Hindawi Publishing Corporation
Abstract and Applied Analysis
Volume 2014, Article ID 416591, 9 pages
http://dx.doi.org/10.1155/2014/416591

*Research Article*

# Cost-Sensitive Support Vector Machine Using Randomized Dual Coordinate Descent Method for Big Class-Imbalanced Data Classification

**Mingzhu Tang,[1,2] Chunhua Yang,[3] Kang Zhang,[1] and Qiyue Xie[1]**

[1] *School of Energy and Power Engineering, Changsha University of Science & Engineering, Changsha 410114, China*
[2] *Hunan Province University Key Laboratory of Bridge Engineering, Changsha University of Science & Technology, Changsha, Hunan 410114, China*
[3] *School of Information Science and Engineering, Central South University, Changsha 410083, China*

Correspondence should be addressed to Mingzhu Tang; tmzhu@163.com

Received 14 April 2014; Accepted 31 May 2014; Published 3 July 2014

Academic Editor: Fuding Xie



Cost-sensitive support vector machine is one of the most popular tools to deal with class-imbalanced problem such as fault diagnosis. However, such data appear with a huge number of examples as well as features. Aiming at class-imbalanced problem on big data, a cost-sensitive support vector machine using randomized dual coordinate descent method (CSVM-RDCD) is proposed in this paper. The solution of concerned subproblem at each iteration is derived in closed form and the computational cost is decreased through the accelerating strategy and cheap computation. The four constrained conditions of CSVM-RDCD are derived. Experimental results illustrate that the proposed method increases recognition rates of positive class and reduces average misclassification costs on real big class-imbalanced data.

## 1. Introduction

The most popular strategy for the design of classification algorithms is to minimize the probability of error, assuming that all misclassifications have the same cost and classes of dataset are balanced [1–6]. The resulting decision rules are usually denoted as cost insensitive. However, in many important applications of machine learning, such as fault diagnosis [7] and fraud detection, certain types of error are much more costly than others. Other applications involve significantly class-imbalanced datasets, where examples from different classes appear with substantially different probability. Cost-sensitive support vector machine (CSVM) [2] is one of the most popular tools to deal with class-imbalanced problem and unequal misclassification problem. However, in many applications, such data appear with a huge number of examples as well as features.

In this work we consider the cost-sensitive support vector machine architecture [1]. Although CSVMs are based on a very solid learning-theoretic foundation and have been successfully applied to many classification problems, it is not well understood how to design big data learning of the CSVM algorithm. CSVM usually maps training vectors into a high dimensional space via a nonlinear function. Due to the high dimensionality of the weight vector, one solves the dual problem of CSVM by the kernel trick. In some applications, data appear in a rich dimensional feature space; the performances are similar with/without nonlinear mapping. If data are not mapped, we can often train much data set.

Recently, many methods have been proposed for linear SVM in large-scale scenarios [4]. In all methods, dual coordinate descent methods for dual problem of CSVM are one of popular methods to deal with large-scale convex optimization problem. However, they do not focus on big data learning of CSVM. We focus on big data class-imbalanced learning by CSVM.

This paper is organized as follows. In Section 2 basic theory of cost-sensitive support vector machine is described. In Section 3 we derive our proposed algorithm. Section 4

discusses both speed-ups and four constrained conditions of cost-sensitive support vector machine. Implementation issues are investigated in Section 5. Experiments show efficiently our proposed method.

## 2. Basic Theory of Cost-Sensitive Support Vector Machine

Cost-sensitive support vector machine such as 2C-SVM is proposed in [1]. Consider examples set $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, $\mathbf{x}_i \in R^d$, $y_i \in Y = \{+1, -1\}$, $I_+ = \{i : y_i = +1\}$, $I_- = \{i : y_i = -1\}$. The 2C-SVM has primal optimization problem:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\gamma \sum_{i\in I_+} \xi_i + C(1-\gamma) \sum_{i\in I_-} \xi_i \\ \text{s.t.} \quad & y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \xi_i \quad i = 1, 2, \ldots, l \\ & \xi_i \geq 0 \quad i = 1, 2, \ldots, l \\ & C > 0. \end{aligned} \tag{1}$$

Lagrange multipliers method can be used to solve the constrained optimization problem. In this method, the Lagrange equation is defined as follows:

$$\begin{aligned} & L(\mathbf{w}, b, \xi) \\ & = \frac{1}{2}\|\mathbf{w}\|^2 + C\gamma \sum_{i\in I_+} \xi_i + C(1-\gamma) \sum_{i\in I_-} \xi_i \\ & \quad + \sum_{i=1}^{l} \alpha_i \left[1 - \xi_i - y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right)\right] + \sum_{i=1}^{l} \beta_i \left[-\xi_i\right], \end{aligned} \tag{2}$$

where the $\alpha_i$, $\beta_i$'s are called the Lagrange multipliers. $L$'s partial derivatives are set to be zero as follows:

$$\frac{\partial L(\mathbf{w}, b, \xi)}{\partial \mathbf{w}} = 0, \qquad \frac{\partial L(\mathbf{w}, b, \xi)}{\partial b} = 0, \qquad \frac{\partial L(\mathbf{w}, b, \xi)}{\partial \xi_i} = 0. \tag{3}$$

And solve $\mathbf{w}, b, \xi$. The formula (3) is extended to be

$$\begin{aligned} \frac{\partial L(\mathbf{w}, b, \xi)}{\partial \mathbf{w}} &= \mathbf{w} + \sum_{i=1}^{l} \alpha_i \left[-y_i\mathbf{x}_i\right] = 0 \\ \frac{\partial L(\mathbf{w}, b, \xi)}{\partial b} &= \sum_{i=1}^{l} \alpha_i \left(-y_i\right) = 0 \\ \frac{\partial L(\mathbf{w}, b, \xi)}{\partial \xi_i} &= C\gamma - \alpha_i - \beta_i = 0, \quad i \in I_+, \\ \frac{\partial L(\mathbf{w}, b, \xi)}{\partial \xi_i} &= C(1-\gamma) - \alpha_i - \beta_i = 0, \quad i \in I_-. \end{aligned} \tag{4}$$

