An integrated classification model for incremental learning

Ji Hu¹ · Chenggang Yan¹ · Xin Liu¹ · Zhiyuan Li¹ · Chengwei Ren¹ · Jiyong Zhang¹ · Dongliang Peng¹ · Yi Yang²

Received: 18 May 2020 / Revised: 25 August 2020 / Accepted: 7 October 2020 /

Published online: 21 October 2020

© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract

Incremental Learning is a particular form of machine learning that enables a model to be modified incrementally, when new data becomes available. In this way, the model can adapt to the new data without the lengthy and time-consuming process required for complete model re-training. However, existing incremental learning methods face two significant problems: 1) noise in the classification sample data, 2) poor accuracy of modern classification algorithms when applied to modern classification problems. In order to deal with these issues, this paper proposes an integrated classification model, known as a Pre-trained Truncated Gradient Confidence-weighted (Pt-TGCW) model. Since the pre-trained model can extract and transform image information into a feature vector, the integrated model also shows its advantages in the field of image classification. Experimental results on ten datasets demonstrate that the proposed method outperform the original counterparts.

Keywords Incremental learning · Transfer learning · Confidence weight · Image classification · Masked-face dataset

1 Introduction

Classification tasks are widely used in image classification, personal credit evaluation, depiction of user portrait, and so on. These scenarios rely on streaming data to ensure lower latency. Under the premise of big data, it is necessary to use efficient incremental learning algorithms to process streaming data in these application scenarios. Clearly, these application scenarios require the highest possible accuracy. For these reasons, improving the accuracy of classification algorithms is an important problem [10] that must be addressed.

✉ Ji Hu
huji@hdu.edu.cn

At present, there are two methods to solve this problem, namely batch learning [8] and Online learning [19]. In recent years, the emergence of a large amount of data has exceeded the size of memory, and online learning algorithms have attracted attention in the field of machine learning. Many online incremental learning [18] algorithms have been proposed, such as the Perceptron algorithm [14], the Passive Aggressive algorithm (PA) [1, 9], the Online Gradient Descent algorithm (OGD) [3], the Stochastic Gradient Descent algorithm (SGD) [2], the Truncated Gradient algorithm (TG) [3], the Weight Adaptive Regularization algorithm [5, 13], and the Confidence-Weighted algorithm (CW) [4, 6, 7].

However, some of these, such as the Perceptron and the OGD algorithms, effectively add noise to the model during each update, resulting in the inability to converge, or low and unstable convergence efficiency. Other algorithms cannot produce a sparse solution while maintaining convergence speed. These problems greatly increase the complexity of prediction and affect the classification accuracy of the algorithm.

In order to deal with the existing problems, this paper proposes an integrated classification model for incremental learning. This method consists of two parts: a pre-trained (Pt) model and a novel Truncated Gradient Confidence-weighted online classification model (TGCW). The pre-trained model is trained from the existing data samples in the neural network, and uses the transfer learning theory to use the pre-trained model to extract the feature vectors of all data to eliminate the noise from the data itself. Then, the feature vectors extracted by the pre-trained model are sent to the novel online classification model for incremental learning – this enables superior classification results. Compared with the baseline algorithm, our method combines the existing deep learning and incremental learning methods, making better use of the advantages of neural networks in data feature extraction and the continuous learning ability facilitated by incremental learning. Through the combination of these two methods the performance of the original algorithm is widely improved. Experiments have shown, conclusively, that our method has higher classification accuracy and faster convergence speed.

The remainder of this paper is organized as follows. Section 2 reviews other online learning algorithms related to our work. Section 3 proposes the Integrated classification learning framework, describing in detail the required steps. Section 4 conducts extensive experiments on our proposed algorithm and other state-of-the-art algorithms. Section 5 concludes this work.

2 Related work

With the development of machine learning theory, many algorithms to deal with classification problems have been developed. Most of these algorithms are based on batch learning, which relies on the assumption that all data samples can be obtained before training. The disadvantage of this kind of algorithm is that the learning stops after training on the existing data samples. For example, a deep neural network (DNN) [15, 25] uses a multilayer artificial neural network between input and output, and finds a mathematical operation that can minimize the loss function by adjusting the weights of the network. A DNN has the ability to capture complex nonlinear behavior, however it is easy to over fit and it typically requires a significant amount of training data. Also, DNNs require the training dataset to contain sufficient information that they can be applied to as-before-unseen data, post model training, while still obtain reasonable results.

