Dynamic Weighted Majority for Incremental Learning of Imbalanced Data Streams with Concept Drift*

Yang Lu[1], **Yiu-ming Cheung**[1,2], **Yuan Yan Tang**[3]
[1]Department of Computer Science, Hong Kong Baptist University, Hong Kong SAR, China
[2]HKBU Institute of Research and Continuing Education, Shenzhen, China
[3]Department of Computer and Information Science, University of Macau, Macau SAR, China
{yanglu,ymc}@comp.hkbu.edu.hk, yytang@umac.mo

Abstract

Concept drifts occurring in data streams will jeopardize the accuracy and stability of the online learning process. If the data stream is imbalanced, it will be even more challenging to detect and cure the concept drift. In the literature, these two problems have been intensively addressed separately, but have yet to be well studied when they occur together. In this paper, we propose a chunk-based incremental learning method called Dynamic Weighted Majority for Imbalance Learning (DWMIL) to deal with the data streams with concept drift and class imbalance problem. DWMIL utilizes an ensemble framework by dynamically weighting the base classifiers according to their performance on the current data chunk. Compared with the existing methods, its merits are four-fold: (1) it can keep stable for non-drifted streams and quickly adapt to the new concept; (2) it is totally incremental, i.e. no previous data needs to be stored; (3) it keeps a limited number of classifiers to ensure high efficiency; and (4) it is simple and needs only one thresholding parameter. Experiments on both synthetic and real data sets with concept drift show that DWMIL performs better than the state-of-the-art competitors, with less computational cost.

1 Introduction

In the online learning process, the underlying data distribution may change over time. This phenomenon is known as concept drift or non-stationary environment [Ditzler *et al.*, 2015]. To deal with it, one must confront the problem of stability-plasticity dilemma [Ditzler and Polikar, 2013]. That is, an algorithm has to balance the learning focus between the past information and the incoming distribution changes. Based on Bayes' theorem, the concept drift can occur in three of four components in the Bayes' formula and the rest one can be obtained by the other three [Ditzler *et al.*, 2015]:

- Data distribution $p(\mathbf{x})$: It is also called virtual drift. The distribution of $\mathbf{x}$ changes without affecting the decision hyperplane.
- Posterior probability $p(y|\mathbf{x})$: It is also called real drift. The decision hyperplane is shifted because the conditional probability changes.
- Class prior $p(y)$: It affects the imbalance ratio that the position of the minority class and the majority class may be switched.

These three types of drifts may occur separately or simultaneously at any time in the data stream. Let us take fault diagnosis as an example that the faulty samples are the minority class [Wang *et al.*, 2013b]. If the percentages of producing different types of samples change, it corresponds to the drift of $p(\mathbf{x})$. If the standard to diagnose a fault is improved so that more samples are identified as faults sometime, it corresponds to the drift of $p(y|\mathbf{x})$. Furthermore, if the cause of the fault is not fixed, the number of the faulty samples will increase and finally becomes the majority class, it corresponds to the drift of $p(y)$.

Classifying imbalanced data is another challenging problem [He and Garcia, 2009; Branco *et al.*, 2016]. Directly applying standard learning algorithm tends to ignore the minority class samples because of their subtle influence on the accuracy of an algorithm. Evidently, the problem will become more complicated if concept drift occurs together with class imbalance. When the concept of the minority class drifts, it is difficult to learn the new concept because the minority class samples seldom appear in the data stream. Thus far, only several methods have been proposed to tackle the problem of concept drift with class imbalance. In the literature, adaptive methods [Gao *et al.*, 2007; Wu *et al.*, 2014] and ensemble methods [Elwell and Polikar, 2011; Krawczyk *et al.*, 2017] are two major groups. The former updates the classifier on the current chunk, while the latter builds a model for each chunk and combines them as an ensemble. As for the issue of imbalance, adaptive methods collect the minority class samples in the past chunks and integrate them into the training samples in the current chunk for minority class augmentation. However, it is not always effective because the concept of the minority class may vary over time if

*Yiu-ming Cheung is the corresponding author. This work was supported by the National Natural Science Foundation of China with the Grant Numbers: 61672444 and 61272366, by the SZSTI Grant: JCYJ20160531194006833, and by the Faculty Research Grant of Hong Kong Baptist University (HKBU) with the Project Codes: FRG2/16-17/051 and FRG2/15-16/049.

concept drift of $p(y)$ occurs. By contrast, ensemble methods usually apply imbalance oriented learner as the base classifier in the ensemble. However, it is hard to weigh the base classifiers trained on different timestamps and maintain a limited number of classifiers to prevent the number of base classifiers infinitely increasing.

