Kernelized Online Imbalanced Learning

HU, Junjie

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong
June 2015

Abstract of thesis entitled:

Kernelized Online Imbalanced Learning

Submitted by HU, Junjie

for the degree of Master of Philosophy

at The Chinese University of Hong Kong in June 2015

Imbalanced streaming data are prevalent in many real-world applications because people usually are attracted by rare events. These data exhibit the characteristics of huge volume, high velocity, extreme imbalance, and perhaps, nonlinearity and heterogeneity. Learning binary classification models from imbalanced data, where the number of samples from one class is significantly larger than that from the other class, is an important research topic in machine learning and data mining. To learn from imbalanced data, online AUC (area under the ROC curve) maximization is a promising tool. However, it is not well suited for handling nonlinearity and heterogeneity of the data.

This thesis is mainly described in two parts. In the first part, motivated by the effectiveness of kernel methods, we propose a kernelized online imbalanced learning (KOIL) algorithm to produce a nonlinear classifier for imbalanced data. We consequently formulate the problem by maximizing the AUC score while minimizing the functional regularizer. We address two major challenges: 1) How to control the number of support vectors without sacrificing model performance, and 2) how to control the fluctuation of the learned decision function to achieve smooth updating. To this end, we introduce two buffers with fixed budgets (buffer sizes) for the positive and negative class, respectively, to store the corresponding learned support vectors. The buffers allow us to capture the global information of the decision boundary. To control the update fluctuation, we confine the weight of a new support vector to be influenced only by its k-nearest opposite support vectors. More importantly, we design a sophisticated scheme to compensate for the loss of information after replacement is conducted when either buffer is full. With such a compensation scheme, the learned model can be shown to approach a nonlinear classifier that is learned with infinite budgets. Theoretical results for two convex surrogates of the AUC metric, namely the pairwise hinge loss and its smooth variant, are derived to rigorously justify the model performance. We further conduct a series of experiments on both synthetic and real-world benchmark datasets to demonstrate the effectiveness of the proposed approach.

In the second part, We exploit the online Multiple Kernel Learning (MKL) framework to automatically determine good kernels for accurate data similarity representation. That is, we try to learn multiple kernel classifiers from a pool of predefined kernels and their linear combination coefficients in an online mode. The empirical evaluation shows that online MKL is effective for determining the kernel representation.

KOIL with a single kernel in the first part is computationally efficient in the online learning process, while KOIL with multiple kernels in the second part can be used to effectively select good kernels when prior knowledge on the kernel representation is unknown.

摘要

不平衡流式數據在很多實際應用中很普遍，因為人們經常對稀有的事件比較感興趣。這些數據呈現出大容量，高速度，極度不平衡，非線性以及異構性等特點。在機器學習及數據挖掘領域，根據不平衡數據（一個類別的樣本數量遠大於另一個類別的樣本數量）學習二值分類器是一個重要的研究問題。為了學習這些數據，在線最大化 AUC（ROC 曲線下的面積）是一種有效的工具。然而，這方法卻不適用於處理數據的非線性及異構性的問題。

這篇論文主要描述了兩部分。第一部分，基於核方法的有效性，我們提出了基於核函數的在線不平衡學習 (KOIL）算法，用於學習一個非線性的分類器。我們將問題明確地表達為最大化 AUC 得分同時最小化決策函數的正則項。我們解決了以下兩個主要挑戰：1）如何控制支持向量的數量卻不犧牲模型的效果，以及 2）如何控制學習的決策函數的波動性從而實現平穩的更新。因此，我們使用了兩個固定大小的緩存區，用於存儲已經學習到的正負類支持向量。這兩個緩存區使得我們可以獲得決策函數的全局信息。為了控制更新的波動性，我們限定了每個新的支持向量的權重只會受到最靠近它的 k 個相反類別的支持向量的影響。更重要的是，我們設計了一種補償因緩

存區滿了之後更新支持向量帶來的信息損失的有效方法。通過這種補償方法，學習到的模型能近似于通過使用著無限空間的緩存區來學習到的非線性分類器。針對兩種替代 AUC 度量的凸函數，即成偶铰链損失及平滑的成偶铰链損失，理論結果嚴格證明了模型的性能。我們還進一步在實際數據和模擬數據中做了一系列實驗，以驗證提出的模型的有效性。

在第二部分，我們利用了在線多核函數學習的框架，用於自動決定好的核函數，從而獲得準確的數據相似性表達。也即是，我們通過在線學習模式，嘗試在一些預先定義好的核函數中學習多個核分類器以及這些分類器權重的線性組合。實驗效果顯示我們提出的在線多核函數學習方法能夠有效地確定出好的核表達。

在第一部分的 KOIL 算法在實時在線學習中高效地利用計算資源，然而當核函數的先驗知識不知道的情況下，第二部分提出的多核 KOIL 算法能有效地挑選出好的核函數表達。

Acknowledgement

I would like to thank my supervisors Prof. Irwin King and Prof. Michael R. Lyu for their guidance and support of my graduate study at the Chinese University of Hong Kong. My motivation to continue my further study comes partially from my passion for machine learning research, and partially from the influence my supervisors have had on my development. Prof. King's efforts in educational projects, e.g.,Veriguide, encouraged me to conduct interesting research that can serve our community. Prof. Lyu's dedicated attitude towards teaching deeply impressed me and set the example for me to be a responsible teacher when I was the tutor of his course. As a researcher I would like to devote all my passion towards my research, and as a teaching assistant I would like to altruistically give to others what my teachers have so generously given me. The time I spend on my study at CUHK would be one of the most important periods in my life. I also want to give my sincerest thanks to my friends Dr. Haiqin Yang and Yuxin Su for their helps to my research at CUHK. I am also grateful to my graduation committee members, Prof. Kevin Yip, Prof. Laiwan

Chan and Prof. Qin Lu, for their revision suggestions on my thesis and their efforts in evaluating my work at CUHK.

Finally, I would like to express my heartfelt thanks to my family for their supports and love ever since I was born. Without their support and love, I would not be the person that I am today. This thesis is dedicated to all of them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Summary

In this chapter, we investigate the properties of imbalanced streaming data, and describe the challenges and motivations of learning imbalanced streaming data, and finally highlight the contributions of the proposed framework Kernelized Online Imbalanced Learning, which is described in details in Chapter 3 and Chapter 4.

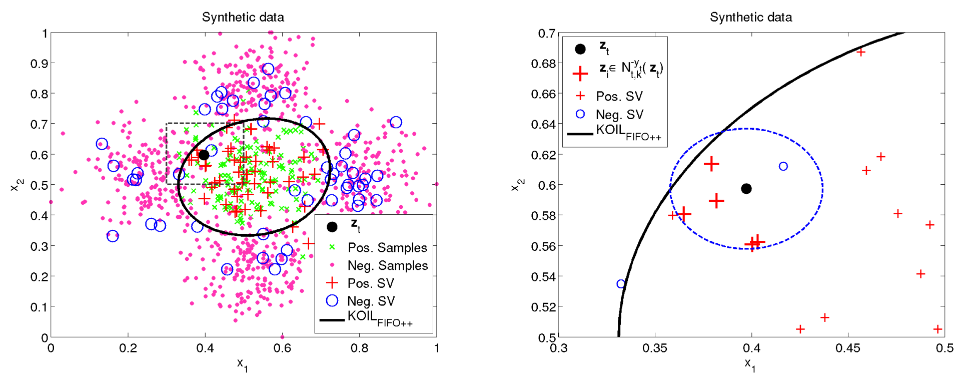
Imbalanced streaming data are prevalent in various real-world applications, such as network intrusion detection [44], purchasing or clicking analysis for customer relationship [12, 18]. These data exhibit the following prominent characteristics:

1) Huge volume: The volume of data increases tremendously, from Petabyte to Exabyte, or even Zettabyte. For example, the number of webpages indexed by Google is nearly 1 million

in 1998, 1 billion in 2000, and more than 1 trillion in 2008 [14].

2) High velocity: They are streaming data, generated in seconds or microseconds, from various online applications. The data may change dynamically.
3) Extreme imbalance: The imbalanced ratio can be $100 : 1$, or even $10,000 : 1$ for a standard binary classification task, where the important class is very rare. For example, people usually pay more attention on the detection of noisy signals out of a huge number of normal signals.
4) Nonlinearity and heterogeneity: In general, only nonlinear classifiers can produce a more accurate decision boundary for data with low dimensionality; see Fig. 1.1(a) for an example. The heterogeneity among different types of data, e.g., medical data and text data, poses difficulty in designing a general algorithm and defining data similarity.

Learning binary classification models from imbalanced data has become an important research topic in both machine learning and data mining [3, 32, 48, 54]. In the literature, researchers aim at maximizing the area under the ROC curve (AUC) because it provides an effective metric to measure the performance of classifiers for imbalanced data [1, 2, 19, 21, 25]. A detailed introduction of AUC and the comparision between AUC and other metrics are given in [15]. Although area under the precision-recall curve is also a good metric in evaluating the performance of a classifier on imbalanced data in a batch training procedure, the ROC curve has the advantage of visualing



(a) (b)

Figure 1.1: Figure 1.1(a) shows the decision function in a black solid curve, the new instance in a big •, the positive samples in small ×'s, the negative samples in small •'s, the positive support vectors in big +'s and the negative support vectors in big ◦'s. Figure 1.1(b) zooms into the local region of a new instance $\mathbf{z}_t$ and shows how its influence is being controlled.

and judging a classifier's performance irrespective of dynamic changes of class distributions and different misclassification costs. This ability is conductive to investigating learning skewed classes and cost-sensitive learning [15]. Hence, the AUC can reveal a stabler performance of a classifier in a dynamic online enviroment than the area under precision-recall curve. Moreover, the AUC holds a significant statistical property over other metrics. The AUC of a classifier essentially equals to the probability that a classifier ranks a randomly selected positive sample higher than a randomly selected negative sample. In terms of imbalanced streaming data, researchers have proposed the tools of online AUC maximization [17, 60]. However, these algorithms only produce a linear classifier and are not well suited for handling the nonlinearity and heterogeneity of the data.

By contrast, we focus on seeking an online nonlinear classifier with kernels – a less explored but important research topic in both theory and applications. This gives rise to two significant challenges. First, the learned kernel-based estimator becomes more complex as the number of samples increases [29, 57]. Without a suitable stream oblivious strategy which is designed to remove out-dated support vectors, in the extreme case, the number of learned support vectors can tend to infinity. This is undesirable for large-scale applications. Although in the literature, refinement techniques, e.g., Projectron [35], and online learning algorithms with fixed budgets, such as randomized budget perceptron [4] and Forgetron [10], have been proposed, it is non-trival to tackle online imbalanced learning since the stored support vectors of both classes may also be imbalanced. Second, fluctuation due to outliers is unavoidable in online learning [6, 27, 38]. Thus, additional effort is required to achieve smooth updating. Third, the kernel representation is effective for capturing the nonlinearity and heterogeneity of the data [29, 52]. However, how to effectively determine kernels for good performance is still a challenging issue.

To tackle the above challenges, we propose a Kernelized Online Imbalanced Learning algorithm with fixed budgets, or *KOIL* in short, to achieve online nonlinear AUC maximization. We highlight the contributions of this article in the followings:

1. We seek the fixed-budget kernel representation of KOIL by

maximizing the localized AUC metric with the appearance of new instances. That is, we maintain two buffers with the same fixed buffer size to store the most informative data from each class as learned support vectors. This fixed-budget strategy is important for handling imbalanced streaming data and sets it apart from Projectron [35], which may include too many support vectors.

2. We update the weights of the new and old support vectors in KOIL by confining the influence of a new instance to its k-nearest opposite support vectors; see Fig. 1.1(b) for an example. This leads to smooth updating and makes KOIL different from previously proposed online AUC maximization algorithms [17, 60], which update the weight of a new instance based on all the information stored in the buffers.
3. Other than the standard stream oblivious policies, such as First-In-First-Out (FIFO) and reservoir sampling (RS), which replace a support vector when either buffer is full, we design a sophisticated scheme to compensate for the loss when a support vector is removed; see Fig. 1.2 for an illustration of the idea. Different from other online learning algorithms with fixed budgets [4, 10], which tend to discard information during training, our proposed compensation scheme indeed can avoid information loss. The empirical results show that after compensation, the learned decision function by KOIL approaches the one learned with infinite

budgets.

4. Without the ideal assumption that proper kernels are given prior to training, we exploit online Multiple Kernel Learning (MKL) framework to automatically determine good kernels for accurate data similarity representation. That is, we try to learn multiple kernel classifiers from a pool of predefined kernels and their linear combination coefficients in an online mode. Different from existing online MKL algorithms [24], our KOIL focuses on the pairwise loss function and discounts the weights of multiple kernel classifiers when there are errors. The empirical evaluation shows that online MKL is effective for determining the kernel representation.

In Figure 1.1, we give an illustration of KOIL with k-nearest neighbor confinement on a synthetic data in 2-D space. Figure 1.1(a) shows that the decision function learned by our proposed KOIL with the extended FIFO updating policy can classify the data well. Figure 1.1(b) zooms into the local region of a new instance $\mathbf{z}_t$ of the negative class and shows how its influence is being controlled. Here, it can only affect its k-nearest opposite support vectors (big +'s), where $k = 5$. Obviously, restricting the influence of the new instance to a local region is safe since it will not affect those positive support vectors that are far away from it. Moreover, this restriction prevents the decision function from deteriorating by outliers which not only affect support vectors of the opposite class in the whole buffer but also be assigned to a large initial weight. Figure 1.2 shows the compensation scheme in the Reproducing Kernel Hilbert Space. By the two assumptions $k(\mathbf{x}, \mathbf{x}) \le X^2$ and $k(\mathbf{x}_r, \mathbf{x}_c) \ge \xi_2^2$, we have $\|\phi(\mathbf{x}_c)\|_{\mathcal{H}} \cos\theta \ge \frac{\xi_2^2}{X}$, where $\phi(\mathbf{x}_c) = k(\mathbf{x}_c, \cdot)$.

