

# Singapore Management University Institutional Knowledge at Singapore Management University

Research Collection School Of Information Systems

School of Information Systems

2-2019

## Adaptive cost-sensitive online classification

Peilin ZHAO

*South China University of Technology*

Yifan ZHANG

*South China University of Technology*

Min WU

*Institute for InfoComm Research*

Steven C. H. HOI

*Singapore Management University, CHHOI@smu.edu.sg*


Mingkui TAN

*South China University of Technology*

*See next page for additional authors*

**DOI:** <https://doi.org/10.1109/TKDE.2018.2826011>

Follow this and additional works at: [https://ink.library.smu.edu.sg/sis\\_research](https://ink.library.smu.edu.sg/sis_research)

 Part of the [Databases and Information Systems Commons](#), and the [Numerical Analysis and Scientific Computing Commons](#)

### Citation

ZHAO, Peilin; ZHANG, Yifan; WU, Min; HOI, Steven C. H.; TAN, Mingkui; and HUANG, Junzhou. Adaptive cost-sensitive online classification. (2019). *IEEE Transactions on Knowledge and Data Engineering*. 31, (2), 214-228. Research Collection School Of Information Systems.

**Available at:** [https://ink.library.smu.edu.sg/sis\\_research/4036](https://ink.library.smu.edu.sg/sis_research/4036)

This Journal Article is brought to you for free and open access by the School of Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email [libIR@smu.edu.sg](mailto:libIR@smu.edu.sg).

---

**Author**

Peilin ZHAO, Yifan ZHANG, Min WU, Steven C. H. HOI, Mingkui TAN, and Junzhou HUANG

# Adaptive Cost-Sensitive Online Classification

Peilin Zhao<sup>1</sup>, Yifan Zhang<sup>1</sup>, Min Wu<sup>2</sup>, Steven C. H. Hoi<sup>3</sup>, Mingkui Tan<sup>4</sup>, 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 and (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], [6]. 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 [7], [29] and portfolio selection [8]. In addition, online learning is also good at solving large-scale learning tasks, e.g., learning support vector machine from billions of data [9].

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 [10], [11], [12]. To solve this problem, researchers have suggested to use more meaningful metrics, such as the weighted sum of *sensitivity* and *specificity* [13], [14], and the weighted *misclassification cost* [10], [15] 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 [10], [15]. However, these batch algorithms often suffer from poor scalability and efficiency for large-scale tasks, which make 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 [16], [17] 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 takes the first order information of samples (i.e., weighted mean of the gradient). It is obviously insufficient, since many recent studies [3], [18], [19], [20] 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

- 
- P. Zhao, Y. Zhang and M. Tan are with the South China University of Technology, Guangzhou, Guangdong 510630, China. E-mail: peilinzhao@hotmail.com, sezyifan@mail.scut.edu.cn, mingkuitan@scut.edu.cn.
  - M. Wu is with the Institute for Infocomm Research, Singapore 138632. E-mail: wumin@i2r.a-star.edu.sg.
  - S. C. Hoi is with Singapore Management University, Singapore 188065. E-mail: chhoi@smu.edu.sg.
  - J. Huang is with Tencent AI Lab, Shenzhen, Guangdong, China. E-mail: joehhuang@tencent.com.

Manuscript received 7 Dec. 2017; revised 27 Mar. 2018; accepted 30 Mar. 2018. Date of publication 12 Apr. 2018; date of current version 9 Jan. 2019. (Corresponding author: Mingkui Tan.)

Recommended for acceptance by L. B. Holder.

For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.

Digital Object Identifier no. 10.1109/TKDE.2018.2826011

Cost-Sensitive Online Gradient Descent algorithms (named ACOG), based on the state-of-the-art Confidence Weighted strategy [3], [18], [19], [20]. We theoretically analyze their regret bounds [21] 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 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 [22], [23], [24], [25], 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 specialties 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 [26]. 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 Appendixes.<sup>1</sup> 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 first introduce the framework and formulation setting of the Cost-Sensitive Online Classification problem [16], [17]. 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  $\{(x_1, y_1), (x_2, y_2), \dots, (x_T, y_T)\}$ , 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  $\hat{y}_t = \text{sign}(w_t^\top x_t)$ ,

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  $\mathcal{M} = \{t \mid y_t \neq \text{sign}(w_t \cdot x_t), \forall t \in [T]\}$  is the mistake index set,  $\mathcal{M}_p = \{t \in \mathcal{M} \text{ and } y_t = +1\}$  is the positive set of mistake index and  $\mathcal{M}_n = \{t \in \mathcal{M} \text{ and } y_t = -1\}$  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  $\mathcal{I}_T^p = \{i \in [T] \mid y_i = +1\}$  and  $\mathcal{I}_T^n = \{i \in [T] \mid y_i = -1\}$ , 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):

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

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*:

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

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:

$$\text{cost} = c_p \times M_p + c_n \times M_n,$$

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 [16], [17], both objectives are equivalent to minimizing the following objective:

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

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 section, 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

1. <https://arxiv.org/pdf/1804.02246.pdf>

variants (either one of the following two functions):

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

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

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 [21], based on either loss functions  $\ell^I(w; (x, y))$  or  $\ell^{II}(w; (x, y))$ :

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

where  $w^* = \arg \min_w \sum_{t=1}^T \nabla \ell(w; (x_t, y_t))$ . To solve this optimization problem, the cost-sensitive online gradient descent algorithms (COG) [16], [17] were proposed:

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

where  $\eta$  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], [18], [19], [20]. 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  $\mu$  is the mean value vector of distribution and  $\Sigma$  is the covariance matrix of distribution. Then, we can predict the class label of an sample  $x$  based on  $\text{sign}(w^\top x)$ , 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  $\text{sign}(\mu^\top x)$  in the following. For better understanding, each mean value  $\mu_i$  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  $\mu_i$  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  $\mu_i$  and  $\mu_j$  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,  $\eta$  is fitting parameter and  $\gamma$  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:

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

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  $\mu$  and  $\Sigma$ , respectively. Then, the updates of mean vector  $\mu$  and covariance matrix  $\Sigma$  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 [28] 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  $\mu$  relies on the confidence parameter  $\Sigma$ , we thus propose to update  $\mu$  based on the updated covariance matrix  $\Sigma_{t+1}$  instead of the old one  $\Sigma_t$ , which should be more accurate:

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

This is different from AROW [20], where the updating rule of  $\mu_t$  based on the old matrix  $\Sigma_t$ . To intuitively understand this change, let us assume  $\Sigma_{t+1}$  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  $\mu$ , 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  $\eta$ ; regularized parameter  $\gamma$ ; bias parameter

$\rho = \frac{\alpha_p T_n}{\alpha_n T_p}$  for “sum” and  $\rho = \frac{c_p}{c_n}$  for “cost”.

