Online Nonlinear AUC Maximization for Imbalanced Data Sets

Junjie Hu, Haiqin Yang, *Member, IEEE*, Michael R. Lyu, *Fellow, IEEE*, Irwin King, *Senior Member, IEEE*, and Anthony Man-Cho So, *Member, IEEE*

***Abstract*— Classifying binary imbalanced streaming data is a significant task in both machine learning and data mining. Previously, online area under the receiver operating characteristic (ROC) curve (AUC) maximization has been proposed to seek a linear classifier. However, it is not well suited for handling nonlinearity and heterogeneity of the data. In this paper, we propose the kernelized online imbalanced learning (KOIL) algorithm, which produces a nonlinear classifier for the data by maximizing the AUC score while minimizing a functional regularizer. We address four major challenges that arise from our approach. First, to control the number of support vectors without sacrificing the model performance, we introduce two buffers with fixed budgets to capture the global information on the decision boundary by storing the corresponding learned support vectors. Second, to restrict the fluctuation of the learned decision function and achieve smooth updating, we confine the influence on a new support vector to its *k*-nearest opposite support vectors. Third, to avoid information loss, we propose an effective compensation scheme after the replacement is conducted when either buffer is full. With such a compensation scheme, the performance of the learned model is comparable to the one learned with infinite budgets. Fourth, to determine good kernels for data similarity representation, we exploit the multiple kernel learning framework to automatically learn a set of kernels. Extensive experiments on both synthetic and real-world benchmark data sets demonstrate the efficacy of our proposed approach.**

***Index Terms*— Area under the ROC curve (AUC) maximization, budget, imbalanced data, kernel.**

I. Introduction

IMBALANCED streaming data are prevalent in various real-world applications, such as network intrusion detection [41], purchasing or clicking analysis for customer relationship [12], [16], and so on. These data exhibit the following prominent characteristics.

1) *Huge Volume:* The volume of the data increases tremendously, from petabyte to exabyte, or even zettabyte.
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 due to the nature of human attention.
4) *Nonlinearity and Heterogeneity:* Only nonlinear classifiers can produce a more accurate decision boundary [see Fig. 1(a) for an example]. The heterogeneity poses difficulty in defining data similarity.

Learning binary classification models from imbalanced data has become an important research topic in both machine learning and data mining [3], [30], [45], [49]. In the literature, researchers aim at maximizing the area under the receiver operating characteristic (ROC) curve (AUC) instead of accuracy, because the AUC score is effective in measuring the performance of classifiers for imbalanced data [1], [2], [17], [19], [23]. To deal with the imbalanced streaming data, researchers have proposed the online AUC maximization approach [15], [55]. However, the resulting algorithms only produce a linear classifier and are not well suited for handling the nonlinearity and heterogeneity of the data.

In this paper, we focus on seeking an online nonlinear classifier with kernels—a less explored but important research topic in the literature. There are three major obstacles to this approach. First, the learned kernel-based estimator becomes more complex as the number of samples increases [27], [52]. Without a suitable stream oblivious strategy, the number of learned support vectors may grow to infinity, which is obviously undesirable for the large-scale applications. In the literature, various refinement techniques have been proposed. They include projection-based methods [9], [13], [32], fixed-budget strategies [4], [10], and sparse kernel learning via weighted sampling [53]. However, extending the above-mentioned methods to tackle the imbalanced data seems to be a nontrivial task. Second, fluctuation due to outliers is unavoidable in online learning [6], [25], [35]. Thus, additional effort is required to achieve smooth updating. Third, the kernel representation is effective in capturing nonlinearity and heterogeneity of the data [21], [27]. However, it is not clear how to effectively determine a good kernel representation.

Manuscript received March 28, 2016; revised September 8, 2016; accepted September 12, 2016. Date of publication January 27, 2017; date of current version March 15, 2018. This work was supported in part by the National Natural Science Foundation of China under Project 61332010, in part by the Research Grants Council of the Hong Kong Special Administrative Region, China under Project CUHK 14203314 and Project CUHK 415113, and in part by the 2015 Microsoft Research Asia Collaborative Research Program under Project FY16-RES-THEME-005. This paper was presented at the 29th AAAI Conference on Artificial Intelligence (AAAI-15) [21]. *(Corresponding author: Haiqin Yang.)*

J. Hu is with the Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA.

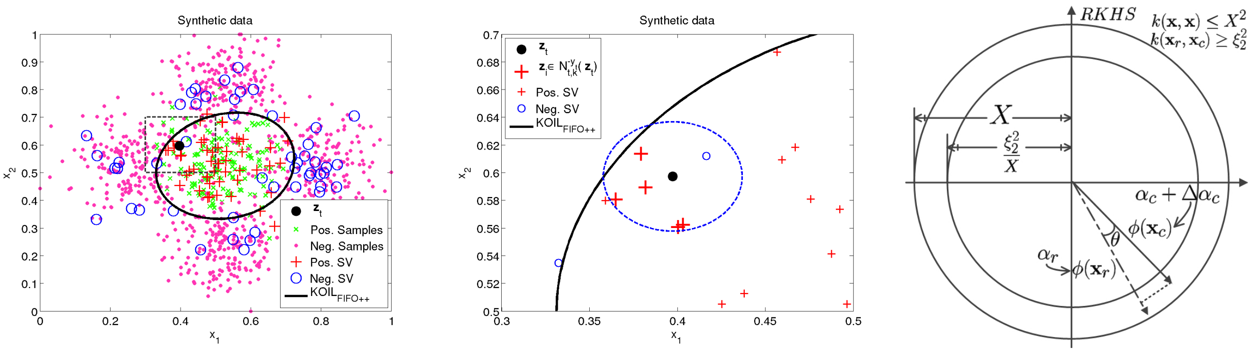
H. Yang is with the Department of Computing, Hang Seng Management College, Hong Kong (e-mail: hqyang@ieee.org).

M. R. Lyu and I. King are with the Shenzhen Key Laboratory of Rich Media Big Data Analytics and Applications, Shenzhen Research Institute, The Chinese University of Hong Kong (CUHK), Hong Kong, and also with the Department of Computer Science and Engineering, CUHK, Hong Kong.

A. Man-Cho So is with the Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2016.2610465



(a) (b) (c)

Fig. 1. Illustration of the KOIL algorithm with the k-nearest neighbor confinement and the extended updating policy on a synthetic data in 2-D space. (a) Decision function in black solid curve, the new instance in big $\bullet$, the positive samples in small $\times$'s, the negative samples in small $\bullet$'s, the positive support vectors in big $+$'s, and the negative support vectors in big $\circ$'s. It is shown that the decision function learned by our proposed KOIL algorithm with the extended first-in-first-out (FIFO) updating policy can classify the data well. (b) Local region of a new instance $\mathbf{z}_t$ and 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. (c) 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. By the two assumptions $k(\mathbf{x},\mathbf{x}) \leq X^2$ and $k(\mathbf{x}_r,\mathbf{x}_c) \geq \xi_2^2$, we have $\|\phi(\mathbf{x}_c)\|_{\mathcal{H}} \cos\theta \geq (\xi_2^2/X)$, where $\phi(\mathbf{x}_c) = k(\mathbf{x}_c,\cdot)$.

To overcome the above obstacles, we propose the kernelized online imbalanced learning (KOIL) algorithm with fixed budgets to achieve online nonlinear AUC maximization. We highlight our contributions as follows.

1) To better control the computational cost, we fix the budget (buffer size) of the buffer for each data class in the KOIL algorithm to store the learned support vectors.
2) We propose a smooth update rule by confining the influence on a new instance to its k-nearest opposite support vectors (see Fig. 1(b) for an example). Our KOIL algorithm can thus limit the effect of outliers.
3) We design an effective scheme to compensate for the loss when a support vector is removed. The idea is to transfer the weight of the removed support vector to its closest support vector in the buffer (see Fig. 1(c) for an illustration). As a result, the learned model typically approaches the one learned with infinite budgets.
4) We exploit the online multiple kernel learning (MKL) framework to automatically determine a good kernel representation. Specifically, we try to learn multiple kernel classifiers and the corresponding linear combination coefficients from a pool of predefined kernels in an online mode. Different from existing online MKL (OMKL) algorithms [22], our KOIL algorithm focuses on the pairwise loss function and discounts the weights of multiple kernel classifiers when there are errors. Empirical results show that OMKL is effective in determining the kernel representation.

II. Related Work

We review some prior work in closely related areas: machine learning from imbalanced data, online learning, and MKL.

Learning from imbalanced data is an important task in machine learning and data mining [3], [30], [45]. Some algorithms have been developed to train classifiers by maximizing the AUC metric, such as Wilcoxon–Mann–Whitney statistic optimization [48] and RankOpt [19]. Some investigations extend support vector machine (SVM) to optimize the AUC metric [2]. A general framework for optimizing multivariate nonlinear performance measures, such as AUC and F1, is proposed in [23]. Cost-sensitive multilayer perceptron (MLP) is also proposed to improve the discrimination ability of MLPs [3]. One major weakness of these methods is that they train the model in the batch mode, which is inefficient when new training samples appear sequentially.

Online learning algorithms are important as they can adaptively update the models based on the new training samples. The oldest and most well-known online learning algorithm is the perceptron [34]. Many variants have been proposed in the literature [5], [14]. Some are inspired by the maximum margin principle [8], [29], [54]. To learn from imbalanced data, algorithms for online AUC maximization are proposed in [11], [15], and [55]. Several works have established the generalization error bounds for online learning algorithms with pairwise loss functions [24], [44]. However, these algorithms only focus on linear classifiers, which are not sufficient to capture the heterogeneity and nonlinearity embedded in the data [50], [51]. In the literature, various kernel-based online learning algorithms have been proposed. They include online learning algorithms in a reproducing kernel Hilbert space (RKHS) [10], [27], [32], [39], online Gaussian process [9], [18], [26], [38], kernelized recursive least-square algorithms [13], [42], and so on. A key challenge in online learning with kernels is that the computational complexity scales with the number of training samples. To tackle this challenge, various strategies have been proposed, including projection-based methods [9], [13], [26], [32], fixed-budget strategies [4], [10], and sparse kernel learning via weighted sampling [53]. However, these strategies aim at directly maximizing the accuracy and cannot handle the task of imbalanced learning properly. Some other methods adopt the strategy of projecting or deleting support vectors to maintain the buffer size [9], [32], [42].

However, such strategy could lead to unbounded support vectors or information loss.

The MKL framework is a well-known and effective tool for kernel learning. It aims to combine multiple kernels by optimizing the performance of the kernel-based learning methods (e.g., support vector machine) [33], [40]. To attain good model performance, MKL with different norm regularizers have been proposed [28], [47], [51]. Recently, OMKL has been proposed to simultaneously learn multiple kernel classifiers and the corresponding coefficients from a pool of predefined kernels in an online mode [20], [22]. Similar ideas have been applied to solve problems in image search and regression [36], [46]. However, existing algorithms do not consider the task of imbalanced learning.