In this paper, we propose a new chunk-based incremental learning method called Dynamic Weighted Majority for Imbalance Learning (DWMIL) to deal with binary class imbalanced data streams with concept drift. It utilizes an ensemble framework with dynamic base classifier weighting. The classifier weights depend on the imbalance aware error of the classifier tested on the current chunk. The classifier with smaller error will receive higher weight in the ensemble because it may be trained on the chunk with similar concept as the current one. The classifier weights are naturally reduced over time based on the accumulated testing error. The advantages of DWMIL are four-fold:

- It can remain stable for stationary streams and quickly adapt to the new concept for non-stationary environment.
- The method is totally incremental. No previous data needs to be stored to help prediction. Therefore, the method does not suffer from memory problems.
- The number of stored classifiers is much less than the current timestamp. The dated classifiers will be discarded so that the ensemble size will not expand infinitely.
- It is simple and efficient. Only one thresholding parameter is needed. The computational cost is very low compared with the other methods.

The main contributions of this paper are summarized below:

- We propose an effective and efficient online learning method DWMIL for imbalanced data stream classification with concept drift.
- Dynamic Weighted Majority (DWM) is extended to the chunk-by-chunk version with more robust performance.

2 Related Work

For adaptive methods, Gao et al. [Gao *et al.*, 2007] proposed to accumulate the minority class samples in the current chunk with all past chunks, while carrying on undersampling on the majority class samples to balance the classes. A bagging like ensemble framework is then used for classification. However, it suffers from memory limitation to store the past data and lacks of the ability to quickly adapt to the new concept. As an improvement, SERA [Chen and He, 2009] and REA [Chen and He, 2011] select only parts of the minority class samples from the past chunks based on the similarity to the minority class samples in the current chunk. Moreover, HUWRS.IP [Hoens and Chawla, 2012] calculates the Hellinger distance between the samples in the current and the past chunks to detect the concept drift. The distance is used as the weight of the base classifier built on different feature subspaces in the ensemble. To handle the imbalance problem, HUWRS.IP creates a Naïve Bayes classifier on the current chunk to select the similar minority class samples from the past chunks. DFGW-IS [Wu *et al.*, 2014] further improves HUWRS.IP by incorporating the importance sampling technique to collect the positive class samples from the past chunks. In summary, these methods assume that the minority class does not change, and they can therefore utilize the past information. However, from the practical viewpoint, if $p(y)$ changes over time, the past minority class may not be the same as the current minority class.

The ensemble methods usually adopt an incremental fashion, i.e. the data in the past is not saved, because the information of the past data is kept by the base classifiers in the ensemble. Learn++.CDS and Learn++.NIE [Ditzler and Polikar, 2013] create one base classifier for each chunk and use the ensemble to predict the incoming data. The base classifiers are weighed by a time decay function and their performance on the current chunk. ESOS-ELM [Mirza *et al.*, 2015] uses the ensemble of extreme learning machine, where the weight matrices trained on each chunk are stored for ensemble. When the concept drift is detected by a change detector, the ensemble model will be reinitialized.

The methods mentioned above consider the concept drifts provided that the imbalance ratio is fixed. On the other hand, the effect of class prior drift for online learning from imbalanced data has been studied by some researchers. Wang et al. studied the effect of imbalance ratio change and proposed to adjust the training process by dynamic sampling rates [Wang *et al.*, 2015; 2016]. CBCE [Sun *et al.*, 2016] focuses the class evolution problem for multi-class classification. It uses one-versus-rest strategy to create one classifier for each class, and conducts undersampling for the majority class. [Wang *et al.*, 2013a] is the only work to study and analyze the case that class prior drift happens together with the other kinds of concept drift, where a drift detector is proposed with an imbalance detector to deal with the joint concept drift.

3 Proposed Method

At timestamp $t-1$, DWMIL maintains m classifiers in the set $\mathcal{H}^{(t-1)} = \{H_1^{(t-1)}, ..., H_m^{(t-1)}\}$ trained on the data chunks from timestamp 1 to $t-1$. When DWMIL receives a new data chunk $\mathcal{D}^{(t)}$ at timestamp t, it learns a new classifier H on the current data chunk and merges H with $\mathcal{H}^{(t-1)}$ to form $\mathcal{H}^{(t)}$. The classifiers trained on each chunk are associated with the vector of weights, denoted as $\mathbf{w}^{(t)} = [w_1^{(t)}, ..., w_m^{(t)}]^T$, which measures the importance of the classifiers in the ensemble. When the new classifier H is created, its initial weight is set at 1 and m is increased by 1. In order to adapt to new concept and make the previously learned classifiers less influential in the ensemble, the weight $w_j^{(t)}$ for classifier $H_j^{(t)}$ is reduced on each timestamp after it is created:

