# Dynamic Classifier Selection for Effective Mining from Noisy Data Streams

Xingquan Zhu, Xindong Wu, and Ying Yang

*Department of Computer Science, University of Vermont, Burlington VT 05405, USA*

*{xqzhu, xwu, yyang}@cs.uvm.edu*

## Abstract

*Recently, mining from data streams has become an important and challenging task for many real-world applications such as credit card fraud protection and sensor networking. One popular solution is to separate stream data into chunks, learn a base classifier from each chunk, and then integrate all base classifiers for effective classification. In this paper, we propose a new dynamic classifier selection (DCS) mechanism to integrate base classifiers for effective mining from data streams. The proposed algorithm dynamically selects a single "best" classifier to classify each test instance at run time. Our scheme uses statistical information from attribute values, and uses each attribute to partition the evaluation set into disjoint subsets, followed by a procedure that evaluates the classification accuracy of each base classifier on these subsets. Given a test instance, its attribute values determine the subsets that the similar instances in the evaluation set have constructed, and the classifier with the highest classification accuracy on those subsets is selected to classify the test instance. Experimental results and comparative studies demonstrate the efficiency and efficacy of our method. Such a DCS scheme appears to be promising in mining data streams with dramatic concept drifting or with a significant amount of noise, where the base classifiers are likely conflictive or have low confidence.*

## 1. Introduction

The ultimate goal of effective mining from data streams (from the classification point of view) is to achieve the best possible classification performance for the task at hand. This objective has traditionally led to an intuitive solution: separate stream data into chunks, and then integrate the classifiers learned from each chunk for a final decision [11, 22, 24]. Given a huge volume of data, such an intuitive solution can easily result in a large number of base classifiers, where the techniques from Multiple Classifier Systems (*MCS*) [1-2] are involved to integrate base classifiers. The fact behind the merit of *MCS* is from the following underlying assumption: Each participating classifier in the *MCS* has a merit that deserves exploitation [3], i.e., each base classifier has a particular subdomain from which it is most reliable, especially when different classifiers are built using different subsets of features, different subsets of the data, and/or different mining algorithms.

Roughly, existing integration techniques can be distinguished into two categories:

1. Combine base classifiers for the final decision. When classifying a test instance, the results from all base classifiers are combined to work out the final decision. We refer it to *Classifier Combination* (*CC*) techniques.
2. Select a single "best" classifier from base classifiers for the final decision, where each base classifier is evaluated with an evaluation set to explore its domain of expertise. When classifying an instance, only the "best" classifier is used to determine the classification of the test instance. We name it *Classifier Selection* (*CS*) techniques.

In [4], the *CC* techniques were categorized into three types, depending on the level of information being exploited. Type 1 makes use of class labels. Type 2 uses class labels plus a priority ranking assigned to each class. Finally, Type 3 exploits the measurements of each classifier and provides each classifier with some measure of support for the classifier's decision. The *CS* takes the opposite direction. Instead of adopting the combining techniques, it selects the "best" classifier to classify a test instance. Two types of techniques are usually adopted:

1. *Static Classifier Selection* (*SCS*). The selection of the best classifier is specified during a training phase, prior to classifying a test instance [5-6].
2. *Dynamic Classifier Selection* (*DCS*). The choice of a classifier is made during the classification phase. We call it "dynamic" because the classifier used critically depends on the test instance itself [7-10].

Many existing data stream mining efforts are based on the *Classifier Combination* techniques [11, 22-24], and as they have demonstrated, a significant amount of improvement could be achieved through the ensemble classifiers. However, given a data stream, it usually results in a large number of base classifiers, where the classifiers from the historical data may not support (or even conflict with) the learner from the current data. This situation is compounded when the underlying concept of the data stream experiences dramatic changes or evolving, or when the data suffers from a significant amount of noise, because the classifiers learned from the data may vary dramatically in accuracy or in their domain of expertise (i.e., they appear to be conflictive). In these situations, choosing the most reliable one becomes more reasonable than relying on a whole bunch of likely contradictive base classifiers.

In this paper, we propose a new *DCS* mechanism for effective mining from noisy data streams. Our intuitive assumption is that the data stream at hand suffers from dramatic concept drifting, or a significant amount of noise, so the existing *CC* techniques become less effective. We will first review related work in Section 2; and then propose our new method in Section 3. In Section 4, we discuss about applying the proposed *DCS* scheme in noisy datasets. Our experimental results and comparative studies in Section 5 indicate that the proposed *DCS* scheme outperforms most *CC* or *CS* methods in many situations and appears to be a good solution for mining real-world data.

## 2. Related Work

The two main reasons of employing multiple classifiers for data stream mining are efficiency and accuracy. Although the efficiency could be the most attractive reason for adopting multiple classifiers, because a data stream can always involve a huge volume of data which turns to be a nightmare for any

single learner. The accuracy of *MCS* in handling stream data is also remarkable: especially when the concept in the data stream is subject to evolving, changing or drifting [11, 24]. Like many partitioning-based or scale-up learning algorithms (e.g., Bagging [12], Boosting [13] and Meta learning [14]) have demonstrated, by partitioning the whole dataset into subsets, the system efficiency can be dramatically improved, with a limited sacrifice of the accuracy.

When adopting *MCS* in stream data, the most intuitive (and probably also the simplest) scheme is simple voting which is also called Select All Majority (*SAM*) [7], where the prediction from each base classifier is equally weighted to vote for the final prediction. Although simple, *SAM* has been proved to be effective to integrate multiple classifiers, and many revised versions [11, 22, 24] have been successfully developed to handle data streams. In comparison with *CC* based schemes, the *CS* schemes select one classifier for the final decision, where two kinds of techniques, *SCS* and *DCS*, are usually adopted. Among all *SCS* schemes, the most intuitive one is Cross-Validation Majority (*CVM*)[5]. In *CVM*, cross-validation is adopted and the base classifier with the highest classification accuracy from the cross-validation is selected to classify all test instances.

In comparison with *SCS* where the "best" classifier has been selected before the testing phase, *DCS* dynamically selects the "best" classifier for each test instance. Among different *DCS* schemes, the most representative one is *Dynamic Classifier Selection by Local Accuracy* (*DCS_LA*) [10] which explores a local community for each test instance to evaluate the base classifiers, where the local community is characterized as the *k Nearest Neighbors* (*kNN*) of the test instance in the evaluation set *Z*. Although *DCS_LA* has been widely integrated in many systems, it suffers from the following three disadvantages:

(1) The selection of *k* and the adopted distance function critically affect the system performance.
(2) The speed factor. Given a test instance, *DCS_LA* has to go through the whole evaluation set to find its neighborhood. Its time complexity is unbearable for stream data.
(3) Sensitive to noise. Usually, the number of instances in a local community is relatively small, the existence of noise will critically affect the performance of *DCS_LA*.

