An Improved Online Multiclass Classification Algorithm Based on Confidence-Weighted

Ji HU†, Chenggang YAN†, Jiyong ZHANG††, Dongliang PENG†, Chengwei REN†, Nonmembers, and Shengying YANG††, Member

SUMMARY Online learning is a method which updates the model gradually and can modify and strengthen the previous model, so that the updated model can adapt to the new data without having to relearn all the data. However, the accuracy of the current online multiclass learning algorithm still has room for improvement, and the ability to produce sparse models is often not strong. In this paper, we propose a new Multiclass Truncated Gradient Confidence-Weighted online learning algorithm (MTGCW), which combine the Truncated Gradient algorithm and the Confidence-weighted algorithm to achieve higher learning performance. The experimental results demonstrate that the accuracy of MTGCW algorithm is always better than the original CW algorithm and other baseline methods. Based on these results, we applied our algorithm for phishing website recognition and image classification, and unexpectedly obtained encouraging experimental results. Thus, we have reasons to believe that our classification algorithm is clever at handling unstructured data which can promote the cognitive ability of computers to a certain extent.

key words: cognitive system, online learning, multiclass classification, streaming data, confidence-weighted

1. Introduction

Multiclass classification tasks are widely used in personal credit evaluation, depiction of user portrait, image classification and so on. These scenarios rely on streaming data for a better user experience and lower latency. Under the premise of big data, it is effective to use efficient online learning algorithms to process streaming data in these application scenarios. And no doubt all of these application scenarios require the highest possible accuracy. For these reasons, how to improve the accuracy of multiclass classification tasks is the main problem that multiclass classifiers need to solve. Thus, in recent years, many online multiclass learning algorithms were proposed and most of them were extended from binary classification tasks [1], such as Multiclass Perceptron [2], Multiclass Confidence-Weighted algorithm [3], Multiclass Passive-Aggressive algorithm and so on. Since the above multiclass classification algorithms are based on classic online learning algorithms, they inherit the disadvantages of difficulty in reducing the streaming data dimension and poor robustness. As the most advanced of these classic algorithms, Confidence-Weighted (CW) [4] algorithm has the advantages of relatively high classification accuracy and large margin. We hope to overcome its shortcomings while retaining its advantages so that both performance and feature selection capability can be improved.

In this paper, we extended the CW algorithm for multiclass classification and added a new operation to its update steps to enhance the feature selection capability of the model. After each weight update, our algorithm will check whether the gradient exceeds the threshold value. If the gradient value exceeds the threshold value, the gradient in that direction is truncated. Each truncation operation can effectively reduce the complexity of the model and increase the computing efficiency. In addition, we introduce a controllable parameter to weigh the accuracy and robustness, which make our algorithm perform better in different tasks.

The rest of this paper is organized as following: Sect. 2 reviews related work; Sect. 3 introduces previous work that most closely relates to our method; Sect. 4 proposes the Multiclass Truncated Gradient Confidence-Weighted algorithm and gives detailed operation steps; Sect. 5 conducts extensive experiments on our proposed algorithm and other state-of-the-art algorithms. Section 6 concludes this work.

2. Related Works

Online learning is a continuous training process in which input values are fed into the model in each round of training, and the model outputs prediction results based on the current parameters [5]. If the predicted class label is equal to the input class label value, the model will continue to be used for the next round of input values; If not, the model will suffer a loss and update to make better predictions for future data [21], [22].

Our proposed online learning for multiclass classification algorithm uses One-versus-Rest (OvR) strategy [6] and is related to a variety of basic online learning methods, including Perceptron algorithm [7], [8], CW learning, Soft Confidence-Weighted (SCW) algorithm [9], Passive-Aggressive (PA) algorithm [10], Online Gradient Descent (OGD) algorithm [11]–[13] and Truncated Gradient (TG) algorithm.

We will introduce some of these algorithms below.
2.1 One-versus-Rest Strategy

The OvR strategy is a way to solve multiclass classification problems by splitting them into multiple binary classification problems. This strategy involves training a unique binary classifier for each class, with all samples belonging to this category being positive and the rest negative. It requires that the two classifiers not only predict a class label, but also generate a real-valued confidence score for decision-making. In our method, we need the OvR strategy to generate classifiers to get the prediction result to calculate the suffered loss.

2.2 Perceptron Algorithm

The basic idea of Perceptron algorithm is to find a hyperplane \( \omega^T x + b \) in the sample space to divide the dataset into two categories, so the function used to determine the class labels is formulated as:

\[
\hat{y}_t = \text{sign}(\omega^T x + b) = \begin{cases} +1, & \omega^T x + b \geq 0 \\ -1, & \omega^T x + b < 0 \end{cases}
\]  

(1)

\( \omega \) is a column vector of weight parameter and \( b \) is the bias. We can fix \( b \) and update the parameter \( \omega \), then the weight adjustment of Perceptron learning is as follows:

\[
\Delta \omega = \eta(y_t - \hat{y}_t)x \\
\omega' = \omega + \Delta \omega
\]  

(2, 3)

\( \eta \) is learning rate and it is usually between 0 and 1.

When Perceptron is used for multiclass classification tasks, it is divided into three multiclass Perceptron algorithms according to different allocation strategies: max-score multiclass Perceptron, uniform multiclass Perceptron and proportion multiclass Perceptron. Here we only introduce max-score multiclass Perceptron. According to the suffered loss, the update rule is expressed as:

\[
\omega_{t+1,i} = \omega_{t,i} + \alpha_{t,i}x_i \\
\alpha_{t,i} = \begin{cases} 
-1 & \text{if } i = \arg \max_j \omega_{t,j}x_i \\
+1 & \text{if } i = y_t \\
0 & \text{otherwise}
\end{cases}
\]  

(4, 5)

Where \( \omega \) is a matrix, \( t \) represents the \( t \)-th round, \( i \)-th row in \( \omega \) is the linear classifier for the \( i \)-th label.

