Adaptive Cost-Sensitive Online Classification

Peilin Zhao, Yifan Zhang, Min Wu, Steven C. H. Hoi, Mingkui Tan, and Junzhou Huang

Abstract—Cost-Sensitive Online Classification has drawn extensive attention in recent years, where the main approach is to directly online optimize two well-known cost-sensitive metrics: (i) weighted sum of sensitivity and specificity; (ii) weighted misclassification cost. However, previous existing methods only considered first-order information of data stream. It is insufficient in practice, since many recent studies have proved that incorporating second-order information enhances the prediction performance of classification models. Thus, we propose a family of cost-sensitive online classification algorithms with adaptive regularization in this paper. We theoretically analyze the proposed algorithms and empirically validate their effectiveness and properties in extensive experiments. Then, for better trade off between the performance and efficiency, we further introduce the sketching technique into our algorithms, which significantly accelerates the computational speed with quite slight performance loss. Finally, we apply our algorithms to tackle several online anomaly detection tasks from real world. Promising results prove that the proposed algorithms are effective and efficient in solving cost-sensitive online classification problems in various real-world domains.

Index Terms—Cost-Sensitive Classification; Online Learning; Adaptive Regularization; Sketching Learning.

1 Introduction

With the rapid growth of datasets, the technologies of machine learning and data mining power many respects of modern society: from content filtering to web searches on social networks, and from goods recommendations to intelligent customer services on e-commerce. Gradually, many real-world large-scale applications make use of a family of techniques called online learning, which has been extensively studied for many years in machine learning and data mining literatures [1], [2], [3], [4], [5]. In general, online learning is a class of efficient and scalable machine learning methods, whose goal is to incrementally learn a model to make correct predictions on a stream of samples. This family of methods provides an opportunity to solve many real-world applications that data arrives sequentially while predictions must be made instantly, such as malicious URL detection [6], [28] and portfolio selection [7]. In addition, online learning is also good at solving large-scale learning tasks, e.g., learning *support vector machine* from billions of data [8].

However, although online learning was studied widely, most existing methods were inappropriate to solve cost-sensitive classification problems, because most of them seek performance based on measurable *accuracy* or *mistake rate*, which are obviously cost-insensitive. As a result, these algorithms are difficult to handle numerous real-world problems, where datasets are always class-imbalanced, i.e., the mistake costs of samples are significantly different [9], [10], [11]. To solve this problem, researchers have suggested to use more meaningful metrics, such as the weighted sum of *sensitivity* and *specificity* [12], [13], and the weighted *misclassification cost* [9], [14] to replace old ones. Based on this, many batch classification algorithms are proposed to directly optimize prediction performance for cost-sensitive classification over the past decades [9], [14]. However, these batch algorithms often suffer from poor scalability and efficiency for large-scale tasks, which makes them inappropriate for online classification applications.

Although both *online classification* and *cost-sensitive classification* were studied widely, quite few literatures study cost-sensitive online classification. As results, the Cost-Sensitive Online Classification framework [15], [16] was recently proposed to fill the gap between online learning and cost-sensitive classification. According to this framework, a class of algorithms named as Cost-Sensitive Online Gradient Descend (COG) was proposed to directly optimize predefined cost-sensitive metrics (e.g., weighted sum or weighted misclassification cost) based on online gradient descent technique. Particularly, compared with other traditional online algorithms, COG shows strong empirical performance in solving cost-sensitive online classification problems.

However, although COG is able to handle the Cost-sensitive online classification tasks, it only take the first order information of samples (i.e., weighted mean of the gradient). It is obviously insufficient, since many recent studies [3], [17], [18], [19] have shown that comprehensive consideration with second-order information (i.e., the correlations between features) significantly enhances the performance of online classification.

As an attempt to remedy the limitation of first-order approaches, we propose the Adaptive Regularized Cost-Sensitive Online Gradient Descent algorithms (named ACOG), based on the state-of-the-art Confidence Weighted strategy [3], [17], [18], [19]. We theoretically analyze their regret bounds [20] and their cost-sensitive metric bounds. Corresponding conclusions confirm the good convergence of ACOG algorithms.

Furthermore, although enjoying the advantage of second-order information, our proposed algorithms are at

- *P. Zhao is with the South China University of Technology, China. E-mail: peilinzhao@hotmail.com.*
- *Y. Zhang is with the South China University of Technology, China. E-mail: sezyifan@mail.scut.edu.cn.*
- *M. Wu is with the Institute for Infocomm Research, Singapore. E-mail: wumin@i2r.a-star.edu.sg.*
- *S. C. Hoi is with the Singapore Management University, Singapore. E-mail: chhoi@smu.edu.sg.*
- *M. Tan is with the South China University of Technology, China. E-mail: mingkuitan@scut.edu.cn.*
- *J. Huang is with Tencent AI Lab, China. Email: joehhuang@tencent.com.*

the cost of higher running time, because the updating process of correlation matrix is time-consuming. As results, it may be inappropriate for some real-world applications with quite high-dimensional datasets. Thus, for better trade off between the efficiency and performance, we further propose an updated version of ACOG algorithms based on sketching techniques [21], [22], [23], [24], whose running time is linear in the dimensions of samples, just like the first order methods.

Next, we conduct extensive experiments to evaluate the performance and specialities of our proposed algorithms and then apply them to solve online anomaly detection tasks from several real-world domains. Promising results confirm the effectiveness and efficiency of our methods in real-world cost-sensitive online classification problems.

Note that a brief version of this paper had been published in the IEEE ICDM conference [25]. Compared with it, this journal manuscript makes several significant extensions, including (1) an updated variant with sketching methods and some theoretical analyses about its time complexity; (2) an extension of ACOG with an additional loss function and theoretical analyses; (3) more extensive empirical studies to evaluate the proposed algorithms.

The rest of this paper is organized as follows. We present the problem formulation and the proposed algorithms with theoretical analyses in section 2. To save space, we provide theorem proofs and related work in supplemental materials. Next, we propose an efficient version based on sketching techniques in section 3. After that, section 4 empirically evaluates the performance and properties of our algorithms, and section 5 shows an application to real-world anomaly detection tasks. Finally, section 6 concludes the paper.

2 Setup and Algorithm

In this section, we firstly introduce the framework and formulation setting of the Cost-Sensitive Online Classification problem [15], [16]. Then, we present the proposed Adaptively Regularized Cost-Sensitive Online Gradient Descent algorithms (ACOG) in detail.

2.1 Problem Setting

Without loss of generality, we consider online binary classification problems here. The main goal is to learn a linear classification model with an updatable predictive vector $w \in \mathbb{R}^d$, based on a stream of training samples $\{(x_1, y_1), (x_2, y_2), ..., (x_T, y_T)\}$, where T is the total quantity of samples, $x_t \in \mathbb{R}^d$ is the d-dimensional sample at time t, and $y_t \in \{-1, 1\}$ is the corresponding true class label. In detail, at the t-th round of learning, the learner obtains a sample x_t and then predicts its estimated class label $\hat{y}_t = \mathrm{sign}(w_t^\top x_t)$, where w_t is the model predictive vector learnt from the previous $t-1$ samples. Then, the model receives the ground truth of instance $y_t \in \{-1, 1\}$, which is the label of true class. If $\hat{y}_t = y_t$, the model makes a correct prediction; otherwise, it makes a mistake and suffers a loss. In the end, the learner updates its predictive vector w_t based on the received painful loss.

For convenience, we define $\mathcal{M} = \{t \mid y_t \neq \mathrm{sign}(w_t \cdot x_t), \forall t \in [T]\}$ is the mistake index set, $\mathcal{M}_p = \{t \in \mathcal{M}$ and $y_t = +1\}$ is the positive set of mistake index and $\mathcal{M}_n = \{t \in \mathcal{M}$ and $y_t = -1\}$ is the negative one. In addition, we set $M = |\mathcal{M}|$, $M_p = |\mathcal{M}_p|$ and $M_n = |\mathcal{M}_n|$ to denote the number of total mistakes, positive mistakes and negative mistakes. Moreover, we denote the index sets of all positive samples and all negative samples by $\mathcal{I}_T^p = \{i \in [T] | y_i = +1\}$ and $\mathcal{I}_T^n = \{i \in [T] | y_i = -1\}$, where $T_p = |\mathcal{I}_T^p|$ and $T_n = |\mathcal{I}_T^n|$ denote the number of positive samples and negative samples.

For performance metrics of this problem, we first assume the positive samples as rare class, i.e., $T_p \leq T_n$. Generally, traditional online classification approaches are eager to maximize accuracy (or minimize mistake rate equivalently):

$$accuracy = \frac{T - M}{T}.$$

However, this metric is inappropriate for imbalanced data, because models can easily obtain high accuracy, even simply classifying all imbalanced samples as negative class. So, a more suitable approach is to measure the sum of weighted $sensitivity$ and $specificity$:

$$sum = \alpha_p \times \frac{T_p - M_p}{T_p} + \alpha_n \times \frac{T_n - M_n}{T_n},$$

where $\alpha_p, \alpha_n \in [0, 1]$ are weight parameters for trade off between sensitivity and specificity, and $\alpha_p + \alpha_n = 1$. Note that if $\alpha_p = \alpha_n = 0.5$, the sum metric becomes the famous balanced $accuracy$ metric.

In addition, another metric to measure is the misclassification $cost$ suffered by the model:

$$cost = c_p \times M_p + c_n \times M_n,$$

where $c_p, c_n \in [0, 1]$ are misclassification cost parameters for positive and negative instances, and $c_p + c_n = 1$. Generally, either the higher of the sum value or the lower of the $cost$ value, the better performance of classification.

Then, we can adjust our focus to maximize sum metric or minimize $cost$ metric. As is known in [15], [16], both objectives are equivalent to minimizing the following objective:

$$\sum_{y_t=+1} \rho \mathbb{I}_{(y_t w \cdot x_t < 0)} + \sum_{y_t=-1} \mathbb{I}_{(y_t w \cdot x_t < 0)}, \tag{1}$$

where $\rho = \frac{\alpha_p T_n}{\alpha_n T_p}$ for weighted sum metric and $\rho = \frac{c_p}{c_n}$ for weighted $cost$ metric.

2.2 Algorithm

In this subsection, we present the proposed ACOG algorithms by optimizing the objective from Eq. (1). However, this objective function is non-convex. Thus, to facilitate the optimization, we replace the indicator function with its convex variants (either one of the following two functions):

$$\ell^I(w; (x, y)) = \max(0, (\rho * \mathbb{I}_{(y=1)} + \mathbb{I}_{(y=-1)}) - y(w \cdot x)), \tag{2}$$

$$\ell^{II}(w; (x, y)) = (\rho * \mathbb{I}_{(y=1)} + \mathbb{I}_{(y=-1)}) * \max(0, 1 - y(w \cdot x)). \tag{3}$$

For $\ell^I(w; (x, y))$, the change of margin yields more "frequent" updates for specific class, compared to the traditional hinge loss; while for $\ell^{II}(w; (x, y))$, the change of the slope causes to more "aggressive" updates for specific class.

Then, our aim is to minimize the regret of learning process [20], based on either loss functions $\ell^I(w; (x, y))$ or $\ell^{II}(w; (x, y))$:

$$Regret := \sum_{t=1}^{T} \ell(w_t; (x_t, y_t)) - \sum_{t=1}^{T} \ell(w^*; (x_t, y_t)), \tag{4}$$

where $w^* = \arg\min_t \sum_{t=1}^{T} \nabla\ell(w;(x_t,y_t))$. To solve this optimization problem, the cost-sensitive online gradient descent algorithms (COG) [15], [16] were proposed:

$$w_{t+1} = w_t - \eta\nabla\ell_t(w_t),$$

where η is the learning rate and $\ell_t(w_t) = \ell(w;(x_t,y_t))$. However, COG algorithms only consider the first order gradient information of the sample stream to update the learner, which is clearly insufficient since many recent studies have shown the significance of incorporating the second order information [3], [17], [18], [19]. Motivated by this discovery, we propose to introduce adaptive regularization to promote the cost-sensitive online classification.

Let us assume the online model satisfies a multivariate Gaussian distribution, i.e., $w \sim \mathcal{N}(\mu,\Sigma)$, where μ is the mean value vector of distribution and Σ is the covariance matrix of distribution. Then, we can predict the class label of an sample x based on $\mathrm{sign}(w^\top x)$, when given a definite multivariate Gaussian distribution. In reality, it is more practical to make predictions by simply using distribution mean $\mathbb{E}[w] = \mu$ rather than w. So, the rule of model prediction actually adopts $\mathrm{sign}(\mu^\top x)$ in the following. For better understanding, each mean value μ_i can be regarded as the model's knowledge about the feature i; while the diagonal entry of covariance matrix $\Sigma_{i,i}$ is regarded as the confidence of feature i. Generally, the smaller of $\Sigma_{i,i}$, the more confidence in the mean weight μ_i for feature i. In addition to diagonal values, other covariance terms $\Sigma_{i,j}$ can be understood as the correlations between two mean weight value μ_i and μ_j for feature i and j.

