, the first three parts of (C.1) and (C.2) are identical. Therefore, we only have to show how to minimize (C.2) with respect to $\boldsymbol{z}$. For that, we can use the conjugate function as in the case of minimization of (C.1) 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 (C.2d) yields the third part of the objective function (4.8a), which finishes the proof. ■

C.2 Coordinate Descent Algorithm

C.2.1 Family of *TopPushK* Formulations

Hinge Loss

Proposition 4.7: Update rule (4.14a) for problem (4.16)

Consider problem (4.16), update rule (4.14a), indices $1 \leq k \leq n_+$ and $1 \leq l \leq n_+$ and Notation 4.6. Then the optimal solution $\Delta^\star$ is given by (4.15) 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 4.7 on page 43:
Constraint (4.16b) is always satisfied from the definition of the update rule (4.14a), and constraint (4.16d) is always satisfied since no β_j was updated and the sum of all α_i did not change. Constraint (4.16c) 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 (4.14a), objective function (4.16a) can be rewritten as a quadratic function with respect to Δ

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

Finally, the optimal solution $\Delta^\star$ is given by (4.15). ■

Proposition 4.8: Update rule (4.14b) for problem (4.16)

Consider problem (4.16), update rule (4.14b), indices $1 \le k \le n_+$ and $n_+ + 1 \le l \le \tilde{n}$ and Notation 4.6. 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 (4.15) where

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

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

$$\gamma = -\frac{s_k+s_l-1}{\mathbb{K}_{kk}+\mathbb{K}_{ll}+\mathbb{K}_{kl}+\mathbb{K}_{lk}}.$$

Proof of Proposition 4.8 on page 44:
Constraint (4.16b) is always satisfied from the definition of the update rule (4.14b). Constraint (4.16c) reads $-\alpha_{\hat{k}} \le \Delta \le C-\alpha_{\hat{k}}$. Using the definition of $\beta_{\max}$, constraint (4.16d) for any $K \ge 2$ reads

$$0 \le \beta_{\max} \le \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i + \frac{\Delta}{K} \quad \Longrightarrow \quad K\beta_{\max} - \sum_{i=1}^{n_+}\alpha_i \le \Delta,$$

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

The combination of these bounds yields the lower bound Δ_{lb} and upper bound Δ_{ub}. If $K=1$, the upper bound in (4.16d) is always satisfied due to (4.16b) and the lower and upper bound of Δ can be simplified.

Using the update rule (4.14b), objective function (4.16a) can be rewritten as a quadratic function with respect to Δ

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

Finally, the optimal solution $\Delta^\star$ is given by (4.15). ■

Proposition 4.9: Update rule (4.14c) for problem (4.16)

Consider problem (4.16), update rule (4.14c), indices $n_+ + 1 \leq k \leq \tilde{n}$ and $n_+ + 1 \leq l \leq \tilde{n}$ and Notation 4.6. Then the optimal solution $\Delta^\star$ is given by (4.15) where

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

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

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

Proof of Proposition 4.9 on page 44:

Constraint (4.16b) is always satisfied from the definition of the update rule (4.14c), and constraint (4.16c) is satisfied since no α_i is updated. Constraint (4.16d) 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 (4.16d) is always satisfied due to (4.16b) and the lower and upper bound of Δ can be simplified.

Using the update rule (4.14c), objective function (4.16a) can be rewritten as a quadratic function with respect to Δ

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

Finally, the optimal solution $\Delta^\star$ is given by (4.15). ■

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

$$\underset{\boldsymbol{\alpha}, \boldsymbol{\beta}}{\text{maximize}} \quad -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha} \\ \boldsymbol{\beta}\end{pmatrix}^\top \mathbb{K}\begin{pmatrix}\boldsymbol{\alpha} \\ \boldsymbol{\beta}\end{pmatrix} + \sum_{i=1}^{n_+} \alpha_i - \frac{1}{4C}\sum_{i=1}^{n_+} \alpha_i^2 \tag{C.3a}$$

$$\text{subject to} \quad \sum_{i=1}^{n_+} \alpha_i = \sum_{j=1}^{\tilde{n}} \beta_j, \tag{C.3b}$$

$$0 \leq \alpha_i, \qquad i = 1, 2, \ldots, n_+, \tag{C.3c}$$

$$0 \leq \beta_j \leq \frac{1}{K}\sum_{i=1}^{n_+} \alpha_i, \quad j = 1, 2, \ldots, \tilde{n}, \tag{C.3d}$$

Similarly to the previous case, the form of $\mathbb{K}$ and $\tilde{n}$ depends on the used formulation and the upper bound in (C.3d) can be omitted for $K = 1$.

