Dynamic Weighting Ensembles for Incremental Learning

Xinzhu Yang, Bo Yuan, Wenhuang Liu

Graduate School at Shenzhen, Tsinghua University, Shenzhen 518055, P. R. China
E-mail: yangxz03@mails.tsinghua.edu.cn, {yuanb, liuwh}@sz.tsinghua.edu.cn

Abstract: This paper investigates an interesting question of solving incremental learning problems using ensemble algorithms. The motivation is to help classifiers learn additional information from new batches of data incrementally while preserving previously acquired knowledge. Experimental results show that the proposed dynamic weighting scheme can achieve better performance compared to the fixed weighting scheme on a variety of standard UCI benchmark datasets.

Key Words: Incremental Learning, Dynamic Weighting, Ensembles

1 INTRODUCTION

Supervised learning methods have been applied to many real world classification problems, such as high-value customer prediction, disease pre-diagnosis and so on. However, the generalization performance of any learning algorithm relies heavily on adequate and representative training data [1]. For many practical applications, new data may become available ceaselessly along with new customer registrations or new patients arriving. In the meantime, data collection is often an expensive and time consuming process and data are often acquired in batches over time [1]. Moreover, new data may bring distinct changes of distributions and even features and ignoring these new data may lead to deteriorating prediction accuracies in the changing environments. As a result, how to incorporate the new data into an existing classifier has become an important challenge in practice.

A simple approach to learning new information involves discarding the current classifier, and retraining a new classifier using all of the data that have been accumulated so far [2]. However, this method will inevitably result in the loss of all previously acquired knowledge, which is known as catastrophic forgetting [2]. Furthermore, collecting all available data to retrain a new classifier may also lead to a much more complicated classification problem, which is difficult to learn. By contrast, it is more desirable to train a classifier on available data and incrementally update it as new data become available, without compromising its performance on existing data [1].

A number of algorithms have already been suggested in the literature for incremental learning in different scenarios [2]. In this paper, the term "incremental learning" is defined in a way similar to the definition given by Polikar *et al.* [2], which focuses on the capability of learning additional information from new data and preserving previously acquired knowledge. The main difference is that, in our work, unrestricted access to the old data is allowed as in practice old data such as the demographic information and transaction histories of customers are usually kept in record.

Since ensemble learning can combine a group of individual classifiers with different strengths in order to help improve the accuracy of a single learning model, it has been a natural solution to the issue of incremental learning. An ensemble of classifiers refers to a set of classifiers whose individual decisions are combined in some way to determine the class labels of unknown samples [3]. Individual classifiers are trained using different distributions of the training data, which generate diversity among the basic classifiers. The motivation of this paper is to investigate and compare different schemes for using ensemble methods to address incremental learning problems.

The rest part of this paper is organized as follows. Section 2 gives an overview of the related work including incremental learning and dynamic weighting ensembles. In Section 3, the details of the dynamic weighting scheme are presented. Experimental results are analyzed in Section 4. This paper is concluded in Section 5 with a list of directions for future work.

2 RELATED WORK

2.1 Incremental Learning

Broadly speaking, the phrase "incremental learning" refers to classification tasks where only part of the training data are available in the initial stage and the classifier has to keep learning from new data whenever possible. In this situation, it is desirable for the classifier to be able to learn new patterns and never forget old memories (knowledge) due to the learning of new patterns [4]. Within this scope, various algorithms have been proposed focusing on the growing or pruning of classifier architectures, selection of the most informative training samples, or the modification of classifier weights [2].

Polikar *et al.* [2] give the term "incremental learning" more restrictions. In addition to the above two conditions, algorithms should not require access to the original data and should be able to accommodate new classes that may be introduced from the new data. Under these conditions, the authors propose the algorithm Learn++, which inherits a popular ensemble method AdaBoost [5] to solve incremental learning problems. It iteratively generates an ensemble of classifiers for each data set that becomes available, and combines them using weighted majority voting [2]. Later on, a series of extensions and improvements of Learn++ have been introduced, including Learn++ for concept drifting [6], Learn++ for data fusion [7] and Learn++ for imbalanced data [8].