$$w_j^{(t)} = (1 - \epsilon_j^{(t)}) w_j^{(t-1)},$$

where $j = 1, ..., m-1$ and $\epsilon_j^{(t)}$ is the testing error of $H_j^{(t)}$ on the current data chunk $\mathcal{D}^{(t)}$. The error $\epsilon_j^{(t)}$ can be calculated by any error function like F1 or G-mean. Thus, the weights of the classifiers trained on the past chunks are reduced based

on the their performance on the current data chunk. Since this weight reduction is accumulated over time, the weight $w_j^{(t)}$ is actually equals to:

$$w_j^{(t)} = \prod_{\tau=l+1}^{t} (1 - \epsilon_j^{(\tau)}), \quad (1)$$

where l is the timestamp when $H_j^{(t)}$ is created. As $1 - \epsilon_j^{(\tau)} \leq 1$, the classifier weight is getting smaller over time according to its error on each chunk after it is created. Then, the classifiers with weight less than the threshold θ are removed and the counter m is also reduced according to the number of classifiers left. If a classifier is going to be removed, there are two factors that make its weight lower than θ. One is that the classifier is trained on a very early timestamp that makes the production in Equation (1) small. The other is that the concept changes in recent chunks and the testing error of the classifier is large on those chunks. Thus, this kind of classifier is less likely to provide positive effects to the prediction on the current and following chunks. Finally, the model predicts the incoming data $\mathbf{x}$ in $\mathcal{D}^{(t+1)}$ by the ensemble of $\mathcal{H}^{(t)}$ associated with $\mathbf{w}^{(t)}$:

$$sign(\sum_{j=1}^{m} w_j^{(t)} H_j^{(t)}(\mathbf{x})).$$

The training process of the proposed method DWMIL is shown in Algorithm 1. In DWMIL, we use UnderBagging as the base learner to train imbalanced data as shown in Algorithm 2. In each bagging iteration, we carry on undersampling on the majority class to make the training data balanced.

The relations between DWMIL and the existing methods are discussed as follows:

- Relation with Dynamic Weighted Majority (DWM) [Kolter and Maloof, 2007]: Both DWMIL and DWM keep a number of classifiers during learning the data streams. Despite the use of the imbalance oriented classifier UnderBagging as the base classifier in DWMIL, DWMIL has some other improvements over DWM. DWM processes the data one-by-one and updates the classifiers on every p incoming samples. If the data stream is imbalanced, the occurrence frequency of the minority class sample will be very low. Thus, updating a long sequence of majority class sample will be likely to make the model biased. By contrast, processing the imbalanced data stream chunk-by-chunk as DWMIL does will be more stable. DWM decides to create a new classifier according to the prediction performance of a single sample. If the data is imbalance, the probability that the sample belongs to the majority class is much higher than the minority class, and the probability to misclassify a majority class sample is low. Thus, the chance to create new classifiers on imbalanced data stream is low. It turns out that it may not efficiently adapt to the new concept. In comparison, DWMIL creates a new classifier for each chunk to learn the new concept in time. Besides, in DWM, the weights are reduced by a fixed parameter β and actually reduced again after normalization. Instead, DWMIL reduces the weight based on the performance without any normalization. Thus, a classifier can last longer to contribute to the prediction if the current concept is similar to the one when the classifier is created.
- Relation with Learn++ framework [Elwell and Polikar, 2011]: Both Learn++ and DWMIL create classifiers for each chunk and use classification error to adjust the weights. However, Learn++ uses a time decay function σ to reduce the weights of the classifiers trained on the past chunks. This σ depends on two free parameters: a and b, where different values of them will produce diverse results. In DWMIL, the weight reduction only depends on the performance of the classifiers without free

Algorithm 1 Train_DWMIL

Input: Data chunk at timestamp t: $\mathcal{D}^{(t)} = \{\mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}\}$, $i = 1, ..., N$, threshold for deleting base classifiers θ, base classifier set $\mathcal{H}^{(t-1)} = \{H_1^{(t-1)}, ..., H_m^{(t-1)}\}$, weight of base classifiers $\mathbf{w}^{(t-1)}$, number of base classifiers m, ensemble size T.