Given a multivariate Gaussian distribution, we naturally recast the object functions by minimizing the following unconstraint objective, based on the divergence between empirical distribution and probability distribution:

$$D_{KL}(\mathcal{N}(\mu,\Sigma)||\mathcal{N}(\mu_t,\Sigma_t)) + \eta\ell_t(\mu) + \frac{1}{2\gamma}x_t^\top\Sigma x_t,$$

where D_{KL} is the Kullback-Leibler divergence, η is fitting parameter and γ is regularized parameter. Specifically, this objective helps to reach trade off between distribution divergence (first term), loss function (second term) and model confidence (third term). In other word, the objective would like to make the least adjustment at each round to minimize the loss and optimize the confidence of model. To solve this optimization problem, we first depict the Kullback-Leibler divergence explicitly:

$$\begin{aligned}&D_{KL}\big(\mathcal{N}(\mu,\Sigma)||\mathcal{N}(\mu_t,\Sigma_t)\big)\\ &=\frac{1}{2}\log\Big(\frac{\det\Sigma_t}{\det\Sigma}\Big) + \frac{1}{2}\mathrm{Tr}(\Sigma_t^{-1}\Sigma) + \frac{1}{2}||\mu_t-\mu||^2_{\Sigma_t^{-1}} - \frac{d}{2}.\end{aligned}$$

However, this optimization function dose not have the closed-form solution. Thus, we change the loss term $\ell_t(\mu)$ with its first order Taylor expansion $\ell_t(\mu_t) + g_t^\top(\mu-\mu_t)$, where $g_t = \partial\ell_t(\mu_t)$. Now, we obtain the final optimization objective by removing constant terms:

$$f_t(\mu,\Sigma) = D_{KL}(\mathcal{N}(\mu,\Sigma)||\mathcal{N}(\mu_t,\Sigma_t)) + \eta g_t^\top\mu + \frac{1}{2\gamma}x_t^\top\Sigma x_t, \quad (5)$$

which is much easier to be solved.

A simple method to solve this objective function is to decompose it into two parts depending on μ and Σ, respectively. Then, the updates of mean vector μ and covariance matrix Σ can be performed independently:

- Update the mean parameter:

$$\mu_{t+1} = \arg\min_{\mu} f_t(\mu,\Sigma);$$

- If $\ell_t(\mu_t) \neq 0$, update the covariance matrix:

$$\Sigma_{t+1} = \arg\min_{\Sigma} f_t(\mu,\Sigma).$$

For the update of mean parameter, setting the derivative of $\partial_\mu f_t(\mu_{t+1},\Sigma)$ as zero will give:

$$\Sigma_t^{-1}(\mu_{t+1}-\mu_t) + \eta g_t = 0 \implies \mu_{t+1} = \mu_t - \eta\Sigma_t g_t,$$

while for covariance matrix, setting the derivative of $\partial_\Sigma f_t(\mu,\Sigma_{t+1})$ as zero will result in:

$$-\Sigma_{t+1}^{-1} + \Sigma_t^{-1} + \frac{x_t x_t^\top}{\gamma} = 0 \implies \Sigma_{t+1}^{-1} = \Sigma_t^{-1} + \frac{x_t x_t^\top}{\gamma},$$

where adopting the Woodbury identity [27] will give:

$$\Sigma_{t+1} = \Sigma_t - \frac{\Sigma_t x_t x_t^\top\Sigma_t}{\gamma + x_t^\top\Sigma_t x_t}. \quad (6)$$

Note that the update of mean parameter μ relies on the confidence parameter Σ, we thus propose to update μ based on the updated covariance matrix Σ_{t+1} instead of the old one Σ_t, which should be more accurate:

$$\mu_{t+1} = \mu_t - \eta\Sigma_{t+1}g_t. \quad (7)$$

This is different from AROW [19], where the updating rule of μ_t based on the old matrix Σ_t. To intuitively understand this change, let us assume Σ_{t+1} as a diagonal matrix. Then, we can find that the updating process actually assigns the updating value of each dimension with different self-adaptive learning rates. So, it is more appropriate to update μ, with the learning rate that considers the current sample. In other words, the more unconfident of the weight, the more aggressive of its updates. Then, we summarize the proposed Adaptive Regularized Cost-Sensitive Online Gradient Descent (ACOG) in Alogrithm 1.

Algorithm 1 Adaptive Regularized Cost-Sensitive Online Gradient Descent (ACOG)

Input learning rate η; regularized parameter γ; bias parameter $\rho = \frac{\alpha_p * T_n}{\alpha_n * T_p}$ for "sum" and $\rho = \frac{c_p}{c_n}$ for "cost".
Initialization $\mu_1 = 0$, $\Sigma_1 = I$.

```
for t = 1 → T do
    Receive sample x_t;
    Compute ℓ_t(μ_t)=ℓ*(μ_t;(x_t,y_t)), where * ∈ {I, II};
    if ℓ_t(μ_t) > 0 then
        Σ_{t+1} = Σ_t − (Σ_t x_t x_t^T Σ_t)/(γ + x_t^T Σ_t x_t);
        μ_{t+1} = μ_t − η Σ_{t+1} g_t, where g_t = ∂_μ ℓ_t(μ_t);
    else
        μ_{t+1} = μ_t, Σ_{t+1} = Σ_t;
    end if
end for
```

For simplification, we ignore the sample numbers T in the analyses of algorithms efficiency. Thus the time complexity for the updates of Σ_{t+1} and μ are both $\mathcal{O}(d^2)$, so the overall time complexity for ACOG is $\mathcal{O}(d^2)$, which is quite slower than the first order COG algorithms, especially for high-dimensional datasets. To reduce the time complexity, We propose to use the diagonal version of ACOG (i.e., ACOG$_{diag}$), which accelerates the speed of ACOG algorithms to $\mathcal{O}(d)$. Specifically, only a diagonal version Σ_t would be maintained and updated at round t, which can improve computational efficiency and save memory cost.

Remark. In ACOG algorithms, one practical concern is the setting of the value of ρ, when optimizing the weighted sum performance. Normally, ρ is denoted as $\rho = \frac{\alpha_p T_n}{\alpha_n T_p}$ for sum metric. However, the value of T_p might be unknown in advance on real-world online classification tasks. A practical method is to approximate the ratio $\frac{T_n}{T_p}$ according to the empirical distribution of the past training instances, and adaptively update $\frac{T_n}{T_p}$ during the online learning process. In addition, we would empirical examine this problem in experiments.

2.3 Theoretical Analysis

In this subsection, we theoretically analyze the proposed ACOG algorithms in terms of two cost-sensitive metrics. Before that, we first prove an important theorem, which gives the regret bounds for algorithms that contributes to later theoretical analyses.

Theorem 1. *Let $(x_1, y_1), (x_2, y_2), ..., (x_T, y_T)$ be a sequence of samples, where $x_t \in \mathbb{R}^d, y_t \in \{-1, 1\}$. Then for any $\mu \in \mathbb{R}^d$, by setting $\eta = \sqrt{\frac{\max_{t\le T}||\mu_t-\mu||^2 \text{Tr}(\Sigma_{T+1}^{-1})}{\gamma \log(|\Sigma_{T+1}^{-1}|)}}$, the proposed ACOG-I satisfies:*

$$Regret \le D_\mu \sqrt{\gamma \text{Tr}(\Sigma_{T+1}^{-1}) \log(|\Sigma_{T+1}^{-1}|)},$$

where $D_\mu = \max_t ||\mu_t - \mu||$. In addition, by setting $\eta = \sqrt{\frac{\max_{t\le T}||\mu_t-\mu||^2 \text{Tr}(\Sigma_{T+1}^{-1})}{\rho^2\gamma \log(|\Sigma_{T+1}^{-1}|)}}$, ACOG-II satisfies:

$$Regret \le \rho D_\mu \sqrt{\gamma \text{Tr}(\Sigma_{T+1}^{-1}) \log(|\Sigma_{T+1}^{-1}|)}.$$

Remark. Let us suppose $||x_t|| \le 1$, it is easy to discover $\text{Tr}(\Sigma_{T+1}^{-1}) \le O(T/\gamma)$, which means the regrets of ACOG are in the order of $O(\sqrt{T})$. This order of regret is the optimal, since the loss function is not strongly convex [42].

Theorem 2. *Under the same assumptions in the Theorem 1, by setting $\rho = \frac{\alpha_p T_n}{\alpha_n T_p}$, for any $\mu \in \mathbb{R}^d$ the ACOG-I satisfies:*

$$sum \ge 1 - \frac{\alpha_n}{T_n}[\sum_{t=1}^{T} \ell_t(\mu) + D_\mu \sqrt{\gamma \text{Tr}(\Sigma_{T+1}^{-1}) \log(|\Sigma_{T+1}^{-1}|)}],$$

and the ACOG-II satisfies:

$$sum \ge 1 - \frac{\alpha_n}{T_n}[\sum_{t=1}^{T} \ell_t(\mu) + \rho D_\mu \sqrt{\gamma \text{Tr}(\Sigma_{T+1}^{-1}) \log(|\Sigma_{T+1}^{-1}|)}].$$

Remark. It is easy to verify that $\sum_{t=1}^{T} \ell_t(\mu)$ is a convex estimate of $\rho M_p + M_n$ for μ, so $\frac{\alpha_n}{T_n}\sum_{t=1}^{T} \ell_t(\mu)$ is an estimate of $\alpha_p \frac{M_p}{T_p} + \alpha_n \frac{M_n}{T_n}$. In addition, it is worthy noting that α_n cannot be set as zero, since $\rho = \frac{\alpha_p T_n}{\alpha_n T_p}$. However, one limitation here is that we may not know $\frac{T_n}{T_p}$ in advance for a real-world online learning task. To solve this issue, an alternative approach is to consider the $cost$ metric, which does not need the $\frac{T_n}{T_p}$ term in advance because $\rho = \frac{c_p}{c_n}$.

Theorem 3. *Under the same assumptions in the Theorem 1, by setting $\rho = \frac{c_p}{c_n}$, for any $\mu \in \mathbb{R}^d$, the ACOG-I satisfies:*

$$cost \le c_n[\sum_{t=1}^{T} \ell_t(\mu) + D_\mu \sqrt{\gamma \text{Tr}(\Sigma_{T+1}^{-1}) \log(|\Sigma_{T+1}^{-1}|)}],$$

and the ACOG-II satisfies:

$$cost \le c_n[\sum_{t=1}^{T} \ell_t(\mu) + \rho D_\mu \sqrt{\gamma \text{Tr}(\Sigma_{T+1}^{-1}) \log(|\Sigma_{T+1}^{-1}|)}].$$

Remark. For the $cost$ metric, $\sum_{t=1}^{T} \ell_t(\mu)$ is a convex estimate of $\frac{c_p}{c_n} M_p + M_n$, and so $c_n \sum_{t=1}^{T} \ell_t(\mu)$ is an estimate of $c_p M_p + c_n M_n$. Moreover, one should note that c_n cannot be set as zero because of $\rho = \frac{c_p}{c_n}$.

3 Enhanced Algorithm with Sketching

As mentioned above, the time complexity of ACOG is $O(d^2)$ and its diagonal version is $O(d)$. However, the diagonal ACOG cannot enjoy the correlation information between different dimensions of samples. When instances have low *effective rank*, the regret bound of diagonal ACOG may be much worse than its full-matrix version due to the lack of enough dependance on the data dimensionality [23]. Unfortunately, real-world high-dimensional datasets are common to have such low rank settings with abundant correlations between features. So for those real-world datasets, it is more appropriate to choose the full matrix version. However, ACOG has one limitation that it will take a large amount of time, when receiving quite high-dimensional samples. To better balance the performance and the running time, we propose an enhanced version of our algorithms, named Sketched Adaptive Regularized Cost-Sensitive Online Gradient Descent (SACOG).

3.1 Sketched Algorithm

In this subsection, we will present the enhanced version of ACOG via **Oja's sketch method** [21], [40], [41], which is designed to accelerate computation efficiency when the second order matrix of sequential data is low rank.

In detail, the main idea of SACOG is to approximate the second covariance matrix Σ by a small number of carefully selective directions, called as a $sketch$.

According to Eq. (6-7), we know the updating rule of model parameter μ:

$$\mu_{t+1} = \mu_t - \eta \Sigma_{t+1} g_t,$$

and the incremental formula of covariance matrix:

$$\Sigma_{t+1}^{-1} = \Sigma_t^{-1} + \frac{x_t x_t^\top}{\gamma},$$

which can be expressed in another way:

$$\Sigma_{t+1}^{-1} = I_d + \sum_{i=1}^{t} \frac{x_i x_i^\top}{\gamma}, \tag{8}$$

where d is the dimensionality of instance.

Let $X_t \in \mathbb{R}^{t\times d}$ be a matrix, whose t-th row is $\hat{x}_t^\top$, where we define $\hat{x}_t = \frac{x_t}{\sqrt{\gamma}}$ as the $to\text{-}sketch\ vector$. Then, the Eq.(8) can be written as:

$$\Sigma_{t+1}^{-1} = I_d + X_t^\top X_t.$$

Now, we define $S_t \in \mathbb{R}^{m\times d}$ as sketch matrix to approximate X_t, where the sketch size $m \ll d$ is a small constant.

When m is chosen so that $X_t^\top X_t$ can be approximated by $S_t^\top S_t$ well, the Eq.(8) can be redefined as:

$$\Sigma_{t+1}^{-1} = I_d + S_t^\top S_t.$$

Then by the Woodbury identity [27], we have:

$$\Sigma_{t+1} = I_d - S_t^\top H_t S_t, \tag{9}$$

where $H_t = (I_m + S_t S_t^\top)^{-1} \in \mathbb{R}^{m\times m}$. Then, we rewrite the updating rule of parameter μ:

$$\mu_{t+1} = \mu_t - \eta(g_t - S_t^\top H_t S_t g_t). \tag{10}$$

Based on above, we summarize Sketched ACOG in Algorithm 2.