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

III. KOIL for AUC Maximization

A. Notations and Problem Definition

Throughout this paper, bold small letter, e.g., $\mathbf{x}$, denotes a vector. Letter in calligraphic font, e.g., $\mathcal{X}$, indicates a set. We use $\mathbb{R}^d$ to denote a 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 are interested in the imbalanced binary classification problem, where our goal is 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 the 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 $\mathbf{z}$ and have the label of $\tilde{y}$ at the tth 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. We define the index sets I_t^+ and I_t^- to record the indices of the positive and negative support vectors at the tth trial, respectively. Moreover, for simplicity, we define two buffers $\mathcal{K}_t^+$ and $\mathcal{K}_t^-$ to store the learned information, namely, the weight and support vector, from the two classes at the tth trial, respectively

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

Here, $\alpha_{i,t}$ denotes the weight of the support vector that first occurred at the ith trial and updated at the tth trial. We fix the budgets (the buffer sizes) to be the same, i.e., $|I_t^+| = |I_t^-| = N$ for all t.

At the tth trial, our proposed KOIL algorithm computes a decision function f_t of the form

$$f_t(\mathbf{x}) = \sum_{i \in I_t^+} \alpha_{i,t}^+ k(\mathbf{x}_i, \mathbf{x}) + \sum_{j \in I_t^-} \alpha_{j,t}^- k(\mathbf{x}_j, \mathbf{x}) \qquad (1)$$

where $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a predefined kernel [27]. The corresponding weights and support vectors are stored in $\mathcal{K}_t^+$ and $\mathcal{K}_t^-$, respectively. Then, given a new instance $\mathbf{x}$, we can predict its class by $\operatorname{sgn}(f_t(\mathbf{x}))$, where f_t encodes the nonlinearity and heterogeneity of the data and is generally an element of an RKHS $\mathcal{H}$, i.e., $f_t(\mathbf{x}) = \langle f_t(\cdot), k(\mathbf{x}, \cdot)\rangle_{\mathcal{H}}$ for all $\mathbf{x} \in \mathcal{X}$, where $k(\mathbf{x}, \cdot) \in \mathcal{H}$ [37]. In the following, we will motivate and describe our strategy for updating f_t.

B. Learning With Kernels for AUC Maximization

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

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

where $\mathbb{I}[\pi]$ is the indicator function, i.e., $\mathbb{I}[\pi] = 1$ when π is true and $\mathbb{I}[\pi] = 0$, otherwise. It is clear that maximizing $AUC(f)$ is equivalent to minimizing $\sum_{i \in I^+} \sum_{j \in I^-} \mathbb{I}[f(\mathbf{x}_i) - f(\mathbf{x}_j) \leq 0]$. Since the direct maximization of the AUC score is an NP-hard problem [7], the indicator function is usually replaced by a convex surrogate, which may yield suboptimal performance. A widely used surrogate is the pairwise hinge loss function [15], [55]

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

where $[v]_+ = \max\{0, v\}$. This gives rise to the problem of regularized minimization as follows:

$$\mathcal{L}(f) = \frac{1}{2}\|f\|_{\mathcal{H}}^2 + C \sum_{i \in I^+} \sum_{j \in I^-} \ell_h(f, \mathbf{z}_i, \mathbf{z}_j). \qquad (3)$$

Here, $(1/2)\|f\|_{\mathcal{H}}^2$ is a regularization term that controls the functional complexity and $C > 0$ is a penalty parameter associated with the training errors.

C. Online AUC Maximization by KOIL

Following the derivation in [21], we update the kernel decision function by minimizing the following *localized instantaneous regularized risk of AUC* associated with the arrival of a new instance $\mathbf{z}_t$:

$$\hat{\mathcal{L}}_t(f) := \hat{\mathcal{L}}(f, \mathbf{z}_t) = \frac{1}{2}\|f\|_{\mathcal{H}}^2 + C \sum_{\mathbf{z}_i \in N_{t,k}^{-y_t}(\mathbf{z}_t)} \ell_h(f, \mathbf{z}_t, \mathbf{z}_i) \qquad (4)$$

where k is a predefined constant. Two remarks are in order as follows.

1) The risk defined in (4) measures the pairwise losses between $\mathbf{z}_t$ and its k-nearest opposite support vectors in

Algorithm 1 KOIL With Fixed Budgets

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

Algorithm 2 UpdateKernel

```
Input:
    • newly received sample with label $\mathbf{z}_t$
    • $\mathcal{K}^{-y_t}$ and $\mathcal{K}^{y_t}$ for support vectors with the opposite
      label to $\mathbf{z}_t$ and the same label as $\mathbf{z}_t$, respectively
    • penalty parameter $C$, learning rate $\eta$, and number of
      the nearest neighbors $k$
Output: updated $\mathcal{K}^{-y_t}$, $\mathcal{K}^{y_t}$ and weight $\alpha_{t,t}$ for $\mathbf{z}_t$
Initialize: $V_t = \emptyset$, compute $f_{t-1}$ by Eq. (1)
for $i \in I_t^{-y_t}$ do
    if $1 > y_t(f_{t-1}(\mathbf{x}_t) - f_{t-1}(\mathbf{x}_i))$ then
        $V_t = V_t \cup \{i\}$
    end if
end for
if $|V_t| > k$ then
    $Sim(i) = k(\mathbf{x}_t, \mathbf{x}_i), \quad \forall i \in V_t$
    $[Sim', idx] = \text{Sort}(Sim, \text{‘descend’})$
    $idx_k = idx(1:k)$
    $V_t = V_t(idx_k)$
end if
update $\alpha_{i,t}$ by Eq. (8)
return $\mathcal{K}^{-y_t}, \mathcal{K}^{y_t}, \alpha_{t,t}$
```

Algorithm 3 UpdateBuffer–RS++

```
Input:
    • received sample $\mathbf{z}_t$ and its weight $\alpha_t$
    • buffer $\mathcal{K}$ to be updated
    • buffer size $N$
    • number of instances received until trial $t$, $N_t$
Output: updated buffer $\mathcal{K}$
if $|\mathcal{K}.\mathcal{B}| < N$ then
    $\mathcal{K}.\mathcal{A} = \mathcal{K}.\mathcal{A} \cup \{\alpha_t\}$, $\mathcal{K}.\mathcal{B} = \mathcal{K}.\mathcal{B} \cup \{\mathbf{z}_t\}$
else
    sample $Z$ from a Bernoulli distribution with $\Pr(Z = 1) = N/N_t$
    if $Z = 1$ then
        uniformly select an instance $\mathbf{z}_r$
        update $\mathcal{K}.\mathcal{A}$: $\mathcal{K}.\mathcal{A} = \mathcal{K}.\mathcal{A} \setminus \{\alpha_{r,t}\} \cup \{\alpha_{t,t}\}$
        update $\mathcal{K}.\mathcal{B}$: $\mathcal{K}.\mathcal{B} = \mathcal{K}.\mathcal{B} \setminus \{\mathbf{z}_r\} \cup \{\mathbf{z}_t\}$
    else
        $\mathbf{z}_r = \mathbf{z}_t$, $\alpha_{r,t} = \alpha_{t,t}$
    end if
    find $\mathbf{z}_c = \arg\max_{\mathbf{z}_i \in \mathcal{K}.\mathcal{B}} \{k(\mathbf{x}_r, \mathbf{x}_i)\}$
    set $\alpha_{c,t} = \alpha_{c,t} + \alpha_{r,t}$ and update $\alpha_{c,t}$ in $\mathcal{K}.\mathcal{A}$
end if
return $\mathcal{K}$
```

the buffer. This can resolve the scalability issue and is different from NORMA [27], whose risk only measures the predictive error of the new instance.

2) The advantage of our approach is twofold. First, two buffers with relatively large budgets can keep track of the global information on the decision boundary. Second, by considering the k-nearest opposite support vectors of the new instance, we can utilize the local information around the new instance and avoid the fluctuation of the decision function.

Algorithm 1 shows the KOIL framework, which consists of two key components: UpdateKernel (Algorithm 2) and UpdateBuffer (Algorithm 3).

1) UpdateKernel: We apply the gradient descent method to update the decision function at each trial, that is

$$f_t := f_{t-1} - \eta \partial_f \hat{\mathcal{L}}_t(f_{t-1}) \tag{5}$$

where ∂_f is shorthand for $\partial/\partial f$ (the gradient with respect to f) and $\eta \in (0, 1)$ is the learning rate, which can be a constant or decreases with the number of trials, as long as it guarantees descent, i.e., $\hat{\mathcal{L}}_t(f_t) \leq \hat{\mathcal{L}}_t(f_{t-1})$. We initialize $f_0 = 0$. To compute $\partial_f \hat{\mathcal{L}}_t(f)$, we first calculate $\partial_f \ell_h(f, \mathbf{z}_t, \mathbf{z}_i)$ by

$$\partial_f \ell_h(\cdot) = \begin{cases} 0, & \ell_h(f, \mathbf{z}_t, \mathbf{z}_i) = 0 \\ -\varphi(\mathbf{z}_t, \mathbf{z}_i), & \ell_h(f, \mathbf{z}_t, \mathbf{z}_i) > 0 \end{cases} \tag{6}$$

where $\varphi(\mathbf{z}_t, \mathbf{z}_i) = y_t(k(\mathbf{x}_t, \cdot) - k(\mathbf{x}_i, \cdot))$. Using (4) and (6), we then obtain

$$\begin{aligned} &\partial_f \hat{\mathcal{L}}_t(f_{t-1}) \\ &= f_{t-1} - C \sum_{\mathbf{z}_i \in N_{t,k}^{-y_t}(\mathbf{z}_t)} \mathbb{I}[\ell_h(f_{t-1}, \mathbf{z}_t, \mathbf{z}_i) > 0] \varphi(\mathbf{z}_t, \mathbf{z}_i). \end{aligned} \tag{7}$$

Now, define V_t to be the set of indices for which the indicator function in (7) evaluates to 1 (the valid set) and $\overline{V}_t$ to be its complement, that is

$$\begin{aligned} V_t &:= \{i \in I_t^{-y_t} \mid \mathbf{z}_i \in N_{t,k}^{-y_t}(\mathbf{z}_t) \wedge \ell_h(f_{t-1}, \mathbf{z}_t, \mathbf{z}_i) > 0\} \\ \overline{V}_t &:= I_t^{-y_t} \setminus V_t. \end{aligned}$$

It then follows from (5) and (7) that:

$$f_t = (1 - \eta) f_{t-1} + \eta C y_t |V_t| k(\mathbf{x}_t, \cdot) - \eta C y_t \sum_{i \in V_t} k(\mathbf{x}_i, \cdot).$$

In particular, since $I_t^{y_t} = I_{t-1}^{y_t} \cup \{t\}$ and $I_t^{-y_t} = I_{t-1}^{-y_t} = V_t \cup \overline{V}_t$, we see that if f_{t-1} is of the form in (1), then so is f_t as long as we make the following correspondence between the kernel weights at the $(t-1)$st trial and the tth trial:

$$\alpha_{i,t} = \begin{cases} \eta C y_t |V_t|, & i = t \\ (1-\eta)\alpha_{i,t-1} - \eta C y_t, & i \in V_t \\ (1-\eta)\alpha_{i,t-1}, & i \in I_{t-1}^{y_t} \cup \overline{V}_t. \end{cases} \tag{8}$$

It is instructive to take a closer look at the update rule in (8). It divides the data into three classes. The first involves the new instance $\mathbf{z}_t$. In this case, at most k of the opposite support vectors are used in the pairwise loss calculation. This prevents the fluctuation of the decision function. The second involves the k-nearest opposite support vectors of the new instance $\mathbf{z}_t$, i.e., the support vectors in $N_{t,k}^{-y_t}(\mathbf{z}_t)$. In this case, their corresponding weights change by a magnitude of $|\eta C y_t|$, which favors a more balanced updating. The third covers the case where the new instance does not incur errors or the labels of the previously learned support vectors are the same as that of the new instance. The corresponding weights of those previously learned support vectors are then reduced by a factor of $1-\eta$, which is the same as NORMA [27].