Intuitively, for real-world datasets, not all attributes have the same importance in classification. Instead of using all attributes to evaluate the base classifiers, a more reasonable way may consider the important attributes of each base classifier. Accordingly, a referee based dynamic classifier selection scheme was proposed in [9] where referees, in the form of decision trees, partition the whole evaluation set into subsets, and each base classifier is evaluated with these subsets to explore its domain of expertise. The advantage of the Referee is that it partitions the evaluation set into subsets by joining the features of the base classifiers. The less important attributes won't be used in partitioning the evaluation set, and the partitioned subsets tend to be more reasonable in exploring the domain expertise. However, the drawbacks are threefold: (1) to learn each referee, one has to explore the features of each base classifier; (2) it uses the learned decision trees to partition the original training set into subsets, and the system performance will critically depend on the quality of learned decision trees; and (3) because each classifier has its own referee, and the reliabilities from different referees are evaluated from different subsets. Without the same measurements, the selected classifier might not work well.

The above review shows that in order to explore the domain expertise of each base classifier, the *DCS* schemes have to evaluate the classifiers from either an entire or partial evaluation set to determine which classifier is the best for the test instance at the current stage. Such a mechanism inherently provides an adaptive scheme that is most suitable for mining data streams with dramatic changes, where most existing mining efforts appear to be less effective. However, all existing *DCS* approaches use either distance-based schemes or classification trees (or classification rules) to partition the evaluation set into subsets, where the quality of the partitioned subsets critically depends on the performance of adopted distance functions or classification trees. In this paper, we present a new *DCS* scheme that evaluates base classifiers with subsets of the evaluation set, where the subsets are constructed with statistical information of attribute values. We believe such a partitioning scheme is more natural and intuitive in exploring the domain expertise of each base classifier for effective data stream mining.

# 3. AO-DCS: Attribute-Oriented Dynamic Classifier Selection

In this section, we present a new dynamic classifier selection scheme, called AO-DCS (*Attribute-Oriented Dynamic Classifier Selection*). We use attribute values of instances to partition the evaluation set into subsets for evaluation purpose. If the instances in a dataset have only one attribute, and we use this attribute to partition the evaluation set into disjoint subsets with each subset corresponding to one value of this attribute, the classification accuracy of each base classifier with these subsets is the performance of the classifier with the instances characterized by each attribute value. Then each base classifier's performance will reflect its domain of expertise. Our *AO-DCS* takes the following three steps.

1. Statically partition the evaluation set into subsets by using the attribute values of the instances. We denote the aggregation of all constructed subsets by $\Pi$.
2. Evaluate the classification accuracy of each base classifier on all subsets in $\Pi$. We call this accuracy "attribute-oriented" classification accuracy.
3. For a test instance, use its attribute values to select the corresponding subsets from $\Pi$, and select the base classifier $C_j$ that has the highest classification accuracy from the selected subsets as the "best" classifier to classify the test instance.

## 3.1 Constructing Subsets

Given a dataset *D*, let *X*, *Y* and *Z* be the training, test and evaluation set, with the numbers of instances in *X*, *Y* and *Z* denoted by $N_X$, $N_Y$ and $N_Z$ respectively, and $C_1, C_2,.., C_L$ be the *L* base classifiers from *X*. The objective of *AO-DCS* is to select the "best" classifier $C^*$ to classify each instance $\hat{I}_x$ in *Y*. Our algorithm first acquires statistical attribute information of all instances in *D*. Assuming the instances in *D* have *M* attributes $A_1, A_2,..,A_M$, and each attribute $A_i$ contains $n_i$ values $V_1^{A_i},..,V_{n_i}^{A_i}$ (we will discretize numerical attributes into discrete values). For an attribute $A_i$, we use its values to partition the evaluation set *Z* into $n_i$ subsets $S_1^{A_i},..,S_{n_i}^{A_i}$, where $S_1^{A_i} \cup..\cup S_{n_i}^{A_i} = Z$. To partition *Z*, we design a procedure in Fig. 1.

**Procedure** **PartitionEvaluationSet** ( $A_i$ )
**Input:** Attribute $A_i$ of the evaluation set $Z$.
**Output:** $n_i$ subsets $S_1^{A_i},..,S_{n_i}^{A_i}$ determined by $A_i$.

(1) $S_1^{A_i} \leftarrow \varnothing, S_2^{A_i} \leftarrow \varnothing, .., S_{n_i}^{A_i} \leftarrow \varnothing$
(2) **For** $k$ =1; $k \le N_Z$; $k$++
(3) For $r$ =1; $r \le n_i$; $r$ ++
(4) $S_r^{A_i} = S_r^{A_i} \cup \{I_k \mid I_k \in Z, I_k^{A_i} = V_r^{A_i}\}$
**End for**

**Fig. 1. Evaluation set partitioning by attribute values**

In Fig. 1, $I_k^{A_i}$ denotes instance $I_k$*'s* value on attribute $A_i$ in the evaluation set $Z$. Given $A_i$, we can use its values to partition the evaluation set $Z$ into $n_i$ disjoint subsets, and instances with the same values on $A_i$ are put in the same subset. We partition the evaluation set $Z$ by each attribute, and we can hereby construct $\sum_i^M n_i$ subsets from all attribute values. An example of evaluation set partitioning from a dataset containing only two attributes $A_1$ and $A_2$ is pictorially depicted in Fig. 2, where the $x$-axis denotes the values of attribute $A_1$, and the $y$-axis represents the values of attribute $A_2$ (assuming $A_1$ and $A_2$ contain 3 and 4 values respectively). The $x$-axis and $y$-axis span the space ($\Re$) of all instances in this dataset. Using three attribute values $V_1^{A_1}$, $V_2^{A_1}$ and $V_3^{A_1}$ of $A_1$, we construct three subsets, $S_1^{A_1}$, $S_2^{A_1}$ and $S_3^{A_1}$, with $S_1^{A_1} \cap S_2^{A_1} \cap S_3^{A_1} = \varnothing$ and $S_1^{A_1} \cup S_2^{A_1} \cup S_3^{A_1} = \Re$. Four subsets constructed by $A_2$ are also depicted in Fig. 2. Any two subsets from $A_1$ and $A_2$, e.g., $S_1^{A_1}$ and $S_3^{A_2}$, have a small portion of overlapping, and the overlapping region indicates the instances that have the same attribute values on $A_1$ and $A_2$.

