Adaptive Cost-Sensitive Online Classification

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

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

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

1 INTRODUCTION

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

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

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

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

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

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

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

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

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

## 2 Setup and Algorithm

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

### 2.1 Problem Setting

Without loss of generality, we consider online binary classification problems here. The main goal is to learn a linear classification model with an updatable predictive vector \( w \) to obtain a sample \( y_t \). Let \( \ell \) be a hinge loss; while for \( \ell \) with sigmoid or logistic function, the objective function is non-convex. Thus, to facilitate the optimization, we replace the indicator function with its convex variants (either one of the following two functions):

\[
\ell^I(w; (x, y)) = \max(0, (\ell(w \cdot x)) - y(w \cdot x)) \tag{I}
\]

\[
\ell^H(w; (x, y)) = (\ell(y(w \cdot x)) - \max(0, 1 - y(w \cdot x))) \tag{II}
\]

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 \( \ell^H(w; (x, y)) \), the change of the slope causes to more “aggressive” updates for specific class.

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

\[
\text{Regret} := \sum_{t=1}^{T} \ell^I(w_t; (x_t, y_t)) - \sum_{t=1}^{T} \ell(w^*; (x_t, y_t)). \tag{4}
\]
where \( w^* = \arg\min_{w} \sum_{t=1}^{T} \ell(w; (x_t, y_t)) \). To solve this optimization problem, the cost-sensitive online gradient descent algorithms (COG) [15], [16] were proposed:

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

where \( \eta \) 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], [17], [18], [19]. Motivated by this discovery, we propose to introduce adaptive regularization to promote the cost-sensitive online classification.

Let us assume the online model satisfies a multivariate Gaussian distribution, i.e., \( w \sim N(\mu, \Sigma) \), 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 \( \text{sign} (\mu^T x) \), 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 \( \text{sign}(\mu^T x) \) 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 values \( \mu_i \) and \( \mu_j \) for feature \( i \) and \( j \).

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

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

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:

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

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 \( \ell_t(\mu_t) + \eta g_t (\mu_t - \mu_t) \), where \( g_t = \partial \ell_t(\mu_t) \). Now, we obtain the final optimization objective by removing constant terms:

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

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:

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

where adopting the Woodbury identity [27] will give:

\[ \Sigma_{t+1} = \Sigma_t - \frac{\Sigma_t x_t x_t^\top \Sigma_t}{\gamma + x_t^\top \Sigma_t x_t}. \]

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:

\[ \mu_{t+1} = \mu_t - \eta \Sigma_t g_t. \]

This is different from AROW [19], 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 Algorithm 1.

---

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

**Input** learning rate \( \eta \); regularized parameter \( \gamma \); bias parameter \( \rho = \frac{\alpha_p + T_p}{\alpha_p + T_p} \) for “sum” and \( \rho = \frac{\alpha_p}{c_n} \) 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 \( O(d^2) \), so the
overall time complexity for ACOG is $O(d^2)$, which is quite slower than the first order COG algorithms, especially for high-dimensional datasets. To reduce the time complexity, we propose to use the diagonal version of ACOG (i.e., ACOG$_{diag}$), which accelerates the speed of ACOG algorithms to $O(d)$. Specifically, only a diagonal version of $\Sigma_t$ 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 $\rho = \alpha_T/T_p$ for sum metric. However, the value of $T_p$ might be unknown in advance on real-world online classification tasks. A practical method is to approximate the ratio $T_p$ according to the empirical distribution of the past training instances, and adaptively update $T_p$ during the online learning process. In addition, we would empirical examine this problem in experiments.