The original Learn++ suffers from the "out-voting" problem, which means that when a new class is introduced, the decisions of latter classifiers that recognize it are out-voted by previous classifiers that do not recognize it, until a sufficient number of new classifiers are generated [9]. Some modified versions, such as Learn++.MT [9] using Dynamic Weighted Voting (DWV) and Learn++.NC [10] using Dynamically Weighted Consult and Vote (DW-CAV), are proposed to address this issue. In order to avoid relying on the previous data, the class-specific Mahalanobis distance metric is introduced to compute the distance between the training data and the unknown instance for each classifier in order to calculate the dynamic weights. However, the Mahalanobis distance implicitly assumes that the underlying data distribution is Gaussian, which in general is not the case [1].

2.2 Dynamic Weighting Ensembles

Many strategies for combining individual classifiers have been developed, such as unweighted voting in Bagging [11] and Random Forests [12], weighted voting in Boosting [5], learning a combiner function [13] and stacking [14]. In weighted voting schemes, the weights are based on each basic classifier's error on the corresponding weighted training set, which means that the weights of classifiers are constant and will not change for different new input samples. Obviously, this approach ignores the inherent performance variability of classifiers on new samples with different feature values. In fact, the same basic classifier may not be as effective at classifying samples in some areas of the feature space as in other areas.

As a result, various dynamic weighting schemes have been proposed in recent years, which take the features of input samples into consideration. Typical examples include dynamic voting (DV), dynamic selection (DS), and dynamic voting selection (DVS) [15]. Some of these schemes are independent of the underlying ensemble algorithms. For example, DV, DS, DVS have been applied to Bagging, Boosting and Random Forests respectively [15, 16]. Other strategies focus on specific ensemble methods, among which Boosting is the most popular one. A number of Boosting methods with dynamic weighting schemes have been proposed independently with proven superior performance over standard Boosting algorithms, including RegionBoost [17], iBoost [18], WeightBoost [19] and local Boost [20].

However, the essential ideas are very similar: an extra learner is introduced for every basic classifier as a competency predictor to evaluate the dynamic input-dependent weights for each classifier. Generally speaking, the extra learner is trained to indicate whether a certain basic classifier is likely to yield accurate results given an unknown sample. Many of the commonly used classification methods have been employed as the competency predictors such as k-Nearest Neighbor (RegionBoost), Neural Networks (RegionBoost) and Decision Trees (iBoost). It is clear that one of the key issues in dynamic weighting schemes is how to construct the competency predictors to appropriately determine the weights, which can significantly affect the performance of the combined classifiers.

Considering the situation where new data are acquired in batches over time, a practical strategy is to train a new ensemble of classifiers for each set of new data and put all ensembles together as an updated version of the original classifier. However, data presented to each ensemble are likely to have different distribution patterns and it is unreliable to adopt the fixed weights of the basic classifiers for all kinds of unknown samples. On the contrary, dynamic weights estimated using the local information can be expected to solve this issue to some extent. In the next, we will present the key idea of using dynamic weighting ensembles for incremental learning.

3 ALGORITHM FRAMEWORK

This section introduces an algorithm framework for using dynamic weighting ensembles to effectively learn new batches of data appearing over time without the need of retraining. There are two phases in the proposed algorithm (see Fig. 1). The first phase is to train an ensemble of classifiers based on each batch of the input data. The second phase is to combine the outputs from individual classifiers.

In the first phase, when a new batch of data becomes available, a new ensemble of basic classifiers is built solely on it so that the new information can be effectively extracted, without interfering with existing classifiers. Another advantage is that the training process has much better flexibility than the retraining strategy. For example, when the new data contain attributes different from the current data, it would be very difficult, if not impossible, to train a new classifier based on the mixed dataset. For each ensemble, the training process is the same as AdaBoost, which has proven to be able to help improve the accuracy of a single classifier in many applications.

In the second phase, the dynamic weighting scheme in RegionBoost is used to combine all the classifiers. The main idea behind RegionBoost is to build an extra model upon each basic classifier based on its training results (whether a training sample is classified correctly or not). By doing so, this new model is able to estimate the accuracy or the competency of the classifier for each new sample.