2.3 Online Gradient Descent Algorithm

OGD is an effective algorithm, which is simple and easy to operate, and is widely used in online learning. The algorithm flow is very simple, two steps are performed for each round of update, first perform a gradient descent on the current model and the loss function according to the data of the round. If the updated parameters exceed the definition domain, then project them back to the domain. Unlike the previous offline gradient descent, the OGD algorithm only uses the current data to calculate the gradient once to update, while offline gradient descent uses all the data to obtain the gradient to optimize the entire model parameters, so the advantage of the OGD algorithm is that the cost is small when used for multiclass classification tasks.

Here we only present the suffered loss of OGD and its update rule:

The hinge loss:

\[
\ell(\omega; (x, y)) = \max\{0, 1 - y(\omega^T x)\}
\]  

(6)

The update rule:

\[
\omega_{t+1,i} = \omega_{t,i} + \alpha_{t,i}x_i \\
\alpha_{t,i} = \begin{cases} 
-1/\sqrt{t} & \text{if } i = \arg \max_j \omega_{t,j}x_i \\
1/\sqrt{t} & \text{if } i = y_t \\
0 & \text{otherwise}
\end{cases}
\]  

(7, 8)

2.4 Passive-Aggressive Algorithm

The Passive-Aggressive algorithm is an online learning algorithm proposed by Koby Crammer in 2006. This algorithm idea is simple but has been proven to be superior to other learning algorithms in multiclass tasks. Similar to the Perceptron algorithm, a weight vector \( \omega \) is given and the loss function is based on the hinge loss function:

\[
\ell(\omega; (x, y)) = \max\{0, 1 - y(\omega^T x)\}
\]  

(9)

Where \( y(\omega^T x) \) is the margin.

And the optimization of the PA learning on round \( t \) is:

\[
\omega_{t+1} = \arg \min_{\omega} \frac{1}{2} \|\omega - \omega_t\|^2 \\
\text{s.t. } \ell(\omega; (x, y)) = 0
\]  

(10)

The above equation-constrained optimization problem has a closed-form update date rule:

\[
\omega_{t+1} = \omega_t + \tau_t x_i y_i, \text{ where } \tau_t = \frac{l_t}{\|x_t\|^2}
\]  

(11)

Further, introduce a parameter \( C \) to let PA be able to handle non-separable instances and more robust.

\[
\omega_{t+1} = \arg \min_{\omega} \frac{1}{2} \|\omega - \omega_t\|^2 + C\ell(\omega; (x_i, y_i))
\]  

(12)

Where \( C \) is a parameter to trade-off between passiveness and aggressiveness, higher \( C \) value yield stronger aggressiveness. Similarly, in multiclass cases, PA update rule can be extended as:

If \( \ell(\omega; (x_i, y_i)) > 0 \):

\[
\omega_{t+1,i} = \omega_{t,i} + \alpha_{t,i}x_i \\
\alpha_{t,i} = \begin{cases} 
-\min_{\omega} \frac{\ell(\omega; (x_i, y_i))}{2\|x_t\|^2} & \text{if } i = \arg \max_j \omega_{t,j}x_i \\
+\min_{\omega} \frac{\ell(\omega; (x_i, y_i))}{2\|x_t\|^2} & \text{if } i = y_t \\
0 & \text{otherwise}
\end{cases}
\]  

(13, 14)
3. Preliminaries

The CW algorithm has advanced classification performance due to the introduction of the probabilistic model. [24] The TG algorithm truncates the gradient within the threshold, which is an effective method to simplify the model. In this section, we will give a detailed introduction to these two algorithms we are based on.

3.1 Truncated Gradient Algorithm

In high-dimensional feature vectors and big datasets, it is very important that the model coefficients have good sparsity.

The truncated gradient algorithm was proposed by John Langford, Lihong Li and Tong Zhang in 2009 [14], the sparsity of the feature vector is well controlled and greatly improved the ability of feature selection and the interpretability at the same time by this algorithm. Thus, TG is often used for online learning to enhance learning performance and obtain sparse models.

3.2 Confidence-Weighted Algorithm

The online learning classifier can be divided into the first-generation perceptron algorithm and the second-generation online passive-aggressive learning algorithm according to the update mode. The CW algorithm can be considered as the third-generation online learning algorithm [15].

CW combines confidence with the probability distribution and assumes that \( \omega \) follows the Gaussian distribution. The sign of \( \omega^T x \) is used as the prediction result like Perceptron. The margin of an example \((x, y)\) is given by \( y(\omega^T x) \) and if the margin value is positive, the model predicts correctly on this simple. The absolute value of the margin is used as the confidence, so a larger confidence value indicates more accurate prediction results.