**Initialization**  $\mu_1 = 0, \Sigma_1 = I$ .

```

1: for  $t = 1 \rightarrow T$  do
2:   Receive sample  $x_t$ ;
3:   Compute  $\ell_t(\mu_t) = \ell^*(\mu_t; (x_t, y_t))$ , where  $*$   $\in \{I, II\}$ ;
4:   if  $\ell_t(\mu_t) > 0$  then
5:      $\Sigma_{t+1} = \Sigma_t - \frac{\Sigma_t x_t x_t^\top \Sigma_t}{\gamma + x_t^\top \Sigma_t x_t}$ ;
6:      $\mu_{t+1} = \mu_t - \eta \Sigma_{t+1} g_t$ , where  $g_t = \partial_\mu \ell_t(\mu_t)$ ;
7:   else
8:      $\mu_{t+1} = \mu_t, \Sigma_{t+1} = \Sigma_t$ ;
9:   end if
10: end for

```

---

For simplification, we ignore the sample numbers  $T$  in the analyses of algorithms efficiency. Thus the time complexity for the updates of  $\Sigma_{t+1}$  and  $\mu$  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<sub>diag</sub>), which accelerates the speed of ACOG algorithms to  $\mathcal{O}(d)$ . Specifically, only a diagonal version  $\Sigma_t$  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  $\rho$ , when optimizing the weighted *sum* performance. Normally,  $\rho$  is denoted as  $\rho = \frac{\alpha_p T_n}{\alpha_n T_p}$  for *sum* metric. However, the value of  $T_p$  and  $T_n$  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 section, 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  $(x_1, y_1), (x_2, y_2), \dots, (x_T, y_T)$  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  $\eta = \sqrt{\frac{\max_{t \leq T} \|\mu_t - \mu\|^2 \text{Tr}(\Sigma_{T+1}^{-1})}{\gamma \log(|\Sigma_{T+1}^{-1}|)}}$ , the proposed ACOG-I satisfies:

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

where  $D_\mu = \max_t \|\mu_t - \mu\|$ . In addition, by setting

$\eta = \sqrt{\frac{\max_{t \leq T} \|\mu_t - \mu\|^2 \text{Tr}(\Sigma_{T+1}^{-1})}{\rho^2 \gamma \log(|\Sigma_{T+1}^{-1}|)}}$ , ACOG-II satisfies:

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

**Remark.** Let us suppose  $\|x_t\| \leq 1$ , it is easy to discover  $\text{Tr}(\Sigma_{T+1}^{-1}) \leq \mathcal{O}(T/\gamma)$ , which means the regrets of ACOG are in the order of  $\mathcal{O}(\sqrt{T})$ . This order of regret is the optimal, since the loss function is not strongly convex [43].

**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:

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

and the ACOG-II satisfies:

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

**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  $\mu$ , 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  $\alpha_n$  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:

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

and the ACOG-II satisfies:

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

**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  $\mathcal{O}(d^2)$  and its diagonal version is  $\mathcal{O}(d)$ . 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 dependence on the data dimensionality [24]. 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 section, we will present the enhanced version of ACOG via *Oja's sketch method* [22], [41], [42], 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  $\Sigma$  by a small number of carefully selective directions, called as a *sketch*.

According to Eqs. (6) and (7), we know the updating rule of model parameter  $\mu$ :

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

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:

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

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-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[28], we have:

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

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  $\mu$ :

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

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

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 [41], [42], the updating rules of  $\Lambda_t$  and  $V_t$  are defined as:

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

$$V_t \stackrel{\text{orth}}{\leftarrow} V_{t-1} + \Gamma_t V_{t-1} \hat{x}_t \hat{x}_t^\top, \quad (12)$$

where learning rate  $\Gamma_t = \frac{1}{t} I_m \in \mathbb{R}^{m \times m}$  is a diagonal matrix, and  $\stackrel{\text{orth}}{\leftarrow}$  represents an orthonormalizing step.<sup>2</sup> Then, the sketch matrices can be obtained by:

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

$$H_t = \text{diag}\left\{\frac{1}{1+t\Lambda_{1,1}}, \dots, \frac{1}{1+t\Lambda_{m,m}}\right\}.$$

---

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

---

**Input** learning rate  $\eta$ ; regularized parameter  $\gamma$ ; sketch size  $m$ ; bias  $\rho = \frac{\alpha_n * I_n}{\alpha_n * I_p}$  for "sum" and  $\rho = \frac{c_p}{c_n}$  for "cost".

**Initialization**  $\mu_1 = 0$ , sketch( $S_0, H_0$ )  $\leftarrow$  **SketchInit**( $m$ ).

- 1: **for**  $t = 1 \rightarrow T$  **do**
  - 2:   Receive sample  $x_t$ ;
  - 3:   Compute  $\ell_t(\mu_t) = \ell^*(\mu_t; (x_t, y_t))$ , where  $*$   $\in \{I, II\}$ ;
  - 4:   Compute the  $t$ -sketch vector  $\hat{x}_t = \frac{x_t}{\sqrt{\gamma}}$ ;
  - 5:   ( $S_t, H_t$ )  $\leftarrow$  **SketchUpdate**( $\hat{x}$ );
  - 6:   **if**  $\ell_t(\mu_t) > 0$  **then**
  - 7:      $\mu_{t+1} = \mu_t - \eta (g_t - S_t^\top H_t S_t g_t)$ , where  $g_t = \partial_\mu \ell_t(\mu_t)$ ;
  - 8:   **else**
  - 9:      $\mu_{t+1} = \mu_t$ .
  - 10:   **end if**
  - 11: **end for**
- 

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.

---

**Algorithm 3.** Oja's Sketch for SACOG

---

**Input**  $m, \hat{x}$  and stepsize matrix  $\Gamma_t$ .

**Internal State**  $t, \Lambda, V$  and  $H$ .

**SketchInit**( $m$ )

- 1: 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;
- 2: Return ( $S, H$ ).

**SketchUpdate**( $\hat{x}$ )

- 1: Update  $t \leftarrow t + 1$ ;
  - 2: Update  $\Lambda = (I_m - \Gamma_t) \Lambda + \Gamma_t \text{diag}\{V \hat{x}\}^2$ ;
  - 3: Update  $V \stackrel{\text{orth}}{\leftarrow} V + \Gamma_t V \hat{x} \hat{x}^\top$ ;
  - 4: Set  $S = (t\Lambda)^{\frac{1}{2}} V$ ;
  - 5: Set  $H = \text{diag}\{\frac{1}{1+t\Lambda_{1,1}}, \dots, \frac{1}{1+t\Lambda_{m,m}}\}$ ;
  - 6: 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  $O(md)$  [44]. Another concern is the regret guarantee, which is not clear now because existing analysis for Oja's algorithm is only for the stochastic situation [22]. 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

<sup>2</sup> 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  $\stackrel{\text{orth}}{\leftarrow}$  operation always keeps the same dimensionality of  $V_t$ .

