An Automatic Construction and Organization Strategy for Ensemble Learning on Data Streams

Yi Zhang
School of Software
Tsinghua University, Beijing, 100084 China
zhang-yi@mails.tsinghua.edu.cn

Xiaoming Jin
School of Software
Tsinghua University, Beijing, 100084 China
xmjin@tsinghua.edu.cn

ABSTRACT

As data streams are gaining prominence in a growing number of emerging application domains, classification on data streams is becoming an active research area. Currently, the typical approach to this problem is based on ensemble learning, which learns basic classifiers from training data stream and forms the global predictor by organizing these basic ones. While this approach seems successful to some extent, its performance usually suffers from two contradictory elements existing naturally within many application scenarios: firstly, the need for gathering sufficient training data for basic classifiers and engaging enough basic learners in voting for bias-variance reduction; and secondly, the requirement for significant sensitivity to concept-drifts, which places emphasis on using recent training data and up-to-date individual classifiers. It results in such a dilemma that some algorithms are not sensitive enough to concept-drifts while others, although sensitive enough, suffer from unsatisfactory classification accuracy. In this paper, we propose an ensemble learning algorithm, which: (1) furnishes training data for basic classifiers, starting from the up-to-date data chunk and searching for complement from past chunks while ruling out the data inconsistent with current concept; (2) provides effective voting by adaptively distinguishing sensible classifiers from the else and engaging sensible ones as voters. Experimental results justify the superiority of this strategy in terms of both accuracy and sensitivity, especially in severe circumstances where training data is extremely insufficient or concepts are evolving frequently and significantly.

1. INTRODUCTION

In many emerging applications such as network monitoring, sensor networks, etc., data are produced continually in the form of high-speed streams, which are required to be analyzed on-line. Thus, the applications which aim to classifying data streams rather than static relations are needed. Given the fact that data streams always have the properties such as high-velocity, extremely large volume, and frequently evolving concepts, today's classification techniques meet unprecedented challenges: bounded memory usage, high processing speed, one-pass scanning, any-time available, and so on [4]. Especially, underlying concept of steaming data often alters (termed *concept drift*), which requests that algorithms must be sensitive enough to the up-to-date concept under the data stream [4, 13].

Many strategies have been proposed in order to deal with concept-drifts. For instance, adapting existent models to data streams scenarios [7]; using novel data structure to maintain training data stream and to classify on demand [1]; exhaustively selecting training data by comparing all the sensible choices [5]; or building concept history and combining proactive and reactive modes in prediction [15]. Besides these technologies, the ensemble learning approach [2] appears as a promising solution: it seems reasonable to train individuals to deal with different parts of stream and organize these individual classifiers to make the final decision. This motivates more than a few attempts to develop novel ensemble learning mechanisms for data streams [9, 11, 12, 16]. However, all these models, although effective to some extent, do not provide satisfying solution to some open problems, due to the difficulties of: (1) seeking enough training data for individual classifiers with the guarantee that not importing old concepts; (2) finding adequate voters in global-prediction, while ensuring that experts (i.e. basic classifiers) built upon old concepts are excluded. We discuss these aspects as follows:

Firstly, when building each basic classifier, we want to collect enough data while guarantee that concept-drifts are not imported into training data. To handle this problem, some works split the training data stream into data chunks, and build basic learner from each chunk [11, 12, 16]; while other works use incremental learner as the basic expert, i.e. each expert, after being built, keeps on updating itself until discarded [9]. In fact, both of these two methods can not furnish ideal solution. On the one hand, fixing the amount of training data for basic classifier by size of chunk is questionable. Given the fact that the velocity of training data stream is often limited by the manual labeling process, the size of data chunk can not be very large because large chunk needs relatively long period to be accumulated, thus leads to high possibility that concept-drift happens in this period. Nonetheless, if basic classifiers can not obtain sufficient training data, the ensemble will not work effectively. On the other hand, using incremental classifier

also suffers from some flaws. It is true that allowing each individual expert to adjust itself according to future training data is beneficial to this individual [9]. But this approach has negative effects on the whole ensemble: when an old learner is incompatible with the latest concept, the most optimal policy is discarding it and allowing the "right ones" to make decision, rather than adjusting (if possible) the elder, which actually postpones its retirement. Moreover, though incremental learning gives the individual the chance for improving itself, the bias can not be completely corrected in that the learner is built from old data and merely "update" itself based on newcome data.