Algorithm 2 Sketched Adaptive Regularized Cost-Sensitive Online Gradient Descent (SACOG)

Input learning rate η; regularized parameter γ; sketch size m; bias $\rho = \frac{\alpha_p * T_n}{\alpha_n * T_p}$ for "sum" and $\rho = \frac{c_p}{c_n}$ for "cost".
Initialization $\mu_1 = 0$, sketch$(S_0, H_0) \leftarrow$ **SketchInit**(m).
for $t = 1 \to T$ **do**
  Receive sample x_t;
  Compute $\ell_t(\mu_t) = \ell^*(\mu_t; (x_t, y_t)), where * \in \{I, II\}$;
  Compute the t-$sketch\ vector$ $\hat{x}_t = \frac{x_t}{\sqrt{\gamma}}$;
  $(S_t, H_t) \leftarrow$ **SketchUpdate**$(\hat{x})$;
  if $\ell_t(\mu_t) > 0$ **then**
    $\mu_{t+1} = \mu_t - \eta(g_t - S_t^\top H_t S_t g_t),\ where\ g_t = \partial_\mu \ell_t(\mu_t)$;
  else
    $\mu_{t+1} = \mu_t$.
  end if
end for

Then we discuss how to maintain the matrices S_t and H_t efficiently via sketching technique, where we compute eigenvalues and eigenvectors of sequential data through online gradient descent with to-$sketch\ vector$ $\hat{x}_t$ as input.

In detail, let the diagonal matrix $\Lambda_t \in \mathbb{R}^{m\times m}$ contain the approximated eigenvalues and $V_t \in \mathbb{R}^{m\times d}$ be the estimated eigenvectors at round t. According to Oja's algorithm [40], [41], the updating rules of Λ_t and V_t are defined as:

$$\Lambda_t = (I_m - \Gamma_t)\Lambda_{t-1} + \Gamma_t diag\{V_{t-1}\hat{x}_t\}^2, \tag{11}$$

$$V_t \xleftarrow{orth} V_{t-1} + \Gamma_t V_{t-1}\hat{x}_t\hat{x}_t^\top, \tag{12}$$

where learning rate $\Gamma_t = \frac{1}{t} I_m \in \mathbb{R}^{m\times m}$ is a diagonal matrix, and $\xleftarrow{orth}$ represents an orthonormalizing step[1]. Then, the sketch matrices can be obtained by:

$$S_t = (t\Lambda)^{\frac{1}{2}} V_t, \tag{13}$$

$$H_t = diag\{\frac{1}{1 + t\Lambda_{1,1}}, ..., \frac{1}{1 + t\Lambda_{m,m}}\}.$$

Since the rows of S_t are always orthogonal, H_t is an efficiently maintainable diagonal matrix all the way. We summarize the Oja's sketching technique in Algorithm 3.

1. For sake of simplicity, $V_t + \Gamma_{t+1} V_t \hat{x}_t \hat{x}_t^\top$ is assumed as full rank with rows all the way, so that the $\xleftarrow{orth}$ operation always keeps the same dimensionality of V_t.

Algorithm 3 Oja's Sketch for SACOG

Input m, $\hat{x}$ and stepsize matrix Γ_t.
Internal State t, Λ, V and H.
SketchInit(m)
  Set $t = 0, S = 0_{m\times d}, H = I_m, \Lambda = 0_{m\times m}$ and V to any $m \times d$ matrix with orthonormal rows;
  Return (S, H).

SketchUpdate$(\hat{x})$
  Update $t \leftarrow t + 1$;
  Update $\Lambda = (I_m - \Gamma_t)\Lambda + \Gamma_t diag\{V\hat{x}\}^2$;
  Update $V \xleftarrow{orth} V + \Gamma_t V \hat{x}\hat{x}^\top$;
  Set $S = (t\Lambda)^{\frac{1}{2}} V$;
  Set $H = diag\{\frac{1}{1+t\Lambda_{1,1}}, ..., \frac{1}{1+t\Lambda_{m,m}}\}$;
  Return (S, H).

Remark. The time complexity of this algorithm is $O(m^2 d)$ per round because of the orthonormalizing operation, and one can update the sketch every m rounds to improve time complexity to $O(md)$ [43]. Another concern is the regret guarantee, which is not clear now because existing analysis for Oja's algorithm is only for the stochastic situation [21]. However, SACOG provides good empirical performance.

3.2 Sparse Sketched Algorithm

However, even via sketching, SACOG algorithms are still quite slower than most online first order methods, because they cannot enjoy the sparse information of samples while first-order algorithms can. The question is that in many real-world applications, the samples are normally high sparse that the number of nonzero elements satisfies $||x||_0 \leq s$ with some small constants $s \ll d$.

As results, many first order methods can enjoy a per-round running time depending on s rather than d. But for SACOG, even when samples are sparse, the sketch matrix S_t still becomes dense quickly, because of the orthonormalizing updating of V_t. For this reason, the updates of μ_t cannot enjoy the sparsity of samples. To handle this question, we propose an enhanced sparse version of SACOG to achieve a purely sparsity-dependent time cost.

The main idea is that we adjust the formulations of eigenvector V_t and predictive vector μ_t, so that the updates of them are always sparse. In detail, there are two key modifications for SACOG: (1) The Eigenvectors V_t are modified as $V_t = F_t Z_t$, where $F_t \in \mathbb{R}^{m\times m}$ is an orthonormalizing matrix so that $F_t Z_t$ is orthonormal, and $Z_t \in \mathbb{R}^{m\times d}$ is a sparsely updatable direction. (2) The weights μ_t fall into two parts $\mu_t = w_t + Z_{t-1}^\top b_t$, where $w_t \in \mathbb{R}^d$ captures the sparsely updating weights on the complementary subspace, and $b_t \in \mathbb{R}^m$ captures the weights on the subspace form V_{t-1} (same as Z_{t-1}). Then, we describe how to sparsely update two weight parts w_t and b_t. Firstly, from Eq. (13), we know $S_t = (t\Lambda)^{\frac{1}{2}} V_t = (t\Lambda)^{\frac{1}{2}} F_t Z_t$. Then, we have:

$$\begin{aligned}\mu_{t+1} =& \mu_t - \eta(I_d - S_t^\top H_t S_t) g_t \\ =& w_t + Z_{t-1}^\top b_t - \eta g_t + \eta Z_t^\top F_t^\top (t\Lambda H_t) F_t Z_t g_t \\ =& [\underbrace{w_t - \eta g_t - (Z_t - Z_{t-1})^\top b_t}_{w_{t+1}}] + Z_t^\top [\underbrace{b_t + \eta F_t^\top (t\Lambda H_t) F_t Z_t g_t}_{b_{t+1}}].\end{aligned}$$

According to this, we can define the updating rule of w_t:

$$\begin{aligned} w_{t+1} =& w_t - \eta g_t - (Z_t - Z_{t-1})^\top b_t \\ =& w_t - \eta g_t - \hat{x}_t \delta_t^\top b_t, \end{aligned} \tag{14}$$

and the updating rule of b_t:

$$b_{t+1} = b_t + \eta F_t^\top (t\Lambda_t H_t) F_t Z_t g_t. \tag{15}$$

Based on above, we summarize the sparse SACOG in Algorithm 4.

Algorithm 4 Sparse Sketched Adaptive Regularized Cost-Sensitive Online Gradient Descent (SACOG)

Input learning rate η; regularized parameter γ; sketch size m; bias $\rho = \frac{\alpha_p * T_n}{\alpha_n * T_p}$ for "sum" and $\rho = \frac{c_p}{c_n}$ for "cost".
Initialization $w_1 = 0_{d\times 1}$, $b_1 = 0_{m\times 1}$;
Initialization Sketch $(\Lambda_0, F_0, Z_0, H_0) \leftarrow$ **SketchInit**(m);
for $t = 1 \rightarrow T$ **do**
  Receive sample x_t;
  Compute $\ell_t(\mu_t) = \ell^*(\mu_t; (x_t, y_t)), where * \in \{I, II\}$;
  Compute the t-*sketch vector* $\hat{x}_t = \frac{x_t}{\sqrt{\gamma}}$;
  $(\Lambda_t, F_t, Z_t, H_t, \delta_t) \leftarrow$ **SketchUpdate**$(\hat{x})$;
  if $\ell_t(\mu_t) > 0$ **then**
    $w_{t+1} = w_t - \eta g_t - \hat{x}_t \delta_t^\top b_t$;
    $b_{t+1} = b_t + \eta F_t^\top (t\Lambda_t H_t) F_t Z_t g_t$;
    $\mu_{t+1} = w_{t+1} + Z_t^\top b_{t+1}$;
  else
    $\mu_{t+1} = \mu_t,\ w_{t+1} = w_t,\ b_{t+1} = b_t$.
  end if
end for

Next, we describe how to update Λ_t, F_t and Z_t. First, we rewrite the updating rule of eigenvalues Λ_t from Eq. (11):

$$\Lambda_t = (I_m - \Gamma_t)\Lambda_{t-1} + \Gamma_t diag\{F_{t-1} Z_{t-1} \hat{x}_t\}^2. \tag{16}$$

Then from Eq. (12), we have:

$$\begin{aligned} F_t Z_t \xleftarrow{orth} & F_{t-1} Z_{t-1} + \Gamma_t F_{t-1} Z_{t-1} \hat{x}_t \hat{x}_t^\top, \\ =& F_{t-1}(Z_{t-1} + F_{t-1}^{-1} \Gamma_t F_{t-1} Z_{t-1} \hat{x}_t \hat{x}_t^\top). \end{aligned} \tag{17}$$

We let $Z_t = Z_{t-1} + \delta_t \hat{x}_t^\top$, where $\delta_t = F_{t-1}^{-1} \Gamma_t F_{t-1} Z_{t-1} \hat{x}_t$ (note that F_t is always invertible because of Footnote 1). Now, it is easy to note that $Z_t - Z_{t-1}$ is a sparse rank-one matrix, which represents the update of w_t is efficient.

Finally, for the update of F_t so that $F_t Z_t$ is also orthonormalizing, we apply the Gram-Schmidt algorithm to F_{t-1} in a Banach space, where the inner product is defined as $\langle a, b \rangle = a^\top K_t b$ and $K_t = Z_t Z_t^\top$ is the Gram matrix (See Algorithm 6). Then, we can update K_t efficiently based on the update of Z_t:

$$\begin{aligned} K_t =& Z_t Z_t^\top, \\ =& (Z_{t-1} + \delta_t \hat{x}_t^\top)(Z_{t-1} + \delta_t \hat{x}_t^\top)^\top, \\ =& K_{t-1} + Z_{t-1} \hat{x}_t \delta_t^\top + \delta_t \hat{x}_t^\top Z_{t-1}^\top + \delta_t \hat{x}_t^\top \hat{x}_t \delta_t^\top. \end{aligned} \tag{18}$$

we summarize the Sparse Oja's algorithm for SACOG in Algorithm 5.

Remark. Note that the most time-consuming step is the update of F_t (See line 3 in Algorithm 6), which is $O(m^3)$. In addition, the time complexity for update of w_t is $O(ms)$ and that of b_t is $O(m^2 + ms)$. Thus, the overall time complexity

Algorithm 5 Sparse Oja's Sketch for SACOG

Input m, $\hat{x}$ and stepsize matrix Γ_t.
Internal State t, Λ, F, Z, K and H.
SketchInit(m)
 Set $t = 0, F = K = H = I_m, \Lambda = 0_{m\times m}$ and Z to any $m \times d$ matrix with orthonormal rows;
 Return (Λ, F, Z, H).

SketchUpdate$(\hat{x})$
 Update $t \leftarrow t + 1$;
 $\Lambda = (I_m - \Gamma_t)\Lambda + \Gamma_t diag\{FZ\hat{x}\}^2$;
 Set $\delta = F^{-1}\Gamma_t FZ\hat{x}^\top$;
 $K \leftarrow K + Z\hat{x}\delta^\top + \delta\hat{x}^\top Z^\top + \delta\hat{x}^\top\hat{x}\delta^\top$;
 $Z \leftarrow Z + \delta\hat{x}^\top$;
 $(L, Q) \leftarrow$ Decompose(F, K), where $LQZ = FZ$ and QZ is orthogonal;
 Set $F = Q$;
 Set $H = diag\{\frac{1}{1+t\Lambda_{1,1}}, ..., \frac{1}{1+t\Lambda_{m,m}}\}$;
 Return $(\Lambda, F, Z, H, \delta)$.