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  $\mu_t$  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  $\mu_t$ , 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  $\mu_t$  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 from  $V_{t-1}$  (same as  $Z_{t-1}$ ). Then, we describe how to sparsely update two weight parts  $w_t$  and  $b_t$ . First, 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  $w_t$ :

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

where  $Z_t = Z_{t-1} + \delta_t \hat{x}_t^\top$ , and define the updating rule of  $b_t$ :

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

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

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

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

Then from Eq. (12), we have:

$$\begin{aligned} F_t Z_t &\stackrel{\text{eq. 11}}{=} F_{t-1} Z_{t-1} + \Gamma_t F_{t-1} Z_{t-1} \hat{x}_t \hat{x}_t^\top, \\ &= F_{t-1} (Z_{t-1} + F_{t-1}^{-1} \Gamma_t F_{t-1} Z_{t-1} \hat{x}_t \hat{x}_t^\top). \end{aligned} \quad (17)$$

Here,  $Z_t = Z_{t-1} + \delta_t \hat{x}_t^\top$ , 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  $Z_t$ :

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

---

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

---

**Input** learning rate  $\eta$ ; regularized parameter  $\gamma$ ; sketch size  $m$ ; bias  $\rho = \frac{\alpha_p * \Gamma_n}{\alpha_n * \Gamma_p}$  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**  $(\Lambda_0, F_0, Z_0, H_0) \leftarrow \text{SketchInit}(m)$ ;

- 1: **for**  $t = 1 \rightarrow T$  **do**
  - 2:   Receive sample  $x_t$ ;
  - 3:   Compute  $\ell_t(\mu_t) = \ell^*(\mu_t; (x_t, y_t))$ , where  $*$   $\in \{I, II\}$ ;
  - 4:   Compute the  $t$ -sketch vector  $\hat{x}_t = \frac{x_t}{\sqrt{p}}$ ;
  - 5:    $(\Lambda_t, F_t, Z_t, H_t, \delta_t) \leftarrow \text{SketchUpdate}(\hat{x}_t)$ ;
  - 6:   **if**  $\ell_t(\mu_t) > 0$  **then**
  - 7:      $w_{t+1} = w_t - \eta g_t - \hat{x}_t \delta_t^\top b_t$ ;
  - 8:      $b_{t+1} = b_t + \eta F_t^\top (t\Lambda_t H_t) F_t Z_t g_t$ ;
  - 9:      $\mu_{t+1} = w_{t+1} + Z_t^\top b_{t+1}$ ;
  - 10:   **else**
  - 11:      $\mu_{t+1} = \mu_t$ ,  $w_{t+1} = w_t$ ,  $b_{t+1} = b_t$ .
  - 12:   **end if**
  - 13: **end for**
- 

We summarize the Sparse Oja’s algorithm for SACOG in Algorithm 5.

---

#### Algorithm 5. Sparse Oja’s Sketch for SACOG

---

**Input**  $m$ ,  $\hat{x}$  and stepsize matrix  $\Gamma_t$ .

**Internal State**  $t, \Lambda, F, Z, K$  and  $H$ .

**SketchInit**( $m$ )

- 1: Set  $t = 0$ ,  $F = K = H = I_m$ ,  $\Lambda = 0_{m \times m}$  and  $Z$  to any  $m \times d$  matrix with orthonormal rows;
- 2: Return  $(\Lambda, F, Z, H)$ .

**SketchUpdate**( $\hat{x}$ )

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

**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 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.



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     |

---

#### Algorithm 6. Decompose( $F, K$ )

---

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

---

## 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], [38]; (2) Relaxed Online Maximum Margin Algorithm [39] (“ROMMA”); (3) Passive-Aggressive algorithm [36] (“PA-I” and “CPA-PB”); (4) Perceptron Algorithm with Uneven Margin [40] (“PAUM”); (5) Adaptive Regularization of Weight Vector [20] (“AROW”); (6) Cost-Sensitive Online Gradient Descent [16], [17] (“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.<sup>3</sup>

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  $\rho$  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<sub>PB</sub> algorithm,  $\rho(-1, 1)$  was set to 1, and  $\rho(1, -1)$  was  $\rho$ . For PAUM, the uneven margin was set to  $\rho$ . In addition, the parameter of  $C$  for PA-I, learning rate  $\lambda$  for COG and learning rate  $\eta$  for all our proposed algorithms were selected from  $[10^{-5}, 10^{-4}, \dots, 10^5]$ . The regularized parameter  $\gamma$  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 Windows machine.

## 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 summarizes 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<sub>diag</sub> 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 section, we would like to evaluate the *sum* of proposed methods under different cost-sensitive weights. Fig. 2 shows the empirical results under different weights of  $\alpha_n$  and  $\alpha_p$ . 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 summarizes the experimental performance of the ACOG<sub>cost</sub> 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 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<sub>PB</sub>). This implies that introducing the second order information is beneficial to the decrease of misclassification *cost*.

In addition, by examining both *sensitivity* and *specificity* metrics, we observe that our proposed methods

3. <https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/>

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 <sup>2</sup> ) | 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 <sub>PB</sub>       | 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        | <b>92.849 ± 0.153</b> | 5.591   | 45.931 ± 0.486         | 58.858 ± 0.510        | <b>92.849 ± 0.153</b> | 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                 | <b>81.220 ± 0.108</b> | <b>85.269 ± 0.219</b> | 77.171 ± 0.134        | 3.344   | <b>20.307 ± 0.169</b>  | <b>94.079 ± 0.136</b> | 62.107 ± 0.185        | 3.612   |
| ACOG-I <sub>diag</sub>  | 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 <sub>diag</sub> | <b>81.098 ± 0.083</b> | <b>84.705 ± 0.227</b> | 77.491 ± 0.152        | 0.216   | <b>20.661 ± 0.110</b>  | <b>93.352 ± 0.126</b> | 63.212 ± 0.238        | 0.213   |

| Algorithm               | "sum" on covtype      |                       |                       |         | "cost" on covtype      |                       |                       |         |
|-------------------------|-----------------------|-----------------------|-----------------------|---------|------------------------|-----------------------|-----------------------|---------|
|                         | Sum(%)                | Sensitivity(%)        | Specificity (%)       | Time(s) | Cost(10 <sup>2</sup> ) | 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 <sub>PB</sub>       | 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        | <b>65.914 ± 0.213</b> | 22.640  | 687.696 ± 3.148        | 76.580 ± 0.137        | <b>69.579 ± 0.134</b> | 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                  | <b>68.077 ± 0.038</b> | <b>70.565 ± 0.073</b> | <b>65.588 ± 0.082</b> | 13.782  | 466.376 ± 1.190        | 90.693 ± 0.049        | 23.054 ± 0.038        | 18.988  |
| ACOG-II                 | <b>68.020 ± 0.030</b> | <b>71.265 ± 0.070</b> | 64.774 ± 0.068        | 13.528  | <b>305.056 ± 0.355</b> | <b>98.969 ± 0.021</b> | 6.365 ± 0.163         | 13.232  |
| ACOG-I <sub>diag</sub>  | 67.247 ± 0.060        | 69.183 ± 0.076        | <b>65.311 ± 0.082</b> | 1.824   | 469.701 ± 1.377        | 90.594 ± 0.090        | 22.782 ± 0.370        | 1.971   |
| ACOG-II <sub>diag</sub> | 67.225 ± 0.062        | 69.913 ± 0.096        | 64.537 ± 0.086        | 1.805   | <b>308.987 ± 7.944</b> | <b>98.739 ± 0.367</b> | 7.015 ± 0.507         | 1.828   |