1: **for** $i \leftarrow 1$ to N **do**
2: Predict $\mathbf{x}_i$ by the ensemble classifier: $\bar{y}_i \leftarrow sign(\sum_{j=1}^{m} w_j^{(t-1)} H_j^{(t-1)}(\mathbf{x}_i))$;
3: **end for**
4: **for** $j \leftarrow 1$ to m **do**
5: Calculate the error $\epsilon_j^{(t)}$ for classifier $H_j^{(t-1)}$ on $\mathcal{D}^{(t)}$;
6: Update weight of base classifiers: $w_j^{(t)} \leftarrow (1 - \epsilon_j^{(t)}) w_j^{(t-1)}$;
7: **end for**
8: Remove classifiers with weights less than θ: $\mathcal{H}^{(t)} \leftarrow \mathcal{H}^{(t-1)} \backslash \{H_j^{(t-1)} | w_j^{(t)} < \theta\}$; $m \leftarrow |\mathcal{H}^{(t)}|$;
9: Create new base classifier and initialize its weight: $m \leftarrow m + 1$; $H \leftarrow UnderBagging(\mathcal{D}^{(t)}, T)$; $\mathcal{H}^{(t)} \leftarrow \mathcal{H}^{(t)} \cup H$; $w_m^{(t)} \leftarrow 1$;

Output: Base classifier set $\mathcal{H}^{(t)}$, weight of base classifiers $\mathbf{w}^{(t)}$, number of base classifiers m, prediction $\bar{\mathbf{y}}$.

Algorithm 2 UnderBagging

Input: Data $\mathcal{D} = \{\mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}\}$, $i = 1, ..., N$, the number of positive samples N_p, the number of negative samples N_n, ensemble size T.

1: **for** $t \leftarrow 1$ to T **do**
2: **if** $N_p < N_n$ **then**
3: $N_s \leftarrow N_p$;
4: **else**
5: $N_s \leftarrow N_n$;
6: **end if**
7: $\mathcal{D}_p \leftarrow$ Bootstrap N_s positive samples;
8: $\mathcal{D}_n \leftarrow$ Bootstrap N_s negative samples;
9: $h_t \leftarrow BaseLearner(\{\mathcal{D}_p, \mathcal{D}_n\})$;
10: **end for**

Output: Base classifier $H(\mathbf{x}) = sign(\sum_{t=1}^{T} h_t(\mathbf{x}))$;

parameters. Furthermore, in Learn++, the weights depend not only on the current chunk, but also the chunks from the classifier that is created to the current chunk. Under the circumstances, it may produce bias. Specifically, if one classifier performs very well on the chunk where it is created, it will receive continuously higher weights in the following several chunks. If the concept changes, the high weight of the classifier trained on old concept will hinder the prediction. In addition, Learn++ keeps all classifiers over time. The accumulated classifiers will increase the computational burden if the data stream is very long or even life long, because it needs to evaluate the performance of all past classifiers on the current chunk. By contrast, DWMIL discards the dated or useless classifiers to increase the computational efficiency.

4 Experimental Results

To evaluate the effectiveness and efficiency of DWMIL[1], we compare it with four other state-of-the-art methods, which are designed for imbalanced data stream with concept drift, on four synthetic data sets and two real data sets.

4.1 Experiment Settings

The compared methods and their settings are described as follows. Learn++.NIE (LPN) [Ditzler and Polikar, 2013] and Recursive Ensemble Approach (REA) [Chen and He, 2009] are selected as a representative of ensemble methods and adaptive methods, respectively. Class-Based ensemble for Class Evolution (CBCE) [Sun *et al.*, 2016] with drift detector DDM [Wang *et al.*, 2013a] is selected as the representative of active drift detection methods. We also compare the original version of Dynamic Weighted Majority (DWM) [Kolter and Maloof, 2007], which is not essentially designed for imbalanced data stream. Accordingly, we adopt the Undersampling Online Bagging (UOB) technique [Wang *et al.*, 2015] for DWM. The parameters of the compared methods are set at the recommended values in the literature. For DWMIL, LPN and DWM, the number of base classifiers in the ensemble T is set at 11 to prevent draw result. For DWMIL, the threshold to remove the dated classifier θ is set at 0.001. As discussed in [Kolter and Maloof, 2007], the value of θ has nearly no influence to the accuracy, and it only affects the number of stored classifiers. Geometric mean (G-mean) error $\epsilon_{gm} = 1 - \sqrt{TPR \cdot TNR}$ is chosen as the error function used in LPN and DWMIL, where TPR is the true positive rate and TNR is the true negative rate. In fact, any other error functions like F1 measure can also be used as the error function. Here, we only adopt G-mean error as the error function due to space limitation. CART [Breiman *et al.*, 1984] is used as the learner of the base classifier for all methods. All results are averaged by 10 runs of the experiments. We use G-mean and Area Under Curve (AUC) as the evaluation metrics to compare all methods. The test-then-train strategy is adopted to evaluate the performance of the methods on each chunk.

