Expert Systems with Applications 40 (2013) 3606–3622

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

# Single Classifier-based Multiple Classification Scheme for weak classifiers: An experimental comparison

Albert Hung-Ren Ko [a,*], Robert Sabourin [b]

[a] Laboratoire de communication et d'intégration de la microélectronique, École de Technologie Supérieure, University of Quebec, 1100 Notre-Dame West Street, Montreal, Quebec, Canada H3C 1K3

[b] Laboratoire d'imagerie, de vision et d'intelligence artificielle, École de Technologie Supérieure, University of Quebec, 1100 Notre-Dame West Street, Montreal, Quebec, Canada H3C 1K3

ARTICLE INFO

*Keywords:*
Ensemble of classifiers
Multiple Classifier System
Bagging
Boosting
Diversity
Multiple Classification Scheme

ABSTRACT

In this paper, we propose a Single Classifier-based Multiple Classification Scheme (SMCS) that uses only a single classifier to generate multiple classifications for a given test data point. The SMCS does not require the presence of multiple classifiers, and generates diversity through the creation of pseudo test samples. The pseudo test sample generation mechanism allows the SMCS to adapt to dynamic environments without multiple classifier training. Moreover, because of the presence of multiple classifications, classification combination schemes, such as majority voting, can be applied, and so the mechanism may improve the recognition rate in a manner similar to that of Multiple Classifier Systems (MCS). The experimental results confirm the validity of the proposed SMCS as applicable to many classification systems. Even without parameter selection, the average performance of the SMCS is still comparable to that of Bagging or Boosting. Moreover, the SMCS and the traditional MCS scheme are not mutually exclusive, and the SMCS can be applied along with traditional MCS, such as Bagging and Boosting.



## 1. Introduction

The purpose of pattern recognition systems is to achieve the best possible classification performance. In general, a number of classifiers could be tested and evaluated in these systems, and the most appropriate one chosen for the problem at hand. However, it has recently become common practice to use more than one classifier rather than a single one for pattern recognition tasks. This is because different classifiers usually make different errors on different samples, which means that by combining classifiers we can create an ensemble that makes more accurate decisions (Brown, Wyatt, Harris, & Yao, 2005; Kittler, Hatef, Duin, & Matas, 1998; Kuncheva, 2002; Kuncheva & Whitaker, 2003; Opitz & Maclin, 1999; Pekalska, Skurichina, & Duin, 2004; Woloszynski & Kurzynski, 2010; Zouari, Heutte, & Alimi, 2004). In order to have a pool of classifiers with different errors, it is advisable to create diverse classifiers. To achieve this, the classifiers are grouped together into what is known as an Ensemble of Classifiers (EoC) (Kittler et al., 1998), and the approach that uses multiple classifiers to enhance classification accuracy is known as a Multiple Classifier System (MCS).

One of the issues that is critical to the success of an EoC routine is the need for diversity in ensemble creation, because an EoC will not perform well without it (Kuncheva & Whitaker, 2003; Kuncheva, Skurichina, & Duin, 2002; Kittler et al., 1998; Ruta & Gabrys, 2005; Ruta & Gabrys, 2001). For an EoC to perform well, every classifier needs to make errors on different data points, so that when the EoC outputs are combined, most of the errors committed by classifiers will cancel each other out, and so the overall EoC will achieve a more accurate recognition rate (Fumera, Roli, & Serrau, 2008; Kittler et al., 1998; Ko, Sabourin, & de Souza Britto, 2006; Kuncheva, 2002; Kuncheva & Whitaker, 2003; Kuncheva et al., 2002; Ruta & Gabrys, 2001, 2005). This property is referred to as the diversity of an ensemble, and it has more recently become a constant in the creation of ensembles in the EoC community.

Traditionally, there have been several methods for generating diversity, and thereby creating diverse classifiers, among them Random Subspaces (Ho, 1998), Bagging, and Boosting (Grove & Schuurmans, 1998; Kuncheva et al., 2002; Schapire, Freund, Bartlett, & Lee, 1998). The Random Subspaces method, for example, creates various classifiers using different subsets of features to train them. Because problems are represented in different subspaces, different classifiers develop different borders for the classification. Bagging generates diverse classifiers by randomly selecting subsets of samples to train them (Fumera et al., 2008). In fact, intuitively, we would expect classifier behaviors to differ based on the sample subsets chosen.

* Corresponding author. Tel.: +1 514 577 9759.
*E-mail addresses:* drinkblue@gmail.com (A.Hung-Ren Ko), robert.sabourin@etsmtl.ca (R. Sabourin).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.12.067

From these conventional diverse classifier generation methods, we can observe that most EoCs are created using an overproduce-then-select strategy (Santos, Sabourin, & Maupin, 2008), by which an abundance of diverse classifiers is generated, and subsequently an optimal subset of classifiers is selected. Although such an approach is useful for enhancing classification accuracy, it requires a rather complex three-step process (Grove & Schuurmans, 1998; Ho, 1998; Kuncheva et al., 2002; Schapire et al., 1998): In the first step, multiple data subsets are generated; in the second step, multiple classifiers are trained with corresponding data subsets; and in the third step, adequate classifiers need to be selected from the pool of trained classifiers: at the first stage, some meaningful $N$ data subset partitions are required; at the second stage, $N$ classifiers are trained; and, at the last stage, an attempt is made to find the $M$ classifiers among the $N$ classifiers that will form the best ensemble, $M \leqslant N$. The justification for all these stages is that, on the one hand, we know that diversity is necessary to enhance system performance (Brown et al., 2005; Ko et al., 2006; Shipp & Kuncheva, 2002), but, on the other, we acknowledge that raw and unprocessed diversity may not lead to optimal performance (Ko, Sabourin, & de Souza Britto, 2009; Kuncheva & Whitaker, 2003; Ruta & Gabrys, 2001). In other words, diversity must be generated to enhance accuracy, but this is far from sufficient. What is needed in addition is classifier training and classifier selection, which greatly increase the system complexity. The three-step process of overproduction and selection, has, in fact, become a standard process in MCS, and its embedded complexity is accepted as an unavoidable cost.

What makes this issue even more complicated is that there is no universal definition of diversity, in spite of the fact that a number of different diversity measures have been proposed (Kuncheva & Whitaker, 2003). Furthermore, it has been observed that clear correlations between ensemble accuracy and diversity measures cannot be found in any of the existing diversity measures (Brown et al., 2005; Ko et al., 2009; Kuncheva & Whitaker, 2003; Kuncheva et al., 2002). For MCS, this fact has led some researchers to consider diversity measures to be unnecessary for ensemble selection (Ruta & Gabrys, 2005). To summarize, the concept of diversity does help, but both theoretical and experimental approaches show that strong correlations between diversity measures and ensemble accuracy are lacking.

Intriguingly, even though practitioners agree that generated diversity cannot guarantee classification accuracy enhancement without the training of multiple classifiers and classifier selection, the validity of the diversity generation process is rarely questioned. In the light of the lack of correlation between diversity measures and ensemble accuracy, practitioners still stay with the existing diversity generation process. So, in spite the fact that training multiple classifiers and classifier selection increase pattern recognition complexity, many practitioners are willing to accept the additional costs in the name of classification diversity. Is this the only way of applying MCS?

If we examine the process of conventional diversity generation carefully, we quickly conclude that all diversity generation procedures are actually based on data diversity. That is, by partially omitting selected samples from a sample pool for each classifier training operation, we create different data subsets (Grove & Schuurmans, 1998; Ho, 1998; Kuncheva et al., 2002; Schapire et al., 1998). Then, by using these data subsets to train classifiers, every classifier will be different from the others (see Fig. 1). This means that, de facto, diversity exists among classifiers, because each classifier uses a different data subset for its classifier training. There may be some overlaps of samples among data subsets, but no two data subsets are identical. In other words, diversity is generated through the makeup of the data subsets, and not by classifier training. So, classifier training does not create diversity, but simply transforms the diversity embedded in the data subsets, thereby creating different classifiers. From this viewpoint, we do not see any need to use the classifier training process to extract diversity from data, nor can we think of any reason why diversity cannot be extracted directly from data.

If we go further and examine the process of decision making based on the MCS, we can draw another conclusion, which is that final decisions in an MCS system are basically combinations of decisions made by different trained classifiers. Multiple classifiers yield multiple class labels for a given test sample, and we can combine these multiple class labels into a single class label by applying some fusion functions, such as majority voting (Kittler et al., 1998; Ruta & Gabrys, 2005), BKS (Huang & Suen, 1995), Naive Bayes (Shipp & Kuncheva, 2002), averaging, or multiplying (Tax, Van Breukelen, Duin, & Kittler, 2000). Given that each classifier actually draws a boundary between classes, the MCS obtains a new boundary by applying a fusion function, that is, de facto, a combination of different boundaries drawn by different classifiers, as shown in Fig. 2. However, this new boundary is solely dependent on classifiers and on the fusion function applied. By contrast, the true boundary, which always correctly classifies and is usually known as the oracle (Didaci, Giacinto, Roli, & Marcialis, 2005), could be quite fragmented and not able to be represented by a single boundary within the available dimensions. Ideally, we would like a boundary to be as close to the oracle as possible, in order to minimize classification errors. But, we should not be surprised that the ability of a new boundary to imitate the oracle shape is heavily constrained by the classifier boundaries, which are static, and by the number of

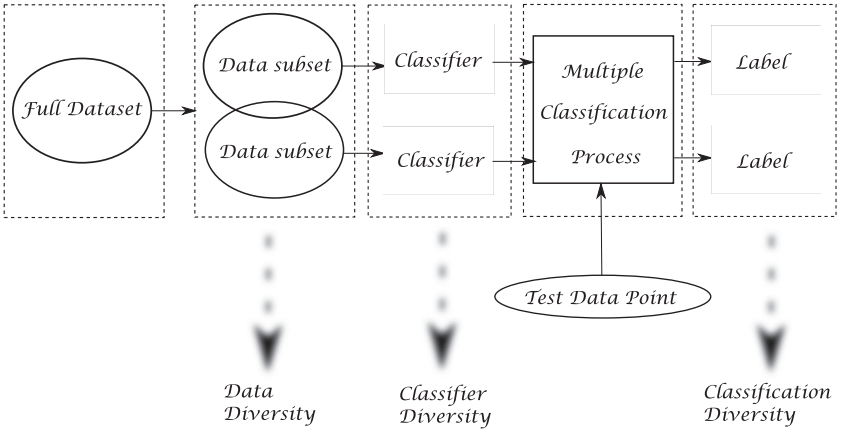


**Fig. 1.** Traditional multiple classifier system: diverse data splitting is implemented to generate diverse data subsets, which are then used to train classifiers. Consequently, classifier outputs are likely to be diverse classifications.

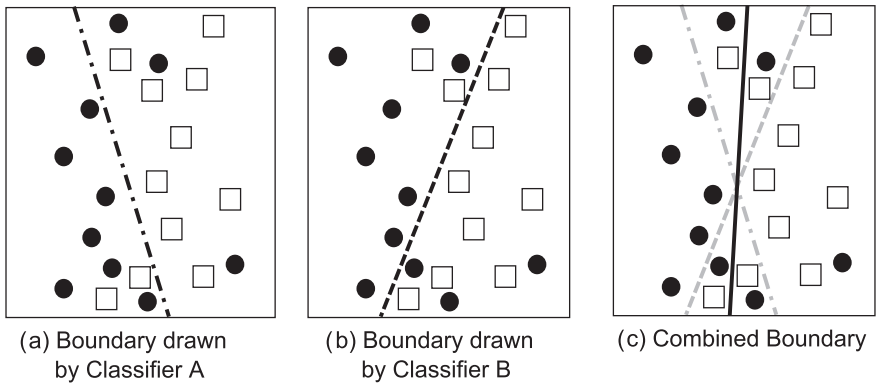


**Fig. 2.** Illustration of a boundary change by applying a classifier combination: dark circles represent samples from one class, and empty rectangles represent samples from another class; lines represent boundaries drawn by classifiers: (a) boundary drawn by classifier A; (b) boundary drawn by classifier B; and (c) new boundary created by combining the boundary from (a) and the boundary from (b), represented by a solid black line.