| Algorithm               | "sum" on german       |                       |                       |         | "cost" on german       |                       |                       |         |
|-------------------------|-----------------------|-----------------------|-----------------------|---------|------------------------|-----------------------|-----------------------|---------|
|                         | Sum(%)                | Sensitivity(%)        | Specificity (%)       | Time(s) | Cost(10 <sup>2</sup> ) | 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 <sub>PB</sub>       | 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        | <b>93.529 ± 1.630</b> | 0.014   | 1.610 ± 0.082          | 43.867 ± 3.364        | <b>86.571 ± 1.543</b> | 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                  | <b>63.150 ± 1.025</b> | 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        | <b>63.000 ± 2.052</b> | 62.021 ± 1.408        | 0.008   | <b>0.875 ± 0.044</b>   | <b>86.883 ± 2.264</b> | 25.564 ± 4.099        | 0.011   |
| ACOG-I <sub>diag</sub>  | 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 <sub>diag</sub> | 62.281 ± 1.428        | 62.883 ± 1.852        | 61.679 ± 1.576        | 0.003   | <b>0.912 ± 0.045</b>   | <b>84.733 ± 0.876</b> | 28.629 ± 4.046        | 0.003   |

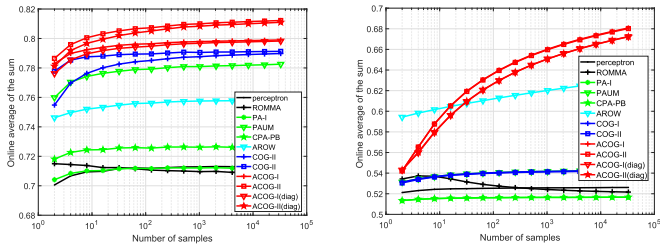
| Algorithm               | "sum" on icjnn1       |                       |                       |         | "cost" on icjnn1       |                       |                       |         |
|-------------------------|-----------------------|-----------------------|-----------------------|---------|------------------------|-----------------------|-----------------------|---------|
|                         | Sum(%)                | Sensitivity(%)        | Specificity (%)       | Time(s) | Cost(10 <sup>2</sup> ) | 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 <sub>PB</sub>       | 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        | <b>98.308 ± 0.074</b> | 0.450   | 28.626 ± 0.401         | 36.208 ± 0.980        | <b>98.308 ± 0.074</b> | 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                 | <b>86.872 ± 0.174</b> | <b>88.924 ± 0.323</b> | 84.820 ± 0.218        | 0.288   | <b>12.279 ± 0.149</b>  | <b>87.626 ± 0.293</b> | 84.770 ± 0.165        | 0.298   |
| ACOG-I <sub>diag</sub>  | 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 <sub>diag</sub> | <b>86.929 ± 0.124</b> | <b>88.205 ± 0.266</b> | 85.652 ± 0.107        | 0.120   | <b>12.016 ± 0.111</b>  | <b>87.164 ± 0.300</b> | 85.801 ± 0.138        | 0.122   |

often achieve the best *sensitivity* result on all datasets, and attain a relatively good *specificity* among all cases.

Moreover, the diagonal ACOG<sub>diag</sub> 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<sub>diag</sub> methods can be regarded as a choice to balance the performance and efficiency.

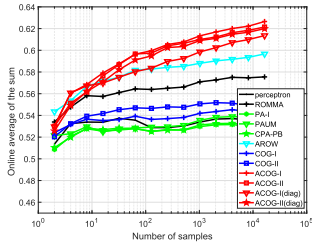
### 4.3.2 Evaluation of Cost under Varying Weights

In this section, 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

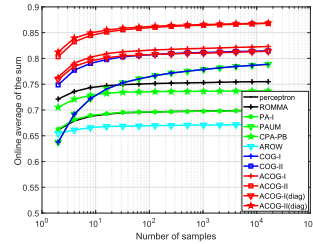


(a) a9a

(b) covtype

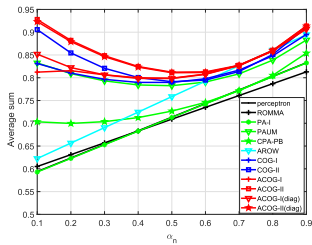


(c) german

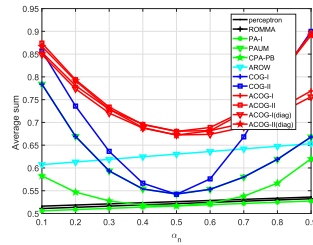


(d) ijcnn1

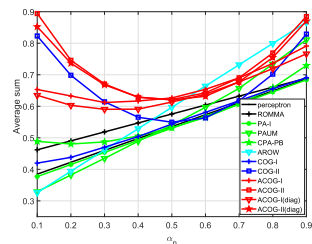
Fig. 1. Evaluation of online “sum” performance of the proposed algorithms on public datasets.



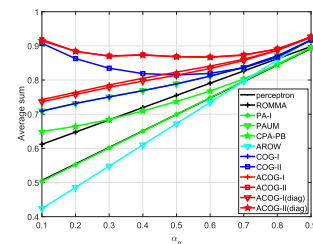
(a) a9a



(b) covtype



(c) german



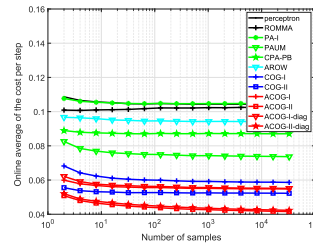
(d) ijcnn1

Fig. 2. Evaluation of weighted “sum” performance under varying weights of sensitivity and specificity.

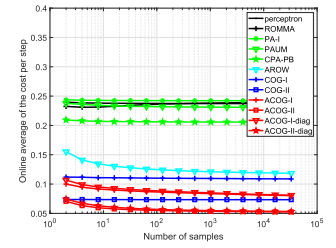
methods. These discoveries imply that our ACOG algorithms have a wide selection range of weight parameters for online classification tasks.

#### 4.4 Evaluation of Algorithm Properties

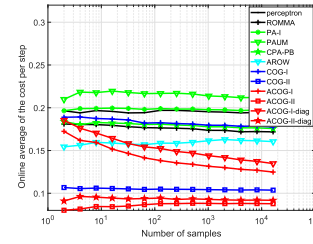
We have evaluated the performance of proposed algorithms in previous experiments, where promising results confirm their great superiority. Next, we are eager to examine their unique properties, including the influence of learning rate, regularized parameter, updating rule, online estimation and generalization 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, while all other variable settings are fixed and similar with before experiments.