[1]Source code and data sets used in the experiments can be obtained from https://github.com/lylylytc/dwmil

data set	#samples	#features	#chunks
Moving Gaussian	50,000	2	50
SEA	100,000	3	100
Hyper Plane	100,000	10	50
Checkerboard	60,000	2	200
Electricity	27,549	7	56
Weather	18,159	8	50

Table 1: Information of six streaming data sets.

4.2 Data Sets

In the experiments, we utilize four synthetic data sets and two real data sets, whose information is summarized in Table 1. The percentage of the minority class samples in each chunk is fixed at 5% of the chunk size.

Moving Gaussian: This data stream consists of two Gaussian distributed classes with identity covariance and 2 dimensions. The initial coordinates of the mean of the two classes are [5,0] and [7,0]. The two classes gradually move to [-5,0] and [-3,0] from the first to the last chunk.

SEA [Street and Kim, 2001]: It contains three attributes ranging from 0 to 10. Only the first two attributes are related to the class that is determined by $attr_1 + attr_2 \leq \alpha$. The third attribute is treated as noise. The control parameter α is set at 15 for the first 1/3 and the last 1/3 chunks, and 5 for the second 1/3 chunks.

Hyper Plane [Street and Kim, 2001]: The gradually changed concepts are calculated by $f(\mathbf{x}) = \sum_{i=1}^{d-1} a_i \cdot \frac{x_i + x_{i+1}}{x_i}$, where the dimension $d = 10$ and a_i is used to control the decision hyperplane.

Checkerboard [Ditzler and Polikar, 2013]: It is a nonlinear XOR classification problem. The data stream is produced by selecting from a size-fixed window in the rotating checkerboard.

Electricity [Kolter and Maloof, 2007]: This data set contains the changes of electricity price according to the time and demand in New South Wales, Australian. The class label is determined by the change of price over the last 24 hours.

Weather [Ditzler and Polikar, 2013]: This data set contains the weather information over 50 years in Bellevue and Nebraska. The task is to predict whether a day is rainy or not.

4.3 Comparative Results

The G-mean and AUC performance on each chunk is shown in Figure 1 and 2. The observation of the experiments on each data set is discussed as follows. The results of data set Moving Gaussian are shown in Figure 1(a) and 2(a). The performance of DWMIL and DWM are comparable through all chunks in terms of G-mean and both of them perform better than LPN. CBCE predicts all samples as negative and therefore receives 0 in G-mean. A plausible reason is that the data set of Moving Gaussian is continuously drifting. Thus, CBCE resets its model on every several samples so that it poorly learns the minority class samples. The performance of REA gradually drops to 0 because the stored past minority class samples are overlapped with the majority class samples in

