Adaptive Classifier Selection Based on Two Level Hypothesis Tests for Incremental Learning

Haixia Chen[1], Senmiao Yuan[1], and Kai Jiang[2]

[1] College of Computer Science and Technology, Jilin University, Changchun 130025, China
hxchen2004@sohu.com
[2] China Electronics Technology Group Corporation No. 45 Research Institute,
Beijing 101601, China
kjiang2004@sohu.com

Abstract. Recently, the importance of incremental learning in changing environments has been acknowledged. This paper proposes a new ensemble learning method based on two level hypothesis tests for incremental learning in concept changing environments. We analyze the classification error as a stochastic variable, and introduce hypothesis test as mechanism for adaptively selecting classifiers. Hypothesis tests are used to distinguish between useful and useless individual classifiers and to identify classifier to be updated. Classifiers deemed as useful by the hypothesis test are integrated to form the final prediction. Experiments with simulated concept changing scenarios show that the proposed method could adaptively choose proper classifiers and adapt quickly to different concept changes to maintain its performance level.

1 Introduction

Most supervised learning algorithms assume stationary target concept over time, require a larger number of training examples beforehand and learn the target concept in batch mode. In real-world applications, however, data are always collected over an extended period of time, and the target concept underlying the data is changing. Thus incremental learning of a classification system must have specially designed mechanism for dealing with drifting concepts.

Methods dealing with concept drift can be classified into two categories: the instance based approaches and the ensemble based approaches. The former one tries to select instances most relevant to the current concept to keep up with the drifting concepts. The FLORA family algorithms [1] and the AQ series algorithms [2] are typical examples of this approach. Problem with those approaches is that information learned from historical data is discarded in order to adapt to the current concept. This is called catastrophic forgetting in literature [3]. The ensemble approaches settle down this problem by storing knowledge learned from historical data. To adapt to the conflicting concepts, they should dynamically delete, reactivate or create new ensemble members based on the base classifiers' consistency with the current data. The first concept drift handling system STAGGER falls into this category [4].

The ensemble based incremental learning approaches can be further divided into two categories: the boosting style method and the bagging style method. Examples of the former one include the Learn++ series algorithms [5] and Adaptive Boosting [6].

D.-Y. Yeung et al. (Eds.): SSPR&SPR 2006, LNCS 4109, pp. 687 – 695, 2006.

In those methods, a new classifier is generated from a skewed distribution adjusted according to the joint performance of former classifiers. There is a strong dependent relationship between those classifiers, and it is problematic to select only partial of those classifiers to reflect drifting concepts. So, it is more beneficial to use bagging style method in concept drifting environments.

Street et al. [7] uses a simple majority vote of classifiers on sequential chunks and responds to concept drift by replacing an unnecessary classifier with a new classifier. Wang et al. [8] uses weight inversely proportional to the error of the classifier on the current chunk to combine their outputs. A few best classifiers are selected to reflect concept drifting. However, in those methods the classifier selection process is managed by some predefined parameters and can't adapt well to the concept change rate. When the concept change rate is low, we hope to keep more classifiers in the system to reduce the classification variance and to make full exploitation of former knowledge. However, when the concept change rate is high, or a conflicting concept appears, we hope to discard those unnecessary former classifiers as soon as possible. So a more flexible classifier evaluation mechanism is needed which can adaptively select base classifiers for integration according to the concept change rate and degree. Chu et al. [9] views the setting of weights for base classifiers as an optimization problem. They use EM algorithm to find outliers and used logistic regression to calculate weights that fit best to the data set excluding those deemed outliers. Computational cost for EM algorithm is high and the validity of model assumption is hardly hold in concept changing environments. Furthermore, for classification systems, fine tuning of the parameters is always a main cause of overfitting and should be avoided.

In this paper, we present a method to incremental Learning in concept drifting environments by explicitly detecting and adapting to drifting concepts. The method is based on two level hypothesis tests. At the first level, the hypothesis test is used to identify the classifiers to be updated on-line. New classifier is built only when no classifiers are updated. So the increase of the ensemble size is controlled. At the second level, the hypothesis test is used to distinguish between useful and useless individual classifiers so as to avoid the interference of conflicting classifiers and make full use of the useful ones. The method belongs to the bagging style ensemble learning method. Compared to the former mentioned approaches, it explicitly monitors the concept changes and can select classifiers accordingly. There have been many work related to explicit detection of concept drift. Methods based on comparison between current performance and a confidence interval adaptively derived from former data batches have been recently proposed [10] [11] [12] [13]. However, since the performance is a random variable, it is difficult to distinguish between a concrete concept drift and a random fluctuation, especially when the environments are noisy and the concept change rate varies too. Chu and Zaniolo [6] viewed the drift detection as a choice between two candidate distributions. Performance probability conditioned on those two distributions are calculated and compared. However, the comparison is based on only one performance observation. So it is highly biased. Furthermore, the comparison threshold is difficult to set. Our method is statistically well-grounded and no complex parameter tuning is necessary.