In order to cope with the continuous growth of massive data, various online incremental learning algorithms have been proposed, such as active learning [26]. 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 [16]. If the predicted classification result agrees with the input classification, the model will continue to be used for the next round of input values. On the other hand, if the input classification differs from that predicted by the model, an update will ensue in an attempt to make improved predictions for future data. Incremental learning means that a model can acquire new knowledge from new samples, while at the same time retain most of the previously learned knowledge – similar to the human learning process: people are exposed to new information every day, in a step by step fashion, and will ideally extract the key learnings, combining them with their prior understanding, thus improving their overall knowledge.

The Perceptron is a linear classification algorithm known as a threshold function, which is shown below. The basic idea of the Perceptron algorithm is to find a hyper plane $\omega^T \mathbf{x} + b$ in the sample space to divide the dataset into two categories. Therefore, a suitable function used to determine the class labels may be formulated as:

$$y_t^\wedge = \text{sign}(\omega^T \mathbf{x} + b) = \begin{cases} +1, \omega^T \mathbf{x} + b \geq 0 \\ -1, \omega^T \mathbf{x} + b < 0 \end{cases} \quad (1)$$

where ω is a column vector of weight parameters and b is the bias. We can fix b and update the parameter ω , then the weight adjustment of the Perceptron learning algorithm proceeds as follows:

$$\begin{aligned} \Delta \omega &= \eta(y^t - y_t^\wedge) \mathbf{x} \\ \omega^* &= \omega + \Delta \omega \end{aligned} \quad (2)$$

where η represents the learning rate, usually between 0 and 1. This is the mechanism by which the Perceptron algorithm updates the model based on the prediction error.

Deviation causes the decision boundary to deviate from the origin and does not depend on any input value. If the training set is linearly separable, the perceptron convergence can be guaranteed. In addition, there is an upper limit for the number of times the perceptron adjusts its weight during training.

In addition to this, a wide range of algorithms based on convex optimization have been proposed, including the PA and OGD algorithms. The core of PA algorithm is based on the samples of support vector machine, which requires the classifier to adjust the distance between the classifier and the real classifier to be as small as possible when encountering the incorrect samples, while at the same time to ensuring the previous samples are correct. Online gradient descent algorithm is the application of offline gradient descent algorithm to an online learning situation. This algorithm is the basis of many online learning algorithms. In each round of training, the algorithm will update the direction of the loss function according to the existing model and parameters.

In 2008, the Confidence-Weighted algorithm (CW) [6, 20] based on the weight probability distribution hypothesis shows a positive results in natural language processing. The CW algorithm imposes a the Gaussian distribution on the parameter vector while updating each new training instance, so that the probability of correct classification of the instance under the updated distribution meets the specified confidence. Subsequent to the CW algorithm, in order to generate a sparse solution, a Truncated Gradient Algorithm (TG) [12] was proposed on the basis of convex optimization. Its core idea is to control the size of the coefficient via an imposed threshold.

In summary, batch learning algorithms cannot solve the problem of incremental data learning, and existing online learning algorithms are faced with problems involving sample noise and low classification accuracy. Ultimately, previous researchers have not addressed these issues in a comprehensive manner. In this paper, therefore, we propose several methods in order to solve these important problems.

3 Proposed methods

The online learning algorithm many be applied in a wide range fields, however, the error rate of the online learning algorithm is relatively high under the classification problem due to its optimization strategy. When the parameters are updated during the training process, problems such as low convergence speed and high noise often occur [3]. In order to solve the existing problems, this paper proposes an integrated classifier model for incremental learning, as shown in Fig. 1. After the data samples are input, the neural network is trained to obtain a pre-trained model, then the obtained pre-trained model is used to process the subsequent input data to obtain feature vectors. Next, the feature vectors are input into the TGCW classifier to obtain the final classification result. The integrated classification model consists of a pre-trained model and a TGCW classification model. The pre-trained model includes a DNN training module as well as a CNN training module [21]. The DNN training module and the CNN training module are two completely independent models. These two sub-modules are used to process vector data and matrix data, respectively. The type of data samples determines which module is selected for training. If the given data samples are vectors, the vector samples are input into the DNN module for training. If the given data sample is an image, the matrix is input into the CNN module for training. After the pre-trained model is obtained, the remaining data is input into the pre-trained model. It generates a set of feature vectors, and the classification model uses these feature vectors to generate prediction labels. If the predicted label is different from the real label, the classification model parameters will be updated, otherwise, the model parameters will remain unchanged.