2.3 Theoretical Analysis

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

**Theorem 1.** Let $(x_t, y_t), (x_{t2}, y_{t2}), \ldots, (x_T, y_T)$ 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 $\eta = \sqrt{\frac{\max(||\mu - \mu||^2 Tr(\Sigma^{-1}_T)\gamma}{\rho^2 Tr(\Sigma^{-1}_T)\gamma}}$, the proposed ACOG-I satisfies:

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

where $D_\mu = \max_{x_t}||\mu - \mu||$. In addition, by setting $\eta = \sqrt{\frac{\max(||\mu - \mu||^2 Tr(\Sigma^{-1}_T)\gamma}{\rho^2 Tr(\Sigma^{-1}_T)\gamma}}$, ACOG-II satisfies:

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

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

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

$$\sum_{t=1}^{T} \ell_t(\mu) + D_\mu \sqrt{\gamma Tr(\Sigma^{-1}_T)\gamma \log(\|\Sigma^{-1}_T\|)},$$

and the ACOG-II satisfies:

$$\sum_{t=1}^{T} \ell_t(\mu) + \rho D_\mu \sqrt{\gamma Tr(\Sigma^{-1}_T)\gamma \log(\|\Sigma^{-1}_T\|)}.$$
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 $X_t^T X_t$ can be approximated by $S_t^T S_t$ well, the Eq. (8) can be redefined as:

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

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

$$
\Sigma_{t+1} = I_d - S_t^T H_t S_t,
$$

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

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

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

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

**Input** learning rate $\eta_t$; regularized parameter $\gamma$; sketch size $m$; bias $\rho = \frac{\eta_t^2}{\alpha + 1}$ for “sum” and $\rho = \frac{\eta_t}{\alpha + 1}$ for “cost”.

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

1. for $t = 1 \to T$ do
2. 
3. 
4. 
5. 
6. 
7. if $\ell_t(\mu_t) > 0$ then
8. 
9. 
10. end if
11. end for

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

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

$$
A_t = (I_m - \Gamma_t) A_{t-1} + \Gamma_t \text{diag}\{V_{t-1}^T \hat{x}_t\}^2, 
$$

$$
V_t \leftarrow \text{orth} V_{t-1} + \Gamma_t V_{t-1}^T \hat{x}_t, 
$$

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

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

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

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

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

3.2 Sparse Sketched Algorithm

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

As results, many first order methods can enjoy a per-round running time depending on $s$ rather than $d$. But for SACOG, even when samples are sparse, the sketch matrix $S_t$ still becomes dense quickly, because of the orthonormalizing updating of $V_t$. For this reason, the updates of $\mu_t$ 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 $\mu_t = w_t + Z_{t-1} b_t$, where $w_t \in \mathbb{R}^d$ captures the sparsely updating weights on the complementary subspace, and $b_t \in \mathbb{R}^m$ captures the weights on the subspace form $V_{t-1}$ (same as $Z_{t-1}$). Then, we describe how to sparsely update two weight parts $w_t$ and $b_t$. 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:

$$
\mu_{t+1} = \mu_t - \eta_t (I_d - S_t^T H_t S_t) g_t
$$

$$
= w_t + Z_{t-1} b_t - \eta_t g_t + \eta_t Z_t^T (t\Lambda H_t) F_t Z_t g_t
$$

$$
= \left[ w_t - \eta_t g_t - (Z_t - Z_{t-1})^T b_t \right] + Z_t^T \left[ b_t + \eta F_t^T (t\Lambda H_t) F_t Z_t g_t \right].
$$

**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 $\Lambda = (I_m - \Gamma_t)^2 + \Gamma_t$ 
3. Update $V = \text{orth} V + \Gamma_t V \hat{x}_t^T$;
4. Set $S = (t\Lambda)^\frac{1}{2} V$;
5. Set $H = \text{diag}\{\frac{1}{1 + t\Lambda_{1,1}}, \ldots, \frac{1}{1 + t\Lambda_{m,m}}\}$;
6. Return $(S, H)$.
According to this, we can define the updating rule of $w_t$:

$$w_{t+1} = w_t - \eta g_t - (Z_t - Z_{t-1})^T b_t$$

$$= w_t - \eta g_t - \hat{x}_t \delta^T b_t,$$

and the updating rule of $b_t$:

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

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

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

**Input** learning rate $\eta$, regularized parameter $\gamma$; sketch size $m$; bias $\rho = \frac{\alpha^*}{\sum_{i=1}^m x_i^2}$ for “sum” and $\rho = \frac{\sigma}{c_n}$ for “cost”.