The rest of this paper is organized as follows. We give a detailed explanation of the principle of the proposed method in Section 2. Experimental results are given in Section 3, followed by conclusion in Section 4.

2 Incremental Learning Based on Two Level Hypothesis Tests for Changing Concepts

In the concept changing environments, the problem of incremental learning boils down to decide how to detect the change of the target concept and how to adapt to these changes.

2.1 Detection of Concept Change

Take the prediction of the classifier *h* as a stochastic event which has two possible outcomes δ for an example: $\delta = 0$ for correct prediction and $\delta = 1$ for wrong. Let *R* be the proportion of $\delta = 1$ in experiments with *n* test examples. Then *R* follows a Binomial distribution with parameters *n* and *p*:

$$\Pr(R = r) = \binom{n}{nr} p^{nr} (1-p)^{n-nr}, nr = 0,1,...,n \,. \tag{1}$$

Where *p* is the expected error rate of classifier *h* with respect to target concept *f* and distribution $\mathcal{D}$: $p = error_{\mathcal{D}}(h) \equiv \Pr_{x \in \mathcal{D}}[f(x) \neq h(x)]$. Moreover if the sample size are large ($n \geq 30$), the distribution of *R* is approximately Normal (a well-known result derived by the Central Limit Theorem). For a sequence $r_1,...,r_k$ with *k* observations of *R*, statistic $t = \sqrt{k}(\overline{r} - p)/s$ can be used for a *t* test to check if its expectation *p* has changed (alternative hypothesis) or not (null hypothesis). Where $\overline{r}$ is the mean value of the sequence, *s* is the standard deviation of the sequence.

The expected error rate *p* of a classifier is composed of three parts: Bayes error determined by the distribution underlying the target concept, bias imposed by the learning algorithm and variance caused by finite training examples[14]. For a classifier, bias imposed by the learning algorithm is fixed. For fixed batch size, variance caused by the finite training examples is stable. Thus, the change of the expected error rate is mainly caused by the change of the distribution. Therefore, the change of the expected error rate of a classifier on different data batches with the same size is a good indicator of the change of the target concept.

However, the expected error rate of a classifier on its target concept is unavailable. We can only estimate it from the training data set used to induce the classifier. In our implementation, we sample with stratified sampling method *n* ($n \geq 30$ for justification of the normal distribution approximation) examples from the training dataset for a performance test. The process repeated *k* times. And the expected error rate is estimated as the average of those tests. Given a classifier *h* and a new data set *D*, the pseudo code for hypotheses test is shown in Fig. 1. First, *k* test data sets are generated using the sampling method mentioned before, and base classifiers are tested on those data set. *L*-1 sequences, each with *k* observations and for each classifier, are generated. Then, the *t* statistic for each sequence is calculated and is compared to $t_{\alpha/2}(k-1)$, where α is the significant level. If $|t_i| \geq t_{\alpha/2}(k-1)$, *i*=1,…, *L*-1, then concept change is suspected. Otherwise, the concept is deemed as stationary.

```
function HypothesisTest(h,D,k,n,α){
  [SD(1),...,SD(k)]=SampleDataset(D,k,n);
  for(i=1;i<=k;i++){
    err(i)=Error(h, SD(i));
  }
  t=Statistic(err,h);
  if( abs(t)>=t_{α/2}(k-1) ){
    return FALSE;
  }
  return TRUE;
}
```

Fig. 1. Pseudo code for hypothesis test of concept change

2.2 Update of the Classification System

We have developed a general algorithm for handling changing concepts based on two level hypothesis tests. The pseudo code is presented in Fig. 2. Let H be the classifiers ensemble to be updated, $H=\{h_l\}$. $h_l : X \to Y$ is a base classifier in the classification system, l=1,…,L. L is the number of base classifiers in the ensemble. Let $D(t)$ be the data batch used to update the classification system at time t. In each time step, the algorithm begins by running through the current classification system to find the type of concept change for each base classifier. If the concept is stationary, then the classifier is updated online using $D(t)$ and is denoted as useful. If the concept is suspected to drift(change gradually), then the classifier is denoted as useful. If the concept is suspected to shift(change abruptly), the classifier is denoted as useless. After a pass through the ensemble, if no classifier is updated, then a new classifier will be induced from scratch using $D(t)$ as training data. To give the final predication of a new example, all the classifiers denoted as useful are integrated by weights inversely proportional to the error of the classifier on the current chunk for the final prediction.