Algorithm 6 Decompose(F, K)

Input $F \in \mathbb{R}^{m\times m}$ and Gram matrix $K = ZZ^\top \in \mathbb{R}^{m\times m}$;
Initialization $L = 0_{m\times m}$ and $Q = 0_{m\times m}$;
1: **for** $i = 1 \rightarrow m$ **do**
2:   Let $f^\top$ be the i-th row of F;
3:   Compute $\alpha = QKf$, $\beta = f - Q^\top\alpha$ and $c = \sqrt{\beta^\top K\beta}$;
4:   **if** $c \neq 0$ **then**
5:     Insert $\frac{1}{c}\beta^\top$ to the i-th row of Q;
6:   **end if**
7:   Set the i-th entry of α to be c;
8:   Insert α to the i-th row of L;
9: **end for**
10: Delete the all-zero columns of L and all-zero rows of Q;
11: Return (L, Q).

of sparse ACOG per round is $O(m^3 + ms)$. One can improve the running time per round to $O(m^2 + ms)$ by only updating the sketch every m rounds. To the best of our knowledge, this is the first time that sparse Oja's sketch method is applied to the cost-sensitive online classification problem.

4 Experiments

In this section, we first evaluate the performance and characteristics of the original algorithms (i.e., ACOG and its diagonal version). After that, we further evaluate the effectiveness and efficiency of sketched variants (i.e., SACOG and its sparse version).

4.1 Experimental Testbed and Setup

At the beginning, we compare ACOG and its diagonal variant, with several famous standard online learning algorithms as follows: (1) Perceptron Algorithm [1], [37]; (2) Relaxed Online Maximum Margin Algorithm [38] ("ROMMA"); (3) Passive-Aggressive algorithm [35] ("PA-I" and "CPA-PB"); (4) Perceptron Algorithm with Uneven Margin [39] ("PAUM"); (5) Adaptive Regularization of Weight Vector [19] ("AROW"); (6) Cost-Sensitive Online Gradient Descent [15], [16] ("COG-I" and "COG-II"), from which ACOG was derived. All algorithms were evaluated on 4 benchmark datasets as listed in Table 1, which are obtained from LIBSVM[2].

For data preprocessing, all samples are normalized by $x_t \leftarrow \frac{x_t}{\|x_t\|_2}$, which is extensively used in online learning, since samples are obtained sequentially.

For a valid comparison, all algorithms used the same experimental settings. We set $\alpha_p = \alpha_n = 0.5$ for sum, and $c_p = 0.9$ and $c_n = 0.1$ for cost. The value of ρ was set to $\frac{\alpha_p * T_n}{\alpha_n * T_p}$ for sum and $\frac{c_p}{c_n}$ for cost, respectively. For CPA$_{PB}$ algorithm, $\rho(-1,1)$ was set to 1, and $\rho(1,-1)$ was ρ. For PAUM, the uneven margin was set to ρ. In addition, the parameter of C for PA-I, learning rate λ for COG and learning rate η for all our proposed algorithms were selected from $[10^{-5}, 10^{-4}, ..., 10^{5}]$. The regularized parameter γ for AROW and all our algorithms were set as 1.

On each dataset, experiments were conducted over 20 random permutations of instances. Results are reported through the average performance of 20 runs and evaluated by 4 metrics: $sensitivity$, $specificity$, the weighted sum of sensitivity and specificity, and the weighted $cost$ of misclassification. All algorithms were implemented in MATLAB on a 3.40GHz Winodws machine.

TABLE 1: List of Binary Datasets in Experiments

Dataset	#Examples	#Features	#Pos:#Neg
covtype	581012	54	1:1
german	1000	24	1:2.3
a9a	48842	123	1:3.2
ijcnn1	141691	22	1:9.4

4.2 Evaluation with Sum Metrics

4.2.1 Evaluation of Weighted Sum Performance

First of all, we aim to evaluate the weighted sum performance of ACOG and its diagonal version. Table 2 summaries the experimental results on 4 datasets, and Fig. 1 shows the development of online average sum performance on all datasets, respectively.

From Fig. 1 and Table 2, we can find that second-order algorithms (i.e., our proposed ACOG algorithms and regular AROW algorithm) outperform first-order algorithms on almost all datasets. This confirms the effectiveness of introducing the second order information into online classification. At the same time, ACOG algorithms significantly outperform all other online learning algorithms including AROW on all datasets, which confirms the superiority of combination between the second order information and cost-sensitive online classification.

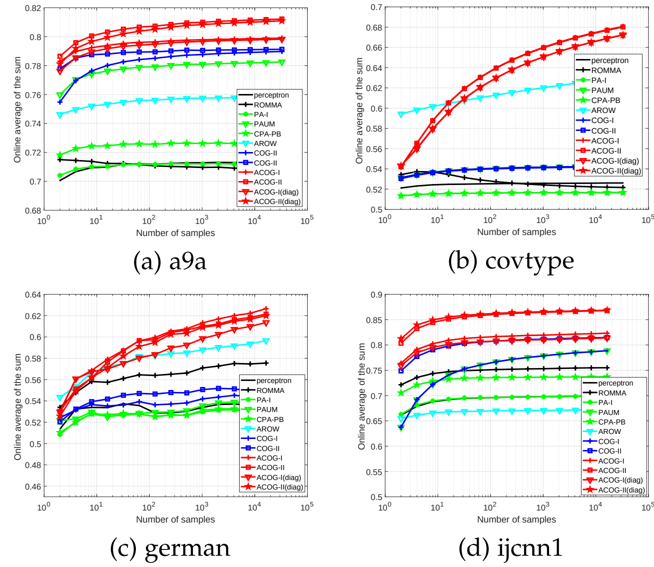
Then by evaluating both $sensitivity$ and $specificity$ metrics, our proposed algorithms not only achieve the best $sensitivity$ on all datasets, but also produce a fairly good $specificity$ for most datasets. This implies the proposed ACOG approaches are effective in improving prediction accuracy for rare class samples.

Moreover, while ACOG$_{diag}$ algorithms achieve smaller sum than ACOG algorithms, their computations are faster. This indicates the diagonal ACOG algorithms have ability to balance the effectiveness and efficiency.

4.2.2 Evaluation of Sum under Varying Weights

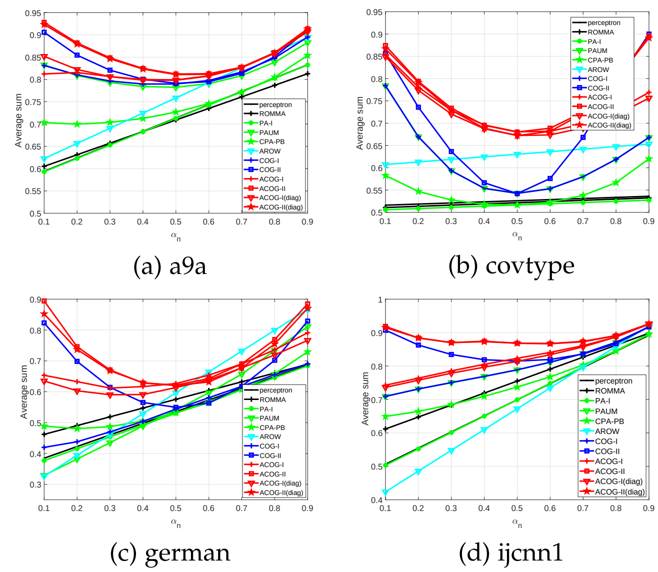
In this subsection, we would like to evaluate the sum of proposed methods under different cost-sensitive weights.

2. https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/



(a) a9a (b) covtype (c) german (d) ijcnn1

Fig. 1: Evaluation of online "sum" performance of the proposed algorithms on public datasets.



(a) a9a (b) covtype (c) german (d) ijcnn1

Fig. 2: Evaluation of weighted "sum" performance under varying weights of sensitivity and specificity.

Fig. 2 shows the empirical results under different weights of α_n and α_p. We find that our proposed algorithms consistently outperform all other algorithms under different values of weight on almost all datasets. This further validates the effectiveness of the proposed methods.

4.3 Evaluation with Cost Metrics

4.3.1 Evaluation of Weighted Cost Performance

Table 2 summaries the experimental performance of the ACOG$_{cost}$ on 4 datasets in terms of $cost$ metrics, and Fig. 3 illustrates the development of online $cost$ performance at each iteration.

By evaluating the cost performance in Fig. 3 and Table 2, our proposed methods achieve much lower misclassification $cost$ than other methods among all cases. For example, the

TABLE 2: Evaluation of the Cost-Sensitive Classification Performance of ACOG and Other Algorithms