**Initialization** $w_1 = (\delta_{1 \times 1}, b_1 = 0_{m \times 1}$.

**Initialization Sketch** $(\Lambda_0, F_0, Z_0, H_0) \leftarrow$ 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 $\hat{x}_t = \frac{x_t}{\nu}$;
5: $(\Lambda_t, F_t, Z_t, H_t, \delta_t) \leftarrow$ SketchUpdate($\hat{x}_t$);
6: if $\ell_t(\mu_t) > 0$ then
7: $w_{t+1} = w_t - \eta g_t - \hat{x}_t \delta^T b_t$;
8: $b_{t+1} = b_t + \eta F_t^T (t \Lambda_t H_t) F_t Z_t g_t$;
9: $\mu_{t+1} = w_{t+1} + Z_t^T \delta b_t$;
10: else
11: $\mu_{t+1} = \mu_t$, $w_{t+1} = w_t$, $b_{t+1} = b_t$.
12: end if
13: end for

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):

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

Then from Eq. (12), we have:

$$F_t Z_t \leftarrow \text{orth} F_{t-1} Z_{t-1} + \Gamma_t F_{t-1} Z_{t-1} \hat{x}_t \hat{x}_t^T,$$

$$= F_{t-1} (Z_{t-1} + F_{t-1}^{-1} \Gamma_t F_{t-1} Z_{t-1} \hat{x}_t \hat{x}_t^T).$$

We let $Z_t = Z_{t-1} + \delta_t \hat{x}_t^T$, 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 $\langle a, b \rangle = a^T K_t b$ and $K_t = Z_t Z_t^T$ is the Gram matrix (See Algorithm 6). Then, we can update $K_t$ efficiently based on the update of $Z_t$:

$$K_t = \nu Z_t Z_t^T,$$

$$= (Z_{t-1} + \delta_t \hat{x}_t^T) (Z_{t-1} + \delta_t \hat{x}_t^T)^T,$$

$$= K_{t-1} + Z_{t-1} \hat{x}_t \delta_t^T + \delta_t \hat{x}_t Z_{t-1}^T + \delta_t \hat{x}_t \hat{x}_t^T \delta_t^T.$$  

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

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

**Algorithm 5** Sparse Oja’s Sketch for SACOG

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

**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 $\alpha = K f_j / f_i - Q \alpha$ 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 Setup and Testbed**

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

For data preprocessing, all samples are normalized by $x_i \leftarrow \frac{x_i}{∥x_i∥}$, 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 $\alpha_c = \alpha_n = 0.5$ for sum, and $c_p = 0.9$ and $c_n = 0.1$ for cost. The value of $\rho$ was set to $\alpha_p x_p / c_p$ for sum and $\alpha_n x_n / c_n$ for cost, respectively. For CPA$_{PB}$ 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 $[10^{-5}, 10^{-4}, ..., 10^5]$. 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 Winodws machine.

TABLE 1: List of Binary Datasets in Experiments

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Samples</th>
<th>#Features</th>
<th>#Pos/#Neg</th>
</tr>
</thead>
<tbody>
<tr>
<td>covtype</td>
<td>881012</td>
<td>54</td>
<td>1.1</td>
</tr>
<tr>
<td>german</td>
<td>1000</td>
<td>24</td>
<td>1.23</td>
</tr>
<tr>
<td>a9a</td>
<td>48842</td>
<td>123</td>
<td>1.32</td>
</tr>
<tr>
<td>icnn1</td>
<td>141691</td>
<td>22</td>
<td>1.94</td>
</tr>
</tbody>
</table>