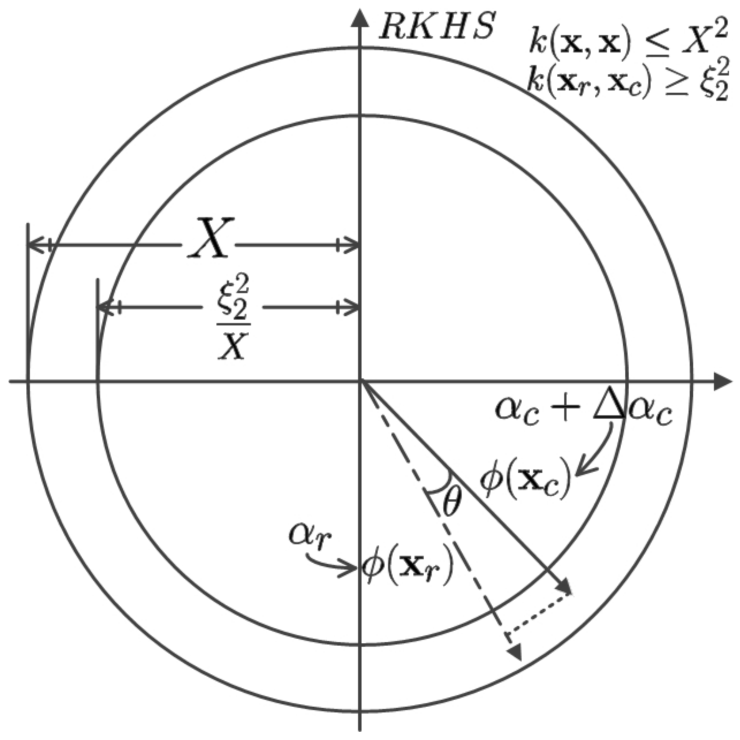

$RKHS$
$k(\mathbf{x}, \mathbf{x}) \le X^2$
$k(\mathbf{x}_r, \mathbf{x}_c) \ge \xi_2^2$
X
$\frac{\xi_2^2}{X}$
$\alpha_c + \Delta\alpha_c$
$\phi(\mathbf{x}_c)$
θ
α_r
$\phi(\mathbf{x}_r)$

Figure 1.2: Figure 1.2 shows the removed support vector $\mathbf{x}_r$ in the dotted arrow, the compensated support vector $\mathbf{x}_c$ in the solid arrow, and the angle θ between them.

□ **End of chapter.**

Chapter 2

Preliminaries

Summary

In this chapter, We review some prior work in closely related areas: machine learning from imbalanced data, online learning, and multiple kernel learning. Then we introduce some notations used throughout this thesis, and define the problem discussed in this thesis.

2.1 Background Study

Learning from Imbalanced Data

Learning from imbalanced data has become an important task in machine learning and data mining [3, 32, 48]. Some algorithms have been developed to train classifiers by maximizing the AUC metric, such as Wilcoxon-Mann-Whitney statistic optimization [51] and RankOpt [21]. Some investigations extend

SVM to optimize the AUC metric [2]. A general framework for optimizing multivariate nonlinear performance measures, such as AUC and F1, is proposed in [25]. Cost-sensitive multilayer Perceptron is also proposed to improve the discrimination ability of multilayer perceptron (MLP) [3]. One major weakness of these methods is that they train the model in batch-mode, which is inefficient when new training samples appear dynamically.

Online Learning

Online learning algorithms are significant as they can adaptively update the models based on new training samples. The oldest and most well-known online learning algorithm is the Perceptron [37]. Many variants have been proposed in the literature [5, 16]. Some are inspired by the maximum margin principle [8, 31, 59]. To learn from imbalanced data, algorithms for online AUC maximization are proposed in [11, 17, 60]. Several pieces of theoretical work are also published to derive generalization error bounds of online learning algorithms for pairwise loss functions [26, 47]. However, these algorithms only focus on linear classifiers, which are not sufficient to capture the heterogeneity and nonlinearity embedded in the data [55, 56]. In the literature, kernel-based online learning algorithms, such as online learning algorithms in a Reproducing Kernel Hilbert Space [10, 29, 35, 42], online Gaussian Process [9, 20, 28, 41], and kernelized recursive least-square algorithms [13, 45], have been proposed. However, a key challenging issue in online learning with kernels is that the computation complexity scales with the number of training samples. Hence, strategies such as Forgetron [10], Randomized Replacement [4], and projection scheme [9, 28, 35, 58] have been proposed. However, these strategies aim at directly maximizing the accuracy, and it is known that it is inappropriate to evaluate the model performance on imbalanced data by the accuracy. For example, randomly labeling all samples on a imbalanced dataset as negative results in a high accuracy but a high misclassication cost of positive samples.

Multiple Kernel Learning

The Multiple kernel Learning (MKL) framework is a well-known and effective tool for kernel learning. It aims to seek the combination of multiple kernels by optimizing the performance of kernel based learning methods (e.g., Support Vector Machine) [36, 43]. Subsequently, MKL with different norm regularizers are proposed to attain good model performance [30, 50, 56]. Recently, online MKL (OMKL) is proposed to simultaneously learn multiple kernel classifiers and their linear combinations from a pool of predefined kernels in an online mode [22, 24]. Similar ideas are applied to solve the problems in image search and regression [39, 49]. However, the existing algorithms do not consider the task of imbalanced learning.

In summary, the previously proposed algorithms cannot handle nonlinearity and heterogeneity in the imbalanced streaming data well. This motivates us to seek for a nonlinear classifier for imbalanced classification in online training mode.

2.2 Notations and Problem Definition

We first introduce the notations that will be used throughout the thesis. Bold-faced small letters, e.g., $\mathbf{x}$, denote vectors. Letters in calligraphic font, e.g., $\mathcal{X}$, indicate sets. We use $\mathbb{R}^d$ to denote the d-dimensional Euclidean space and $\mathcal{H}$ to denote a Hilbert space. The inner product of $\mathbf{x}$ and $\mathbf{y}$ on $\mathcal{H}$ is denoted by $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{H}}$.

We focus on the imbalanced binary classification problem and aim to learn a nonlinear decision function $f : \mathbb{R}^d \to \mathbb{R}$ from a sequence of feature-labeled pair instances $\{\mathbf{z}_t = (\mathbf{x}_t, y_t) \in \mathcal{Z}, t \in [T]\}$, where $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, $\mathbf{x}_t \in \mathcal{X} \subseteq \mathbb{R}^d$, $y_t \in \mathcal{Y} = \{-1, +1\}$, and $[T] = \{1, \ldots, T\}$. Without loss of generality, we assume that the positive class is the minority class while the negative class is the majority class. We denote by $N_{t,k}^{\tilde{y}}(\mathbf{z})$ the set of feature-labeled pair instances that are the k-nearest neighbors of $\mathbf{z}$ in either one of the positive or negative support vector buffer and have the label of $\tilde{y}$ at the t-th trial. Here, the neighborhood is defined by the distance or the similarity between two instances, i.e., the smaller the distance between or the more similar the instances, the closer the neighbors. Besides, we define the index sets I_t^+ and I_t^- to record the indices of positive and negative support vectors at the t-th trial. Similarly, we define the index sets I^+ and I^- to record the indices of positive and negative samples in the whole data stream. Moreover, for simplicity, we define two buffers $\mathcal{K}_t^+$ and $\mathcal{K}_t^-$ to store the learned information, weight and support vector, from the two classes at the t-th trial, respectively:

$$\mathcal{K}_t^+.\mathcal{A} = \{\alpha_{i,t}^+ \,|\, \alpha_{i,t}^+ \neq 0, i \in I_t^+\}, \quad \mathcal{K}_t^+.\mathcal{B} = \{\mathbf{z}_i \,|\, y_i = +1, i \in I_t^+\},$$
$$\mathcal{K}_t^-.\mathcal{A} = \{\alpha_{i,t}^- \,|\, \alpha_{i,t}^- \neq 0, i \in I_t^-\}, \quad \mathcal{K}_t^-.\mathcal{B} = \{\mathbf{z}_i \,|\, y_i = -1, i \in I_t^-\}.$$

Here, $\alpha_{i,t}$ denotes the weight of the support vector that first occurred in the i-th trial and updated at the t-th trial. To the purpose of rebalancing the number of positive and negative support vectors, we fix the budgets (the buffer sizes) to be the same, i.e., $|I_t^+| = |I_t^-| = N$ for all t.

At the t-th trial, our proposed algorithm KOIL computes a decision function f_t of the form

$$f_t(\mathbf{x}) = \sum\nolimits_{i\in I_t^+} \alpha_{i,t}^+ k(\mathbf{x}_i, \mathbf{x}) + \sum\nolimits_{j\in I_t^-} \alpha_{j,t}^- k(\mathbf{x}_j, \mathbf{x}), \qquad (2.1)$$

where $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a predefined kernel [29]. The corresponding weights and support vectors are stored in $\mathcal{K}_t^+$ and $\mathcal{K}_t^-$, respectively.

The prediction of a new sample $\mathbf{x}$ can be made by $\text{sgn}(f_t(\mathbf{x}))$, where $\text{sgn}(\dot{})$ is a function that outputs the sign of a real number. More generally, $f_t(\mathbf{x})$ is an element of a Reproducing Kernel

Hilbert Space and can be expressed as $f_t(\mathbf{x}) = \langle f_t(\cdot), k(\mathbf{x}, \cdot)\rangle_{\mathcal{H}}$ to capture the nonlinearity and heterogeneity of the data [40]. In the following, we will motivate and describe our strategy of updating f_t.

□ **End of chapter.**

Chapter 3

KOIL with A Single Kernel

Summary

In this chapter, we describe the detailed design of Kernelized Online Imbalanced Learning algorithm with fixed budgets which achieves AUC maximization.

3.1 Non-smooth Pairwise Hinge Loss

Given the positive dataset $\mathcal{D}^+ = \{\mathbf{z}_i | y_i = +1, i \in I^+\}$ and the negative dataset $\mathcal{D}^- = \{\mathbf{z}_j | y_j = -1, j \in I^-\}$, the AUC metric for a kernel representation function f is calculated by

$$\begin{aligned} AUC(f) &= \frac{\sum_{i \in I^+} \sum_{j \in I^-} \mathbb{I}[f(\mathbf{x}_i) - f(\mathbf{x}_j) > 0]}{|I^+||I^-|} \\ &= 1 - \frac{\sum_{i \in I^+} \sum_{j \in I^-} \mathbb{I}[f(\mathbf{x}_i) - f(\mathbf{x}_j) \leq 0]}{|I^+||I^-|}, \end{aligned} \tag{3.1}$$

where $\mathbb{I}[\pi]$ is the indicator function that equals 1 when π is true and 0 otherwise. Hence, maximizing $AUC(f)$ is equivalent to minimizing $\sum_{i\in I^+}\sum_{j\in I^-}\mathbb{I}[f(\mathbf{x}_i)-f(\mathbf{x}_j)\le 0]$. Since directly maximizing AUC score results in an NP-hard combinatorial optimization problem [7], the indicator function is usually replaced by a convex surrogate, such as the following pairwise hinge loss function [17, 60]:

$$\ell_h(f,\mathbf{z},\mathbf{z}')=\frac{|y-y'|}{2}\left[1-\frac{1}{2}(y-y')(f(\mathbf{x})-f(\mathbf{x}'))\right]_+, \quad (3.2)$$

where $[v]_+=\max\{0,v\}$.

This suggests that we can find the decision function in Eq. (2.1) for AUC maximization by minimizing

$$\mathcal{L}(f)=\frac{1}{2}\|f\|^2_{\mathcal{H}}+C\sum_{i\in I^+}\sum_{j\in I^-}\ell_h(f,\mathbf{z}_i,\mathbf{z}_j), \quad (3.3)$$

where $\frac{1}{2}\|f\|^2_{\mathcal{H}}$ is the regularization term controlling the functional complexity and $C>0$ is a penalty parameter balancing the functional complexity and training errors.

3.2 Online AUC Maximization by KOIL

Following the derivation in [23], we aim to update the kernel decision function based on the arrival of a new instance $\mathbf{z}_t$ by minimizing the following *localized instantaneous regularized risk of AUC* associated with $\mathbf{z}_t$:

$$\hat{\mathcal{L}}_t(f):=\hat{\mathcal{L}}(f,\mathbf{z}_t)=\frac{1}{2}\|f\|^2_{\mathcal{H}}+C\sum_{\mathbf{z}_i\in N_k^{-y_t}(\mathbf{z}_t)}\ell_h(f,\mathbf{z}_t,\mathbf{z}_i). \quad (3.4)$$

where k is a predefined constant and is usually set to a small value, e.g., around 10% of the budget. The points $\mathbf{z}_i$ are stored in either buffer, i.e., $\mathcal{K}^+.\mathcal{B}$ or $\mathcal{K}^-.\mathcal{B}$. They are the k-nearest opposite support vectors in the buffer to the new instance $\mathbf{z}_t$, i.e., the k-nearest support vectors with the opposite label of $\mathbf{z}_t$.

There are some remarks about the above formulation:

- Different from NORMA [29], whose risk only measures the predictive error of the new instance, the risk defined in Eq. (3.4) involves pairwise losses between $\mathbf{z}_t$ and its k-nearest opposite support vectors in the buffer. This can resolve the scalability issue of online kernelized learning [53].
- This setting brings the following advantages: 1) maintaining two buffers with relatively large budgets can keep track of the global information of the decision boundary; 2) only considering the k-nearest opposite support vectors of the new instance allows us to utilize the local information around the new instance and avoid the fluctuation of the decision function.

We show the KOIL framework in Algorithm 1, which consists of two key components: UpdateKernel of Algorithm 2 and UpdateBuffer of Algorithm 3.

Update Kernels

We apply the stochastic gradient descent method to update the decision function at the t-th trial as follows:

$$f_{t+1} := f_t - \eta \partial_f \hat{\mathcal{L}}_t(f)|_{f=f_t}, \tag{3.5}$$

Algorithm 1 Kernelized Online Imbalanced Learning (KOIL) with Fixed Budgets

```
1: Input:
     • the penalty parameter $C$ and the learning rate $\eta$
     • the maximum positive budget $N^+$ and negative budget $N^-$
     • the number of nearest neighbors $k$
2: Initialize $\mathcal{K}^+.\mathcal{A} = \mathcal{K}^-.\mathcal{A} = \emptyset$, $\mathcal{K}^+.\mathcal{B} = \mathcal{K}^-.\mathcal{B} = \emptyset$, $N_p = N_n = 0$
3: for $t = 1$ to $T$ do
4:   Receive a training sample $\mathbf{z}_t = (\mathbf{x}_t, y_t)$
5:   if $y_t == +1$ then
6:     $N_p = N_p + 1$
7:     $[\mathcal{K}^-, \mathcal{K}^+, \alpha] = \text{UpdateKernel}(\mathbf{z}_t, \mathcal{K}^-, \mathcal{K}^+, C, \eta, k)$
8:     $\mathcal{K}^+ = \text{UpdateBuffer}(\alpha, \mathbf{z}_t, \mathcal{K}^+, k, N^+, N_p)$
9:   else
10:    $N_n = N_n + 1$
11:    $[\mathcal{K}^+, \mathcal{K}^-, \alpha] = \text{UpdateKernel}(\mathbf{z}_t, \mathcal{K}^+, \mathcal{K}^-, C, \eta, k)$
12:    $\mathcal{K}^- = \text{UpdateBuffer}(\alpha, \mathbf{z}_t, \mathcal{K}^-, k, N^-, N_n)$
13:  end if
14: end for
```

where ∂_f is shorthand for $\partial/\partial f$ (the gradient with respect to f), and $\eta > 0$ is the learning rate which can either be a constant or have a value that decreases with the number of trials.