The optimization task of the algorithm is to update the weight distribution by minimizing the Kullback-Leibler divergence between the new weight distribution and the old weight while ensuring that the probability of a correct prediction for training instance is no smaller than the confidence hyperparameter \( \eta \). [25]

In Multiclass Confidence-weighted learning, the coefficients are updated similarly with the binary cases. Multiclass CW optimization problem has a closed-form update rule:

If \( \alpha_i > 0 \):

\[
\begin{align*}
\mu_{t+1} &= \mu_t + \alpha_i \Sigma_t \Delta \psi_i, \\
\Sigma_{t+1} &= \Sigma_t - \beta_i \Sigma_t \Delta \psi_i \Delta \psi_i^T \Sigma_t
\end{align*}
\]

(15) (16)

Where

\[
\begin{align*}
\Delta \psi_i &= \psi(x_t, y_t) - \psi(x_t, y^*_t) \\
\hat{y}_t &= \text{argmax}(\mu_t, x_t)
\end{align*}
\]

(17) (18)

\[
\alpha_i = \max\left\{0, \frac{-1 - 2\phi m_t + \sqrt{(1 + 2\phi m_t)^2 - 8\phi (m_t - \phi v_t)}}{4\phi v_t}\right\}
\]

(19)

\[
\beta_i = \frac{1}{1/(2\alpha_i \phi) + v_t}
\]

(20)

\[
v_t = \Delta \psi_i^T \Sigma_t \Delta \psi_i, m_t = \mu_t^T \Delta \psi_i, \phi = \Phi^{-1}(\eta)
\]

(21)

Experiments can prove that the introduction of the confidence can effectively induce online learning in multiclass classification tasks.

4. Proposed Methods

When dealing with multiclass classification problems, a common approach is to combine the OvR strategy with the above binary learning algorithm and modify update rules to accommodate multiclass cases. However, using only the OvR strategy to extend the binary cases will make the resulting multi-classification to be difficult to suffer loss to update the coefficients based on the prediction results as in the binary classification algorithm. Moreover, this will cause the generated multiple classifiers to inherit and amplify the shortcomings of the binary classification algorithm.

To address the limitations above, in this section, we propose a new online learning algorithm suitable for multiclass classification of streaming data, named Multiclass Truncated Gradient Confidence-Weighted (MTGCW), which aims to overcome the shortcomings of CW and further improve prediction accuracy and feature selection ability of the model. Like the previous transformation method, we also adopt the OvR strategy, but the difference is that we integrate the CW algorithm with the TG algorithm when generating a single binary classifier.

The CW algorithm itself has some disadvantages: (1) its updating strategy is very aggressive. When there is noise in the streaming data, the CW algorithm will greatly modify the parameters, resulting in a decrease in accuracy. So when used for multiclass classification, it will not obtain satisfactory performance. (2) The poor sparsity of model parameters will lead to the poor interpret-ability and performance of the model to some extent. To overcome these disadvantages we have done two aspects of work. (1) a parameter \( C \) similar to PA algorithm and SCW learning is introduced to adjust the aggressiveness and passivity. But the difference is that the parameter \( C \) can be fixed or dynamically changed in our algorithm. The following methods can be used to set \( C \): setting a fixed value before training; setting a series of candidate values, dynamically selecting the parameter according to the input data [16]. It should be noted that in this paper we use the first method, i.e., the fixed \( C \).

(2) Streaming data makes feature selection difficult, so we introduce the TG algorithm to truncate coefficients smaller than the threshold \( \theta \) to reduce the dimension of streaming data.

We assume that the coefficient follows the Gaussian
distribution with the mean vector $\mu_i$ and the covariance matrix $\Sigma_i$ same as CW.

First, we need to use the OvR strategy to generate classifiers. Binary classifications are extended to multiclass classifications, and of course the weight vector $\omega_{\text{col}}$ must be expanded into a weight matrix $\omega_{\text{row}}$, where $n$ is the total number of categories and $i$ is the number of attributes. Multiply $\omega_{\text{row}}$ and $x$ to get a column vector $\delta_{\text{col}}$. Each element of $\delta$ is the product of the row vector of matrix $\omega_{\text{row}}$ and $x$, so the $j$-th element of the column vector is the real-value confidence score of the current sample classified into the $j$-th class ($j$ is an integer between 1 and $n$). The above operation makes each row of $\omega_{\text{row}}$ a linear classifier labeled $j$. Thus, We select the maximum value of the element in $\delta$, and the row number of that element is used as the prediction result [11]. Then we need to use the predicted label and the real label to calculate the loss function.

In multiclass cases, we set a threshold $\eta$ to control the probability of the difference between the minimum score in all relevant classes and the maximum score in all irrelevant classes. So the constraint is:

$$P_r(x_i, y_i) = \arg\min \mu_i x_i, \quad s.t. = \arg\min \mu_i x_i.$$  

Here we only use the optimization of multiclass CW (MSCW) [17] as an example to introduce our work for its better robustness. [22]

$$(\mu_{t+1}, r_t, \Sigma_{t+1}) = \arg\min D_{KL}(N(\mu, \Sigma) || N(\mu_{t}, \Sigma) + D_{KL}(N(\mu, \Sigma) || N(\mu_{t}, \Sigma)) + C \ell^d(N(\mu, \mu_t, \Sigma); (x_t, y_t)) \quad (23)$$

where the loss function is:

$$\ell^d(N(\mu, \mu_t, \Sigma); (x_t, y_t)) = \max \left\{ 0, -1 - 2 \phi m + \sqrt{(1 + 2 \phi m)^2 - 8 \phi (m - \phi v) } \right\} \quad (24)$$

where

$$v_t = \Delta \psi_t^T \Sigma \Delta \psi_t, \quad m_t = \mu_t^T \Delta \psi_t, \quad \phi = \Phi^{-1}(\eta) \quad (25)$$

In the origin multiclass CW (MCW) algorithm, the constraint is:

$$s.t. \quad \ell^d(N(\mu, \mu_t, \Sigma); (x_t, y_t)) = 0 \quad (26)$$

We can see that in the optimization above, this formula is multiplied by the parameter $C$ and then added after the optimization formula, which highly reflects that the soft-margin classification can tolerate a small number of errors and the trade-off between keeping the previous information and updating the current information while minimizing the objective function. This can prevent fluctuation and effectively reduces the effects of noise. Therefore, the parameter $C$ in the objective function can also be regarded as a penalty parameter. The value of $C$ can be estimated before model training based on the cost of misclassification. A large value of $C$ increases the penalty for misclassification, and a small value of $C$ decreases the penalty for misclassification.