Proposition C.6: Update rule (4.14a) for problem (C.3)

Consider problem (C.3), update rule (4.14a), indeices $1 \le k \le n_+$ and $1 \le l \le n_+$ and Notation 4.6. Then the optimal solution $\Delta^\star$ is given by (4.15) 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 (C.3b) is always satisfied from the definition of the update rule (4.14a). Constraint (C.3d) is also always satisfied since no β_j was updated and the sum of all α_i did not change. Constraint (C.3c) reads

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

which gives the lower and upper bound of Δ.

Using the update rule (4.14a), objective function (C.3a) can be rewritten as a quadratic function with respect to Δ

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

Finally, the optimal solution $\Delta^\star$ is given by (4.15). ■

Proposition C.7: Update rule (4.14b) for problem (C.3)

Consider problem (C.3), update rule (4.14b), indeices $1 \le k \le n_+$ and $n_+ + 1 \le l \le \tilde{n}$ and Notation 4.6. 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 (4.15) 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 (C.3b) is always satisfied from the definition of the update rule (4.14b). Constraint (C.3c) reads $-\alpha_{\hat{k}} \le \Delta$. Using the definition of $\beta_{\max}$, constraint (C.3d) for any $K \ge 2$ reads

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

The combination of these bounds yields the lower bound Δ_{lb} and upper bound Δ_{ub}. If $K = 1$, the upper bound in (C.3d) is always satisfied due to (C.3b) and the lower and upper bound of Δ can be simplified.

Using the update rule (4.14b), objective function (C.3a) can be rewritten as a quadratic function with respect to Δ

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

Finally, the optimal solution $\Delta^\star$ is given by (4.15). ■

Proposition C.8: Update rule (4.14c) for problem (C.3)

Consider problem (C.3), update rule (4.14c), indices $n_+ + 1 \leq k \leq \tilde{n}$ and $n_+ + 1 \leq l \leq \tilde{n}$ and Notation 4.6. Then the optimal solution $\Delta^\star$ is given by (4.15) 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 (C.3b) is always satisfied from the definition of the update rule (4.14c). Constraint (C.3c) is also always satisfied since no α_i is updated. Constraint (C.3d) 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 (C.3d) is always satisfied due to (C.3b) and the lower and upper bound of Δ can be simplified.

Using the update rule (4.14c), objective function (C.3a) can be rewritten as a quadratic function with respect to Δ

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

Finally, the optimal solution $\Delta^\star$ is given by (4.15). ■

Initialization

Theorem 4.10

Consider problem (4.17), 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

$$\sum_{j=1}^{K}\left(\beta^0_{[\tilde{n}-K+j]} + \max_{i=1,\dots,n_+} \alpha_i^0\right) \leq 0, \tag{4.18}$$

the optimal solution of (4.17) amounts to $\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0}$. In the opposite case, the following system of two equations

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

$$\sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu]}\left(\beta_j^0 + \lambda\right) - K\mu = 0, \tag{4.19b}$$

has a solution (λ, μ) with $\mu > 0$, and the optimal solution of (4.17) is equal to

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

Proof of Theorem 4.10 on page 45:
The Lagrangian of (4.17) reads

$$\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\beta}; \lambda, \boldsymbol{p}, \boldsymbol{q}, \boldsymbol{u}, \boldsymbol{v}) = \frac{1}{2}\left\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^0\right\|^2 + \frac{1}{2}\left\|\boldsymbol{\beta} - \boldsymbol{\beta}^0\right\|^2 + \lambda\left(\sum_{i=1}^{n_+}\alpha_i - \sum_{j=1}^{\tilde{n}}\beta_j\right)$$
$$- \sum_{i=1}^{n_+} p_i\alpha_i + \sum_{i=1}^{n_+} q_i(\alpha_i - C_1) - \sum_{j=1}^{\tilde{n}} u_j\beta_j + \sum_{j=1}^{\tilde{n}} v_j\left(\beta_j - \frac{1}{K}\sum_{i=1}^{n_+}\alpha_i\right).$$

The KKT conditions then amount to

$$\frac{\partial \mathcal{L}}{\partial \alpha_i} = \alpha_i - \alpha_i^0 + \lambda - p_i + q_i - \frac{1}{K}\sum_{j=1}^{\tilde{n}} v_j = 0, \qquad i = 1, 2, \dots, n_+, \tag{C.4a}$$

$$\frac{\partial \mathcal{L}(\cdot)}{\partial \beta_j} = \beta_j - \beta_j^0 - \lambda - u_j + v_j = 0, \qquad j = 1, 2, \dots, \tilde{n}, \tag{C.4b}$$

the primal feasibility conditions (4.17), 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

$$p_i \alpha_i = 0, \qquad i = 1, 2, \ldots, n_+, \tag{C.4c}$$
$$q_i(\alpha_i - C_1) = 0, \qquad i = 1, 2, \ldots, n_+, \tag{C.4d}$$
$$u_j \beta_j = 0, \qquad j = 1, 2, \ldots, \tilde{n}, \tag{C.4e}$$
$$v_j \left(\beta_j - \frac{1}{K} \sum_{i=1}^{n_+} \alpha_i \right) = 0, \qquad j = 1, 2, \ldots, \tilde{n}. \tag{C.4f}$$