To compute the gradient of $\hat{\mathcal{L}}_t(f)$ with respect to f, we first calculate the gradient of ℓ_h with respect to f, i.e., $\partial_f \ell_h(f, \mathbf{z}_t, \mathbf{z}_i)$, by

$$\partial_f \ell_h(\cdot) = \begin{cases} 0, & \ell_h(f, \mathbf{z}_t, \mathbf{z}_i) \leq 0, \\ -\varphi(\mathbf{z}_t, \mathbf{z}_i), & \ell_h(f, \mathbf{z}_t, \mathbf{z}_i) > 0, \end{cases} \tag{3.6}$$

where $\varphi(\mathbf{z}_t, \mathbf{z}_i) = y_t(k(\mathbf{x}_t, \cdot) - k(\mathbf{x}_i, \cdot))$.

Algorithm 2 UpdateKernel

```
1: Input:
     • the newly received sample with label $\mathbf{z}_t$,
     • $\mathcal{K}$ and $\mathcal{K}'$ for support vectors with the opposite and the same label
       to $\mathbf{z}_t$ respectively,
     • the penalty parameter $C$, the learning rate $\eta$, and the number of
       the nearest neighbors $k$.
2: Output: the updated $\mathcal{K}$, $\mathcal{K}'$ and the weight $\alpha_t$ for $\mathbf{z}_t$
3: Initialize: $V_t = \emptyset$, compute $f_t$ by Eq. (2.1)
4: for $i \in I_t^{-y_t}$ do
5:   if $1 > y_t(f_t(\mathbf{x}_t) - f_t(\mathbf{x}_i))$ then
6:     $V_t = V_t \cup \{i\}$
7:   end if
8: end for
9: if $|V_t| > k$ then
10:   $Sim(i) = k(\mathbf{x}_t, \mathbf{x}_i), \quad \forall\, i \in V_t$
11:   $[Sim', idx] = \text{Sort}(Sim, \text{`descend'})$
12:   $idx_k = idx(1:k)$
13:   $V_t = V_t(idx_k)$
14: end if
15: Update $\alpha_{i,t}$ by Eq. (3.10)
16: return $\mathcal{K}, \mathcal{K}', \alpha_{t,t}$
```

Using Eq. (3.4) and Eq. (3.6), we obtain

$$\partial_f \hat{\mathcal{L}}_t(f_t) = f_t - C \sum_{\mathbf{z}_i \in N_k^{-y_t}(\mathbf{z}_t)} \mathbb{I}[\ell_h(f, \mathbf{z}_t, \mathbf{z}_i) > 0] \varphi(\mathbf{z}_t, \mathbf{z}_i). \quad (3.7)$$

Practically, we initialize the first decision function to zero, i.e., $f_1 = 0$, and express the decision function at the t-th trial as a kernel expansion defined in Eq.(2.1). We then update the

$(t+1)$-th trial in incremental mode:

$$f_{t+1}(\mathbf{x}) = f_t(\mathbf{x}) + \alpha_{t,t} k(\mathbf{x}_t, \mathbf{x}). \tag{3.8}$$

For simplicity, we define V_t to be the set of indices satisfying the indicator function in Eq. (3.7) (the valid set) and $\bar{V}_t$ to be its complementary set, i.e.

$$\begin{aligned} V_t &:= \{i \in I_t^{-y_t} \mid \mathbf{z}_i \in N_k^{-y_t}(\mathbf{z}_t) \wedge \ell_h(f, \mathbf{z}_t, \mathbf{z}_i) > 0\}, \\ \overline{V_t} &:= I_t^{-y_t} \setminus V_t. \end{aligned} \tag{3.9}$$

Then, the corresponding updating rule for the kernel weights at the t-th trial is given by

$$\alpha_{i,t} = \begin{cases} \eta C y_t |V_t|, & i = t, \\ (1-\eta)\alpha_{i,t-1} - \eta C y_t, & \forall i \in V_t, \\ (1-\eta)\alpha_{i,t-1}, & \forall i \in I_t^{y_t} \cup \overline{V_t}. \end{cases} \tag{3.10}$$

The updating rule in Eq. (3.10) divides the data into three cases and several remarks are in order.

- For a new instance, we only count at most k of its opposite pairwise losses. This is key to preventing the fluctuation of the decision function.
- For the k-nearest opposite support vectors to the new instance $\mathbf{z}_t$, i.e., the support vectors in $N_k^{-y_t}(\mathbf{z}_t)$, the absolute values of the weights are added by $|\eta C y_t|$; see the second case in Eq. (3.10). This will keep a balanced updating, which is in favor of imbalanced data.
- When the new instance does not incur errors or the labels of previously learned support vectors are the same as that

of the new instance, the updating rule is the same as NORMA [29], i.e., just decaying the weight by a constant factor, $1 - \eta$.

Algorithm 3 UpdateBuffer–RS++

```
1: Input:
     • the received sample $\mathbf{z}_t$ and its weight $\alpha_t$
     • the buffer $\mathcal{K}$ to be updated
     • the buffer size $N$
     • the number of instances received until trial $t$, $N_t$
2: Output: the updated buffer $\mathcal{K}$
3: if $|\mathcal{K}.\mathcal{B}| < N$ then
4:   $\mathcal{K}.\mathcal{A} = \mathcal{K}.\mathcal{A} \cup \{\alpha_t\}$, $\mathcal{K}.\mathcal{B} = \mathcal{K}.\mathcal{B} \cup \{\mathbf{z}_t\}$
5: else
6:   Sample $Z$ from a Bernoulli distribution with $\Pr(Z = 1) = N/N_t$
7:   if $Z = 1$ then
8:     Uniformly select an instance $\mathbf{z}_r$
9:     Update $\mathcal{K}.\mathcal{A}$: $\mathcal{K}.\mathcal{A} = \mathcal{K}.\mathcal{A} \setminus \{\alpha_{r,t}\} \cup \{\alpha_{t,t}\}$
10:    Update $\mathcal{K}.\mathcal{B}$: $\mathcal{K}.\mathcal{B} = \mathcal{K}.\mathcal{B} \setminus \{\mathbf{z}_r\} \cup \mathbf{z}_t$
11:  else
12:    $\mathbf{z}_r = \mathbf{z}_t$, $\alpha_{r,t} = \alpha_{t,t}$
13:  end if
14:  Find $\mathbf{z}_c = \arg\max_{\mathbf{z}_i \in \mathcal{K}.\mathcal{B}} \{k(\mathbf{x}_r, \mathbf{x}_i)\}$
15:  Set $\alpha_{c,t} = \alpha_{c,t} + \alpha_{r,t}$ and update $\alpha_{c,t}$ in $\mathcal{K}.\mathcal{A}$
16: end if
17: return $\mathcal{K}$
```

Update Buffers

The setting of fixed budgets raises the problem of updating the buffer when it is full. A key challenge is to maintain the buffers with the most informative support vectors so as to achieve stability of the model performance during the training. Traditionally, First-In-First-Out (FIFO) and Reservoir Sampling (RS) are two typical stream oblivious policies [46] to update the buffers and have demonstrated their effectiveness in online linear AUC maximization [60]. However, they will degrade the performance of the kernel-based online learning algorithms as they will discard support vectors [52].

To alleviate information loss, we design a sophisticated compensation scheme. Let the removed support vector be $\mathbf{z}_r = (\mathbf{x}_r, y_r)$. We find the most similar support vector $\mathbf{z}_c = (\mathbf{x}_c, y_c)$ with $y_c = y_r$ in $\mathcal{K}_t^{y_r}$ and update its corresponding weight. By considering the updating rule in Eq. (3.8), we obtain the new decision function as follows:

$$\hat{f}_{t+1}(\mathbf{x}) = f_{t+1}(\mathbf{x}) - \alpha_{r,t} k(\mathbf{x}_r, \mathbf{x}).$$

We determine the updated weight $\Delta\alpha_{c,t}$ of the compensated support vector $\mathbf{z}_c$ by keeping track of all information with a change in the value of the decision function. That is,

$$\begin{aligned} f_{t+1}(\mathbf{x}) &= \hat{f}_{t+1}(\mathbf{x}) + \Delta\alpha_{c,t} \cdot k(\mathbf{x}_c, \mathbf{x}) \\ &= f_{t+1}(\mathbf{x}) - \alpha_{r,t} k(\mathbf{x}_r, \mathbf{x}) + \Delta\alpha_{c,t} \cdot k(\mathbf{x}_c, \mathbf{x}). \end{aligned} \tag{3.11}$$

Hence, we set $\Delta\alpha_{c,t} = \alpha_{r,t} \frac{k(\mathbf{x}_r, \mathbf{x})}{k(\mathbf{x}_c, \mathbf{x})} \approx \alpha_{r,t}$ due to the similarity of the removed support vector $\mathbf{x}_r$ and the compensated support vector $\mathbf{x}_c$. We then obtain the updating rule for f_{t+1} with its compensation, f_{t+1}^{++}:

$$\begin{aligned} f_{t+1}^{++} = (1-\eta)f_t^{++} + \eta\partial_f\hat{\mathcal{L}}_t(f)|_{f=f_t^{++}} \\ + \alpha_{r,t}\left(k(\mathbf{x}_c,\cdot) - k(\mathbf{x}_r,\cdot)\right), \end{aligned} \tag{3.12}$$

where f_t^{++} is the previously compensated decision function. When either buffer is not full, $f_t^{++} = f_t$ and the update is done by Eq. (3.5). Ideally, if $k(\mathbf{x}_c, \mathbf{x})$ equals $k(\mathbf{x}_r, \mathbf{x})$, f_t^{++} incorporates all the support vectors and corresponds to the one learned with infinite budgets.

Algorithm 3 shows the procedure of the extended Reservoir Sampling (RS++) when the compensation scheme is incorporated.

- In line 3 to line 4, if the buffer is not full, i.e., $|\mathcal{K}.\mathcal{B}| < N$, the new instance becomes a support vector and is directly added into the buffer $\mathcal{K}$.
- In line 6 to line 10, if the buffer is full, reservoir sampling is performed. That is, with probability $\frac{N}{N_t}$, we update the buffer by randomly replacing one support vector $\mathbf{z}_r$ in $\mathcal{K}.\mathcal{B}$ with $\mathbf{z}_t$.
- In line 12, if replacement is not conducted, the removed support vector $\mathbf{z}_r$ is set to the new instance $\mathbf{z}_t$.
- In line 14 to line 15, this is the extension of RS. We find the most similar support vector $\mathbf{z}_c$ to the removed support vector $\mathbf{z}_r$, update its weight and put its weight back to the buffer $\mathcal{K}.\mathcal{A}$.

Similarly, we can define the extended FIFO strategy, namely

FIFO++. For FIFO++, lines 6 to 13 in Algorithm 3 are replaced by removing the first support vector in the buffer and adding the new instance as a new support vector to the end of the buffer.

3.3 Smooth Pairwise Hinge Loss

In [17], the proposed algorithm which uses a smooth pairwise loss function has been proved to achieve a faster convergence rate $O(1/T)$. Hence, to exploit the smoothness of the loss function, we consider the square of the pairwise hinge loss function, which is given by

$$\ell_{sh}(f, \mathbf{z}, \mathbf{z}') = \left(\frac{|y-y'|}{2}\left[1 - \frac{1}{2}(y-y')(f(\mathbf{x}) - f(\mathbf{x}'))\right]_+\right)^2. \tag{3.13}$$

We substitute Eq. (3.13) into Eq. (3.4) and compute the decision function by minimizing the following *smooth localized instantaneous regularized risk of AUC* associated with $\mathbf{z}_t$:

$$\tilde{\mathcal{L}}_t(f) := \tilde{\mathcal{L}}_t(f) = \frac{1}{2}\|f\|^2_{\mathcal{H}} + C\sum\nolimits_{\mathbf{z}_i} \ell_{sh}(f, \mathbf{z}_t, \mathbf{z}_i). \tag{3.14}$$

Using the standard stochastic gradient descent method to update the decision function, we have $\tilde{f}_1 = 0$ and the updating rule is given by

$$\tilde{f}_{t+1} := \tilde{f}_t - \eta \partial_f \tilde{\mathcal{L}}_t(f)|_{f=\tilde{f}_t}, \tag{3.15}$$

where $\eta > 0$ is the learning rate and the partial derivative is

$$\partial\tilde{\mathcal{L}}_t(\tilde{f}_t) = \tilde{f}_t - 2C\sum_{\mathbf{z}_i}\mathbb{I}[\ell_h(\cdot) > 0]\cdot\ell_h(\cdot)\cdot\varphi(\mathbf{z}_t,\mathbf{z}_i). \tag{3.16}$$

Here, $\ell_h(\cdot)$ means $\ell_h(\tilde{f},\mathbf{z}_t,\mathbf{z}_i)$, which is defined by Eq. (3.2).

Similarly, we define the valid set V_t and its complementary set $\overline{V_t}$ at the t-th trial as follows:

$$\begin{aligned} V_t &:= \{i \in I_t^{-y_t} | \mathbf{z}_i \in N_{t,k}^{-y_t}(\mathbf{z}_t) \wedge \ell_h(\tilde{f},\mathbf{z}_t,\mathbf{z}_i) > 0\}, && (3.17)\\ \overline{V_t} &:= I_t^{-y_t} \setminus V_t. \end{aligned}$$

Then, the corresponding updating rule for the kernel weights at the t-th trial is derived as follows:

$$\alpha_{i,t} = \begin{cases} 2\eta C y_t \sum_{i\in V_t}\ell_h(\tilde{f}_t,\mathbf{z}_t,\mathbf{z}_i), & i = t,\\ (1-\eta)\alpha_{i,t-1} - 2\eta C y_t \ell_h(\tilde{f}_t,\mathbf{z}_t,\mathbf{z}_i), & \forall i \in V_t,\\ (1-\eta)\alpha_{i,t-1}, & \forall i \in I_t^{y_t}\cup\overline{V_t}. \end{cases} \tag{3.18}$$

Similarly, we express the updating rule for $\tilde{f}_{t+1}$ with compensation, i.e., $\tilde{f}_{t+1}^{++}$, as

$$\begin{aligned} \tilde{f}_{t+1}^{++} = (1-\eta)\tilde{f}_t^{++} + \eta\partial_f\tilde{\mathcal{L}}_t(f)|_{f=\tilde{f}_t^{++}} && (3.19)\\ + \alpha_{r,t}\left(k(\mathbf{x}_c,\cdot) - k(\mathbf{x}_r,\cdot)\right), \end{aligned}$$

where $\tilde{f}_t^{++}$ is the previous decision function. When either buffer is not full, $\tilde{f}_t^{++}$ corresponds to the original decision function without compensation, i.e., $\tilde{f}_t$ updated by Eq. (3.15).