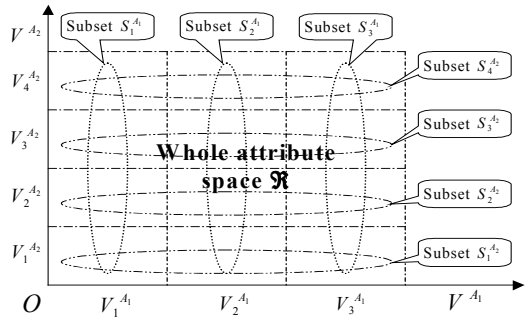



**Fig. 2. An example of subset partitioning with two attributes ($V^{A_1}$ and $V^{A_2}$)**

After we have constructed $\sum_i^M n_i$ subsets by all attributes, we calculate the classification accuracy of each base classifier $C_1,..C_j, ..,C_L$ on each of these $\sum_i^M n_i$ subsets, and we evaluate the classification accuracy of each base classifier on each subset with the procedure in Fig. 3. We denote the acquired accuracy matrix by $Acy_{C_j}^{S_l^{A_i}}, i=1,..,M;\ l=1,..,n_i;\ j=1,..,L$.

Finally, the classification accuracy of each base classifier from all constructed subsets ($\sum_i^M n_i$) can be worked out. An example accuracy matrix is given in Fig. 4, where the first column denotes the base classifiers, and the first row represents the subsets constructed from all attribute values. Each $(i, j)^{\text{th}}$ cell indicates the classification accuracy of classifier $C_i$ on subset $S_j$.

**Procedure** **GenerateAccuracyMatrix()**
**Input:** An evaluation set $Z$
**Output:** The accuracy matrix $Acy_{C_j}^{S_l^{A_i}}$

(1) **For** $i$=1; $i \le M$; $i$++
(2) For each attribute $A_i$
(3) ParitionEvaluationSet($A_i$);
(4) For $l$=1; $l \le n_i$; $l$++
(5) **For** $j$=1; $j \le L$; $j$++
(6) Calculate $Acy_{C_j}^{S_l^{A_i}}$, i.e., the classification accuracy of classifier $C_j$ on dataset $S_l^{A_i}$;
**End**
**End for**

**Fig. 3. Evaluating each base classifier on constructed subsets**

Subsets from attribute $A_1$ | Subsets from attribute $A_i$ | Subsets from attribute $A_M$

| classifiers | $S_1^{A_1}$ | … | $S_{n_1}^{A_1}$ | … | $S_1^{A_i}$ | … | $S_{n_i}^{A_i}$ | … | $S_1^{A_M}$ | … | $S_{n_M}^{A_M}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $C_1$ | 0.81 | … | 0.67 | … | | | | | | | |
| $C_2$ | 0.79 | … | 0.71 | … | | | | | | | |
| … | … | … | … | … | | | | | | | |
| $C_L$ | 0.83 | … | 0.82 | … | | | | | | | |

**Fig. 4. The classification accuracy from the base classifiers on all constructed subsets**

## 3.2 Dynamic Classifier Selection

With the acquired classification matrix, $Acy_{C_j}^{S_l^{A_i}}$, $i$=1, .., $M$, $l$=1, .., $n_i$; $j$=1,.., $L$. given a test instance $\hat{I}_x$, AO-DCS performs classifier selection with the procedure below:

**Procedure** **DynamicClassifierSelection**( $\hat{I}_x$ )
**Input:** A test instance $\hat{I}_x$ and a classification matrix $Acy_{C_j}^{S_l^{A_i}}, i=1,..,M;\ l=1,..,n_i;\ j=1,..,L$
**Output:** The best classifier for instance $\hat{I}_x$, $C^*(\hat{I}_x)$.

(1) AverageAcy[] $\leftarrow$ 0;
(2) **For** $j$=1; $j \le L$ ; $j$++
(3) **For** $i$=1; $i \le M$; $i$++
(4) AverageAcy[$j$]= AverageAcy[$j$]+ $Acy_{C_j}^{S_l^{A_i}} \mid l = \arg_b \{V_b^{A_i} = \hat{I}_x^{A_i};\ b = 1,..n_i\}$;
**End for**
(5) AverageAcy[$j$]= AverageAcy[$j$] / $M$;
**End for**
(6) $C^*(\hat{I}_x) = C_k(\hat{I}_x) \mid k = \arg_j \{\max_{j=1,2,...,L}\{AverageAcy\ [j]\}\}$;

**Fig. 5. Dynamic classifier selection of AO-DCS**

In Fig. 5, $\hat{I}_x^{A_i}$ denotes $\hat{I}_x$*'s* value on $A_i$. Given $\hat{I}_x$, *AO-DCS* first acquires $\hat{I}_x$*'s* values on all attributes $\hat{I}_x^{A_i}$, *i=1,..,M.* These attribute values can be used to find specific subsets that were constructed by similar instances in the evaluation set. The classifier that receives the highest average accuracy with all selected subsets is identified as the "best" classifier for $\hat{I}_x$.

### 3.3 Remarks on Relevant Research Efforts

Our method can accommodate both missing values and numerical attributes. A missing value is treated as a specific additional value. Numerical attributes can be converted into nominal ones using discretization techniques. In our system, we adopt the *k*-means clustering algorithm [15] to convert numerical attributes into nominal ones.

By using attribute values to partition the evaluation set into subsets in advance, *AO-DCS* works similar to the static dynamic classifier selection in [16] where it also partitions the evaluation set into subsets to evaluate the performance of base classifiers. However, there are two key distinctions:

- The method in [16] directly assigns the best classifier for each predefined region. *AO-DCS*, however, creates some small regions by using the attribute values from the evaluation set, and uses each test instance to determine the final subsets for evaluating the base classifiers.
- Instead of adopting a clustering technique that has to use distance functions to find small regions, we use attribute values to partition an evaluation subset, which is more natural and intuitive from the data viewpoint.

By using attribute values to partition the evaluation dataset into subsets, our method is also somewhat similar to the rudimentary rule induction algorithm — 1R [17]. This method generates a one-level decision tree, which is expressed in the form of a set of rules that all test on one particular attribute. 1R is a simple, cheap method that often comes up with quite good rules for characterizing the structure in data. Perhaps this is because the structure underlying many real-world datasets is rudimentary, and just one attribute is sufficient to determine the class of an instance accurately. With the surprising results from the 1R algorithm, we can find that using a single attribute to explore the domain of expertise might be more reasonable to some degree, especially in noisy environments.