Fig. 1 Integrated classifier architecture

It should be pointed out that automatic data type detection is not implemented in the input layer of our model. In our later experiment section, the datasets are manually classified and entered into the model, separately. However, such an automatic detection feature could be easily realized. Here we provide an operational idea: a certain amount of data can be randomly selected from the dataset to be processed, and input into the vector pre-training module and matrix pre-training module, respectively. Then, using cross validation, a comparison of the performance of the two models may be carried out, with the model with superior performance then chosen for processing the complete dataset. This method is feasible because the better performance shows whether the vector or matrix methods can better describe the internal structure of the dataset. Cross validation can also be replaced by other validation methods that are more suitable for the current data.

3.1 Vector pre-trained model

For vector data, the pre-trained model uses a DNN without a convolution layer for training. The network uses three fully connected layers, with 64 nodes in the first two fully connected layers, as shown in Fig. 2. When the data is in vector format, a deep neural network without convolution layer is used for pre-training. In the pre-training step, three fully-connected layers are used to learn the characteristics of data, and the parameters of neurons in each layer are updated through back propagation to retain the information of vector quantity. The activation function uses the ReLU function, and the Softmax function is used as the activation function in the third fully-connected layer. At

Fig. 2 Vector pre-trained model

the same time, the Dropout layer is used between two adjacent fully-connected layers, and each time the entire network is trained, the Dropout layer will be discarded with a certain probability for each node.

3.2 Matrix pre-trained model

For matrix data, the pre-trained model uses CNN for training [17], as shown in Fig. 3. There are five convolution layers. Each convolution layer uses a 3×3 convolution kernel and 32 nodes. At the same time, the batch normalization layer is used to normalize the data after each convolution to speed up the search for the optimal solution of gradient descent. After the first, third and fifth convolution layers, a MaxPooling layer and the Dropout layer are added. The pool size of the MaxPooling layer is 2×2 , and the step size is 2. Two fully connected layers are used after the fully connected layer. The first fully-connected layer has 1024 nodes. The activation function uses the ReLU function. After the fully-connected layer, the Batch Normalization layer is used to normalize the data. Then the Dropout layer is used to avoid overfitting. When inputting image data, convolution and maximum pool are used to extract and represent image data in the deep pre-training part containing convolution neural network. After convolution for many times, the obtained feature information is output to all the connection layers of 1024 nodes in a layer to obtain the image feature vector.

3.3 TGCW classification model

In this section, we propose a new online learning algorithm suitable for binary classification of streamed data, named TGCW, which aims to further improve the prediction accuracy and feature selection capability of the model. 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. When used for binary classification, it will not obtain satisfactory performance. (2) The sparsity of the model parameters is poor, which will lead to low interpretability and performance degradation, at least to some extent.

To overcome these disadvantages, we have considered two adjustments: (1) similar to the PA algorithm and the SCW learning process, we introduce a parameter C to control the aggressiveness versus passiveness [11] of the learning algorithm; (2) streaming data makes

Fig. 3 Matrix pre-trained model

feature selection difficult, so we introduce the TG algorithm to truncate coefficients to be smaller than the threshold θ to reduce the dimension of streaming data.

Below we will elaborate our method. We assume that the coefficient follows a Gaussian distribution with the mean vector μ_i and the covariance matrix Σ_i , as per the CW method, and allow an immediate update of the variables μ_{t+1} and Σ_{t+1} . To deal with the incremental streaming data, we apply the cost-sensitive strategy, i.e.,

$$(\mu_{t+1}, \Sigma_{t+1}) = \underset{\mu, \Sigma}{\operatorname{argmin}} D_{KL} \left(N(\mu, \Sigma) \| N(\mu_t, \Sigma_t) \right) + C_{cs} \ell^\phi(N(\mu, \Sigma); (x_t, y_t)) \quad (3)$$

Where, C_{cs} is the cost assigned to the suffered loss. The quantity ℓ^ϕ is the loss that measures the probability of the misclassification based on the calculated Gaussian distribution, where

$$D_{KL}(N(\mu, \Sigma); (x_t, y_t)) = \frac{1}{2} \log \left(\frac{\det \Sigma_t}{\det \Sigma} \right) + \frac{1}{2} \operatorname{Tr}(\Sigma_t^{-1} \Sigma) + \frac{1}{2} (\mu_t - \mu)^T \Sigma_t^{-1} (\mu_t - \mu) - \frac{d}{2} \quad (4)$$

is the KL divergence, measuring the difference between the two distributions.