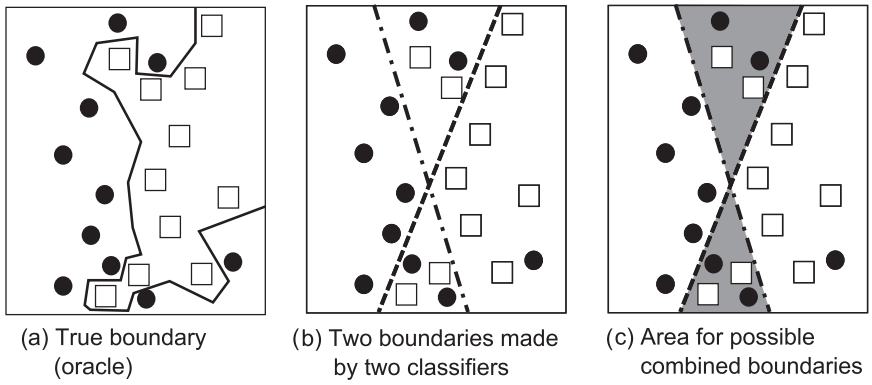


**Fig. 3.** Difficulty in achieving the oracle by classifier combination: dark circles represent samples from one class, and empty rectangles represent samples from another class; lines represent boundaries drawn by classifiers: (a) the true boundary, known as the oracle; (b) two boundaries drawn by two different classifiers; and (c) the area in which a new boundary may be drawn by combining two existing boundaries.

classifiers, which is limited. Fig. 3 shows that it can be rather difficult to achieve the oracle with a limited number of classifiers using a linear fusion function. From this viewpoint, we do not see classifier combination as the best way to enhance classification accuracy.

Many issues need to be resolved, such as:

- the possibility of extracting diversity from a dataset directly without training multiple classifiers,
- complexity reduction for the MCS by omitting multiple classifier training and classifier selection,
- the feasibility of multiple classifications without multiple classifiers by using diversity,
- the extent to which multiple classifications using diversity without multiple classifiers can improve classification accuracy, and
- the design of a Multiple Classification Scheme that is not constrained, by either a classifier boundary or the number of classifiers.

To summarize, we would simply like to have a decision boundary with the potential to be as close to the oracle as possible.

These issues are not attracting much attention in the literature, but we believe that they are fundamental, and may have a substantial impact on how much we can improve classification accuracy by applying the MCS. In an attempt to resolve them, we have designed a mechanism called a Single-Classifier-based Multiple Classification Scheme (SMCS) and tested its validity.

In this paper, we propose an unconventional SMCS approach that is similar to the MCS, but without the need to train multiple classifiers. We propose a mechanism that achieves multiple classifications with a single classifier, and so benefits from the logic of an MCS without repetitive classifier training (Fig. 4) and without classifier selection. Given a test sample to classify and some training samples, our method divides the training samples into two groups: one containing what we call reference samples, and the other containing what we call evaluation samples. We use different reference samples to generate different pseudo test data points, each of which constitutes a different combination of an original test sample and some reference samples (Fig. 5), and we use evaluation samples to select adequate reference samples for pseudo test data point generation. Because we use different reference samples to generate pseudo test data points, data diversity is extracted from the original training data in a way similar to that in MCS. Furthermore, because of the generation of multiple pseudo test data points for an original test sample, we can obtain multiple classifications for that test sample. Consequently, traditional classification combination schemes in the MCS, such as the majority voting fusion function, can be implemented to generate a final class label. The proposed method can somehow be related to local learning (Bottou & Vapnik, 1992), in which local information is exploited to facilitate classification task.

Note that the generation of pseudo data points to improve classification accuracy is not new. The generation of artificial training examples, known as virtual examples, have been proposed for Support Vector Machine (SVM) (Baird, 1990; Decoste & Schoelkopf, 2002; Poggio & Vetter, 1992). However, this is different from the proposed methods in three perspectives: (a) The virtual examples are to generate virtual training data, whereas the proposed method is to generate pseudo test data; the scopes are different; (b) The

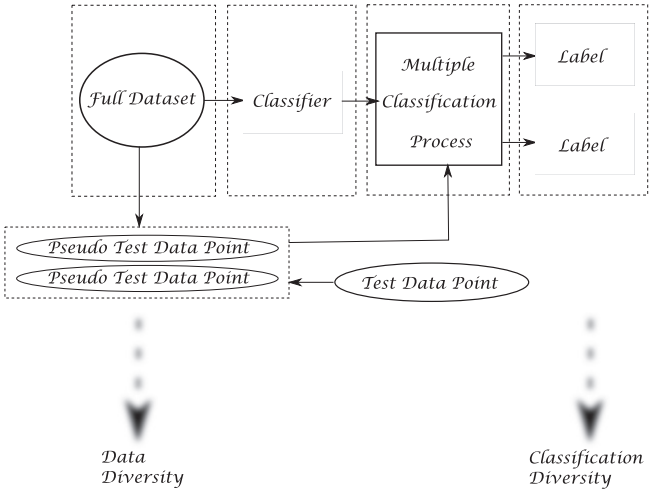


**Fig. 4.** Proposed scheme for the single classifier-based multiple classifier system: For a test sample, multiple pseudo test data points are generated. This results in multiple classifications with a single classifier. Classification diversity is implied by the fact that pseudo test data points are generated using different training data points.

virtual examples are generated so that the learning machine will extract the invariances from the artificially enlarged training data (Baird, 1990; Poggio & Vetter, 1992), whereas our proposed method is to generate pseudo test data points so that the learning machine can combine them and enhance accuracy; the purposes are different; and (c) Virtual examples are designed specifically for SVM, whereas the proposed method is suitable for all kinds of classifiers; the scales are different.

Also note that there are some fundamental differences between the MCS and the SMCS. In the MCS, we benefit from the fact that each classifier has a different perception of how a test sample should be classified. Because the decision boundary made by each classifier is different, there is diversity among decision boundaries drawn by different classifiers. Given that classifiers make different errors on different test samples, diversity can actually help improve classification accuracy. So, in the MCS, one of the core issues is to generate, select, and combine multiple classifiers, such that the combined decision boundary is better than any existing single boundary. In the SMCS, we not only try to find a better decision boundary, but one with the potential to be close to the oracle and not constrained by an existing classifier boundary and the number of classifiers. In designing the SMCS, we acknowledged the fact that a decision boundary drawn by a classifier might never be optimal; so, instead of refining several existing decision boundaries by combining them, we are trying to explore and make use of information in the neighborhood of a single decision boundary. In this way, we are looking for diversity that is already present in the neighborhood, rather than trying to benefit from diversity embedded in different classifiers. Consequently, diversity is extracted not from diverse decision boundaries, but from diverse pseudo test data points. The core issue is then to adequately generate, select, and combine multiple pseudo test data points for a test sample, rather than generating, selecting, and combining multiple classifiers.

We focus on three main questions in this paper:

(1) Can we extract diversity from a dataset directly without training multiple classifiers?
(2) Can multiple classification without multiple classifiers enhance classification accuracy?
(3) How can we compare SMCS performance with MCS performance?

In this paper, we strive to answer these questions by establishing a framework and testing its validity with a number of experiments and comparisons. The paper is organized as follows: we discuss a general framework on multiple classification in Section 2, and describe our SMCS approach in Section 3. The experimental protocols and results are presented in Section 4, followed by a discussion and our conclusions.

## 2. A general framework for multiple classification

The main focus of the problem is diversity creation during the EoC generation. The traditional approach is to feed diverse training data subsets to classifiers during the classifier training stage. Once the classifiers have all been trained using different data subsets, they will likely produce diverse classification outputs on a test data point.

Our concern is whether or not diversity must be created through repetitive classifier training. In order to clarify the problem, we first review the traditional diversity generation approach in EoC, and then discuss possible alternatives.

### 2.1. Review of the diversity generation process

The traditional approach to generating diversity is to inject different training data subsets into different classifiers, and then collect diverse classification outputs from these classifiers. We review the mechanism for doing this in the section below.

#### 2.1.1. Diversity in data subsets

Given $N$ training data points $\mathbf{X} = \{x_1, x_2, \cdots, x_N\}$, the traditional ensemble generation methods, such as Bagging or Boosting, gener-

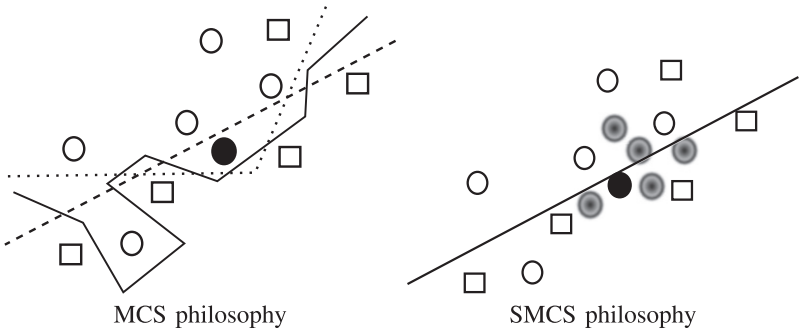


**Fig. 5.** The multiple classification philosophies of the MCS and the SMCS compared. Empty circles represent reference samples of class *A*, and empty rectangles represent reference samples of class *B*. The dark circle represents a test sample to be classified, gray circles represent pseudo test data points generated by the SMCS, and lines represent the decision boundaries drawn by classifiers: (a) the MCS relies on multiple classifiers to generate multiple decision boundaries, and so multiple classifications are generated to classify a test sample; and (b) the SMCS relies on a single classifier, and so there is only one decision boundary, but multiple pseudo test data points are generated to subsequently generate multiple classifications.

ate $K$ data subsets: $\{\mathbf{X_k}\}, 1 \leqslant k \leqslant K$, such that each data subset is different from the others:

$$\forall i,j, 1 \leqslant i,j \leqslant K, \mathbf{X_i}, \mathbf{X_j} \subset \mathbf{X} \quad (1)$$

$$\forall i,j, i \neq j, \mathbf{X_i} \not\subseteq \mathbf{X_j}, \mathbf{X_j} \not\subseteq \mathbf{X_i} \quad (2)$$

We should point out that, although we can say with confidence that a data subset $\mathbf{X_i}$ is different from a data subset $\mathbf{X_j}$ if their data samples are not 100% identical, we have no means to measure their diversity at the data sample level. Since that diversity should take into account the similarity among data samples as well as their class labels, the embedded complexity may not be trivial. As far as the authors know, there are no diversity measures for data samples in the literature.

#### 2.1.2. Classifier diversity

Using these $K$ data subsets to train $K$ classifiers, $\{C_k\}, 1 \leqslant k \leqslant K$, each classifier $C_k$ is a function of the corresponding data subset $\{\mathbf{X_k}\}$.

Again, for any two classifiers $C_i$ and $C_j$ trained with the same classification algorithm and the same parameters, because $\mathbf{X_i} \not\subseteq \mathbf{X_j}, \mathbf{X_j} \not\subseteq \mathbf{X_i}$, we have:

$$\forall i,j, i \neq j, C_i \neq C_j \quad (3)$$

Here, we point out that, although classifier $C_i$ and classifier $C_j$ are inherently different, it is difficult to measure their diversity without any test samples. Traditionally, the diversity between any two classifiers is measured by the difference in the classifications on all the test samples. Without the presence of test samples, measuring the diversity between classifiers seems a challenging task. Again, as far as we know, no method has ever been proposed in the literature for measuring the diversity of two classifiers without the use of test samples.

By contrast, with the test samples, we can measure classification diversity as a proxy to classifier diversity (Brown et al., 2005; Ko et al., 2009; Kuncheva & Whitaker, 2003; Kuncheva et al., 2002). In fact, these are diversity measures on a classification level, and not on a classifier level. We provide more details below.