4.2 Evaluation with Sum Metrics

4.2.1 Evaluation of Weighted Sum Performance

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

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

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

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

4.2.2 Evaluation of Sum under Varying Weights

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

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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Sum(%)</th>
<th>Sensitivity(%)</th>
<th>Specificity(%)</th>
<th>Time(s)</th>
<th>Cost(1e6)</th>
<th>Sensitivity(%)</th>
<th>Specificity(%)</th>
<th>Time(s)</th>
<th>Cost(1e6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perception</td>
<td>1.233</td>
<td>0.357</td>
<td>0.957</td>
<td>0.459</td>
<td>4.046</td>
<td>0.353</td>
<td>0.957</td>
<td>0.459</td>
<td>4.046</td>
</tr>
<tr>
<td>ROMMA</td>
<td>0.036</td>
<td>0.357</td>
<td>0.957</td>
<td>0.459</td>
<td>4.046</td>
<td>0.353</td>
<td>0.957</td>
<td>0.459</td>
<td>4.046</td>
</tr>
<tr>
<td>PA-I</td>
<td>0.209</td>
<td>0.357</td>
<td>0.957</td>
<td>0.459</td>
<td>4.046</td>
<td>0.353</td>
<td>0.957</td>
<td>0.459</td>
<td>4.046</td>
</tr>
<tr>
<td>PA-U</td>
<td>0.209</td>
<td>0.357</td>
<td>0.957</td>
<td>0.459</td>
<td>4.046</td>
<td>0.353</td>
<td>0.957</td>
<td>0.459</td>
<td>4.046</td>
</tr>
<tr>
<td>COG-I</td>
<td>0.209</td>
<td>0.357</td>
<td>0.957</td>
<td>0.459</td>
<td>4.046</td>
<td>0.353</td>
<td>0.957</td>
<td>0.459</td>
<td>4.046</td>
</tr>
<tr>
<td>COG-II</td>
<td>0.209</td>
<td>0.357</td>
<td>0.957</td>
<td>0.459</td>
<td>4.046</td>
<td>0.353</td>
<td>0.957</td>
<td>0.459</td>
<td>4.046</td>
</tr>
<tr>
<td>COG-IA</td>
<td>0.209</td>
<td>0.357</td>
<td>0.957</td>
<td>0.459</td>
<td>4.046</td>
<td>0.353</td>
<td>0.957</td>
<td>0.459</td>
<td>4.046</td>
</tr>
<tr>
<td>COG-Ia</td>
<td>0.209</td>
<td>0.357</td>
<td>0.957</td>
<td>0.459</td>
<td>4.046</td>
<td>0.353</td>
<td>0.957</td>
<td>0.459</td>
<td>4.046</td>
</tr>
</tbody>
</table>

Overall, cost of ACOG is about less than half of cost made by all regular first-order algorithms (i.e., Perceptron, ROMMA, PA-I, PA-U and CPA-P). 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 often achieve the best sensitivity result on all datasets, and attain a relatively good specificity among all cases.

Moreover, the diagonal COG-IA algorithms methods achieve higher cost value than ACOG methods, but their running time is lower. This is similar with the situation based on sum metric. Thus, the COG-IA methods can be regarded as a choice to balance the performance and efficiency.

4.3.2 Evaluation of Cost under Varying Weights

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

4.4 Evaluation of Algorithm properties

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

Moreover, we find the proposed methods with objective function $\ell_{\Theta}(w; (x, y))$ can achieve relatively higher performance than the methods with $\ell_{\Theta}(w; (x, y))$, which means that ACOG-II and ACOG-II$_{diag}$ 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 $\Sigma_{t+1} = \Sigma_t - \sum_{x,y} x^t y^t \Sigma_t / \sum_{x,y}^t$, 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 $[10^{-4}, 10^{-3}, ..., 10^3, 10^4]$ for $sum$ metrics.

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

### 4.4.3 Evaluation of Updating Rule

As mentioned in Section 2, the predictive vector $\mu$ 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 subsection, we would like to evaluate the difference between two updating rules based on $sum$ metrics for proposed methods, where the invariant versions (i.e., green line in Fig. 7) depending on old $\Sigma_t$.