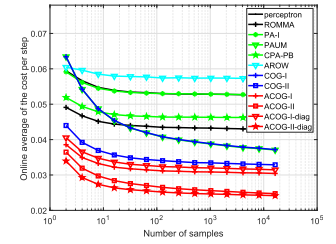
(a) a9a



(b) covtype

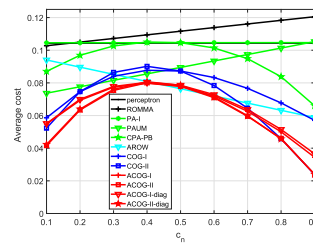


(c) german

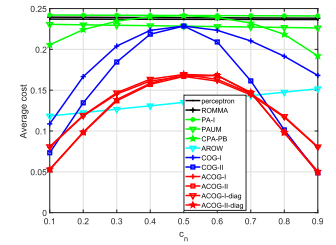


(d) ijcnn1

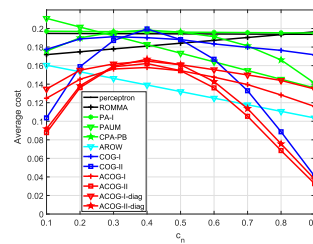
Fig. 3. Evaluation of online “cost” performance of the proposed algorithms on public datasets.



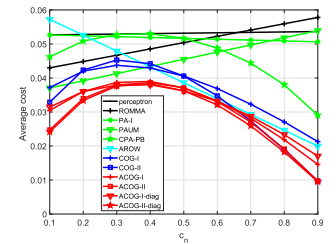
(a) a9a



(b) covtype



(c) german



(d) ijcnn1

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

##### 4.4.1 Evaluation of Learning Rate

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

In Fig. 5, we find that ACOG algorithms would achieve relatively higher result, when we choose proper learning rate (i.e., relatively higher  $\eta$  in general). This is easy to understand because the values of covariance matrix  $\Sigma$  are normally small. Specifically, when a misclassification happened at time  $t$ , we update the predictive vector  $\mu$  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  $\Sigma$  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.

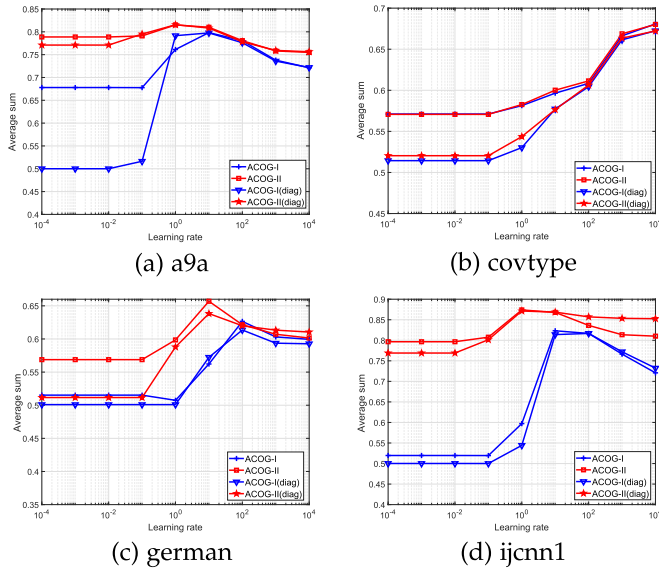


Fig. 5. Performance under varying learning rates.

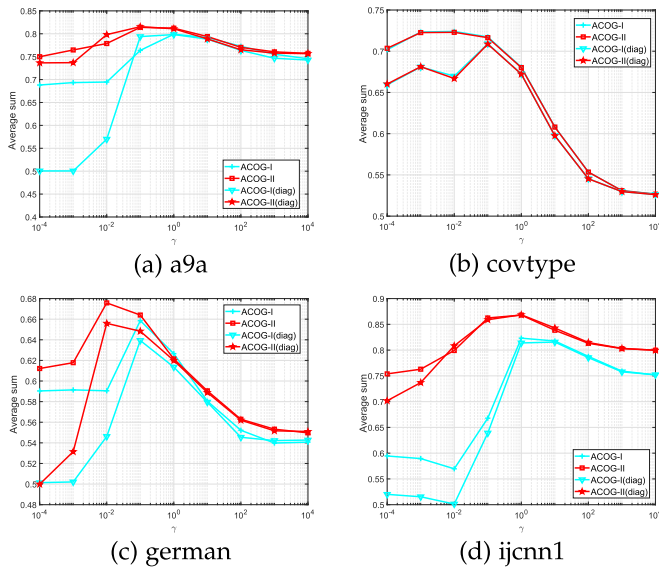


Fig. 6. Performance under different regularized parameters.

Moreover, we find the proposed methods with objective function  $\ell^I(w; (x, y))$  can achieve relatively higher performance than the methods with  $\ell^J(w; (x, y))$ , which means that ACOG-II and ACOG-II<sub>diag</sub> are more robust to different learning rate  $\eta$  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  $\Sigma$  by  $\Sigma_{t+1} = \Sigma_t - \frac{\Sigma_t x_t x_t^T \Sigma_t}{\gamma + x_t^T \Sigma_t x_t}$  with default regularized parameter  $\gamma$  as 1. However, the rationality of this setting is not verified. Thus, we examine the performance of our algorithms with different regularized parameters  $\gamma$  from  $[10^{-4}, 10^{-3}, \dots, 10^3, 10^4]$  for  $sum$  metrics.

The results in Fig. 6 show that the optimal parameter normally is different according to datasets; while in most cases,

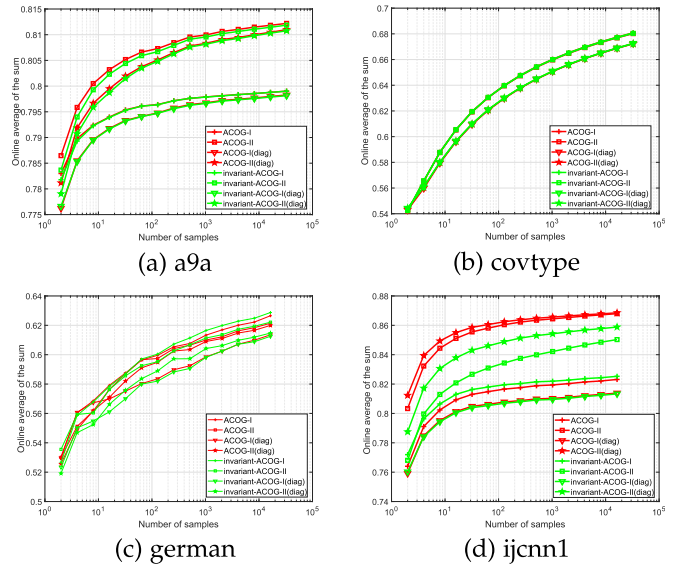


Fig. 7. Evaluation of updating rules.

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  $\mu$  is updated by  $\mu_{t+1} = \mu_t + \eta \Sigma_{t+1} g_t$ , which is different from AROW where the updating rule for  $\mu$  relies on the old  $\Sigma_t$ . In this section, 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  $\Sigma_t$ .

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

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

In the remark of Algorithm 1, we analyzed the parameter  $\rho = \frac{\eta_n T_n}{\eta_n T_p}$  for ACOG<sub>sum</sub> algorithms, where the main question is that the value of  $T_p$  and  $T_n$  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 that have been seen, 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<sub>sum</sub> algorithms.