One of the intuitive approaches to estimating model competency is to use the kNN (*k*-Nearest Neighbor) method to find the *k* points in the training set nearest to the new sample to be classified and use the performance of each classifier on these *k* points as the measurement [17]. More specifically, the weight of each classifier for this new sample to be classified is determined by the percentage of points (out of *k* points) correctly classified by this classifier [17].

Note that, the new data set is put together with all the old sets to evaluate each basic classifier. At last, given an unknown input, all the individual classifiers are combined using the dynamic weights to form the final hypothesis.

```
Input:
  • Old training sets $S^k = \{(x_1^k, y_1^k), ..., (x_m^k, y_m^k)\}$,
    $k = 1, 2, ..., K-1$.
  • New training set $S^K = \{(x_1^K, y_1^K), ..., (x_m^K, y_m^K)\}$.
  • Old learners $L^k = \{h_1^1, ..., h_T^1, ..., h_1^{K-1}, ..., h_T^{K-1}\}$.
For $S^K$: Initialize $w_1(i) = D(i) = 1/m, \forall i$.
  Do for $t = 1, 2, ..., T$:
    1. Call WeakLearn,
       providing it with $S^K$, according to $D_t^K$.
    2. Obtain $h_t^K : X \to Y^K$, and calculate its error
       $\varepsilon_t^K = \sum_{i: h_t^K(x_i) \neq y_i} D_t^K(i)$. If $\varepsilon_t^K = 0$, then set $\varepsilon_t^K = 10^{-10}$.
    3. Calculate $\alpha_t^K = \frac{1}{2}\log(\frac{1-\varepsilon_t^K}{\varepsilon_t^K})$.
    4. $w_{t+1}^K(i) = w_t^K(i) \times \exp(-\alpha_t^K h_t^K(x_i) y_i)$,
       set $D_{t+1}^K = w_t^K / \sum_{i=1}^{m} w_t^K(i)$ so that $D_{t+1}^K$ is a distribution.
For each testing data $x_j, j = 1, ..., J$
  1. Combine training sets $S = \bigcup_{k=1}^{K} S^k$.
  2. Find $p$ nearest neighbors $N_j$ in $S$.
  3. Call Dynamic Weighted Voting:
     $\beta_t^k = \frac{1}{p}\sum_{i \in N_j}(h_t^k(x_i) = y_i), k = 1, ..., K, t = 1, ..., T$.
  4. Final hyopthesis: $H(x_j) = \sum_{k=1}^{K}\sum_{t=1}^{T}\beta_t^k h_t^k(x_j)$.
```

Fig. 1: The framework of the incremental learning algorithm based on dynamic weighting ensembles

4 EXPERIMENTS

This section presents some empirical studies of the incremental learning algorithm based on dynamic weighting ensembles on a variety of datasets.

4.1. A Toy Example

In order to illustrate the advantage of the dynamic weighting scheme for incremental learning problems, a toy experiment on a 2D synthetic dataset was conducted. Three batches of training data (400 samples each) were made available in turn and the distributions of the datasets are shown in Fig. 2. Once a new batch of data was available, a classifier was trained solely based on this specific dataset. Note that due to the simplicity of the classification tasks, a single LDA classifier was used for each dataset instead of an ensemble of classifiers.

Next, the results of the combined classifiers using fixed weighted voting (FWV) and dynamic weighted voting (DWV) were compared. The test set contained 1200 samples generated from the same distributions as the training data respectively (400 samples each). Note that the FWV scheme weighted the outputs by the accuracy of each corresponding classifier. By contrast, the DWV scheme calculated the weights based on the performance of each classifier on the *k* (*k*=7) nearest samples (among 1200 training samples) of the each test sample. The results of retraining using LDA were also presented.

Table 1 presents the predication errors after each batch of data was available (the test set was fixed). It is clear that the DWV scheme achieved the best performance in terms of learning from new training data. The retraining accuracy was not competitive because the distribution of the complete training data was too difficult for LDA to learn well.

Fig. 3 shows the combined decision boundaries resulted from FWV and DWV respectively. It is easy to see that when the distributions of training data change significantly as shown in Fig. 2, it is no long appropriate to use the fixed weights based on the training errors of classifiers as they cannot comfortably represent the performance of the classifiers on the entire dataset. On the other hand, the dynamic weighting scheme focuses on the local accuracy of each classifier for each unknown sample, which is more flexible and reliable.