#### 2.1.3. Classification diversity

To a given test data point $\tilde{x}_t$, a class label $\tilde{y}_{t,k}$ is assigned by each classifier $C_k$:

$$C_k : \tilde{x}_t \mapsto \tilde{y}_{t,k} \quad (4)$$

Based on the classification labels $\tilde{y}_{t,k}$ assigned by the different classifiers $C_k$, $1 \leqslant k \leqslant K$, we can measure their diversity. Almost all the diversity measures proposed in the literature fall into this category; for example, the disagreement measure (Ho, 1998), the double-fault (Giacinto & Roli, 2001), Kohavi-Wolpert variance (Kohavi & Wolpert, 1996), the interrater agreement (Fleiss, Levin, & Paik, 2003), the entropy measure (Kuncheva & Whitaker, 2003), the difficulty measure (Hansen & Salamon, 1990), generalized diversity (Partridge & Krzanowski, 1997), coincident failure diversity (Partridge & Krzanowski, 1997), $Q$-statistics (Afifi & Azen, 1979), and the correlation coefficient (Kuncheva & Whitaker, 2003).

#### 2.1.4. Alternative diversity generation

The difficulty is to design a system in such a way that diversity can be properly generated through any test data point. First, we recognize that diversity cannot exist in a single classification. Diversity is created because there are multiple classifications, and each classification is different from the others. Understanding the need for multiple diverse classifications, the issue becomes one of finding a proper mechanism to generate them.

Eq. (4) clearly indicates that a classification output (class label) is the product of a classifier and a test sample. In order to obtain multiple classifications, we have only two options: (a) multiple classifiers; or (b) pseudo test data points that originate from the same test sample.

If we examine the generation of diversity through classifiers, we find that multiple classifications are the result of multiple classifiers. In this way, diversity is injected into classifiers, and multiple classifiers are created for this purpose. Similarly, we can reason that if diversity is to be injected through the use of a test sample, then multiple pseudo test data points must be generated. It is critical, therefore, to generate multiple pseudo test data points based on a single test sample.

Our objective is to propose a mechanism to generate multiple pseudo test data points, and, as far as we know, no one in the EoC community has ever tried to do this. If we can achieve multiple diverse classifications without training multiple classifiers, then this will constitute another research opportunity for the MCS. Our proposal for such a mechanism is given in the following section.

### 2.2. A Framework for pseudo data point generation

From the above discussion, we can see that there is another way to generate multiple diverse classifications, which is through pseudo data point generation. Below, we discuss possible mechanisms for generating pseudo data points.

#### 2.2.1. Requirements for generated pseudo data points

Given a test sample $\tilde{x}_t$ and $K$ training data subsets, we need to generate $K$ pseudo data points, $\{\hat{x}_{k,t}\}, 1 \leqslant k \leqslant K$. Intuitively, it does not make much sense to generate pseudo test points that are identical to the original test sample, or points that are all the same. We can foresee that identical pseudo test points would yield identical classification outputs (class labels) for a test sample, which means that such multiple classifications would not enhance classification accuracy.

As a result, we impose two requirements for the pseudo test data points generated:

(1) Generated pseudo test data points should be different from the original test sample.

$$\hat{x}_{k,t} \neq \tilde{x}_t, 1 \leqslant k \leqslant K \quad (5)$$

(2) No two generated pseudo data points should be the same.

$$\forall i \neq j, \hat{x}_{i,t} \neq \hat{x}_{j,t}, 1 \leqslant i,j \leqslant K \quad (6)$$

Based on these two requirements, we must find diversity and then inject this diversity into the generated pseudo data points.

#### 2.2.2. Diversity injection for pseudo data point generation

Recall that in the section above we concluded that all diversity embedded in diverse classifiers is derived from data subset diversity. When we use different data subsets to train different classifiers, these trained classifiers will yield diverse classifications, and so we wonder whether or not it is possible to use data subsets to generate diverse pseudo data points in the same way that we generate diverse classifiers.

Assuming that this scheme is feasible, we can state that each pseudo test data point $\hat{x}_{k,t}$ is a function of the corresponding data subset $\{\mathbf{X_k}\}$ and the original test data point $\tilde{x}_t$:

$$\hat{x}_{k,t} = f(\tilde{x}_t, \{\mathbf{X_k}\}) \quad (7)$$

Consequently, the $K$ pseudo test data points for each original test sample will result in $K$ classifications with a single classifier. Multiple classification results can then be combined to boost classification accuracy. Fig. 4 illustrates a general scheme for generating diverse classifications without multiple classifiers.

The problem is therefore to design a pseudo test data point generation function $f$ that can generate adequate pseudo test points. Moreover, the combination of multiple classifications based on these pseudo test points should also enhance classification accuracy. To meet this challenge, we have designed a Single-Classifier-based Multiple Classification Scheme (SMCS), which we outline in the next section.

## 3. Proposed method

Given a training dataset $\mathbf{X}$, we first divide the training samples into $N$ reference samples $\mathbf{X_r} = \{x_1, x_2, \ldots, x_N\}$, and $M$ evaluation samples $\check{\mathbf{X}}_\mathbf{e} = \{\check{x}_1, \check{x}_2, \ldots, \check{x}_M\}$, a single classifier $C_\mathbf{X}$ trained by all the available training samples $\mathbf{X}$, and a test data point $\tilde{x}_t$. The mechanism involves the creation of $K$ pseudo test data points, $\hat{\mathbf{X}}_\mathbf{t} = \hat{x}_{1,t}, \hat{x}_{2,t}, \ldots, \hat{x}_{K,t}$, which would result in $K$ corresponding classification outputs $\hat{y}_{1,t}, \hat{y}_{2,t}, \ldots, \hat{y}_{K,t}$ after being classified by $C_\mathbf{X}$.

The purpose of this mechanism is to generate $\hat{\mathbf{X}}_\mathbf{t}$, such that the combination of classification outputs on these $K$ pseudo test data points $\hat{y}_t$ will be as close to the true class label $y_t$ as possible. Note that:

$$\hat{y}_t = g(\hat{y}_{1,t}, \hat{y}_{2,t}, \cdots, \hat{y}_{K,t}) \tag{8}$$

where $g(\cdot)$ is the classification combination function, such as majority voting.

Here, the main problem is to design a stable mechanism that generates pseudo test data points that improve the overall classification result. We decompose this problem into two sub problems, expressed as the following two questions:

(1) What is the function $f(\cdot)$ used to generate pseudo data points, given a test sample and several reference samples?
(2) How do we decide which reference samples to use to generate pseudo data points, given a test sample?

We address these two sub problems in the sections below and describe them in more detail.

### 3.1. Define a function to generate pseudo test data points

There are a number of ways to solve the first component of the problem, which is to decide how to generate a pseudo test data point given a test sample and one or more reference samples.

A pseudo test data point can be generated as a combination of a test sample and a reference sample, or as a combination of a test sample and several reference samples. It can be generated in a deterministic way, or with some random factors. To gain some insight into the properties of the SMCS, we start with a simple and deterministic function to generate pseudo test data points. In our method, each pseudo test point is based on an original test sample and a single reference sample:

$$\hat{x}_{i,t} = f(x_i, \tilde{x}_t) \tag{9}$$

where $\hat{x}_{i,t}$ indicates a generated pseudo test data point, $\tilde{x}_t$ is the original test sample, and $x_i$ is a reference sample.

For example, if each data point has $L$ feature dimensions, then the feature $l$ of the generated pseudo test data point $\hat{x}_{i,t}$ will simply be a weighted average of the same feature of the test sample $\tilde{x}_t$ and that of the reference sample $x_i$:

$$\hat{x}_{i,t,l} = \alpha x_{i,l} + (1-\alpha)\tilde{x}_{t,l}, \quad 1 \leqslant l \leqslant L, 0 \leqslant \alpha \leqslant 1 \tag{10}$$

where $\hat{x}_{i,t,l}$ indicates the value of the feature $l$ of the generated pseudo test data point $\hat{x}_{i,t}$, $x_{i,l}$ indicates the value of the feature $l$ of the reference sample $x_i$, and $\tilde{x}_{t,l}$ indicates the value of the feature $l$ of the test sample $\tilde{x}_t$. Also note that $\alpha$ controls the noise and diversity present in pseudo test data points: the larger it is, the greater the diversity and the noise.

### 3.2. Select reference samples to generate pseudo test data points

Not every generated pseudo data point will be adequate for classification. The fitness of a pseudo data point will largely depend on the "chemistry" between the test sample $\tilde{x}_t$ and the reference sample $x_i$.

In order to evaluate the fitness of each reference sample $x_i$ for a test sample $\tilde{x}_t$ in an attempt to generate adequate pseudo data points, we propose a three-step scheme:

(1) Identify valid [evaluation sample–reference sample] pairs- Remember that we divide training samples into evaluation samples and reference samples. We will use these $M$ evaluation samples to determine the fitness of a reference sample. Each evaluation sample will generate a pseudo data point using the reference sample, and then the pseudo data point will be classified. If the classification of this pseudo data point has the same label as the evaluation sample, then this [evaluation sample–reference sample] pair is regarded as valid; otherwise, it is regarded as invalid.
(2) Assign weight to reference samples For a given test sample, we find the $m$ nearest evaluation samples. Then, every reference sample is assigned a weight based on its validity with respect to these $m$ evaluation samples, which is obtained as the sum of the $m$ [evaluation sample–reference sample] pairs.
(3) Select reference samples and generate pseudo test data points We set a threshold for the reference samples, and select only those with weights higher than that threshold for pseudo test data point generation.

Here we notice two things. First, each function can be manipulated independently and so is subject to optimization. This modular approach gives our proposed method a great deal of flexibility, and it can be adapted to various pattern recognition problems. Second, the step of identifying valid [evaluation sample–reference sample] pairs needs only to be performed once for all the test samples, whereas the other two steps need to be carried out for each individual test sample. Given that the first step is more time consuming, and the second and the third steps are fairly straightforward and less time consuming, the overall process can be implemented in the real world without incurring enormous cost. We provide more details below.

#### 3.2.1. Identify valid [evaluation sample–reference sample] pairs

The first step is to evaluate the fitness of each reference sample by using several evaluation samples from a evaluation dataset, $\check{\mathbf{X}}_\mathbf{e} = \{\check{x}_1, \check{x}_2, \cdots, \check{x}_M\}$. For a reference sample $x_i$ to be evaluated, an evaluation sample $\check{x}_k$ generates a pseudo data point $\hat{x}_{i,k}$ using this reference sample. Then, the generated pseudo data point is classified by a classifier:

$$\hat{x}_{i,k} \mapsto \hat{y}_{i,k} \tag{11}$$

Since we already know the class label $y_k$ of each evaluation sample $\check{x}_k$, we can determine whether or not the classification of this pseudo data point is correct, meaning that it has the same class label as that of the evaluation sample. We repeat the same process between all the reference and evaluation samples, and then define a validity measure $v_{i,k}$ for each [evaluation sample $\check{x}_k$–training data point $x_i$] pair, $1 \leqslant k \leqslant M$, $1 \leqslant i \leqslant N$, and set the validity to 1 for correct classification and to 0 for incorrect classification:

$$v_{i,k} = 1, \quad \text{if } \hat{y}_{i,k} = y_k \tag{12}$$

$$v_{i,k} = 0, \quad \text{otherwise} \tag{13}$$

Fig. 6 shows the process of identifying valid [evaluation sample–reference sample] pairs. The validity measures are then used to evaluate the fitness of each training data point $x_i$, as described in the next section.

*3.2.2. Assign weight to reference samples*

The validity measure $v_{i,k}$ for each [evaluation sample $\check{x}_k$–reference sample $x_i$] pair tells us whether or not a reference sample $x_i$ is fit to generate a pseudo test point with an evaluation sample $\check{x}_k$, but it does not tell us whether or not a reference sample $x_i$ is fit to generate a pseudo test point with a test sample $\tilde{x}_t$.

In order to decide whether or not we should use a reference sample $x_i$ to generate a pseudo test point with a test sample $\tilde{x}_t$, first we try to find the $m$ nearest evaluation samples from the test sample $\tilde{x}_t$. The idea behind this action is that these evaluation samples can be seen as proxies of the test sample $\tilde{x}_t$. If they all qualify as correct classifications with the use of the reference sample $x_i$ to generate pseudo test points, then the test sample $\tilde{x}_t$ can use this reference sample $x_i$ to generate pseudo test points as well.