#### 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

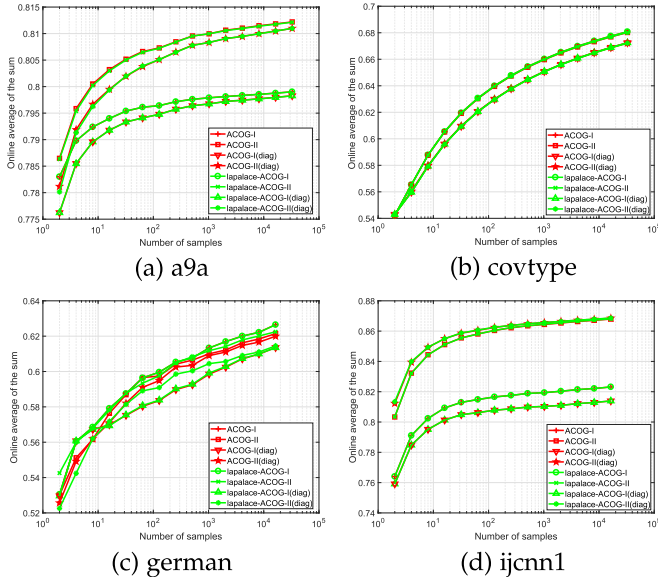


Fig. 8. Evaluation of online estimation of  $\frac{T_p}{T_p}$ .

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<sub>diag</sub>” and “ACOG-II<sub>diag</sub>”, 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 [22]. Moreover, the learning rate was selected from  $[10^{-5}, 10^{-4}, \dots, 10^5]$ , where other implementation details are similar with [22]. In addition, all experimental settings for other algorithms are same as previous experiments.

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 <sub>PB</sub>       | 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        | <b>72.428</b> | 62.954        | 82.926        |
| ACOG-II                 | <b>81.630</b> | <b>72.632</b> | 60.928        | <b>87.730</b> |
| ACOG-I <sub>diag</sub>  | 80.118        | 71.051        | <b>64.389</b> | 82.334        |
| ACOG-II <sub>diag</sub> | <b>81.752</b> | 71.311        | <b>66.036</b> | <b>87.628</b> |

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      |

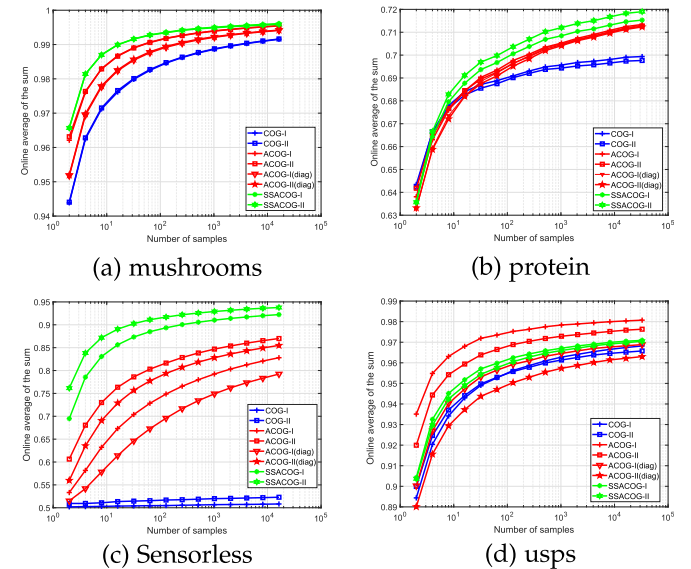


Fig. 9. Weighted “sum” performance of SACOG.

#### 4.5.1 Evaluation of Weighted Sum Performance

In this section, 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<sub>diag</sub>, it enjoys higher performance due

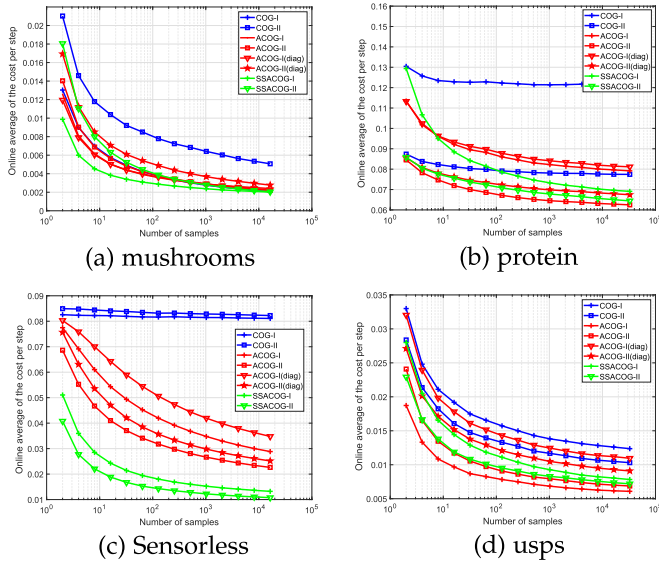


Fig. 10. Weighted “cost” performance of SACOG.

to the advantage of sufficient second-order information. This 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 SSACOG. 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.

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:

- 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<sup>4</sup>. The main goal is to detect the breast cancer from X-ray images, where “benign” is assigned as normal and “malignant” is abnormal.

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

### 5.2 Empirical Evaluation Results

In this section, 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<sub>PB</sub>) 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<sub>diag</sub>, 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<sub>diag</sub> as a choice.

4. <http://www.sigkdd.org/kddcup/>

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        | <b>15.760 ± 2.496</b> | 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  | <b>0.043</b> | 18.735 ± 1.062        | 99.939 ± 0.051 | 95.802 ± 0.443  | <b>0.085</b> |
| ACOG-II                 | 99.572 ± 0.033        | 99.794 ± 0.080 | 99.349 ± 0.075  | <b>0.045</b> | 16.770 ± 1.546        | 99.932 ± 0.035 | 96.373 ± 0.412  | <b>0.054</b> |
| ACOG-I <sub>diag</sub>  | 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 <sub>diag</sub> | 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                | <b>99.628 ± 0.052</b> | 99.798 ± 0.066 | 99.459 ± 0.102  | <b>0.038</b> | <b>15.880 ± 1.677</b> | 99.930 ± 0.043 | 96.623 ± 0.303  | <b>0.041</b> |
| SSACOG-II               | <b>99.606 ± 0.050</b> | 99.805 ± 0.062 | 99.408 ± 0.093  | <b>0.038</b> | <b>15.560 ± 3.870</b> | 99.869 ± 0.040 | 97.291 ± 1.063  | <b>0.034</b> |

| 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  | <b>14.603</b> | 1406.660 ± 28.314        | 87.072 ± 0.428 | 52.671 ± 0.576  | <b>16.922</b> |
| ACOG-II                 | 71.265 ± 0.235        | 71.678 ± 0.398 | 70.852 ± 0.501  | <b>14.446</b> | <b>1110.075 ± 12.274</b> | 94.972 ± 0.172 | 22.753 ± 0.853  | <b>13.601</b> |
| ACOG-I <sub>diag</sub>  | 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 <sub>diag</sub> | 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                | <b>71.532 ± 0.198</b> | 66.861 ± 0.530 | 76.203 ± 0.485  | <b>0.355</b>  | 1227.345 ± 16.904        | 90.608 ± 0.225 | 44.148 ± 0.342  | <b>0.393</b>  |
| SSACOG-II               | <b>71.323 ± 0.132</b> | 71.725 ± 0.305 | 72.075 ± 0.403  | <b>0.352</b>  | 1144.680 ± 13.087        | 94.053 ± 0.144 | 26.224 ± 0.661  | <b>0.348</b>  |