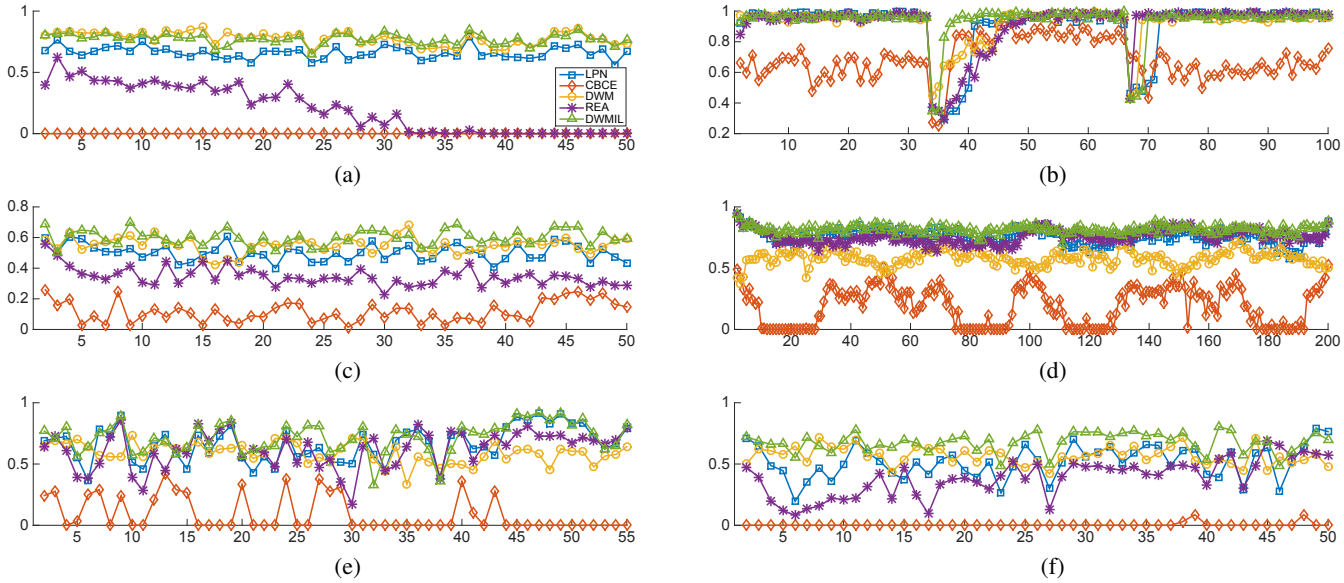

Figure 1: G-mean performance of each chunk on data set (a) Moving Gaussian, (b) SEA, (c) Hyper Plane, (d) Checkerboard, (e) Electricity and (f) Weather.

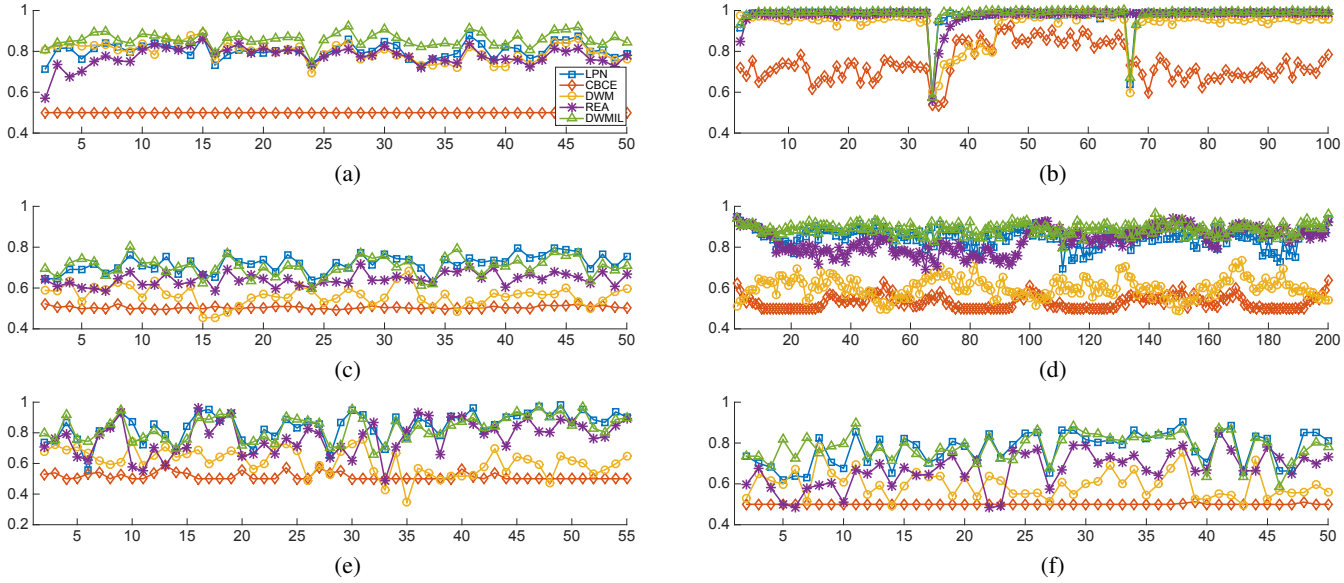

Figure 2: AUC performance of each chunk on data set (a) Moving Gaussian, (b) SEA, (c) Hyper Plane, (d) Checkerboard, (e) Electricity and (f) Weather.

the current chunk due to the distribution moving. Incorporating them into the training set equals to adding noises to the minority class. In terms of AUC, DWMIL shows better performance than the others along all chunks. The results of data set SEA are shown in Figure 1(b) and 2(b). The two significant performance drops correspond to the drifts of the decision hyperplane. In terms of G-mean, DWMIL recovers its performance quickly on the first drift compared with the other methods, which shows the ability of DWMIL to adapt to the new concept. DWM also responds quickly, but it spends around 15 chunks to recover the performance. LPN stays at the valley for around 7 chunks before starting to recover the performance. In terms of AUC, DWMIL and LPN show comparable results and recovery speed. The AUC performance of REA shows good ability to resist concept drift by the stored samples. It is not influenced by the second drift. The results of data set Hyper Plane are shown in Figure 1(c) and 2(c). DWMIL shows slightly better performance than the DMW in terms of G-mean. The AUC performance of DWMIL is close to LPN and both of them are better than DWM, REA and CBCE. The results of data set CheckerBoard are shown in Figure 1(d) and 2(d). In terms of G-mean, DWMIL, REA