Equation (4) can be reformatted as

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i, \tag{5}$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0, \tag{6}$$

$$\begin{aligned} C\gamma - \alpha_i - \beta_i = 0, \quad & i \in I_+, \\ C(1-\gamma) - \alpha_i - \beta_i = 0, \quad & i \in I_-. \end{aligned} \tag{7}$$

By incorporating (5) to (7), Lagrange equation (2) is rewritten as

$$\begin{aligned} & L(w, b, \xi) \\ & = \frac{1}{2}\left\|\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i\right\|^2 + C\gamma \sum_{i\in I_+} \xi_i + C(1-\gamma) \sum_{i\in I_-} \xi_i \\ & \quad + \sum_{i=1}^{l} \alpha_i \left[1 - \xi_i - y_i\left(\left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i\right)^T \mathbf{x}_i + b\right)\right] \\ & \quad + \sum_{i\in I_+} (C\gamma - \alpha_i)\left[-\xi_i\right] + \sum_{i\in I_-} (C(1-\gamma) - \alpha_i)\left[-\xi_i\right]. \end{aligned} \tag{8}$$

Lagrange equation (8) is simplified as

$$\begin{aligned} & L(w, b, \xi) \\ & = -\frac{1}{2}\sum_{i=1}^{l} \alpha_i \left[y_i\left(\left(\sum_{j=1}^{l} \alpha_j y_j \mathbf{x}_j\right)^T \mathbf{x}_i\right)\right] + \sum_{i=1}^{l} \alpha_i. \end{aligned} \tag{9}$$

Recall that the equation above is obtained by minimizing $L$ with respect to $\mathbf{w}$ and $b$. Putting this together with the constraints $\alpha_i \geq 0$ and the constraints (5) to (7), the following dual optimization problem of 2C-SVM is obtained as

$$\begin{aligned} \min_{\alpha} \quad & f(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j - \sum_{i=1}^{l} \alpha_i \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C\gamma, \quad i \in I_+ \\ & 0 \leq \alpha_i \leq C(1-\gamma), \quad i \in I_- \\ & \sum_{i=1}^{l} \alpha_i y_i = 0, \end{aligned} \tag{10}$$

where $\alpha_i, i = 1, \ldots, l$ are Lagrange multipliers and $C\gamma$, $C(1-\gamma)$, $C > 0$, $\gamma \in [0, 1]$ are misclassification cost parameters. The matrixes $Q$ and $\boldsymbol{\alpha}$ are defined, respectively, as follows:

$$\begin{aligned} & Q = \left[Q_{ij}\right]_{l\times l}, \quad \text{where } Q_{ij} = \alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j, \\ & \boldsymbol{\alpha} = \left[\alpha_1, \ldots, \alpha_i, \ldots, \alpha_j, \ldots, \alpha_l\right]. \end{aligned} \tag{11}$$

## 3. Cost-Sensitive Support Vector Machine Using Randomized Dual Coordinate Descent Method

In this section, randomized dual coordinate descent method is used to solve 2C-SVM that is one version of cost-sensitive support vector machine. The optimization process starts from an initial point $\boldsymbol{\alpha}^0 \in \mathbf{R}^l$ and generates a sequence of vectors $\boldsymbol{\alpha}^k$, $k = 0, \ldots, \infty$. The process from $\boldsymbol{\alpha}^k$ to $\boldsymbol{\alpha}^{k+1}$ is called an outer iteration. In each outer iteration $l$ is called inner iteration, so that $\alpha_1, \alpha_2, \ldots, \alpha_l$ are sequentially updated. Each outer iteration thus generates vectors $\boldsymbol{\alpha}^{k,i} \in \mathbf{R}^l, i = 1, 2, \ldots, l+1$, such that

$$\begin{aligned} \boldsymbol{\alpha}^{k,1} &= \boldsymbol{\alpha}^k, \quad i = 1, \\ \boldsymbol{\alpha}^{k,i} &= \left[\alpha_1^{k+1}, \ldots, \alpha_{i-1}^{k+1}, \alpha_i^k, \ldots, \alpha_l^k\right], \quad \forall i = 2, \ldots, l, \\ \boldsymbol{\alpha}^{k,l} &= \boldsymbol{\alpha}^k, \quad i = l+1. \end{aligned} \tag{12}$$

For updating $\boldsymbol{\alpha}^{k,i}$ to $\boldsymbol{\alpha}^{k,i+1}$, the following one-variable subproblem is solved as

$$\min_d f\left(\boldsymbol{\alpha}^{k,i} + de_i\right), \tag{13}$$

where $e_i = [0, \ldots, 1, \ldots, 0]^T$. This is general process of one-variable coordinate descent method. However, the sum constrained condition of the optimization problem (13) is denoted as follows:

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \tag{14}$$

where $\alpha_i$ is exactly determined by the other $\alpha_i$,s, and if we were to hold $\{\alpha_i\}_{i=1}^n \setminus \alpha_i$ fixed, then we cannot make any change $\alpha_i$ without violating the constraint condition (14) in the optimization problem.