Secondly, when using basic classifiers to form global predictor, we want to engage adequate voters in final decision for the sake of bias-variance reduction [2], while ensure that outmoded classifiers are obviated. Although recent works place much stress on this point, none of them can make good balance. In [11], the global prediction is made by majority voting among N "high quality" individuals. The drawback of this method is clear-cut: Only after more than $N/2$ members in ensemble mastery the new concept (which needs at least $N/2$ new data chunks after concept-drift occurs), the majority voting will make correct prediction. Thereafter, some works focus on improving voting's sensitivity to concept-drift [9, 12]. For example in [9]: (1) the ensemble is composed of classifiers whose "quality" larger than an threshold q_0 rather than uses fixed amount of basic classifiers; (2) The global prediction is based on weighted voting rather than majority voting. Although this approach improves ensemble's sensitivity to concept-drift, it still has problems. First of all, q_0 is difficult to choose: we want good voters, but we also need enough voters. Second of all, weight-based voting can not eliminate the negative effect of out-of-date experts ---- they still can overwhelm the sensible ones by larger total weight. Since neither of majority voting and weight-based voting can produce sensitive ensemble, the "apparently" substituted way is "trusting in" the best rather than voting by the masses [16]. Whereas, simply engaging the best classifier will lose important advantage of voting-based ensemble: bias and variance reduction [2]. In fact, when using some unstable learners such as C4.5 [10], voting-based ensemble such as bagging can improve the accuracy by dramatically reducing variance [2, 3]. Even for stable classifiers such as naïve Bayes [8], voting strategy as boosting [6] has positive effect by decreasing bias [2].

In this paper, we propose a dynamic ensemble learning algorithm, termed Dynamic Construction and Organization (DCO), which concentrates on these two difficulties. The contributions and key ideas of this work are: (1) the *individual-construction strategy* which provides training data for basic classifiers, starting from the latest data chunk and searching complement from history while excluding the data inconsistent with current concept; (2) the *global-prediction policy* which offers effective voting by adaptively differentiating between sensible experts and the else and engaging sensible ones as voters. Experimental results show that our ensemble approach achieves high accuracy and remains sensitivity to concept-drifts.

This paper is organized as follows. Section 2 describes our approach, section 3 provides the experimental results, and section 4 concludes the paper.

2. Dynamic Construction and Organization Strategy for Ensemble Learning

In this section, we put forward our DCO (Dynamic Construction and Organization) approach. After introducing the problem definition and framework of the algorithm, we mainly focus on the individual-construction and global-prediction strategies. It is assumed that training data and testing data are given as data streams, termed S and T in our paper, respectively. Data items in S are divided into data chunks, with size of *chunkSize*. As a rule, we set the latest chunk from S as evaluation dataset. When future chunk is available, current evaluation dataset can be used as training chunk and the coming chunk is set as new evaluation dataset. The algorithm framework is: (1) when a new training chunk is available, we use *individual-construction* strategy to create a new basic classifier from this chunk plus the old chunks; (2) we set the most recent N basic classifiers as the ensemble; (3) for each test point, we use the ensemble to classify the data based on *global-prediction* strategy.

2.1 Individual-Construction Strategy

Table 2 shows our ***Individual-Construction Strategy*** which pursues a balance between data sufficiency and sensitivity, especially when single chunk is not enough for training basic learner. Function ***create*** is depend on the basic learner. In this paper, we have tested both C4.5 [10] and naïve Bayes [8], see section 3 for details. What is more, there are two additional functions, ***dataSelect*** and ***outperform***, discussed in following subsections.