3.4 Regret Analysis

Regret analysis [33] is widely used in analyzing learning performance of online learning algorithms. In this section, we derive regret bounds for the non-smooth pairwise hinge loss and the smooth pairwise hinge loss.

Regret for the Non-smooth Pairwise Hinge Loss

Recall that the regret at time T is defined as the difference between the objective value up to the T-th trial and the smallest objective value from hindsight, i.e.,

$$R_T = \sum_{t=1}^{T} \hat{\mathcal{L}}_t(f_t) - \hat{\mathcal{L}}_t(f^*), \tag{3.20}$$

where f^* is the optimal decision function solved by Eq. (3.3) with the pairwise hinge loss function defined by Eq. (3.2) from hindsight, and f_t corresponds to the updating in Eq. (3.5).

In the following, unless otherwise specified, $\mathbf{z}_i$ means $\mathbf{z}_i \in N_{t,k}^{-y_t}(\mathbf{z}_t)$. That is, $\mathbf{z}_i$ is one of the k-nearest opposite support vectors to $\mathbf{z}_t$. Let $0 < \xi_1 \leq X$ be such that $k(\mathbf{x}_t, \mathbf{x}_i) \geq \xi_1^2$, $\forall\, \mathbf{z}_i = (\mathbf{x}_i, y_i) \in N_t^{-y_t}(\mathbf{z}_t)$. To simplify the notation, we define the following constant as the bound on the distance between the points in the neighbor set:

$$\|\varphi(\mathbf{z}_t, \mathbf{z}_i)\|_{\mathcal{H}} \leq c_p := \sqrt{2X^2 - 2\xi_1^2}. \tag{3.21}$$

We first present the bound on the norm of the decision function and the bound of the pairwise hinge loss function.

Lemma 1 *Suppose that for all* $\mathbf{x} \in \mathbb{R}^d$, $k(\mathbf{x}, \mathbf{x}) \leq X^2$, *where* $X > 0$. *Let* $0 < \xi_1 \leq X$ *be such that* $k(\mathbf{x}_t, \mathbf{x}_i) \geq \xi_1^2$, $\forall\, \mathbf{z}_i = (\mathbf{x}_i, y_i) \in N_t^{-y_t}(\mathbf{z}_t)$. *With* $f_1 = 0$ *and the updating rule in Eq. (3.5), we have*

$$\|f_{t+1}\|_{\mathcal{H}} \leq Ckc_p. \tag{3.22}$$

Lemma 2 *With the same assumption as Lemma 1 and the pairwise hinge loss function* $\ell_h : \mathcal{H} \times \mathcal{Z} \times \mathcal{Z} \to [0, U]$ *defined in Eq. (3.2), we have*

$$U = 1 + Ckc_p^2. \tag{3.23}$$

The proofs of the above two Lemmas are provided in Appendix A.

Now we derive the regret bound of the decision function updated by Eq. (3.5):

Theorem 1 *Suppose that for all* $\mathbf{x} \in \mathbb{R}^d$, $k(\mathbf{x}, \mathbf{x}) \leq X^2$, *where* $X > 0$. *Let* $0 < \xi_1 \leq X$ *be such that* $k(\mathbf{x}_t, \mathbf{x}_i) \geq \xi_1^2$, $\forall\, \mathbf{z}_i = (\mathbf{x}_i, y_i) \in N_t^{-y_t}(\mathbf{z}_t)$. *Given* $k > 0, C > 0, \eta > 0$, *and a bounded pairwise hinge loss function* $\ell_h : \mathcal{H} \times \mathcal{Z} \times \mathcal{Z} \to [0, U]$, *with* $f_1 = 0$ *and* f_t *updated by Eq. (3.5), we have*

$$R_T \leq \frac{\|f^*\|_{\mathcal{H}}^2}{2\eta} + \eta Ck \sum_{t=1}^{T} \left((U-1) + \frac{1}{2}(k+1)Cc_p^2 \right). \tag{3.24}$$

Moreover, suppose that $\forall i \in I_t^+ \cup I_t^-$, $|\alpha_{i,t}| \in [0, \gamma\eta]$, *and* $k(\mathbf{x}_r, \mathbf{x}_c) \geq \xi_2^2$ *with* $0 < \xi_2 \leq X$ *for any replaced support vector* $\mathbf{x}_r$ *and compensated support vector* $\mathbf{x}_c$ *in any trial. With* $f_1^{++} = 0$

and f_t^{++} updated by Eq. (3.12), we have

$$R_T^{++} \leq R_T + T\left(2\gamma C k c_p c_d + \gamma^2 c_d^2\right). \tag{3.25}$$

Details of the proof are given in Appendix A. Several remarks include:

- By setting η to $O(1/\sqrt{T})$, we can see that $R_T \sim O(\sqrt{T})$, which is equivalent to the $O(1/\sqrt{T})$ average regret for KOIL. The bounds we derived are the same as standard online learning algorithms. It should be noted that our derived regret bounds are also different from the mistake bounds derived in [4, 10, 35], which aim at maximizing classification accuracy.
- The regret bound R_T^{++} is larger than R_T with an undesired term related to T. However, we argue that it is meaningful as γ can be restricted to be proportional to $O(1/\sqrt{T})$, which yields a regret bound of $O(\sqrt{T})$. It would be interesting to see if a better bound is possible, and we leave this as a future work. Moreover, when $\xi_2^2 = X^2$, i.e., the compensated support vector is the same as the replaced support vector, we can obtain $R_T^{++} = R_T$. This result implies that the decision function learned by the replacement with compensation updating strategy can approach the decision function learned with infinite budgets. The experimental results also verify this observation.
- The derived regret bounds are proportional to k, not N (the budget), which also implies that $k = 1$ will yield the smallest

theoretical regret bound. The result is different from the regret bound derived in online AUC maximization [60]. After tracing the model, we know it is because we explore a localized AUC metric defined in Eq. (3.4). Our empirical evaluation shows that the actual best k is not 1, but set around 10% of the budget; see detailed results in the experimental section. We conjecture a more accurate surrogate of the AUC metric can overcome this issue and leave this as a future work.

- By exploiting the convexity of the localized instantaneous regularized risk of AUC defined in Eq. (3.4) and confining the range of $|\alpha_t|$ from 0 to $\gamma\eta$, we can derive the corresponding regret bound for the decision updated by Eq. (3.12) as in [23]. However, the derived regret bound is proportional to T, which is a little loss. We leave the work of exploring more advanced techniques to derive a tighter bound in the future.

Regret for the Smooth Pairwise Hinge Loss

As before, we define the regret for the smooth pairwise hinge loss by

$$\tilde{R}_T = \sum_{t=1}^{T} \tilde{\mathcal{L}}_t(\tilde{f}_t) - \tilde{\mathcal{L}}_t(\tilde{f}^*), \tilde{R}_T^{++} = \sum_{t=1}^{T} \tilde{\mathcal{L}}_t(\tilde{f}_t^{++}) - \tilde{\mathcal{L}}_t(\tilde{f}^*), \quad (3.26)$$

where $\tilde{f}^*$ is the optimal decision function solved by Eq. (3.3) with the smooth pairwise hinge loss function defined by Eq. (3.13) from hindsight, and $\tilde{f}_t$ corresponds to the updating rule in

Eq. (3.15).

Theorem 2 *Suppose that for all $\mathbf{x} \in \mathbb{R}^d$, $k(\mathbf{x}, \mathbf{x}) \le X^2$, where $X > 0$. Let $0 < \xi_1 \le X$ be such that $k(\mathbf{x}_t, \mathbf{x}_i) \ge \xi_1^2$, $\forall\, \mathbf{z}_i = (\mathbf{x}_i, y_i) \in N_t^{-y_t}(\mathbf{z}_t)$. Given $k > 0, C > 0$, and $\sum_{t=1}^{T} \tilde{\mathcal{L}}_t(\tilde{f}^*) \le TL^*$, with $\tilde{f}_1 = 0$ and $\tilde{f}_t$ updated by Eq. (3.15), we have*

$$\tilde{R}_T \le 2\tau \|\tilde{f}^*\|_{\mathcal{H}}^2 + \|\tilde{f}_{\mathcal{H}}^*\| \sqrt{2\tau T L^*}, \tag{3.27}$$

where $\tau = (1+\zeta)$, $\zeta = 2Ck^2c_p^2$, and η is set to $1/\left(\tau + \sqrt{\tau^2 + \tau T L^* / \|\tilde{f}\|_{\mathcal{H}}}\right)$. Moreover, suppose that $\forall i \in I_t^+ \cup I_t^-$, $\alpha_{i,t} \in [0, \gamma\eta]$, and $k(\mathbf{x}_r, \mathbf{x}_c) \ge \xi_2^2$ with $0 < \xi_2 \le 0$ for any removed support vector $\mathbf{x}_r$ and compensated support vector $\mathbf{x}_c$ in any trial. With $\tilde{f}_1^{++} = 0$ and $\tilde{f}_t^{++}$ updated by Eq. (3.19), we have

$$\tilde{R}_T^{++} \le \tilde{R}_T + T\left(\gamma C k \left(\frac{\eta}{\nu} + 2\tilde{U}\right) c_p c_d + \frac{\gamma^2 \eta}{\nu} c_d^2\right), \tag{3.28}$$

where $\nu := \eta - 2kC\eta c_p^2$.

The proof of Theorem 2 is provided in Appendix A. The result shows that if the data is separable, i.e., $L^* = 0$, KOIL with the smooth pairwise loss function can attain $O(1/T)$ average regret bound. For general case, the bound is $O(1/\sqrt{T})$. We also leave the regret bound of the decision function updated by Eq. (3.15) as future work.

3.5 Experiment

In this section, we conduct extensive experiments on both synthetic and benchmark datasets to evaluate the performance of our proposed KOIL [1] algorithm with fixed budgets.

Compared Algorithms

We compare our proposed KOIL with the state-of-the-art online learning algorithms. Since we only focus on online imbalanced learning, for fair comparison, we do not compare with existing batch-trained imbalanced learning algorithms. Specifically, we compare online linear algorithms and kernel-based online learning algorithms with a finite or infinite buffer size.

- "Perceptron": the classical perceptron algorithm [37];
- "OAM_{seq}": an online linear AUC maximization algorithm [60];
- "OPAUC": One-pass AUC maximization [17];
- "NORMA": online learning with kernels [29];
- "RBP": Randomized budget perceptron [4];
- "Forgetron": a kernel-based perceptron on a fixed budget [10];
- "Projectron/**Projectron**++": a bounded kernel-based perceptron [35];
- "$\text{KOIL}_{\text{RS++}}$/$\text{KOIL}_{\text{FIFO++}}$: our proposed kernelized online imbalanced learning algorithm with the pairwise hinge loss function and fixed budgets updated by RS++ and FIFO++, respectively.
- "$\text{KOIL}^2_{\text{RS++}}$/$\text{KOIL}^2_{\text{FIFO++}}$": our proposed kernelized online

[1]Demo codes in both C++ and Matlab can be downloaded at `https://www.dropbox.com/sh/nuepinmqzepx54r/AAAKuL4NSZeOIRpGuNIsuxQxa?dl=0`.

imbalanced learning algorithm with the smooth pairwise hinge loss function and fixed budgets updated by RS++ and FIFO++, respectively.

Experimental Setup

To make fair comparisons, all algorithms adopt the same setup. We set the learning rate to a small constant $\eta = 0.01$ and apply a 5-fold cross validation to find the penalty cost $C \in 2^{[-10:10]}$ in the validation set. For kernel-based methods, we use the Gaussian kernel and tune its parameter $\sigma \in 2^{[-10:10]}$ by a 5-fold cross validation in the validation set. For NORMA, we apply a 5-fold cross validation to select λ and $\nu \in 2^{[-10:10]}$. For Projectron, we apply a similar 5-fold cross validation to select the parameter of projection difference $\eta \in 2^{[-10:10]}$.

Experiments on Synthetic Datasets

To illustrate KOIL and show the power of the kernel method, we generate a synthetic dataset in 2D space; see an example in Fig 1.1(a). The positive samples are generated from the Gaussian distribution with the mean at $(\frac{1}{2}, \frac{1}{2})$ and the standard deviation being 0.1 in two dimensions. The negative samples are generated from a mixture of four Gaussian with the same standard deviation as the positive samples and means at $(\frac{1}{6}, \frac{1}{2})$, $(\frac{1}{2}, \frac{1}{6})$, $(\frac{1}{2}, \frac{5}{6})$, $(\frac{5}{6}, \frac{1}{2})$, respectively. The Bayes error of this dataset is about 4.5%.

Table 3.1: Summary of all datasets.

Dataset	Samples	Dimensions	T^-/T^+
Syn1	1,000	2	4
Syn2	1,100	2	10
Syn3	5,100	2	50
Syn4	10,100	2	100
sonar	208	60	1.144
australian	690	14	1.248
heart	270	13	1.250
ionosphere	351	34	1.786
diabetes	768	8	1.866
glass	214	9	2.057
german	1,000	24	2.333
svmguide2	391	20	2.342
segment	2,310	19	6.000
satimage	4,435	36	9.687
vowel	528	10	10.000
letter	15,000	16	26.881
poker	25,010	10	47.752
shuttle	43,500	9	328.546

Following the above setup, we generate different synthetic datasets with different imbalanced ratios to explore the performance of KOIL in different scenarios. The datasets consist of

- Syn1: a set of data with imbalanced ratio 1:4 consisting of 200 positive samples and 800 negative samples;
- Syn2: a set of data with imbalanced ratio 1:10 consisting of 100 positive samples and 1,000 negative samples;
- Syn3: a set of data with imbalanced ratio 1:50 consisting of 100 positive samples and 5,000 negative samples;

- Syn4: a set of data with imbalanced ratio 1:100 consisting of 100 positive samples and 10,000 negative samples.

The detailed statistics of the synthetic datasets can be found in Table 3.1. Experimental results on all the synthetic datasets are reported in Table 3.2. Obviously, these four datasets are linearly non-separable in the original space. Kernel-based online learning algorithms significantly outperform the online linear algorithms. For example, in the Syn1 dataset, Perceptron and the OAM_{seq} with buffer size being 100 for each class only attain AUC scores of 0.495 ± 0.031 and 0.467 ± 0.027, respectively. These are even poorer than random guesses. For NORMA with an infinite buffer size, it achieves an AUC score of 0.940 ± 0.013. Our proposed $\text{KOIL}_{\text{RS++}}$ and $\text{KOIL}_{\text{FIFO++}}$ with a buffer size of only 50 for each class and $k = 5$ can improve the AUC scores to 0.961 ± 0.016 and 0.960 ± 0.014, respectively. Our KOIL with the smooth pairwise loss function can attain comparable or even better performance than that with the non-smooth loss function.