Thus if we want to update some subject of the $\alpha_i$,s, we must update at least two of them simultaneously in order to keep satisfying the constraints. We currently have some setting of the $\alpha_i$s that satisfy the constraint conditions (14) and suppose we have decided to hold $\{\alpha_i\}_{i=1}^n \setminus \alpha_i, \alpha_j$ fixed and reoptimize the dual problem of CSVM (10) with respect to $\alpha_i$, $\alpha_j$ (subject to the constraints). Equation (6) is reformatted as

$$\alpha_i y_i + \alpha_j y_j = \sum_{m=1}^{l} \alpha_m y_m - \alpha_i y_i - \alpha_j y_j. \tag{15}$$

Since the right hand side is fixed, we can just let it be denoted by some constant $\zeta$:

$$\zeta = \sum_{m=1}^{l} \alpha_m y_m - \alpha_i y_i - \alpha_j y_j. \tag{16}$$

We can form the updated coordinates before and after with respect to $\alpha_i$, $\alpha_j$:

$$\alpha_i^k y_i + \alpha_j^k y_j = \alpha_i^{k+1} y_i + \alpha_j^{k+1} y_j = \zeta. \tag{17}$$

$\alpha_i^k y_i$, $\alpha_i^{k+1} y_i$ are updated coordinates before and after with respect to $\alpha_i^k$. Consider the following dual two-variable subproblem of 2C-SVM:

$$\begin{aligned} &f\left(\boldsymbol{\alpha}^{k,i} + d_i e_i\right) + f\left(\boldsymbol{\alpha}^{k,j} + d_j e_j\right) \\ &\quad = Q_{ii} d_i^2 + \left\{2(\alpha^{k,i}\overline{Q})_i - 1\right\} d_i \\ &\qquad + Q_{jj} d_j^2 + \left\{2(\alpha^{k,j}\overline{Q})_j - 1\right\} d_j + c, \end{aligned} \tag{18}$$

where $c$ is constant, $Q_i = [Q_{i,s}, \ldots, Q_{i,s-1}, Q_{i,s}, \ldots, Q_{i,l}]$, and

$$(\alpha^{k,i}\overline{Q})_i = \sum_{s=1}^{l} Q_{i,s} \alpha_i^{k,s}. \tag{19}$$

From (18), we obtain

$$f'(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^T Q - \mathbf{1}, \tag{20}$$

$$\frac{\partial f\left(\boldsymbol{\alpha}^{k,i}\right)}{\partial \alpha_i} = 2(\boldsymbol{\alpha}^{k,i}\overline{Q})_i - 1, \tag{21}$$

where $\partial f(\boldsymbol{\alpha}^{k,i})/\partial \alpha_i$ is the $i$th component of the gradient $f'(\boldsymbol{\alpha})$. By incorporating (21), (18) is rewritten as

$$\begin{aligned} &f\left(\boldsymbol{\alpha}^{k,i} + d_i e_i\right) + f\left(\boldsymbol{\alpha}^{k,j} + d_j e_j\right) \\ &\quad = Q_{ii} d_i^2 + \frac{\partial f\left(\boldsymbol{\alpha}^{k,i}\right)}{\partial \alpha_i} d_i + Q_{jj} d_j^2 + \frac{\partial f\left(\boldsymbol{\alpha}^{k,j}\right)}{\partial \alpha_j} d_j + c. \end{aligned} \tag{22}$$

From (17), we have $d_j = -d_i y_i y_j$. Equation (22) is reformed as follows:

$$\begin{aligned} &f\left(\boldsymbol{\alpha}^{k,i} + d_i e_i\right) + f\left(\boldsymbol{\alpha}^{k,j} + d_j e_j\right) \\ &\quad = Q_{ii} d_i^2 + \frac{\partial f\left(\boldsymbol{\alpha}^{k,i}\right)}{\partial \alpha_i} d_i + Q_{jj} d_i^2 - \frac{\partial f\left(\boldsymbol{\alpha}^{k,j}\right)}{\partial \alpha_j} d_i y_i y_j + c \\ &\quad = \left(Q_{ii} + Q_{jj}\right) d_i^2 + \left(\frac{\partial f\left(\boldsymbol{\alpha}^{k,i}\right)}{\partial \alpha_i} - \frac{\partial f\left(\boldsymbol{\alpha}^{k,j}\right)}{\partial \alpha_j} y_i y_j\right) d_i + c. \end{aligned} \tag{23}$$

Treating $(Q_{ii} + Q_{jj})$, $((\partial f(\boldsymbol{\alpha}^{k,i})/\partial \alpha_i) - (\partial f(\boldsymbol{\alpha}^{k,j})/\partial \alpha_j) y_i y_j)$, $c$ as constants, we should be able to verify that this is just some quadratic function in $d_i$. We can easily maximize this quadratic function by setting its derivative to zero and solving the optimization problem. The following two-variable optimization problem is obtained as

$$\min_{d_i} f\left(\boldsymbol{\alpha}^{k,i} + d_i e_i\right) + f\left(\boldsymbol{\alpha}^{k,j} + d_j e_j\right). \tag{24}$$

The closed form is derived as follows:

$$d_i = \frac{\left(\partial f\left(\boldsymbol{\alpha}^{k,i}\right)/\partial\alpha_i\right) - \left(\partial f\left(\boldsymbol{\alpha}^{k,j}\right)/\partial\alpha_j\right)\left(y_i y_j\right)}{2\left(Q_{ii} + Q_{jj}\right)}. \tag{25}$$

We now consider the box constraints (10) of the two-variable optimization problem. The box constraints as $0 \le \alpha_i \le C\gamma$, $i \in I_+$, $0 \le \alpha_i \le C(1-\gamma)$, $i \in I_-$ are classified to four boxes constraints according to the labels of examples from some two coordinates.