Table 2. Individual-construction strategy

Input:
  $D_n, D_{n-1}, \ldots D_1$: data chunks available
Output:
  C_n: resulting new expert
Variable:
  D: training data for new basic learner
  $\triangle$:selected data from old chunk
  C_n': alternative expert

$D \leftarrow D_n$
$C_n \leftarrow$ ***create*** (D)
for $i = n-1$ to 1
  $\triangle \leftarrow$ ***dataSelect*** (D_i)
  $C_n' \leftarrow$ ***create*** $(D+\triangle)$

```
        if outperform(C_n', C_n )
            C_n ← C_n'
            D ← D+△
        else
            return C_n
        end-if
end-for
return C_n
```

2.1.1 Data Selection Function

This function aims at selecting complementary data for D. Here we assume no concept-drift in D_i (*outperform* will deal with concept-drift). But even under stationary concept, unselectively importing old data is harmful because (1) it makes the learner over-fit the old part; (2) unnecessarily large amount of training data slows down the learning. In this sense, we define the *dataSelect* as choosing: (1) data in D_i that are misclassified by C_n, plus (2) data that are misclassified by previous learner C_{n-1}. Choosing data misclassified by C_n is based on the hypothesis that C_n has not mastered this part of data and thus needs further learning. The idea of importing data misclassified by C_{n-1} is inspired by Boosting [2, 6]: each learner puts emphasis on the "difficult" part for its predecessor. From this perspective, *dataSelect* may bring additive benefits in two aspects [2, 6]: (1) reducing bias; (2) augmenting the diversity among individuals. Both of these will improve the performance of ensemble.

2.1.2 Evaluation Function

Outperform evaluates C_n and C_n', and makes decision that whether importing $\triangle$ to D is sensible. Since $\triangle$ is made up of misclassified data, we must be wary of two possibilities: (1) Misclassification caused by concept-drift; (2) Misclassification caused by noise. In these two cases, introducing such misclassified data will do harm to training. Furthermore, when improvement is insignificant, importing should also be stopped for the sake of efficiency.

The process for evaluating C_n and C_n' is as follows: Firstly, compute the prediction accuracy of C_n and C_n' (termed p and p', respectively) based upon evaluation dataset. Secondly, calculate lower-bound (termed *low* and *low'*) for p and p' under confidence *conf*, according to equation (1). Thirdly, if and only if $low'-low > \varepsilon$ holds for threshold ε, we judge that C_n' *outperform* C_n.

$$low=\left(p+\frac{z^2}{2V}-z\sqrt{\frac{p}{V}-\frac{p^2}{V}+\frac{z^2}{4V^2}}\right)\Big/\left(1+\frac{z^2}{V}\right) \quad (1)$$

In equation (1), z satisfies $P(X \geq z) = conf$ under normal distribution and $V = chunkSize$. The intuition of this equation is: given a prediction accuracy p based on a test set of size V, we assume p is a random variable that has mean m and standard deviation $\sqrt{m(1-m)/V}$. Then (2) holds, which naturally leads to (1) where *low* is one solution of m.

$$P\left(-z < (p-m)\Big/\sqrt{m(1-m)/V} < z\right) = conf \quad (2)$$

2.2 Global-Prediction Strategy

In section 1, we have reviewed different policies to organize global predictor, such as majority voting, weight-based voting and select-best. In fact, the ideal strategy should strike a balance between these choices. On one hand, it should retain the benefits of voting by masses rather than simply select the best individual. On the other hand, we want the sensible experts to dominate the voting, thus render the global predictor sensitive to concept-drift.

2.2.1 Dynamic Voting

Our strategy is based upon the fact that we only want to divide the ensemble into two categories: the "good enough" experts and the else. Since we assume merely two categories in basic learners, it is reasonable to expect certain simple method to "judge good and evil in such a melodrama". Here we put forward an efficient procedure to choose voters from ensemble.