Online learning requires the sequential arrival of the streaming data that cannot be reused. This constraint makes it difficult to perform feature selection. Truncated gradient (TG) is proposed to select the discriminate features. After the cost-sensitive CW update at each iteration, we apply TG to select the most active features. Specifically, we define:

$$w = TG(w) = T(w - \gamma \nabla l(w; (x_t, y_t)), \gamma g, \theta) \quad (5)$$

$$T(w_j, a, \theta) = \begin{cases} \max(0, w_j - a), & \text{if } w_j \in (0, \theta) \\ \min(0, w_j + a), & \text{if } w_j \in (-\theta, 0) \\ w_j, & \text{otherwise.} \end{cases} \quad (6)$$

Where,

$$l(w; (x_t, y_t)) = \max(0, 1 - y_t (w^T x_t)) \quad (7)$$

γ is the learning rate, g is the gravity parameter, and θ is the threshold. x_t and y_t represent the sample value and label of the current iteration round, respectively. This TG operation can be performed in every K online steps.

3.4 Robustness analysis

Because of the feature extraction in the pre-trained part, the performance of the whole model is greatly improved. In addition, we also find that the robustness of the model has been significantly improved at the same time for the following reasons. (1) The convolution layer and the pooling layer in the CNN can filter at least some of the noise. (2) CNN Local Connectivity and Shared Weights strategy can also weaken the influence of the noise. If the classification accuracy is further improved, a convolution attention network can be used for preprocessing [22]. (3) For the DNN network for vector samples, the network structure with dense connection of DNN can improve the robustness of the whole model, which is more reliable than the model without pre-trained layer. (4) DNN network can extract features from datasets, and then put the feature vector into the TGCW

model, which is conducive to weakening the direct impact of noise on model parameters. Based on the above four points, we believe that the pre-trained model has significant robustness, which is also confirmed by subsequent experiments. Details of our experiment are given in Section 4.

4 Experiments and analysis

4.1 Datasets

The main task of online classification is to minimize cumulative errors to obtain higher accuracy. Therefore, in order to test the performance of our proposed integrated classification model, we carefully selected five datasets from different fields from the UC Irvine Machine Learning Repository and the KEEL dataset repository. *Glass1*, *Glass2*, *Glass5*, *Ecoli3* and *Haberman* are all well-known classically used test datasets. The *Glass* datasets contain attributes about several glass types that can be used in criminological investigations.

The *Ecoli* data contains protein localization sites and the *Haberman* dataset contains cases from a study that was conducted between 1958 and 1970 at the University of Chicago's Billings Hospital on the survival of patients who had undergone surgery for breast cancer. In addition, the *masked_faces* dataset was collected by Yin Xiangzhi and others. It contains 2858 pictures of faces wearing masks and 1221 pictures of faces not wearing masks. The faces in the dataset have different directions and occlusion degrees. The *Dogs vs. Cats* dataset comes from the Kaggle Dogs vs. Cats competition dataset, which includes 12,500 pictures of cats and 12,500 pictures of dogs. The color, age, and position of the cats and dogs in the picture are different. The *Animals10* dataset comes from the Kaggle *Animals-10* competition dataset. This dataset contains images of 10 kinds of animals, including dog, cat, horse, spider, butterfly, chicken, sheep, cow, squirrel, elephant. We choose horses and elephants from the *animals10* dataset as the dataset used in the experiment. This part of the dataset contains 2623 images of horses and 1436 images of elephants. The *MNIST* database is a large database of handwritten digits that is commonly used for training various image processing systems. We selected 4132 images of "0" and 4684 images of "1" from the *MNIST* dataset for experiments. The *CIFAR-10* dataset contains 10

Table 1 Dataset summary

dataset	#samples	#attributes	#ratio
Animals10	4059	3072	0.55
Cifar-10	10,000	3072	1.00
Dogs vs. Cats	25,000	3072	1.00
Mnist	8816	1024	1.13
Glass1	214	9	1.82
Masked_face	4097	3072	2.34
Haberman	306	3	2.78
Ecoli3	36	7	8.19
Glass2	214	9	10.39
Glass5	214	9	23.42

different classes. The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. We choose 5000 images of airplanes and 5000 images of trucks for experiment. These datasets are used to test the accuracy of integrated classification models in image classification applications. In addition, under the current Covid-19 pandemic situation, the algorithm has certain practical value in the detection of mask wearing. Table 1 shows the specifications of the 10 datasets (Fig. 4).

4.2 Performance for online incremental classifier

Accuracy is the most important metric for evaluating the model performance in the classification problem. To show the efficiency of our online learning algorithm, we compare our method with five kinds of state-of-the-art online learning algorithms, including Perceptron, OGD, PA, CW and IELLIP [23, 24].