Firstly, suppose that the labels of examples are as

$$y_i = +1, \qquad y_j = -1. \tag{26}$$

The sum constraints of Lagrange multipliers according to these labels are as

$$\alpha_i y_i + \alpha_j y_j = \zeta. \tag{27}$$

Another expressing of sum constraints of Lagrange multipliers is as

$$\begin{aligned} &\alpha_i - \alpha_j = \zeta, \\ &\text{or } \alpha_i = \alpha_j + \zeta. \end{aligned} \tag{28}$$

The box constraints of Lagrange multipliers are defined as

$$\begin{aligned} &0 \le \alpha_i \le \gamma C, \\ &0 \le \alpha_j \le (1-\gamma) C. \end{aligned} \tag{29}$$

We obtain the new expressing of box constraints of Lagrange multipliers as the following:

$$\begin{aligned} &0 \le \zeta \le (1-\gamma) C, \\ &0 \le \alpha_i \le \alpha_i - \alpha_j - (1-\gamma) C, \\ &\text{or } -(1-\gamma) C \le \zeta \le 0, \\ &\alpha_i - \alpha_j \le \alpha_i \le \gamma C. \end{aligned} \tag{30}$$

Thus we obtain stricter box constraints of Lagrange multipliers $\alpha_i$, $\alpha_j$ according to $y_i = +1$, $y_j = -1$ as the following:

$$\begin{aligned} U_1 &= \max\left(0, \alpha_i - \alpha_j\right), \\ V_1 &= \min\left(\alpha_i - \alpha_j - (1-\gamma) C, \gamma C\right). \end{aligned} \tag{31}$$

Secondly, suppose that the labels of examples are as

$$y_i = +1, \qquad y_j = +1. \tag{32}$$

Similarly, similar stricter box constraints of Lagrange multipliers $\alpha_i$, $\alpha_j$ are obtained as follows:

$$\begin{aligned} U_2 &= \max\left(0, \alpha_i + \alpha_j - \gamma C\right), \\ V_2 &= \min\left(\alpha_i + \alpha_j, \gamma C\right). \end{aligned} \tag{33}$$

Thirdly, suppose that the labels of examples are as

$$y_i = -1, \qquad y_j = -1. \tag{34}$$

Similarly, the similar stricter box constraints of Lagrange multipliers $\alpha_i, \alpha_j$ are obtained as follows:

$$\begin{aligned} U_3 &= \max\left(0, -\alpha_i - \alpha_j - (1-\gamma) C\right), \\ V_3 &= \min\left(-\alpha_i - \alpha_j, (1-\gamma) C\right). \end{aligned} \tag{35}$$

Finally, suppose that the labels of examples are as

$$y_i = -1, \qquad y_j = +1. \tag{36}$$

Similarly, the similar stricter box constraints of Lagrange multipliers $\alpha_i, \alpha_j$ are obtained as follows:

$$\begin{aligned} U_4 &= \max\left(0, -\alpha_i - \alpha_j\right), \\ V_4 &= \min\left(\gamma C - \alpha_i - \alpha_j, (1-\gamma) C\right). \end{aligned} \tag{37}$$

For the simplification, set

$$\alpha_i^{\text{temp}} = \alpha_i^{k,i} + d_i, \tag{38}$$

$\alpha_i^{k,i+1}$ is defined as the temp solution, which would be edited to satisfy

$$\alpha_i^{k,i+1} = \begin{cases} V_i, & \text{if } \alpha_i^{\text{temp}} > V_i, \ i = 1,2,3,4 \\ \alpha_i^{\text{temp}}, & \text{if } U_i < \alpha_i^{\text{temp}} < V_i, \ i = 1,2,3,4 \\ U_i, & \text{if } \alpha_i^{\text{temp}} < U_i, \ i = 1,2,3,4. \end{cases} \tag{39}$$

From linear constraints with respect to $\alpha_i^{k,i+1}$ in (17), the value of $\alpha_i^{k,j+1}$ is obtained as

$$\alpha_i^{k,j+1} = y_j\left(y_i \alpha_i^{k,i} + y_j \alpha_i^{k,j} - y_i \alpha_i^{k,i+1}\right). \tag{40}$$

## 4. The Modified Proposed Method

*4.1. Two Speeding-Up Strategies.* To avoid duplicate and invalid iterative process calculations, the two given conditions are not updated. If one of two conditions is met, then the algorithm skips this iteration that can significantly reduce the computational amount and accelerate the convergence speed.

*Condition 1.* If $\alpha_i^{k,i} = \alpha_i^{k,j} = 0$ or $\alpha_i^{k,i} = \alpha_i^{k,j} = C$, then the constrained conditions are not updated.

Due to the box constraints of (10), there will appear a lot of boundary points of coordinates ($\alpha_i^{k,i} = C$ or $\alpha_i^{k,i} = 0$) in computing process. If there are two coordinates ($\alpha_i^{k,i}$ and $\alpha_i^{k,i}$) as the value of 0 or $C$ in an iteration process, the analytical solution of the two subvariables updates coordinates without calculating. The reason is that the formula (17) guarantees $\alpha_i^{k,i} + \alpha_i^{k,j} = 0$ or $\alpha_i^{k,i} + \alpha_i^{k,j} = 2C$, while double restricted box constrained optimization $0 \le \alpha_i^{k,i} \le C$ if the result is 0 or $C$. Constrained conditions will be edited ultimately as 0 or $C$. The constrained conditions are not updated.

*Condition 2.* If projected gradient is 0, the constrained conditions are not updated.