2) UpdateBuffer: Since the buffers have a fixed budget, they have to be updated when they are full. Traditional stream oblivious policies, such as FIFO and reservoir sampling (RS) [43], have been adopted in online linear AUC maximization [55] and shown to be effective in that setting. However, these policies will discard support vectors, which could lead to a degradation in the performance of kernel-based online learning algorithms [21].

To avoid information loss, we need to design a more sophisticated compensation scheme. Toward that end, let $\mathbf{z}_r = (\mathbf{x}_r, y_r)$ be the removed support vector. We find the support vector $\mathbf{z}_c = (\mathbf{x}_c, y_c)$ with $y_c = y_r$ in $\mathcal{K}_t^{y_r}$ that is most similar to $\mathbf{z}_r$ and update its corresponding weight to compensate for the loss of information due to the removal of $\mathbf{z}_r$. Specifically, let $\Delta\alpha_{c,t}$ be the updated weight of the compensated support vector $\mathbf{z}_c$. By keeping the track of the change in the value of the decision function, we would like to find $\Delta\alpha_{c,t}$ such that

$$f_t(\mathbf{x}) \approx f_t(\mathbf{x}) - \alpha_{r,t} k(\mathbf{x}_r, \mathbf{x}) + \Delta\alpha_{c,t} \cdot k(\mathbf{x}_c, \mathbf{x}).$$

This suggests that we should set $\Delta\alpha_{c,t} = \alpha_{r,t}((k(\mathbf{x}_r, \mathbf{x}))/(k(\mathbf{x}_c, \mathbf{x}))) \approx \alpha_{r,t}$. Consequently, we propose the following update rule for the compensated version of f_t, which we denote by f_t^{++}:

$$\begin{aligned} f_t^{++} := {} & (1-\eta) f_{t-1}^{++} - \eta \partial_f \hat{\mathcal{L}}_t\left(f_{t-1}^{++}\right) \\ & + \alpha_{r,t}(k(\mathbf{x}_c, \cdot) - k(\mathbf{x}_r, \cdot)). \end{aligned} \tag{9}$$

Here, f_{t-1}^{++} is the compensated decision function from the previous trial. When neither buffer is full, we have $f_t^{++} = f_t$ and the update is done by (5). Ideally, if $k(\mathbf{x}_c, \mathbf{x})$ equals $k(\mathbf{x}_r, \mathbf{x})$, then f_t^{++} incorporates all the learned support vectors and is equivalent to the one learned with infinite budgets.

Algorithm 3 shows the procedure of the extended RS (RS++). Some elaborations are in order as follows.

1) In lines 3–4, if the buffer is not full (i.e., $|\mathcal{K}.\mathcal{B}| < N$), then the new instance becomes a support vector and is directly added into the buffer $\mathcal{K}$.
2) In lines 6–10, if the buffer is full, then RS is performed. Specifically, with probability N/N_t, we update the buffer by randomly replacing one support vector $\mathbf{z}_r$ in $\mathcal{K} \cdot \mathcal{B}$ with $\mathbf{z}_t$.
3) In line 12, if replacement is not conducted, then the removed support vector $\mathbf{z}_r$ is set to be the new instance $\mathbf{z}_t$.
4) In lines 14–15, we extend the classic RS strategy by finding the support vector $\mathbf{z}_c$ that is most similar to the removed support vector $\mathbf{z}_r$, updating its weight, and putting its weight back to the buffer $\mathcal{K}.\mathcal{A}$.

In a similar manner, we can define the extended FIFO strategy, namely, FIFO++. For FIFO++, we modify lines 6–13 in Algorithm 3 so that the first support vector in the buffer is removed and the new instance is added to the end of the buffer as a new support vector.

IV. Regret Analysis

In this section, we derive a regret bound for the KOIL algorithm with update rule in (5) under the nonsmooth pairwise hinge loss in (2). Recall that the regret at time T is defined as the difference between the objective value up to the Tth trial and the smallest objective value from hindsight, that is

$$R_T = \sum_{t=1}^{T} (\hat{\mathcal{L}}_t(f_t) - \hat{\mathcal{L}}_t(f^*)) \tag{10}$$

where f^* is the optimal decision function obtained in hindsight by minimizing (3) and $\{f_t\}_{t=1}^T$ are obtained by (5).

In the following, unless otherwise specified, we assume that $\mathbf{z}_i \in N_{t,k}^{-y_t}(\mathbf{z}_t)$, i.e., $\mathbf{z}_i$ is one of the k-nearest opposite support vectors of $\mathbf{z}_t$. We first establish some auxiliary results that will be useful for our derivation of the regret bound.

Lemma 1: 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 $k(\mathbf{x}_t, \mathbf{x}_i) \ge \xi_1^2$ for all $\mathbf{z}_i = (\mathbf{x}_i, y_i) \in N_{t,k}^{-y_t}(\mathbf{z}_t)$. With $f_0 = 0$ and the update rule in (5), we have

$$\|f_t\|_{\mathcal{H}} \le C k c_p$$

for $t \in [T]$, where

$$c_p := \sqrt{2X^2 - 2\xi_1^2}. \tag{11}$$

Lemma 2: Suppose that the assumptions of Lemma 1 hold. With $f_0 = 0$ and the update rule in (5), the pairwise hinge loss function $\ell_h : \mathcal{H} \times \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}_+$ defined in (2) satisfies

$$\ell_h(f_{t-1}, \mathbf{z}_t, \mathbf{z}_i) \le U := 1 + C k c_p^2$$

for $t \in [T]$, where c_p is defined in (11).

The proofs of Lemmas 1 and 2 can be found in Appendixes A and B, respectively. Now, we are ready to present the advertised regret bound.

Theorem 1: Suppose that the assumptions of Lemma 1 hold. With $f_0 = 0$ and the update rule in (5), where $\eta \in (0, 1)$

at each trial is chosen to guarantee descent, we have

$$R_T \le \frac{\|f^*\|_{\mathcal{H}}^2}{2\eta} + \eta C k\left((U-1)+\frac{1}{2}(k+1)Cc_p^2\right)T \quad (12)$$

where c_p is defined in (11).

The proof of Theorem 1 is given in Appendix C. Before we proceed, let us make several remarks.

1) The regret bound R_T can be further bounded by $O(\sqrt{T})$ if we set η to $O(1/\sqrt{T})$. This bound is the same as that for standard online learning algorithms, but it is different from the mistake bounds derived in [4], [10], and [32], which aim at maximizing classification accuracy.
2) The expression in (12) seems to suggest that the smallest regret bound is attained at $k = 1$. However, when $k = 1$, the learned decision function cannot utilize the localized information in the buffers and will yield suboptimal performance. Our empirical evaluation shows that the best choice of k is around 10% of the budget (see detailed results in Section VII). We conjecture that a more accurate surrogate of the AUC metric can provide a better indication on the regret-minimizing value of k. We leave this as a future direction.
3) By exploiting the convexity of the localized instantaneous regularized risk of AUC defined in (4) and confining the range of $|\alpha_t|$ to $[0, \gamma\eta]$, we can derive the corresponding regret bound for the update rule in (9) [21]. However, the regret bound we obtained via this approach is proportional to T. We leave the derivation of a tighter bound as a future work.

V. Extension to a Smooth Pairwise Hinge Loss

In this section, we extend the results developed earlier to the case of a smooth pairwise hinge loss function. Specifically, consider the square of the pairwise hinge loss function, that is

$$\ell_{sh}(f,\mathbf{z},\mathbf{z}') = \left(\frac{|y-y'|}{2}\left[1-\frac{1}{2}(y-y')(f(\mathbf{x})-f(\mathbf{x}'))\right]_+\right)^2. \quad (13)$$

We substitute (13) into (4) and compute the decision function by minimizing the following *smooth localized instantaneous regularized risk of AUC* associated with $\mathbf{z}_t$:

$$\tilde{\mathcal{L}}_t(f) := \tilde{\mathcal{L}}_t(f) = \frac{1}{2}\|f\|_{\mathcal{H}}^2 + C\sum_{\mathbf{z}_i\in N_{t,k}^{-y_t}(\mathbf{z}_t)} \ell_{sh}(f,\mathbf{z}_t,\mathbf{z}_i). \quad (14)$$

As before, we initialize $\tilde{f}_0 = 0$ and apply the standard gradient descent method to update the decision function at each trial, that is

$$\tilde{f}_t := \tilde{f}_{t-1} - \eta\partial_f\tilde{\mathcal{L}}_t(\tilde{f}_{t-1}) \quad (15)$$

where $\eta \in (0, 1)$ is the learning rate and

$$\begin{aligned}&\partial\tilde{\mathcal{L}}_t(\tilde{f}_{t-1})\\ &= \tilde{f}_{t-1} - 2C\sum_{\mathbf{z}_i\in N_{t,k}^{-y_t}(\mathbf{z}_t)}\Big[\mathbb{I}[\ell_h(\tilde{f}_{t-1},\mathbf{z}_t,\mathbf{z}_i)>0]\\ &\qquad\qquad \times\ell_h(\tilde{f}_{t-1},\mathbf{z}_t,\mathbf{z}_i)\cdot\varphi(\mathbf{z}_t,\mathbf{z}_i)\Big].\end{aligned}$$

In addition, we define the valid set V_t and its complement $\overline{V}_t$ at the tth trial as follows:

$$\begin{aligned}V_t &:= \left\{i\in I_t^{-y_t}\,\middle|\,\mathbf{z}_i\in N_{t,k}^{-y_t}(\mathbf{z}_t)\wedge\ell_h(\tilde{f}_{t-1},\mathbf{z}_t,\mathbf{z}_i)>0\right\}\\ \overline{V}_t &:= I_t^{-y_t}\setminus V_t.\end{aligned}$$

Then, the corresponding update rule for the kernel weights at the tth trial is given by

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

Finally, we have the following update rule for the compensated version $\tilde{f}_t^{++}$ of $\tilde{f}_t$:

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

where $\tilde{f}_{t-1}^{++}$ is the compensated decision function from the previous trial. When neither buffer is full, we have $\tilde{f}_t^{++} = \tilde{f}_t$ and the update is done by (15).