Experiments on Benchmark Real-world Datasets

We conduct experiments on 14 benchmark datasets obtained from the UCI and LIBSVM websites. These benchmark datasets exhibit different degrees of class skew, where the imbalanced ratio ranges from 1.144 to 328.546. To evaluate the effectiveness of nonlinear classification models, we select

Table 3.2: Average AUC performance (mean±std) on the synthetics datasets, •/∘ (-) indicates that both/one of $\text{KOIL}_{\text{RS++}}$ and $\text{KOIL}_{\text{FIFO++}}$ are/is significantly better (worse) than the corresponding method (pairwise t-tests at 95% significance level).

Data	$\text{KOIL}_{\text{RS++}}$	$\text{KOIL}_{\text{FIFO++}}$	$\text{KOIL}^2_{\text{RS++}}$	$\text{KOIL}^2_{\text{FIFO++}}$
Syn1	.961±.016	.960±.014	.967±.011	**.968**±.011
Syn2	.959±.022	.958±.018	.961±.017	**.962**±.018
Syn3	.939±.029	.941±.025	**.943**±.022	.942±.023
Syn4	.965±.014	.966±.013	**.968**±.013	.966±.015
win/tie/loss			0/4/0	0/4/0

Data	Perceptron	OAM_{seq}	OPAUC	NORMA	RBP	Forgetron	Projectron	Projectron++
Syn1	.495±.031•	.501±.021•	.503±.032•	.940±.013•	.948±.021•	.878±.147•	.954±.019	.953±.017
Syn2	.484±.037•	.502±.032•	.508±.032•	.937±.041•	.887±.062•	.954±.023	.941±.032•	.944±.023•
Syn3	.495±.025•	.499±.022•	.492±.020•	.769±.087•	.872±.081•	.807±.130•	.901±.064•	.922±.039•
Syn4	.510±.023•	.495±.026•	.499±.022•	.834±.205•	.892±.069•	.844±.097•	.962±.015	.948±.024•
win/tie/loss	4/0/0	4/0/0	4/0/0	4/0/0	4/0/0	3/1/0	2/2/0	3/1/0

data with comparatively low dimensionalities with respect to the number of data. Data on these benchmark datasets are collected from a wide range of application domains, and some datasets, e.g.,shuttle, poker and letter, have sufficiently large volumes of data to evaluate the scalability of online learning algorithms. The detailed statistics of the datasets is summarized in Table 3.1.

For each dataset, we conduct 5-fold cross validation on all the algorithms, where four folds of the data are used for training while the rest for testing. The 5-fold cross validation is independently repeated four times. We set the buffer size to 100 for each class for all related algorithms, including OAM_{seq}, RBP, and Forgetron. We then average the AUC performance of 20 runs and report the results in Table 3.3.

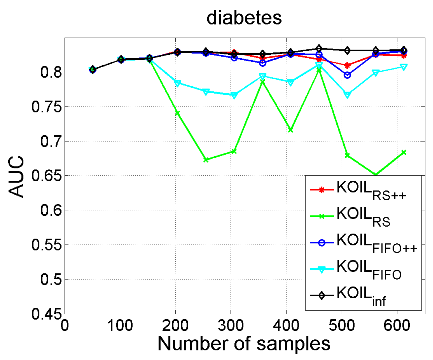
Table 3.3: Average AUC performance (mean±std) on the benchmark datasets, ●/○ (-) indicates that both/one of KOIL$_{\mathrm{RS++}}$ and KOIL$_{\mathrm{FIFO++}}$ are/is significantly better (worse) than the corresponding method (pairwise t-tests at 95% significance level).

Data	KOIL$_{\mathrm{RS++}}$	KOIL$_{\mathrm{FIFO++}}$	KOIL$^2_{\mathrm{RS++}}$	KOIL$^2_{\mathrm{FIFO++}}$
sonar	.955±.028	.955±.028	**.957**±.031	**.957**±.031
australian	.923±.023	.922±.026	.919±.024	.920±.026
heart	.908±.040	.910±.040	.911±.038	.908±.037
ionosphere	**.985**±.015	**.985**±.015	.959±.026●	.952±.031●
diabetes	.826±.036	.830±.030	.817±.037○	.825±.028
glass	**.887**±.053	.884±.054	.885±.048	.885±.048
german	.769±.032	.778±.031	.774±.030	.769±.037○
svmguide2	**.897**±.040	.885±.043	.891±.042	.882±.040○
segment	.983±.008	**.985**±.012	.970±.012●	.959±.015●
satimage	**.924**±.012	.923±.015	.922±.012	.922±.013
vowel	**1.000**±.000	**1.000**±.001	.998±.007●	.993±.014●
letter	.933±.021	**.942**±.017	.926±.022●	.932±.017○
poker	.681±.031	**.693**±.032	.654±.023●	.676±.031●
shuttle	.950±.040	.956±.021	.946±.039	.953±.020
win/tie/loss			6/8/0	7/7/0

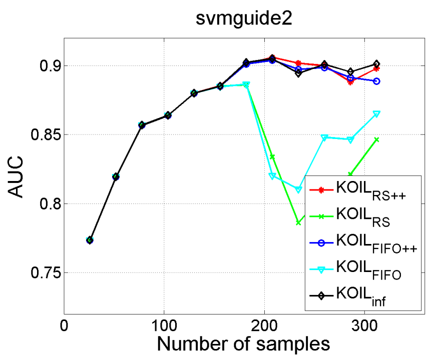
Data	Perceptron	OAM$_{\mathrm{seq}}$	OPAUC	NORMA	RBP	Forgetron	Projectron	Projectron++
sonar	.803±.083●	.843±.056●	.844±.077●	.925±.044●	.913±.032●	.896±.054●	.896±.049●	.896±.049●
australian	.869±.035●	**.925**±.024	.923±.025	.919±.023	.911±.017●	.912±.026●	.923±.024	.923±.024
heart	.876±.066●	**.912**±.040	.901±.043○	.890±.051●	.865±.043●	.900±.053○	.902±.038	.905±.042
ionosphere	.851±.056●	.905±.041●	.888±.046●	.961±.016●	.960±.030●	.945±.031●	.964±.025●	.963±.027●
diabetes	.726±.059●	.827±.033	.805±.035●	.792±.032●	.828±.034	.820±.027○	.832±.033	**.833**±.033
glass	.810±.065●	.827±.064●	.800±.074●	.811±.077●	.811±.071●	.813±.075●	.811±.070●	.781±.076●
german	.748±.033●	.777±.027	**.787**±.026 -	.766±.032○	.699±.038●	.712±.054●	.769±.028○	.770±.024
svmguide2	.860±.037●	.886±.045○	.859±.050●	.865±.046●	.890±.038	.864±.045●	.886±.044○	.886±.045○
segment	.875±.020●	.919±.020●	.882±.019●	.910±.042●	.969±.017●	.943±.038●	.979±.013●	.978±.016●
satimage	.700±.015●	.755±.018●	.724±.016●	.914±.025●	.899±.018●	.892±.032●	.910±.015●	.904±.011●
vowel	.848±.070●	.905±.024●	.885±.034●	.996±.005●	.968±.017●	.987±.027●	.982±.013●	.994±.019●
letter	.767±.029●	.827±.021●	.823±.018●	.910±.027●	.928±.011○	.815±.102●	.926±.016●	.926±.015●
poker	.514±.030●	.503±.024●	.509±.031●	.577±.040●	.501±.031●	.572±.029●	.675±.027●	.675±.027●
shuttle	.520±.134●	**.999**±.000 -	.754±.043●	.725±.053●	.844±.041●	.839±.060●	.873±.063●	.795±.063●
win/tie/loss	14/0/0	9/4/1	12/1/1	13/1/0	12/2/0	14/0/0	11/3/0	10/4/0

Several observations of the results in Table 3.3 can be drawn as described in the followings:

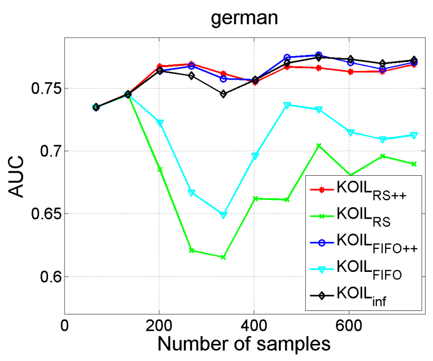
- Our KOIL with RS++ and FIFO++ updating policies perform better than online linear AUC maximization algorithms in most datasets. By examining the results of OAM_{seq} on the datasets of australian, heart, diabetes, german, and shuttle and those of OPAUC on australian and german, we speculate that for these datasets, a linear classifier is enough to achieve good performance, while a nonlinear classifier can be affected by outliers.
- Our proposed KOIL with the pairwise hinge loss function significantly outperforms all competing kernel-based algorithms in nearly all datasets. The results demonstrate the effectiveness of KOIL in imbalanced learning.
- Comparing the non-smooth KOIL with its smooth version, we observe that KOIL with the non-smooth loss function beats its smooth version in five datasets while being comparable in the rest nine datasets.
- In most datasets, kernel-based algorithms show better AUC performance than the linear algorithms in most of datasets. This again demonstrates the power of kernel methods in classifying real-world datasets.
- We observe that the performance of OAM_{seq} on satimage dataset is not as good as that in [60] and [52]. We check that this is mainly due to the different partition of the training and test data.



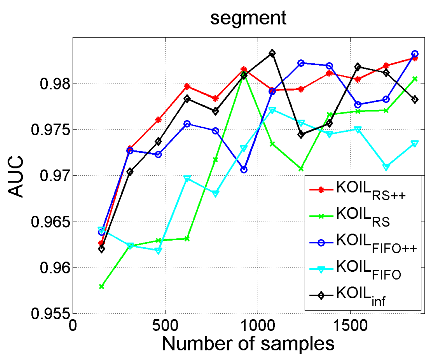
(a) diabetes



(b) svmguide2



(c) german



(d) segment

Figure 3.1: Average AUC performance of four datasets obtained by different updating policies of KOIL.

Evaluation on Updating Policies

We test the improvement of our updating policies, RS++ and FIFO++, with the original updating policies, RS and FIFO. We show in Figure 3.1 the average AUC performance of 20 runs on four typical datasets. The results of KOIL$_{\text{inf}}$, i.e., learning with infinite budgets, are provided for reference. We have the following observations:

- KOIL$_{\text{RS++}}$ and KOIL$_{\text{FIFO++}}$ attain nearly the same performance as KOIL_{inf}. The results confirm that the extended policies indeed compensate for the information that is lost when a support vector is replaced.

- Our KOIL with extended updating policies significant outperforms our KOIL with corresponding original stream oblivious policy when either buffer is full. Without compensation, the performance fluctuates and is easily affected by noisy samples. With compensation, KOIL can maintain the performance smoothly.

Sensitivity Evaluation of KOIL

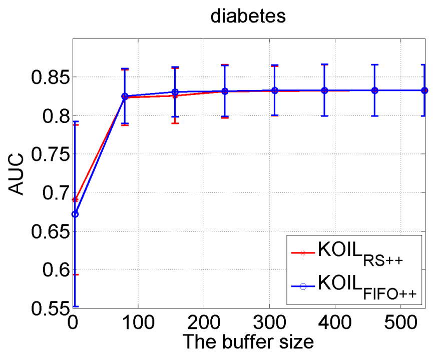
We first test the performance of KOIL with different buffer sizes. From Figure 3.2, we observe that the performance increases gradually with the increase in the buffer size and it is saturated when the size is relatively large. This is similar to the observations in [52, 60].

Next, we test the performance of KOIL with different k, which determines the number of localized support vectors. From Figure 3.3, we have the following observations:

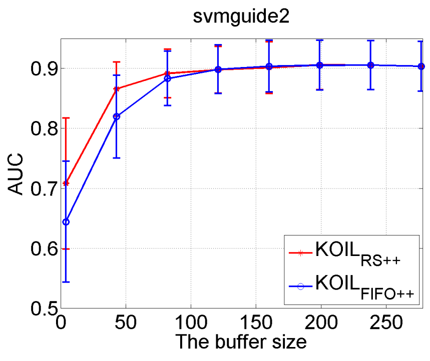
- When k is extremely small, say $k = 1$, KOIL only considers the pairwise loss incurred by the nearest opposite support vector of the new instance and cannot fully utilize the localized information. The updating weight is similar to NORMA, which adds a constant weight, $|\eta C y_t|$, to the misclassified new instance.

- KOIL usually attains the best performance when k equals 10% of the buffer size. The performance decreases when k increases. The results consistently show that by only utilizing the local information of the new instance, one can indeed prevent the effect of outliers.
- For some datasets, such as svmguide2 and german, the performance is not so sensitive to k. The reason may be that the learned support vectors in these datasets are well-separated when the buffers are full. Hence, new instances have little influence on the computation of the decision function.

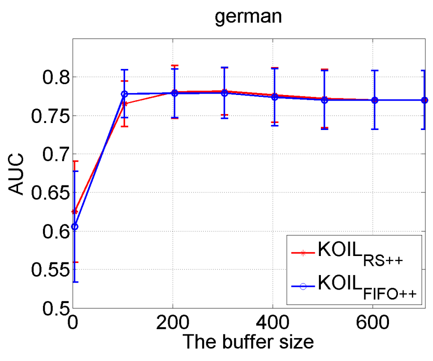
□ **End of chapter.**



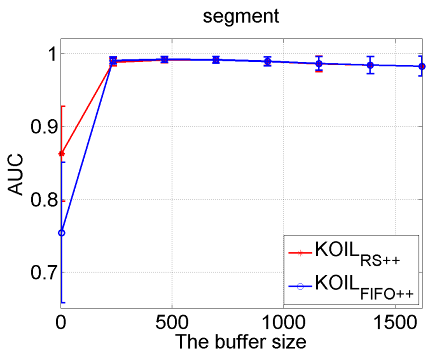
(a)



(b)

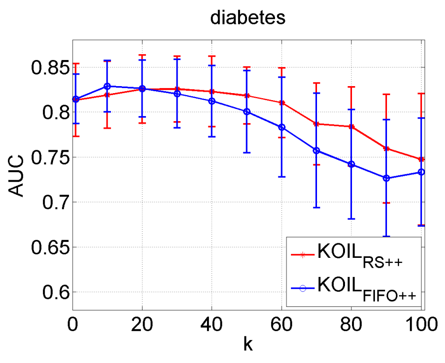


(c)

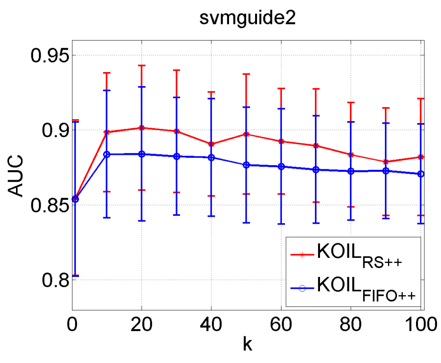


(d)

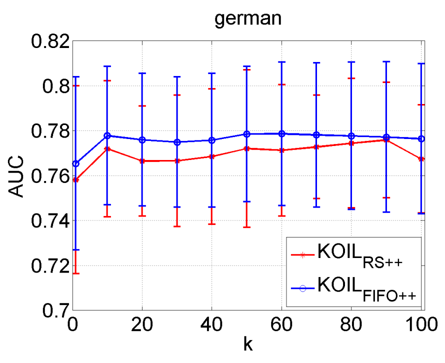
Figure 3.2: Average AUC of KOIL with different buffer sizes.