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 (C.4d) implies $q_i = 0$ for all i and all complementarity conditions are satisfied. Moreover, optimality condition (C.4a) 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 (C.4b) 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

$$K\bar{v} - \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,+\infty)} \left(\beta_j^0 + \max_{i=1,\ldots,n_+} \alpha_i^0 + \bar{v} \right) \geq 0. \tag{C.5}$$

Denote by $\beta_{[j]}^0$ the sorted version of β_j^0. Then the function on the left-hand side of (C.5) as a function of $\bar{v}$ is increasing on $\left(-\infty, -\beta_{[n_+-K+1]}^0 - \max_i \alpha_i^0\right]$ and non-increasing otherwise. Thus, (C.5) can be satisfied if and only if its function value at $-\beta_{[n_+-K+1]}^0 - \max_i \alpha_i^0$ is non-negative

$$\begin{aligned}
&K\left(-\beta_{[n_+-K+1]}^0 - \max_{i=1,\ldots,n_+} \alpha_i^0 \right) - \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,+\infty)} \left(\beta_j^0 + \max_{i=1,\ldots,n_+} \alpha_i^0 - \beta_{[n_+-K+1]}^0 - \max_{i=1,\ldots,n_+} \alpha_i^0 \right) \\
&= K\left(-\beta_{[n_+-K+1]}^0 - \max_{i=1,\ldots,n_+} \alpha_i^0 \right) - \sum_{j=1}^{K} \left(\beta_{[n_+-K+j]}^0 - \beta_{[n_+-K+1]}^0 \right) = -\sum_{j=1}^{K} \left(\beta_{[n_+-K+j]}^0 + \max_{i=1,\ldots,n_+} \alpha_i^0 \right) \geq 0,
\end{aligned}$$

which is precisely condition (4.18).

Case 2: If (4.18) 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 (C.4a) 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 (C.4c, C.4d) to obtain

$$\alpha_i = \operatorname{clip}_{[0,\, C_1]}\big(\alpha_i^0 - \lambda + \bar{v}\big). \tag{C.6}$$

Similarly for any fixed j, we combine the optimality condition (C.4b) 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 (C.4e, C.4f) to obtain

$$\beta_j = \operatorname{clip}_{[0,\, \bar{\alpha}]}\big(\beta_j^0 + \lambda\big), \tag{C.7}$$

$$v_j = \operatorname{clip}_{[0,\, +\infty)}\big(\beta_j^0 + \lambda - \bar{\alpha}\big). \tag{C.8}$$

Summing equations (C.6), (C.7) and (C.8) respectively with respect to i and j results in

$$K\bar{\alpha} = \sum_{i=1}^{n_+} \operatorname{clip}_{[0,\, C_1]}\big(\alpha_i^0 - \lambda + \bar{v}\big), \tag{C.9a}$$

$$K\bar{\beta} = \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\, \bar{\alpha}]}\big(\beta_j^0 + \lambda\big), \tag{C.9b}$$

$$K\bar{v} = \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\, +\infty)}\big(\beta_j^0 + \lambda - \bar{\alpha}\big). \tag{C.9c}$$

We denote $\mu = \bar{\alpha}$. Then (4.19a) results by plugging (C.9c) into (C.9a) while (4.19b) follows from (C.9b) and $\sum_i \alpha_i = \sum_j \beta_j$. ■

Lemma 4.11

Even though $\lambda(\mu)$ is not unique, function h from (4.20) 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 4.11 on page 46:
Recall that based on (4.19b) we defined

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

and solutions of $g(\lambda; \mu) = 0$ for a fixed μ are denoted by $\lambda(\mu)$.

Let us first consider the case, when the solution to $g(\lambda) = 0$ is not unique. Since function $g(\cdot; \mu)$ is non-decreasing and K is an integer, it can happen only if the solution $\lambda(\mu)$ satisfies

$$\beta_{[j]}^0 + \lambda(\mu) \begin{cases} \ge \mu & \text{for } j = \tilde{n} - K + 1, \dots, \tilde{n}, \\ \le 0 & \text{otherwise.} \end{cases}$$

Here, we again denote $\beta^0_{[\cdot]}$ to be the sorted version of β^0_j. Then h defined in (4.20) 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 that h is independent of the choice of $\lambda(\mu)$.