Table 1: Error rates on the synthetic test set using different schemes

Schemes	Data 1	Data 2	Data 3
FWV	0.4952	0.4708	0.4448
DWV	0.4976	**0.0496**	**0.0576**
Retraining	-	-	0.5076

4.2. UCI Datasets

Additional experiments were conducted based on 8 popular benchmark datasets chosen from the UCI machine learning repository [21]. A summary of the datasets is given in Table 2.

Table 2: A summary of UCI datasets

Dataset	Attributes	Samples	Class
Sonar	60	208	2
Ionosphere	34	351	2
Breast	30	569	2
Credit Ger	24	1000	2
Credit Aus	14	690	2
Heart	13	270	2
Pima	8	768	2
Segmentation	19	2310	7

All datasets were normalized to the range [0, 1] in each dimension. For the 2-class datasets, each dataset was divided randomly into 3 subsets to be used as training sets. In order to make the test set representative of the real distribution of the data, the SMOTE [22] method is used to generate the test set with the same size as the original dataset. For each training session, only one set of training data was used, and an ensemble of classifiers with 20 basic learners (Decision Trees) were generated following the procedure of AdaBoost. In the second phase, kNN (k=7) was used to determine the dynamic weights for each basic classifier. The distances between the training data and the test samples were calculated using Euclidean distance. For comparison, experimental results using FWV in which each classifier was weighted according to its original weight assigned in the training process were also reported.

Table 3 shows the classification errors of two methods on the above UCI datasets. Note that the columns under "Data 1", "Data 2" and "Data 3" refer to the error rates after 1, 2 and 3 batches of data were available respectively (the test set was fixed). As far as the classification accuracy is concerned, after 3 batches of data were presented to the classifiers, the DWV scheme achieved better performance compared to the FWV scheme on all of the 7 two-class datasets. This evidence shows clearly that FWV assigned weights to each classifier with regard to each test sample more properly and consequently improved the classification accuracy. Note that more distinct differences between DWV and FWV are expected to be observed if the three batches of training data are deliberately made different instead of being generated randomly.

Finally, the *segmentation* dataset was used to evaluate the performance of DWV in situations where new classes may be introduced from the new training data. In the experiment, the original dataset was also divided into 3 training sets and 1 test set randomly. However, the first training set only contained samples from 5 classes, while the second one contained samples from 6 classes, and the last training set and test set covered all 7 classes. The results in Table 3 show again that DWV was able to handle this variable-class problem and achieved better accuracy compared to FWV after 3 batches of data. The classification error curves after each iteration (Fig. 4) also indicate that DWV can help the classifiers converge much faster than FWV in the changing environments.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we investigated the question of how to use ensemble algorithms to solve incremental learning problems where the training data become available in batches. The motivation was to show that ensemble algorithms, when applied smartly, are able to learn additional information from new data and preserve previously acquired knowledge.

Experimental results showed that the DWV scheme produced better overall performance compared to the FWV scheme, especially when the distributions of data change significantly over batches. It is also found that, by creating a separate ensemble for each new dataset, the complexity of the classification problem can be reduced compared to the retraining strategy.

In addition to the preliminary results presented here, there is still plenty of room for future research. For example, the potential of the proposed techniques can be further tested in situations where the training data may take different numbers of features, making it extremely challenging for the retraining strategy. In the meantime, more complex real world problems with incremental learning features can be also included in the empirical studies.

REFERENCES

[1] A. Gangardiwala and R. Polikar, Dynamically Weighted Majority Voting for Incremental Learning and Comparison of Three Boosting Based Approaches. *Proc. IJCNN*, Montreal, Canada, 2005, pp. 1131-1136.

[2] R. Polikar, L. Udpa, S. S. Udpa and V. Honavar, Learn++: An incremental learning algorithm for supervised neural networks, *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 31(4): 497-508, 2001.

[3] T. G. Dietterich, Machine Learning Research: Four Current Directions, *AI Magazine*, 18(4): 97-136, 1997.