*4.2. Dual Problem of Cost-Sensitive Support Vector Machine Using Randomized Dual Coordinate Descent Method and Its Complexity Analysis.* From the above algorithm derivation of view, solving of CSVM seems to have been successful, but the computational complexity of the solving process is also larger. Assume that the average value of nonzero feature of each sample is $\overline{n}$. Firstly, the computational complexity of the inner product matrix $Q_{ii} = y_i x_i^T x_i y_i$, $i = 1, 2, \ldots, l$ is $O(l\overline{n})$, but the process can be operated in advance and stored into memory. Secondly, the calculation of $(\alpha^{k,i}\overline{Q})_i = \sum_{s=1}^{l} Q_{i,s}\alpha_i^{k,s}$ takes the computational complexity $O(l\overline{n})$. The amount of calculation is very great when the data size is large. However, there is a linear relationship model CSVM:

$$w = \sum_{s=1}^{l} \alpha_i^{k,s} x_s y_s. \tag{41}$$

Thus, $(\alpha^{k,i}\overline{Q})_i$ is further simplified as follows:

$$(\alpha^{k,i}\overline{Q})_i = \sum_{s=1}^{l} Q_{i,s}\alpha_i^{k,s} = \sum_{s=1}^{l} \left(\alpha_i^{k,s} y_s x_s^T\right) x_s y_s = w^T x_i y_i. \tag{42}$$

Solving the corresponding formula (23) can be simplified as follows:

$$\begin{aligned} d_i &= \frac{2(\boldsymbol{\alpha}^{k,i}\overline{Q})_i - 1 - \left(2(\boldsymbol{\alpha}^{k,j}\overline{Q})_j - 1\right) y_i y_j}{2\left(Q_{ii} + Q_{jj}\right)} \\ &= \frac{2w^T x_i y_i - 1 - \left(2w^T x_j y_j - 1\right) y_i y_j}{2\left(Q_{ii} + Q_{jj}\right)}. \end{aligned} \tag{43}$$

As can be seen, computing $d_i$ with the complexity of $O(l\overline{n})$ becomes computing $w^T x_i y_i$ with the complexity of $O(\overline{n})$. Thus the calculation times reduce $l - 1$. However,

$$w = \sum_{s=1}^{l} \alpha_i^{k,s} x_s y_s, \tag{44}$$

where $w$ is the updated still computational complexity $O(l\overline{n})$. The amount of calculation can be reduced significantly when $w$ is updated by changing $\alpha_i^{k,s}$. Let $\alpha_i^{k,i}$, $\alpha_i^{k,j}$ be the values of the current selection; $\alpha_i^{k,i+1}$, $\alpha_i^{k,j+1}$ are updated values, which can be updated via a simple way:

$$w \longleftarrow w + \left(\alpha_i^{k,i} - \alpha_i^{k,i+1}\right) x_i y_i + \left(\alpha_i^{k,j} - \alpha_i^{k,j+1}\right) x_j y_j. \tag{45}$$

Its computational complexity only is $O(\overline{n})$. So, whether calculating $d_i$ or updating $w$, coordinate gradient computation complexity is $O(\overline{n})$, which is one of the coordinate gradient method rapid convergence speed reasons.

When assigned an initial value and constraints based on

$$\begin{aligned} \sum_{i=1}^{n} \alpha_i y_i &= 0, \\ 0 \le \alpha_i &\le \gamma C, \\ 0 \le \alpha_j &\le (1-\gamma) C, \end{aligned} \tag{46}$$

set the initial point

$$\boldsymbol{\alpha}^0 = [\ldots, \alpha_i, \ldots, \alpha_j, \ldots,]_{1\times l}^T, \tag{47}$$

where $\alpha_i = 1/t$, $\alpha_j = (1/(l-t))l, t$ are the total number of samples and the number of positive samples, respectively. Thus the weight vector $w$ of the original problem is obtained by optimizing Lagrangian multipliers $\boldsymbol{\alpha}$ of the dual problem.

*4.3. Description of Cost-Sensitive Support Vector Machine Using Randomized Dual Coordinate Descent Method.* Accelerated conditions are judged by Section 4.1. One chooses the coordinate optimized number $i, j$. We use formula (43) to calculate $d_i$, formulas (39) and (40) to update $\alpha_i^{k,i}$, $\alpha_i^{k,j}$, and the formula (45) to update $w$, respectively. We can see that inner iteration takes $O(\overline{n})$ effort. The computer memory is mainly used to store samples information $x_1, x_2, \ldots, x_l$ and each sample point and their inner products $Q_{ii} = y_i x_i^T x_i y_i$. Cost-sensitive support vector machine using dual randomized coordinate gradient descent algorithm is described as follows.

*Algorithm 3.* Cost-sensitive support vector machine using randomized dual coordinate descent algorithm (CSVM-RDCD).

Input: sample information $\{(x_i, y_i)\}_{i=1}^{l}$, $\boldsymbol{\alpha}^0$.

Output: $\boldsymbol{\alpha}^T w$.

Initialize $\boldsymbol{\alpha}^0$ and the corresponding $w = \sum_{i=1}^{l} \alpha_i x_i y_i$.

For $k = 1, 2, \ldots, T$ do

$$\boldsymbol{\alpha}^{k,1} \longleftarrow \boldsymbol{\alpha}^k. \tag{48}$$

*Step 1.* Randomly choose $i, j \in \{1, 2, \ldots, l\}$, $i \ne j$ and the corresponding $y_i$, $y_j$.

Table 1: Specification of the benchmark datasets. Number of ex. is the number of example data points. Number of feat. is the number of features. Ratio is the class imbalance ratio. Target specifies the target or positive class. Number is the number of the datasets.

| Dataset | Number of ex. | Number of feat. | Ratio | Positive class | Number |
|---|---|---|---|---|---|
| KDD99 (intrusion detection) | 5,209,460 | 42 | 1 : 4 | Normal | 1 |
| Web span | 350,000 | 254 | 1 : 2 | −1 | 2 |
| MNIST | 70,000 | 780 | 1 : 10 | 5 | 3 |
| Covertype | 581,012 | 54 | 1 : 211 | Cottonwood/Willow(4) | 4 |
| SIAM1 | 28,596 | 30,438 | 1 : 2000 | 1, 6, 7, 11 | 5 |
| SIAM11 | 28,596 | 30,438 | 1 : 716 | 11, 12 | 6 |