In the previous paragraph, we prove, that h gives the same value for every choice of $\lambda(\mu)$. Now we need to show that h is a decreasing function for the arbitrary choice of $\lambda(\mu)$. Fix any $\mu_2 > \mu_1 > 0$. From (4.19b) we have

$$\sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu_1]}\left(\beta_j^0 + \lambda(\mu_1)\right) - K\mu_1 = 0, \tag{C.10}$$

$$\sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu_2]}\left(\beta_j^0 + \lambda(\mu_2)\right) - K\mu_2 = 0. \tag{C.11}$$

Equation (C.10) 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

$$\sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu_2]}\left(\beta_j^0 + \lambda(\mu_1)\right) \le \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu_1]}\left(\beta_j^0 + \lambda(\mu_1)\right) + K(\mu_2 - \mu_1) = K\mu_2, \tag{C.12}$$

where the equality follows from (C.10). Comparing (C.11) and (C.12) 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 (C.10) we have $|J| \ge K$. Moreover, the definition of J and (C.10) yields

$$\sum_{j\in J} \operatorname{clip}_{[0,\,\mu_1]}\left(\beta_j^0 + \lambda(\mu_1)\right) - K\mu_1 = \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\mu_1]}\left(\beta_j^0 + \lambda(\mu_1)\right) - K\mu_1 = 0. \tag{C.13}$$

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 (C.13) and finally, the last inequality follows from $|J| \ge K$. The chain above together with (C.11) 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 (4.20) is non-increasing which is precisely the lemma statement. ∎

C.2.2 Family of *Pat&Mat* Formulations

In this section, we derive a coordinate descent algorithm for solving dual formulation (4.8) for the family of *Pat&Mat* formulations. We follow the same approach as for *TopPushK* family in Section4.3.1, i.e. we use update rules (4.14). In this case, we must also consider the third primary variable δ. Then the dual formulation (4.8) 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

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

Since we assume one of the update rule (4.14), the constrain (4.8b) 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 (4.8) yields

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta},\delta}{\text{maximize}} \quad -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^\top \mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix} + \sum_{i=1}^{n_+}\alpha_i + \frac{1}{\vartheta}\sum_{j=1}^{\tilde{n}}\beta_j - \delta\tilde{n}\tau \tag{C.14a}$$

$$\text{subject to} \quad \sum_{i=1}^{n_+}\alpha_i = \sum_{j=1}^{\tilde{n}}\beta_j, \tag{C.14b}$$

$$0 \leq \alpha_i \leq C, \quad i = 1,2,\ldots,n_+, \tag{C.14c}$$

$$0 \leq \beta_j \leq \delta\vartheta, \quad j = 1,2,\ldots,\tilde{n}, \tag{C.14d}$$

$$\delta \geq 0. \tag{C.14e}$$

Since we know the form of the optimal solution (4.15), we only need to show how to compute Δ_{lb}, Δ_{ub} and γ for all update rules (4.14). 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 (C.14a) with respect to δ yields

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

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

$$\delta^* = \frac{1}{\vartheta}\max_{j\in\{1,2,\ldots,\tilde{n}\}}\beta_j. \tag{C.15}$$

The following three propositions provide closed-form formulas for all three update rules.

Proposition C.11: Update rule (4.14a) for problem (C.14)

Consider problem (C.14), update rule (4.14a), indices $1 \le k \le n_+$ and $1 \le l \le n_+$ and Notation 4.6. Then the optimal solution $\Delta^\star$ is given by (4.15) 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 (C.14b) is always satisfied from the definition of the update rule (4.14a). Constraint (C.14d) is also always satisfied since no β_j was updated and the sum of all α_i did not change. Constraint (C.14c) reads

$$0 \le \alpha_{\hat{k}} + \Delta \le C \implies -\alpha_{\hat{k}} \le \Delta \le C - \alpha_{\hat{k}}$$
$$0 \le \alpha_{\hat{l}} - \Delta \le C \implies \alpha_{\hat{l}} - C \le \Delta \le \alpha_{\hat{l}}$$

which gives the lower and upper bound of Δ.

Using the update rule (4.14a), objective function (C.14a) can be rewritten as a quadratic function with respect to Δ

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

The optimal solution $\Delta^\star$ is given by (4.15). Finally, since optimal δ is given by (C.15) and no β_j was updated, the optimal δ does not change. ■

Proposition C.12: Update rule (4.14b) for problem (C.14)

Consider problem (C.14), update rule (4.14b), indices $1 \le k \le n_+$ and $n_+ + 1 \le l \le \tilde{n}$ and Notation 4.6. Let us define

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

Then the bounds from (4.15) 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 (4.15) 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 (4.15) 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 (C.14b) is always satisfied from the definition of the update rule (4.14b). Constraint (C.14c) reads $-\alpha_{\hat{k}} \le \Delta \le C - \alpha_{\hat{k}}$. Using the definition of $\beta_{\max}$, constraint (C.14d)

reads $\beta_{\max} \leq \delta\vartheta$ and $0 \leq \beta_{\hat{l}} + \Delta \leq \delta\vartheta$. Since the optimal δ is given by (C.15), there are only two possible choices: $\delta_1^\star = \frac{\beta_{\max}}{\vartheta}$ and $\delta_2^\star = \frac{\beta_{\hat{l}}+\Delta}{\vartheta}$. If δ is feasible, all upper bounds in constraint (C.14d) 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 (4.14b), objective function (C.14a) can be rewritten as a quadratic function with respect to Δ as

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

   The optimal solution $\Delta_1^\star$ is given by (4.15) and is feasible if $\beta_{\hat{l}} + \Delta_1^\star \leq \beta_{\max}$.