The closed-form solution of MSCW optimization problem can be expressed as:

$$\mu_{t+1, r_t} = \mu_{t, r_t} + \alpha_t y_t \Sigma_t x_t \quad (27)$$

$$\mu_{t+1, r_t} = \mu_{t, r_t} - \alpha_t y_t \Sigma_t x_t \quad (28)$$

$$\Sigma_{t+1} = \Sigma_t - \beta_t \Sigma_t x_t^T x_t \Sigma_t \quad (29)$$

The update rule of the coefficient:

$$\alpha_t = \min \left\{ C, \max  \left\{ 0, \frac{1}{2 v_t \psi} \left[ m_t \psi + \sqrt{m_t^2 \psi^2 - \left( m_t^2 \psi^2 + 2 v_t \psi \phi_t \right)^2} \right] \right\} \right\} \quad (30)$$

$$\beta_t = \frac{\alpha_t \phi}{\sqrt{2 m_t + \alpha_t m_t} \phi_t} \quad (31)$$

Where

$$u_t = \frac{1}{8} \left( - \alpha_t v_t \phi + \sqrt{\alpha_t^2 v_t^2 \phi^2 + 8 m_t} \right)^2 \quad (32)$$

$$v_t = x_t^T \Sigma_t x_t, \quad m_t = \mu_{t, r_t} x_t - \mu_{t, r_t} x_t \quad (33)$$

$$\phi = \Phi^{-1}(\eta), \quad \psi = 1 + \frac{\phi^2}{2} \quad (34)$$

Now that we have the update rules, then we will integrate the TG algorithm into the update steps. The function of truncation can be expressed as follows:

$$T_1(u_t, \alpha, \theta) = \begin{cases} \max (0, v_t - \alpha) & \text{if } v_t \in [0, \theta] \\ \min (0, v_t + \alpha) & \text{if } v_t \in [-\theta, 0] \\ v_t & \text{otherwise} \end{cases} \quad (35)$$

$$f(\omega_t) = T_1(\omega_t - \gamma F_1(\omega_t, \xi_t, y_t), \theta) \quad (36)$$

The truncation is not performed every round, but every K online steps, i.e., if $i/K$ is not an integer, $g_i = 0$; if $i/K$ is an integer, $g_i = K g$ where $g$ is a parameter greater than 0. The gravity parameter $g$ and the threshold parameter $\theta$ jointly control the sparsity of the model.

So every K round We bring the weight vector $\omega_{t+1}$ which has been updated by the rule above into the truncation function as below:

$$\omega_{t+1} = TG_{\alpha} \omega_{t+1}$$

$$= T(\omega_{t+1} - \gamma F_1(\omega_{t+1}, (x_{t+1}, y_{t+1})), y_g, \theta) \quad (37)$$

$$T_1(\omega_{t+1}, \alpha, \theta) = \begin{cases} \max (0, \omega_{t+1} - \alpha) & \text{if } \omega_{t+1} \in [0, \theta] \\ \min (0, \omega_{t+1} + \alpha) & \text{if } \omega_{t+1} \in [-\theta, 0] \\ \omega_{t+1} & \text{otherwise} \end{cases} \quad (38)$$

Where

$$l(\omega_{t+1}; (x_{t+1}, y_{t+1})) = \max (0, 1 - y_{t+1} (\omega_{t+1} T x)) \quad (39)$$

$\gamma$ is the learning rate, which controls the learning speed and the learning accuracy, usually between 0 and 1.

After truncating the gradient, the sparsity of the model
has achieved the expected progress. The MTGCW algorithm is be shown in Table 1.

5. Experiments and Analysis

5.1 Datasets and Compared Algorithms

The main task of multiclass learning is to minimize cumulative errors to obtain higher accuracy. Therefore, in order to test the performance of our proposed MTGCW algorithm, we carefully selected 7 data sets from different fields from the UC Irvine Machine Learning Repository and KEEL dataset repository. Iris, seeds, and car evaluation are all classic datasets, the former two are plant classification, and the last is simple hierarchical evaluation of cars. The wine dataset is the result of chemical analysis of wines from the same region of Italy but from three different varieties containing 13 ingredients. The glass dataset contains attributes about several glass types that can be used to drive industrial row vector. In addition, in order to speed up the convergence of the models, the gray-scale value of each pixel is anti-aliased into a standard length vector, we anti-aliased each picture into a standard 28*28 gray-scale image, and then pulled it into 784 dimensional row vector. In addition, in order to speed up the convergence of the models, the gray-scale value of each pixel of the image are normalized to the range of −1 to 1. Table 2 shows the detailed statistics of the datasets we used.