Given a test sample $\tilde{x}_t$, let us consider the nearest $m$ evaluation samples to be trustworthy for this test sample, noting that $m \ll M$. We then use these $m$ evaluation data points to evaluate the fitness of reference samples for the test sample $\tilde{x}_t$. The weight of a reference sample $x_i$ is assigned as follows:

$$w_i = \sum_{k=1}^{m} \delta_{i,k} v_{i,k} \tag{14}$$

where $v_{i,k}$ is a validity measure $v_{i,k}$ for the [evaluation sample $\check{x}_k$–reference sample $x_i$] pair, and $\delta_{i,k}$ is a weighting adjustment based on distance or other factors.

Fig. 7 demonstrates a general scheme for assigning weight to reference samples through the aggregation of multiple validity measures between a reference sample and evaluation samples. In this paper, we define the weighting adjustment $\delta_{i,\,k}$ as:

$$\delta_{i,k} = \frac{d(\check{x}_k, \tilde{x}_t) + d(\check{x}_k, x_i)}{d(\tilde{x}_t, x_i)} \tag{15}$$

where $d(\cdot)$ indicates a Euclidean distance function, $\check{x}_k$ is an evaluation sample, $x_i$ is a reference sample, and $\tilde{x}_t$ is a test sample.

So, $d(\check{x}_k, \tilde{x}_t)$ is the distance between the evaluation sample $\check{x}_k$ and the test sample $\tilde{x}_t$, $d(\check{x}_k, x_i)$ is the distance between the evaluation sample $\check{x}_k$ and the reference sample $x_i$, and $d(\tilde{x}_t, x_i)$ is the distance between the reference sample $x_i$ and the test sample $\tilde{x}_t$.

In our weighting adjustment $\delta_{i,k}$, the weight $w_i$ increases with $\frac{d(\check{x}_k, x_i)}{d(\tilde{x}_t, x_i)}$, because knowing that the reference sample produces correct pseudo data points for an evaluation sample, the distance between the evaluation sample $\check{x}_k$ and the reference sample $x_i$ signals the robustness of the reference sample; whereas the distance between the test sample $\tilde{x}_t$ and the reference sample $x_i$ scales down this robustness measure. The weight $w_i$ also increases with $\frac{d(\check{x}_k, x_i)}{d(\tilde{x}_t, x_i)}$, because the ratio of the distance $d(\check{x}_k, x_i)$ to the distance $d(\tilde{x}_t, x_i)$ represents the validity to approximate the test sample $\tilde{x}_t$ using the evaluation sample $\check{x}_k$.

Note that other weighting mechanisms may be suitable as well. This is simply the one that we chose to implement.

*3.2.3. Select reference samples and generate pseudo test data points*

Given a test sample $\tilde{x}_t$, once all reference samples are evaluated using the nearest $m$ evaluation samples from that test sample, we can proceed to select adequate reference samples for the purpose of pseudo test data point generation.

Again, we can only evaluate the nearest $n$ reference samples for the test sample. We also define a threshold $\theta$. Therefore, the selection criterion for reference samples is:

$$\text{if } w_i \geqslant \theta \quad s_i = 1 \tag{16}$$

$$\text{else} \quad s_i = 0 \tag{17}$$

where $s_i$ is the selection decision on reference sample $x_i$. The threshold $\theta$ is defined as:

$$\theta = \rho \max\{w_i\}, \quad 0 < \rho \leqslant 1 \tag{18}$$

Fig. 8 shows the process of reference sample selection that we use to generate pseudo data points for a test data point. Once multiple diverse pseudo test data points are generated, we can use them to feed a classifier to produce multiple classifications for a single test sample. However, these multiple classifications need to be combined in order to produce a final class label for the test sample. We describe the process below.

### 3.3. Combine multiple classification outputs

Supposing that $I$ reference samples $x_i, 1 \leqslant i \leqslant I$ are selected for a test sample $\tilde{x}_t$, corresponding pseudo test data points can be generated:

$$\hat{x}_{i,t} = f(x_i, \tilde{x}_t) \tag{19}$$

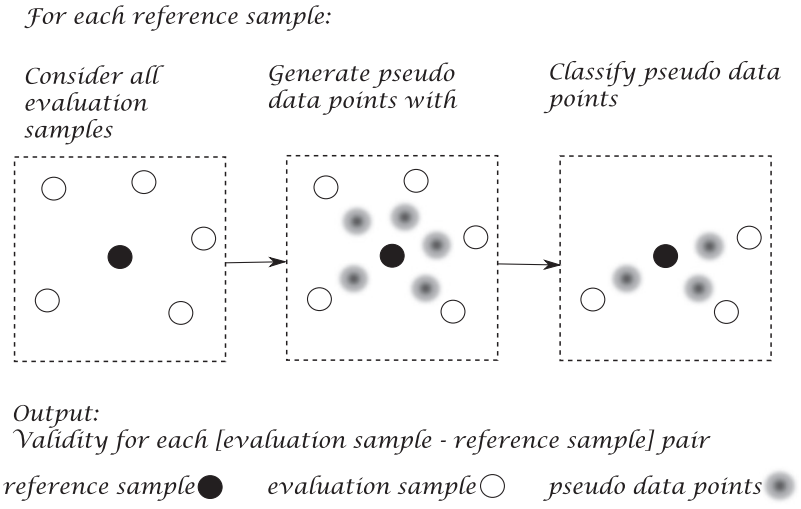


**Fig. 6.** Each evaluation sample generates a pseudo data point using a reference sample. The classification result of this pseudo data point provides an indication of the fitness of the [evaluation sample–test sample] pair. Solid circles represent reference samples, white circles represent evaluation samples, and gray circles represent generated pseudo data points.

**Algorithm 1.** Pseudo-code of proposed Single Classifier-based Multiple Classification System.

Define reference dataset $\mathbf{X_r} = \{x_1, x_2, \ldots, x_N\}$, evaluation dataset $\check{\mathbf{X}}_\mathbf{e} = \{\check{x}_1, \check{x}_2, \ldots, \check{x}_M\}$ and a test sample $\tilde{x}_t$
Define pseudo data point generation function $f$: $f : \tilde{x}_t \mapsto \hat{x}$
Train a single classifier $C_\mathbf{X}$ using all reference samples and evaluation samples

Identify valid [evaluation sample–reference sample] pairs:
**for** all evaluation samples $\check{x}_k, k = 1, \ldots, M$ **do**
  **for** all reference samples $x_i, i = 1, \ldots, N$ **do**
    Generate pseudo data point $\hat{x}_{i,k}$ with function $f$ and reference sample $x_i$ for evaluation sample $\check{x}_k$
    Classify pseudo data point $\hat{x}_{i,k}$ with classifier $C_\mathbf{X}$:
    **if** Classification output $\hat{y}_{i,k}$ of pseudo data point $\hat{x}_{i,k}$ equals the class label $\check{y}_k$ of evaluation sample $\check{x}_k$ **then**
      Set validity $v_{i,k}$ of [evaluation sample $\check{x}_k$–reference sample $x_i$] pair to be 1
    **else**
      Set validity $v_{i,k}$ of [evaluation sample $\check{x}_k$–reference sample $x_i$] pair to be 0
    **end if**
  **end for**
**end for**

Assign weight to reference samples for test sample $\tilde{x}_t$:
**for** all reference samples $x_i$, $i = 1, \ldots, N$ **do**
  Initialize $\omega_i$ of reference sample $x_i$ to be 0
  **for** Nearest $m$ evaluation samples from test sample $\tilde{x}_t, \check{x}_k, k = 1, \ldots, m$ **do**
    Evaluate the weight of reference samples $x_i$ for test sample $\tilde{x}_t$:
    **if** validity $v_{i,k}$ of [evaluation sample $\check{x}_k$–reference sample $x_i$] pair equals to 1 **then**
      Increase the weight $\omega_i$ of reference samples $x_i$: $\omega_i = \omega_i + \delta_{i,k} v_{i,k}$
    **end if**
  **end for**
**end for**

Select reference samples and generate pseudo test data points:
**for** Nearest $n$ reference samples from test sample $\tilde{x}_t$, $x_i$, $i = 1, \ldots, n$ **do**
  **if** the weight $\omega_i$ of reference samples $x_i$ is greater than the threshold $\theta$ **then**
    Genarate a pseudo test data point $\hat{x}_{i,t}$ for test sample $\tilde{x}_t$ based on reference sample $x_i$:

$$\hat{x}_{i,t} = f(\tilde{x}_t, x_i)$$

    Classify pseudo data point $\hat{x}_i$ with classifier $C_\mathbf{X}$, and store the classification output $\hat{y}_i$
  **end if**
**end for**

Apply a fusion function $g$ to combine multiple classification outputs: $\hat{y}_t = g(\{\hat{y}_{i,t}\})$
Return combined classification output, $\hat{y}_t$, as final result.

By applying a single classifier $C_\mathbf{X}$ that is trained with all the available reference samples $\mathbf{X}$, multiple classification outputs can be obtained:

$$\hat{x}_{i,t} \mapsto \hat{y}_{i,t} \tag{20}$$

Once we have multiple classification outputs, a fusion function $g$ is implemented to combine them:

$$\hat{y}_t = g(\hat{y}_{1,t}, \hat{y}_{2,t}, \cdots, \hat{y}_{I,t}) \tag{21}$$

As a result, we obtain the final class label output for the test sample concerned. Below, we provide the pseudo code for our proposed method to better illustrate the methodology.

## 4. Experiments

In order to verify the validity of the proposed SMCS, to understand effects of neighborhood sizes $m$ and $n$ and reference sample selection, and to measure the performance of the SMCS, we carried out a number of experiments on different datasets extracted from the UCI Machine Learning Repertoire. The experiments were conducted in MATLAB using PRTools (Duin et al., 2007).

### 4.1. Experimental protocol

We tested 10 UCI datasets with predetermined SMCS parameters. These datasets were selected for testing on the basis that they contain only numerical features. These datasets are the following: Breast Tissue (9 features), Bupa (6 features), Glass (9 features), Image Segmentation (19 features), Iris (4 features), Parkinsons (22 features), Vowel (11 features), Wine (13 features), Wisconsin Breast Cancer (wdbc, 30 features), and Yeast (8 features).

Six different classification methods were tested: Linear Discriminant Classifier (LDC), Quadratic Discriminant Classifier (QDC), K-Nearest-Neighbors (KNN) with $K = 15$, Parzen Windows (PW), Multi-Layer Perceptron of 3 hidden layers with Backpropagation (MLP), and Decision Trees (Tree). A total of $10 \times 6 = 60$ classifier-dataset combinations were tested.

We split each dataset into two subsets: a training dataset to be used for training classifiers, and a test dataset to be used for accuracy assessment. The training datasets are further split into a reference dataset, to be used for pseudo data point generation, and an evaluation dataset, to be used for reference sample evaluation purposes in the SMCS. In general, an original dataset, whether reference, evaluation, or test, is split into equal parts. The Image Segmentation dataset is an exception, as the training dataset is predefined in the UCI Machine Learning Repertoire (Table 1).

For all classifier-dataset combinations, three different multiple classification schemes were applied: the proposed Single Classifier-based Multiple Classification Scheme (SMCS), and two traditional Multiple Classifier Systems (MCS) (Bagging and Boosting).

For experiments using the SMCS, only one classifier was trained for each classifier-dataset combination, and so this classifier was used on all the available samples in the training dataset. For pseudo data point generation, we implemented a weighted combination of a reference sample and a test sample, as described in Eq. (16), with equal weights for both, so $\alpha = \frac{1}{2}$. We also tested different ranges of parameters for $m$, $n$, and $\rho$, in an effort to gain more insight: the value of $m$ was set equal to $n$, and we tested $m = n = 9$, $m = n = 11, \cdots$, and $m = n = 21$, for a total of 7 different values of $m$ and $n$. For the threshold setting $\rho$, we tested $\rho = 0.95$, $\rho = 0.9$, $\rho = 0.85$, and $\rho = 0.80$, for a total of 4 values of $\rho$. Combining $m$ and $n$, $7 \times 4 = 28$ different parameter sets were tested.

Note that when $m$ and $n$ increase, we can include more reference samples and generate more diversity. The same applies when we decrease the value of $\rho$. However, diversity is usually accompanied by noise, and so there is a tradeoff between diversity and noise.