| 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  | <b>0.503</b> | 1622.600 ± 32.312       | 72.563 ± 0.709 | 94.188 ± 0.078  | <b>0.480</b> |
| ACOG-II                 | 87.398 ± 0.186        | 88.088 ± 0.284 | 86.708 ± 0.178  | <b>0.486</b> | 1283.350 ± 16.768       | 87.247 ± 0.264 | 87.350 ± 0.131  | <b>0.455</b> |
| ACOG-I <sub>diag</sub>  | 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 <sub>diag</sub> | 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                | <b>92.432 ± 0.213</b> | 89.818 ± 0.442 | 95.047 ± 0.047  | <b>0.322</b> | <b>753.695 ± 21.185</b> | 89.482 ± 0.476 | 95.296 ± 0.066  | <b>0.285</b> |
| SSACOG-II               | <b>93.913 ± 0.129</b> | 94.487 ± 0.181 | 93.339 ± 0.123  | <b>0.296</b> | <b>615.625 ± 12.280</b> | 94.166 ± 0.194 | 93.676 ± 0.096  | <b>0.264</b> |

| 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                  | <b>98.073 ± 0.115</b> | 97.822 ± 0.242 | 98.323 ± 0.095  | <b>0.271</b> | <b>44.365 ± 3.448</b> | 96.671 ± 0.321 | 98.591 ± 0.070  | <b>0.151</b> |
| ACOG-II                 | <b>97.633 ± 0.148</b> | 97.998 ± 0.230 | 97.268 ± 0.176  | <b>0.239</b> | <b>50.100 ± 4.321</b> | 98.241 ± 0.172 | 94.883 ± 0.488  | <b>0.252</b> |
| ACOG-I <sub>diag</sub>  | 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 <sub>diag</sub> | 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  | <b>0.077</b> | 57.055 ± 4.251        | 95.657 ± 0.384 | 98.296 ± 0.076  | <b>0.054</b> |
| SSACOG-II               | 97.048 ± 0.163        | 97.010 ± 0.237 | 97.085 ± 0.190  | <b>0.074</b> | 52.420 ± 4.009        | 97.647 ± 0.192 | 95.550 ± 0.360  | <b>0.054</b> |

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 <sup>2</sup> ) | Time(s) | Sum(%)           | Time(s) | Cost(10 <sup>2</sup> ) | 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 <sup>2</sup> ) | Time(s) | Sum(%)         | Time(s) | Cost(10 <sup>2</sup> ) | 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   |

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.

## 6 CONCLUSION

In this paper, to remedy the weakness of first-order cost-sensitive online learning algorithms, we propose to

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  |

TABLE 8  
Evaluation for Online Anomaly Detection

| Algorithm               | "sum" on Australian   |         | "sum" on Magic04      |         |
|-------------------------|-----------------------|---------|-----------------------|---------|
|                         | Sum(%)                | Time(s) | 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 <sub>PB</sub>       | 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                 | <b>69.228 ± 0.733</b> | 0.005   | 68.345 ± 1.822        | 0.092   |
| ACOG-I <sub>diag</sub>  | 68.464 ± 0.936        | 0.002   | <b>73.268 ± 0.158</b> | 0.033   |
| ACOG-II <sub>diag</sub> | 68.510 ± 0.917        | 0.002   | 73.035 ± 0.187        | 0.033   |
| Algorithm               | "sum" on DNA          |         | "sum" on KDDCUP08     |         |
|                         | Sum(%)                | Time(s) | 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 <sub>PB</sub>       | 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                  | <b>91.490 ± 0.416</b> | 0.104   | 55.088 ± 0.936        | 4.531   |
| ACOG-II                 | 90.872 ± 0.677        | 0.234   | <b>71.920 ± 2.016</b> | 5.803   |
| ACOG-I <sub>diag</sub>  | 89.498 ± 0.633        | 0.006   | 55.293 ± 0.852        | 0.384   |
| ACOG-II <sub>diag</sub> | 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   |

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.

## ACKNOWLEDGMENTS

This research is partly supported by the National Research Foundation, Prime Ministers Office, Singapore under its International Research Centres in Singapore Funding Initiative, and partly supported by the National Natural Science Foundation of China (NSFC) under Grant 61602185, and the Fundamental Research Funds for the Central Universities under Grant D2172480. Y. Zhang is the co-first author with equal contribution.

## REFERENCES

- [1] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Rev.*, vol. 65, no. 6, 1958, Art. no. 386.
- [2] P. Zhao, S. C. Hoi, and R. Jin, "Double updating online learning," *J. Mach. Learn. Res.*, vol. 12, pp. 1587–1615, 2011.
- [3] J. Wang, P. Zhao, and S. C. Hoi, "Exact soft confidence-weighted learning," in *Proc. Int. Conf. Mach. Learn.*, 2012, pp. 107–114.
- [4] Q. Wu, H. Wu, X. Zhou, M. Tan, Y. Xu, Y. Yan, and T. Hao, "Online transfer learning with multiple homogeneous or heterogeneous sources," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 7, pp. 1494–1507, Jul. 2017.
- [5] P. Zhao and S. C. Hoi, "Cost-sensitive online active learning with application to malicious URL detection," in *Proc. ACM Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 919–927.
- [6] S. C. Hoi, D. Sahoo, J. Lu, and P. Zhao, "Online learning: A comprehensive survey," arXiv preprint arXiv:1802.02871, 2018.
- [7] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Learning to detect malicious urls," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, 2011, Art. no. 30.
- [8] B. Li, S. C. Hoi, P. Zhao, and V. Gopalkrishnan, "Confidence weighted mean reversion strategy for online portfolio selection," *ACM Trans. Knowl. Discovery Data*, vol. 7, no. 1, 2013, Art. no. 4.
- [9] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: Primal estimated sub-gradient solver for SVM," *Math. Program.*, vol. 127, no. 1, pp. 3–30, 2011.
- [10] C. Elkan, "The foundations of cost-sensitive learning," in *Proc. Int. Joint Conf. Artif. Intell.*, pp. 973–978, vol. 17, 2001.
- [11] K. Veropoulos, C. Campbell, and N. Cristianini, "Controlling the sensitivity of support vector machines," in *Proc. Int. Joint Conf. Artif. Intell.*, 1999, pp. 55–60.
- [12] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.
- [13] J. Han, J. Pei, and M. Kamber, "Data mining: Concepts and techniques," Elsevier, 2011.
- [14] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, "The balanced accuracy and its posterior distribution," in *Proc. Int. Conf. Pattern Recognit.*, 2010, pp. 3121–3124.
- [15] R. Akbani, S. Kwek, and N. Japkowicz, "Applying support vector machines to imbalanced datasets," in *Proc. Eur. Conf. Mach. Learn.*, 2004, pp. 39–50.
- [16] J. Wang, P. Zhao, and S. C. Hoi, "Cost-sensitive online classification," in *Proc. IEEE Int. Conf. Data Mining*, 2012, 1140–1145.
- [17] J. Wang, P. Zhao, and S. C. Hoi, "Cost-sensitive online classification," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 10, pp. 2425–2438, Oct. 2014.
- [18] M. Dredze, K. Crammer, and F. Pereira, "Confidence-weighted linear classification," in *Proc. Int. Conf. Mach. Learn.*, 2008, 264–271.
- [19] K. Crammer, M. Dredze, and F. Pereira, "Exact convex confidence-weighted learning," in *Proc. Advances Neural Inf. Process. Syst.*, 2009, pp. 345–352.
- [20] K. Crammer, A. Kulesza, and M. Dredze, "Adaptive regularization of weight vectors," in *Proc. Advances Neural Inf. Process. Syst.*, 2009, pp. 414–422.