| Table 1 MTGCW Online learning algorithm |
|-----------------------------------------|
| Algorithm 1: MTGCW online learning algorithm (MTGCW) |
| **Input:** parameters $C > 0, q > 0, q > 0, K > 0$ |
| **1. Initialization:** $\mu_{i,k} = (0, \ldots, 0), \Sigma_i = I.$ |
| **2. for** $t = 1, \ldots, T$ **do** |
| **3. Receive an example** $x_t \in R^d,$ |
| **4. Make prediction:** $\hat{y}_t = \arg\max_{y} \{\mu_{i,y} \cdot x_t, r \in [1, k] \};$ |
| **5. Receive true label** $y_t;$ |
| **6. Suffer loss** $\ell^y (\mu_{i,y}, \Sigma_i) \cdot (y_t, y_t);$ |
| **7. if** $\ell^y (\mu_{i,y}, \Sigma_i) \cdot (y_t, y_t) > 0$ **then** |
| $\alpha_t = \min \left\{ C, \max \left\{ 0, \frac{1}{\sqrt{N \cdot \sigma_{th}}} \left( -m_{vq} + \sqrt{m_{vq}^2 - m_{vq}^2 + 2\sigma_{th}} \right) \right\} \right\}$ |
| $\beta_t = \frac{\alpha_t \sigma_{th}}{\sqrt{N \cdot \sigma_{th}}} \phi \left( \frac{\alpha_t \sigma_{th}}{\sqrt{N \cdot \sigma_{th}}} \phi \right)$ |
| $\gamma_t = \frac{1}{2} \left( -\alpha_t \sigma_{th} + \sqrt{\alpha_t^2 \sigma_{th}^2 + 8\sigma_{th}} \right)^2$ |
| $\mu_{i,y_t} = \mu_{i,y_t} + \alpha_t \gamma_t x_t$ |
| $\Sigma_{i+1} = \Sigma_i - \beta_t \Sigma_i x_t x_t^T$ |
| **end** |
| **15. end** |
| **end** |

| MTGCW Online learning algorithm |
|---------------------------------|
| **Input:** parameters $C > 0, q > 0, q > 0, K > 0$ |
| **1. Initialization:** $\mu_{i,k} = (0, \ldots, 0), \Sigma_i = I.$ |
| **2. for** $t = 1, \ldots, T$ **do** |
| **3. Receive an example** $x_t \in R^d,$ |
| **4. Make prediction:** $\hat{y}_t = \arg\max_{y} \{\mu_{i,y} \cdot x_t, r \in [1, k] \};$ |
| **5. Receive true label** $y_t;$ |
| **6. Suffer loss** $\ell^y (\mu_{i,y}, \Sigma_i) \cdot (y_t, y_t);$ |
| **7. if** $\ell^y (\mu_{i,y}, \Sigma_i) \cdot (y_t, y_t) > 0$ **then** |
| $\alpha_t = \min \left\{ C, \max \left\{ 0, \frac{1}{\sqrt{N \cdot \sigma_{th}}} \left( -m_{vq} + \sqrt{m_{vq}^2 - m_{vq}^2 + 2\sigma_{th}} \right) \right\} \right\}$ |
| $\beta_t = \frac{\alpha_t \sigma_{th}}{\sqrt{N \cdot \sigma_{th}}} \phi \left( \frac{\alpha_t \sigma_{th}}{\sqrt{N \cdot \sigma_{th}}} \phi \right)$ |
| $\gamma_t = \frac{1}{2} \left( -\alpha_t \sigma_{th} + \sqrt{\alpha_t^2 \sigma_{th}^2 + 8\sigma_{th}} \right)^2$ |
| $\mu_{i,y_t} = \mu_{i,y_t} + \alpha_t \gamma_t x_t$ |
| $\Sigma_{i+1} = \Sigma_i - \beta_t \Sigma_i x_t x_t^T$ |
| **end** |
| **15. end** |
| **end** |

has achieved the expected progress. The MTGCW algorithm is be shown in Table 1.

5. Experiments and Analysis

5.1 Datasets and Compared Algorithms

The main task of multiclass learning is to minimize cumulative errors to obtain higher accuracy. Therefore, in order to test the performance of our proposed MTGCW algorithm, we carefully selected 7 data sets from different fields from the UC Irvine Machine Learning Repository and KEEL dataset repository. Iris, seeds, and car evaluation are all classic datasets, the former two are plant classification, and the last is simple hierarchical evaluation of cars. The wine dataset is the result of chemical analysis of wines from the same region of Italy but from three different varieties containing 13 ingredients. The glass dataset contains attributes about several glass types that can be used to drive criminological investigations. The thyroid dataset detects whether a given patient is normal or has hyperthyroidism or hypothyroidism to tests the performance level of each algorithm in medical testing. The final robot navigation dataset includes sensor data from the robot, and the algorithms need to analyze this data to determine the robot’s motion. Website phishing is an important issue in e-commerce and other industries, and it involves payment and transaction security. A website can be divided into Legitimate, Suspicious, and Phishy according to the phishing detection. There-
The table, when $\eta$ the MTGCW algorithm as an example. As we can see in the range of parameter $C$ is $\{2^{-4}, 2^{-3}, \ldots, 2^{3}, 2^{4}\}$ and parameter $\eta$ is $\{0.55, 0.60, \ldots, 0.90, 0.95\}$ [20].

Table 3 lists the process of selecting the parameter $\eta$ of the MTGCW algorithm as an example. As we can see in the table, when $\eta$ is 0.900, the mistake rate is the smallest of all rounds, so the best value of $\eta$ we selected has been bolded in the table, which is 0.900. After determining the best parameters, 20 randomly arranged sequences are selected from the training set, then each algorithm uses these sequences to train 20 times, and finally takes the average of these training results as the final result. Three indicators are used to measure the performance of the algorithm: online cultivate mistake rate, number of updates and running time cost. The online cultivate mistake rate is calculated by dividing the number of misclassifications in the sequence by the total number, which is highly related to prediction accuracy, is our most concerned indicator. All the experiments were implemented in Matlab and run on a regular PC.