(1) Sort N basic classifiers in ensemble according to their accuracies on evaluation dataset.
(2) Among N-1 distances between sorted classifiers, find the maximal one.
(3) The maximal distance naturally divides the learners into two groups.
(4) Engage the "better" group as voting group.

The time complexity of this procedure depends on sorting step, which is trivial when N only refers to the capacity of ensemble. Furthermore, this procedure is executed only when the evaluation dataset is replaced by new chunk (the ensemble will be updated at the same time). Based on this voting policy, choosing ensemble capacity N is easy ---- we can choose a larger quantity than other voting-based algorithms, for the reason that outmoded experts in ensemble will be excluded from voting group by our dynamic voting. It will benefit in two aspects: (1) Under stationary concept, large ensemble furnishes sufficient voters; (2) In concept-drift scenario, large ensemble offers more opportunities for finding sensible experts, especially when concept switches in a repeated way.

2.2.2 Discussion: Other Choices?

Now we discuss that whether some other simper strategies can be used instead of our dynamic voting: (1) "select best-k": For N experts in ensemble, select k best experts as voters. (2) "Performance threshold": according to a threshold p_0, define experts in ensemble whose

accuracies higher than p_0 as voters. Firstly, the "select best-k" policy aims at retaining the sensitivity of "select-best" policy and gaining the benefits of voting. Nonetheless, this strategy is obviously incompetent in that it is actually the similar with "majority voting" where $N = k$, whose flaws have been discussed in Section 1. Secondly, the "performance threshold" is not an ideal approach, either. In fact, we can not decide this threshold in order to divide the ensemble into "sensible" ones and the else: (1) Performance of basic classifier changes dramatically on different classification problems. (2) It is unknown that to what extent the concept-drift will degrade the performance of outmoded experts and where should we set this threshold.

3. Empirical Study and Results

This section presents the results of our experimental evaluation of the proposed method. The goal of our experiments is to demonstrate the ability of our algorithm to: (1) handle data insufficiency when training basic classifiers; (2) form effective voting; (3) keep sensitive to concept-drifts.

3.1 Dataset and System Implementation

To determine the performance of our algorithm on problems involving concept-drifts, we design the problem in which each data points has three attributes $x, y, z \in R$, randomly sampled from range [0, 10]. The data point that satisfies the target concept $x^2 + y^2 + z^2 < r^2$ is labeled by 1. Otherwise the item will be labeled as 0. Radius r is used to control the concept-drifts. Experiments are implemented on Weka toolkit [14].

3.2 Concept-drift Tests and Results

Four algorithms are tested: (1) SEA: algorithm in [11]; (2) DWM: algorithm in [9]; (3) DCS: algorithm in [16]; (4) DCO: our algorithm. DWM does not take part in *Test1* and *Test2* since it must use incremental basic classifier. All results are averaged from 30 independent runs.

(1) *Test1*: Testing SEA, DCS, and DCO based on C4.5, *chunkSize* = 50.

(2) *Test2*: Testing SEA, DCS, and DCO based on C4.5, *chunkSize* = 100.

(3) *Test3*: Testing SEA, DCS, DCO and DWM based on Naïve Bayes, *chunkSize* = 50.

(4) *Test4*: Testing SEA, DCS, DCO and DWM based on Naïve Bayes, *chunkSize* = 100.

The procedure of experiment is: There are entirely 50 *chunkSize* training data points. For the first fourth the radius r in target concept is 9; for the second r = 11.5; for the third r = 8.5; for the last r = 11. For each fourth, we randomly generate a testing dataset of 2500 data points on corresponding radius. Each time after *chunkSize* training

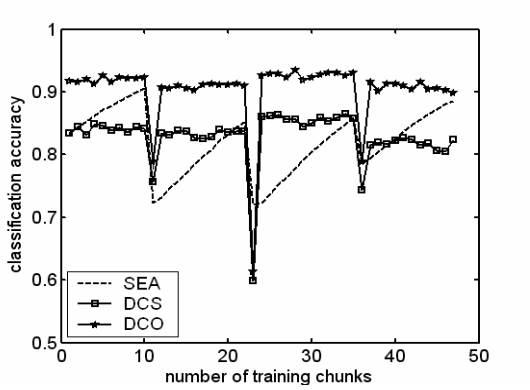

Figure 1: Results of *test 1*.