*Step 2.*

$$\begin{aligned}
&\text{If } \left(y_i = +1, y_j = -1\right)\\
&\left\{U_1 = \max\left(0, \alpha_i - \alpha_j\right);\right.\\
&\quad \left.V_1 = \min\left(\alpha_i - \alpha_j - (1-\gamma)C, \gamma C\right);\right\}\\
&\text{if } \left(y_i = +1, y_j = +1\right)\\
&\left\{U_2 = \max\left(0, \alpha_i + \alpha_j - \gamma C\right);\right.\\
&\quad \left.V_2 = \min\left(\alpha_i + \alpha_j, \gamma C\right);\right\}\\
&\text{if } \left(y_i = -1, y_j = -1\right)\\
&\left\{U_3 = \max\left(0, -\alpha_i - \alpha_j - (1-\gamma)C\right);\right.\\
&\quad \left.V_3 = \min\left(-\alpha_i - \alpha_j, (1-\gamma)C\right);\right\}\\
&\text{if } \left(y_i = -1, y_j = +1\right)\\
&\left\{U_4 = \max\left(0, -\alpha_i - \alpha_j\right);\right.\\
&\quad \left.V_4 = \min\left(\gamma C - \alpha_i - \alpha_j, (1-\gamma)C\right);\right\}.
\end{aligned} \tag{49}$$

*Step 3.*

$$\alpha_i^{\text{temp}} = \alpha_i^{k,i} + \frac{2w^T x_i y_i - 1 - \left(2w^T x_j y_j - 1\right) y_i y_j}{2\left(Q_{ii} + Q_{jj}\right)}. \tag{50}$$

*Step 4.*

$$\begin{aligned}
&\text{if } \left(y_i = 1, y_j = -1\right) m = 1;\\
&\text{if } \left(y_i = 1, y_j = 1\right) m = 2;\\
&\text{if } \left(y_i = -1, y_j = -1\right) m = 3;\\
&f\left(y_i = -1, y_j = +1\right) m = 4;
\end{aligned} \tag{51}$$

$$\alpha_i^{k,i+1} = \begin{cases} V_m, & \text{if } \alpha_i^{\text{temp}} > V_m \\ \alpha_i^{\text{temp}}, & \text{if } U_m < \alpha_i^{\text{temp}} < V_m \\ U_m, & \text{if } \alpha_i^{\text{temp}} < U_m. \end{cases}$$

*Step 5.*

$$\alpha_i^{k,j+1} = y_j\left\{y_i\alpha_i^{k,i} + y_j\alpha_j^{k,j} - y_i\alpha_i^{k,i+1}\right\}. \tag{52}$$

*Step 6.*

$$w \longleftarrow w + \left(\alpha_i^{k,i} - \alpha_i^{k,i+1}\right) x_i y_i + \left(\alpha_i^{k,j} - \alpha_i^{k,j+1}\right) x_j y_j. \tag{53}$$

Until A stopping condition is satisfied,

$$\boldsymbol{\alpha}^{k+1} \longleftarrow \boldsymbol{\alpha}^k. \tag{54}$$

End for.

## 5. Experiments and Analysis

*5.1. Experiment on Big Class-Imbalanced Benchmark Datasets Classification Problem.* In this section, we analyze the performance of the proposed cost-sensitive support vector machine using randomized dual coordinate descent method (CSVM-RDCD). We compare our implementation with the state-of-the-art cost-sensitive support vector. Three implementations of related cost-sensitive SVMs are compared. We proposed the cost-sensitive SVM using randomized dual coordinate descent method (CSVM-RDCD) by modifying the LibSVM [8, 9] source code. Eitrich and Lang [10] proposed parallel cost-sensitive support vector machine (PCSVM). [6] proposed cost-sensitive support vector machines (CSSVM).

Table 1 lists the statistics of data sets. KDD99, Web Spam, Covertype, MNIST, SIAM1, and SIAM11 are obtained from the following data website. *KDD99* is at http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. Web Spam is at http://www.cc.gatech.edu/projects/doi/WebbSpamCorpus.html. Covertype is at http://archive.ics.uci.edu/ml/datasets/Covertype. MNIST is at http://yann.lecun.com/exdb/mnist/. SIAM is at https://c3.nasa.gov/dashlink/resources/138/.

To evaluate the performance of CSVM-RDCD method, we use a stratified selection to split each dataset to 9/10 training and 1/10 testing. We briefly describe each set below. For each dataset we choose the class with the higher cost or fewer data points as the target or positive class. All multiclass datasets were converted to binary data sets. In particular, the binary datasets SIAM1 and SIAM2 are datasets which have been constructed from the same multiclass dataset but with different target class and different imbalance ratios.

Table 2: Cost-sensitive parameters of three algorithms using CSVM on 6 datasets.

| Dataset | $C\gamma$ | $C(1-\gamma)$ |
|---|---|---|
| KDD99 (intrusion detection) | 4 | 1 |
| Web span | 2 | 1 |
| MNIST | 10 | 1 |
| Covertype | 211 | 1 |
| SIAM1 | 2000 | 1 |
| SIAM11 | 716 | 1 |

Evaluation of the performance of the three algorithms using CSVM was 10-fold cross-validation tested average misclassification cost, the training time, and the recognition rate of the positive class (i.e., the recognition rate of the minority class).

Average misclassification cost (AMC) represents 10-fold cross-validation average misclassification cost on test datasets for related CSVMs, described as follows:

$$\text{AMC} = \frac{fp \times C\gamma + fn \times C(1-\gamma)}{N}, \tag{55}$$

where $fp$ and $fn$ represent the number of the positive examples misclassified as the negative examples in the test dataset and the number of the negative examples misclassified as the positive examples in the test data set, respectively. $C\gamma$ and $C(1-\gamma)$ denote the cost of the positive examples misclassified as the negative examples in the test data set and the cost of the negative examples misclassified as the positive examples in the test data set, respectively. $N$ denotes the number of test examples.

The recognition rate of the positive class is the number of classified positive classes and the number of the positive classes on testing dataset.