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 $\frac{T_p}{T_p}$

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

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

Fig. 8 shows the performance of online estimation. We find that the online laplace estimation performs quite similar results with the original one. This discovery validates the practical value of the proposed ACOG$_{\text{sum}}$ 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 5-fold cross-validation for better validation of the general performance.

Table 3 summary the consequences on $\text{sum}$ 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.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>a9a</th>
<th>covtype</th>
<th>german</th>
<th>ijcnn1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perceptron</td>
<td>68.649</td>
<td>51.353</td>
<td>53.737</td>
<td>70.045</td>
</tr>
<tr>
<td>ROMMA</td>
<td>72.467</td>
<td>67.059</td>
<td>58.614</td>
<td>76.818</td>
</tr>
<tr>
<td>PA-I</td>
<td>71.986</td>
<td>51.283</td>
<td>51.363</td>
<td>70.410</td>
</tr>
<tr>
<td>PAUM</td>
<td>79.323</td>
<td>53.354</td>
<td>52.126</td>
<td>82.012</td>
</tr>
<tr>
<td>CPA$_{FB}$</td>
<td>73.668</td>
<td>51.279</td>
<td>52.768</td>
<td>73.942</td>
</tr>
<tr>
<td>AROW</td>
<td>75.961</td>
<td>64.928</td>
<td>54.575</td>
<td>67.642</td>
</tr>
<tr>
<td>COG-I</td>
<td>79.705</td>
<td>53.354</td>
<td>52.258</td>
<td>82.012</td>
</tr>
<tr>
<td>COG-II</td>
<td>78.559</td>
<td>68.897</td>
<td>50.784</td>
<td>82.849</td>
</tr>
<tr>
<td>ACOG-I</td>
<td>80.026</td>
<td>72.428</td>
<td>62.954</td>
<td>82.926</td>
</tr>
<tr>
<td>ACOG-II</td>
<td>81.630</td>
<td>72.632</td>
<td>60.928</td>
<td>87.730</td>
</tr>
<tr>
<td>ACOG-I$_{\text{diag}}$</td>
<td>80.118</td>
<td>71.051</td>
<td>64.389</td>
<td>82.334</td>
</tr>
<tr>
<td>ACOG-II$_{\text{diag}}$</td>
<td>81.752</td>
<td>71.311</td>
<td>66.036</td>
<td>87.628</td>
</tr>
</tbody>
</table>

**Table 3** Evaluation of generalization ability with $\text{sum}$

4.5 Performance and Efficiency of Sketched ACOG

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

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

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

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

**TABLE 4: Datasets for Evaluation of Sketched Algorithm**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Examples</th>
<th>#Features</th>
<th>#Pos#Neg</th>
</tr>
</thead>
<tbody>
<tr>
<td>mushrooms</td>
<td>8124</td>
<td>172</td>
<td>1:1.37</td>
</tr>
<tr>
<td>protein</td>
<td>17766</td>
<td>357</td>
<td>1:1.7</td>
</tr>
<tr>
<td>usps</td>
<td>7291</td>
<td>256</td>
<td>1:5.11</td>
</tr>
<tr>
<td>Sensorless</td>
<td>58509</td>
<td>48</td>
<td>1:10</td>
</tr>
</tbody>
</table>

### 4.5.1 Evaluation of Weighted Sum Performance

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

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

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

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

Then, We would like to compare the performance and running time between SACOG and its sparse version 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:
TABLE 5: Evaluation of the Cost-Sensitive Classification Performance of SSACOG