- [21] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proc. Int. Conf. Mach. Learn.*, 2003, pp. 928–936.
- [22] H. Luo, A. Agarwal, and N. Cesa-Bianchi, "Efficient second order online learning by sketching," in *Proc. Advances Neural Inf. Process. Syst.*, 2016, pp. 902–910.
- [23] Woodruff and P. David, "Sketching as a tool for numerical linear algebra," *Foundations Trends Theoretical Comput. Sci.*, vol. 10, no. 1–2, pp. 1–157, 2014.
- [24] G. Krummenacher, B. McWilliams, Y. Kilcher, J. M. Buhmann, and N. Meinshausen, "Scalable adaptive stochastic optimization using random projections," in *Proc. Advances Neural Inf. Process. Syst.*, 2016, pp. 1750–1758.
- [25] D. Wang, P. Wu, P. Zhao, Y. Wu, C. Miao, and S. C. Hoi, "High-dimensional data stream classification via sparse online learning," in *Proc. IEEE Int. Conf. Data Mining*, 2014, pp. 1007–1012.
- [26] P. Zhao, F. Zhuang, M. Wu, X. Li, and S. C. H. Hoi, "Cost-sensitive online classification with adaptive regularization and its applications," in *Proc. IEEE Int. Conf. Data Mining*, 2015, pp. 649–658.
- [27] Z. H. Zhou and X. Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 1, pp. 63–77, Jan. 2006.
- [28] R. Horn, *Matrix Analysis*, Cambridge, U.K.: Cambridge Univ. Press, 1985.
- [29] P. Zhao and S. C. Hoi, "Cost-sensitive double updating online learning and its application to online anomaly detection," in *Proc. SIAM Int. Conf. Data Mining*, 2013, pp. 207–215.
- [30] D. Sahoo, S. C. Hoi, and P. Zhao, "Cost sensitive online multiple kernel classification," in *Proc. Asian Conf. Mach. Learn.*, 2016, pp. 65–80.
- [31] P. Zhao and S. C. Hoi, "OTL: A framework of online transfer learning," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 1231–1238.
- [32] P. Zhao, R. Jin, T. Yang, and S. C. Hoi, "Online AUC maximization," in *Proc. Int. Conf. Mach. Learn.*, 2011, pp. 233–240.
- [33] P. Zhang, C. Zhou, P. Wang, B. J. Gao, X. Zhu, and L. Guo, "E-tree: An efficient indexing structure for ensemble models on data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 2, pp. 461–474, Feb. 2015.
- [34] Q. Zhang, P. Zhang, G. Long, W. Ding, C. Zhang, and X. Wu, "Online learning from trapezoidal data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 10, pp. 2709–2723, Oct. 2016.
- [35] Y. Yan, Q. Wu, M. Tan, M. K. Ng, H. Min, and I. W. Tsang, "Online heterogeneous transfer by hedge ensemble of offline and online decisions," *IEEE Trans. Neural Netw. and Learning Syst.*, 2017.
- [36] 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, 2006.
- [37] Y. Zhang, G. Shu, and Y. Li, "Strategy-updating depending on local environment enhances cooperation in prisoners dilemma game," *Appl. Math. Comput.*, vol. 301, pp. 224–232, 2017.
- [38] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," *Mach. Learn.*, vol. 37, pp. 277–296, 1999.
- [39] Y. Li and P. M. Long, "The relaxed online maximum margin algorithm," in *Proc. Advances Neural Inf. Process. Syst.*, 2000, pp. 498–504.
- [40] Y. Li, H. Zaragoza, R. Herbrich, J. Shawe-Taylor, and J. Kandola, "The perceptron algorithm with uneven margins," in *Proc. Int. Conf. Mach. Learn.*, 2002, pp. 379–386.
- [41] E. Oja, "Simplified neuron model as a principal component analyzer," *J. Math. Biology*, vol. 15, no. 3, pp. 267–273, 1982.
- [42] E. Oja and J. Karhunen, "On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix," *J. Math. Anal. Appl.*, vol. 106, pp. 69–84, 1985.
- [43] J. Abernethy, P. L. Bartlett, A. Rakhlin, and A. Tewari, "Optimal strategies and minimax lower bounds for online convex games," pp. 1–17, 2008.
- [44] M. Hardt and E. Price, "The noisy power method: A meta algorithm with application," in *Proc. Advances Neural Inf. Process. Syst.*, 2014, pp. 2861–2869.



**Peilin Zhao** received the bachelor's degree from Zhejiang University, and the PhD degree from Nanyang Technological University. He is currently a researcher with SCUT, China. His research interests include large-scale machine learning and its applications to big data analytics. Previously, he has worked at Ant Financial, the Institute for Infocomm Research (I2R), and Rutgers University. In his research areas, he has published more than 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 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** received the BS degree in computer science from the University of Science and Technology of China, 2006, and the PhD degree from Nanyang Technological University, Singapore, in 2011. He is a research scientist in the Data Analytics Department, Institute for Infocomm Research. His research interests include graph mining from large-scale networks, learning from heterogeneous data sources, ensemble learning, and bioinformatics.



**Steven C. H. Hoi** received the bachelor's degree from Tsinghua University, and the master's and PhD degrees from the Chinese University of Hong Kong. He is an associate professor at Singapore Management University (SMU), Singapore. Prior to joining SMU, he was a tenured associate professor at Nanyang Technological University (NTU), Singapore. His research interests include large-scale machine learning with application to a wide range of real-world applications. He has published more than 150 papers in premier conferences and journals, and served as an organizer, area chair, senior PC, TPC member, editor, 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 with 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** received the BE degree from the Huazhong University of Science and Technology, China, the MS degree from the Chinese Academy of Sciences, China, and the PhD degree from Rutgers University. He is an associate professor with the Computer Science and Engineering Department, University of Texas at Arlington. 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.

▷ For more information on this or any other computing topic, please visit our Digital Library at [www.computer.org/publications/dlib](http://www.computer.org/publications/dlib).