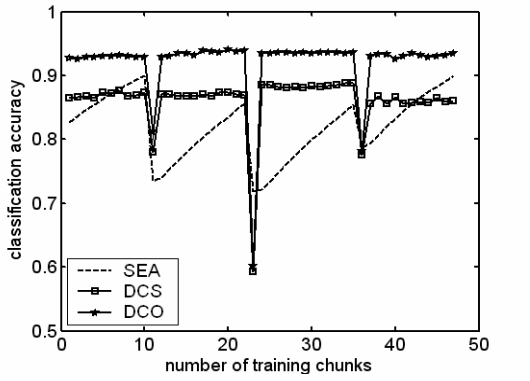

Figure 2: Results of *test 2*.

data points are offered, we test all the algorithms using appropriate testing dataset. For our algorithm, *conf* = 0.9 and ε = 1% in *outperform* function. For all algorithms with fixed-size ensemble, we set N = 50. Other parameters are set according to original papers. See Fig.1~Fig.4 for results, the analysis of these results is as follows:

(1) **Prediction accuracy:** DCO has the best classification accuracy, and this advantage appears more evident when the size of data chunk is limited (*chunkSize* = 50). Such superiority dues great part to our novel strategies for individual-construction and global-prediction. In one sense, the former policy guarantees the sufficiency of training data for basic learners, augments the diversity among individuals, and reduces the bias of basic learners. In another sense, the latter strategy strikes a balance between the quantity of voters and the quality of voters, and thus renders the voting process much more effective in terms of variance-reduction.

(2) **Sensitivity for concept-drift:** DCO and DCS are quite sensitive to concept-drift: they recover from misclassification very fast; DWM is not as sensitive as DCO and DCS, but still much better than SEA. DCS's sensitivity obviously dues to its "select-best" policy; DCO relies on dynamic voting to exclude outmoded experts, thus remains as sensitive as DCS; DWM uses weighted-based

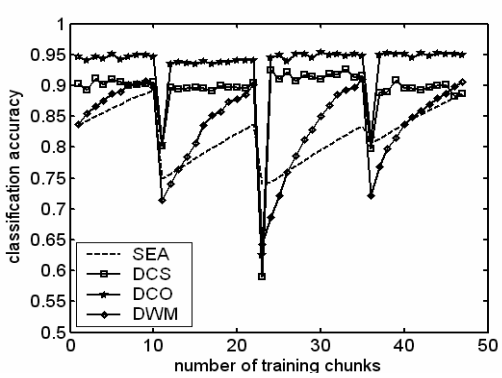

Figure 3: Results of *test 3*.

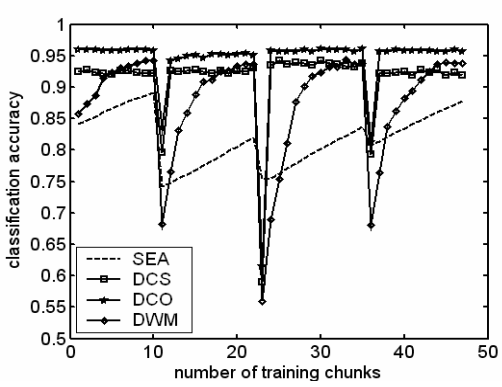

Figure 4: Results of *test 4*.