5.2 Experimental Results and Analysis

Table 4 summarizes the results of our empirical evaluation of cumulative performance of the proposed MTGCW and other multiclass learning methods. The bold data in the table is the best performance of this indicator in this data set.

From the table, It can be clearly seen that the proposed MTGCW algorithm achieve the best performance in terms of mistake rate in all datasets. We found that when the dataset is relatively small, like iris, wine, glass, thyroid and seeds, the accuracy of the MTGCW algorithm can far surpass other algorithms except MSCW, and slightly exceed SCW. However, when the dataset is large like robotnavigation, car evaluation and website phishing, the accuracy of the MTGCW algorithm can outperform all compared algorithms including MSCW. Moreover, we can see from the last row of Table 4 that MTGCW significantly outperform any other 6 models in online image classification tasks, its performance on mistake rate is up to 22.5% higher than the strongest baseline method in masked faces detection, which means that when handling the number of samples in the mask detection dataset reaches 10000, our approach will be up to about 900 or more cases more accurate than the state-of-the-art algorithms. These results validates how effective our approach is and we have reason to believe that when the dataset is much larger, the performance of our model will be further improved. We also believe that if our algorithm is used to detect whether people going to public places wear masks during the epidemic, it will definitely be able to achieve superior performance and make a great contribution.

Further, MTGCW also performed well on the number of updates, which can reach the best of all in some datasets. However, the online cumulative time of MTGCW is sometimes more than some other algorithms. This is because we have added the operation of truncating the gradient every K rounds to our proposed algorithm, which increases the time consumption to a certain extent. Besides, we can see from Table 4, in the first few datasets, the standard deviation of MTGCW’s mistake rate is slightly higher than other algorithms. We believe that this is due to the introduction of the TG algorithm, which results in the convergence of MTGCW being slightly worse than other algorithms. However, this problem disappeared in the next few large datasets in the experiment, because the convergence is somehow related to the size of the dataset, i.e., large data sets often lead to better convergence. So when MTGCW is applied to streaming big data, this disadvantage of MTGCW can be ignored. By examining the overall mistakes, we also found that added “confidence” property algorithms like MCW, MSCW and MTGCW can always outperform other classic algorithms such as MPerceptron, etc. Moreover, Fig. 1 shows the online results of 6 algorithms on 10 datasets of different sizes and domains. And this verifies that the addition of the “confidence” property has a great promotion effect on model training, and the introduction of TG operation makes the performance even higher again.
| Algorithm  | Mistakes | Updates | Time(s) | Mistakes | Updates | Time(s) |
|-----------|----------|---------|---------|----------|---------|---------|
| M_Perceptron | 0.393±0.041 | 59.0±6.1 | 0.008±0.001 | 0.633±0.030 | 112.8±5.3 | 0.011±0.002 |
| M_OGD | 0.345±0.026 | 57.2±10.2 | 0.008±0.001 | 0.612±0.038 | 109.1±6.5 | 0.011±0.001 |
| M_PA | 0.402±0.027 | 113.8±5.7 | 0.010±0.002 | 0.653±0.028 | 150.2±5.2 | 0.012±0.001 |
| M_CW | 0.122±0.013 | 33.9±4.0 | 0.009±0.001 | 0.386±0.135 | 69.3±23.9 | 0.013±0.002 |
| M_IELLIP | 0.131±0.015 | 38.3±3.7 | 0.010±0.002 | 0.229±0.022 | 60.9±4.3 | 0.013±0.001 |
| M_TGCW | 0.120±0.019 | 47.5±5.3 | 0.011±0.002 | 0.216±0.068 | 47.5±10.5 | 0.014±0.001 |

| Algorithm  | Mistakes | Updates | Time(s) | Mistakes | Updates | Time(s) |
|-----------|----------|---------|---------|----------|---------|---------|
| M_Perceptron | 0.598±0.023 | 128.0±25.0 | 0.011±0.001 | 0.411±0.027 | 88.3±5.8 | 0.011±0.001 |
| M_OGD | 0.542±0.023 | 183.7±5.2 | 0.012±0.002 | 0.374±0.029 | 81.0±6.3 | 0.010±0.001 |
| M_PA | 0.590±0.033 | 189.9±5.7 | 0.012±0.001 | 0.449±0.028 | 148.2±5.8 | 0.012±0.001 |
| M_CW | 0.515±0.026 | 160.3±4.9 | 0.016±0.001 | 0.097±0.012 | 38.6±4.4 | 0.011±0.001 |
| M_IELLIP | 0.471±0.028 | 158.6±5.7 | 0.016±0.001 | 0.105±0.010 | 48.5±3.8 | 0.012±0.001 |
| M_TGCW | 0.469±0.040 | 160.2±5.8 | 0.019±0.001 | 0.091±0.011 | 39.7±4.6 | 0.014±0.001 |

| Algorithm  | Mistakes | Updates | Time(s) | Mistakes | Updates | Time(s) |
|-----------|----------|---------|---------|----------|---------|---------|
| M_Perceptron | 0.579±0.031 | 121.5±6.5 | 0.011±0.001 | 0.451±0.005 | 2462.9±25.0 | 0.159±0.016 |
| M_OGD | 0.519±0.034 | 111.2±7.0 | 0.012±0.001 | 0.400±0.006 | 2543.3±37.6 | 0.160±0.011 |
| M_PA | 0.619±0.025 | 181.2±5.2 | 0.012±0.001 | 0.451±0.005 | 4153.8±32.7 | 0.169±0.010 |
| M_CW | 0.161±0.066 | 48.1±10.8 | 0.012±0.001 | 0.424±0.005 | 3552.8±37.0 | 0.223±0.022 |
| M_IELLIP | 0.157±0.016 | 57.0±4.2 | 0.013±0.001 | 0.317±0.0056 | 2130.7±103.4 | 0.196±0.014 |
| M_TGCW | 0.141±0.026 | 50.1±4.1 | 0.014±0.001 | 0.299±0.0045 | 2464.6±66.2 | 0.231±0.021 |