Training time in seconds is used to evaluate the convergence speeding of three algorithms using CSVM on the same computer.

Cost-sensitive parameters of three algorithms using CSVM are specified as the following Table 2. Cost-sensitive parameters $C\gamma$ and $C(1-\gamma)$ are valued according to the class ratio of datasets, namely, the class ratio of minority class and majority class.

Three datasets with relative class imbalance are examined. Namely, KDD99 (intrusion detection), Web span, and MNIST datasets are considered. Three datasets with severe class imbalance are examined. Namely, Covertype, SIAM1, and SIAM11 datasets are considered. The average misclassification cost comparison of three algorithms using CSVM is shown in Figure 1 for each of the datasets. The CSVM-RDCD algorithm outperforms the PCSVM and CSVM on all datasets.

The recognition rate of positive class comparison of three algorithms using CSVM is shown in Figure 2 for each of the datasets. The CSVM-RDCD algorithm outperforms the PCSVM and CSSVM on all datasets, surpasses the PCSVM on four datasets, and ties with the PCSVM on two datasets.

We examine large datasets with relative imbalance ratios and severe imbalance ratios to evaluate the convergence

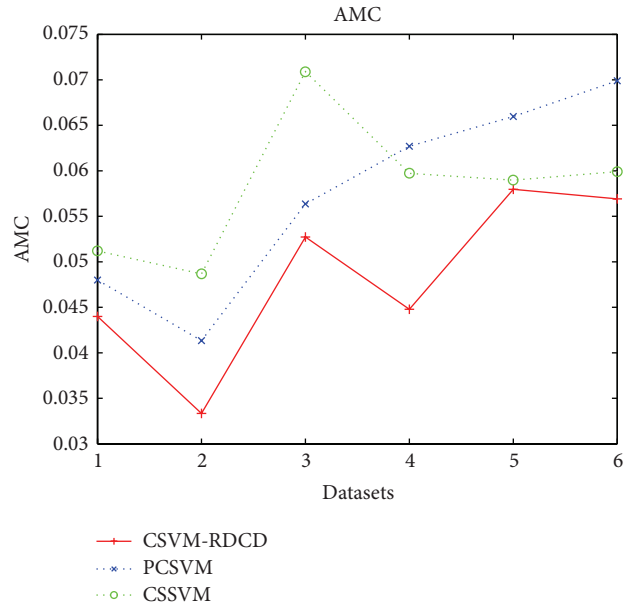


Figure 1: Average misclassification cost comparison of three algorithms using CSVM on 6 datasets.

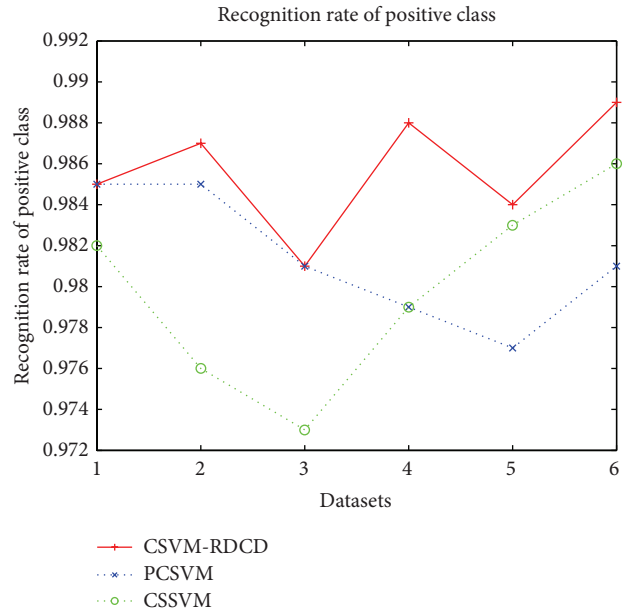


Figure 2: The positive class recognition rate comparison of three algorithms using CSVM.

speed of CSVM-RDCD algorithm. The training time comparison of three algorithms using CSVM is shown in Figure 3 for each of the datasets. The CSVM-RDCD algorithm outperforms the PCSVM and CSSVM on all datasets.

*5.2. Experiment on Real-World Big Class-Imbalanced Dataset Classification Problems.* In order to verify the effectiveness of the proposed algorithm CSVM-RDCD on real-world big class-imbalanced data classification problems, it was

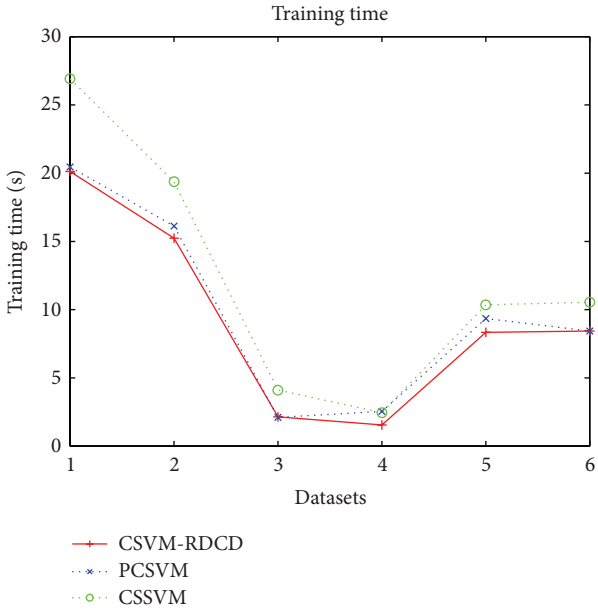


Figure 3: Training time (s) comparison of three algorithms using CSSVM.