Data set	LPN	CBCE	DWM	REA	DWMIL
Moving Gaussian	0.6419±0.0314	0.0000±0.0000	**0.7783±0.0447•**	0.2003±0.0358	0.7565±0.0205
SEA	0.8907±0.0095	0.6862±0.0141	0.9201±0.0211	0.9022±0.0156	**0.9256±0.0080**
Hyper Plane	0.4941±0.0455	0.1173±0.0496	0.5511±0.0223	0.3390±0.0555	**0.5889±0.0342•**
Checkerboard	0.7507±0.0152	0.1931±0.0124	0.5820±0.0477	0.7546±0.0203	**0.8123±0.0138•**
Electricity	0.6510±0.0438	0.0903±0.0153	0.5914±0.0436	0.6095±0.0546	**0.7062±0.0409•**
Weather	0.5174±0.0708	0.0041±0.0075	0.5672±0.0619	0.3820±0.1286	**0.6641±0.0566•**

Table 2: G-mean performance averaged on all chunks. The boldface indicates the best result and the symbol • indicates significant difference with the second best result by Wilcoxon signed rank test with 95% confidence interval.

Data set	LPN	CBCE	DWM	REA	DWMIL
Moving Gaussian	0.7970±0.0149	0.5000±0.0000	0.7982±0.0167	0.7697±0.0319	**0.8517±0.0138•**
SEA	0.9725±0.0021	0.7444±0.0087	0.9355±0.0167	0.9705±0.0043	**0.9776±0.0027•**
Hyper Plane	**0.7053±0.0171**	0.5048±0.0051	0.5635±0.0163	0.6396±0.0323	0.7007±0.0216
Checkerboard	0.8477±0.0091	0.5313±0.0027	0.6016±0.0319	0.8331±0.0170	**0.8876±0.0109•**
Electricity	**0.8311±0.0180**	0.5141±0.0036	0.6116±0.0397	0.7656±0.0428	0.8271±0.0273
Weather	0.7683±0.0248	0.5004±0.0011	0.5961±0.0432	0.6676±0.0573	**0.7725±0.0309**

Table 3: AUC performance averaged on all chunks. The boldface indicates the best result and the symbol • indicates significant difference with the second best result by Wilcoxon signed rank test with 95% confidence interval.

and LPN produce similar overall results. However, it can be observed that DWMIL is more stable against concept drifts, while the AUC performance of LPN and REA appear to have small drops frequently. The results of data set Electricity and Weather are shown in Figure 1(e-f) and 2(e-f), respectively. DWMIL shows generally better performance than LPN and REA in terms of G-mean, and comparable performance with them in terms of AUC.

In order to compare the overall performance, the averaged numerical results over all chunks are shown in Table 2 and 3. The significance test is conducted between the best and the second best result by Wilcoxon signed rank test with 95% confidence interval. It can be observed that, for G-mean, DWMIL achieves significantly better results than the other methods on four data sets. For AUC, DWMIL significantly outperforms the other methods on three data sets and has comparable results with LPN without significant difference on the other three data sets. The numerical results show that DWMIL can generally produce better results than the other methods on the data sets with the different kinds of concept drifts.

4.4 Efficiency Analysis

The running time for the compared methods on six data sets is shown in Table 4. The computational cost of DWMIL is the lowest compared with the other methods on four data sets. REA has the lowest cost on data sets Electricity and Weather. The reason is that DWMIL adopts bagging with T decision trees as the base classifier on each chunk while REA adopts single decision tree. Thus, REA is faster for the data sets with a smaller number of samples and chunks. However, REA keeps all the classifiers trained on each chunk and searches kNN of all past minority class samples, thus it is slow when the number of chunks is large. For example, on Checkerboard which has 200 chunks, the cost of REA is around five times as high as the cost of DWMIL. Besides, LPN is generally the third fastest but spends almost double time than DWMIL. It also keeps all the classifiers trained on each chunk and the prediction cost will be increasingly higher over time. CBCE and DWM are the slowest because they are one-by-one learning methods. DWM is even slower than CBCE because it updates an ensemble of classifiers for each incoming sample, while CBCE only updates two binary classifiers.

Data set	LPN	CBCE	DWM	REA	DWMIL
Moving Gaussian	18.6	32.2	194.6	11.7	**10.9**
SEA	49.0	133.6	207.1	44.6	**34.5**
Hyper Plane	22.1	114.4	324.2	14.5	**11.8**
Checkerboard	110.5	277.5	146.1	282.0	**57.8**
Electricity	20.2	37.5	84.5	**5.4**	12.1
Weather	17.4	39.3	76.7	**4.4**	15.2

Table 4: The running time (s) of compared methods on each data set. The boldface indicates the best result.