Perceptron is a classifier that uses hyper planes for binary classification, and uses new data instances, predictions, comparisons, and updates each time to adjust the location of hyper planes. The learning process of PA is basically the same as that of Perceptron. When the weight is revised, a parameter T_t is added. When the prediction is correct, there is no need to adjust the weight. When the prediction is wrong, the weight is actively adjusted. Each learning parameter of the CW method has a degree of trust. Parameters with a small degree of trust should be trained, so more frequent opportunities for modification occur. The degree of confidence is expressed by the Gaussian distribution of the parameter vector. OGD uses the obtained data to perform a gradient descent every time, and updates the parameters according to the result of the gradient descent. The key idea of IELLIP is to approximate by an ellipsoid the classification hypotheses. In addition to the classical ellipsoid method, an improved version for online learning is also presented.

We carried out the experiments by first obtaining the optimal parameters for other algorithms for each method on each dataset, then applied each algorithm 10 times using these optimal parameters on each dataset, each time with a randomly permuted sequence. All results are reported by averaging over these 10 runs. There are three performance metrics which are used to evaluate the performance of online learning algorithm, (1) online cumulative mistake rate, (2) number of updates (which would be closely related to the potential number of support vectors in the kernel extension), and (3) the cost of running time.

Table 2 lists the results of our empirical evaluation of cumulative performance of the proposed TGCW and other algorithms in five classes of datasets, where we show the six kinds of online learning algorithms. The bold elements indicate the best performance in different

Fig. 4 Masked_faces dataset and Dogs vs. Cats dataset

Table 2 Evaluation of cumulative performance of the proposed TGCW and other algorithms