## 4. Applying AO-DCS in Data Stream Mining

In many applications, data is not static but arrives in data streams, and the stream data is also characterized by drifting concepts. In other words, the underlying data generation models, or the concepts that we try to learn from the data, are constantly evolving, even dramatically. Meanwhile, real-world stream data is never perfect and can often suffer from corruptions (noise) that may impact interpretations of the data, models created from the data and decisions made based on the data [25]. Due to the inherent huge volume of the size, it is actually hard to apply general data cleansing mechanisms [18] to stream data for better data quality. Therefore, the above two facts (concept drifting and noise) imply that the classifiers learned from a small portion of the stream may vary significantly in performance, which makes *DCS* a promising solution for effective mining from a real-world data stream. In this section, we propose a framework to apply *AO-DCS* to mine noisy data streams. This framework is general enough to allow any exiting learning algorithms to be incorporated to handle any real-world data streams.

As shown in Fig. 6, we first partition streaming data into a series of chunks, $S_1$, $S_2$, .. $S_i$,.., each of which is small enough to be processed by an induction algorithm at one time. Then we learn a base classifier $C_i$ from each chunk $S_i$. To evaluate all base classifiers (in the case that the number of base classifiers is too large, we can keep only the most recent $K$ classifiers) and determine the "best" one for each test instance, we will dynamically construct an evaluation set $Z$ (using the most recent instances, because they are likely consistent with the current test instances). When classifying a test instance, $I_k$, we will employ *AO-DCS* (and the evaluation set $Z$) to integrate existing base classifiers and select the "best" classifier to classify $I_k$.

In real-world situations, many data streams contain a certain level of noise. There are two common types of noise: attribute noise and class noise [18]. In this paper, we assume the data stream suffers from a certain level of class noise (which means the errors are introduced in the class labels), and will extensively evaluate the performance of different *DCS* schemes in handling noisy data, where various levels of class noise are manually introduced (before the data partitioning) to simulate real-world scenarios. This should provide interested readers with valuable knowledge about mining from real-world data streams.

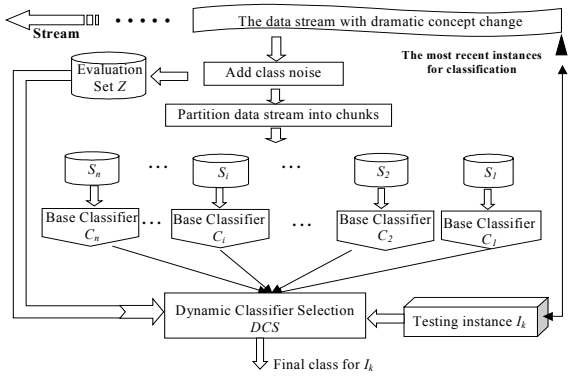



**Fig. 6. Applying DCS in noisy data streams**

## 5. Experimental Results and Comparisons

In this section, we design two sets of experiments, (1) *DCS* from classifiers that vary significantly in performance; and (2) *DCS* in classifying noisy datasets, to evaluate the performance of our proposed scheme. We take 8 datasets (including synthetic [19] and real-world data from the *UCI* database repository [20]) as benchmark data streams (by assuming the data comes in a time series manner) to evaluate the system performance.

Our purpose of the first set of experiments is to evaluate *DCS* in mining data streams with dramatic concept drifting. Unfortunately, although mining data streams with concept drifting has been addressed by many research efforts, we cannot find any benchmark dataset with dramatic concept changes (most existing efforts evaluate their algorithms with synthetic data). So we use the following design to simulate the scenarios in this regard. Given a dataset $X$, we first execute c4.5rules [21] on $X$ to learn a classifier $C_0$, and then split $X$ into $2^\eta$ chunks (using proportional partitioning [14]), and randomly select one chunk to induce a base classifier $C_\eta$, until we get $\eta$ base classifiers $C_1$, $C_2$,.. $C_\eta$. Normally, given a dataset $X$, the less the instances are used for training, the worse is the learned classifier in addressing the genuine concept of $X$. Therefore, from $C_1$ to $C_\eta$, the classifier becomes weaker and weaker in performance (we assume it is the result of the dramatic change of the underlying concept). We use $C_1$, $C_2$,.., $C_\eta$ to evaluate the proposed *DCS* algorithms. In the second set of experiments, *DCS* schemes are used to integrate classifiers learned from noisy datasets.

We compare the proposed *AO-DCS* with four multiple classifier mechanisms: *SAM* [7], *CVM* [5], *DCS_LA* [10] and *Referee* [9]. For *DCS_LA*, we use the overall accuracy from the $k$ nearest neighbors to evaluate the local accuracy. Meanwhile, we set $k$=10 for all experiments (as recommended by the original authors). For *Referee*, we use decision rules [21] as the referee, because decision rules are easier to manage. To compare the results of *DCS* schemes in classifying noisy data streams, we implement the *Arbiter* scheme proposed by Chang [14].

For each experiment, we execute 10-fold cross-validation and use the average accuracy as the final result. The classification accuracy in all tables and figures below denotes the accuracy evaluated from the test sets.

To add class noise, we adopt a pairwise scheme [18]: given a pair of classes ($\varepsilon_x$, $\varepsilon_y$) and a noise level $\gamma$, an instance with its label $\varepsilon_x$ has a $\gamma$·100% chance to be corrupted and mislabeled as $\varepsilon_y$, so does an instance of class $\varepsilon_y$. We use this method because in realistic situations, only certain types of classes are likely to be mislabeled. With this scheme, the percentage of the entire training set that is corrupted will be less than $\gamma$·100% because only some pairs of classes are considered problematic. In experiments below, we corrupt only one pair of classes (usually the pair of classes with the highest proportion of instances) in each dataset and only report the value $\gamma$ in all tables and figures.

## 5.1 DCS from Classifiers Vary Significantly in Performance

### 5.1.1 Classification accuracy of base classifiers

To evaluate whether base classifiers actually vary significantly in performance, we add different levels of noise into the data and generate 5 ($\eta$=5) base classifiers. We then compare the accuracy of each base classifier with *SAM*, *CVM* and *AO-DCS*, and demonstrate the results in Table 1, where the first column indicates the noise level, columns 2 to 6 represent the accuracy of each base classifier, and column 7 is the average accuracy of all base classifiers.