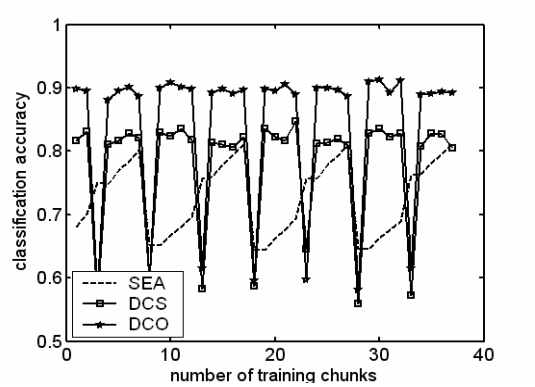

Figure 5: Results of *test 5*.

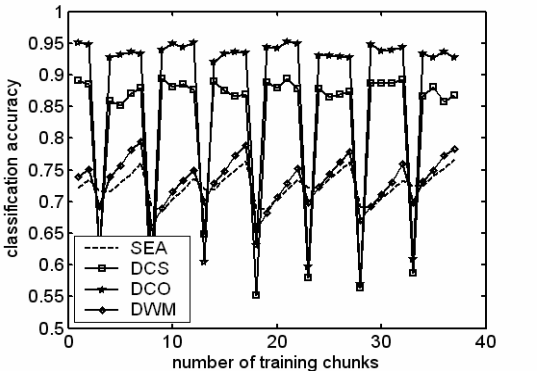

Figure 6: Results of *test 6*.

voting, and does not fix the capacity of ensemble, therefore has better alertness than SEA, which engages majority voting on fixed amount of voters in ensemble.

(3) **DCS and basic learner:** DCS algorithm performed much better under naïve Bayes than using C4.5, because the former is a stable basic learner which is not in dire need of voting to reduce its variance. However, using unstable classifiers such as C4.5, DCS will appear ineffective. Furthermore, DCO beats DCS even based on naïve Bayes, since DCO enhances data sufficiency, individual diversity, and further reduces the bias-variance by dynamic voting.

(4) **Efficiency of algorithms:** We test the efficiency of the four algorithms, represented by the time consumed in their 30 independent runs and shown in Table 1. SEA is time-consuming, especially when using Naïve Bayes. DCO is as efficient as DCS based on C4.5, and retains a reasonable speed on Naïve Bayes. In fact, the most complex part of DCO, the individual construction process mentioned in section 2.1, always stops after combining a few old blocks. Note that DWM used more time in test3 than in test4 since small data blocks lead to frequent creation of new classifers.

3.3 Performance in Severe Circumstance

What is more interesting is the performance of these algorithms in severe conditions: data insufficiency plus frequent and sudden concept-drifts, where data insufficiency calls for the ability to seek enough data for training basic classifiers; and frequent abrupt concept-drifts require excluding old concepts from training data. We set *chunkSize* as 25. For totally 40 *chunkSize* training data points, concept-drift happens after each 4 chunks. The radius starts with r=8, switches between 8 and 12 (i.e. r = 8, 12, 8, 12 …). For each concept, we randomly generate 2500 data points by corresponding radius. Each time after a chunk of training data is offered, we test all the algorithms using appropriate testing points. Other settings are similar with section 3.2. *Test5* concerns SEA, DCS, and DCO based on C4.5; *test6* measures SEA, DCS, DCO and DWM on naïve Bayes. The results are shown in Fig. 5 and Fig. 6. We can observe that: (1) the individual-construction policy effectively handles the data insufficiency; (2) the dynamic voting strategy furnishes successful voting, while retains the sensitivity to sudden and frequent concept-drifts.

Table 1. Efficiency of Algorithms

	SEA	DCS	DCO	DWM
Test1	6min 11sec	1min 4sec	1min 39sec	---
Test2	8min 1sec	1min 46sec	1min 51sec	---
Test3	51min 9sec	3min 7sec	14min 9sec	11min 5sec
Test4	78min 4sec	5min 21sec	15min 3sec	6min 1sec