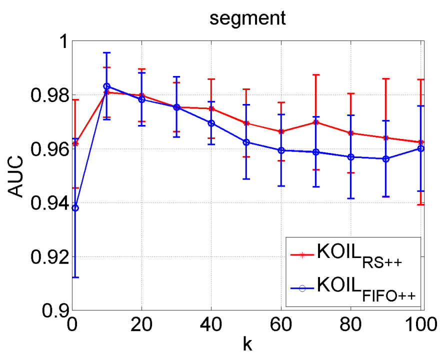
(a)



(b)



(c)



(d)

Figure 3.3: Average AUC of KOIL with different k. Here $k = [1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]$ and the budget is 100 for each buffer.

Chapter 4

KOIL with Multiple Kernel Learning

Summary

In this chapter, we exploit the multiple kernel learning framework to attain an accurate data representation for good performance.

4.1 Notations

Given a set of kernel functions $K = \{k_l : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, l = 1, \ldots, m\}$, we aim to learn a linear combination of these kernel functions for the decision function. First, we denote $\mathbf{w}^t = [w_1^t, \ldots, w_m^t]$ as the unnormalized weight for m kernel classifiers and $\mathbf{q}^t = [q_1^t, \ldots, q_m^t]$ as the normalized weight for m kernel classifiers learned up to the t-th trial, i.e., $\mathbf{q}^t = \mathbf{w}^t/W^t$

where $W^t = |\mathbf{w}^t|$, and $\sum_{l=1}^m q_l^t = 1$. Hence, the decision function can be defined as

$$F_t(\mathbf{x}) = \sum_{l=1}^{m} q_l^t \cdot \text{sign}(f_{l,t}(\mathbf{x})), \tag{4.1}$$

where $f_{l,t}(\mathbf{x})$ is an element of the Reproducing Kernel Hilbert Space $\mathcal{H}_{k_l}$ endowed with the inner product k_l. The l-th kernel classifier at the t-th trial is defined to have the same form as in Eq. (2.1):

$$f_{l,t}(\mathbf{x}) = \sum\nolimits_{i \in I_t^+} \alpha_{l,i,t}^+ k_l(\mathbf{x}_i, \mathbf{x}) + \sum\nolimits_{j \in I_t^-} \alpha_{l,j,t}^- k_l(\mathbf{x}_j, \mathbf{x}).$$

Similarly, we define two buffers $\mathcal{K}_{l,t}^+$ and $\mathcal{K}_{l,t}^-$ to store the corresponding information (i.e., weights and support vectors) for the l-th kernel classifier at the t-th trial.

4.2 Online Multiple Kernel Selection

We adopt the similar workflow of online multiple kernel learning in [24] to automatically select good kernels. However, different from [24], our KOIL with Multiple Kernel Learning (MKL) focuses more on the pairwise loss function and selects good kernels from the predifined kernels with higher probability in the next trial of the online process. More specifically, we associate each kernel function k_l with a corresponding value w_l^t in the weight vector $\mathbf{w}^t$ at the t-th trial. The probability that the l-th kernel function k_l is selected in the next trial is propotional to w_l^t. Besides, a poor kernel yielding a large value of the pairwise loss at t-th trial will obtain a deduction on its corresponding value in $\mathbf{w}^t$. Hence, after iteratively learning from streaming data in an online mode, $\mathbf{w}^t$ will eventually converge. That is, our KOIL with MKL can eventually select good kernels from the predefined kernels by using the weighted sum of multiple kernel decision functions in Eq. (4.1).

Algorithm 4 shows the procedure of KOIL with multiple kernels.

- In line 6, we select the classifier based on the Bernoulli distribution which is proportional to the weight of the classifier. Since it is divided by the maximum weight of all classifiers, at least one classifier will be selected at each trial.
- In line 7 to line 15, it updates the predictor of the sampled classifier. It is noted that in order to avoid excessive update fluctuation, we define the loss function $\check{\mathcal{L}}_t$ as in Eq. (3.4) and Eq. (3.14), but without the regularization term. This necessitates a change in the update of α in the function UpdateKernel. Specifically, in UpdateKernel2, for the pairwise hinge loss function, α is updated by

$$\alpha_{i,t} = \begin{cases} \eta C y_t |V_t|, & i = t, \\ \alpha_{i,t-1} - \eta C y_t, & \forall i \in V_t, \\ \alpha_{i,t-1}, & \forall i \in I_t^{y_t} \cup \overline{V_t}. \end{cases} \tag{4.2}$$

For the smooth pairwise hinge loss function, α is updated

Algorithm 4 KOIL with MKL

```
1: Input:
     • the penalty parameter $C$, the learning rate $\eta$ and $\lambda$.
     • the maximum positive budget $N^+$ and negative budget $N^-$
     • the number of nearest neighbors $k$
2: Initialize $\mathbf{w}^1 = \mathbf{1}$, $\mathcal{K}_l^+.\mathcal{A} = \mathcal{K}_l^-.\mathcal{A} = \emptyset$, $\mathcal{K}_l^+.\mathcal{B} = \mathcal{K}_l^-.\mathcal{B} = \emptyset$, $N_{p,l} = N_{n,l} = 0$, $\forall l \in [1, m]$.
3: for $t = 1$ to $T$ do
4:   Receive a training sample $\mathbf{z}_t = (\mathbf{x}_t, y_t)$
5:   for $l = 1$ to $m$ do
6:     if BernSample($\mathbf{w}_l^t/[\max_j \mathbf{w}_j^t]$) == 1 then
7:       if $y_t == +1$ then
8:         $N_{p,l} = N_{p,l} + 1$
9:         $[\mathcal{K}_l^-, \mathcal{K}_l^+, \alpha_l]$ = UpdateKernel2($\mathbf{z}_t, \mathcal{K}_l^-, \mathcal{K}_l^+, C, \eta, k$)
10:        $\mathcal{K}_l^+$ = UpdateBuffer($\alpha_l, \mathbf{z}_t, \mathcal{K}_l^+, k, N^+, N_{p,l}$)
11:      else
12:        $N_{n,l} = N_{n,l} + 1$
13:        $[\mathcal{K}_l^+, \mathcal{K}_l^-, \alpha_l]$ = UpdateKernel2($\mathbf{z}_t, \mathcal{K}_l^+, \mathcal{K}_l^-, C, \eta, k$)
14:        $\mathcal{K}_l^-$ = UpdateBuffer($\alpha_l, \mathbf{z}_t, \mathcal{K}_l^-, k, N^-, N_{n,l}$)
15:      end if
16:      $w_{i_t}^{t+1} = w_{i_t}^t \exp(-\eta \check{\mathcal{L}}_t(f_{i_t}, y_t))$
17:    end if
18:  end for
19:  $\mathbf{q}^{t+1} = \mathbf{w}^{t+1}/|\mathbf{w}^{t+1}|$
20: end for
```

by

$$\alpha_{i,t} = \begin{cases} 2\eta C y_t \sum_{i \in V_t} \ell_h(\tilde{f}_t, \mathbf{z}_t, \mathbf{z}_i), & i = t, \\ \alpha_{i,t-1} - 2\eta C y_t \ell_h(\tilde{f}_t, \mathbf{z}_t, \mathbf{z}_i), & \forall i \in V_t, \\ \alpha_{i,t-1}, & \forall i \in I_t^{y_t} \cup \overline{V_t}. \end{cases} \tag{4.3}$$

- In line 16, the weight of the sampled kernel is updated by the exponential weighted average algorithm, where the weight is discounted by a large factor when the loss is large.

It is noted that in order to avoid fluctuation, we do not add the smoothing term to update the probability of selecting classifiers as those in [22, 24, 39, 49, 57].

4.3 Regret Analysis

Similar to Eq. (3.20) and Eq. (3.26), we can define the corresponding regret for $F_t(\mathbf{x})$ and obtain the expected regret bound associated with Algorithm 4.

Theorem 3 *Assuming the loss function is non-negative,* $\check{\mathcal{L}}_t(f_{i,t})$ *is bounded by* L *for all kernel predictors over all trials (i.e.,* $\max_{t=1}^T \check{\mathcal{L}}_t(f_{i,t}) \leq L$*), and* $\|\partial_f \check{\mathcal{L}}_t(f_{i,t})\| \leq G$*, we have*

$$E\left[\sum_{t=1}^{T}\sum_{i=1}^{m} q_i^t \check{\mathcal{L}}_t(f_{i,t})\right] \leq \min_{1\leq i\leq m} \min_{f\in\mathcal{H}_{k_i}} \sum_{t=1}^{T} \check{\mathcal{L}}_t(f)) + \frac{\|f\|_{\mathcal{H}_{k_i}}}{2\lambda} + \frac{T}{2}(\eta L^2 + \lambda G^2), \quad (4.4)$$

where $q_i^t = w_i^t / [\sum_{i=1}^m w_i^t]$.

Note that by assuming the optimal kernel predictor is bounded and setting $\eta, \lambda \propto 1/\sqrt{T}$, we can obtain a regret bound of $O(\sqrt{T})$ by following the proof in [57]. Differently, we can utilize the AM-GM inequality to remove the constant term $\ln m/\eta$ in [57]. Due to the space limitation, we omit the proof here. The proof of Theorem 3 is provided in Appendix A.

4.4 Experiment

We evaluate the performance of KOIL with MKL following the setting in [24]. That is, we use 16 kernel functions in our experiment, including 3 polynomial kernels (i.e., $k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^p$ of degree parameter $p = 1$, 2, and 3), and 13 Gaussian kernels (i.e., $k(\mathbf{x}_i, \mathbf{x}_j)$ =exp$(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2)$ of kernel width parameter σ in $2^{[-6:1:6]}$). The learning rates, η and λ, are set to 0.01. A 5-fold cross validation is applied to find the best penalty cost C from $2^{[-10:1:10]}$. Table 4.1 summarizes the results and reveals the following observations:

- KOIL with MKL attain better or comparable performance to KOIL using the optimal kernel selected by a 5-fold cross validation. Especially, KOIL with the smooth loss function can gain better performance on at least 7 datasets among all 14 datasets. For KOIL with the non-smooth loss function, the performance of KOIL with MKL is comparable in most datasets. We conjecture this may be due to the non-smoothness of the loss function.
- For some datasets, such as sonar and ionosphere, KOIL with MKL cannot beat KOIL with the tuned optimal kernel. We conjecture this may be due to the limitation of training data in these datasets. Training with multi-epoches as that in [57] may be a possible solution to improve the model performance.

Table 4.1: Average AUC performance (mean±std) on the benchmark datasets. ● (-) indicates that the performance by KOIL with MKL is significantly better than (comparable to) that by KOIL with the tuned optimal kernel (pairwise t-tests at 95% significance level).

Data	$\text{KOIL}^{\text{MKL}}_{\text{RS++}}$	$\text{KOIL}^{\text{MKL}}_{\text{FIFO++}}$	$\text{KOIL}^{\text{MKL 2}}_{\text{RS++}}$	$\text{KOIL}^{\text{MKL 2}}_{\text{FIFO++}}$
sonar	0.893±0.053	0.899±0.047	0.946±0.040 -	0.949±0.031 -
australian	0.922±0.027 -	0.919±0.028 -	0.918±0.026 -	0.911±0.024
heart	0.906±0.044 -	0.907±0.042 -	0.906±0.040 -	0.904±0.038 -
ionosphere	0.953±0.062	0.957±0.073	0.972±0.039 -	0.972±0.042●
diabetes	0.826±0.035 -	0.831±0.032 -	0.827±0.036●	0.822±0.033 -
glass	0.890±0.056 -	0.891±0.051 -	0.890±0.053 -	0.893±0.052 -
german	0.771±0.042 -	0.769±0.033 -	0.774±0.033 -	0.768±0.039 -
svmguide2	0.906±0.040●	0.896±0.049●	0.905±0.041●	0.903±0.043●
segment	0.993±0.004●	0.994±0.004●	0.991±0.005●	0.990±0.009●
satimage	0.937±0.015●	0.939±0.015●	0.938±0.012●	0.937±0.014●
vowel	0.999±0.002	0.999±0.002 -	0.999±0.002 -	0.998±0.003 -
letter	0.954±0.013●	0.959±0.014●	0.962±0.011●	0.968±0.008●
poker	0.690±0.035●	0.707±0.027●	0.709±0.023●	0.705±0.020●
shuttle	0.948±0.028 -	0.926±0.032	0.888±0.029	0.886±0.032
win/tie/loss	5/6/3	5/3/4	6/7/1	6/6/2

□ **End of chapter.**

Chapter 5

Conclusion

Summary

This chapter concludes the contributions of this thesis.

We proposed a kernel-based online learning algorithm to tackle the imbalanced binary classification problem. We maintained two buffers with fixed budgets to control the number of support vectors, which keep track of the global information of the decision boundary. We updated the weight of a new support vector by confining its influence on only its k-nearest opposite support vectors. More importantly, we designed a sophisticated compensation scheme to avoid information loss by transferring the weight of the removed support vector to its most similar one when either buffer is full. We showed that this compensation can make our learned decision function approach the one learned with infinite budgets. Furthermore, we exploit multiple kernel learning framework to determine the kernel for KOIL. Finally, we conducted extensive experiments to demonstrate the effectiveness of our proposed approach.

□ **End of chapter.**

Appendix A

Theoretical proof

Summary

Here we give the detailed proofs of the theorem in this thesis.

Proof of Lemma 1

Proof: First, since $\mathbf{z}_i$ is one of the k-nearest opposite support vectors of $\mathbf{z}_t$, the assumption $k(\mathbf{z}_t, \mathbf{z}_i) \geq \xi_1^2$ with $\xi_1^2 > 0$ is justified. We then have

$$\|\varphi(\mathbf{z}_t, \mathbf{z}_i)\|_{\mathcal{H}} = \sqrt{k(\mathbf{x}_t, \mathbf{x}_t) - 2k(\mathbf{x}_t, \mathbf{x}_i) + k(\mathbf{x}_i, \mathbf{x}_i)} \leq c_p, \quad \text{(A.1)}$$

where c_p is defined in Eq. (3.21).