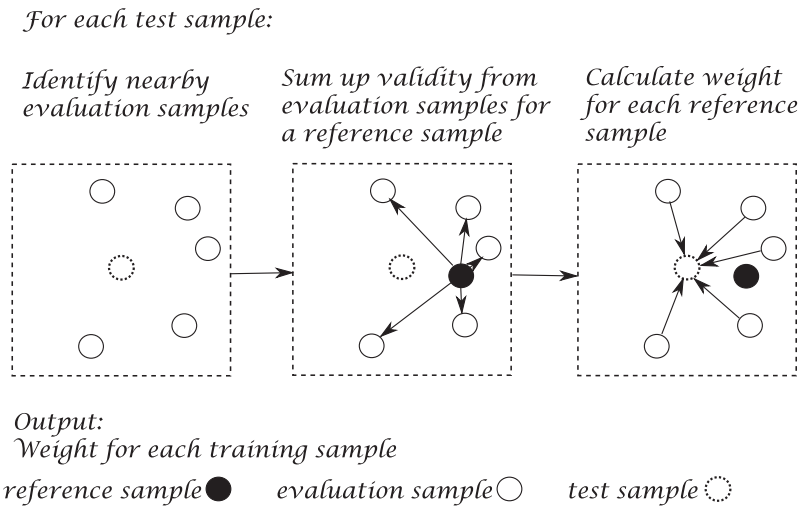


**Fig. 7.** For each test sample, each reference sample is evaluated indirectly by aggregating the validity of [evaluation sample–reference sample] pairs from the nearby evaluation samples. Now, the weighting of reference samples can be adjusted by distances. Solid circles represent reference samples, white circles with a dotted contour represent test samples, and white circles with a solid contour represents evaluation samples.

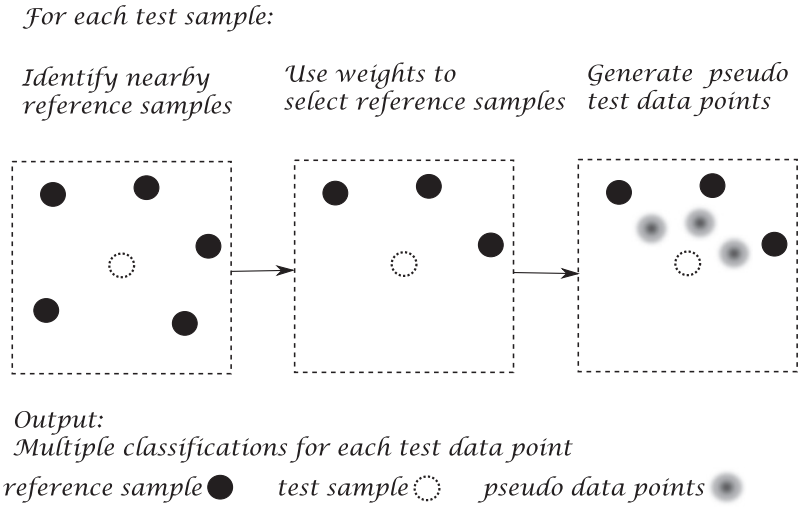


**Fig. 8.** Adequate reference samples are then selected by the weights that each test sample assigns to them. The selected reference samples then generate pseudo test data points for the original test data point. Solid circles represent reference samples, white circles with a dotted contour represent a test sample, and gray circles represent generated pseudo test data points.

In order to compare the proposed SMCS with traditional MCS, Bagging and Boosting (Kuncheva et al., 2002; Fumera et al., 2008) in MCS were also implemented on the same datasets.

For Bagging and Boosting, we tested classifier pools with of size 3, 5, ..., and 15, constructing seven ensembles per scheme per classifier-dataset combination. Every classifier in a classifier pool used 67% data points in the training dataset for classifier training, and the experiments were repeated 30 times, and we show the averages of the MCS performances for the two schemes. For the proposed SMCS and Bagging and Boosting in MCS, all the multiple classifications generated were combined using majority voting (Ruta & Gabrys, 2005) to arrive at a final decision on the class label of the test samples.

### 4.2. Experimental results

#### 4.2.1. Single Classifier-based Multiple Classification results

Our experimental results suggest that the proposed SMCS works to some extent, with an average improvement of 16.31% over the 60 dataset-classification algorithm combinations. The injected diversity seems to enhance the accuracy of the recognition rates in most cases, and generally the noise that is inherent in diversity does not degrade the classification results. Table 2 provides a summary of SMCS error rates on various dataset-classifier combinations. The only apparent exceptions are the Iris data with MLP, where the error rate increases from 4.00% to 4.50%, and the Iris data with the QDC, where the error rate increases from 4.00% to 4.43%, although, in fact, the Wine data with the QDC is another exception, where the error rate increases from 0.00% to 0.61%. The increase in the error rate in these individual cases may be due to the small data size, since the Iris has only 17 samples per class and the Wine data has 20 samples per class for the reference and evaluation datasets.

We note that the improvement achieved with the SMCS also depends on the classification methods of the trained classifiers.

For example, in general, the LDC and Decision Tree classifiers almost always benefit from the SMCS, even on small datasets. This is especially noticeable with the Decision Tree classifiers, where the improvement on the 10 datasets ranges from 13.89% to 70.78%. In contrast, Parzen Windows, MLP, and KNN only demon-

**Table 1**
The datasets used in our experiments.

| Datasets | Number of classes | Dimension | Reference dataset size | Evaluation dataset size | Test dataset size | Dataset size per class |
|---|---|---|---|---|---|---|
| Breast-tissue | 6 | 9 | 35 | 36 | 35 | 6 |
| Bupa-liver | 2 | 6 | 115 | 115 | 115 | 58 |
| Glass | 6 | 9 | 72 | 71 | 71 | 12 |
| Image segmentation | 7 | 19 | 210 | 1050 | 1050 | 30/150 |
| Iris | 3 | 4 | 50 | 50 | 50 | 17 |
| Parkinsons | 2 | 22 | 65 | 65 | 65 | 32 |
| Vowel | 11 | 11 | 330 | 330 | 330 | 30 |
| Wdbc | 2 | 30 | 190 | 189 | 190 | 95 |
| Wine | 3 | 13 | 59 | 60 | 59 | 20 |
| Yeast | 10 | 8 | 494 | 495 | 495 | 49 |

strate accuracy improvement on larger datasets, with more than 40 samples per class, such as the Bupa (liver), Image Segmentation, wdbc, and Yeast data.

It seems that the efficiency of the proposed SMCS largely depends on the number of samples per class in the training dataset, i.e. the number of samples per class in the evaluation dataset and in the reference dataset.

Furthermore, we need to mention that the SMCS performances are evaluated without parameter optimization. We show the average performance of all the parameters tested. This means that, by fine-tuning parameters such as $m$, $n$, and $\rho$, we can expect better performances. The key issue is that parameter optimization may require more samples, which might not be appropriate for small datasets.

#### 4.2.2. Multiple Classifier System results

We also applied the traditional MCS schemes Bagging and Boosting, in an attempt to compare the results with those of the SMCS. For Bagging and Boosting, we show the average ensemble accuracy of all the ensembles constructed with different classifier pool sizes (3–15). We show the results of average combined classifiers, i.e. ensemble error rate after applying a fusion function for classifier combination and denoted "average bagging" or "average boosting," and that of average constructed classifiers, i.e. simply the average error rate of constructed classifiers without applying any fusion function and denoted "classifier pool". We also list the classifier error rate using all the training samples used for training as the baseline.

Table 3 shows average error rates of constructed MCS using for Bagging, and Table 4 shows those for Boosting.

#### 4.2.3. Improvement rate comparison

In an effort to compare the performance of the proposed SMCS with Bagging and Boosting, we calculated the proportion of decreased error rates on different datasets with different classifiers over the baseline. The results are contained in Tables 2–4.

**Table 2**
The error rates of a Single Classifier-based Multiple Classification Scheme with parameters $n = m = 9$–21 and $\rho = 0.8$–0.95. We show the error rates of a single classifier (denoted "single classifier") as the baseline, and those of an average SMCS without any parameter selection (denoted "average SMCS").

| Dataset | Method | LDC (%) | QDC (%) | KNN (%) | PW (%) | MLP (%) | Tree (%) |
|---|---|---|---|---|---|---|---|
| Breast-tissue | Single classifier | 31.43 | 40.00 | 48.57 | 51.43 | 54.29 | 40.00 |
| Breast-tissue | Average SMCS | 24.29 | 29.49 | 48.57 | 48.57 | 43.67 | 32.04 |
| Breast-tissue | Error change | −22.73 | −26.28 | 0.00 | −5.56 | −19.55 | −19.90 |
| Bupa-liver | Single classifier | 33.04 | 41.74 | 32.17 | 46.09 | 33.91 | 40.00 |
| Bupa-liver | Average SMCS | 33.51 | 33.66 | 32.02 | 45.22 | 30.99 | 31.43 |
| Bupa-liver | Error change | 1.41 | −19.35 | −0.48 | −1.89 | −8.61 | −21.43 |
| Glass | Single classifier | 30.99 | 36.62 | 33.80 | 40.85 | 61.97 | 25.35 |
| Glass | Average SMCS | 22.48 | 35.97 | 31.09 | 40.85 | 61.97 | 21.83 |
| Glass | Error change | −27.44 | −1.79 | −8.04 | 0.00 | 0.00 | −13.89 |
| Image segmentation | Single classifier | 7.33 | 8.00 | 10.86 | 6.19 | 39.43 | 11.62 |
| image segmentation | Average SMCS | 5.68 | 6.36 | 10.16 | 4.66 | 34.34 | 6.65 |
| Image segmentation | error change | −22.59 | −20.49 | −6.42 | −24.73 | −12.90 | −42.80 |
| Iris | Single classifier | 8.00 | 4.00 | 10.00 | 8.00 | 4.00 | 24.00 |
| Iris | Average SMCS | 6.07 | 4.43 | 5.07 | 6.00 | 4.50 | 15.43 |
| Iris | Error change | −24.11 | 10.71 | −49.29 | −25.00 | 12.50 | −35.71 |
| Parkinsons | Single classifier | 15.38 | 15.38 | 20.00 | 16.92 | 12.31 | 13.85 |
| Parkinsons | Average SMCS | 13.08 | 13.52 | 20.00 | 16.92 | 11.37 | 10.93 |
| Parkinsons | Error change | −15.00 | −12.14 | 0.00 | 0.00 | −7.59 | −21.03 |
| Vowel | Single classifier | 39.09 | 11.21 | 29.09 | 3.03 | 84.24 | 30.61 |
| Vowel | Average SMCS | 33.17 | 7.90 | 24.10 | 3.16 | 83.64 | 19.32 |
| Vowel | Error change | −15.14 | −29.54 | −17.15 | 4.29 | −0.72 | −36.88 |
| Wdbc | Single classifier | 8.42 | 6.84 | 6.32 | 7.37 | 3.68 | 10.00 |
| Wdbc | Average SMCS | 5.62 | 5.30 | 6.05 | 6.11 | 2.18 | 5.11 |
| Wdbc | Error change | −33.26 | −22.53 | −4.17 | −17.09 | −40.82 | −48.87 |
| Wine | Single classifier | 1.69 | 0.00 | 35.59 | 30.51 | 37.29 | 18.64 |
| Wine | Average SMCS | 0.12 | 0.61 | 35.59 | 30.51 | 37.35 | 5.45 |
| Wine | Error change | −92.86 | *Inf* | 0.00 | 0.00 | 0.16 | −70.78 |
| Yeast | Single classifier | 41.01 | 100.00 | 42.42 | 42.22 | 48.89 | 52.53 |
| Yeast | Average SMCS | 37.86 | 100.00 | 38.97 | 38.89 | 47.58 | 42.76 |
| Yeast | Error change | −7.67 | 0.00 | −8.15 | −7.89 | −2.67 | −18.60 |

**Table 3**
The error rates of Bagging on the experimental datasets. All the experiments were repeated 30 times. We show the error rates of a single classifier as the baseline (denoted "single classifier"), those of classifiers trained with only 67% of the samples (denoted "classifier pool"), and those of classifiers created with Bagging and combined by majority voting (denoted "average bagging").