Algorithm	"sum" on a9a				"$cost$" on a9a			
	Sum(%)	Sensitivity(%)	Specificity (%)	Time(s)	Cost(10^2)	Sensitivity(%)	Specificity (%)	Time(s)
Perceptron	71.343 ± 0.215	56.406 ± 0.327	86.280 ± 0.102	0.196	50.951 ± 0.382	56.406 ± 0.327	86.280 ± 0.102	0.191
ROMMA	70.904 ± 0.239	57.918 ± 0.493	83.889 ± 0.262	0.225	50.184 ± 0.361	57.989 ± 0.346	83.863 ± 0.227	0.224
PA-I	71.274 ± 0.169	56.310 ± 0.277	86.237 ± 0.113	0.212	51.068 ± 0.311	56.310 ± 0.277	86.237 ± 0.113	0.212
PAUM	78.255 ± 0.155	70.868 ± 0.345	85.643 ± 0.116	0.192	35.976 ± 0.346	70.868 ± 0.345	85.643 ± 0.116	0.197
CPA_{PB}	72.678 ± 0.209	62.818 ± 0.345	82.537 ± 0.145	0.254	42.517 ± 0.326	66.818 ± 0.285	79.503 ± 0.132	0.246
AROW	75.854 ± 0.188	58.858 ± 0.510	**92.849 ± 0.153**	5.591	45.931 ± 0.486	58.858 ± 0.510	**92.849 ± 0.153**	5.104
COG-I	78.978 ± 0.128	71.967 ± 0.264	85.990 ± 0.137	0.192	28.632 ± 0.263	79.390 ± 0.241	81.284 ± 0.107	0.190
COG-II	79.126 ± 0.103	81.038 ± 0.243	77.213 ± 0.168	0.201	25.527 ± 0.182	89.013 ± 0.171	62.398 ± 0.243	0.193
ACOG-I	79.903 ± 0.109	73.561 ± 0.347	86.244 ± 0.162	3.080	26.760 ± 0.291	81.129 ± 0.340	81.398 ± 0.219	2.837
ACOG-II	**81.220 ± 0.108**	**85.269 ± 0.219**	77.171 ± 0.134	3.344	**20.307 ± 0.169**	**94.079 ± 0.136**	62.107 ± 0.185	3.612
ACOG-I_{diag}	79.827 ± 0.094	73.361 ± 0.245	86.293 ± 0.103	0.202	26.917 ± 0.253	80.990 ± 0.282	81.369 ± 0.147	0.205
ACOG-II_{diag}	**81.098 ± 0.083**	**84.705 ± 0.227**	77.491 ± 0.152	0.216	**20.661 ± 0.110**	**93.352 ± 0.126**	63.212 ± 0.238	0.213
Algorithm	"sum" on covtype				"$cost$" on covtype			
	Sum(%)	Sensitivity(%)	Specificity (%)	Time(s)	Cost(10^2)	Sensitivity(%)	Specificity (%)	Time(s)
Perceptron	52.609 ± 0.057	51.364 ± 0.058	53.854 ± 0.057	1.649	1377.464 ± 1.638	51.364 ± 0.058	53.854 ± 0.057	1.662
ROMMA	52.164 ± 0.674	50.819 ± 0.731	53.509 ± 0.647	2.233	1391.250 ± 19.560	50.860 ± 0.702	53.541 ± 0.614	2.295
PA-I	51.666 ± 0.056	50.324 ± 0.061	53.008 ± 0.063	1.869	1406.500 ± 1.675	50.324 ± 0.061	53.008 ± 0.063	1.913
PAUM	54.268 ± 0.059	52.588 ± 0.089	55.949 ± 0.066	1.693	1340.022 ± 2.311	52.588 ± 0.089	55.949 ± 0.066	1.709
CPA_{PB}	51.667 ± 0.057	50.552 ± 0.063	52.781 ± 0.065	2.135	1194.433 ± 1.911	59.661 ± 0.070	44.275 ± 0.072	2.199
AROW	63.036 ± 0.033	60.158 ± 0.244	**65.914 ± 0.213**	22.640	687.696 ± 3.148	76.580 ± 0.137	**69.579 ± 0.134**	22.556
COG-I	54.268 ± 0.059	52.588 ± 0.089	55.949 ± 0.066	1.637	631.834 ± 1.721	84.036 ± 0.070	24.494 ± 0.062	1.710
COG-II	54.208 ± 0.051	54.038 ± 0.096	54.377 ± 0.055	1.643	426.122 ± 0.834	94.088 ± 0.031	7.501 ± 0.107	1.657
ACOG-I	**68.077 ± 0.038**	**70.565 ± 0.073**	**65.588 ± 0.082**	13.782	466.376 ± 1.190	90.693 ± 0.049	23.054 ± 0.038	18.988
ACOG-II	**68.020 ± 0.030**	**71.265 ± 0.070**	64.774 ± 0.068	13.528	**305.056 ± 0.355**	**98.969 ± 0.021**	6.365 ± 0.163	13.232
ACOG-I_{diag}	67.247 ± 0.060	69.183 ± 0.076	**65.311 ± 0.082**	1.824	469.701 ± 1.377	90.594 ± 0.090	22.782 ± 0.370	1.971
ACOG-II_{diag}	67.225 ± 0.062	69.913 ± 0.096	64.537 ± 0.086	1.805	**308.987 ± 7.944**	**98.739 ± 0.367**	7.015 ± 0.507	1.828
Algorithm	"sum" on german				"$cost$" on german			
	Sum(%)	Sensitivity(%)	Specificity (%)	Time(s)	Cost(10^2)	Sensitivity(%)	Specificity (%)	Time(s)
Perceptron	53.760 ± 1.655	35.133 ± 2.343	72.386 ± 0.977	0.003	1.945 ± 0.070	35.133 ± 2.343	72.386 ± 0.977	0.003
ROMMA	57.625 ± 2.943	43.550 ± 4.496	71.700 ± 1.710	0.004	1.721 ± 0.128	43.650 ± 4.372	71.536 ± 1.932	0.004
PA-I	53.043 ± 1.902	34.000 ± 2.818	72.086 ± 1.128	0.003	1.977 ± 0.083	34.000 ± 2.818	72.086 ± 1.128	0.003
PAUM	54.145 ± 1.335	26.483 ± 3.633	81.807 ± 1.341	0.003	2.112 ± 0.091	26.483 ± 3.633	81.807 ± 1.341	0.003
CPA_{PB}	53.185 ± 1.948	37.883 ± 2.925	68.486 ± 1.144	0.004	1.759 ± 0.082	44.317 ± 2.883	63.464 ± 1.287	0.004
AROW	59.948 ± 1.295	26.367 ± 3.893	**93.529 ± 1.630**	0.014	1.610 ± 0.082	43.867 ± 3.364	**86.571 ± 1.543**	0.016
COG-I	54.424 ± 1.474	36.083 ± 2.203	72.764 ± 0.807	0.003	1.770 ± 0.081	42.933 ± 3.010	67.200 ± 0.990	0.003
COG-II	54.952 ± 1.359	54.833 ± 1.318	55.071 ± 1.442	0.003	1.035 ± 0.033	81.067 ± 0.799	25.200 ± 1.983	0.003
ACOG-I	**63.150 ± 1.025**	49.050 ± 1.932	77.250 ± 1.489	0.008	1.232 ± 0.049	62.750 ± 2.017	67.671 ± 1.394	0.010
ACOG-II	62.511 ± 1.190	**63.000 ± 2.052**	62.021 ± 1.408	0.008	**0.875 ± 0.044**	**86.883 ± 2.264**	25.564 ± 4.099	0.011
ACOG-I_{diag}	61.765 ± 1.195	47.517 ± 2.610	76.014 ± 1.022	0.003	1.330 ± 0.064	58.967 ± 2.362	68.300 ± 0.901	0.003
ACOG-II_{diag}	62.281 ± 1.428	62.883 ± 1.852	61.679 ± 1.576	0.003	**0.912 ± 0.045**	**84.733 ± 0.876**	28.629 ± 4.046	0.003
Algorithm	"sum" on ijcnn1				"$cost$" on ijcnn1			
	Sum(%)	Sensitivity(%)	Specificity (%)	Time(s)	Cost(10^2)	Sensitivity(%)	Specificity (%)	Time(s)
Perceptron	69.988 ± 0.252	45.926 ± 0.455	94.051 ± 0.050	0.112	26.303 ± 0.221	45.926 ± 0.455	94.051 ± 0.050	0.114
ROMMA	75.547 ± 0.229	57.689 ± 0.439	93.405 ± 0.111	0.124	21.467 ± 0.207	57.666 ± 0.459	93.404 ± 0.108	0.128
PA-I	69.980 ± 0.312	45.542 ± 0.579	94.418 ± 0.083	0.119	26.305 ± 0.274	45.542 ± 0.579	94.418 ± 0.083	0.124
PAUM	79.066 ± 0.275	64.377 ± 0.590	93.755 ± 0.092	0.112	18.378 ± 0.239	64.377 ± 0.590	93.755 ± 0.092	0.118
CPA_{PB}	73.745 ± 0.209	57.328 ± 0.371	90.161 ± 0.091	0.155	23.096 ± 0.200	57.215 ± 0.407	90.233 ± 0.094	0.160
AROW	67.258 ± 0.460	36.208 ± 0.980	**98.308 ± 0.074**	0.450	28.626 ± 0.401	36.208 ± 0.980	**98.308 ± 0.074**	0.465
COG-I	79.066 ± 0.275	64.377 ± 0.590	93.755 ± 0.092	0.109	18.441 ± 0.236	64.171 ± 0.590	93.814 ± 0.096	0.116
COG-II	81.520 ± 0.232	81.940 ± 0.363	81.100 ± 0.182	0.112	16.398 ± 0.197	81.683 ± 0.311	81.394 ± 0.205	0.116
ACOG-I	82.375 ± 0.230	71.010 ± 0.607	93.740 ± 0.178	0.212	15.197 ± 0.123	71.996 ± 0.352	93.429 ± 0.102	0.218
ACOG-II	**86.872 ± 0.174**	**88.924 ± 0.323**	84.820 ± 0.218	0.288	**12.279 ± 0.149**	**87.626 ± 0.293**	84.770 ± 0.165	0.298
ACOG-I_{diag}	81.468 ± 0.225	69.007 ± 0.502	93.929 ± 0.092	0.114	15.681 ± 0.227	70.680 ± 0.624	93.631 ± 0.127	0.122
ACOG-II_{diag}	**86.929 ± 0.124**	**88.205 ± 0.266**	85.652 ± 0.107	0.120	**12.016 ± 0.111**	**87.164 ± 0.300**	85.801 ± 0.138	0.122

overall $cost$ of ACOG is about less than half of $cost$ made by all regular first-order algorithms (i.e., perceptron, ROMMA, PA-I, PAUM and CPA_{PB}). This implies that introducing the second order information is beneficial to the decrease of misclassification $cost$.

In addition, by examining both $sensitivety$ and $specificity$ metrics, we observe that our proposed methods often achieve the best $sensitivity$ result on all datasets, and attain a relatively good $specificity$ among all cases.

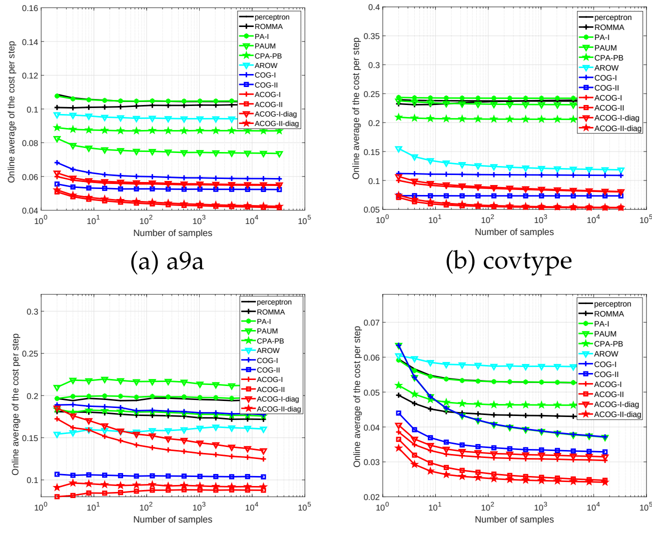
Moreover, the diagonal $ACOG_{diag}$ methods achieve higher $cost$ value than ACOG methods, but their running time are lower. This is similar with the situation based on sum metric. Thus, the $ACOG_{diag}$ methods can be regarded as a choice to balance the performance and efficiency.

4.3.2 Evaluation of Cost under Varying Weights

In this subsection, we examine the $cost$ performance under different cost-sensitive weights c_n and c_p for our proposed algorithms. From the results in Fig. 4, we observe that the proposed algorithms outperform almost all other algorithms under different weights. And only on a few datasets, AROW can achieve similar performance with our proposed methods. These discoveries imply that our ACOG algorithms have a wide selection range of weight parameters for online classification tasks.

4.4 Evaluation of Algorithm properties

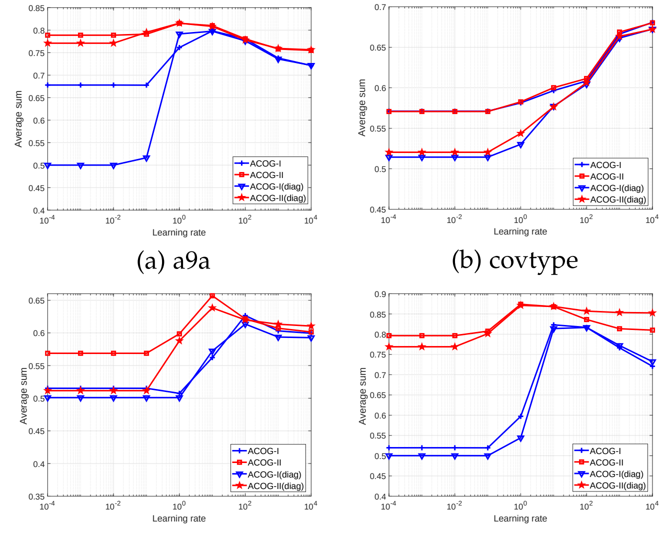
We have evaluated the performance of proposed algorithms in previous experiments, where promising results confirm their great superiority. Next, we are eager to examine their unique properties, including the influence of learning rate, regularized parameter, updating rule, online estimation and generalization ability. These examinations contribute to better understanding and applications of proposed methods. For simplicity, all experiments are based on sum metric, and every experiment only considers one objective or variable,



(a) a9a (b) covtype

(c) german (d) ijcnn1

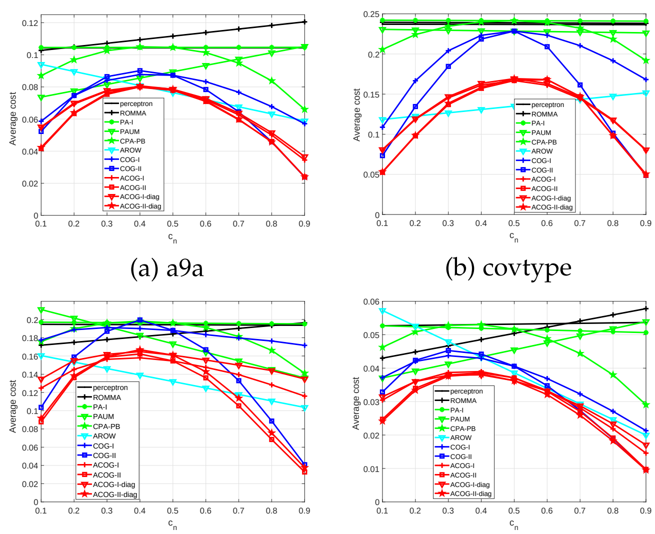
Fig. 3: Evaluation of online "cost" performance of the proposed algorithms on public datasets.



(a) a9a (b) covtype

(c) german (d) ijcnn1

Fig. 5: Performance under varying learning rates.



(a) a9a (b) covtype

(c) german (d) ijcnn1

Fig. 4: Evaluation of weighted "cost" performance under varying weights for False Positives and False Negatives.

while all other variable settings are fixed and similar with before experiments.

4.4.1 Evaluation of Learning Rate

In this subsection, we evaluate the influence of learning rate. In detail, we examine the sum performances of proposed methods with different learning rates η from $[10^{-4}, 10^{-3}, ..., 10^{3}, 10^{4}]$.

In Fig. 5, we find that ACOG algorithms would achieve relatively higher result, when we choose proper learning rate (i.e. relatively higher η in general). This is easy to understand because the values of covariance matrix Σ are normally small. Specifically, when a misclassification happened at time t, we update the predictive vector μ by $\mu_{t+1} = \mu_t + \eta \Sigma_{t+1} g_t$, where $g_t = \partial \ell_t(\mu_t)$. Because the values of covariance matrix Σ are normally small, the values of $\Sigma_{t+1} g_t$ thus are small. So if we want to obtain excellent performance, it would be better to choose properly higher learning rates as updating steps.

Moreover, we find the proposed methods with objective function $\ell^{II}(w; (x, y))$ can achieve relatively higher performance than the methods with $\ell^{I}(w; (x, y))$, which means that ACOG-II and ACOG-II$_{diag}$ are more robust to different learning rate η and consequently have a wider parameter choice space.

4.4.2 Evaluation of Regularized Parameter

Now, we aim to examine the influence of regularized parameters on our proposed algorithms.

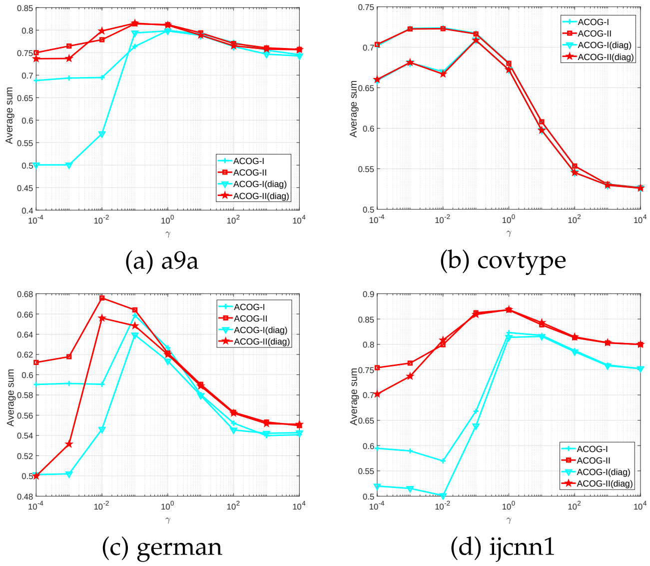
When the learner makes a mistake, we update the covariance matrix Σ by $\Sigma_{t+1} = \Sigma_t - \frac{\Sigma_t x_t x_t^\top \Sigma_t}{\gamma + x_t^\top \Sigma_t x_t}$ with default regularized parameter γ as 1. However, the rationality of this setting is not verified. Thus, we examine the performance of our algorithms with different regularized parameters γ from $[10^{-4}, 10^{-3}, ..., 10^{3}, 10^{4}]$ for sum metrics.