Now we derive the bound on the norm of the decision function:

$$\begin{aligned}
&\|f_{t+1}\|_{\mathcal{H}} \\
\le&(1-\eta)\|f_t\|_{\mathcal{H}} + \eta C \sum_{\mathbf{z}_i} \mathbb{I}[\ell_h(f, \mathbf{z}_t, \mathbf{z}_i) > 0] \cdot \|\varphi(\mathbf{z}_t, \mathbf{z}_i)\|_{\mathcal{H}} \\
\le&(1-\eta)\|f_t\|_{\mathcal{H}} + \eta C k c_p. \qquad \text{(A.2)}
\end{aligned}$$

In the above, the first inequality is by substituting Eq. (3.7) into Eq. (3.5) to calculate f_{t+1} and the triangle inequality. The second inequality is due to Eq. (A.1) and the fact that the number of elements in $N_{t,k}^{-y_t}(\mathbf{z}_t)$ is at most k.

By expanding $\|f_t\|_{\mathcal{H}}$ iteratively, we have

$$\|f_{t+1}\|_{\mathcal{H}} \le (1-\eta)^t \|f_1\|_{\mathcal{H}} + \frac{1-(1-\eta)^t}{\eta} \eta C k c_p \le C k c_p.$$

The second inequality holds when $\eta < 1$, $1-(1-\eta)^t \le 1$ for $t \in [T]$ and $f_1 = 0$. $\square$

Proof of Lemma 2

Proof: Based on the pairwise hinge loss defined in Eq. (3.2), we have

$$\begin{aligned}
\ell_h(f_t, \mathbf{z}_t, \mathbf{z}_i) &\le 1 + |f_t(\mathbf{x}_t) - f_t(\mathbf{x}_i)| \\
&= 1 + |\langle f_t, k(\mathbf{x}_t, \cdot) - k(\mathbf{x}_i, \cdot)\rangle_{\mathcal{H}}| \\
&\le 1 + \|f_t\|_{\mathcal{H}} \cdot \|k(\mathbf{x}_t, \cdot) - k(\mathbf{x}_i, \cdot)\|_{\mathcal{H}} \\
&\le 1 + C k c_p^2 \quad (:= U).
\end{aligned}$$

In the above, the first inequality is due to the triangle inequality and $\frac{1}{2}|y_t - y_i| \leq 1$. The second inequality is due to the Cauchy-Schwarz inequality. The third inequality is due to the bound on the decision function in Lemma 1 and Eq. (A.1). □

Proof of Theorem 1

Proof: Let f^* be the optimal solution from hindsight. We define the distance between f_t and f^* at the t-th trial as $\|f_t - f^*\|_{\mathcal{H}}$. Then we have

$$\begin{aligned}&\|f_{t+1} - f^*\|_{\mathcal{H}}^2 - \|f_t - f^*\|_{\mathcal{H}}^2\\ =&\|f_t - \eta\partial\hat{\mathcal{L}}_t(f_t) - f^*\|_{\mathcal{H}}^2 - \|f_t - f^*\|_{\mathcal{H}}^2\\ =&\eta^2\|\partial\hat{\mathcal{L}}_t(f_t)\|_{\mathcal{H}}^2 - 2\eta\langle\partial\hat{\mathcal{L}}_t(f_t), f_t - f^*\rangle_{\mathcal{H}}.\end{aligned}$$

By summing over $t = 1, \ldots, T$, we have

$$\begin{aligned}&\|f_{T+1} - f^*\|_{\mathcal{H}}^2 - \|f_1 - f^*\|_{\mathcal{H}}^2\\ =&-2\eta\sum_{t=1}^{T}\langle\partial\hat{\mathcal{L}}_t(f_t), f_t - f^*\rangle_{\mathcal{H}} + \eta^2\sum_{t=1}^{T}\|\partial\hat{\mathcal{L}}_t(f_t)\|_{\mathcal{H}}^2.\end{aligned}$$

Due to the convexity of $\hat{\mathcal{L}}_t(f_t)$, we have

$$\begin{aligned}R_T &\leq \sum_{t=1}^{T}\langle\partial\hat{\mathcal{L}}_t(f_t), f_t - f^*\rangle_{\mathcal{H}}\\ &\leq \frac{\|f_1 - f^*\|_{\mathcal{H}}^2}{2\eta} - \frac{\|f_{T+1} - f^*\|_{\mathcal{H}}^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\|\partial\hat{\mathcal{L}}_t(f_t)\|_{\mathcal{H}}^2.\end{aligned}$$

Since $f_1 = 0$ and $\|f_{T+1} - f^*\|_{\mathcal{H}}^2 \geq 0$, we have

$$R_T \leq \frac{\|f^*\|_{\mathcal{H}}^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \|\partial \hat{\mathcal{L}}_t(f_t)\|_{\mathcal{H}}^2.$$

We now bound $\|\partial \hat{\mathcal{L}}_t(f_t)\|_{\mathcal{H}}^2$:

$$\begin{aligned}
&\|\partial \hat{\mathcal{L}}_t(f_t)\|_{\mathcal{H}}^2 \\
=&\left\| f_t - C \sum_{\mathbf{z}_i} \mathbb{I}[\ell_h(f, \mathbf{z}_t, \mathbf{z}_i) > 0] \cdot \varphi(\mathbf{z}_t, \mathbf{z}_i) \right\|_{\mathcal{H}}^2 \\
=&\|f_t\|_{\mathcal{H}}^2 + \left\| C \sum_{\mathbf{z}_i} \mathbb{I}[\ell_h(f, \mathbf{z}_t, \mathbf{z}_i) > 0] \cdot \varphi(\mathbf{z}_t, \mathbf{z}_i) \right\|_{\mathcal{H}}^2 \\
&- 2C \sum_{\mathbf{z}_i} \mathbb{I}[\ell_h(f, \mathbf{z}_t, \mathbf{z}_i) > 0] \langle f_t, \varphi(\mathbf{z}_t, \mathbf{z}_i) \rangle_{\mathcal{H}}.
\end{aligned} \tag{A.3}$$

From Lemma 1, we know that the first term above is bounded by

$$\|f_t\|_{\mathcal{H}}^2 \leq C^2 k^2 c_p^2. \tag{A.4}$$

Now, by Eq. (3.21), we have

$$\begin{aligned}
&\left\| C \sum_{\mathbf{z}_i} \mathbb{I}[\ell_h(f, \mathbf{z}_t, \mathbf{z}_i) > 0] \cdot \varphi(\mathbf{z}_t, \mathbf{z}_i) \right\|_{\mathcal{H}}^2 \\
\leq & C^2 \sum_{\mathbf{z}_i} \mathbb{I}[\ell_h(f, \mathbf{z}_t, \mathbf{z}_i) > 0] \cdot \|\varphi(\mathbf{z}_t, \mathbf{z}_i)\|_{\mathcal{H}}^2 \\
\leq & C^2 k c_p^2.
\end{aligned} \tag{A.5}$$

The above inequalities hold due to the triangle inequality, the bound in Eq. (A.1), and the number of elements in $N_{t,k}^{-y_t}(\mathbf{z}_t)$ bounded by k.

Next, we bound the third term of Eq.(A.3). First, using the facts that the decision function is an element of a Reproducing Kernel Hilbert Space (RKHS) and the pairwise loss function $\ell_h : \mathcal{H} \times \mathcal{Z} \times \mathcal{Z} \to [0, U]$ is bounded, we have

$$\langle f_t, \varphi(\mathbf{z}_t, \mathbf{z}_i) \rangle_{\mathcal{H}} = \frac{1}{2}(y_t - y_i)\left(f_t(\mathbf{x}_t) - f_t(\mathbf{x}_i)\right) \geq (1 - U). \quad \text{(A.6)}$$

Hence, we have

$$\begin{aligned} & -2C \sum_{\mathbf{z}_i} \mathbb{I}[\ell_h(f, \mathbf{z}_t, \mathbf{z}_i) > 0] \langle f_t, \varphi(\mathbf{z}_t, \mathbf{z}_i) \rangle_{\mathcal{H}} \\ \leq\ & 2C \sum_{\mathbf{z}_i} \mathbb{I}[\ell_h(f, \mathbf{z}_t, \mathbf{z}_i) > 0](U - 1) \\ \leq\ & 2Ck(U - 1). \end{aligned} \quad \text{(A.7)}$$

In the above, the first inequality follows from Eq. (A.6). The second inequality holds since the number of elements in $N_{t,k}^{-y_t}(\mathbf{z}_t)$ is at most k.

By combining Eq. (A.4), Eq. (A.5), and Eq. (A.7), we have a bound for Eq. (A.3). That is,

$$\|\partial \hat{\mathcal{L}}_t(f_t)\|_{\mathcal{H}}^2 \leq C^2 k(k+1)c_p^2 + 2Ck(U - 1).$$

We then obtain the bound on R_T in Eq. (3.24) by summing $\partial \hat{\mathcal{L}}_t(f_t)$ over all $t \in [T]$.

$\square$

Proof of Theorem 2

Proof: The proof is similar to that of Theorem 1. The main difference is to exploit the smoothness of the loss function.

First, by the convexity of the objective function in Eq. (3.14), we have

$$\tilde{\mathcal{L}}_t(\tilde{f}_t) - \tilde{\mathcal{L}}_t(\tilde{f}^*) \leq \langle \partial \tilde{\mathcal{L}}_t(\tilde{f}_t), \tilde{f}_t - \tilde{f}^* \rangle. \tag{A.8}$$

We can derive the second derivative of Eq. (3.14) with respect to f as follows:

$$\frac{\partial^2 \tilde{\mathcal{L}}_t}{\partial \tilde{f}^2} = I + 2C \sum_{\mathbf{z}_i, \mathbf{z}_j} \mathbb{I}_{\mathbf{z}_i} \mathbb{I}_{\mathbf{z}_j} \varphi(\mathbf{z}_t, \mathbf{z}_i) \varphi(\mathbf{z}_t, \mathbf{z}_j)^\top, \tag{A.9}$$

where $\mathbb{I}_{\mathbf{z}_i}$ is defined by $\mathbb{I}[\ell_h(\tilde{f}, \mathbf{z}_t, \mathbf{z}_i) > 0]$ for simplicity. Hence, we have

$$\|\partial \tilde{\mathcal{L}}_t(\tilde{f}_t) - \partial \tilde{\mathcal{L}}_t(\tilde{f}^*)\|_{\mathcal{H}} \leq (1+\zeta) \|\tilde{f}_t - \tilde{f}^*\|_{\mathcal{H}}, \tag{A.10}$$

where $\zeta = 2Ck^2c_p^2$ is obtained by the summation in Eq. (A.9) and the following fact

$$\langle \varphi(\mathbf{z}_t, \mathbf{z}_i), \varphi(\mathbf{z}_t, \mathbf{z}_j) \rangle_{\mathcal{H}} \leq \|\varphi(\mathbf{z}_t, \mathbf{z}_i)\|_{\mathcal{H}} \cdot \|\varphi(\mathbf{z}_t, \mathbf{z}_j)\|_{\mathcal{H}} \leq c_p^2.$$

The optimality of $\tilde{f}^*$ implies that $\partial \tilde{\mathcal{L}}_t(\tilde{f}^*) = 0$ for convex and smooth $\tilde{\mathcal{L}}$. Based on [34, Theorem 2.1.5], we have

$$\|\partial \tilde{\mathcal{L}}_t(\tilde{f}_t)\|_{\mathcal{H}}^2 = \|\partial \tilde{\mathcal{L}}_t(\tilde{f}_t) - \partial \tilde{\mathcal{L}}_t(\tilde{f}^*)\|_{\mathcal{H}}^2 \leq 2(1+\zeta) \tilde{\mathcal{L}}_t(\tilde{f}_t), \tag{A.11}$$

where the inequality holds by $\tilde{\mathcal{L}}_t(\tilde{f}^*) \geq 0$ and $\partial \tilde{\mathcal{L}}_t(\tilde{f}^*) = 0$.

Moreover, we have

$$\begin{aligned} &\|\tilde{f}_{t+1} - \tilde{f}^*\|_{\mathcal{H}}^2 - \|\tilde{f}_t - \tilde{f}^*\|_{\mathcal{H}}^2 \\ =&\eta^2 \|\partial \tilde{\mathcal{L}}_t(\tilde{f}_t)\|_{\mathcal{H}}^2 - 2\eta \langle \partial \tilde{\mathcal{L}}_t(\tilde{f}_t), \tilde{f}_t - \tilde{f}^* \rangle_{\mathcal{H}}. \end{aligned} \tag{A.12}$$

By combining Eq. (A.8), Eq. (A.12), and Eq. (A.11), we obtain

$$(1 - (1+\zeta)\eta) \tilde{\mathcal{L}}_t(\tilde{f}_t) - \tilde{\mathcal{L}}_t(\tilde{f}^*) \leq \frac{\|\tilde{f}_t - \tilde{f}^*\|_{\mathcal{H}}^2 - \|\tilde{f}_{t+1} - \tilde{f}^*\|_{\mathcal{H}}^2}{2\eta}.$$

Summing over $t = 1, \dots, T$ and rearranging, we obtain

$$\begin{aligned}&\sum_{t=1}^{T}(1-(1+\zeta)\eta)\tilde{\mathcal{L}}_t(\tilde{f}_t) - \sum_{t=1}^{T}\tilde{\mathcal{L}}_t(\tilde{f}^*)\\ \le&\frac{1}{2\eta}\left(\|\tilde{f}_1 - \tilde{f}^*\|_{\mathcal{H}}^2 - \|\tilde{f}_{T+1} - \tilde{f}^*\|_{\mathcal{H}}^2\right) \le \frac{1}{2\eta}\|\tilde{f}^*\|_{\mathcal{H}}^2.\end{aligned}$$

In the above, the second inequality is due to $\tilde{f}_1 = 0$ and $\|\tilde{f}_{T+1} - \tilde{f}^*\|_{\mathcal{H}}^2 \ge 0$.