By defining the regret at time T as

$$\tilde{R}_T = \sum_{t=1}^{T}(\tilde{\mathcal{L}}_t(\tilde{f}_t) - \tilde{\mathcal{L}}_t(\tilde{f}^*)) \quad (16)$$

where $\tilde{f}^*$ is the optimal decision function obtained in hindsight by minimizing (3) with ℓ_h replaced by ℓ_{sh} defined in (13), and $\{\tilde{f}_t\}_{t=1}^T$ are obtained by the update rule in (15), we have the following regret bound.

Theorem 2: Suppose that the assumptions of Lemma 1 hold. Suppose further that $(1/T)\sum_{t=1}^T\tilde{\mathcal{L}}_t(\tilde{f}^*) \le L^*$ for some $L^* > 0$. With $\tilde{f}_0 = 0$ and the update rule in (15), where $\eta \in (0, 1)$ at each trial is chosen to guarantee descent, we have

$$\tilde{R}_T \le \frac{1}{1-(1+\zeta)\eta}\left(\frac{1}{2\eta}\|\tilde{f}^*\|_{\mathcal{H}}^2 + (1+\zeta)\eta L^* T\right)$$

where $\zeta = 2Ck^2c_p^2$ and c_p is defined in (11).

The proof of Theorem 2 is provided in Appendix D. The result shows that, in general, our KOIL algorithm can attain an $O(\sqrt{T})$ regret bound under the smooth pairwise loss function in (13). Again, we leave the derivation of a tighter regret bound as future work.

VI. KOIL With Multiple Kernel Learning

In this section, we exploit the MKL framework to obtain an accurate data representation for good performance. Given a set of kernel functions $K = \{k_l : \mathcal{X}\times\mathcal{X}\to\mathbb{R}, l\in[m]\}$, we aim to learn a linear combination of these functions to obtain the decision function

$$F_t(\mathbf{x}) = \sum_{l=1}^{m} q_l^t\cdot\operatorname{sgn}(f_{l,t}(\mathbf{x}))$$

where $\mathbf{q}^t = [q_1^t,\ldots,q_m^t]$ is the normalized (i.e., $\sum_{l=1}^m q_l^t = 1$) weight for multiple kernel classifiers learned up to the tth trial and $f_{l,t}$ is an element of the RKHS $\mathcal{H}_{k_l}$ endowed with the

Algorithm 4 KOIL With MKL

```
1: Input:
     • penalty parameter $C$ and learning rates $\eta$, $\lambda$
     • maximum positive and negative budgets $N^+$ and $N^-$, respectively
     • number of nearest neighbors $k$
2: Initialize $\mathbf{w}^1 = \mathbf{1}$, $\mathcal{K}_l^+.\mathcal{A} = \mathcal{K}_l^-.\mathcal{A} = \emptyset$, $\mathcal{K}_l^+.\mathcal{B} = \mathcal{K}_l^-.\mathcal{B} = \emptyset$, $N_{p,l} = N_{n,l} = 0$, for $l \in [m]$
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:      $w_l^{t+1} = w_l^t \exp(-\lambda \check{\mathcal{L}}_t(f_{l,t}))$
17:    end if
18:  end for
19:  $\mathbf{q}^{t+1} = \mathbf{w}^{t+1}/|\mathbf{w}^{t+1}|$
20: end for
```

inner product k_l. The lth kernel classifier at the tth trial is defined to have the same form as in (1)

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

As before, 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 lth kernel classifier at the tth trial.

Algorithm 4 shows the KOIL algorithm with multiple kernels.

1) In line 6, we select the classifier based on the Bernoulli distribution that is proportional to the weight of the classifier. Since the weight is divided by the maximum weight of all classifiers, at least one classifier will be selected at each trial.
2) In lines 7–15, we update the predictor of the sampled classifier. To avoid excessive update fluctuation, we define the loss function $\check{\mathcal{L}}_t$ as in (4) and (14), but without the regularization term. This necessitates a change in the update rule for $\alpha_{i,t}$ in the function UpdateKernel. Specifically, in UpdateKernel2, we update $\alpha_{i,t}$ by

$$\alpha_{i,t} = \begin{cases} \eta C y_t |V_t|, & i = t \\ \alpha_{i,t-1} - \eta C y_t, & i \in V_t \\ \alpha_{i,t-1}, & i \in I_{t-1}^{y_t} \cup \overline{V}_t \end{cases}$$

if the pairwise hinge loss function in (2) is used, and by

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

if the smooth pairwise hinge loss function in (13) is used.
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 should be noted that in order to avoid fluctuation, we do not add a smoothing term to update the probability of selecting classifiers as in [20], [22], [36], [46], and [52].

Similar to (10) and (16), we can define the corresponding regret for $\{f_{l,t}\}$ and obtain an expected regret bound for Algorithm 4.

Theorem 3: Suppose that the loss function is nonnegative, $\max_{t=1}^T \check{\mathcal{L}}_t(f_{l,t-1}) \leq L$, and $\|\partial_f \check{\mathcal{L}}_t(f_{l,t-1})\|_{\mathcal{H}_{k_l}} \leq G$ for some $L, G > 0$. With $f_{l,0} = 0$ and suitable choices of $\eta \in (0, 1)$, $\lambda > 0$, we have

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

where $q_l^t = w_l^t/[\sum_{l=1}^m w_l^t]$.

Note that by assuming the boundedness of the optimal kernel predictor and setting $\eta, \lambda = O(1/\sqrt{T})$, we can obtain a regret bound of $O(\sqrt{T})$ by following the proof in [52]. Alternatively, we can utilize the inequality of arithmetic and geometric means (AM-GM inequality) to remove the term $\ln m/\eta$ in [52]. Due to space limitation, we omit the proof here.

VII. Experiments

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

A. Compared Algorithms

We compare our proposed KOIL algorithm with the state-of-the-art online learning algorithms. Since our focus is on online imbalanced learning, for fairness' sake, we do not consider the batch-trained imbalanced learning algorithms in our comparison. Rather, we consider the online linear algorithms and the kernel-based online learning algorithms with a finite or infinite buffer size.

1) "Perceptron": The classical perceptron algorithm [34].
2) "OAM$_{\text{seq}}$": An online linear AUC maximization algorithm [55].

[1] Demo codes written in both C++ and MATLAB can be downloaded at https://github.com/JunjieHu/koil.

3) "OPAUC": One-pass AUC maximization [15].
4) "NORMA": Online learning with kernels [27].
5) "RBP": Randomized budget perceptron [4].
6) "Forgetron": A kernel-based perceptron on a fixed budget [10].
7) "Projectron/**Projectron++**": A bounded kernel-based perceptron [32].
8) "$\text{KOIL}_{\text{RS++}}$/$\text{KOIL}_{\text{FIFO++}}$": Our proposed algorithm with the pairwise hinge loss function in (2) and fixed budgets updated by RS++ and FIFO++, respectively.
9) "$\text{KOIL}^2_{\text{RS++}}$/$\text{KOIL}^2_{\text{FIFO++}}$": Our proposed algorithm with the smooth pairwise hinge loss function in (13) and fixed budgets updated by RS++ and FIFO++, respectively.

B. Experimental Setup

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

C. Experiments on Synthetic Data Sets

To illustrate the KOIL algorithm and show the power of the kernel method, we generate a synthetic data set in 2-D space [see the example in Fig. 1(a)]. The positive samples are generated from the 2-D Gaussian distribution with mean $(1/2, 1/2)$ and standard deviation 0.1. The negative samples are generated from a mixture of four Gaussians with the same standard deviation as the positive samples and means at $(1/6, 1/2)$, $(1/2, 1/6)$, $(1/2, 5/6)$, and $(5/6, 1/2)$, respectively.

Following the above-mentioned setup, we generate different synthetic data sets with different imbalanced ratios to explore the performance of KOIL in different scenarios. The data sets consist of the following.

1) *Syn1:* A set of data with imbalanced ratio 1:4 consisting of 200 positive samples and 800 negative samples.
2) *Syn2:* A set of data with imbalanced ratio 1:10 consisting of 100 positive samples and 1000 negative samples.
3) *Syn3:* A set of data with imbalanced ratio 1:50 consisting of 100 positive samples and 5000 negative samples.
4) *Syn4:* A set of data with imbalanced ratio 1:100 consisting of 100 positive samples and 10 000 negative samples.

Obviously, these four data sets are linearly nonseparable in the original space. From Table II, we can observe that the kernel-based learning algorithms significantly outperform the online linear algorithms. For example, in the Syn1 data set, perceptron and the OAM_{seq} with buffer size 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 algorithm with the smooth pairwise loss function can attain comparable or even better performance than that with the nonsmooth loss function.

TABLE I

SUMMARY OF ALL DATA SETS (C^* AND γ^* ARE THE CORRESPONDING OPTIMAL HYPERPARAMETERS TUNED BY FIVEFOLD CROSS VALIDATION)

Dataset	T	d	T^-/T^+	C^*	γ^*
Syn1	1,000	2	4	2^4	2^7
Syn2	1,100	2	10	2^{-9}	2^5
Syn3	5,100	2	50	2^{-10}	2^4
Syn4	10,100	2	100	2^{-6}	2^5
sonar	208	60	1.144	2^4	1
australian	690	14	1.248	2^2	1
heart	270	13	1.250	2^7	2^{-8}
ionosphere	351	34	1.786	2^5	2
diabetes	768	8	1.866	2^5	2
glass	214	9	2.057	2^5	2^5
german	1,000	24	2.333	2^7	2^{-4}
svmguide2	391	20	2.342	2^8	2^{-5}
segment	2,310	19	6.000	2^3	2^3
satimage	4,435	36	9.687	2^6	2
vowel	528	10	10.000	2^4	2^3
letter	15,000	16	26.881	2^5	2^5
poker	25,010	10	47.752	2^5	2^{-4}
shuttle	43,500	9	328.546	2^7	2^{10}

D. Experiments on Benchmark Real-World Data Sets

The 14 well-known benchmark data sets, whose imbalanced ratios range from 1.144 to 328.546, are obtained from the UCI and LIBSVM websites for evaluation. Table I summarizes the detailed statistics of the data sets.