Table 3: Specification of the benchmark datasets and cost-sensitive parameters on the dataset.

| Dataset | Fault diagnosis of wind turbine |
|---|---|
| Number of ex. | 20000 |
| Number of feat. | 56 |
| Ratio | 190 |
| Positive class | Local crack fault |
| $C\gamma$ | 190 |
| $C(1-\gamma)$ | 1 |

evaluated using the real vibration data measured in the wind turbine. The experimental data were from the SKF WindCon software and collected from a wind turbine gearbox type TF138-A [11]. The vibration signals were continuously acquired by an accelerometer mounted on the outer case of the gearbox.

All parameter settings for the dataset are listed in Table 3. The statistical results of the big class-imbalanced data problems that measure the quality of results (average misclassification cost, recognition rate of positive class, and training time) are listed in Table 4. From Table 4, it can be concluded that CSVM-RDCD is able to consistently achieve superior performance in the big class-imbalanced data classification problems.

Experimental results show that it is applicable to solve cost-sensitive SVM dual problem using randomized dual coordinate descent method on the large-scale experimental data sets. The proposed method can achieve superior performance in the average misclassification cost, recognition rate of positive class, and training time. Large-scale experimental data sets show that cost-sensitive support vector machines using randomized dual coordinate descent method run more efficiently than both PCSVM and CSSVM; especially randomized dual coordinate descent algorithm has advantage of training time on large-scale data sets. CSSVM needs to build complex whole gradient and kernel matrix $Q$ and needs to select the set of complex work in solving process of decomposition algorithm. Decomposition algorithm updates full uniform gradient information as a whole, the computational complexity $O(\overline{ln})$ for the full gradient update. PCSVM also has similar computational complexity. Randomized dual coordinate gradient method updates linearly the coordinates, its computational complexity as $O(\overline{n})$, which increases considerably the convergence speed of the proposed method.

Table 4: Classification performance of three algorithms using CSVM on the dataset.

| Name of algorithms | AMC | Recognition of positive class | Training time |
|---|---|---|---|
| CSVM-RDCD | 0.01825 | 98.5% | 5.43 s |
| PCSVM | 0.0334 | 97.6% | 6.15 s |
| CSSVM | 0.0379 | 97.2% | 7.34 s |

## 6. Conclusions

Randomized dual coordinate descentmethod (RDCD) is the optimization algorithm to update the global solution which is obtained by solving an analytical solution of the suboptimal problem. The RDCD method has the rapid convergence rate, which is mainly due to the following: (1) the subproblem has formal analytical solution, which is solved in solving process without complex numerical optimization; (2) the next component of RDCD method in solving process is updated on the basis of a previous component; compared with the full gradient information updated CSSVM method as a whole, the objective function of RDCD method can decline faster; (3) the single coordinate gradient calculation of RDCD method is simpler and easier than the full gradient calculation.

Randomized dual coordinate descent method is applied to cost-sensitive support vector machine, which expanded the scope of application of the randomized dual coordinate descent method. For large-scale class-imbalanced problem, a cost-sensitive SVM using randomized dual coordinate descent method is proposed. Experimental results and analysis show the effectiveness and feasibility of the proposed method.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

This work is partially supported by the National Science Fund for Distinguished Young Scholars of China (no. 61025015), the National Natural Science Foundation of China

(nos. 51305046 and 61304019), the Research Foundation of Education Bureau of Hunan Province, China (no. 12A007), Open Fund of Hunan Province University Key Laboratory of Bridge Engineering (Changsha University of Science and Technology), the Key Laboratory of Renewable Energy Electric-Technology of Hunan Province (Changsha University of Science and Technology), the Key Laboratory of Efficient and Clean Energy Utilization, College of Hunan Province, and the Introduction of Talent Fund of Changsha University of Science and Technology.

## References

[1] M. A. Davenport, "The 2nu-SVM: a cost-sensitive extension of the nu-SVM," Tech. Rep. TREE 0504, Department of Electrical and Computer Engineering, Rice University, 2005.

[2] M. Kim, "Large margin cost-sensitive learning of conditional random fields," *Pattern Recognition*, vol. 43, no. 10, pp. 3683–3692, 2010.

[3] Y.-J. Park, S.-H. Chun, and B.-C. Kim, "Cost-sensitive case-based reasoning using a genetic algorithm: application to medical diagnosis," *Artificial Intelligence in Medicine*, vol. 51, no. 2, pp. 133–145, 2011.

[4] J. Kim, K. Choi, G. Kim, and Y. Suh, "Classification cost: an empirical comparison among traditional classifier, Cost-Sensitive Classifier, and MetaCost," *Expert Systems with Applications*, vol. 39, no. 4, pp. 4013–4019, 2012.

[5] C.-Y. Yang, J.-S. Yang, and J.-J. Wang, "Margin calibration in SVM class-imbalanced learning," *Neurocomputing*, vol. 73, no. 1–3, pp. 397–411, 2009.

[6] H. Masnadi-Shirazi, N. Vasconcelos, and A. Iranmehr, "Cost-sensitive supportvector machines," http://arxiv.org/abs/1212.0975.

[7] Y. Artan, M. A. Haider, D. L. Langer et al., "Prostate cancer localization with multispectral MRI using cost-sensitive support vector machines and conditional random fields," *IEEE Transactions on Image Processing*, vol. 19, no. 9, pp. 2444–2455, 2010.

[8] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan, "A dual coordinate descent method for large-scale linear SVM," in *Proceedings of the 25th International Conference on Machine Learning*, pp. 408–415, July 2008.

[9] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: a library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[10] T. Eitrich and B. Lang, "Parallel cost-sensitive support vector machine software for classification," in *Proceedings of the Workshop from Computational Biophysics to Systems Biology*, NIC Series, pp. 141–144, John von Neumann Institute for Computing, Jülich, Germany, 2006.

[11] B. Tang, W. Liu, and T. Song, "Wind turbine fault diagnosis based on Morlet wavelet transformation and Wigner-Ville distribution," *Renewable Energy*, vol. 35, no. 12, pp. 2862–2866, 2010.