Table 1 illustrates that in most situations, the accuracy from $C_1$ to $C_5$ is getting worse. However, in a single run, there may have exceptions, Fig. 7 shows the accuracy of the base classifiers from 10 runs (with 15% noise). One can find that from $C_1$ to $C_5$, the overall accuracy is getting worse, but in a single run, the accuracy of the base classifiers can be different from this trend. E.g., in the third run of Fig. 7, the accuracy of $C_3$ is better than $C_1$ and $C_2$. Actually, the same phenomenon has been found from some of the other datasets. From Fig. 7, one can find that by dynamic selecting the "best" classifier, the *AO-DCS* outperforms all base classifiers in any single run.

Table 1 also indicates that at most noise levels, the classification accuracy from the classifier selection schemes (*CVM* and *AO-DCS*) is better than the combination scheme (*SAM*). However, when the noise becomes extremely serious, the results from *SAM* become closer to (or even better than) *CVM* and *AO-DCS*, as demonstrated on the eighth to tenth columns of Table 1. One possible reason is that when the noise level increases, each base classifier becomes weaker. In the case that all base classifiers have a low confidence, combing results from base classifiers becomes more reasonable. Comparing the results from *CVM* and *AO-DCS*, one can find that the latter outperforms the former at almost every noise level. As we analyzed before, *CVM* statistically selects the single best classifier by evaluating the overall performance of the base classifiers. And by integrating the proposed *DCS* scheme, we can explore the merit of each base classifier and improve the system performance.

**Table 1. Classification accuracy of base classifiers with various MCS schemes (Car dataset, 5 base classifiers)**

| Noise Level % | $C_1$ (%) | $C_2$ (%) | $C_3$ (%) | $C_4$ (%) | $C_5$ (%) | AVG (%) | SAM (%) | CVM (%) | AO-DCS (%) |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 88.5 | 85.1 | 81.6 | 77.9 | 76.1 | 81.8 | 85.4 | 88.5 | 88.8 |
| 10 | 87.0 | 84.0 | 78.5 | 73.8 | 71.9 | 79.1 | 83.7 | 87.9 | 88.2 |
| 20 | 80.4 | 77.6 | 73.0 | 73.7 | 71.6 | 75.2 | 80.4 | 80.4 | 81.2 |
| 30 | 74.0 | 74.5 | 68.3 | 64.2 | 63.9 | 69.0 | 76.8 | 74.6 | 76.4 |
| 40 | 61.4 | 62.9 | 61.0 | 62.8 | 57.4 | 61.1 | 67.1 | 60.7 | 63.4 |

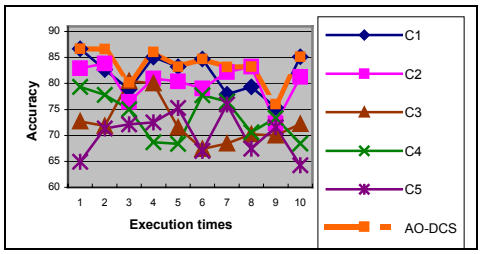



**Fig. 7. Classification accuracy of base classifiers and AO-DCS from 10 runs (Car dataset, 15% noise)**

### 5.1.2 Comparative studies on accuracy

In this subsection, we compare *AO-DCS* with three *CS* algorithms (*DCS_LA*, *Referee*, and *CVM*) and one *CC* scheme (*SAM*). We add various levels of noise into original data (and the evaluation set) to evaluate the performance of various algorithms in noisy environments. Table 2 shows the results from the Car dataset, where the first column indicates the noise level and the other columns denote the accuracy from different methods.

From Table 2, one can find that with five base classifiers varying in degrees of accuracy, the *CS* schemes achieve better performances than the *CC* scheme (*SAM*) at most noise levels. When the noise level is low, all *CS* methods outperform *SAM* with *DCS_LA* attaining the highest performance. However, with the increase of the noise level, *DCS_LA* receives the most dramatic decline in comparison with Referee and *AO-DCS*. Meanwhile, although the accuracy of Referee is usually lower than *AO-DCS* and *DCS_LA*, it was found to be the most noise tolerant, i.e., it receives the lowest decline caused by the increase of the noise level. In Table 2, when noise increases from 0% to 40%, the performance of *DCS_LA* and *AO-DCS* drops 30.57% (from 90.65% to 62.94%) and 29.58% respectively. But, Referee receives only 21.14% decrease. Actually, our analysis in Section 5.2 will indicate that Referee is more sensitive to the increase of the chunk number, and *DCS_LA* is more sensitive to noise.

**Table 2. DCS accuracies from Car dataset (5 base classifiers)**

| Noise Level (%) | SAM (%) | CVM (%) | Referee (%) | DCS_LA (%) | AO-DCS (%) |
|---|---|---|---|---|---|
| 0 | 85.72 | 89.23 | 89.56 | 90.65 | 89.73 |
| 10 | 82.33 | 82.75 | 85.03 | 87.33 | 85.46 |
| 20 | 78.45 | 77.83 | 80.31 | 80.26 | 78.15 |
| 30 | 71.86 | 73.03 | 77.43 | 73.84 | 74.65 |
| 40 | 61.62 | 61.49 | 70.63 | 62.94 | 63.19 |

### 5.1.3 Comparative studies on efficiency

The time complexity of *DCS* schemes comes from two phases: 1) evaluate the classification accuracy of base classifiers with predefined subsets, $TS_1$; and 2) select the "best" classifier for test instances, $TS_2$. Given a dataset $D$ with $M$ attributes, assume the numbers of instances in the evaluation and test sets are $N_Z$ and $N_Y$ respectively, and the number of induced base classifier is $L$. Assume further that *DCS_LA* selects $k$ nearest neighbors for each test instance, and no indexing structure is adopted to facilitate the *kNN* search. For each test instance, $TS_1$ of *DCS_LA* is zero, and $TS_2$ is the time to go through all $N_Z$ evaluation instances to find the $k$ nearest neighbors and rank $L$ base classifiers accordingly, which is $O(N_Z \log k + L \log L)$. For $N_Y$ test instances, the total complexity is denoted by Eq. (1)

$$Complexity_{DCS\_LA} = O(\underbrace{0}_{TS_1} + \underbrace{N_Y N_Z \log k + N_Y L \log L}_{TS_2}) \quad (1)$$

For the *Referee* method, $TS_1$ is the time to construct the referee for each base classifier by using instances in the evaluation set, and we assume a quadratic complexity for such a procedure (Although other better (less that quadratic) learning algorithms are available, we assume the quadratic complexity for the worst case), where the complexity to construct $L$ referees is $O(L \cdot (N_Z)^2)$. $TS_2$ is the time to pass all referees and select the "best" classifier, which is $O(L \log L)$. Therefore for $N_Y$ test instances, the total complexity denoted by Eq. (2).

$$Complexity_{Referee} = O(\underbrace{L \cdot (N_Z)^2}_{TS_1}) + \underbrace{N_Y L \log L}_{TS_2}) \quad (2)$$

For *AO-DCS*, $TS_1$ is the time to evaluate the accuracy of each base classifier with the subsets constructed by attributes, where each classifier needs to go through the evaluation set for $M$ times (where $M$ is the number of attributes). So $TS_1$ of *AO-DCS* is $O(LMN_Z)$. $TS_2$ for *AO-DCS* is the time to select the "best" classifier that has the highest accuracy with the subsets determined by the test instance, which is $O(M \log L)$. For $N_Y$ test instances the total complexity of *AO-DCS* is denoted by Eq. (3).

$$Complexity_{AO-DCS} = O(\underbrace{LMN_Z}_{TS_1} + \underbrace{N_Y M \log L}_{TS_2}) \quad (3)$$

For normal datasets, it is obvious that ($M$, $L$, $k$) << ($N_Y$, $N_Z$), therefore complexity of *AO-DCS* appears to be the lowest one.