For each data set, we conduct fivefold cross validation on all the algorithms, where four folds of the data are used for training while the rest for testing. The fivefold 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. The results are reported in Table III. From the table, we have the following observations.

1) Our KOIL algorithm with the RS++ and FIFO++ updating policies performs better than the online linear AUC maximization algorithms in most data sets. By examining the results of OAM_{seq} on australian, heart, diabetes, german, and shuttle, as well as the results of OPAUC on australian and german, we speculate that a linear classifier is enough to achieve good performance on these data sets while a nonlinear classifier can be adversely affected by outliers.
2) Under the pairwise hinge loss function, the KOIL algorithm significantly outperforms all competing kernel-based algorithms in nearly all data sets. The results demonstrate the effectiveness of our proposed approach.
3) We observe that the KOIL algorithm with nonsmooth loss beats the one with smooth loss in five data sets while being comparable in the remaining nine data sets.

TABLE II

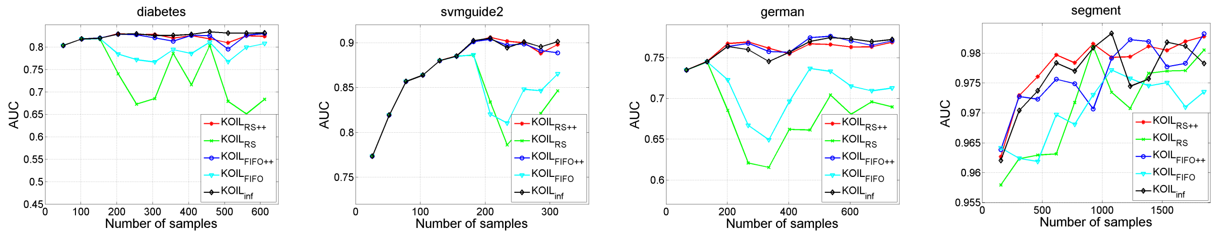
AVERAGE AUC PERFORMANCE (MEAN±STD) ON THE SYNTHETICS DATA SETS. ●/○ (-) INDICATES THAT BOTH/ONE OF KOIL$_{RS++}$ AND KOIL$_{FIFO++}$ ARE/IS SIGNIFICANTLY BETTER (WORSE) THAN THE CORRESPONDING METHOD (PAIRWISE t-TESTS AT 95% SIGNIFICANCE LEVEL)

Data	KOIL$_{RS++}$	KOIL$_{FIFO++}$	KOIL$^2_{RS++}$	KOIL$^2_{FIFO++}$	Perceptron	OAM$_{seq}$	OPAUC	NORMA	RBP	Forgetron	Projectron	Projectron++
Syn1	.961±.016	.960±.014	.967±.011	**.968**±.011	.495±.031●	.501±.021●	.503±.032●	.940±.013●	.948±.021●	.878±.147●	.954±.019	.953±.017
Syn2	.959±.022	.958±.018	.961±.017	**.962**±.018	.484±.037●	.502±.032●	.508±.032●	.937±.041●	.887±.062●	.954±.023	.941±.032●	.944±.023●
Syn3	.939±.029	.941±.025	**.943**±.022	.942±.023	.495±.025●	.499±.022●	.492±.020●	.769±.087●	.872±.081●	.807±.130●	.901±.064●	.922±.039●
Syn4	.965±.014	.966±.013	**.968**±.013	.966±.015	.510±.023●	.495±.026●	.499±.022●	.834±.205●	.892±.069●	.844±.097●	.962±.015	.948±.024●
win/tie/loss			0/4/0	0/4/0	4/0/0	4/0/0	4/0/0	4/0/0	4/0/0	3/1/0	2/2/0	3/1/0

TABLE III

AVERAGE AUC PERFORMANCE (MEAN±STD) ON THE BENCHMARK DATA SETS. ●/○ (-) INDICATES THAT BOTH/ONE OF KOIL$_{RS++}$ AND KOIL$_{FIFO++}$ ARE/IS SIGNIFICANTLY BETTER (WORSE) THAN THE CORRESPONDING METHOD (PAIRWISE t-TESTS AT 95% SIGNIFICANCE LEVEL)

Data	KOIL$_{RS++}$	KOIL$_{FIFO++}$	KOIL$^2_{RS++}$	KOIL$^2_{FIFO++}$	Perceptron	OAM$_{seq}$	OPAUC	NORMA	RBP	Forgetron	Projectron	Projectron++
sonar	.955±.028	.955±.028	**.957**±.031	**.957**±.031	.803±.083●	.843±.056●	.844±.077●	.925±.044●	.913±.032●	.896±.054●	.896±.049●	.896±.049●
australian	.923±.023	.922±.026	.919±.024	.920±.026	.869±.035●	**.925**±.024	.923±.025	.919±.023	.911±.017●	.912±.026●	.923±.024	.923±.024
heart	.908±.040	.910±.040	.911±.038	.908±.037	.876±.066●	**.912**±.040	.901±.043○	.890±.051●	.865±.043●	.900±.053○	.902±.038	.905±.042
ionosphere	**.985**±.015	**.985**±.015	.959±.026●	.952±.031●	.851±.056●	.905±.041●	.888±.046●	.961±.016●	.960±.030●	.945±.031●	.964±.025●	.963±.027●
diabetes	.826±.036	.830±.030	.817±.037○	.825±.028	.726±.059●	.827±.033	.805±.035●	.792±.032●	.828±.034	.820±.027○	.832±.033	**.833**±.033
glass	**.887**±.053	.884±.054	.885±.048	.885±.048	.810±.065●	.827±.064●	.800±.074●	.811±.077●	.811±.071●	.813±.075●	.811±.070●	.781±.076●
german	.769±.032	.778±.031	.774±.030	.769±.037○	.748±.033●	.777±.027	**.787**±.026 -	.766±.032○	.699±.038●	.712±.054●	.769±.028○	.770±.024
svmguide2	**.897**±.040	.885±.043	.891±.042	.882±.040○	.860±.037●	.886±.045○	.859±.050●	.865±.046●	.890±.038	.864±.045●	.886±.044○	.886±.045○
segment	.983±.008	**.985**±.012	.970±.012●	.959±.015●	.875±.020●	.919±.020●	.882±.019●	.910±.042●	.969±.017●	.943±.038●	.979±.013●	.978±.016●
satimage	**.924**±.012	.923±.015	.922±.012	.922±.013	.700±.015●	.755±.018●	.724±.016●	.914±.025●	.899±.018●	.892±.032●	.910±.015●	.904±.011●
vowel	**1.000**±.000	**1.000**±.001	.998±.007●	.993±.014●	.848±.070●	.905±.024●	.885±.034●	.996±.005●	.968±.017●	.987±.027●	.982±.013●	.994±.019●
letter	.933±.021	**.942**±.017	.926±.022●	.932±.017○	.767±.029●	.827±.021●	.823±.018●	.910±.027●	.928±.011○	.815±.102●	.926±.016●	.926±.015●
poker	.681±.031	**.693**±.032	.654±.023●	.676±.031●	.514±.030●	.503±.024●	.509±.031●	.577±.040●	.501±.031●	.572±.029●	.675±.027●	.675±.027●
shuttle	.950±.040	.956±.021	.946±.039	.953±.020	.520±.134●	**.999**±.000 -	.754±.043●	.725±.053●	.844±.041●	.839±.060●	.873±.063●	.795±.063●
win/tie/loss			6/8/0	7/7/0	14/0/0	9/4/1	12/1/1	13/1/0	12/2/0	14/0/0	11/3/0	10/4/0



(a) (b) (c) (d)

Fig. 2. Average AUC performance on four data sets obtained by different updating policies of the KOIL algorithm. (a) Diabetes. (b) Svmguide2. (c) German. (d) Segment.

4) In most of the data sets, kernel-based algorithms show better AUC performance than the linear algorithms. This again demonstrates the power of kernel methods in classifying real-world data sets.
5) We observe that the performance of OAM$_{seq}$ on satimage is not as good as that in [21] and [55]. This can be attributed to the different partition of the training and test data.

E. Evaluation of Updating Policies

We compare the compensation schemes RS++ and FIFO++ with the original updating policies RS and FIFO and show the average AUC performance of 20 runs on four typical data sets in Fig. 2. Here, KOIL$_{inf}$ denotes KOIL learned with infinite budgets and is used as a reference. From Fig. 2, we have the following observations.

1) KOIL$_{RS++}$ and KOIL$_{FIFO++}$ have nearly the same performance as KOIL$_{inf}$. This confirms that the extended policies indeed compensate for the lost information when a support vector is replaced.
2) The KOIL algorithm with extended updating policies significant outperforms the one with original stream oblivious policy when either buffer is full. Without compensation, the performance fluctuates and decays when support vectors are removed. With compensation, the performance is rather stable.

F. Sensitivity Analysis

In this section, we study the sensitivity of the KOIL algorithm to the input parameters. First, we test the performance of the KOIL algorithm as the buffer size varies. From Fig. 3, we observe that the performance follows similar trend in [21] and [55], i.e., improving gradually with the increase of the buffer size and becoming stable when the size is relatively large.

Next, we test the performance of the KOIL algorithm as the number of localized support vectors k varies. From Fig. 4, we have the following observations.

1) When $k = 1$, the smallest possible value of k, the performance of the KOIL algorithm is usually poor,

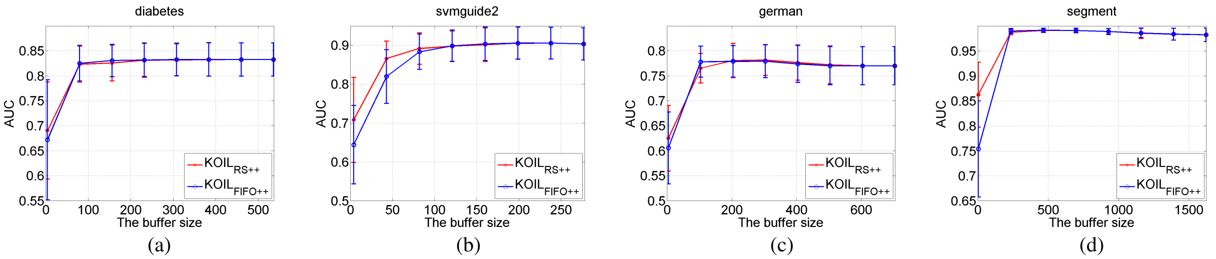


Fig. 3. Average AUC of the KOIL algorithm with different buffer sizes.