| Dataset | Method | LDC (%) | QDC (%) | KNN (%) | PW (%) | MLP (%) | Tree (%) |
|---|---|---|---|---|---|---|---|
| Breast-tissue | Single classifier | 31.43 | 40.00 | 48.57 | 51.43 | 54.29 | 40.00 |
| Breast-tissue | Classifier pool | 54.46 | 52.69 | 53.36 | 53.36 | 53.27 | 53.24 |
| Breast-tissue | Average bagging | 44.57 | 44.84 | 44.63 | 44.78 | 44.61 | 44.16 |
| Breast-tissue | Error change | 41.81 | 12.10 | −8.11 | −12.93 | −17.82 | 10.40 |
| Bupa-liver | Single classifier | 33.04 | 41.74 | 32.17 | 46.09 | 33.91 | 40.00 |
| Bupa-liver | Classifier pool | 38.55 | 38.30 | 38.89 | 38.96 | 39.12 | 38.55 |
| Bupa-liver | Average bagging | 37.76 | 37.90 | 37.79 | 37.81 | 37.93 | 37.95 |
| Bupa-liver | Error change | 14.27 | −9.20 | 17.46 | −17.96 | 11.84 | −5.13 |
| Glass | Single classifier | 30.99 | 36.62 | 33.80 | 40.85 | 61.97 | 25.35 |
| Glass | Classifier pool | 46.75 | 46.29 | 46.89 | 45.33 | 46.07 | 46.75 |
| Glass | Average SMCS | 42.25 | 42.11 | 41.88 | 41.86 | 41.50 | 42.03 |
| Glass | Error change | 36.35 | 14.99 | 23.89 | 2.48 | −33.03 | 65.78 |
| Image segmentation | Single classifier | 7.33 | 8.00 | 10.86 | 6.19 | 39.43 | 11.62 |
| Image segmentation | Classifier pool | 13.49 | 15.32 | 14.60 | 14.78 | 14.15 | 15.10 |
| Image segmentation | Average bagging | 8.55 | 8.27 | 8.12 | 8.25 | 8.36 | 8.54 |
| Image segmentation | Error change | 16.64 | 3.37 | −25.23 | 33.28 | −78.80 | −26.51 |
| Iris | Single classifier | 8.00 | 4.00 | 10.00 | 8.00 | 4.00 | 24.00 |
| Iris | Classifier pool | 11.03 | 11.26 | 11.00 | 10.28 | 10.90 | 10.12 |
| Iris | Average bagging | 9.90 | 9.58 | 9.49 | 9.37 | 9.50 | 9.71 |
| Iris | Error change | 23.75 | 139.50 | −5.10 | 17.12 | 137.50 | −59.54 |
| Parkinsons | Single classifier | 15.38 | 15.38 | 20.00 | 16.92 | 12.31 | 13.85 |
| Parkinsons | Classifier pool | 17.64 | 17.49 | 17.42 | 17.95 | 17.43 | 17.81 |
| Parkinsons | Average bagging | 16.37 | 16.21 | 16.15 | 16.20 | 16.31 | 16.45 |
| Parkinsons | Error change | 6.41 | 5.37 | −19.25 | −4.27 | 32.52 | 18.81 |
| Vowel | Single classifier | 39.09 | 11.21 | 29.09 | 3.03 | 84.24 | 30.61 |
| Vowel | Classifier pool | 37.24 | 36.98 | 36.98 | 36.84 | 36.96 | 37.09 |
| Vowel | Average bagging | 29.01 | 29.11 | 29.29 | 29.17 | 29.04 | 28.98 |
| Vowel | Error change | −25.79 | 159.63 | 0.68 | 862.61 | −65.53 | −5.31 |
| Wdbc | Single classifier | 8.42 | 6.84 | 6.32 | 7.37 | 3.68 | 10.00 |
| Wdbc | Classifier pool | 7.06 | 7.01 | 7.16 | 7.05 | 7.41 | 7.13 |
| Wdbc | Average bagging | 6.74 | 6.68 | 6.74 | 6.70 | 6.70 | 6.74 |
| Wdbc | Error change | −19.96 | −2.37 | 6.72 | −9.07 | 81.86 | −32.60 |
| Wine | Single classifier | 1.69 | 0.00 | 35.59 | 30.51 | 37.29 | 18.64 |
| Wine | Classifier pool | 18.01 | 16.98 | 17.97 | 17.89 | 17.50 | 17.86 |
| Wine | Average bagging | 15.55 | 15.02 | 15.16 | 15.08 | 15.05 | 15.05 |
| Wine | Error change | 817.45 | *Inf* | −57.41 | −50.57 | −59.64 | −19.28 |
| Yeast | Single classifier | 41.01 | 100.00 | 42.42 | 42.22 | 48.89 | 52.53 |
| Yeast | Classifier pool | 59.20 | 59.10 | 58.68 | 59.23 | 58.71 | 58.81 |
| Yeast | Average bagging | 53.93 | 53.77 | 53.61 | 53.65 | 53.39 | 53.71 |
| Yeast | Error change | 31.50 | −46.23 | 26.37 | 27.07 | 9.21 | 2.26 |

The improvement rate $\tau$ was calculated as follows:

$$\tau = \frac{\eta_{\text{base}} - \eta_m}{\eta_{\text{base}}} \tag{22}$$

where $\eta_{base}$ is the error rate of the baseline classification method, and $\eta_m$ is the error rate of the multiple classification method. For the SMCS, $\eta_m$ is the error rate of the proposed method, and $\eta_{\text{base}}$ is the error rate of a single classifier constructed with the whole training dataset. For Bagging and Boosting, $\eta_m$ is the error rate of the implemented MCS method, and $\eta_{\text{base}}$ is, again, the error rate of a single classifier constructed with the whole training dataset.

With respect to SMCS performance, we observe that the proposed method works with most classifier-dataset combinations, even without parameter selection. Moreover, if we examine the details, we can select the best parameter sets from 28 possible combinations. We can see that these sets are not equally distributed among our 60 classifier-dataset combinations, and that the threshold of $\rho = 0.95$ and the neighborhood size of $m = n = 9$ are often the best choices (Fig. 11).

With respect to Bagging as an MCS, we observe general improvements in the performance of ensembles over that of constructed classifiers, i.e. classifiers trained with 67% of all training samples. However, a more fair comparison may involve using the performance of a single classifier constructed with the whole training dataset as a baseline, because this is the performance that we can achieve without using an MCS. In spite of substantial improvements on some classifier-dataset combinations, Bagging did not always improve accuracy on all datasets and all classifiers. The reason for this is that Bagging constructs some weak classifiers by using only a part of the training sample set, and the improvement brought about by the classifier combination scheme cannot always compensate for the loss of accuracy caused by these weak classifiers.

For Bagging, the small data size compounded with high dimensionality may result in the improvement from combining multiple classifiers not being able to compensate for the loss of accuracy from classifier training using only 67% of the data points for each classifier. Since we tested ensembles composed of 3–15 classifiers, the average ensemble size is about 9 classifiers, which may not be enough to overcome the loss of information on some datasets.

But such constraints can be easily circumvented by the SMCS, where training a single classifier makes use of all the available samples. With Boosting, we observe similar phenomena to those in Bagging. However, the use of more difficult data points for training may actually make the problem of dimensionality worse in some datasets. Consequently, in our experiments, Boosting may

**Table 4**
The error rates of Boosting on the experimental datasets. All the experiments were repeated 30 times. We show the error rates of a single classifier as the baseline (denoted "single classifier"), those of classifiers trained with only 67% of the samples (denoted "classifier pool"), and those of classifiers created with Boosting and combined by majority voting (denoted "average boosting").

| Dataset | Method | LDC (%) | QDC (%) | KNN (%) | PW (%) | MLP (%) | Tree (%) |
|---|---|---|---|---|---|---|---|
| Breast-tissue | Single classifier | 31.43 | 40.00 | 48.57 | 51.43 | 54.29 | 40.00 |
| Breast-tissue | Classifier pool | 56.57 | 56.99 | 56.63 | 57.51 | 57.89 | 57.12 |
| Breast-tissue | Average boosting | 51.58 | 52.03 | 51.88 | 52.14 | 52.22 | 51.63 |
| Breast-tissue | Error change | 64.11 | 30.07 | 6.81 | 1.37 | −3.82 | 29.08 |
| Bupa-liver | Single classifier | 33.04 | 41.74 | 32.17 | 46.09 | 33.91 | 40.00 |
| Bupa-liver | Classifier pool | 38.70 | 38.82 | 38.65 | 38.99 | 38.97 | 38.91 |
| Bupa-liver | Average boosting | 37.64 | 37.51 | 37.48 | 37.55 | 37.77 | 37.93 |
| Bupa-liver | Error change | 13.92 | −10.13 | 16.51 | −18.53 | 11.38 | −5.18 |
| Glass | Single classifier | 30.99 | 36.62 | 33.80 | 40.85 | 61.97 | 25.35 |
| Glass | Classifier pool | 48.79 | 47.89 | 48.26 | 48.28 | 48.03 | 48.75 |
| Glass | Average SMCS | 45.03 | 45.13 | 44.98 | 45.12 | 44.95 | 44.69 |
| Glass | Error change | 45.31 | 23.24 | 33.07 | 10.45 | −27.47 | 76.31 |
| Image segmentation | Single classifier | 7.33 | 8.00 | 10.86 | 6.19 | 39.43 | 11.62 |
| Image segmentation | Classifier pool | 16.97 | 15.66 | 15.79 | 15.92 | 16.13 | 16.79 |
| Image segmentation | Average boosting | 12.85 | 12.84 | 12.81 | 12.73 | 12.72 | 12.78 |
| Image segmentation | Error change | 75.31 | 60.50 | 17.96 | 105.65 | −67.74 | 9.98 |
| Iris | Single classifier | 8.00 | 4.00 | 10.00 | 8.00 | 4.00 | 24.00 |
| Iris | Classifier pool | 11.61 | 12.35 | 12.39 | 13.68 | 11.95 | 12.66 |
| Iris | Average boosting | 11.19 | 11.27 | 11.30 | 11.32 | 11.50 | 11.63 |
| Iris | Error change | 39.88 | 181.67 | 12.95 | 41.55 | 187.38 | −51.55 |
| Parkinsons | Single classifier | 15.38 | 15.38 | 20.00 | 16.92 | 12.31 | 13.85 |
| Parkinsons | Classifier pool | 18.65 | 18.25 | 18.59 | 18.40 | 19.06 | 18.78 |
| Parkinsons | Average boosting | 17.93 | 17.81 | 17.65 | 17.68 | 17.84 | 17.96 |
| Parkinsons | Error change | 16.61 | 15.80 | −11.76 | 4.48 | 44.91 | 29.70 |
| Vowel | Single classifier | 39.09 | 11.21 | 29.09 | 3.03 | 84.24 | 30.61 |
| Vowel | Classifier pool | 37.94 | 37.83 | 37.43 | 37.76 | 37.70 | 37.97 |
| Vowel | Average boosting | 31.33 | 31.37 | 31.48 | 31.80 | 31.68 | 31.88 |
| Vowel | Error change | −19.85 | 179.80 | 8.21 | 949.63 | −62.39 | 4.15 |
| Wdbc | Single classifier | 8.42 | 6.84 | 6.32 | 7.37 | 3.68 | 10.00 |
| Wdbc | Classifier pool | 6.88 | 6.95 | 6.88 | 6.88 | 6.89 | 6.87 |
| Wdbc | Average boosting | 6.58 | 6.61 | 6.56 | 6.61 | 6.61 | 6.62 |
| Wdbc | Error change | −21.89 | −3.34 | 3.78 | −10.26 | 79.66 | −33.83 |
| Wine | Single classifier | 1.69 | 0.00 | 35.59 | 30.51 | 37.29 | 18.64 |
| Wine | Classifier pool | 19.86 | 19.72 | 19.51 | 19.35 | 19.60 | 19.43 |
| Wine | Average boosting | 1923 | 19.04 | 19.04 | 18.99 | 18.95 | 18.85 |
| Wine | Error change | 137.58 | *Inf* | −46.50 | −37.75 | −49.18 | 1.10 |
| Yeast | Single classifier | 41.01 | 100.00 | 42.42 | 42.22 | 48.89 | 52.53 |
| Yeast | Classifier pool | 58.92 | 58.69 | 58.82 | 59.06 | 58.36 | 58.46 |
| Yeast | Average boosting | 53.62 | 53.40 | 53.81 | 53.32 | 53.65 | 53.74 |
| Yeast | Error change | 30.76 | −46.60 | 26.84 | 26.30 | 9.74 | 2.30 |

not perform as well as Bagging in some classifier-dataset combinations. Again, the curse of dimensionality may be a potential cause, given that some datasets may not have enough samples for adequate classifier training.