| Algorithm | #sum | sensitivity | specificity | cost | #sum | sensitivity | specificity | cost | #sum | sensitivity | specificity | cost | #sum | sensitivity | specificity | cost | #sum | sensitivity | specificity | cost | #sum | sensitivity | specificity | cost |
|-----------|------|-------------|-------------|------|------|-------------|-------------|------|------|-------------|-------------|------|------|-------------|-------------|------|------|-------------|-------------|------|------|-------------|-------------|------|------|
| COG-I     | 88.620 ± 0.195 | 86.816 ± 0.345 | 0.97 | 16.082 ± 0.796 | 0.96 | 71.544 ± 0.197 | 3.769 | 122.880 ± 17.094 | 3.302 |
| COG-II    | 99.504 ± 0.040 | 97.996 ± 0.230 | 9.279 | 17.905 ± 1.852 | 0.035 | 71.907 ± 0.180 | 3.705 | 1147.775 ± 14.364 | 2.373 |
| SASCOG-I  | 93.913 ± 0.129 | 98.158 ± 0.230 | 1.227 | 7.895 ± 0.435 | 0.097 | 71.900 ± 0.204 | 0.285 | 1144.680 ± 13.087 | 0.239 |

TABLE 6: Evaluation between SACOG and Sparse SACOG

| Algorithm | #sum | sensitivity | specificity | cost | #sum | sensitivity | specificity | cost | #sum | sensitivity | specificity | cost | #sum | sensitivity | specificity | cost | #sum | sensitivity | specificity | cost | #sum | sensitivity | specificity | cost |
|-----------|------|-------------|-------------|------|------|-------------|-------------|------|------|-------------|-------------|------|------|-------------|-------------|------|------|-------------|-------------|------|------|-------------|-------------|------|------|
| COG-I     | 99.620 ± 0.043 | 99.423 ± 0.094 | 0.993 | 17.590 ± 2.796 | 0.969 | 99.286 ± 0.070 | 9.068 ± 0.004 | 0.020 |
| COG-II    | 99.211 ± 0.057 | 99.423 ± 0.094 | 0.993 | 17.590 ± 2.796 | 0.969 | 99.286 ± 0.070 | 9.068 ± 0.004 | 0.020 |
| ACOG-I    | 99.586 ± 0.027 | 99.810 ± 0.070 | 9.356 | 17.835 ± 1.062 | 0.043 | 9.187 ± 0.293 | 9.093 ± 0.170 | 9.952 ± 0.443 | 0.085 |
| ACOG-II   | 99.572 ± 0.033 | 99.794 ± 0.080 | 9.349 | 17.670 ± 0.776 | 0.045 | 9.187 ± 0.293 | 9.093 ± 0.170 | 9.952 ± 0.443 | 0.085 |
| ACOG-II_marg | 99.447 ± 0.052 | 99.652 ± 0.077 | 9.243 | 17.520 ± 0.782 | 0.019 | 9.187 ± 0.293 | 9.093 ± 0.170 | 9.952 ± 0.443 | 0.085 |
| ACOG-II_diag | 99.457 ± 0.052 | 99.652 ± 0.077 | 9.243 | 17.520 ± 0.782 | 0.019 | 9.187 ± 0.293 | 9.093 ± 0.170 | 9.952 ± 0.443 | 0.085 |
| SSACOG-I  | 99.626 ± 0.052 | 99.798 ± 0.066 | 9.499 | 17.880 ± 1.677 | 0.038 | 9.187 ± 0.293 | 9.093 ± 0.170 | 9.952 ± 0.443 | 0.085 |
| SSACOG-II | 99.606 ± 0.050 | 99.805 ± 0.062 | 9.408 | 17.560 ± 3.870 | 0.038 | 9.187 ± 0.293 | 9.093 ± 0.170 | 9.952 ± 0.443 | 0.085 |

- 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 1920 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. 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 summarize the detailed information for each dataset in Table 7.

5.2 Empirical Evaluation Results

In this subsection, our algorithms are applied to address real-world anomaly detection tasks with 4 datasets from different domains, where we use the balanced accuracy metric to avoid inflated performance evaluations on imbalanced datasets. In addition, we apply our sparse s-ketched 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\textsubscript{FB}) 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 on-line optimization using the second order information.