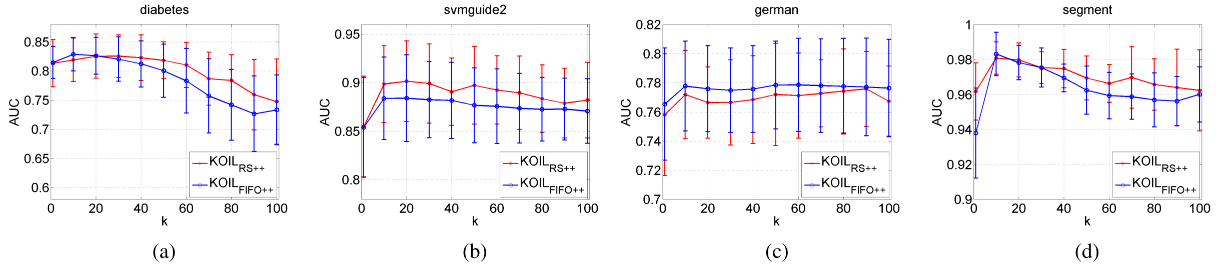


Fig. 4. Average AUC of the KOIL algorithm with different k values. Here, $k = [1, 10{:}10{:}100]$ and the budget is 100 for each buffer.

because it only considers the pairwise loss incurred by the nearest opposite support vector of the new instance and cannot fully utilize the localized information.

2) The KOIL algorithm usually attains the best performance when k is approximately 10% of the buffer size. As k further increases, the performance starts to deteriorate. Our results consistently demonstrate that the effect of outliers can be alleviated by utilizing the localized information of the new instance.

3) For some data sets, such as svmguide2 and german, the performance of the KOIL algorithm is not too sensitive to k. The reason could be that the learned support vectors in these data sets are well separated when the buffers are full. As a result, new instances have little influence on the updating of the decision function.

In sum, a key step in maintaining the model performance is the compensation scheme; see the results in Fig. 2. The setting of localized AUC is also crucial to good performance as it can mitigate the effect of noise (see results in Fig. 4). Fig. 3 suggests that the budget just needs to be sufficiently large, say several hundreds.

G. Evaluation of the KOIL Algorithm With MKL

We evaluate the performance of the KOIL algorithm with MKL using the setting in [22]. Specifically, we use 16 kernel functions in our experiment, including 3 polynomial kernels (i.e., $k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^p$ with degree parameters $p = 1$, 2, and 3) and 13 Gaussian kernels (i.e., $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2)$ with kernel width parameter $\sigma \in 2^{[-6:1:6]}$). For simplicity, the learning rates η and λ are both set to 0.01. A fivefold cross validation is applied to find the best penalty cost C from $2^{[-10:1:10]}$. Table IV summarizes the results and reveals the following.

TABLE IV

AVERAGE AUC PERFORMANCE (MEAN±STD) ON THE BENCHMARK DATA SETS. ● (-) 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	KOIL$_{RS++}^{MKL}$	KOIL$_{FIFO++}^{MKL}$	KOIL$_{RS++}^{MKL\ 2}$	KOIL$_{FIFO++}^{MKL\ 2}$
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

1) The KOIL algorithm with MKL attains better or comparable performance than the one with tuned optimal kernel. Indeed, under the smooth loss function, the former has a better performance in at least 6 out of the 14 data sets. On the other hand, under the nonsmooth loss function, both versions of the KOIL algorithm have comparable performance on most data sets. We conjecture that this may be due to the nonsmoothness of the loss function.

2) For some data sets, such as sonar and ionosphere, the KOIL algorithm with MKL cannot beat the one with the tuned optimal kernel. We conjecture that this may be due to the limitation of the training data in these data sets. Training with multiepoches [52] could be a promising approach to improving the model performance.

VIII. CONCLUSION

We focused on the imbalanced streaming binary classification problem and proposed a kernel-based online learning algorithm to seek a nonlinear classifier. Our algorithm is based on three crucial ideas. First, we adopt two fixed-budget buffers to control the number of support vectors and maintain the global information on the decision boundary. Second, we update the weight of a new arriving support vector by confining its influence on only its k-nearest opposite support vectors. Third, we transfer the weight of the removed support vector to its most similar one when either buffer is full, so as to avoid information loss. We also exploited the MKL framework to determine the kernel our KOIL algorithm. Finally, we conducted extensive experiments to demonstrate the efficacy and superiority of our proposed approach.

Several challenging but promising directions can be considered in the future. First, the current KOIL algorithm only explores a localized surrogate of the AUC metric. Investigating more accurate surrogate functions for the AUC metric is significant in both theory and practice. Second, the current regret bound only applies to the case where there is no compensation. A natural direction is to derive a regret bound for the case where the compensation scheme is used. Third, it would be interesting to investigate and evaluate more efficient update rules for the KOIL algorithm with MKL.

Appendix A
Proof of Lemma 1

Proof: First, the assumptions $k(\mathbf{x},\mathbf{x}) \le X^2$ for all $\mathbf{x} \in \mathbb{R}^d$ and $k(\mathbf{x}_t,\mathbf{x}_i) \ge \xi_1^2 > 0$ yield

$$\|\varphi(\mathbf{z}_t,\mathbf{z}_i)\|_{\mathcal{H}} = \sqrt{k(\mathbf{x}_t,\mathbf{x}_t) - 2k(\mathbf{x}_t,\mathbf{x}_i) + k(\mathbf{x}_i,\mathbf{x}_i)} \le c_p \tag{17}$$

where c_p is defined in (11). Now, using (5), (7), and the triangle inequality, we compute

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

where the last inequality is due to (17) 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 and using $f_0 = 0$, we have

$$\|f_t\|_{\mathcal{H}} \le (1-\eta)^t \|f_0\|_{\mathcal{H}} + \frac{1-(1-\eta)^t}{\eta}\eta C k c_p \le C k c_p$$

where the second inequality is due to the fact that when $\eta \in (0,1)$, we have $1-(1-\eta)^t \le 1$ for $t \in [T]$. This completes the proof. □

Appendix B
Proof of Lemma 2

Proof: Based on the pairwise hinge loss defined in (2), we have

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

where the first inequality is due to the triangle inequality and $(1/2)|y_t - y_i| \le 1$; the second inequality is due to the Cauchy–Schwarz inequality; the third inequality is due to Lemma 1 and (17). □

Appendix C
Proof of Theorem 1

Proof: Since the learning rate $\eta \in (0,1)$ at each trial is chosen to guarantee descent and $\hat{\mathcal{L}}_t$ is convex, we have

$$\begin{aligned}R_T &= \sum_{t=1}^{T}(\hat{\mathcal{L}}_t(f_t) - \hat{\mathcal{L}}_t(f^*)) \le \sum_{t=1}^{T}(\hat{\mathcal{L}}_t(f_{t-1}) - \hat{\mathcal{L}}_t(f^*)) \\ &\le \sum_{t=1}^{T}\langle \partial\hat{\mathcal{L}}_t(f_{t-1}), f_{t-1} - f^*\rangle_{\mathcal{H}}.\end{aligned} \tag{18}$$

Now, observe that

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

By summing the above identity over $t \in [T]$, we have

$$\begin{aligned}&\|f_T - f^*\|_{\mathcal{H}}^2 - \|f_0 - f^*\|_{\mathcal{H}}^2 \\ &\quad = -2\eta\sum_{t=1}^{T}\langle\partial_f\hat{\mathcal{L}}_t(f_{t-1}), f_{t-1} - f^*\rangle_{\mathcal{H}} \\ &\qquad + \eta^2\sum_{t=1}^{T}\|\partial_f\hat{\mathcal{L}}_t(f_{t-1})\|_{\mathcal{H}}^2.\end{aligned} \tag{19}$$

Upon combining (18) and (19) and using the fact that $f_0 = 0$, $\|f_T - f^*\|_{\mathcal{H}}^2 \ge 0$, we obtain

$$R_T \le \frac{\|f^*\|_{\mathcal{H}}^2}{2\eta} + \frac{\eta}{2}\sum_{t=1}^{T}\|\partial_f\hat{\mathcal{L}}_t(f_{t-1})\|_{\mathcal{H}}^2.$$

We now proceed to bound $\|\partial_f\hat{\mathcal{L}}_t(f_{t-1})\|_{\mathcal{H}}^2$. Observe that

$$\begin{aligned}&\|\partial_f\hat{\mathcal{L}}_t(f_{t-1})\|_{\mathcal{H}}^2 \\ &= \left\| f_{t-1} - C\sum_{\mathbf{z}_i\in N_{t,k}^{-y_t}(\mathbf{z}_t)}\mathbb{I}[\ell_h(f_{t-1},\mathbf{z}_t,\mathbf{z}_i) > 0]\cdot\varphi(\mathbf{z}_t,\mathbf{z}_i)\right\|_{\mathcal{H}}^2 \\ &= \|f_{t-1}\|_{\mathcal{H}}^2 \\ &\quad + \left\| C\sum_{\mathbf{z}_i\in N_{t,k}^{-y_t}(\mathbf{z}_t)}\mathbb{I}[\ell_h(f_{t-1},\mathbf{z}_t,\mathbf{z}_i) > 0]\cdot\varphi(\mathbf{z}_t,\mathbf{z}_i)\right\|_{\mathcal{H}}^2 \\ &\quad - 2C\sum_{\mathbf{z}_i\in N_{t,k}^{-y_t}(\mathbf{z}_t)}\mathbb{I}[\ell_h(f_{t-1},\mathbf{z}_t,\mathbf{z}_i) > 0]\langle f_{t-1},\varphi(\mathbf{z}_t,\mathbf{z}_i)\rangle_{\mathcal{H}}.\end{aligned}$$

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

$$\|f_{t-1}\|_{\mathcal{H}}^2 \le C^2k^2c_p^2. \tag{20}$$

Now, by (17) and the fact that the number of elements in $N_{t,k}^{-y_t}(\mathbf{z}_t)$ is at most k, we can bound the second term by

$$\begin{aligned}\left\| C \sum_{\mathbf{z}_i \in N_{t,k}^{-y_t}(\mathbf{z}_t)} \mathbb{I}[\ell_h(f_{t-1}, \mathbf{z}_t, \mathbf{z}_i) > 0] \cdot \varphi(\mathbf{z}_t, \mathbf{z}_i) \right\|_{\mathcal{H}}^2 \\ \le C^2 \sum_{\mathbf{z}_i \in N_{t,k}^{-y_t}(\mathbf{z}_t)} \mathbb{I}[\ell_h(f_{t-1}, \mathbf{z}_t, \mathbf{z}_i) > 0] \cdot \|\varphi(\mathbf{z}_t, \mathbf{z}_i)\|_{\mathcal{H}}^2 \\ \le C^2 k c_p^2. \qquad (21)\end{aligned}$$

Finally, since f_{t-1} is an element of an RKHS and $\ell_h(f_{t-1}, \mathbf{z}_t, \mathbf{z}_i) \le U$ by Lemma 2, we have

$$\begin{aligned}\langle f_{t-1}, \varphi(\mathbf{z}_t, \mathbf{z}_i)\rangle_{\mathcal{H}} &= \frac{1}{2}(y_t - y_i)\,(f_{t-1}(\mathbf{x}_t) - f_{t-1}(\mathbf{x}_i)) \\ &\ge 1 - U.\end{aligned}$$

It follows that the third term can be bounded by:

$$\begin{aligned}-2C \sum_{\mathbf{z}_i \in N_{t,k}^{-y_t}(\mathbf{z}_t)} \mathbb{I}[\ell_h(f_{t-1}, \mathbf{z}_t, \mathbf{z}_i) > 0] \langle f_{t-1}, \varphi(\mathbf{z}_t, \mathbf{z}_i)\rangle_{\mathcal{H}} \\ \le 2C \sum_{\mathbf{z}_i \in N_{t,k}^{-y_t}(\mathbf{z}_t)} \mathbb{I}[\ell_h(f_{t-1}, \mathbf{z}_t, \mathbf{z}_i) > 0](U - 1) \\ \le 2Ck(U - 1) \qquad (22)\end{aligned}$$

where the second inequality is again due to the fact that the number of elements in $N_{t,k}^{-y_t}(\mathbf{z}_t)$ is at most k.