To summarize, the improvement achieved by combining multiple classifiers cannot compensate for the loss of accuracy from classifier training using only 67% of the data points for each classifier. Moreover, the Boosting mechanism encourages the use of more difficult data points for classifier training. While this may work reasonably well with a large data size, it may not be an optimal strategy with a small one.

Insufficient data samples with a high proportion of difficult samples may even degrade classifier accuracy in Boosting.

We should stress again that it may not be fair to compare the SMCS with Bagging and Boosting, given a number of differences in their methodologies. The SMCS deploys an inherently dynamic reference sample selection procedure by using evaluation datasets, whereas Bagging and Boosting simply combine the outputs of all the classifiers, and ensemble selection schemes are not applied in these experiments. Furthermore, if we increase the ensemble size from 15 to 150, and the percentage of samples used for classifier training from 67% to 80%, we may obtain different results with Bagging and Boosting. The same applies if we increase the evaluation dataset size, the neighborhood size, and the SMCS threshold.

Our intention in carrying out comparative experiments is to understand to what extent the SMCS can improve classification performance on different datasets and different classifiers (see Figs. 9 and 10), and not to judge the superiority of the MCS or the SMCS.

## 5. Discussion

The experimental results confirm the validity of the proposed SMCS as an applicable scheme for an MCS. This is especially true when we encounter the curse of dimensionality, and can only train weak classifiers.

Even without parameter selection, the average performance of the SMCS is still comparable to those of Bagging and Boosting (see Fig. 12). Moreover, we observe a generally positive performance enhancement on most datasets and classification methods in our experiment. This indicates that diversity generated from data can turn out to be quite useful, and that the noise inherent in diversity can be controlled.

Because of the different nature of the SMCS, it may not be straightforward to compare its performance with results in the literature, especially when heterogeneous ensembles, i.e. multiple classifiers in an ensemble with different classification methods, were implemented. The complexity of the performance comparison is compounded by the fact that different authors used different dataset partitions on the same database.

Notwithstanding these differences, we can still gain some insight into the capability of the SMCS with several simple comparisons. In Woloszynski and Kurzynski (2010), about 40% of the overall data points were used to train MLP ensembles and different ensemble selection methods were tested. Error rates on the Yeast dataset range from 43.38% to 48.97%, compared to an average 47.58% with the SMCS, and on the Iris dataset, they range from 6.89% to 5.00%, compared with 4.50% with the SMCS. In Polikar, DePasquale, Mohammed, Brown, and Kuncheva (2010), different percentages of overall data points and different numbers of features were used to train MLP ensembles composed of 1000 classifiers, and average error rates on the wdbc dataset range from 4.00% to 13.66%, compared with an average 2.18% with the SMCS. In Polikar et al. (2010), about different percentage of overall data points and different number of features were used to train MLP ensembles composed of 1000 classifiers, and average error rates on wdbc dataset range from 4.00% to 13.66%, compared with an average 2.18% with the SMCS.

In Ulas, Semerci, Yildiz, and Alpaydin (2009), $66.7 \times 80\% = 53.33\%$ overall data points were used to train Decision Tree ensembles. The average error rate on wdbc is 9.1%, compared with 5.11% with the SMCS; that on Bupa is 42.3%, compared with 31.43% with the SMCS; that on Glass is 49.6%, compared with 21.83% with the SMCS; that on Iris is 10.6%, compared with 15.43% with the SMCS; that on Wine is 6.3%, compared with 5.45% with the SMCS; and that on Yeast is 40.3%, compared with 42.76% with the SMCS. Again, given that the training dataset and test dataset are different, plus the classifier parameter difference, the comparison may not be a fair one. Nevertheless, the results seem to suggest that the SMCS and other conventional MCS are comparable in terms of performance.

In fact, however, our purpose is not to demonstrate that the SMCS is superior to the MCS, but rather that it can be applied on classification problems and that it has the potential to be further optimized. The elimination of multiple classifiers means that many essential MCS issues, such as classifier generation, classifier selection, and classifier combination, can be transformed into another set of issues, such as pseudo data point generation, reference sample selection, and multiple classification combination. This means that a dilemma in an MCS issue may not be a problem at all for the SMCS. This may open another door for general classification accuracy enhancement.

Furthermore, we notice that the SMCS does not generate many pseudo data points in our experiment (See Fig. 13), which may be due to the relative small reference dataset size, and the choice of $m$, $n$ and $\rho$. This may also indicate that the SMCS is more cautious than the conventional MCS. In our experiment, the MCS simply generates a fixed number of classifiers and combines their outputs, whereas the SMCS carefully evaluates reference samples and selects adequate ones. Hence, the SMCS may not even generate any pseudo data points for a test sample if it regards all available reference samples as not adequate. The advantage is that the SMCS in general prevents classification accuracy deterioration, but the disadvantage is that there may be certain cases that there is no classification accuracy improvement.

The parameters $m$, $n$ and $\rho$ also have an impact on the number of generated pseudo points. The correlation between $m$,$n$ and the number of pseudo points is 0.0432, and that between $\rho$ and the number of pseudo points is $-0.2428$. Hence, we might attempt to conclude that the smaller the $\rho$, the larger the number of generated pseudo points. Nevertheless, more experiments may be needed to have a better understanding on the effects of $m$, $n$ and $\rho$.

To summarize, there are several critical aspects to the potential impact of the proposed SMCS:

(1) *Dynamic decision boundary:* Unlike the MCS, which attempts to combine multiple decision boundaries from multiple classifiers in order to achieve an optimal decision boundary, the SMCS operates under the assumption that a static optimal boundary is difficult or impossible to draw by combining multiple boundaries, or may not even exist. Instead, it tends to make dynamic decisions given a static decision boundary by generating pseudo data points for multiple classifications, and therefore shifts the complexity of decision boundary optimization to pseudo data point generation optimization.
(2) *Compatibility with the MCS:* Selection of a suitable classification scheme does not have to be an either/or proposition, as it is feasible to apply both the MCS and the SMCS on a dataset, where each classifier trained with an MCS (such as Bagging or Boosting) can further generate multiple pseudo test

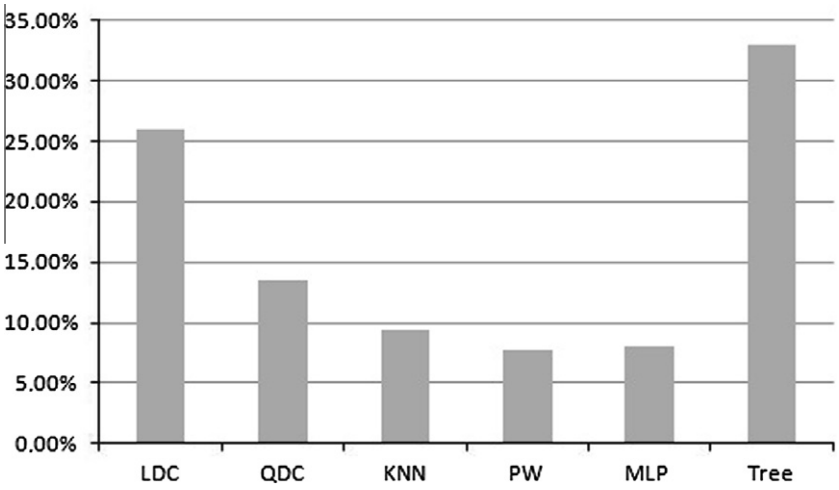


**Fig. 9.** Average improvement rates using the SMCS without parameter selection and with different classification methods. The result of the QDC on the Wine dataset is ignored, because it is infinite.

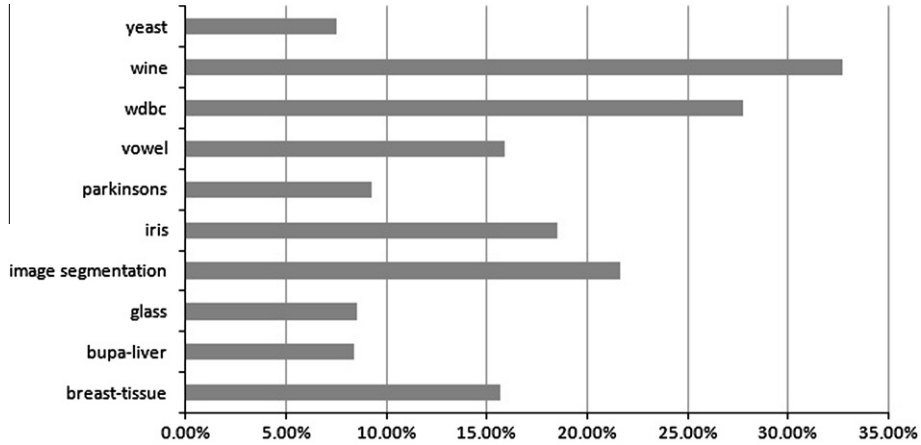


**Fig. 10.** Average improvement rates using the SMCS without parameter selection and on different datasets. Note that the result of the QDC on the Wine dataset is ignored, because it is infinite.

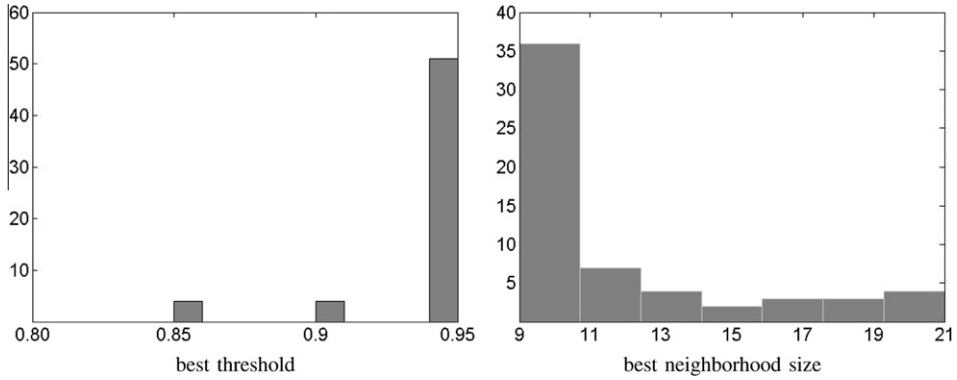


**Fig. 11.** Histograms of the best parameter choices on 60 classifier-dataset combinations for the SMCS.

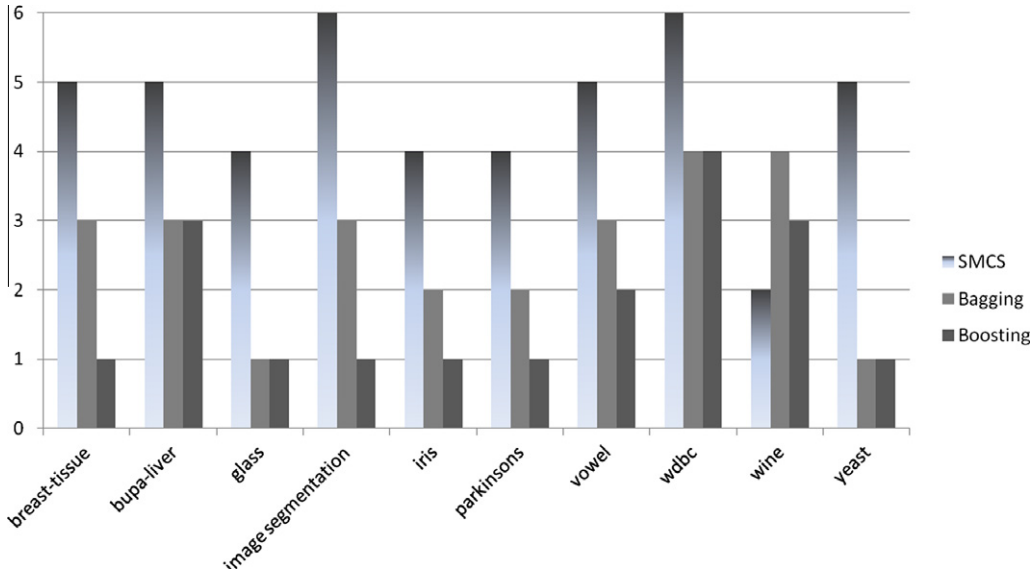


**Fig. 12.** Comparison of performances of the proposed SMCS with Bagging and Boosting. For each dataset, we show the number of classification algorithms implemented with MCS that actually improve the classification accuracy of the base classifier.

data points for each test sample to be classified. In this case, we actually generate a dual multiple classification system, which may further improve the performance of traditional MCS or the proposed SMCS.