2. Using $\delta_2^\star$ and the update rule (4.14b), objective function (C.14a) can be rewritten as a quadratic function with respect to Δ as

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

   The optimal solution $\Delta_2^\star$ is given by (4.15) and is feasible if $\beta_{\hat{l}} + \Delta_2^\star \geq \beta_{\max}$.

The optimal solution is the one, which maximizes the objective (C.14a) and is feasible. ■

Proposition C.13: Update rule (4.14c) for problem (C.14)

Consider problem (C.14), update rule (4.14c), indices $n_+ + 1 \leq k \leq \tilde{n}$ and $n_+ + 1 \leq l \leq \tilde{n}$ and Notation 4.6. Let us define

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

Then the bounds from (4.15) 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 (4.15) 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 (4.15) 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 (4.15) where

$$\gamma = -\frac{s_k - s_l - \frac{\tilde{n}\tau}{\vartheta}}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk}}, \qquad \delta_3^* = \frac{\beta_{\hat{l}} - \Delta_3^\star}{\vartheta}.$$

The optimal solution $\Delta^\star$ is equal to one of them, which maximizes the original objective and is feasible.

Proof:
Constraint (C.14b) is always satisfied from the definition of the update rule (4.14c). Constraint (C.14c) is also always satisfied since no α_i is updated. Using the definition of $\beta_{\max}$,

constraint (C.14d) 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 (C.15), there are only two possible choices

$$\delta_1^\star = \frac{\beta_{\max}}{\vartheta}, \qquad \delta_2^\star = \frac{\beta_{\hat{k}} + \Delta}{\vartheta}, \qquad \delta_3^\star = \frac{\beta_{\hat{l}} - \Delta}{\vartheta}. \tag{C.16}$$

If we use any of these choices which is feasible, all upper bounds in constraint (C.14d) 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 (C.16) and the update rule (4.14c), objective function (C.14a) can be rewritten as a quadratic function with respect to Δ as

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

   The optimal solution $\Delta_1^\star$ is given by (4.15) 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 (C.16) and the update rule (4.14c), objective function (C.14a) can be rewritten as a quadratic function with respect to Δ as

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

   The optimal solution $\Delta_2^\star$ is given by (4.15) 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 (C.16) and the update rule (4.14c), objective function (C.14a) can be rewritten as a quadratic function with respect to Δ as

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

   The optimal solution $\Delta_3^\star$ is given by (4.15) 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 (C.14a) and is feasible. ■

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 (4.8) yields

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta},\delta}{\text{maximize}} \quad -\frac{1}{2}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}^{\top}\mathbb{K}\begin{pmatrix}\boldsymbol{\alpha}\\ \boldsymbol{\beta}\end{pmatrix}+\sum_{i=1}^{n_+}\alpha_i-\frac{1}{4C}\sum_{i=1}^{n_+}\alpha_i^2 \tag{C.17a}$$

$$+\frac{1}{\vartheta}\sum_{j=1}^{\tilde{n}}\beta_j-\frac{1}{4\delta\vartheta^2}\sum_{j=1}^{\tilde{n}}\beta_j^2-\delta\tilde{n}\tau \tag{C.17b}$$

$$\text{subject to} \quad \sum_{i=1}^{n_+}\alpha_i=\sum_{j=1}^{\tilde{n}}\beta_j, \tag{C.17c}$$

$$\alpha_i \geq 0, \qquad i=1,2,\ldots,n_+, \tag{C.17d}$$

$$\beta_j \geq 0, \qquad j=1,2,\ldots,\tilde{n}, \tag{C.17e}$$

$$\delta \geq 0, \tag{C.17f}$$

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 (C.17a-C.17b) 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

$$\delta^*=\sqrt{\frac{1}{4\vartheta^2\tilde{n}\tau}\sum_{j=1}^{\tilde{n}}\beta_j^2}. \tag{C.18}$$

The following three propositions provide closed-form formulas for all three update rules.