| Algorithm  | Mistakes | Updates | Time(s) | Mistakes | Updates | Time(s) |
|-----------|----------|---------|---------|----------|---------|---------|
| M_Perceptron | 0.300±0.009 | 519.1±14.7 | 0.054±0.005 | 0.264±0.006 | 357.1±7.6 | 0.044±0.003 |
| M_OGD | 0.257±0.007 | 564.9±16.5 | 0.056±0.006 | 0.196±0.006 | 466.4±12.4 | 0.046±0.003 |
| M_PA | 0.286±0.008 | 919.8±11.9 | 0.057±0.006 | 0.258±0.006 | 710.7±19.5 | 0.049±0.006 |
| M_CW | 0.266±0.007 | 715.5±11.7 | 0.066±0.007 | 0.269±0.006 | 599.2±14.5 | 0.057±0.005 |
| M_IELLIP | 0.206±0.006 | 475.6±24.4 | 0.063±0.007 | 0.204±0.005 | 399.0±20.3 | 0.053±0.004 |
| M_TGCW | 0.201±0.005 | 514.8±12.6 | 0.073±0.007 | 0.189±0.006 | 366.6±10.7 | 0.059±0.004 |

| Algorithm  | Mistakes | Updates | Time(s) | Mistakes | Updates | Time(s) |
|-----------|----------|---------|---------|----------|---------|---------|
| M_Perceptron | 0.420±0.004 | 1711.6±15.5 | 0.310±0.080 | 0.225±0.002 | 2251.3±22.2 | 0.900±0.166 |
| M_OGD | 0.385±0.006 | 1775.0±21.3 | 0.299±0.032 | 0.167±0.002 | 1743.2±23.6 | 0.799±0.104 |
| M_PA | 0.422±0.006 | 2973.7±23.9 | 0.315±0.035 | 0.189±0.002 | 5681.2±40.4 | 0.859±0.081 |
| M_CW | 0.414±0.005 | 2890.2±18.4 | 34.65±20.63 | 0.134±0.001 | 3103.7±25.6 | 42.79±2.687 |
| M_IELLIP | 0.447±0.006 | 2746.0±24.6 | 35.32±1.501 | 0.137±0.002 | 3052.9±30.6 | 42.83±4.851 |
| M_TGCW | 0.306±0.003 | 2220.6±26.5 | 28.63±1.413 | 0.117±0.003 | 2891.1±54.1 | 39.36±1.787 |
Fig. 1  Evaluation of cumulative performance of the proposed MTGCW and other algorithm
6. Conclusion

This paper investigates how to enhance the ability of feature selection of models in online multiclass learning, with the aim of simultaneously improving the interpretability and performance of models. At the same time, we pointed out some shortcomings of the CW algorithm: (1) weak anti-noise capability (2) poor sparsity of the coefficient. In this regard, we proposed the MTG CW algorithm based on CW which introduced the parameter C to enhance the robustness, and the truncated gradient operation to reduce the dimension of streaming data. The experimental results show that MTG CW is quite effective in multiclass classification tasks for streaming data and is a state-of-the-art online multiclass learning algorithm. Further, our algorithm achieved exciting experimental results in the application areas of phishing website recognition and image classification, especially in the face mask detection. This means that compared with baseline algorithms, our algorithm has a significant improvement in the ability of analyzing unstructured data, such as images, which has a great effect on the improvement of computer visual cognition, i.e., our algorithm is smarter than others in terms of vision. But we also point out that although we use the truncated gradient algorithm to reduce principal component features, this also leads to an increase in time cost. Future work can be extended to how to reduce the computation time of MTG CW or how to tackle the problem of improving the feature selection ability of online learning algorithms.