In addition to the above theoretical analysis, we also perform an empirical analysis. Table 3 shows the execution times of these three schemes from the Mushroom dataset, where we present the actual execution time (in seconds) of each method and their ratio values (We used a PC with Intel Pentium 4 with 2 GHz speed and 512 MB memory). In Table 3, the first column indicates the percentage of the size of the evaluation set in comparison with the size of the whole dataset, and *TS* is the sum of $TS_1$ and $TS_2$. *RatioA* and *RatioB* are given in Eqs. (4) and (5).

$$RatioA = TS \text{ of } Referee \,/\, TS \text{ of } AO\text{-}DCS \quad (4)$$

$$RatioB = TS \text{ of } DCS\_LA \,/\, TS \text{ of } AO\text{-}DCS \quad (5)$$

From the results in Table 3, one can find that *AO-DCS* comprehensively improves the system performance in terms of time efficiency. When the size of the evaluation set increases, both *DCS_LA* and *Referee* suffer from spending a lot of time on finding the nearest neighborhood or inducing referee rules from the evaluation set, which is obviously a nonlinear increase with the size of the evaluation set. But increase the size of the evaluation set has less influence on *AO-DCS*.

## 5.2 DCS in Classifying Noisy Data Datasets

### 5.2.1 Experiments with noise levels

To evaluate the performances of *DCS* schemes in classifying partition-based noisy data datasets, we equally split the dataset into 9 non-overlapping chunks (using proportional sampling). We use c4.5rules to construct a base classifier from each chunk, and apply *DCS* schemes to integrate these base classifiers. In addition, we also add certain levels of class noise into the original data. The experimental results are shown in Tables 4 and 5, where the first column indicates the noise level, the second to seventh columns represent the classification accuracy of each scheme, and the eighth and ninth columns give the average and variance of the 9 base classifiers' accuracy.

From Tables 4 and 5, when we compare the performances of the three *DCS* schemes with *SAM*, *CVM* and *Arbitrator*, we can find that *DCS* schemes receive relatively better performances at various noise levels. If we compare the second and eighth columns, we can find that *SAM*'s accuracy is higher than the average accuracy of all base classifiers, where the improvement from *SAM* can be 2% to 10% compared to the average accuracy of all base classifiers. This confirms that *SAM* works surprisingly (or embarrassingly) well in many circumstances. From the ninth column, one can find that when the noise level increases, the variance of the base classifiers' accuracy becomes large, which indicates that the performance of the base classifier varies significantly. Usually, if the variance of the base classifiers' accuracy becomes significant, *DCS* can receive relatively large improvement.

When we compare the fourth to sixth columns, we can find *DCS_LA* is more sensitive to noise. When the noise level increases, the performance of *DCS_LA* decreases dramatically. The reason is that this method uses the nearest neighbors to evaluate base classifiers, and in high noise-level environments, the selected neighbors may be seriously corrupted by noise. Both *Referee* and *AO-DCS* use statistical information of the evaluation set, so noise has less negative impact on them.

**Table 3. Execution time (in seconds) of different DCS schemes from Mushroom dataset (20% noise, 5 base classifiers)**

| Evaluation Set (%) | *Referee* (s) | | | *DCS_LA* (s) | | | *AO-DCS* (s) | | | *RatioA* | *RatioB* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $TS_1$ | $TS_2$ | TS | $TS_1$ | $TS_2$ | TS | $TS_1$ | $TS_2$ | TS | | |
| 10 | 1.040 | 0.034 | 1.074 | 0 | 0.788 | 0.788 | 0.186 | 0.039 | 0.225 | 4.77 | 3.50 |
| 30 | 7.450 | 0.030 | 7.480 | 0 | 2.178 | 2.178 | 0.346 | 0.039 | 0.385 | 19.43 | 5.66 |
| 50 | 29.990 | 0.034 | 30.024 | 0 | 3.63 | 3.630 | 0.582 | 0.037 | 0.619 | 48.50 | 5.86 |
| 70 | 74.280 | 0.038 | 74.318 | 0 | 5.210 | 5.210 | 0.701 | 0.037 | 0.738 | 100.70 | 7.06 |
| 90 | 127.609 | 0.034 | 127.643 | 0 | 6.842 | 6.842 | 0.808 | 0.036 | 0.844 | 151.25 | 8.11 |
| 100 | 173.689 | 0.032 | 173.721 | 0 | 7.824 | 7.824 | 0.861 | 0.041 | 0.902 | 192.60 | 8.67 |

When we compare *SAM* (the second column) and *Arbiter* (the seventh column), one can find that *Arbiter* generally outperforms *SAM* in most situations. However, when the noise level reaches a relatively high level (30% ~ 40%), the improvement from *Arbiter* disappears. The reason is that when the noise level goes higher, more noisy instances are used to train the Arbiter. Consequently, the ability of the learned Arbiter becomes weaker.