Proposition C.14: Update rule (4.14a) for problem (C.17)

Consider problem (C.17), update rule (4.14a), indices $1 \leq k \leq n_+$ and $1 \leq l \leq n_+$ and Notation 4.6. Then the optimal solution $\Delta^\star$ is given by (4.15) 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:
Constraint (C.17c) is always satisfied from the definition of the update rule (4.14a). Constraint (C.17e) is also always satisfied since no β_j was updated. Constraint (C.17d) 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 (4.14a), objective function (C.17a-C.17b) can be rewritten as a quadratic function with respect to Δ

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

The optimal solution $\Delta^\star$ is given by (4.15). Finally, since optimal δ is given by (C.18) and no β_j was updated, the optimal δ does not change. ■

Proposition C.15: Update rule (4.14b) for problem (C.17)

Consider problem (C.17), update rule (4.14b), indices $1 \le k \le n_+$ and $n_+ + 1 \le l \le \tilde{n}$ and Notation 4.6. Then the optimal solution $\Delta^\star$ is given by (4.15) where

$$\begin{aligned}
\Delta_{lb} &= \max\{-\alpha_{\hat{k}}, -\beta_{\hat{l}}\},\\
\Delta_{ub} &= +\infty,\\
\gamma &= -\frac{s_k+s_l-1+\frac{\alpha_{\hat{k}}}{2C}-\frac{1}{\vartheta}+\frac{\beta_{\hat{l}}}{2\delta\vartheta^2}}{\mathbb{K}_{kk}+\mathbb{K}_{ll}+\mathbb{K}_{kl}+\mathbb{K}_{lk}+\frac{1}{2C}+\frac{1}{2\delta\vartheta^2}},\\
\delta^\star &= \sqrt{\delta^2+\frac{1}{4\vartheta^2\tilde{n}\tau}(\Delta^{\star 2}+2\Delta^\star\beta_{\hat{l}})}.
\end{aligned}$$

Proof:

Constraint (C.17c) is always satisfied from the definition of the update rule (4.14b). Constraints (C.17d) and (C.17e) reads

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

which gives the lower bound of Δ. In this case, Δ has no upper bound.

Using the update rule (4.14b), objective function (C.17a-C.17b) can be rewritten as a quadratic function with respect to Δ

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

The optimal solution $\Delta^\star$ is given by (4.15). We know that the optimal δ^* is given by (C.18), then

$$\delta^* = \sqrt{\frac{1}{4\vartheta^2\tilde{n}\tau}\left(\sum_{j\neq\hat{l}}\beta_j^2+(\beta_{\hat{l}}+\Delta^\star)^2\right)} = \sqrt{\delta^2+\frac{1}{4\vartheta^2\tilde{n}\tau}(\Delta^{\star 2}+2\Delta^\star\beta_{\hat{l}})}.$$

■

Proposition C.16: Update rule (4.14c) for problem (C.17)

Consider problem (C.17), update rule (4.14c) indices $n_+ + 1 \leq k \leq \tilde{n}$ and $n_+ + 1 \leq l \leq \tilde{n}$ and Notation 4.6. Then the optimal solution $\Delta^\star$ is given by (4.15) where

$$\begin{aligned}
\Delta_{lb} &= -\beta_{\hat{k}}, \\
\Delta_{ub} &= \beta_{\hat{l}}, \\
\gamma &= -\frac{s_k - s_l + \frac{1}{2\delta\vartheta^2}(\beta_{\hat{k}} - \beta_{\hat{l}})}{\mathbb{K}_{kk} + \mathbb{K}_{ll} - \mathbb{K}_{kl} - \mathbb{K}_{lk} + \frac{1}{\delta\vartheta^2}}, \\
\delta^\star &= \sqrt{\delta^2 + \frac{1}{2\vartheta^2\tilde{n}\tau}(\Delta^{\star 2} + \Delta^\star(\beta_{\hat{k}} - \beta_{\hat{l}}))}.
\end{aligned}$$

Proof:

Constraint (C.17c) is always satisfied from the definition of the update rule (4.14c). Constraint (C.17d) is also always satisfied since no α_i is updated. Constraint (C.17e) reads

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

which gives the lower and upper bound of Δ.

Using the update rule (4.14c), objective function (C.17a-C.17b) can be rewritten as a quadratic function with respect to Δ as

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

The optimal solution $\Delta^\star$ is given by (4.15). We know that the optimal δ^* is given by (C.18), then

$$\delta^* = \sqrt{\frac{1}{4\vartheta^2\tilde{n}\tau}\left(\sum_{j \notin \{\hat{l}, \hat{k}\}} \beta_j^2 + (\beta_{\hat{k}} + \Delta^\star)^2 + (\beta_{\hat{l}} - \Delta^\star)^2\right)} = \sqrt{\delta^2 + \frac{1}{2\vartheta^2\tilde{n}\tau}(\Delta^{\star 2} + \Delta^\star(\beta_{\hat{k}} - \beta_{\hat{l}}))}.$$

■

Initialization

As in the case of problem (4.5), all update rules (4.14) assume that the current solution $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, δ is feasible. So to create an iterative algorithm that solves problem (C.14) or (C.17), 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 $\boldsymbol{\alpha}^0$, $\boldsymbol{\beta}^0$, δ^0 to the feasible set of solutions

$$\begin{aligned}
\underset{\boldsymbol{\alpha}, \boldsymbol{\beta}, \delta}{\text{minimize}} \quad & \frac{1}{2}\left\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^0\right\|^2 + \frac{1}{2}\left\|\boldsymbol{\beta} - \boldsymbol{\beta}^0\right\|^2 + \frac{1}{2}(\delta - \delta^0)^2 \\
\text{subject to} \quad & \sum_{i=1}^{n_+} \alpha_i = \sum_{j=1}^{\tilde{n}} \beta_j, \\
& 0 \leq \alpha_i \leq C_1, \quad i = 1, 2, \ldots, n_+, \\
& 0 \leq \beta_j \leq C_2\delta, \quad j = 1, 2, \ldots, \tilde{n},, \\
& \delta \geq 0,
\end{aligned} \tag{C.19}$$