Algorithm	#Mistakes	#Updates	#Time(s)	Algorithm	#Mistakes	#Updates	#Time(s)
dataset name: glass1 ($n = 214, d = 9, m = 2$)							
Perceptron	0.464 ± 0.049	99.2 ± 10.5	0.010 ± 0.001	dataset name: haberman ($n = 306, d = 3, m = 2$)	0.381 ± 0.024	116.7 ± 7.5	0.014 ± 0.001
OGD	0.468 ± 0.047	100.5 ± 10.2	0.011 ± 0.001	Perceptron	0.362 ± 0.014	114.7 ± 4.0	0.015 ± 0.001
PA	0.474 ± 0.049	144.2 ± 7.4	0.011 ± 0.001	OGD	0.392 ± 0.025	180.7 ± 8.4	0.015 ± 0.000
CW	0.406 ± 0.039	101.5 ± 10.4	0.014 ± 0.002	PA	0.489 ± 0.074	189.4 ± 21.9	0.022 ± 0.002
IELLIP	0.474 ± 0.050	149.0 ± 6.7	0.014 ± 0.001	CW	0.390 ± 0.025	119.3 ± 7.6	0.018 ± 0.001
TGCW	0.369 ± 0.016	154.2 ± 5.7	0.016 ± 0.006	IELLIP	0.268 ± 0.010	89.1 ± 3.0	0.020 ± 0.005
dataset name: ecolif3 ($n = 336, d = 7, m = 2$)							
Perceptron	0.153 ± 0.012	51.3 ± 4.0	0.015 ± 0.001	dataset name: glass2 ($n = 214, d = 9, m = 2$)	0.147 ± 0.007	31.4 ± 1.5	0.010 ± 0.001
OGD	0.110 ± 0.002	75.4 ± 1.9	0.016 ± 0.001	Perceptron	0.138 ± 0.008	30.0 ± 1.9	0.010 ± 0.001
PA	0.138 ± 0.010	113.8 ± 8.1	0.016 ± 0.001	OGD	0.146 ± 0.006	57.3 ± 4.2	0.010 ± 0.001
CW	0.121 ± 0.009	99.5 ± 7.9	0.017 ± 0.003	PA	0.145 ± 0.005	67.3 ± 6.8	0.014 ± 0.001
IELLIP	0.157 ± 0.009	52.7 ± 3.6	0.021 ± 0.003	CW	0.146 ± 0.005	31.2 ± 1.2	0.012 ± 0.001
TGCW	0.107 ± 0.016	47.1 ± 2.3	0.022 ± 0.002	IELLIP	0.082 ± 0.003	22.0 ± 3.2	0.014 ± 0.003
dataset name: glass5 ($n = 214, d = 9, m = 2$)							
Perceptron	0.087 ± 0.004	18.6 ± 0.8	0.010 ± 0.001	dataset name: Dogs vs. Cats ($n = 25,000, d = 3072, m = 2$)	0.9953 ± 0.0024	24,882.95 ± 59.71	23.1529 ± 4.5283
OGD	0.061 ± 0.011	13.2 ± 2.3	0.011 ± 0.001	Perceptron	0.7301 ± 0.0017	11,452.50 ± 82.98	17.4929 ± 0.3421
PA	0.087 ± 0.004	42.5 ± 5.1	0.011 ± 0.002	OGD	0.7354 ± 0.0017	18,586.05 ± 69.77	20.0809 ± 2.9974
CW	0.087 ± 0.008	44.7 ± 3.6	0.014 ± 0.001	PA	0.7472 ± 0.0035	25,642.95 ± 88.36	46.18.157 ± 5.3682
IELLIP	0.059 ± 0.010	18.6 ± 0.8	0.012 ± 0.001	CW	0.7473 ± 0.0056	26,348.65 ± 53.69	41.13.295 ± 6.3258
TGCW	0.047 ± 0.015	12.8 ± 1.7	0.014 ± 0.001	IELLIP	0.7272 ± 0.0045	17,846.95 ± 95.63	18.71.339 ± 5.3243
dataset name: Masked faces ($n = 4097, d = 3072, m = 2$)							
Perceptron	0.4658 ± 0.0031	11,644.55 ± 77.05	26.1996 ± 9.3194	dataset name: mnist ($n = 8816, d = 1024, m = 2$)	0.7010 ± 0.0011	6179.65 ± 9.40	1.7934 ± 0.0770
OGD	0.4578 ± 0.0028	11,444.75 ± 70.75	22.3816 ± 0.7942	Perceptron	0.5432 ± 0.0005	40.10 ± 9.73	1.7631 ± 0.0916
PA	0.4675 ± 0.0034	18,569.35 ± 61.05	24.5853 ± 1.4433	OGD	0.5424 ± 0.0002	169.25 ± 9.63	1.5806 ± 0.1430
CW	0.4615 ± 0.0046	19,624.65 ± 65.35	24,02.326 ± 1.6344	PA	0.5419 ± 0.0002	298.70 ± 12.22	19.333 ± 1.0478
IELLIP	0.4684 ± 0.0053	18,335.55 ± 74.55	1538.583 ± 1.6653	CW	0.5448 ± 0.0002	4714.90 ± 1.71	159.91 ± 35.8319
TGCW	0.4397 ± 0.0065	11,425.35 ± 85.45	1274.445 ± 1.3653	IELLIP	0.5341 ± 0.0011	61.20 ± 11.51	17.926 ± 0.3653
dataset name: animals10 ($n = 4059, d = 3072, m = 2$)							
Perceptron	0.7045 ± 0.0011	2859.40 ± 4.41	3.0558 ± 0.2644	dataset name: cifar-10 ($n = 10,000, d = 3072, m = 2$)	0.9259 ± 0.0013	9259.20 ± 13.42	7.2931 ± 0.6308
OGD	0.5668 ± 0.0040	1579.05 ± 29.74	3.0752 ± 0.4280	Perceptron	0.6903 ± 0.0022	2695.60 ± 43.25	6.2377 ± 0.2134
PA	0.5773 ± 0.0041	2808.65 ± 34.17	2.2134 ± 0.0795	OGD	0.6842 ± 0.0023	5759.40 ± 48.43	6.5276 ± 0.1275
CW	0.5842 ± 0.0135	3102.45 ± 66.75	448.36 ± 0.0935	PA	0.6843 ± 0.0093	8559.65 ± 56.35	641.61 ± 0.6653
IELLIP	0.5767 ± 0.0945	2945.65 ± 87.15	338.74 ± 0.0639	CW	0.6963 ± 0.0053	6635.30 ± 52.55	875.60 ± 0.5485
TGCW	0.5618 ± 0.0088	1652.50 ± 31.47	201.1599 ± 17.6320	IELLIP	0.6767 ± 0.0054	3685.45 ± 53.79	431.75 ± 4.6903
TGCW				TGCW

Fig. 5 Evaluation of cumulative performance of the proposed TGCW and other algorithms. **a** Glass1 **b** Haberman **c** Ecoli3 **d** Glass2 **e** Glass5

(d)

(e)

Fig. 5 continued.