5 Conclusion

From the practical viewpoint, concept drift and class imbalance are two inevitable problems of learning from data streams. The composition of these two phenomenons will make learning from data stream very challenging. In this paper, we have proposed DWMIL to solve the problem of learning from imbalanced data stream with concept drift. It creates a base classifier for each chunk and weigh them by their performance tested on the current chunk. Thus, a classifier trained recently or on the similar concept as the current chunk will receive high weight in the ensemble to help prediction. Experiments on concept drift data sets have shown that DWMIL performs better and more efficiently compared with its counterparts.

References

[Branco *et al.*, 2016] Paula Branco, Luís Torgo, and Rita P Ribeiro. A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys (CSUR)*, 49(2):31, 2016.

[Breiman *et al.*, 1984] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[Chen and He, 2009] Sheng Chen and Haibo He. Sera: selectively recursive approach towards nonstationary imbalanced stream data mining. In *Proceedings of the 2009 International Joint Conference on Neural Networks*, pages 522–529. IEEE, 2009.

[Chen and He, 2011] Sheng Chen and Haibo He. Towards incremental learning of nonstationary imbalanced data stream: a multiple selectively recursive approach. *Evolving Systems*, 2(1):35–50, 2011.

[Ditzler and Polikar, 2013] Gregory Ditzler and Robi Polikar. Incremental learning of concept drift from streaming imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 25(10):2283–2301, 2013.

[Ditzler *et al.*, 2015] Gregory Ditzler, Manuel Roveri, Cesare Alippi, and Robi Polikar. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25, 2015.

[Elwell and Polikar, 2011] Ryan Elwell and Robi Polikar. Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks*, 22(10):1517–1531, 2011.

[Gao *et al.*, 2007] Jing Gao, Wei Fan, Jiawei Han, and Philip S. Yu. A general framework for mining concept-drifting data streams with skewed distributions. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 3–14. SIAM, 2007.

[He and Garcia, 2009] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[Hoens and Chawla, 2012] Thomas Ryan Hoens and Nitesh V Chawla. Learning in non-stationary environments with class imbalance. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 168–176. ACM, 2012.

[Kolter and Maloof, 2007] J. Zico Kolter and Marcus A. Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *Journal of Machine Learning Research*, 8:2755–2790, 2007.

[Krawczyk *et al.*, 2017] Bartosz Krawczyk, Leandro L Minku, João Gama, Jerzy Stefanowski, and Michał Woźniak. Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37:132–156, 2017.

[Mirza *et al.*, 2015] Bilal Mirza, Zhiping Lin, and Nan Liu. Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift. *Neurocomputing*, 149:316–329, 2015.

[Street and Kim, 2001] W Nick Street and YongSeog Kim. A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 377–382. ACM, 2001.

[Sun *et al.*, 2016] Yu Sun, Ke Tang, Leandro L Minku, Shuo Wang, and Xin Yao. Online ensemble learning of data streams with gradually evolved classes. *IEEE Transactions on Knowledge and Data Engineering*, 28(6):1532–1545, 2016.

[Wang *et al.*, 2013a] Shuo Wang, Leandro L Minku, Davide Ghezzi, Daniele Caltabiano, Peter Tino, and Xin Yao. Concept drift detection for online class imbalance learning. In *Proceedings of the 2013 International Joint Conference on Neural Networks*, pages 1–10. IEEE, 2013.

[Wang *et al.*, 2013b] Shuo Wang, Leandro L Minku, and Xin Yao. Online class imbalance learning and its applications in fault detection. *International Journal of Computational Intelligence and Applications*, 12(04):1 340 001(1–19), 2013.

[Wang *et al.*, 2015] Shuo Wang, Leandro L Minku, and Xin Yao. Resampling-based ensemble methods for online class imbalance learning. *IEEE Transactions on Knowledge and Data Engineering*, 27(5):1356–1368, 2015.

[Wang *et al.*, 2016] Shuo Wang, Leandro L Minku, and Xin Yao. Dealing with multiple classes in online class imbalance learning. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 2118–2124. IJCAI/AAAI, 2016.

[Wu *et al.*, 2014] Ke Wu, Andrea Edwards, Wei Fan, Jing Gao, and Kun Zhang. Classifying imbalanced data streams via dynamic feature group weighting with importance sampling. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 722–730. SIAM, 2014.