By combining (20), (21), and (22), we obtain

$$\|\partial_f \hat{\mathcal{L}}_t(f_{t-1})\|_{\mathcal{H}}^2 \le C^2 k(k+1)c_p^2 + 2Ck(U-1).$$

The bound on R_T in (12) now follows by summing $\partial_f \hat{\mathcal{L}}_t(f_{t-1})$ over $t \in [T]$. This completes the proof. □

Appendix D
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, since the learning rate $\eta \in (0, 1)$ at each trial is chosen to guarantee descent and $\tilde{\mathcal{L}}_t$ is convex, we have

$$\begin{aligned}\tilde{\mathcal{L}}_t(\tilde{f}_t) - \tilde{\mathcal{L}}_t(\tilde{f}^*) &\le \tilde{\mathcal{L}}_t(\tilde{f}_{t-1}) - \tilde{\mathcal{L}}_t(\tilde{f}^*) \\ &\le \langle \partial_f \tilde{\mathcal{L}}_t(\tilde{f}_{t-1}), \tilde{f}_{t-1} - \tilde{f}^*\rangle. \qquad (23)\end{aligned}$$

We also have

$$\begin{aligned}\|\tilde{f}_t - \tilde{f}^*\|_{\mathcal{H}}^2 - \|\tilde{f}_{t-1} - \tilde{f}^*\|_{\mathcal{H}}^2 \\ = \eta^2 \|\partial_f \tilde{\mathcal{L}}_t(\tilde{f}_{t-1})\|_{\mathcal{H}}^2 - 2\eta \langle \partial_f \tilde{\mathcal{L}}_t(\tilde{f}_{t-1}), \tilde{f}_{t-1} - \tilde{f}^*\rangle_{\mathcal{H}}. \\ (24)\end{aligned}$$

Now, we compute

$$\frac{\partial^2 \tilde{\mathcal{L}}_t}{\partial f^2} = I + 2C \sum_{\mathbf{z}_i, \mathbf{z}_j \in N_{t,k}^{-y_t}(\mathbf{z}_t)} \mathbb{I}_{\mathbf{z}_i} \mathbb{I}_{\mathbf{z}_j} \varphi(\mathbf{z}_t, \mathbf{z}_i) \varphi(\mathbf{z}_t, \mathbf{z}_j)^\top \qquad (25)$$

where we denote $\mathbb{I}[\ell_h(\tilde{f}_{t-1}, \mathbf{z}_t, \mathbf{z}_i) > 0]$ by $\mathbb{I}_{\mathbf{z}_i}$ for simplicity. It follows that for any $\tilde{f}, \tilde{g}$:

$$\|\partial_f \tilde{\mathcal{L}}_t(\tilde{f}) - \partial_f \tilde{\mathcal{L}}_t(\tilde{g})\|_{\mathcal{H}} \le (1 + \zeta)\|\tilde{f} - \tilde{g}\|_{\mathcal{H}} \qquad (26)$$

where $\zeta = 2Ck^2c_p^2$ is obtained by the summation in (25) and the bound

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

In particular, suppose that $\tilde{f}_t^*$ minimizes $\tilde{\mathcal{L}}_t$. Then, by the convexity and smoothness of $\tilde{\mathcal{L}}_t$, we have $\partial_f \tilde{\mathcal{L}}_t(\tilde{f}_t^*) = 0$. This, together with (26) and [31, Th. 2.1.5], implies that

$$\begin{aligned}\|\partial_f \tilde{\mathcal{L}}_t(\tilde{f}_{t-1})\|_{\mathcal{H}}^2 &= \left\|\partial_f \tilde{\mathcal{L}}_t(\tilde{f}_{t-1}) - \partial_f \tilde{\mathcal{L}}_t\left(\tilde{f}_t^*\right)\right\|_{\mathcal{H}}^2 \\ &\le 2(1+\zeta)\left(\tilde{\mathcal{L}}_t(\tilde{f}_{t-1}) - \tilde{\mathcal{L}}_t\left(\tilde{f}_t^*\right)\right) \\ &\le 2(1+\zeta)\tilde{\mathcal{L}}_t(\tilde{f}_{t-1}) \qquad (27)\end{aligned}$$

where the last inequality is due to $\tilde{\mathcal{L}}_t(\tilde{f}_t^*) \ge 0$.

By combining (23), (24), and (27), we obtain

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

Upon summing the above inequality over $t \in [T]$ and rearranging, we obtain

$$\begin{aligned}\sum_{t=1}^{T}(1 - (1+\zeta)\eta)\tilde{\mathcal{L}}_t(\tilde{f}_{t-1}) - \sum_{t=1}^{T}\tilde{\mathcal{L}}_t(\tilde{f}^*) \\ \le \frac{1}{2\eta}\left(\|\tilde{f}_0 - \tilde{f}^*\|_{\mathcal{H}}^2 - \|\tilde{f}_T - \tilde{f}^*\|_{\mathcal{H}}^2\right) \le \frac{1}{2\eta}\|\tilde{f}^*\|_{\mathcal{H}}^2.\end{aligned}$$

Here, we use the fact that $\tilde{f}_0 = 0$ and $\|\tilde{f}_T - \tilde{f}^*\|_{\mathcal{H}}^2 \ge 0$. It follows that:

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

as desired. □

References

[1] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognit.*, vol. 30, no. 7, pp. 1145–1159, 1997.

[2] U. Brefeld and T. Scheffer, "AUC maximizing support vector learning," in *Proc. Workshop ROC Anal. Mach. Learn.*, 2005.

[3] C. L. Castro and A. P. Braga, "Novel cost-sensitive approach to improve the multilayer perceptron performance on imbalanced data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 6, pp. 888–899, Jun. 2013.

[4] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile, "Tracking the best hyperplane with a simple budget perceptron," *Mach. Learn.*, vol. 69, no. 2, pp. 143–167, 2007.

[5] N. Cesa-Bianchi, A. Conconi, and C. Gentile, "A second-order perceptron algorithm," *SIAM J. Comput.*, vol. 34, no. 3, pp. 640–668, 2005.

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

[7] C. Cortes and M. Mohri "AUC optimization vs. Error rate minimization," in *Proc. NIPS*, 2003, pp. 313–320.

[8] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *J. Mach. Learn. Res.*, vol. 7, pp. 551–585, Dec. 2006.

[9] L. Csató and M. Opper, "Sparse on-line Gaussian processes," *Neural Comput.*, vol. 14, no. 3, pp. 641–668, 2002.

[10] O. Dekel, S. Shalev-Shwartz, and Y. Singer, “The Forgetron: A kernel-based Perceptron on a budget,” *SIAM J. Comput.*, vol. 37, no. 5, pp. 1342–1372, 2008.

[11] Y. Ding, P. Zhao, S. C. H. Hoi, and Y.-S. Ong, “Adaptive subgradient methods for online AUC maximization,” in *Proc. AAAI*, Feb. 2016, pp. 2568–2574.

[12] G. Dror, N. Koenigstein, Y. Koren, and M. Weimer, “The Yahoo! music dataset and KDD-Cup’11,” in *Proc. KDD Cup*, 2012, pp. 3–18.

[13] Y. Engel, S. Mannor, and R. Meir, “The kernel recursive least-squares algorithm,” *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2275–2285, Aug. 2004.

[14] Y. Freund and R. E. Schapire, “Large margin classification using the perceptron algorithm,” *Mach. Learn.*, vol. 37, no. 3, pp. 277–296, 1999.

[15] W. Gao, L. Wang, R. Jin, S. Zhu, and Z.-H. Zhou, “One-pass AUC optimization,” in *Proc. Artif. Intell.*, Jul. 2016, pp. 906–914.

[16] 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*, vol. 11, no. 2, pp. 68–76, 2009.

[17] J. A. Hanley and B. J. McNeil, “The meaning and use of the area under a receiver operating characteristic (ROC) curve,” *Radiology*, vol 143, no. 1, pp. 29–36, 1982.

[18] J. Hensman, N. Fusi, and N. D. Lawrence, “Gaussian processes for big data,” in *Proc. 29th Conf. Uncertainty Artif. Intell.*, 2013.

[19] A. Herschtal and B. Raskutti, “Optimising area under the ROC curve using gradient descent,” in *Proc. 21st Int. Conf. Mach.*, 2004, p. 49.

[20] S. C. H. Hoi, R. Jin, P. Zhao, and T. Yang, “Online multiple kernel classification,” *Mach. Learn.*, vol. 90, no. 2, pp. 289–316, 2013.

[21] J. Hu, H. Yang, I. King, M. R. Lyu, and A. M.-C. So, “Kernelized online imbalanced learning with fixed budgets,” in *Proc. AAAI*, Austin, TX, USA, Jan. 2015, pp. 25-30.

[22] R. Jin, S. C. H. Hoi, and T. Yang, “Online multiple kernel learning: Algorithms and mistake bounds,” in *Proc. Algorithmic Learn. Theory*, pp. 390–404, 2010.

[23] T. A. Joachims, “A support vector method for multivariate performance measures,” in *Proc. 22nd Int. Conf. Mach.*, 2005, pp. 377–384.

[24] P. Kar, B. K. Sriperumbudur, P. Jain, and H. Karnick, “On the generalization ability of online learning algorithms for pairwise loss functions,” in *Proc. ICML*, 2013, pp. 441–449.

[25] N. Karampatziakis and J. Langford, “Online importance weight aware updates,” in *Proc. Comput. Sci.*, 2011, pp. 392–399.

[26] S. S. Keerthi and W. Chu, “A matching pursuit approach to sparse Gaussian process regression,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2005, pp. 643–650.