datasets. We can draw several observations as follows. First, in terms of the overall mistake rate, our proposed TGCW outperforms other online learning algorithms in all five datasets. Specifically, compared to the strongest baselines, TGCW decreases the mistake rate by 9.1%, 26.0%, 2.7%, 40.6% and 20.3% in the dataset of *Glass1*, *Haberman*, *Ecoli3*, *Glass2* and *Glass5* respectively. Second, by examining the number of updates, we found that TGCW online learning algorithm outperforms other methods. When the number of samples is large, the algorithm has a smaller update rate characteristic. Our proposed TGCW algorithm improves on the CW algorithm, so our method inherits the excellent feature. Moreover, among all the compared algorithms, TGCW often achieves the best or close to the best performance in terms of accuracy and number of updates. Finally, Fig. 5 shows us the online results of six kinds of algorithms with five varied numbers of datasets in the online learning process. The results again validate the advantages of TGCW in both efficacy and efficiency among all the state-of-the-art algorithms.

Fig. 6 Experiment steps and process

4.3 Performance for integrated classification model

In order to verify the performance of the integrated classifier, we designed the experimental steps and flow as shown in Fig. 6. Then, randomly selecting 10% of the dataset as data samples into the neural network for training to obtain a pre-trained model, the pre-trained model is used to process the remaining data to extract the feature vectors of the data. Next, the feature vector is input into the TGCW online classifier for online incremental learning, with 20 training rounds performed in the classifier, and the average value of these results from these 20 training rounds is taken as the final result. In addition, the remaining data is entered into the original TGCW classifier for online learning to obtain the classification result, again taking the average of 20 experimental results, then comparing the performance of the integrated classifier and the original classifier.

The experimental results, as shown in the blue part of Fig. 7, show that the performance of the integrated classifier is greatly improved compared to the original classifier. The way that the integrated classifier uses the data features extracted by the pre-trained model has a greater effect in reducing data noise and speeding up the parameter update process. On ten datasets, the error rate of classification decreased significantly. At the same time, it can be found that the integrated classifier has a better improvement on the Dogs vs. Cats dataset, which shows that the pre-trained model will have a superior improvement effect in the face of relatively complex data. By comparing the performance of the ten datasets, we can find that the performance of the integrated classifier can be greatly improved on the basis of the original classifier.

Then, we randomly added Gaussian noise with a mean of 0, 0.3 or 0.5 and a variance of 0.1, 0.3 or 0.5 in 10% of the data in the original dataset. Then, the TGCW classifiers and integrated classifiers are used for classification. The result is shown as the red part of Fig. 7, and the experimental results show that the integrated classifier still has good classification performance after adding noise, which shows that the integrated classifier has advantages over the traditional classifier in dealing with noisy data. The change of the model before and after adding noise and the comparison between the models can also be seen clearly from Fig. 7.

Fig. 7 Performance comparison between the Pt-TGCW and the TGCW classifier on different datasets when we randomly added Gaussian noise in the original dataset. **a** Masked_faces **b** Dogs vs Cats **c** MNIST **d** Animals10 **e** Cifar-10 **f** Ecoli3 **g** Glass1 **h** Haberman **i** Glass2 **j** Glass5

5 Conclusion

In this paper, a new, integrated classifier for incremental learning is proposed, which uses pre-trained deep neural networks to extract the characteristic vectors of the data, and then uses the TGCW classifier for incremental learning. The experimental results show that the integrated classifier has great advantages in incremental learning. The results of running on ten different types of datasets indicate that the integrated classifier has a significant improvement in performance compared to traditional classifiers. At the same time, the operation results on the noise-added dataset show that after the introduction of noise the average increase of the mistake rate of the integrated classifier is about 1.5%, which is much lower than that of the original classifier (approx. 6%). Hence, the integrated classifier has advantages in processing noisy data, and it also shows that our integrated classifier can have superior robustness and generalization while maintaining the convergence speed. The dense connection structure and the convolution and pooling layer of neural network improves the robustness of the model. Our integrated classifier still has many shortcomings. At present, the ability of the integrated classifier to process complex image data is still poor. The accuracy of some image data is less than 50% and the training overhead is relatively large. Our integrated classifier is still limited to binary classification problems, which restricts its application to many practical situations. Future work will focus on extending our integrated classifier to multi-classification problems. In addition, we will also look for improved pre-trained models or use more classifiers for integrated learning to improve the classification accuracy of complex data.