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

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

We show the way how to solve (C.19) only for hinge loss, since it is trivial to solve it for quadratic hinge. Again, we will follow the same approach as in [43] to solve this optimization problem. In the following theorem, we show that problem (C.19) 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 C.17

Consider problem (C.19) and some initial solution $\boldsymbol{\alpha}^0$, $\boldsymbol{\beta}^0$ and δ^0. Then if the following condition holds

$$\delta^0 \leq -C_2 \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,+\infty)}\left(\beta_j^0 + \max_{i=1,\ldots,n_+} \alpha_i^0\right). \tag{C.20}$$

the optimal solution of (C.19) amounts to $\boldsymbol{\alpha} = \boldsymbol{\beta} = \mathbf{0}$ and $\delta^0 = 0$. In the opposite case, the following system of two equations

$$0 = \sum_{i=1}^{n_+} \operatorname{clip}_{[0,\,C_1]}\left(\alpha_i^0 - \lambda\right) - \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,\lambda+\mu]}\left(\beta_j^0 + \lambda\right), \tag{C.21a}$$

$$\lambda = C_2\delta^0 + C_2^2 \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,+\infty)}\left(\beta_j^0 - \mu\right) - \mu. \tag{C.21b}$$

has a solution (λ, μ) with $\lambda + \mu > 0$ and the optimal solution of (C.19) 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:
The Lagrangian of (C.19) reads

$$\begin{aligned} \mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\beta}; \lambda, \boldsymbol{p}, \boldsymbol{q}, \boldsymbol{u}, \boldsymbol{v}) = \frac{1}{2}\left\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^0\right\|^2 + \frac{1}{2}\left\|\boldsymbol{\beta} - \boldsymbol{\beta}^0\right\|^2 + \frac{1}{2}(\delta - \delta^0)^2 + \lambda\left(\sum_{i=1}^{n_+} \alpha_i - \sum_{j=1}^{\tilde{n}} \beta_j\right) \\ - \sum_{i=1}^{n_+} p_i\alpha_i + \sum_{i=1}^{n_+} q_i(\alpha_i - C_1) - \sum_{j=1}^{\tilde{n}} u_j\beta_j + \sum_{j=1}^{\tilde{n}} v_j(\beta_j - C_2\delta). \end{aligned}$$

The KKT conditions then amount to the optimality conditions

$$\frac{\partial \mathcal{L}}{\partial \alpha_i} = \alpha_i - \alpha_i^0 + \lambda - p_i + q_i = 0, \qquad i = 1, 2, \dots, n_+, \tag{C.22a}$$

$$\frac{\partial \mathcal{L}(\cdot)}{\partial \beta_j} = \beta_j - \beta_j^0 - \lambda - u_j + v_j = 0, \qquad j = 1, 2, \dots, \tilde{n}, \tag{C.22b}$$

$$\frac{\partial \mathcal{L}(\cdot)}{\partial \delta} = \delta - \delta^0 - C_2 \sum_{j=1}^{\tilde{n}} v_j = 0, \tag{C.22c}$$

the primal feasibility conditions (C.19), 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

$$p_i \alpha_i = 0, \qquad i = 1, 2, \dots, n_+, \tag{C.22d}$$

$$q_i(\alpha_i - C_1) = 0, \qquad i = 1, 2, \dots, n_+, \tag{C.22e}$$

$$u_j \beta_j = 0, \qquad j = 1, 2, \dots, \tilde{n}, \tag{C.22f}$$

$$v_j\Big(\beta_j - C_2 \delta\Big) = 0, \qquad j = 1, 2, \dots, \tilde{n}. \tag{C.22g}$$

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 (C.22e) implies $q_i = 0$ and all complementarity conditions are satisfied. Moreover, (C.22a) 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 (C.22b) we deduce

$$v_j = \beta_j^0 + \lambda + u_j \geq \beta_j^0 + \lambda \geq \beta_j^0 + \max_{i=1,\dots,n_+} \alpha_i^0.$$

Since we also have the non-negativity constraint on v_j, this implies

$$v_j \geq \operatorname{clip}_{[0,\,+\infty)}\Big(\beta_j^0 + \max_{i=1,\dots,n_+} \alpha_i^0\Big).$$

Condition (C.22c) implies

$$\delta^0 = -C_2 \sum_{j=1}^{\tilde{n}} v_j \leq -C_2 \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\,+\infty)}\Big(\beta_j^0 + \max_{i=1,\dots,n_+} \alpha_i^0\Big),$$

which is precisely condition (C.20).