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

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

6 CONCLUSION

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

Moreover, to overcome the time-consuming problem of our second-order algorithms, we further study the sketching method in cost-sensitive online classification framework, and then propose sketched second-order 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.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>“(sum)” on Australian</th>
<th>“(sum)” on Magic04</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perception</td>
<td>59.863 ± 1.327</td>
<td>59.154 ± 0.488</td>
</tr>
<tr>
<td>ROMMA</td>
<td>58.732 ± 3.462</td>
<td>64.025 ± 3.277</td>
</tr>
<tr>
<td>PA-I</td>
<td>57.103 ± 1.595</td>
<td>58.029 ± 0.312</td>
</tr>
<tr>
<td>PAUM</td>
<td>62.962 ± 0.941</td>
<td>64.671 ± 2.024</td>
</tr>
<tr>
<td>CPA\textsubscript{FB}</td>
<td>57.110 ± 1.599</td>
<td>58.488 ± 0.360</td>
</tr>
<tr>
<td>AROW</td>
<td>67.174 ± 0.749</td>
<td>70.896 ± 0.190</td>
</tr>
<tr>
<td>COG-I</td>
<td>65.972 ± 0.879</td>
<td>65.913 ± 0.189</td>
</tr>
<tr>
<td>COG-II</td>
<td>67.213 ± 0.787</td>
<td>69.815 ± 0.185</td>
</tr>
<tr>
<td>ACOG-I</td>
<td>68.808 ± 0.894</td>
<td>72.935 ± 0.186</td>
</tr>
<tr>
<td>ACOG-II</td>
<td>69.222 ± 0.733</td>
<td>68.345 ± 1.822</td>
</tr>
<tr>
<td>ACOG\textsubscript{diag}-I</td>
<td>68.464 ± 0.936</td>
<td>73.268 ± 0.158</td>
</tr>
<tr>
<td>ACOG\textsubscript{diag}-II</td>
<td>68.510 ± 0.917</td>
<td>73.035 ± 0.183</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>“(sum)” on DNA</th>
<th>“(sum)” on KDDCUP98</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perception</td>
<td>84.799 ± 0.745</td>
<td>84.918 ± 0.106</td>
</tr>
<tr>
<td>ROMMA</td>
<td>85.782 ± 0.553</td>
<td>85.342 ± 1.581</td>
</tr>
<tr>
<td>PA-I</td>
<td>87.832 ± 0.833</td>
<td>54.053 ± 0.865</td>
</tr>
<tr>
<td>PAUM</td>
<td>88.560 ± 0.737</td>
<td>55.161 ± 4.236</td>
</tr>
<tr>
<td>CPA\textsubscript{FB}</td>
<td>89.401 ± 0.645</td>
<td>57.318 ± 0.629</td>
</tr>
<tr>
<td>AROW</td>
<td>87.183 ± 0.405</td>
<td>50.611 ± 0.422</td>
</tr>
<tr>
<td>COG-I</td>
<td>87.866 ± 0.812</td>
<td>59.044 ± 0.104</td>
</tr>
<tr>
<td>Artificial</td>
<td>87.385 ± 0.330</td>
<td>65.912 ± 1.855</td>
</tr>
<tr>
<td>ACOG-I</td>
<td>91.490 ± 0.416</td>
<td>50.088 ± 0.936</td>
</tr>
<tr>
<td>ACOG-II</td>
<td>90.872 ± 0.677</td>
<td>71.920 ± 2.016</td>
</tr>
<tr>
<td>ACOG\textsubscript{diag}-I</td>
<td>89.498 ± 0.633</td>
<td>55.293 ± 0.852</td>
</tr>
<tr>
<td>ACOG\textsubscript{diag}-II</td>
<td>88.483 ± 0.490</td>
<td>71.661 ± 1.336</td>
</tr>
<tr>
<td>SSACOG-I</td>
<td>89.975 ± 0.516</td>
<td>55.711 ± 0.812</td>
</tr>
<tr>
<td>SSACOG-II</td>
<td>90.444 ± 0.471</td>
<td>70.947 ± 1.179</td>
</tr>
</tbody>
</table>

REFERENCES

M. Hardt, E. Price. The noisy power method: A meta algorithm.


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

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

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

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

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

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