The results in Fig. 6 show that the optimal parameter normally is different according to datasets; while in most cases, the setting $\gamma = 1$ can achieve the best or fairly good results. This discovery confirms the practical value of our algorithms with default settings.

4.4.3 Evaluation of Updating Rule

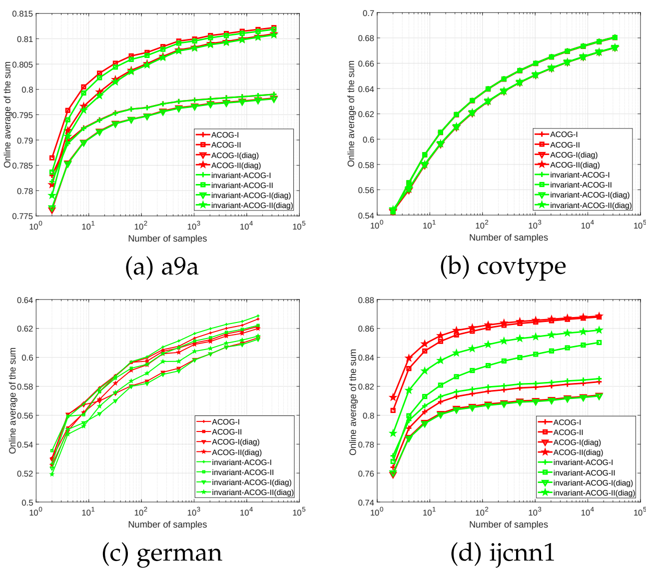
As mentioned in Section 2, the predictive vector μ is updated by $\mu_{t+1} = \mu_t + \eta \Sigma_{t+1} g_t$, which is different from AROW where the updating rule for μ relies on the old Σ_t. In this subsection, we would like to evaluate the difference between two updating rules based on sum metrics for proposed methods, where the invariant versions (i.e., green line in Fig. 7) depending on old Σ_t.

From Fig. 7, we find that although the difference between two updating rules is not obvious, the performance of Σ_{t+1} versions slightly exceed Σ_t versions, which is consistent with our analysis in Section 2.



(a) a9a (b) covtype (c) german (d) ijcnn1

Fig. 6: Performance under different regularized parameters.



(a) a9a (b) covtype (c) german (d) ijcnn1

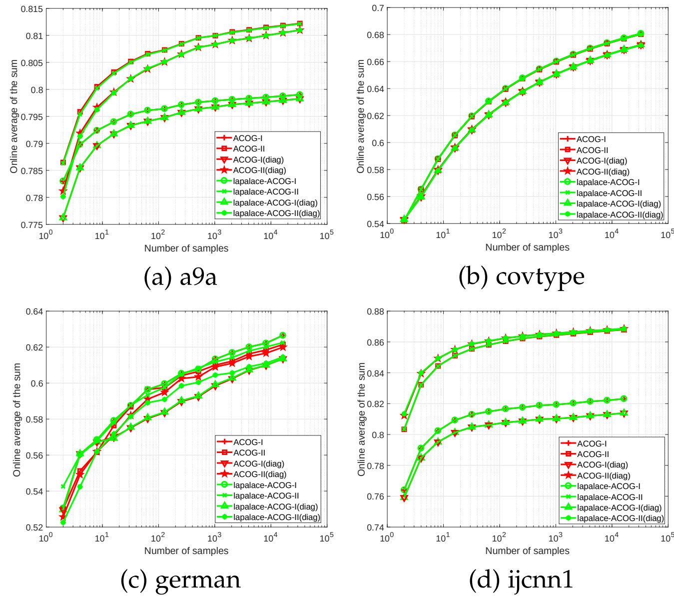
Fig. 7: Evaluation of updating rules.

4.4.4 Evaluation of Online Estimation of $\frac{T_n}{T_p}$

In the **remark** of Algorithm 1, we analyzed the parameter $\rho = \frac{\eta_p T_n}{\eta_n T_p}$ for ACOG$_{sum}$ algorithms, where the main question is that the value of T_p cannot be obtained in advance on real-world online learning.

Thus, we want to evaluate the influence of online estimation $\frac{T_n}{T_p}$ on sum performance, compared with the original algorithms. We adopt the widely used laplace estimation here, which estimates $\frac{T_n}{T_p}$ by $\frac{t_n+1}{t_p+1}$, where t_p and t_n represent the number of positive samples and negative samples at time t, respectively.

Fig. 8 shows the performance of online estimation. We find that the online laplace estimation performs quite similar results with the original one. This discovery validates the practical value of the proposed ACOG$_{sum}$ algorithms.



(a) a9a (b) covtype (c) german (d) ijcnn1

Fig. 8: Evaluation of online estimation of $\frac{T_n}{T_p}$.

TABLE 3: Evaluation of generalization ability with sum

Algorithm	a9a	covtype	german	ijcnn1
Perceptron	68.649	51.553	53.737	70.045
ROMMA	72.467	67.059	58.614	76.818
PA-I	71.986	51.283	51.363	70.410
PAUM	79.323	53.354	52.126	82.012
CPA$_{PB}$	73.668	51.279	52.768	73.942
AROW	75.961	64.928	54.575	67.642
COG-I	79.705	53.354	52.258	82.012
COG-II	78.559	68.897	50.784	82.849
ACOG-I	80.026	**72.428**	62.954	82.926
ACOG-II	**81.630**	**72.632**	60.928	**87.730**
ACOG-I$_{diag}$	80.118	71.051	**64.389**	82.334
ACOG-II$_{diag}$	**81.752**	71.311	**66.036**	**87.628**

4.4.5 Evaluation of Generalization Ability

Then, we evaluate the generalization ability of proposed methods, which may exist problems when converting an online algorithm to a batch training approach. We use 5-fold cross-validation for better validation of the general performance.

Table 3 summary the consequences on sum metrics, in which we discover that our proposed algorithms achieve the best among all algorithms on all datasets. This discovery indicates that our proposed methods have a strong generalized ability and can be regarded as a potentially useful tool to train large-scale cost-sensitive models.

4.5 Performance and Efficiency of Sketched ACOG

In the previous experiments, the evaluations of the proposed ACOG algorithms have shown promising results. However, we can find the implementation of ACOG is time consuming when facing high-dimensional datasets, because of the updating step for covariance matrix. As a result, it is difficult for engineers to address the real-world tasks with quite large-scale datasets.

A simple solution to this question is to implement the diagonal version of ACOG, and then enjoy linear time complexity. However, the gain of diagonal ACOG is at the cost of lower performance, because it abandons the correlation information between sample dimensions, which is quite important and indispensable for datasets with strong inner-correlation. Thus, for better trade off between performance and time efficiency, we propose the Sketched ACOG (named SACOG) and its sparse version (named SSACOG).

In this section, we first evaluate our sketched algorithms with several baseline algorithms: (1) "COG-I" and "COG-II"; (2) "ACOG-I" and "ACOG-II"; (3) "ACOG-I$_{diag}$" and "ACOG-II$_{diag}$", where we adopt 4 relatively high-dimensional datasets from LIBSVM, which are higher than 45 dimensions as list in Table 4. After that, we examine the performance difference between SACOG and SSACOG.

For simplicity, we focus on the case that the sketch size m is fixed as 5 for all sketched algorithms, although our methods can be easily generalized by setting different sketch sizes like [21]. Moreover, the learning rate was selected from $[10^{-5}, 10^{-4}, ..., 10^{5}]$, where other implementation details are similar with [21]. In addition, all experimental settings for other algorithms are same as previous experiments.

TABLE 4: Datasets for Evaluation of Sketched Algorithm

Dataset	#Examples	#Features	#Pos:#Neg
mushrooms	8124	112	1:1.07
protein	17766	357	1:1.7
usps	7291	256	1:5.11
Sensorless	58509	48	1:10

4.5.1 *Evaluation of Weighted Sum Performance*

In this subsection, we would like to examine the performance and efficiency of our sketched algorithms, where we adopt the sparse version (SSACOG) rather than the original SACOG, which is more appropriate for real-world datasets.

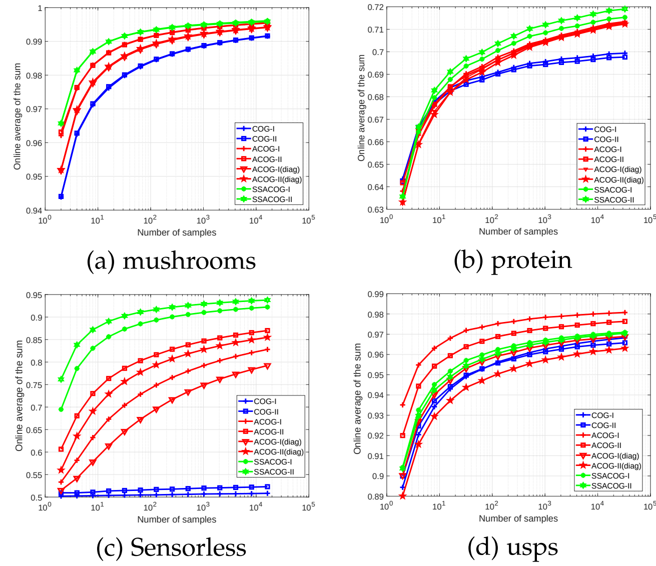
The results are summarized in Fig. 9, Fig. 10 and Table 5 based on two metrics, from which we find that the proposed SSACOG is much faster than ACOG algorithms, while the performance of sketched algorithms is not affected too much and sometimes even better. In addition, the degree of efficiency optimization by sketching technique goes up along with the increase of data dimensions, which is consistent with the common sense.

Note that although the running time of SSACOG is slower than ACOG$_{diag}$, it enjoys higher performance due to the advantage of sufficient second-order information, which confirms the superiority of ACOG with sketching technique.

4.5.2 *Efficiency Comparison between Sketched ACOG and Sparse Sketched ACOG*

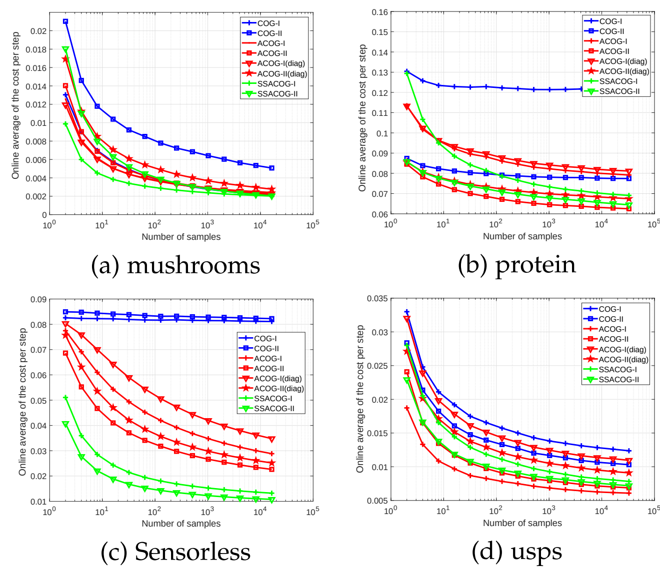
Then, We would like to compare the performance and running time between SACOG and its sparse version S-SACOG. The experimental results based on both metrics are summarized in Table 6.

From results, we find that the running time of SSACOG is lower than SACOG. It is consistent with the time complexity analysis of two algorithms in Section 3. For better understanding, we simply give a analysis. Given sketch size $m = 5$, the time complexity for SACOG is $O(25d)$ according to the analysis of Section 3, while the time complexity for SSACOG is $O(125 + 5s)$. One can accelerate the time complexity to $O(5d)$ for SACOG and $O(25 + 5s)$ for SSACOG by only updating the sketch every m round.



(a) mushrooms (b) protein (c) Sensorless (d) usps

Fig. 9: Weighted "sum" performance of SACOG.



(a) mushrooms (b) protein (c) Sensorless (d) usps

Fig. 10: Weighted "cost" performance of SACOG.

Thus, the time complexity for SACOG is linear in the data dimensionality d, and running time for SSACOG is linear in the data non-sparse degree s. Then, it is easy to understand the SSACOG would be much faster than SACOG, when the data dimensionality d is high and the data sparsity is strong $s \ll d$.

5 Application to Online Anomaly Detection

The proposed adaptive regularized cost-sensitive online classification algorithms can be potentially applied to solve a wide range of real-world applications in data mining. To verify their practical application value, we apply them to tackle several online anomaly detection tasks in this section.