Fig.3 gives an illustration of the effect of α_1, α_2 in determining the types of concept changes. These two parameters divide the t distribution into three regions: D1 is denoted as the blank region under the probability curve, D2 is denoted as the bright gray region under the probability curve, and D3 is denoted as the dark gray region under the probability curve. If the tested statistic value falls in D3, then the hypothesis that the underlying concept is stationary is hold. In this case, we can update the base classifier by online learning algorithm. It should be noted that, in addition to update the classification model, the sample error rate p should also be updated. If the tested statistic value falls in D2, this is unlikely under the null hypothesis, and the target concept should be deemed as changed. But from a stricter perspective, the change is not so serious, and the tested classifier can also provide useful information for classification of the current data set. So the classifier is tagged as useful but not updated to avoid interference of different concepts. If the tested statistic value falls in D3, we suppose that a completely different concept or even a conflicting one appears, and the classifier for the historical data is eliminated from integration, because it may be conflictive with the current one.

```
function SystemUpdate(D(t)){
  S=0; // usefulness flag vector for each classifier
  bUpdate=FALSE;
  for(i=0;i<L;i++){
if(HypothesisTest(H(i),D(t),k,n,α1)==TRUE){//stationary
  OnLineLearning(H(i),D(t));
  S(i)=1; // the classifier is useful
  bUpdate=TRUE;
}elseif(HypothesisTest(H(i),D(t),k,n,α2)==TRUE){//drift
  S(i)=1;
}else{ // shift
  S(i)=0;
}
  }
  if(bUpdate == FLASE){
    BatchLearning(h,D(t)); // train a new classifier
    AddNewClassifier(H,h); // Add it to the ensemble
    AddElement(S,1); // turn on its useful flag
  }
}
```

Fig. 2. Pseudo code for updating the classification system with new data batch

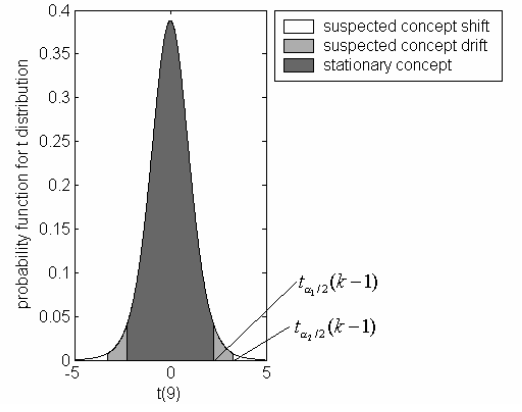


Fig. 3. Illustration of the three kinds of concept changes

3 Experiments

In the experiments, we compare the performance of the proposed algorithm, abbreviated as NBE, to other three algorithms: batch learning using all available training data (abbreviated as BAT), sliding window trained with data only in the current batch (abbreviated as WIN), and ensemble learning with weights proportional to their accuracy on the most recent data block as suggested in Ref. [8] (abbreviated as ENS). At most 10 base classifiers are kept in ENS. Parameters for NBE are: $n = 30$, $k =$

$10, \alpha_1 = 0.05, \alpha_2 = 0.01$. We set n for reason addressed above, set k based on pilot experimental study, and set α_1, α_2 following a general suggestion from statistics. Naïve Bayes [15] are used as base classifiers because it is a stable and robust classification algorithm with little bias and we could focus our attention on detecting and adapting mechanisms of the ensemble but not on fine tuning of the base classifiers.

We use two groups of concepts generally used in literature to simulate different types of concept drifts. The use of artificial datasets allow us to control the points where the concept drifts and recurs, and the degree to which the concept drifts. The first data set is to simulate sudden changes in target concepts and is generated from a group of concepts that is first introduced by Schlimmer and Granger in STAGGER [4]. The concepts are defined on three attributes: $color \in$ {green,blue,red}, $shape \in$ {triangle,circle,rectangle}, $size \in$ {small,medium,large}. We define the target concepts as: concept1 $\Leftrightarrow$ $color$ = red $\wedge$ $size$ = small, concept2 $\Leftrightarrow$ $color$ = green $\vee$ $shape$ = circle, concept3 $\Leftrightarrow$ $size$ = medium $\vee$ $size$ = large, concept4 $\Leftrightarrow$ concept1. Concept2 and concept3 are conflicting with concept1, so behaviors of approaches dealing with conflictive concepts could be observed. Concept4 is identical with concept1. This is designed to investigate the behavior of those approaches when they encounter with recurring concept. For every target concept, ten data chunks, each with 300 examples, are generated sequentially. In each chunk 100 examples are used for training and the other 200 examples for testing.