Hence, we further get

$$\begin{aligned}&\sum_{t=1}^{T}\tilde{\mathcal{L}}_t(\tilde{f}_t) - \sum_{t=1}^{T}\tilde{\mathcal{L}}_t(\tilde{f}^*)\\ \le&\frac{1}{1-(1+\zeta)\eta}\left(\frac{1}{2\eta}\|\tilde{f}^*\|_{\mathcal{H}}^2 + ((1+\zeta)\eta)\sum_{t=1}^{T}\tilde{\mathcal{L}}_t(\tilde{f}^*)\right)\\ \le&\frac{1}{1-(1+\zeta)\eta}\left(\frac{1}{2\eta}\|\tilde{f}^*\|_{\mathcal{H}}^2 + ((1+\zeta)\eta)TL^*\right). \qquad (A.13)\end{aligned}$$

The minimum of the last inequality is obtained by putting

$$\eta = \frac{1}{1+\zeta+\sqrt{(1+\zeta)^2 + 2(1+\zeta)TL^*/\|f^*\|_{\mathcal{H}}^2}}. \qquad (A.14)$$

Putting η in Eq. (A.14) into Eq. (A.13) and using the following power inequality, $\sqrt{(1+\zeta)^2 + 2(1+\zeta)TL^*/\|f^*\|_{\mathcal{H}}^2} \le (1+\zeta) + \sqrt{2(1+\zeta)TL^*/\|f^*\|_{\mathcal{H}}^2}$, we obtain the regret bound in Eq. (3.27).
□

Proof of Theorem 3

Proof: Let denote m_i^t as an indicator valuable for the selection of the i-th kernel classifier at the t-th trial, i.e., $m_i^t =$

BernSample($\mathbf{w}_i^t/[\max_j \mathbf{w}_j^t]$). In Algorithm 4, we have the update rule for the weight $\mathbf{w}^{T+1}$ as follows

$$\begin{aligned} w_i^{T+1} &= w_i^T \exp(-\eta m_i^T \check{\mathcal{L}}_T(f_{i,T})) = w_i^1 \exp(-\eta \sum_{t=1}^{T} m_i^t \check{\mathcal{L}}_t(f_{i,t})) \\ &= \exp(-\eta \sum_{t=1}^{T} m_i^t \check{\mathcal{L}}_t(f_{i,t})). \end{aligned} \tag{A.15}$$

We first derive the lower bound for $\sum_{t=1}^{T} \ln \frac{W^{t+1}}{W^t}$.

$$\begin{aligned} \sum_{t=1}^{T} \ln \frac{W^{t+1}}{W^t} &= \ln \frac{W^{T+1}}{W^1} = \ln \frac{\sum_{i=1}^{m} w_i^{T+1}}{m} \\ &\geq \ln \left(\prod_{i=1}^{m} w_i^{T+1} \right)^{\frac{1}{m}} = -\frac{\eta}{m} \sum_{i=1}^{m} \sum_{t=1}^{T} m_i^t \check{\mathcal{L}}_t(f_{i,t}) \end{aligned} \tag{A.16}$$

Then we derive the upper bound for $\sum_{t=1}^{T} \ln \frac{W^{t+1}}{W^t}$ by using the inequality $e^{-x} \leq 1 - x + \frac{1}{2}x^2, \forall x \geq 0$ and $\ln(1+x) \leq x$,

$$\begin{aligned} \ln \frac{W^{t+1}}{W^t} &= \ln \sum_{i=1}^{m} w_i^t \exp(-\eta \sum_{t=1}^{T} m_i^t \check{\mathcal{L}}_t(f_{i,t})) \\ &\leq \ln \sum_{i=1}^{m} q_i^t \left(1 - \eta m_i^t \check{\mathcal{L}}_t(f_{i,t}) + \frac{1}{2}\eta^2 (m_i^t \check{\mathcal{L}}_t(f_{i,t}))^2 \right) \\ &\leq -\eta \sum_{i=1}^{m} q_i^t m_i^t \check{\mathcal{L}}_t(f_{i,t}) + \frac{1}{2}\eta^2 \sum_{i=1}^{m} q_i^t (m_i^t \check{\mathcal{L}}_t(f_{i,t}))^2 \end{aligned} \tag{A.17}$$

By taking the summation on both sides we have

$$\sum_{t=1}^{T} \ln \frac{W^{t+1}}{W^t} \leq -\eta \sum_{t=1}^{T} \sum_{i=1}^{m} q_i^t m_i^t \check{\mathcal{L}}_t(f_{i,t}) + \frac{1}{2}\eta^2 \sum_{t=1}^{T} \sum_{i=1}^{m} q_i^t (m_i^t \check{\mathcal{L}}_t(f_{i,t}))^2 \tag{A.18}$$

Combing Eq. (A.16) and (A.18), we have

$$\sum_{t=1}^{T}\sum_{i=1}^{m} q_i^t m_i^t \check{\mathcal{L}}_t(f_{i,t}) \leq \frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{T} m_i^t \check{\mathcal{L}}_t(f_{i,t}) + \frac{1}{2}\eta\sum_{t=1}^{T}\sum_{i=1}^{m} q_i^t (m_i^t L)^2, \tag{A.19}$$

where we use upper bound $|\check{\mathcal{L}}_t(f_{i,t})| \leq L$ to bound the second order term $\check{\mathcal{L}}_t^2(f_{i,t})$. Since $\sum_{i=1}^{m} q_i^t = 1$, $E[m_i^t] \leq 1$ and $E[(m_i^t)^2] \leq 1$, we bound the first order term $\check{\mathcal{L}}_t(f_{i,t})$ in Eq. (A.19) by $\check{\mathcal{L}}_t(f), \forall f \in \mathcal{H}_{k_i}, i \in [1:m]$ as follows.

$$\begin{aligned}
& m_i^t(\check{\mathcal{L}}_t(f_{i,t}) - \check{\mathcal{L}}_t(f)) \\
\leq\ & \left\langle f_i^t - f, m_i^t \partial_f \check{\mathcal{L}}_t(f_{i,t}) \right\rangle \\
\leq\ & \frac{1}{2\lambda}\left(\|f_i^t - f\|^2 - \|f_i^{t+1} - f\|^2 + (m_i^t \partial_f \check{\mathcal{L}}_t(f_{i,t})\lambda)^2 \right) \\
\leq\ & \frac{1}{2\lambda}\left(\|f_i^t - f\|^2 - \|f_i^{t+1} - f\|^2 + (m_i^t G\lambda)^2 \right)
\end{aligned} \tag{A.20}$$

Then we take the expectation of Eq. (A.20), we have

$$E\left[\frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{T} m_i^t \check{\mathcal{L}}_t(f_{i,t})\right] \leq \min_{1\leq i\leq m}\min_{f\in\mathcal{H}_{k_i}} \sum_{t=1}^{T} \check{\mathcal{L}}_t(f)) + \frac{\|f\|_{\mathcal{H}_{k_i}}}{2\lambda} + \frac{T\lambda G}{2} \tag{A.21}$$

Then we take the expectation of the second term in the right hand side of Eq. (A.19), we have

$$E\left[\frac{1}{2}\eta\sum_{t=1}^{T}\sum_{i=1}^{m} q_i^t (m_i^t L)^2\right] \leq \frac{\eta T L^2}{2} \tag{A.22}$$

Combine Eq. (A.21) and (A.22), we complete the proof. □

□ **End of chapter.**

Bibliography

[1] A. 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.

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

[3] C. L. Castro and A. de Pádua Braga. Novel cost-sensitive approach to improve the multilayer perceptron performance on imbalanced data. *IEEE Trans. Neural Netw. Learning Syst.*, 24(6):888–899, 2013.

[4] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile. Tracking the best hyperplane with a simple budget perceptron. *Machine Learning*, 69(2-3):143–167, 2007.

[5] N. Cesa-Bianchi, A. Conconi, and C. Gentile. A second-order perceptron algorithm. *SIAM J. Comput.*, 34(3):640–668, 2005.

[6] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, New York, NY, USA, 2006.

[7] C. Cortes and M. Mohri. AUC optimization vs. error rate minimization. In *NIPS*. MIT Press, 2003.

[8] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

[9] L. Csató and M. Opper. Sparse on-line gaussian processes. *Neural Computation*, 14(3):641–668, 2002.

[10] O. Dekel, S. Shalev-Shwartz, and Y. Singer. The Forgetron: A kernel-based perceptron on a budget. *SIAM Journal on Computing*, 37(5):1342–1372, 2008.

[11] Y. Ding, P. Zhao, S. C. H. Hoi, and Y. Ong. An adaptive gradient method for online AUC maximization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 2568–2574, 2015.

[12] G. Dror, N. Koenigstein, Y. Koren, and M. Weimer. The yahoo! music dataset and kdd-cup '11. In *Proceedings of KDD Cup 2011 competition, San Diego, CA, USA, 2011*, pages 8–18, 2012.

[13] Y. Engel, S. Mannor, and R. Meir. The kernel recursive least-squares algorithm. *IEEE Transactions on Signal Processing*, 52(8):2275–2285, 2004.

[14] W. Fan and A. Bifet. Mining big data: current status, and forecast to the future. *SIGKDD Explorations*, 14(2):1–5, 2012.

[15] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[16] Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

[17] W. Gao, R. Jin, S. Zhu, and Z.-H. Zhou. One-pass AUC optimization. In *ICML*, pages 906–914, 2013.

[18] I. Guyon, V. Lemaire, M. Boullé, G. Dror, and D. Vogel. Design and analysis of the KDD cup 2009: fast scoring on a large orange customer database. *SIGKDD Explorations*, 11(2):68–76, 2009.

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

[20] J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In *Proceedings of the Twenty-*

Ninth Conference on Uncertainty in Artificial Intelligence, Bellevue, WA, USA, August 11-15, 2013, 2013.

[21] A. Herschtal and B. Raskutti. Optimising area under the ROC curve using gradient descent. In *ICML*, 2004.

[22] S. C. H. Hoi, R. Jin, P. Zhao, and T. Yang. Online multiple kernel classification. *Machine Learning*, 90(2):289–316, 2013.

[23] J. Hu, H. Yang, I. King, M. R. Lyu, and A. M.-C. So. Kernelized online imbalanced learning with fixed budgets. In *AAAI*, Austin Texss, USA, Jan. 25-30 2015. AR = 531/1991 (26.67%).

[24] R. Jin, S. C. H. Hoi, and T. Yang. Online multiple kernel learning: Algorithms and mistake bounds. In *Algorithmic Learning Theory, 21st International Conference, ALT 2010, Canberra, Australia, October 6-8, 2010. Proceedings*, pages 390–404, 2010.

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

[26] P. Kar, B. K. Sriperumbudur, P. Jain, and H. Karnick. On the generalization ability of online learning algorithms for pairwise loss functions. In *Proceedings of the 30th International Conference on Machine Learning, ICML*

2013, Atlanta, GA, USA, 16-21 June 2013, pages 441–449, 2013.

[27] N. Karampatziakis and J. Langford. Online importance weight aware updates. In *UAI*, pages 392–399, 2011.

[28] S. S. Keerthi and W. Chu. A matching pursuit approach to sparse gaussian process regression. In *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*, pages 643–650, 2005.

[29] J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8):2165–2176, 2004.

[30] M. Kloft, U. Brefeld, S. Sonnenburg, and A. Zien. l_p-norm multiple kernel learning. *Journal of Machine Learning Research*, 12:953–997, 2011.

[31] Y. Li and P. M. Long. The relaxed online maximum margin algorithm. *Machine Learning*, 46(1-3):361–387, 2002.

[32] M. Lin, K. Tang, and X. Yao. Dynamic sampling approach to training neural networks for multiclass imbalance classification. *IEEE Trans. Neural Netw. Learning Syst.*, 24(4):647–660, 2013.

[33] G. Loomes and R. Sugden. Regret theory: An alternative theory of rational choice under uncertainty. *The economic journal*, pages 805–824, 1982.

[34] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers, 2003.

[35] F. Orabona, J. Keshet, and B. Caputo. Bounded kernel-based online learning. *Journal of Machine Learning Research*, 10:2643–2666, 2009.

[36] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet. SimpleMKL. *Journal of Machine Learning Research*, 9:1179–1225, 2008.

[37] F. Rosenblatt. The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[38] S. Ross, P. Mineiro, and J. Langford. Normalized online learning. *CoRR*, abs/1305.6646, 2013.

[39] D. Sahoo, S. C. H. Hoi, and B. Li. Online multiple kernel regression. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 293–302, 2014.

[40] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[41] M. W. Seeger, C. K. I. Williams, and N. D. Lawrence. Fast forward selection to speed up sparse gaussian process regression. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics, AISTATS 2003, Key West, Florida, USA, January 3-6, 2003*, 2003.

[42] S. Smale and Y. Yao. Online learning algorithms. *Foundations of Computational Mathematics*, 6(2):145–170, April 2006.

[43] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.

[44] S. J. Stolfo, W. Lee, P. K. Chan, W. Fan, and E. Eskin. Data mining-based intrusion detectors: An overview of the columbia IDS project. *SIGMOD Record*, 30(4):5–14, 2001.

[45] S. V. Vaerenbergh, M. Lázaro-Gredilla, and I. Santamaría. Kernel recursive least-squares tracker for time-varying regression. *IEEE Trans. Neural Netw. Learning Syst.*, 23(8):1313–1326, 2012.

[46] J. S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, Mar. 1985.

[47] Y. Wang, R. Khardon, D. Pechyony, and R. Jones. Generalization bounds for online learning algorithms with pairwise loss functions. In *COLT 2012 - The 25th*

Annual Conference on Learning Theory, June 25-27, 2012, Edinburgh, Scotland, pages 13.1–13.22, 2012.

[48] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. F. M. Ng, B. Liu, P. S. Yu, Z. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg. Top 10 algorithms in data mining. *Knowl. Inf. Syst.*, 14(1):1–37, 2008.

[49] H. Xia, S. C. H. Hoi, R. Jin, and P. Zhao. Online multiple kernel similarity learning for visual search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(3):536–549, 2014.

[50] Z. Xu, R. Jin, H. Yang, I. King, and M. R. Lyu. Simple and efficient multiple kernel learning by group lasso. In *ICML*, pages 1175–1182, Haifa, Israel, 2010.

[51] L. Yan, R. H. Dodier, M. Mozer, and R. H. Wolniewicz. Optimizing classifier performance via an approximation to the wilcoxon-mann-whitney statistic. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 848–855, 2003.

[52] H. Yang, J. Hu, M. R. Lyu, and I. King. Online imbalanced learning with kernels. In *NIPS Workshop on Big Learning*, Lake Tahoe, USA, 2013.

[53] H. Yang, J. Hu, M. R. Lyu, and I. King. Online imbalanced learning with kernels. In *NIPS Workshop on Big Learning*, Dec. 05-10 2013.

[54] H. Yang and I. King. Ensemble learning for imbalanced e-commerce transaction anomaly classification. In *ICONIP*, pages 866–874, Bangkok, Thailand, 2009.

[55] H. Yang, I. King, and M. R. Lyu. *Sparse Learning Under Regularization Framework*. LAP Lambert Academic Publishing, April 2011.

[56] H. Yang, Z. Xu, J. Ye, I. King, and M. R. Lyu. Efficient sparse generalized multiple kernel learning. *IEEE Transactions on Neural Networks*, 22(3):433–446, March 2011.

[57] T. Yang, M. Mahdavi, R. Jin, J. Yi, and S. C. H. Hoi. Online kernel selection: Algorithms and evaluations. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*, 2012.

[58] L. Zhang, J. Yi, R. Jin, M. Lin, and X. He. Online kernel learning with a near optimal sparsity bound. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 621–629, 2013.

[59] P. Zhao, S. C. H. Hoi, and R. Jin. Double updating online learning. *Journal of Machine Learning Research*, 12:1587–1615, 2011.

[60] P. Zhao, S. C. H. Hoi, R. Jin, and T. Yang. Online auc maximization. In *ICML*, pages 233–240, 2011.