5.1 Application Domains and Testbeds

Below, we first exhibit the related domains of anomaly detection problems:

TABLE 5: Evaluation of the Cost-Sensitive Classification Performance of SSACOG

Algorithm	“*sum*“ on mushrooms				“*cost*“ on mushrooms			
	Sum(%)	Sensitivity(%)	Specificity (%)	Time(s)	Cost	Sensitivity(%)	Specificity (%)	Time(s)
COG-I	99.205 ± 0.047	99.455 ± 0.075	98.956 ± 0.095	0.019	**15.760 ± 2.496**	99.823 ± 0.070	97.688 ± 0.091	0.020
COG-II	99.211 ± 0.057	99.420 ± 0.094	99.003 ± 0.097	0.019	39.180 ± 2.283	99.538 ± 0.055	94.465 ± 0.275	0.019
ACOG-I	99.580 ± 0.027	99.810 ± 0.070	99.350 ± 0.076	**0.043**	18.735 ± 1.062	99.939 ± 0.051	95.802 ± 0.443	**0.085**
ACOG-II	99.572 ± 0.033	99.794 ± 0.080	99.349 ± 0.075	**0.045**	16.770 ± 1.546	99.932 ± 0.035	96.373 ± 0.412	**0.054**
ACOG-I$_{diag}$	99.447 ± 0.052	99.652 ± 0.077	99.243 ± 0.087	0.019	17.520 ± 1.588	99.933 ± 0.045	98.119 ± 0.060	0.020
ACOG-II$_{diag}$	99.457 ± 0.052	99.652 ± 0.086	99.262 ± 0.117	0.019	21.185 ± 1.431	99.792 ± 0.037	96.601 ± 0.167	0.019
SSACOG-I	**99.628 ± 0.052**	99.798 ± 0.066	99.459 ± 0.102	**0.038**	**15.880 ± 1.677**	99.930 ± 0.043	96.623 ± 0.303	**0.041**
SSACOG-II	**99.606 ± 0.050**	99.805 ± 0.062	99.408 ± 0.093	**0.038**	**15.560 ± 3.870**	99.869 ± 0.040	97.291 ± 1.063	**0.034**
Algorithm	“*sum*“ on protein				“*cost*“ on protein			
	Sum(%)	Sensitivity(%)	Specificity (%)	Time(s)	Cost	Sensitivity(%)	Specificity (%)	Time(s)
COG-I	69.935 ± 0.213	68.114 ± 0.343	71.757 ± 0.322	0.127	2156.980 ± 35.558	75.944 ± 0.463	60.071 ± 0.270	0.151
COG-II	69.764 ± 0.230	70.005 ± 0.392	69.523 ± 0.415	0.129	1375.740 ± 12.402	90.618 ± 0.106	28.559 ± 0.607	0.152
ACOG-I	71.340 ± 0.214	69.794 ± 0.427	72.886 ± 0.385	**14.603**	1406.660 ± 28.314	87.072 ± 0.428	52.671 ± 0.576	**16.922**
ACOG-II	71.265 ± 0.235	71.678 ± 0.398	70.852 ± 0.501	**14.446**	**1110.075 ± 12.274**	94.972 ± 0.172	22.753 ± 0.853	**13.601**
ACOG-I$_{diag}$	71.305 ± 0.126	69.825 ± 0.346	72.785 ± 0.257	0.161	1441.505 ± 19.496	86.500 ± 0.269	53.441 ± 0.394	0.171
ACOG-II$_{diag}$	71.233 ± 0.150	71.530 ± 0.365	70.935 ± 0.298	0.158	1198.455 ± 11.459	92.585 ± 0.143	31.925 ± 0.685	0.166
SSACOG-I	**71.532 ± 0.198**	66.861 ± 0.530	76.203 ± 0.485	**0.355**	1227.345 ± 16.904	90.608 ± 0.225	44.148 ± 0.342	**0.393**
SSACOG-II	**71.323 ± 0.132**	71.725 ± 0.305	72.075 ± 0.403	**0.352**	1144.680 ± 13.087	94.053 ± 0.144	26.224 ± 0.661	**0.348**
Algorithm	“*sum*“ on Sensorless				“*cost*“ on Sensorless			
	Sum(%)	Sensitivity(%)	Specificity (%)	Time(s)	Cost	Sensitivity(%)	Specificity (%)	Time(s)
COG-I	50.888 ± 0.227	9.637 ± 0.473	92.139 ± 0.076	0.166	4741.190 ± 21.159	11.155 ± 0.403	90.823 ± 0.038	0.154
COG-II	52.374 ± 0.422	52.717 ± 0.464	52.032 ± 0.387	0.167	4801.600 ± 38.579	50.168 ± 0.415	54.576 ± 0.354	0.149
ACOG-I	83.468 ± 0.308	72.935 ± 0.620	94.001 ± 0.068	**0.503**	1622.600 ± 32.312	72.563 ± 0.709	94.188 ± 0.078	**0.480**
ACOG-II	87.398 ± 0.186	88.088 ± 0.284	86.708 ± 0.178	**0.486**	1283.350 ± 16.768	87.247 ± 0.264	87.350 ± 0.131	**0.455**
ACOG-I$_{diag}$	80.044 ± 0.314	66.427 ± 0.627	93.661 ± 0.051	0.169	1956.200 ± 33.729	65.995 ± 0.668	93.827 ± 0.071	0.157
ACOG-II$_{diag}$	85.968 ± 0.124	86.608 ± 0.178	85.328 ± 0.118	0.173	1422.950 ± 16.836	85.783 ± 0.227	86.043 ± 0.131	0.153
SSACOG-I	**92.432 ± 0.213**	89.818 ± 0.442	95.047 ± 0.047	**0.322**	**753.695 ± 21.185**	89.482 ± 0.476	95.296 ± 0.066	**0.285**
SSACOG-II	**93.913 ± 0.129**	94.487 ± 0.181	93.339 ± 0.123	**0.296**	**615.625 ± 12.280**	94.166 ± 0.194	93.676 ± 0.096	**0.264**
Algorithm	“*sum*“ on usps				“*cost*“ on usps			
	Sum(%)	Sensitivity(%)	Specificity (%)	Time(s)	Cost	Sensitivity(%)	Specificity (%)	Time(s)
COG-I	96.820 ± 0.165	96.361 ± 0.345	97.279 ± 0.116	0.039	90.165 ± 3.851	92.642 ± 0.344	98.179 ± 0.060	0.031
COG-II	96.576 ± 0.139	96.516 ± 0.226	96.637 ± 0.193	0.038	75.135 ± 4.338	96.570 ± 0.215	93.722 ± 0.342	0.030
ACOG-I	**98.073 ± 0.115**	97.822 ± 0.242	98.323 ± 0.095	**0.271**	**44.365 ± 3.448**	96.671 ± 0.321	98.591 ± 0.070	**0.151**
ACOG-II	**97.633 ± 0.148**	97.998 ± 0.230	97.268 ± 0.176	**0.239**	**50.100 ± 4.321**	98.241 ± 0.172	94.883 ± 0.488	**0.252**
ACOG-I$_{diag}$	96.886 ± 0.226	95.641 ± 0.435	98.131 ± 0.076	0.039	79.850 ± 4.773	93.526 ± 0.423	98.314 ± 0.080	0.031
ACOG-II$_{diag}$	96.305 ± 0.182	96.369 ± 0.228	96.240 ± 0.182	0.040	66.300 ± 3.242	96.993 ± 0.149	94.425 ± 0.295	0.030
SSACOG-I	97.091 ± 0.197	96.817 ± 0.323	97.365 ± 0.125	**0.077**	57.055 ± 4.251	95.657 ± 0.384	98.296 ± 0.076	**0.054**
SSACOG-II	97.048 ± 0.163	97.010 ± 0.237	97.085 ± 0.190	**0.074**	52.420 ± 4.009	97.647 ± 0.192	95.550 ± 0.360	**0.054**

TABLE 6: Evaluation between SACOG and Sparse SACOG

Algorithm	“*sum*“ on mushrooms		“*cost*“ on mushrooms		“*sum*“ on protein		“*cost*“ on protein	
	Sum(%)	Time(s)	Cost(10^2)	Time(s)	Sum(%)	Time(s)	Cost(10^2)	Time(s)
SACOG-I	99.620 ± 0.043	0.072	16.020 ± 1.796	0.096	71.544 ± 0.197	3.769	1226.890 ± 17.094	3.302
SACOG-II	99.598 ± 0.040	0.074	13.790 ± 1.852	0.035	71.907 ± 0.180	3.705	1147.775 ± 14.364	2.373
SSACOG-I	99.628 ± 0.052	0.038	15.880 ± 1.677	0.039	71.532 ± 0.198	0.287	1227.345 ± 16.904	0.272
SSACOG-II	99.606 ± 0.050	0.038	15.560 ± 3.870	0.033	71.900 ± 0.204	0.285	1144.680 ± 13.087	0.239
Algorithm	“*sum*“ on Sensorless		“*cost*“ on Sensorless		“*sum*“ on usps		“*cost*“ on usps	
	Sum(%)	Time(s)	Cost(10^2)	Time(s)	Sum(%)	Time(s)	Cost(10^2)	Time(s)
SACOG-I	92.432 ± 0.213	0.232	753.695 ± 21.185	0.235	97.146 ± 0.149	0.135	55.970 ± 3.053	0.078
SACOG-II	93.913 ± 0.129	0.193	615.625 ± 12.280	0.194	97.071 ± 0.169	0.091	53.155 ± 4.815	0.090
SSACOG-I	92.432 ± 0.213	0.239	753.695 ± 21.185	0.225	97.091 ± 0.197	0.057	57.055 ± 4.251	0.052
SSACOG-II	93.913 ± 0.129	0.214	615.625 ± 12.280	0.204	97.048 ± 0.163	0.054	52.420 ± 4.009	0.053

• Finance: The credit card approval problem enjoys a huge demand in financial domains, where our task is to discriminate the credit-worthy customers for the Australian dataset from an Australian credit company.

• Nuclear: We apply our algorithms to the Magic04 dataset with 19020 samples to simulate registration of high gamma particles. The dataset was collected by a ground-based atmospheric Cherenkov gamma telescope. In detail, the “gamma signal“ samples are considered as the normal class, while the hadron ones are treated as outliers.

• Bioinformatics: We address bioinformatics anomaly detection problems with DNA dataset to recognize the boundaries between exons and introns from a given DNA sequence, where exon/intron boundaries are defined as anomalies and others are treated as normal.

• Medical Imaging: We apply our approaches to address the medical image anomaly detection problem with the KDDCUP08 breast cancer dataset[3]. The main goal is to detect the breast cancer from X-ray images, where “benign“ is assigned as normal and “malignant“ is abnormal.

3. http://www.sigkdd.org/kddcup/

To better understand, we summary the detailed information for each dataset in Table 7.

TABLE 7: Datasets for Online Anomaly Detection

Dataset	#Examples	#Features	#Pos:#Neg
Australian	690	14	1:1.25
Magic04	19020	10	1:1.8
DNA	2000	180	1:3.31
KDDCUP08	102294	117	1:163.19

5.2 Empirical Evaluation Results

In this subsection, our algorithms are applied to address real-world anomaly detection tasks with 4 datasets from different domains, where we use the *balanced accuracy* metric to avoid inflated performance evaluations on imbalanced datasets. In addition, we apply our sparse sketched ACOG algorithms (SSACOG) only for two high-dimensional datasets (i.e., DNA and KDDCUP08), because for low-dimensional tasks, the proposed ACOG algorithms are fast enough. Furthermore, all implementation settings are same as Section 4.

Table 8 exhibits the experimental results, from which we can draw several observations. First of all, two cost-sensitive methods (PAUM and CPA_{PB}) outperform their regular methods (Perceptron and PA-I) among all datasets. This confirms the superiority of cost-sensitiveness for online learning. Second, COG algorithms outperform all regular first-order algorithms (i.e., first 5 baselines) on almost all datasets, which demonstrates the effectiveness of direct cost-sensitive optimization in online learning.

Moreover, ACOG algorithms and AROW algorithm outperform all other algorithms, where ACOG is the updated version of COG with adaptive regularization using second order information. This infers the online classification that introduces the second-order inner-correlation information can enjoy a huge performance improvement. Furthermore, the performance of ACOG exceeds all other algorithms, which demonstrates the effectiveness of cost-sensitive online optimization using the second order information.

By the way, although the speed of SSACOG is slightly slower than ACOG_{diag}, its performance is relatively better. On the other hand, SSACOG is much faster than ACOG with slight performance loss. This implies that the sketching version of ACOG is a good choice to balance the performance and efficiency for handling high-dimensional real-world tasks. Furthermore, if someone only wants to pursue the efficiency, they can regard ACOG_{diag} as a choice.

In conclusion, all promising results confirm the superiority of our proposed algorithms for real-world online anomaly detection problems, where datasets are normally high-dimensional and highly class-imbalanced.