3.4 Real-world Dataset

In this section, we proposed our empirical results on "adult" dataset [17]. We test SEA, DCS and DCO upon C4.5. The training and testing dataset contain 32561 and 16281 instances, respectively. Data has 14 attributes such as the age, occupation and sex of a person. The label indicates whether this person has an income larger than 50k dollar. The preprocessing step aims to produce sufficient concept drifts: partition the dataset by "occupation" attribute, then collect instances in three occupations – "Adm-clerical", "Exec-managerial" and "Other-service"; finally we get three training subsets with 3770, 4066 and 3295 instances and three test subsets with 1841, 2020 and 1628 instances, respectively. We set blockSize as 100. Totally 90 data blocks are engaged for training: 15 blocks from subset1, 15 blocks from subset2, 15 blocks from subset3, and then repeat. After each block, we test all the algorithms using corresponding test dataset. For all algorithms, ensemble size is 30. The results are shown in Table 2, which justifies the superiority of DCO.

Table 2. Empirical Results on "Adult" Dataset

SEA	DCS	DCO
0.7773	0.8401	0.8510

4. Conclusions

Current algorithms for mining data streams are confronted with two contradictory elements: Firstly is the need for seeking adequate training data for each basic classifier and gathering sufficient voters for final-decision; and secondly, is the requirement for sensitivity to concept-drift, which calls for using recent training data and up-to-date basic classifier. In this work, we initially point out the essential reasons for the incompetence of several recent algorithms in solving these conflicting elements. Then, we propose a dynamic ensemble learning algorithm, termed DCO (Dynamic Construction and Organization), which aims at reconciling these contradictions. Experimental results justify the superiority of our approach over the state-of-the-art algorithms in that individual-construction strategy provides solution to data insufficiency under concept-drift scenario; and the dynamic voting strategy strikes a balance between the quantity and quality of voters.

5. Acknowledgement

This work is supported by the National Science Foundation of China (60403021) and the 973 Program (2004CB719400).

6. References

[1] CC Aggarwal, J Han, J Wang, PS Yu. On Demand Classification of Data Streams. In Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004.

[2] E. Bauer, R. Kohavi. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. Machine Learning, vol 36, pp 105-139, 1999.

[3] L. Breiman. Bagging Predictors. Machine Learning, vol 24, pp 123-140, 1996.

[4] GZ Dong, JW Han, Laks V.s. Lakshmanan, J Pei, HX Wang, Philip S. Yu. Online Mining of Changes from Data Streams: Research Problems and Preliminary Results. ACM SIGMOD MPDS`03 San Diego, CA, USA.

[5] W Fan. Systematic Data Selection to Mine Concept-Drifting Data Streams. In Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004.

[6] Y Freund, RE Schapire. Experiments with a New Boosting Algorithm. Machine Learning: Proceedings of the 13th International Conference, 1996.

[7] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2001.

[8] G. H. John and P. Langley. Estimating Continuous Distributions in Bayesian Classifiers. In Proc. of the Eleventh Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann, San Mateo, 1995, 338-345.

[9] JZ Kolter, MA Maloof. Dynamic Weighted Majority: A New Ensemble Method for Tracking Concept Drift. Proceedings of the Third IEEE International Conference on Data Mining, 2003.

[10] J. R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA, 1993.

[11] W. N. Street, YS Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In: Proc. of the 7th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining, San Francisco, CA, ACM Press, 2001, 377-382.

[12] H Wang, W Fan, PS Yu, J Han. Mining concept-drifting data streams using ensemble classifiers. In Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003.

[13] G. Widmer and M. Kubat. Learning in the Presence of Concept Drift and Hidden Contexts. Machine Learning, vol23, issue1, 1996, 69-101.

[14] I. H. Witten and E. Frank. 1999. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, San Mateo, CA.

[15] Y Yang, X Wu, X Zhu. Combining Proactive and Reactive Predictions for Data Streams. In Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2005.

[16] XQ Zhu, XD Wu and Y Yang. Dynamic Classifier Selection for Effective Mining from Noisy Data Streams. In: Proc. 4th IEEE Int'l Conf. on Data Mining, 2004, 305-312.

[17] http://www.ics.uci.edu/~mlearn/MLRepository.html