**Table 4. Experimental comparisons at different noise levels from Car dataset (9 chunks)**

| Noise level (%) | SAM (%) | CVM (%) | Referee (%) | DCA_LA (%) | AO-DCS (%) | Arbiter (%) | Avg. (%) | Var. (%) |
|---|---|---|---|---|---|---|---|---|
| 0 | 84.4 | 84.6 | 85.5 | 91.3 | 86.6 | 86.0 | 81.5 | 2.4 |
| 10 | 83.1 | 82.2 | 81.8 | 89.2 | 83.8 | 83.9 | 79.1 | 3.8 |
| 20 | 80.1 | 80.5 | 78.8 | 83.9 | 82.0 | 81.0 | 76.2 | 6.2 |
| 30 | 76.0 | 76.9 | 78.1 | 80.2 | 80.4 | 79.0 | 70.7 | 8.7 |
| 40 | 71.6 | 67.1 | 74.3 | 64.6 | 73.7 | 69.5 | 65.6 | 11.3 |

**Table 5. Experimental comparisons at different noise levels from Krvskp dataset (9 chunks)**

| Noise level (%) | SAM (%) | CVM (%) | Referee (%) | DCA_LA (%) | AO-DCS (%) | Arbiter (%) | Avg. (%) | Var. (%) |
|---|---|---|---|---|---|---|---|---|
| 0 | 97.1 | 97.2 | 97.4 | 98.2 | 97.5 | 96.8 | 95.7 | 0.3 |
| 10 | 95.1 | 95.8 | 96.0 | 97.0 | 95.9 | 95.7 | 93.5 | 0.9 |
| 20 | 93.9 | 93.8 | 94.2 | 93.5 | 94.3 | 93.7 | 90.9 | 1.2 |
| 30 | 92.2 | 92.4 | 92.7 | 84.5 | 93.1 | 91.1 | 82.9 | 10.1 |
| 40 | 72.9 | 73.0 | 76.9 | 66.9 | 75.2 | 71.6 | 62.6 | 13.2 |

### *5.2.2 Experiments with variable number of chunks*

In this subsection, we address the impact of the number of chunks on the performance of the proposed algorithms. We partition the dataset into a different number of chunks (from 3 to up to 63). Meanwhile, we run the experiments at two noise levels (0% and 25%), and show the results in Figs. 9 to 14. From Figs. 9 to 14, we can find that when the number of chunks increases, the overall accuracy from all schemes decreases. However, in a certain range, the chunk number may have less impact on the classification accuracy (and the increase of the number of chunks can even increase the classification accuracy). This phenomenon might come from the intrinsic characteristics of each dataset, e.g., the level of redundancy.

For noise-free datasets, *DCS_LA* can usually acquire the best performance. However, in noisy environments, *DCS_LA* receives less improvement, especially when the number of chunks is relatively large. As shown in Fig. 9(b), where the noise level is 25%, while the number of chunks increases, the performance of *DCS_LA* receives less improvement than *AO-DCS*. The same phenomenon has been found from most other datasets (except from Krvskp, in Fig. 11 (b), where the Referee scheme receives the highest classification accuracy).

When comparing the three *DCS* schemes, we find *Referee* is most sensitive to the number of chunks. Take Fig. 9 as an example. When the chunk number increases from 3 to 63, the performance of *Referee* drops 19.79% (from 90.24% to 72.38%) which is the largest among all three *DCS* schemes (the drop of *DCS_LA* and *AO-DCS* in the same range is 3.93% and 10.17% respectively). The reason behind this phenomenon is that with the increase of the chunk number, the number of instances in each chunk is decreased. Each learned base classifier then tends to bias to only one or two attributes. Consequently, the learned referee of each classifier cannot comprehensively partition the evaluation set into small subsets to explore the domain expertise of each base classifier. Moreover, as we have mentioned in Section 2, Referee uses different subsets to evaluate different base classifiers (each classifier has its own referee), and without the same measurements, the selected "best" classifier might not work well.

The experimental results in Figs. 9 to 14 indicate that with a large number of chunks, the *Arbiter* scheme likely receives the same classification accuracy as *SAM*. The reason is that with the increase of the number of chunks (the number of base classifiers), the learned Arbiter will have less influence on the classification accuracy, because the arbitration rules take effect only if the base classifiers cannot have a majority classification. Meanwhile, the existence of noise could also be fatal to the Arbiter, because the arbiter is learned from uncertain data collected from different chunks. In noisy datasets, there is no doubt that the uncertain data collected from different chunks usually contain significant noise. Then, the learned arbiter has a very limited ability to improve the system performance.

We have performed comparisons with different datasets at two noise levels (0% and 25%). One can go through all figures below to get detailed comparisons. Clearly, these results indicate that *DCS* acquires much better performance than simple classifier combining, especially when a large number of base classifiers is adopted. This conclusion supports our initial motivation that each base classifier has its own merit that deserves exploitation, and finding the domain of expertise of each classifier supplies a new way to improve the system performance.

**Notes:** Figs. 9 to 14 report the number of chunks and the system performance on 6 datasets, where (a) is tested on noise-free data; and (b) is tested with 25% class noise. In Figs. 9 to 14, the *x*-axis denotes the number of chunks and the y-axis represents the classification accuracy, and each curve denotes one method that is specified in Fig. 8.

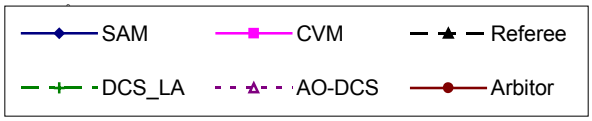
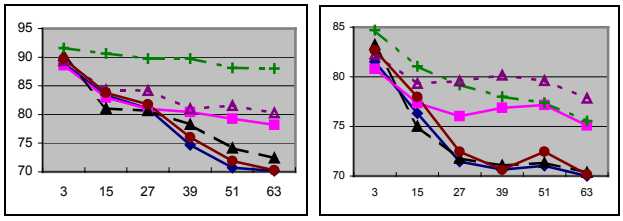
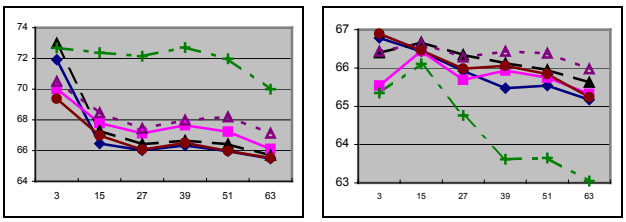



**Fig. 8. The meaning of each curve in Figs 9 to 14**



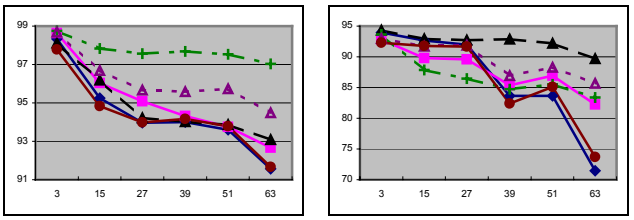

(a) (b)

**Fig. 9. Results from Car dataset**



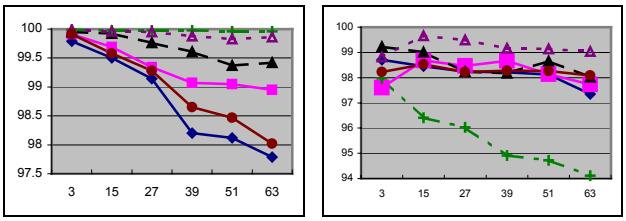

(a) (b)