[4] K. Yamauchi, N. Yamaguchi and N. Ishii, Incremental Learning Methods with Retrieving of Interfered Patterns, *IEEE Transactions on Neural Networks*, 10(6): 1351-1365, 1999.

[5] Y. Freund and R. E. Schapire, A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting, *Journal of computer and system sciences*, 55(1): 119-139, 1997.

[6] M. D. Muhlbaier and R. Polikar, An Ensemble Approach for Incremental Learning in Nonstationary Environments. *LNCS*, vol. 4472, 2007, pp. 490-500.

[7] D. Parikh and R. Polikar, An Ensemble-Based Incremental Learning Approach to Data Fusion, *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37(2): 437-450, 2007.

[8] M. Muhlbaier, A. Topalis and R. Polikar, Incremental Learning from Unbalanced Data. *Proc. IJCNN*, Budapest, Hungary, 2004, pp. 1057-1062.

[9] M. Muhlbaier, A. Topalis and R. Polikar, Learn++.MT: A New Approach to Incremental Learning, *Multiple Classifier Systems*, 52-61, 2004.

[10] M. D. Muhlbaier, A. Topalis and R. Polikar, Learn++.NC: Combining Ensemble of Classifiers With Dynamically Weighted Consult-and-Vote for Efficient Incremental Learning of New Classes, *IEEE Transactions on Neural Networks*, 20(1): 152-168, 2009.

[11] L. Breiman, Bagging Predictors, *Machine Learning*, 24(2): 123-140, 1996.

[12] L. Breiman, Random Forests, *Machine Learning*, 45(1): 5-32, 2001.

[13] M. I. Jordan and R. A. Jacobs, Hierarchical Mixtures of Experts and the EM Algorithm, *Neural Computation*, 6(2): 181-214, 1994.

[14] K. M. Ting and I. H. Witten, Issues in Stacked Generalization, *Journal of Artificial Intelligence Research*, 10: 271-289, 1999.

[15] A. Tsymbal and S. Puuronen, Bagging and Boosting with Dynamic Integration of Classifiers. *PKDD*, vol. 1910, *LNCS*. Lyon, France: Springer, 2000, pp. 116-125.

[16] A. Tsymbal, M. Pechenizkiy and P. Cunningham, Dynamic Integration with Random Forests. *ECML*, vol. 4212, *LNCS*. Berlin, Germany: Springer, 2006, pp. 801-808.

[17] R. Maclin, Boosting Classifiers Regionally. *Proc. AAAI*, Madison, WI, 1998, pp. 700-705.

[18] S. Kwek and C. Nguyen, iBoost: Boosting Using an Instance-Based Exponential Weighting Scheme. *ECML*, vol. 2430, *LNCS*. Helsinki, Finland: Springer, 2002, pp. 245-257.
[19] R. Jin, Y. Liu, L. Si, J. Carbonell and A. G. Hauptmann, A New Boosting Algorithm Using Input-Dependent Regularizer. *Proc. ICML*, Washington D.C., 2003.
[20] C.-X. Zhang and J.-S. Zhang, A Local Boosting Algorithm for Solving Classification Problems, *Computational Statistics & Data Analysis*, 52: 1928-1941, 2008.
[21] A. Asuncion and D. J. Newman, UCI Machine Learning Repository,http://www.ics.uci.edu/~mlearn/MLRepository.html. Irvine, CA: University of California, School of Information and Computer Science, 2007.
[22] N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer, SMOTE: Synthetic Minority Over-sampling Technique, *Journal of Artificial Intelligence Research*, 16: 321-357, 2002.

Table 3: The classification errors of ensembles with FWV and DWV on the UCI datasets

Dataset	**FWV**			**DWV**		
	Data 1	**Data 2**	**Data 3**	**Data 1**	**Data 2**	**Data 3**
Credit Ger	0.0990	**0.1050**	0.1140	0.0930	0.1090	**0.1020**
Pima	0.1979	0.1810	0.1719	0.2044	**0.1615**	**0.1393**
Credit Aus	0.1232	0.0797	0.0478	0.0870	**0.0319**	**0.0174**
Breast	0.0287	**0.0287**	0.0287	0.0335	**0.0287**	**0.0239**
Ionosphere	0.0712	**0.0769**	0.0769	0.0655	**0.0769**	**0.0655**
Heart	0.1852	0.2667	0.2593	0.1593	**0.2074**	**0.2444**
Sonar	0.1971	0.1779	0.1394	0.1731	**0.0817**	**0.0577**
Segmentation	0.2149	0.0314	0.0182	0.2116	**0.0248**	**0.0165**