TABLE 8: Evaluation for online anomaly detection

Algorithm	"sum" on Australian Sum(%)	Time(s)	"sum" on Magic04 Sum(%)	Time(s)
Perceptron	57.863 ± 1.327	0.002	59.154 ± 0.408	0.030
ROMMA	58.732 ± 3.462	0.002	64.025 ± 3.277	0.042
PA-I	57.103 ± 1.595	0.002	58.029 ± 0.312	0.036
PAUM	62.362 ± 0.941	0.002	64.671 ± 0.204	0.030
CPA_{PB}	57.110 ± 1.599	0.003	58.448 ± 0.360	0.043
AROW	67.174 ± 0.749	0.008	70.896 ± 0.190	0.154
COG-I	65.972 ± 0.879	0.002	65.913 ± 0.189	0.030
COG-II	67.213 ± 0.787	0.002	69.815 ± 0.183	0.030
ACOG-I	68.808 ± 0.894	0.005	72.935 ± 0.186	0.088
ACOG-II	**69.228 ± 0.733**	0.005	68.345 ± 1.822	0.092
ACOG-I_{diag}	68.464 ± 0.936	0.002	**73.268 ± 0.158**	0.033
ACOG-II_{diag}	68.510 ± 0.917	0.002	73.035 ± 0.187	0.033

Algorithm	"sum" on DNA Sum(%)	Time(s)	"sum" on KDDCUP08 Sum(%)	Time(s)
Perceptron	84.759 ± 0.575	0.006	54.018 ± 1.056	0.376
ROMMA	85.782 ± 0.553	0.006	54.342 ± 1.581	0.507
PA-I	87.832 ± 0.833	0.005	54.053 ± 0.865	0.414
PAUM	88.560 ± 0.737	0.005	55.161 ± 0.424	0.386
CPA_{PB}	89.401 ± 0.645	0.007	57.318 ± 0.629	0.458
AROW	89.183 ± 0.405	0.269	50.611 ± 0.422	12.554
COG-I	87.886 ± 0.812	0.006	54.094 ± 1.047	0.355
COG-II	87.395 ± 0.530	0.005	69.312 ± 0.475	0.359
ACOG-I	**91.490 ± 0.416**	0.104	55.088 ± 0.936	4.531
ACOG-II	90.872 ± 0.677	0.234	**71.920 ± 2.016**	5.803
ACOG-I_{diag}	89.498 ± 0.633	0.006	55.293 ± 0.852	0.384
ACOG-II_{diag}	88.433 ± 0.490	0.006	71.661 ± 1.334	0.397
SSACOG-I	89.975 ± 0.516	0.016	55.711 ± 0.812	0.810
SSACOG-II	90.444 ± 0.471	0.023	70.947 ± 1.179	0.842

6 CONCLUSION

In this paper, to remedy the weakness of first-order cost-sensitive online learning algorithms, we propose to introduce second-order information into cost-sensitive online classification framework based on adaptive regularization. As a result, a family of second-order cost-sensitive online classification algorithms is proposed, with favourable regret bound and impressive properties.

Moreover, to overcome the time-consuming problem of our second-order algorithms, we further study the sketching method in cost-sensitive online classification framework, and then propose sketched cost-sensitive online classification algorithms, which can be developed as a sparse cost-sensitive online learning approach, with better trade off between the performance and efficiency.

Then for examination of the performance and efficiency, we empirically evaluate our proposed algorithms on many public real-world datasets in extensive experiments. Promising results not only prove the new proposed algorithms successfully overcome the limitation of first-order algorithms, but also confirm their effectiveness and efficiency in solving real-world cost-sensitive online classification problems.

Future works include: (i) further exploration about the in-depth theory of cost-sensitive online learning; (ii) further study about the sparse computation methods in cost-sensitive online classification problems.

REFERENCES

[1] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 1958, Vol. 65, No. 6, pp. 386.

[2] P. Zhao, S. C. Hoi, R. Jin. Double updating online learning. *Journal of Machine Learning Research*, 2011, Vol. 12, pp. 1587-1615.

[3] J. Wang, P. Zhao, S. C. Hoi. Exact soft confidence-weighted learning. *International Conference on Machine Learning*, 2012, pp. 107-114.

[4] Q. Wu, H. Wu, X. Zhou, M. Tan, Y. Xu, Y. Yan, T. Hao. Online Transfer Learning with Multiple Homogeneous or Heterogeneous Sources. *IEEE Transactions on Knowledge and Data Engineering*, 2017, Vol. 29, No. 7, pp. 1494-1507.

[5] P. Zhao, S. C. Hoi. Cost-sensitive online active learning with application to malicious URL detection. *ACM International Conference on Knowledge Discovery and Data Mining*, 2013, pp. 919-927.

[6] J. Ma, L. K. Saul, S. Savage, G. M. Voelker. Learning to detect malicious urls. *ACM Transactions on Intelligent Systems and Technology*, 2011, Vol. 2, No. 3, pp. 30.

[7] B. Li, S. C. Hoi, P. Zhao, V. Gopalkrishnan. Confidence weighted mean reversion strategy for online portfolio selection. *ACM Transactions on Knowledge Discovery from Data*, 2013, Vol. 7, No. 1, pp.4.

[8] S. Shalev-Shwartz, Y. Singer, N. Srebro, A. Cotter, Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical Programming*, 2011, Vol. 127, No. 1, pp. 3-30.

[9] C. Elkan. The foundations of cost-sensitive learning. *International Joint Conference on Artificial Intelligence*, 2001, Vol. 17, pp. 973-978.

[10] K. Veropoulos, C. Campbell, N. Cristianini. Controlling the sensitivity of support vector machines. *International Joint Conference on Artificial Intelligence*, 1999, pp. 55-60.

[11] H. He, E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 2009. Vol. 21, pp. 1263-1284.

[12] J. Han, J. Pei, M. Kamber. Data mining: Concepts and Techniques. *Elsevier*, 2011.

[13] K. H. Brodersen, C. S. Ong, K. E. Stephan, J. M. Buhmann. The balanced accuracy and its posterior distribution. *International Conference on Pattern Recognition*, 2010, pp. 3121-3124.

[14] R. Akbani, S. Kwek, N. Japkowicz. Applying support vector machines to imbalanced datasets. *European Conference on Machine Learning*, 2004, pp. 39-50.

[15] J. Wang, P. Zhao and S. C. H. Hoi. Cost-sensitive online classification. *IEEE Iternational Conference on Data Mining*, 2012, 1140-1145.

[16] J. Wang, P. Zhao and S. C. H. Hoi. Cost-sensitive online classification. *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 10, pp. 2425-2438.

[17] M.Dredze, K.Crammer, F.Pereira.Confidence-weighted linear classification.*International Conference on Machine learning*, 2008, 264-271.

[18] K. Crammer, M. Dredze, F. Pereira. Exact convex confidence-weighted learning. *In Advances in Neural Information Processing Systems*, 2009, pp. 345-352.

[19] K. Crammer, A. Kulesza, M. Dredze. Adaptive regularization of weight vectors. *In Advances in Neural Information Processing Systems*, 2009, pp. 414-422.

[20] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. *International Conference on Machine Learning*, 2003, pp. 928-936.

[21] H. Luo, A. Agarwal, N Cesa-Bianchi. Efficient second order online learning by sketching. *In Advances in Neural Information Processing Systems*, 2016, pp. 902-910.

[22] Woodruff, P. David. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 2014, Vol. 10, No. 1-2, pp. 1-157.

[23] G. Krummenacher, B. McWilliams, Y. Kilcher, J. M. Buhmann, N. Meinshausen. Scalable adaptive stochastic optimization using random projections. *In Advances in Neural Information Processing Systems*, 2016, pp. 1750-1758.

[24] D. Wang, P. Wu, P. Zhao, Y. Wu, C. Miao, S. C. Hoi. High-dimensional data stream classification via sparse online learning. *IEEE International Conference on Data Mining*, 2014, pp. 1007-1012.

[25] P. Zhao, F. Zhuang, M. Wu, X. Li, and S. C. H. Hoi. Cost-sensitive online classification with adaptive regularization and its applications. *IEEE International Conference on Data Mining*, 2015, pp. 649-658.

[26] Z. H. Zhou, X. Y. Liu. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, 2006, Vol. 18, No. 1, pp. 63-77.

[27] R. Horn. Matrix analysis. *Cambridge University Express*, 1985.

[28] P. Zhao, S. C. Hoi. Cost-sensitive double updating online learning and its application to online anomaly detection. *SIAM International Conference on Data Mining*, 2013, pp. 207-215.

[29] D. Sahoo, S. C. Hoi, P. Zhao. Cost Sensitive Online Multiple Kernel Classification. *Asian Conference on Machine Learning*, 2016, pp. 65-80.

[30] P. Zhao, S. C. Hoi. OTL: A framework of online transfer learning. *International Conference on Machine Learning*, 2010, pp. 1231-1238.

[31] P. Zhao, R. Jin, T. Yang, S. C. Hoi. Online AUC maximization. *International Conference on Machine Learning*, 2011, pp. 233-240.

[32] P. Zhang, C. Zhou, P. Wang, B. J. Gao, X. Zhu, L. Guo. E-tree: An efficient indexing structure for ensemble models on data streams. *IEEE Transactions on Knowledge and Data Engineering*, 2015, Vol. 27, No. 2, pp. 461-474.

[33] Q. Zhang, P.Zhang, G. Long, W. Ding, C. Zhang, X. Wu. Online learning from trapezoidal data streams. *IEEE Transactions on Knowledge and Data Engineering*, 2016, Vol. 28, No. 10, pp. 2709-2723.

[34] Y. Yan, Q. Wu, M. Tan, M. K. Ng, H. Min, I. W. Tsang. Online Heterogeneous Transfer by Hedge Ensemble of Offline and Online Decisions. *IEEE Transactions on Neural Networks and Learning Systems*, 2017.

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

[36] Y. Zhang, G. Shu, Y. Li. Strategy-updating depending on local environment enhances cooperation in prisoners dilemma game. *Applied Mathematics and Computation*, 2017, Vol. 301, pp. 224-232.

[37] Y. Freund, R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine learning*, 1999, Vol. 37, pp. 277-296.

[38] Y. Li, P. M. Long. The relaxed online maximum margin algorithm. *In Advances in Neural Information Processing Systems*, 2000, 498-504.

[39] Y. Li, H. Zaragoza, R. Herbrich, J. Shawe-Taylor, J. Kandola. The perceptron algorithm with uneven margins. *International Conference on Machine learning*, 2002, pp. 379-386.

[40] E. Oja. Simplified neuron model as a principal component analyzer. Journal of Mathematical biology, 1982, Vol. 15, No. 3, pp. 267-273.

[41] E. Oja, J. Karhunen. On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Applications*, 1985, vol. 106, pp. 69-84.

[42] J. Abernethy, P. L. Bartlett, A. Rakhlin, A. Tewari. Optimal strategies and minimax lower bounds for online convex games. *Annual Conference on Computational Learning Theory*, 2008.

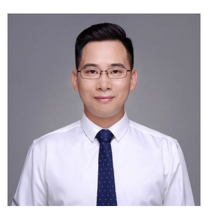
[43] M. Hardt, E. Price, The noisy power method: A meta algorithm with application. *In Advances in Neural Information Processing Systems*, 2014, pp. 2861-2869.

Peilin Zhao is currently a researcher with SCUT, China. His research interests are Large-Scale Machine Learning and its applications to Big Data Analytics. Previously, he has worked at Ant Financial, Institute for Infocomm Research (I2R), Rutgers University. He received his PHD degree from Nanyang Technological University and his bachelor degree from Zhejiang University. In his research areas, he has published over 70 papers in top venues, including JMLR, AIJ, ICML, NIPS, KDD, etc. He has been invited as a PC member or reviewer for many international conferences and journals in his area.

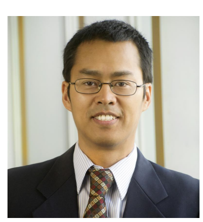
Yifan Zhang received the BE degree in electronic commerce from the Southwest University, China, in 2017. He is working toward the ME degree in the School of Software Engineering, South China University of Technology, China. His research interests include Machine Learning, Reinforcement Learning and their applications in Big Data Analytics.

Min Wu is a research scientist in Data Analytics Department, Institute for Infocomm Research. He received Ph.D degree from Nanyang Technological University, Singapore, in 2011 under the supervision of Prof. Kwoh Chee-Keong and Dr. Li Xiaoli, and received B.S. degree in Computer Science from University of Science and Technology of China, 2006. His research Interests include Graph Mining from Large-Scale Networks, Learning from Heterogeneous Data Sources, Ensemble Learning, and Bioinformatics.

Steven C. H. Hoi is an Associate Professor in Singapore Management University (SMU), Singapore. Prior to joining SMU, he was a tenured Associate Professor at Nanyang Technological University (NTU), Singapore. He received his Bachelor degree from Tsinghua University, and his Master and Ph.D degrees from the Chinese University of Hong Kong. His research interests include large-scale machine learning with application to a wide range of real-world applications. He has published over 150 papers in premier conferences and journals, and served as an organizer, area chair, senior PC, TPC member, editors, and referee for many top conferences and premier journals. He is the recipient of the Lee Kong Chian Fellowship Award due to his research excellence.



Mingkui Tan received the PhD degree in computer science from Nanyang Technological University, Singapore, in 2014. He is a professor in the School of Software Engineering, South China University of Technology. After that, he worked as a senior research associate in the School of Computer Science, University of Adelaide, Australia. His research interests include compressive sensing, big data learning, and large-scale optimization.



Junzhou Huang is an Associate Professor in the Computer Science and Engineering department at the University of Texas at Arlington. He received the B.E. degree from Huazhong University of Science and Technology, China, the M.S. degree from Chinese Academy of Sciences, China, and the Ph.D. degree in Rutgers university. His major research interests include machine learning, computer vision and imaging informatics. He was selected as one of the 10 emerging leaders in multimedia and signal processing by the IBM T.J. Watson Research Center in 2010. He received the NSF CAREER Award in 2016.