Case 2: If (C.20) 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 (C.22a) 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 (C.22d, C.22e) to obtain

$$\alpha_i = \operatorname{clip}_{[0,\,C_1]}\Big(\alpha_i^0 - \lambda\Big). \tag{C.23}$$

Similarly for any fixed j, we combine the optimality condition (C.22b) with the primal feasibility condition $0 \le \beta_j \le C_2\delta$, the dual feasibility conditions $u_j \ge 0$, $v_j \ge 0$ and the complementarity conditions (C.22f, C.22g) to obtain

$$\beta_j = \operatorname{clip}_{[0,\, C_2\delta]}\left(\beta_j^0 + \lambda\right), \tag{C.24}$$

$$v_j = \operatorname{clip}_{[0,\, +\infty)}\left(\beta_j^0 + \lambda - C_2\delta\right). \tag{C.25}$$

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 (C.23) and (C.24) into the feasibility condition $\sum_i \alpha_i = \sum_j \beta_j$ while the second equation follows from plugging (C.25) into (C.22c). Finally, system (C.21) follows after making the substitution $C_2\delta = \lambda + \mu$. ■

System (C.21) is relatively simple to solve, since equation (C.21b) provides an explicit formula for λ. Let us denote it as $\lambda(\mu)$, then we denote the right-hand side of (C.21a) as

$$h(\mu) := \sum_{i=1}^{n_+} \operatorname{clip}_{[0,\, C_1]}\left(\alpha_i^0 - \lambda(\mu)\right) - \sum_{j=1}^{\tilde{n}} \operatorname{clip}_{[0,\, \lambda(\mu)+\mu]}\left(\beta_j^0 + \lambda(\mu)\right). \tag{C.26}$$

Then the system of equations (C.21) 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 C.18

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

Proof of Lemma C.18 on page 119:
Consider any $\mu_1 < \mu_2$. Then from (C.21b) we obtain both $\lambda(\mu_1) \ge \lambda(\mu_2)$ and $\mu_1 + \lambda(\mu_1) \ge \mu_2 + \lambda(\mu_2)$. The statement then follows from the definition of h in (C.26). ■

D

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

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

Proof of Lemma 5.3 on page 53:
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 $j^\star_{\mathrm{mb}} = j^\star$ holds, then the sampled gradient equals to

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

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

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

Proof of Theorem 5.4 on page 53:
The law of total expectation implies

$$\mathbb{E}\nabla \hat{L}(\boldsymbol{w}) = \mathbb{P}\left[j^\star_{\mathrm{mb}} = j^\star\right]\mathbb{E}\left[\nabla \hat{L}(\boldsymbol{w}) \middle| j^\star_{\mathrm{mb}} = j^\star\right] + \mathbb{P}\left[j^\star_{\mathrm{mb}} \neq j^\star\right]\mathbb{E}\left[\nabla \hat{L}(\boldsymbol{w}) \middle| j^\star_{\mathrm{mb}} \neq j^\star\right],$$

from where the statement follows due to definition (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] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[49] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

[50] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7:1–30, 2006.

[51] Milton Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.

[52] Peter Bjorn Nemenyi. *Distribution-free multiple comparisons.* Princeton University, 1963.

[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] Mike Innes. Flux: Elegant machine learning with julia. *Journal of Open Source Software*, 2018.

[55] Michael Innes, Elliot Saba, Keno Fischer, Dhairya Gandhi, Marco Concetto Rudilosso, Neethu Mariya Joy, Tejan Karmali, Avik Pal, and Viral Shah. Fashionable modelling with flux. *CoRR*, abs/1811.01457, 2018.

[56] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[57] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[58] Alex Krizhevsky, Geoffrey Hinton, et al. *Learning multiple layers of features from tiny images*. Citeseer, 2009.

[59] 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.

[60] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[61] Tayana Morkel, Jan HP Eloff, and Martin S Olivier. An overview of image steganography. In *ISSA*, volume 1, pages 1–11, 2005.

[62] Joshua Silman. Steganography and steganalysis: an overview. *Sans Institute*, 3:61–76, 2001.

[63] Rémi Cogranne, Quentin Giboulot, and Patrick Bas. Steganography by minimizing statistical detectability: The cases of jpeg and color images. In *Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security*, pages 161–167, 2020.

[64] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[65] 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.

[66] Simon Mandlik, Matej Racinsky, Viliam Lisy, and Tomas Pevny. Mill. jl and jsongrinder. jl: automated differentiable feature extraction for learning from raw json data. *arXiv preprint arXiv:2105.09107*, 2021.

[67] Wlodzimierz Ogryczak and Arie Tamir. Minimizing the sum of the k largest functions in linear time. *Information Processing Letters*, 85(3):117–122, 2003.