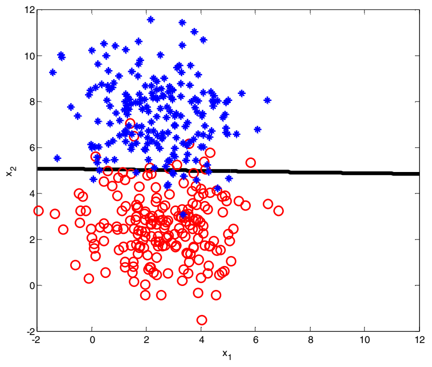
Fig. 2(a): The first batch of data and the classification boundary

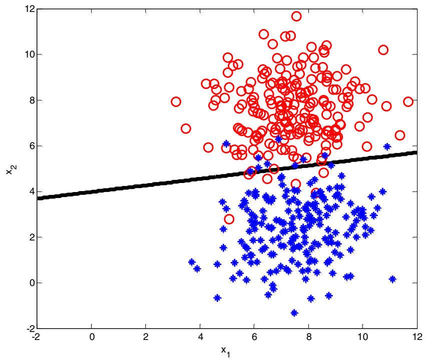
Fig. 2(b): The second batch of data and the classification boundary

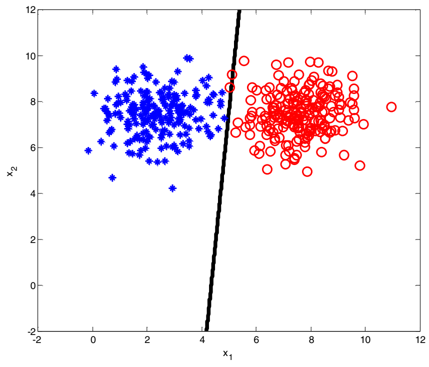
Fig. 2(c): The third batch of data and the classification boundary

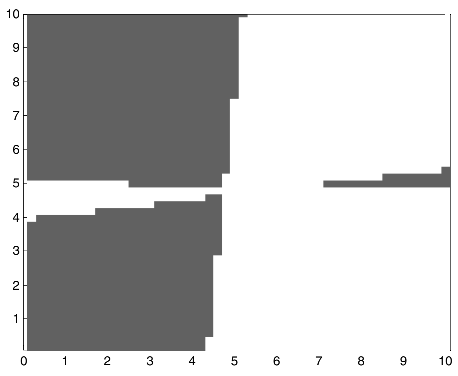
Fig. 3(a): Classification boundary by fixed weighted voting

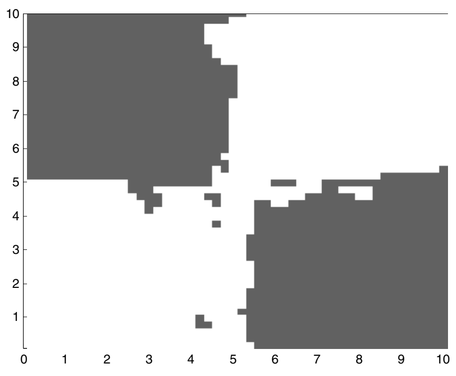
Fig. 3(b): Classification boundary by dynamic weighted voting

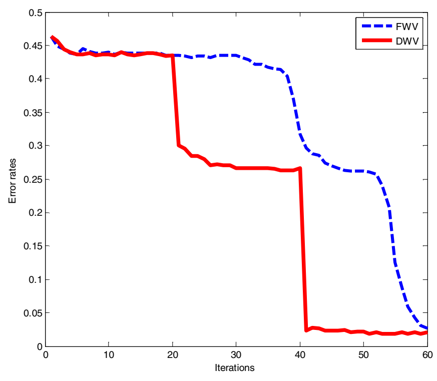

Fig. 4: Classification errors on Segmentation dataset