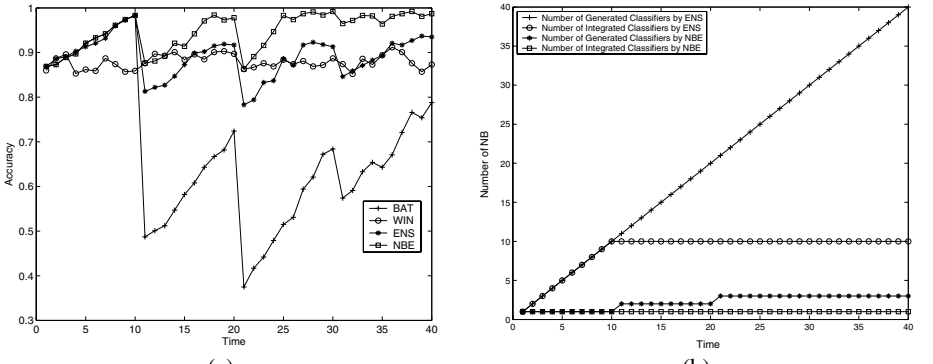
The second data set is proposed to simulate gradual changes in target concepts with a hyperplane in a d-dimensional space:

$$\sum_{i=1}^{d} a_i x_i = a_0 . \tag{2}$$

Where x_i is the coordinate of the ith dimension/feature, $x_i \in \{0,1\}$, and a_i is the weight for the feature, $a_i \in [0,1]$. If $\sum_{i=1}^{d} a_i x_i \geq a_0$, the example is classified as positive. Otherwise it is labeled as negative. By adjusting the weight of each feature, the target concept could change smoothly [16]. We set d=30, and initialized a_i randomly. To simulate a smoothing concept drift, a concept is formed by randomly choosing a feature and increasing its weight by 0.1. The weight is subtracted by 1 if it surpassed 1. In this way, a sudden concept change is inserted in. 40 data batches are generated, each for a new concept. 300 examples are generated in all data batches, among which 100 examples are used for training and the other 200 examples for testing. For each new concept, a data chunk with 300 examples are generated randomly, among which 100 examples are used for training and the other 200 examples for testing.

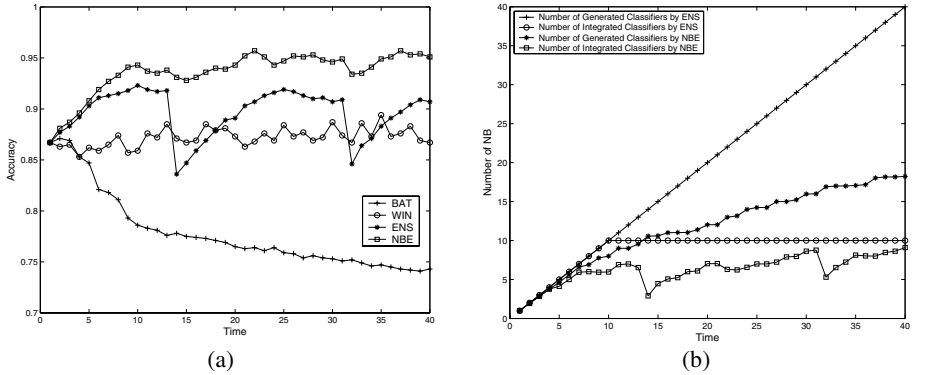
Fig. 4 compares the results of BAT, WIN, ENS and NBE on the first data set representing four concepts averaged over 30 runs. In general, NBE performs better than the other three compared approaches, especially when concept drifts. BAT performs well for stationary concept (time from 1 to 10) as it learns from all cumulated data. However, when concept drifts (time from 11 to 40), historical data becomes a burden for learning new concepts, and its performance degenerates seriously with more different concepts appear. WIN retains its performance for all time steps, even when the target concept drifts. However, it couldn't use all available examples when the target concept is stationary or when the same concept reappear. Both of the ensemble learning method

are better than the above two methods, because they could retain former knowledge in different base classifiers and adapt to new concept by discarding unnecessary ones. But NBE outperforms ENS especially when conflicting concepts appears (time from 11 to 30) or the same concept recurs (time from 31 to 40). We think this is because that number of base classifiers in ENS is a predefined parameter, so the algorithm could not adapt quickly to the sudden change of target concept. When concept changes suddenly, though those unnecessary former classifiers are assigned with relatively low weights, they still work for a period. And they are dominating in quantity. However, NBE find a better compromise between adaptation to drift and waste of knowledge. As it could detect concept drift explicitly, classifiers build for conflicting targets couldn't interfere with the prediction of the current one, and classifiers generated from the same concept could be used jointly with the current one for a better performance.