References

1. Ben-David S, Kushilevitz E, Mansour Y (1997) Online learning versus offline learning. *Mach Learn* 29(1): 45–63
2. Bottou L (2012) Stochastic gradient descent tricks. *Neural networks: tricks of the trade*. https://doi.org/10.1007/978-3-642-35289-8_25
3. Crammer K, Dekel et al (2006) Online passive-aggressive algorithms. *J Mach Learn Res* 7(3):551–585
4. Crammer K, Dredze M, Kulesza (2009) Multi-class confidence weighted algorithms. *Conference on Empirical Methods in Natural Language Processing*, pp. 496–504
5. Crammer K, Kulesza A, Dredze (2013) Adaptive regularization of weight vectors. *Machine Learning*, pp. 345–352
6. Dredze M, Crammer K, Pereira F (2008) Confidence-weighted linear classification. *International Conference DBLP*, pp. 264–271
7. Dredze M, Crammer K, Pereira F (2008) Confidence-weighted linear classification. *Proceedings of the 25th international conference on Machine learning*, pp. 264–271
8. Duchi J, Singer Y (2009) Efficient online and batch learning using forward backward splitting. *J Mach Learn Res* 10:2899–2934
9. Freund Y, Schapire RE (1999) Large margin classification using the perceptron algorithm. *Mach Learn* 37(3):277–296
10. Giraud-Carrier C (2000) A note on the utility of incremental learning. *AI Commun* 13(4):215–223
11. Hu J, Yan C, Liu X, et al (2019) Truncated gradient confidence-weighted based online learning for imbalance streaming data. *IEEE International Conference on Multimedia and Expo (ICME)* pp. 133–138
12. Langford J, Li L, Zhang T (2009) Sparse on learning via truncated gradient. *J Mach Learn Res* 10:777–801
13. Orabona F, Crammer K (2010) New adaptive algorithms for online classification. *Advances in Neural Information Processing Systems*, pp 1840–1848
14. Rosenblatt F (1958) The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol Rev* 65:386–408
15. Schmidhuber J (2015) Deep Learning in Neural Networks: An Overview. *Neural Netw* 61:85–117

16. Shalev-Shwartz S (2012) Online learning and online convex optimization. *Found Trends® Mach Learn* 4(2):107–194
17. Simonyan K, Zisserman A (2014) Very deep convolution networks for large-scale image recognition. In *ICLR*. <https://arxiv.org/abs/1409.1556>
18. Syed NA, Huan S, Kah L, Sung K (1999) Incremental learning with support vector machines. *IEEE International Conference on Data Mining*, IEEE Computer Society, 1999
19. Wang J, Zhao P, Hoi SCH (2014) Cost-sensitive online classification. *IEEE Trans Knowl Data Eng* 26(10): 2425–2438
20. Wang J, Zhao P, Hoi SCH (2016) Soft confidence-weighted learning. *ACM*. 8:1–32. <https://doi.org/10.1145/2932193>
21. Xie H, Yang D, Sun N, Chen Z, Zhang Y (2019) Automated pulmonary nodule detection in CT images using deep convolution neural networks. *Pattern Recogn* 85:109–119
22. Xie H, Fang S, Zha Z, Yang Y et al (2019) Convolution attention networks for scene text recognition. *ACM Trans Multimed Comput Commun Appl* 15:1–17
23. Kivinen, J., Smola, A. J., & Williamson, R. C. (2004) Learning with Kernels. *IEEE Transactions on Signal Processing* 52(8):2165–2176
24. Yang L, Jin R, Lansing E, et al. (2009) Online learning by ellipsoid method. *International Conference on Machine Learning*. ACM
25. Yosinski J, Clune J, Bengio Y, Lipson H (2014) How transferable are features in deep neural networks? *International Conference on Neural Information Processing Systems*, MIT Press, 2014
26. Beichen Zhang, Liang Li, et al. (2020) State-relabeling adversarial active learning, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8753–8762

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Affiliations

Ji Hu¹ · Chenggang Yan¹ · Xin Liu¹ · Zhiyuan Li¹ · Chengwei Ren¹ · Jiyong Zhang¹ · Dongliang Peng¹ · Yi Yang²

Chenggang Yan
cgyan@hdu.edu.cn

Xin Liu
xinliu@hdu.edu.cn

Zhiyuan Li
18061220@hdu.edu.cn

Chengwei Ren
rencw@hdu.edu.cn

Jiyong Zhang
jzhang@hdu.edu.cn

Dongliang Peng
dlpeng@hdu.edu.cn

Yi Yang
yi.yang@uts.edu.au

¹ HangZhou DianZi University, HangZhou, ZheJiang, China

² Centre for Artificial Intelligence, University of Technology Sydney, Ultimo, Australia