References

[1] M. Aly, “Survey on multiclass classification methods,” Neural Netw., vol.19, pp.1–9, 2005.
[2] K. Crammer and Y. Singer, “Ultraconservative online algorithms for multiclass problems,” Journal of Machine Learning Research, vol.3, pp.951–991, 2003.
[3] K. Crammer, M. Dredze, and A. Kulesza, “Multi-class confidence weighted algorithms,” Proc. 2009 Conference on Empirical Methods in Natural Language Processing, vol.2, Association for Computing Machinery, pp.496–504, 2009.
[4] M. Dredze, K. Crammer, and F. Pereira, “Confidence-weighted linear classification,” Proc. 25th international conference on Machine learning, pp.264–271, 2008.
[5] S. Shalev-Shwartz, “Online learning and online convex optimization,” Foundations and Trends® in Machine Learning, vol.4, no.2, pp.107–194, 2012.
[6] D.M.J. Tax and R.P.W. Duin, “Using two-class classifiers for multiclass classification,” Object recognition supported by user interaction for service robots, IEEE, pp.124–127, 2002.
[7] I. Stephen, “Perceptron-based learning algorithms,” IEEE Trans. Neural Netw., vol.50, no.2, p.179, 1990.
[8] Y. Freund and R.E. Schapire, “Large margin classification using the perceptron algorithm,” Machine Learning, vol.37, no.3, pp.277–296, 1999.
[9] J. Wang, P. Zhao, and S.C.H. Hoi, “Exact soft confidence-weighted learning,” arXiv preprint arXiv:1206.4612, 2012.
[10] K. Crammer, O. Dekel, J. Keshet, et al., “Online passive-aggressive algorithms,” Journal of Machine Learning Research, vol.7, pp.551–585, March 2006.
[11] Y. Ying and M. Pontil, “Online gradient descent learning algorithms,” Foundations of Computational Mathematics, vol.8, no.5, pp.561–596, 2008.
[12] M. Zinkevich, “Online convex programming and generalized infinitesimal gradient ascent,” Proc. 20th International Conference on Machine Learning (ICML-03), pp.928–936, 2003.
[13] S. Smale and Y. Yao, “Online learning algorithms,” Foundations of Computational Mathematics, vol.6, no.2, pp.145–170, 2006.
[14] J. Langford, L. Li, and T. Zhang, “Sparse online learning via truncated gradient,” Journal of Machine Learning Research, vol.10, pp.777–801, March 2009.
[15] J. Liu, G. Chi, and X. Luo, “Online Classification Algorithm Based on Confidence-weighted Learning,” Computer Engineering, vol.38, no.9, pp.180–182, 2012.
[16] J. Hu, C. Yan, X. Liu, J. Zhang, D. Peng, and Y. Yang, “Truncated Gradient Confidence-Weighted Based Online Learning for Imbalance Streaming Data,” 2019 IEEE International Conference on Multimedia and Expo (ICME), IEEE, pp.133–138, 2019.
[17] J. Wang, P. Zhao, and S.C.H. Hoi, “Soft confidence-weighted learning,” ACM Transactions on Intelligent Systems and Technology (TIST), vol.8, no.1, pp.1–32, 2016.
[18] N. Abdelhamid, A. Ayesh, and F. Thabet, “Phishing detection based associative classification data mining,” Expert Systems with Applications, vol.41, no.13, pp.5948–5959, 2014.
[19] S. Ge, J. Li, Q. Ye, and Z. Luo, “Detecting Masked Faces in the Wild with LLE-CNNs,” 2017 IEEE Conference on Computer Vision and Pattern Recognition, pp.426–434, 2017. 10.1109/CVPR.2017.53.
[20] S.C.H. Hoi, J. Wang, and P. Zhao. "Libol: A library for online learning algorithms," Journal of Machine Learning Research, vol.15, no.1, 2014.
[21] A. Gepperth and B. Hammer, “Incremental learning algorithms and applications,” 2016 European Symposium on Artificial Neural Networks (ESANN), Bruges, Belgium, 2016.
[22] GB/T 7714 P. Yang, P. Zhao, J. Zhou, and X. Gao, “Confidence weighted multitask learning,” Proc. AAAI Conference on Artificial Intelligence, vol.33, pp.5636–5643, 2019.
[23] N.M. Jagirdar, “Online Machine Learning Algorithms Review and Comparison in Healthcare,” Master’s Thesis, University of Tennessee, 2018.
Appendix:

| Symbol | Specific meaning |
|--------|------------------|
| $\omega$ | Weight vector |
| $x$ | Input vector |
| $b$ | Bias |
| $y_t$ | True label |
| $\hat{y}_t$ | Predicted label |
| $\gamma, \eta$ | Learning rate |
| $\mathcal{L}(\cdot)$ | The loss of $\cdot$ |
| $\mathcal{C}$ | The parameter controls the passiveness of $\omega$ |
| $\sim \mathcal{N}(\mu, \Sigma)$ | $\omega$ follows a normal distribution with mean $\mu$ and variance $\Sigma$ |
| $g$ | The gravity parameter |
| $\theta$ | The threshold parameter |
| $D_{KL}(\cdot||\cdot)$ | The Kullback-Leibler divergence |
| $\ast$ | between $\cdot$ and $\cdot$ |

Ji Hu is a lecturer, PhD candidate. He received the M.S. degree in Electronics Science and Technology Major from Hangzhou Dianzi University in 2004. He visited McMaster University in Canada as a visiting scholar in 2016 and he is currently pursuing the Ph.D. degree in intelligent information processing at Hangzhou Dianzi University. His research interest includes Artificial Intelligence Tools and Application, Automatic Control, Computer Vision and Speech Understanding, Intelligent System Architecture, Deep Neural Networks and Image processing.

Chenggang Yan is a Professor, PhD supervisor. He received the PhD degree in University of Science and Completed postdoctoral research at Tsinghua University. His main research directions include machine learning, pattern recognition, image processing, computer vision, computer graphics, medical image processing, biological information processing.

Jiyong Zhang is a Professor, PhD supervisor. He graduated from the Computer Science Department of Tsinghua University with a bachelor’s degree in 1999, graduated from Tsinghua University with a master’s degree in 2001, and graduated from the Swiss Federal Institute of Technology Lausanne (EPFL) with a PhD degree in 2008. He has long been committed to technical research in the fields of artificial intelligence, big data, cloud computing, and intelligent human-computer interaction.

Dongliang Peng is a Professor, PhD supervisor. He graduated from Zhejiang University with a doctorate degree in engineering majoring in control science and engineering in 2003, visited and studied at George Mason University from 2012 to 2013. He engaged in the research of multi-sensor information fusion technology, detection and estimation.

Chengwei Ren is an undergraduate majoring in Intelligence Science and Technology at the School of Automation, Hangzhou Dianzi University. His research interests are class imbalance and online classification problem in machine learning and feature extraction problem in deep learning.

Shengying Yang is a PhD, lecturer. He graduated with a PhD in Electronic Science and Technology from Hangzhou Dianzi University in 2020. He mainly engaged in research on big data analysis and power electronic device design.