**Fig. 10. Results from Connect-4 dataset**



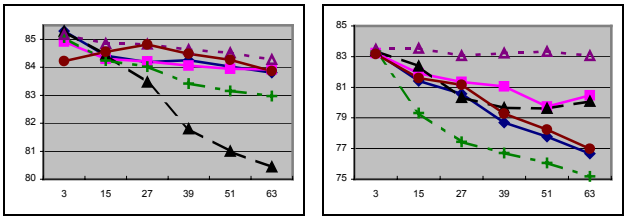

(a) (b)

**Fig. 11. Results from Krvskp dataset**



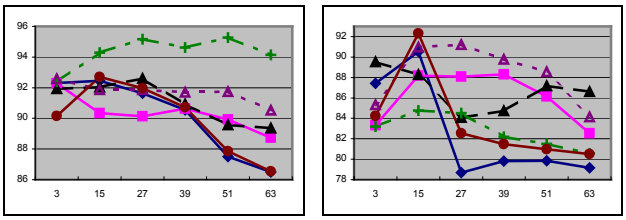

(a) (b)

**Fig. 12. Results from Mushroom dataset**




(a) (b)

**Fig. 13. Results from Adult dataset**




(a) (b)

**Fig. 14. Results from WDBC dataset**

## 6. Conclusions

Traditional data mining algorithms are challenged by two most important features of data streams: huge volumes of data and the underlying concept drifting. These two challenges raise the need for incorporating ensemble classifiers in existing stream mining efforts. This intuitive solution, however, ignores the fact that the base classifier learned from a portion of a data stream carries two important features: (1) weak (even conflictive with others) in overall performance; but (2) still reliable in a specific domain. This fact becomes especially clear if the data stream suffers from dramatic concept drifting or a significant amount of noise. Consequently, instead of adopting any *combination* scheme, we have presented a new *DCS* algorithm that selects the "best" classifier once a time for each test instance in the data stream. We use each attribute to partition the evaluation set into disjoint subsets. We evaluate the classification accuracy of each base classifier on each subset. This accuracy indicates the ability of the base classifiers in classifying the instances characterized by each attribute, and hopefully, helps us explore the merit of each base classifier. Given a test instance, its attribute values will determine the subsets that have been constructed by similar instances in the evaluation set. The base classifier that has the highest accuracy with all these determined subsets is selected to classify the test instance.

Our experimental results have demonstrated that in comparison with classifier combination schemes, the *DCS* algorithms can possibly acquire more accuracy improvement, especially when the performances of the base classifiers vary significantly. When the real-world data suffers from dramatic concept drifting or a certain level of noise, *AO-DCS* turns to be a better choice in integrating multiple classifiers to enhance the system performance.

## References


[1] Ali K. & Pazzani M., Error reduction through learning multiple description, *Machine Learning, vo.24, no.3, 1996*.

[2] Huang Y. S. & Suen C. Y., A method of combining multiple experts for the recognition of unconstrained handwritten numerals, *IEEE Trans. on PAMI, 17(1), pp.90-94, 1995*.

[3] Ueda N., Optimal linear combination of neural networks for improving classification perform., *IEEE Trans. on PAMI, 22, 2000*.

[4] Xu L., Krzyak A. & Suen C., Methods of combining multiple classifiers and their application to handwriting recognition, *IEEE Trans. on Sys., Man and Cyber., 22, 1992*.

[5] Schaffer C., Selecting a classification method by cross-validation, *Machine Learning, vol.13, pp.135-143, 1993*.

[6] Breiman L., Friedman J. H., Olshen R.A. & Stone C. J., Classification and regression trees, *Belmont, CA, 1984*.

[7] Merz C. J., Dynamical selection of learning algorithms, *In: D.Fisher, H.-J.Lenz (eds.), Learning from Data, Artificial Intelligence and Statistics, Springer-Verlag, NY (1996)*.

[8] Merz C. J., Using correspondence analysis to combine classifiers, *Machine Learning, vo.36 (1-2), pp.33-58, 1999*.

[9] Ortega J., Koppel M. & Argamon S., Arbitrating among competing classifiers using learned referees, *Knowledge and Information Systems, vol.3, no.4, 2001*.

[10] Woods K., Kegelmeyer W. P. & Bowyer K., Combination of multiple classifiers using local accuracy estimation, *IEEE Transactions on PAMI, vol.19, no.4, Apr., 1997*.

[11] Wang H., Fan W., Yu P. & Han J., Mining concept-drifting data streams using ensemble classifiers, Proc. of KDD 2003.

[12] Breiman L., Stacked regressions, *Machine Learning*, 24, 1996.

[13] Schapire R., The strength of weak learnability, *Machine Learning*, vol.5, no.2, pp.197-227, 1990.

[14] Chan P., An extensible meta-learning approach for scalable and accurate inductive learning, *Ph.D thesis, Columbia Univ., 1996*.

[15] Jain Anil K. & Dubes R. C., Algorithms for clustering data. *Prentice Hall, 1998*.

[16] Kucheva L.I., Switching between selection and fusion in combining classifiers: An experiment, *IEEE Trans. SMC, 32(2), 2002*.

[17] Holte R.C., Very simple classification rules perform well on most commonly used datasets, *Machine Learning, 11, 1993*.

[18] Zhu X., Wu X. & Chen Q., Eliminating class noise in large datasets, *Prof. of 20th ICML Conf.*, Washington DC, 2003.

[19] IBM Almaden Research, Synthetic data generator, http://www.almaden.ibm.com/software/quest/Resources/datasets/syndata.html#classSynData

[20] Blake C. L. & Merz, UCI Data Repository, 1998.

[21] Quinlan R., C4.5 programs for machine learning, San Mateo, CA, Morgan Kaufmann publisher, 1993.

[22] Domingos P. & Hulten G., Mining high-speed data streams, *Proc. of SIGKDD*, 2000.

[23] Nasraoui O., Cardona C., Rojas C. & González F., TECNO-STREAMS: Tracking Evolving Clusters in Noisy Data Streams with a Scalable Immune System Learning Model, *Proc. of ICDM*, 2003.

[24] Kolter J. & Maloof M., Dynamic weighted majority: a new ensemble method for tracking concept drift, *Proc. of ICDM*, 2003

[25] Zhu X. & Wu X., Class noise vs attribute noise: A quantitative study of their impacts, *Artificial Intelligence Review, in press*, 2004.