(a) (b)

Fig. 4. Performance comparisons between BAT, WIN, ENS and NBE for the first data set. (a) accuracy; (b) number of base classifiers.



(a) (b)

Fig. 5. Performance comparisons between BAT, WIN, ENS and NBE for the second data set. (a) accuracy; (b) number of base classifiers.

Fig. 5 compares the results of BAT, WIN, ENS and NBE on the second data set representing forty concepts averaged over 30 runs. It is noted that NBE performs the best and maintain its performance at a relatively stable level. Although ENS performs better than the other two algorithms in most cases, its performance degenerated seriously sometimes. Fig. 5(b) clearly shows that NBE don't create NB for each data batch and don't use all classifiers in the ensemble for integration. The number of classifiers in ensemble increases sublinearly with the time steps.

4 Conclusion

Nothing remains static. The world around us is evolving all the time. As a mirror of the real world, the classification system should also adapt to the changing target concepts. This paper studies the incremental learning of classification system in concept changing environments. A new ensemble learning method based on two level hypothesis tests is proposed. Different kinds of concept changes are detected by the two level hypothesis tests and treated accordingly. Experiments show that the proposed algorithm could adapt quickly to different kinds of concept changes and achieve better performance by adaptively selecting and integrating individual classifiers. In addition, the number of individual classifiers produced in the learning process is fewer than the other ensemble learning method.

References

1. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine Learning, 23, 1996, 69-101
2. Maloof, M.A., Michalski, R.S.: Incremental learning with partial instance memory. Artificial Intelligence, 154, 2004, 95-126
3. McCloskey, M., Cohen, N.: Catastrophic interference in connectionist networks: the sequential learning problem. The Psychology of Learning and Motivation. Vol. 24, 1989, 109-164
4. Schlimmer, J.C., Granger, R.H. Jr.: Incremental learning from noisy data. Machine Learning, 1, 1986, 317-354
5. Polikar, R., Udpa, L., Udpa, S.S., Honavar, V.: Learn++: an incremental learning algorithm for supervised neural networks. IEEE Transactions on Systems, man, and Cybernetics-Part C: Applications and Reviews, Vol.31, No.4, 2001, 497-508
6. Chu, F., Zaniolo, C.: Fast and light boosting for adaptive mining of data streams. H. Dai, R. Srikant, and C. Zhang (Eds.) PAKDD 2004, LNAI 3056, 2004, 282-292
7. Street, W., Kim, Y.: A streaming ensemble algorithm (SEA) for large-scale classification, Proc. 7th ACM SIGKDD, ACM Press, 2001, 377-382
8. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. SIGKDD'03, August 24-27, 2003, Washington, DC, USA
9. Chu, F., Wang, Y., Zaniolo, C.: Mining noisy data streams via a discriminative model. E. Suzuki and S. Arikawa (Eds.) DS 2004, LNAI 3245, 2004, 47-59
10. Klinkenberg, R., Renz, I.: Adaptive information filtering: Learning in the presence of concept drifts. Learning for Text Categorization, Menlo Park, CA, USA, AAAI Press, 1998, 33-40

11. Fung, G.P.C., Yu, J.X., Lu, H.: Classifying text streams in the presence of concept drifts. H. Dai, R. Srikant, and C. Zhang (Eds.) PAKDD 2004, LNAI 3056, 2004, 373-383
12. Natwichai, J., Li, X.: Knowledge maintenance on data streams with concept drifting. J. Zhang, J.H. He, and Y. Fu (Eds.) CIS 2004, LNCS 3314, 2004, 705-710
13. Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with drift detection. A.L.C. Bazzan and S. Labidi (Eds.) SBIA 2004, LNAI 3171, 2004, 286-295
14. Duda, R.O., Hart, P.E.: Pattern classification and scene analysis. Second Edition, New York, Willey and Sons, 2001
15. Domingos, P., Pazzani, M.: On the optimality of the simple Bayesian classifier under zero-one loss. Machine Learning, 29, 1997, 103-129
16. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 2001, 97-106