[27] J. Kivinen, A. J. Smola, and R. C. Williamson, “Online learning with kernels,” *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.

[28] M. Kloft, U. Brefeld, S. Sonnenburg, and A. Zien, “l_p-norm multiple kernel learning,” *J. Mach. Learn. Res.*, vol. 12, pp. 953–997, Mar. 2011.

[29] Y. Li and P. M. Long, “The relaxed online maximum margin algorithm,” *Mach. Learn.*, vol. 46, no. 1, pp. 361–387, 2002.

[30] M. Lin, K. Tang, and X. Yao, “Dynamic sampling approach to training neural networks for multiclass imbalance classification,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 4, pp. 647–660, Apr. 2013.

[31] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*. Norwell, MA, USA: Kluwer, 2003.

[32] F. Orabona, J. Keshet, and B. Caputo, “Bounded kernel-based online learning,” *J. Mach. Learn. Res.*, vol. 10, pp. 2643–2666, Dec. 2009.

[33] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet, “Simple MKL,” *J. Mach. Learn. Res.*, vol. 9, pp. 1179–1225, 2008.

[34] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain,” *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, 1958.

[35] S. Ross, P. Mineiro, and J. Langford, “*Normalized Online Learning*,” UAI, 2013.

[36] D. Sahoo, S. C. H. Hoi, and B. Li, “Online multiple kernel regression,” in *Proc. Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 293–302.

[37] B. Schölkopf and A. J. Smola, *Learning With Kernels*. Cambridge, U.K.: MIT Press, 2002.

[38] M. Seeger, C. K. I. Williams, and N. D. Lawrence, “Fast forward selection to speed up sparse Gaussian process regression,” in *Proc. Workshop AI & Statist. 9*, 2003.

[39] S. Smale and Y. Yao, “Online learning algorithms,” *Found. Comput. Math.*, vol. 6, no. 2, pp. 145–170, Apr. 2006.

[40] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf, “Large scale multiple kernel learning,” *J. Mach. Learn. Res.*, vol. 7, pp. 1531–1565, Jul. 2006.

[41] 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 Rec.*, vol. 30, no. 4, pp. 5–14, 2001.

[42] V. Van Vaerenberg, M. Lazaro-Gredilla, and I. Santamaria, “Kernel recursive least-squares tracker for time-varying regression,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 8, pp. 1313–1326, Aug. 2012.

[43] J. S. Vitter, “Random sampling with a reservoir,” *ACM Trans. Math. Softw.*, vol. 11, no. 1, pp. 37–57, 1985.

[44] Y. Wang, R. Khardon, D. Pechyony, and R. Jones, “Generalization bounds for online learning algorithms with pairwise loss functions,” in *Proc. Workshop Conf.*, 2012, pp. 13.1–13.22.

[45] X. Wu *et al.*, “Top 10 algorithms in data mining,” *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 1–37, 2008.

[46] 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.*, vol. 36, no. 3, pp. 536–549, Mar. 2014.

[47] Z. Xu, R. Jin, H. Yang, I. King, and M. R. Lyu, “Simple and efficient multiple kernel learning by group lasso,” in *Proc. ICML*, 2010, pp. 1175–1182.

[48] L. Yan, R. H. Dodier, M. Mozer, and R. Wolniewicz, “Optimizing classifier performance via an approximation to the Wilcoxon-Mann-Whitney statistic,” in *Proc. 20th Int. Conf. Mach. Learn. ICML*, 2003, pp. 848–855.

[49] H. Yang and I. King, “Ensemble learning for imbalanced e-commerce transaction anomaly classification,” in *Proc. Neural Inf. Process.*, 2009, pp. 866–874.

[50] H. Yang, I. King, and M. R. Lyu, *Sparse Learning Under Regularization Framework*. San Francisco, CA, USA: Academic, Apr. 2011.

[51] H. Yang, Z. Xu, J. Ye, I. King, and M. R. Lyu, “Efficient sparse generalized multiple kernel learning,” *IEEE Trans. Neural Netw.*, vol. 22, no. 3, pp. 433–446, Mar. 2011.

[52] T. Yang, M. Mahdavi, R. Jin, J. Yi, and S. C. H. Hoi, “Online kernel selection: Algorithms and evaluations,” in *Proc. AAAI*, 2012.

[53] L. Zhang, J. Yi, R. Jin, M. Lin, and X. He, “Online kernel learning with a near optimal sparsity bound,” in *Proc. ICML*, pp. 621–629, 2013.

[54] P. Zhao, S. C. H. Hoi, and R. Jin, “Double updating online learning,” *J. Mach. Learn. Res.*, vol. 12, pp. 1587–1615, May 2011.

[55] P. Zhao, S. C. H. Hoi, R. Jin, and T. Yang, “Online AUC maximization,” in *Proc. ICML*, 2011, pp. 233–240.

Junjie Hu received the B.Eng. degree in computer science and technology from the South China University of Technology, Guangzhou, China, and the M.Phil. degree from the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

He is currently a Graduate Research Assistant with the Language Technologies Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA. His current research interests include machine learning, natural language processing, and human robot interaction.

Mr. Hu received the National Scholarship from the Ministry of Education of China twice from 2010 to 2012, the 2013 IBM Outstanding Student Scholarship, and the 2013 Outstanding Undergraduate Student Award from China Computer Federation. He served as a student helper in 2012 International Conference on Machine Learning and Cybernetics, 2013 International Conference on Wavelet Analysis and Pattern Recognition, and 2013 ACM Recommender Systems Conference.

Haiqin Yang (M'11) received the B.Sc. degree in computer science from Nanjing University, Nanjing, China, and the M.Phil. and Ph.D. degrees from the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

He is currently an Assistant Professor with the Department of Computing, Hang Seng Management College, Hong Kong. His current research interests include machine learning, data mining, and big data analytics. He has authored two books and over 40 technical publications in journals/conferences in his areas of expertise.

Dr. Yang has initiated and co-organized five international workshops on the topics of scalable machine learning and scalable data analytics. He currently serves on the Editorial Board of *Neurocomputing* and also serves as a Program Committee Member and a Reviewer of over 20 top-tier conferences/journals.

Michael R. Lyu (F'04) received the B.S. degree in electrical engineering from the National Taiwan University, Taipei, Taiwan, the M.S. degree in computer engineering from the University of California at Santa Barbara, Santa Barbara, CA, USA, and the Ph.D. degree in computer engineering from the University of California at Los Angeles, Los Angeles, CA.

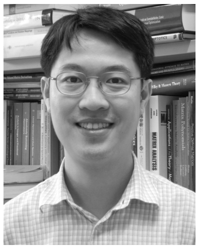
He was with the Jet Propulsion Laboratory, Pasadena, CA, Telcordia Technologies, Piscataway, NJ, USA, and the Bell Laboratory, Murray Hill, NJ, USA, and taught at The University of Iowa, Iowa City, IA, USA. He has participated in more than 30 industrial projects. He is currently a Professor with the Computer Science and Engineering Department, The Chinese University of Hong Kong, Hong Kong. He has authored close to 400 papers in the following areas. His current research interests include software engineering, distributed systems, multimedia technologies, machine learning, social computing, and mobile networks.

Dr. Lyu is a fellow of the American Association for the Advancement of Science. He received the best paper awards in ISSRE in 1998 and 2003, and the SigSoft Distinguished Paper Award in International Conference on Software Engineering in 2010. He initiated the International Symposium on Software Reliability Engineering (ISSRE), and was a Program Chair of ISSRE in 1996, the Program Co-Chair of the Tenth International World Web Conference, the Symposium on Reliable Distributed Systems in 2005, the International Conference on e-Business Engineering in 2007, and the International Conference on Services Computing in 2010. He was the General Chair of ISSRE in 2001, the Pacific Rim International Symposium on Dependable Computing in 2005, and the International Conference on Dependable Systems and Networks in 2011. He has been named by the IEEE Reliability Society as the Reliability Engineer of the Year in 2011, for his contributions to software reliability engineering and software fault tolerance.

Irwin King (SM'08) received the B.Sc. degree in engineering and applied science from the California Institute of Technology, Pasadena, CA, USA, and the M.Sc. and Ph.D. degrees in computer science from the University of Southern California, Los Angeles, CA.

He is currently the Associate Dean (Education) of the Faculty of Engineering, and a Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong. He was with AT&T Labs Research, Florham Park, NJ, USA, and also taught a number of courses at University of California at Berkeley, Berkeley, CA, as a Visiting Professor. His research interests include machine learning, social computing, big data, Web intelligence, data mining, and multimedia information processing. In these research areas, he has authored over 200 technical publications in top international journals and conferences. In addition, he has contributed over 30 book chapters and edited volumes. Moreover, he has over 30 research and applied grants and industry projects. Some notable projects include the VeriGuide system and the Knowledge and Education Exchange Platform.

Dr. King serves as the General Co-Chair of WSDM2011, RecSys2013, and ACML2015. He is an Associate Editor of the *ACM Transactions on Knowledge Discovery from Data* and the *Journal of Neural Networks*. Currently, he is serving as the Vice President and Governing Board Member of both the International Neural Network Society and the Asian Pacific Neural Network Assembly.



Anthony Man-Cho So (M'12) received the B.S.E. degree in computer science from Princeton University, Princeton, NJ, USA, with minors in applied and computational mathematics, engineering and management systems, and German language and culture, the M.Sc. degree in computer science from Stanford University, Stanford, CA, USA, and the Ph.D. degree in computer science with a minor in mathematics from Stanford University.

He joined The Chinese University of Hong Kong (CUHK), Hong Kong in 2007. He currently serves as the Assistant Dean of the Faculty of Engineering and also an Associate Professor with the Department of Systems Engineering and Engineering Management. He also holds a courtesy appointment as an Associate Professor with the CUHKBGI Innovation Institute of Trans-omics, CUHK. His current research interests include the interplay between optimization theory and various areas of algorithm design, such as computational geometry, machine learning, signal processing, bioinformatics, and algorithmic game theory.

Dr. Man-Cho So received the 2015 IEEE Signal Processing Society Signal Processing Magazine Best Paper Award, the 2014 IEEE Communications Society Asia–Pacific Outstanding Paper Award, the 2010 Institute for Operations Research and the Management Sciences Optimization Society Optimization Prize for Young Researchers, and the 2010 CUHK Young Researcher Award. He also received the 2008 Exemplary Teaching Award and the 2011, 2013, 2015 Dean's Exemplary Teaching Award from the Faculty of Engineering, CUHK, and the 2013 Vice-Chancellor's Exemplary Teaching Award from CUHK. He currently serves on the Editorial Board of the IEEE TRANSACTIONS ON SIGNAL PROCESSING, the *Journal of Global Optimization*, *Optimization Methods and Software*, and the *SIAM Journal on Optimization*. He has also served on the Editorial Board of *Mathematics of Operations Research*.