(3) *Flexibility in pseudo data points generation: Unlike* Bagging and Boosting, where the generation of multiple classifiers is generally quite straightforward, the proposed SMCS tends to be more flexible and it can have almost infinite variations;

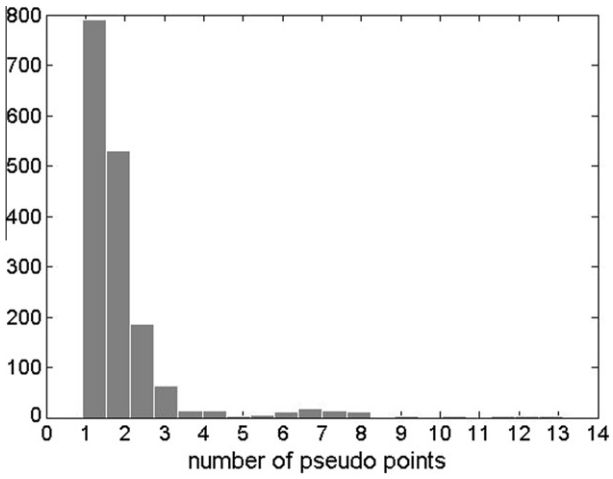


**Fig. 13.** The average number of generated pseudo data points per experiement for test samples on 60 classifier-datasets combinations. With 28 parameter sets (28 combinations of $m$, $n$ and $\rho$), a total of $60 \times 28 = 1680$ average pseudo data points were illustrated.

such as on the choice of the pseudo data point generation function, on the evaluation of reference samples, and on the adjustment of neighborhood size, etc. Consequently, the best SMCS scheme may be different for each dataset, and this indicates more opportunity for performance enhancement.

(4) *Reduced cost in classifier training time:* Compared with traditional MCS, the SMCS requires the training of only one classifier. Suppose, for example, that an MCS needs to train $K$ classifiers and requires training time $T_K$, the proposed SMCS would require training time $T_1$, and $T_1 \approx \frac{T_K}{K}$. This represents a speeding up by a factor of about $K$ for classifier training. When $K$ becomes large, such as $100 \sim 4000$ (Polikar et al., 2010), the gain may be substantial.

(5) *Reduced cost inensemble selection time:* Classifier training represents only a part of the cost of ensemble construction, because subsequent ensemble selection must be conducted to select the best subset of classifiers (Ruta & Gabrys, 2005; Kuncheva & Rodriguez, 2007; Martinez-Munoz, Hernandez-Lobato, & Suarez, 2009; Ko, Sabourin, & Britto, 2008; Santos et al., 2008). Because the SMCS uses only one classifier, there is no need for classifier subset selection. In fact, classifier subset selection is replaced by reference sample selection in the SMCS. This operation in the proposed SMCS is quite straightforward, requiring only nearest neighbor identification and a sum operation. It is therefore less time consuming than traditional classifier selection.

Nevertheless, the classification problem is not solved without cost. Although the SMCS reduces classifier training cost considerably, it actually increases the classification cost on each test sample. In other words, the SMCS shifts the cost of classifier training to the cost of pseudo test data point classification. As a result, if the cost of classification is critical for a system performance and the training cost is negligible, the SMCS may not be suitable for the system. On the contrary, if the training cost is prohibitively high and the classification cost is less substantial, then the SMCS should be considered as a potential solution.

Owing to its relatively low cost in terms of classifier training, it may also be a better choice if frequent classifier updating is required. Assuming $K$ classifiers for an MCS, the updating of classifiers may require the retraining of all $K$ classifiers, whereas the SMCS only needs to retrain a single classifier. The ease of classifier modification may make the SMCS more agile, in terms of adapting to environmental changes, and also more suitable for classifier optimization.

Finally, one drawback of the SMCS is that the current generation of pseudo data points can only be performed on numerical features. If there are categorical features present, the pseudo data points generated should keep their original categorical features, while modifying the numerical ones. However, if a dataset contains only categorical features, then the current SMCS cannot be applied to the problem. It is therefore of interest to investigate how pseudo features can be generated on categorical features.

## 6. Conclusion and future work

In this paper, we propose a Single Classifier-based Multiple Classification Scheme (SMCS) that uses only a single classifier to generate multiple classifications for a given test sample. Because of the presence of multiple classifications, classification combination schemes, such as majority voting, can be applied, and hence the mechanism may improve classification accuracy in a similar way to ensembles of classifiers (EoC).

Unlike EoC, the proposed SMCS only needs a single classifier to generate multiple classifications. This is because the SMCS generates multiple pseudo test data points for each test sample. The pseudo test data points generated are then classified with the original test sample, and the multiple classifications based on these pseudo test data points are then combined to yield a final class label output.

Because the SMCS does not require the presence of multiple classifiers, it reduces the overhead of repetitive classifier training. However, the creation of pseudo test data points does require some calculations. As a result, the proposed SMCS shifts the time cost from the classifier training stage to the test stage. From this perspective, the SMCS will be of greater interest for online classification than for batch mode classification.

More importantly, because it only has one classifier, the SMCS can easily update that classifier in an incremental way at far less cost than the conventional EoC scheme. Moreover, almost infinite variations of the SMCS exist, including the choice of pseudo data point generation function, the evaluation of reference samples, and the adjustment of neighborhood size. These represent a further opportunity to enhance classification accuracy and adapt the SMCS to different classification problems.

Furthermore, the SMCS and traditional MCS can be used together, as they are conceptually compatible. Consequently, the SMCS can be readily implemented on MCS classifiers. For these reasons, we believe that the SMCS will be a useful contribution to the EoC community.

The experimental results indicate that the SMCS works to some degree, and that it is fairly resistant to the curse of dimensionality. Also, the SMCS performances are comparable with those of Bagging and Boosting. Finally, we think it may be useful to implement some parameter optimization schemes by using a second evaluation subset. However, this will require a substantial number of samples, and will be a focus of our future work.

To conclude, we can state that:

(1) Diversity can be extracted from datasets directly, without the training of multiple classifiers, and it can be injected into multiple classifications with pseudo data point generation.
(2) The extent of diversity or noise can be adjusted by controlling the test sample neighborhoods $m$ and $n$.
(3) In order to generate multiple pseudo test data points with inherent diversity, we can use a threshold $\rho$ to decide which

reference samples to select. Consequently, we can use only pseudo test data points generated with selected reference samples.

(4) Given that multiple pseudo test data points are classified to generate multiple classifications, in general we can reduce by an average of 16% the error rates on weak classifiers with parameters $9 \leqslant m = n \leqslant 21$ and $0.80 \leqslant \rho \leqslant 0.95$.

(5) The SMCS shifts the complexity of the static decision boundary optimization into dynamic decision boundary optimization through the use of pseudo test data point generation.

(6) The SMCS reduces the costs of classifier training and classifier selection for an EoC, but incurs extra classification cost. The SMCS also demonstrates some resistance to the curse of dimensionality. However, it can be applied only to numerical features, and not to categorical features.

It is not our intention to assert that the current SMCS mechanism outperforms other MCS, but rather to show that the SMCS is a viable research green field in the MCS community, because of the concept of the dynamic decision boundary, the elimination of multiple classifiers in a system, the infinite number of possible pseudo data point generation variations on the same SMCS principles, compatibility with general MCS, and its suitability for online classification.

So far, there has been no work published in the literature concerning the possibility of using a single classifier for multiple classifications, except for a specific scheme tailored for Ensembles of Hidden Markov Models (HMM) (Ko, Cavalin, Sabourin, & de Souza Britto, 2009). However, the scheme in Ko et al. (2009) is only suitable for HMM and cannot be generalized for other types of classifiers. As far as the authors know, this is the first work on a generic single classifier-based multiple classification scheme. We believe that this work is just a starting point for future multiple classification research, and many questions remain. One interesting research direction will be the optimal partition between reference samples and evaluation samples. Some data samples may be more suitable for use as reference samples, whereas others may be better evaluation samples. We have not yet investigated an optimal partition mechanism, but this will be of interest to pursue in future work.

Furthermore, there is an opportunity to reduce the computation cost associated with the SMCS by reducing the number of reference samples and evaluation samples. The current study does not reveal the relationship between SMCS performance and these numbers of samples, however. Such a study must be conducted on a rather large dataset. The insight gained may even help users further reduce computation complexity.

Still, there remains a considerable amount of work that can be pursued in the future on a pseudo data generation mechanism, for example, or on parameter optimization, combining the SMCS with the MCS, and so forth. In our opinion, the real benefits for the EoC community will come from further investigation of the SMCS.

## References


Afifi, A. A., & Azen, S. P. (1979). *Statistical analysis: A computer oriented approach* (2nd ed.). New York: Academic Press.

Baird, H. (1990). Document image defect models. In *Proceedings, IAPR workshop on syntactic and structural pattern recognition* (pp. 38–46). NJ: Murray Hill.

Bottou, L., & Vapnik, V. N. (1992). Local learning algorithms. *Neural Computation, 4*(6), 888–900.

Brown, G., Wyatt, J., Harris, R., & Yao, X. (2005). Diversity creation methods: A survey and categorisation. *International Journal of Information Fusion, 6*(1), 5–20.

Decoste, D., & Schoelkopf, B. (2002). Training invariant support vector machines. *Machine Learning, 46*(1), 161–190.

Didaci, L., Giacinto, G., Roli, F., & Marcialis, G. L. (2005). A study on the performances of dynamic classifier selection based on local accuracy estimation. *Pattern Recognition, 38*(11), 2188–2191.

Duin, R. P., Juszczak, P., Paclik, P., Pekalska, E., de Ridder, D., Tax, D., & Verzakov, S. (2007). *PR-Tools4.1, a matlab toolbox for pattern recognition*. <http://prtools.org>.

Fleiss, J. L., Levin, B., & Paik, M. C. (2003). *Statistical methods for rates and proportions* (2nd ed.). New York: John Wiley & Sons.

Fumera, G., Roli, F., & Serrau, A. (2008). A theoretical analysis of bagging as a linear combination of classifiers. *IEEE Transactions on Pattern Analysis Machine Intelligence, 30*(7), 1293–1299.

Giacinto, G., & Roli, F. (2001). Design of effective neural network ensembles for image classification purposes. *Image and Vision Computing, 19*(9–10), 699–707.

Grove, A., & Schuurmans, D. (1998). Boosting in the limit: Maximizing the margin of learned ensembles. In: Proceedings of the fifteenth national conference on artificial intelligence (pp. 692–699).

Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 12*, 993–1001.

Ho, T. K. (1998). The random space method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 20*(8), 832–844.

Huang, Y. S., & Suen, C. Y. (1995). A method of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 17*, 90–93.

Kittler, J., Hatef, M., Duin, R., & Matas, J. (1998). On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 20*(3), 226–239.

Ko, A. H. R., Sabourin, R., & de Souza Britto, A., Jr. (2006). Combining diversity and classification accuracy for ensemble selection in random subspaces. In: Proceedings of the international joint conference on neural networks (pp. 2144–2151).

Ko, A. H. R., Sabourin, R., & de Souza Britto, A. Jr., (2009). Compound diversity functions for ensemble selection. *International Journal of Pattern Recognition and Artificial Intelligence, 23*(4), 659–686.

Ko, A. H. R., Cavalin, P., Sabourin, R., & de Souza Britto, A. Jr., (2009). Leave-one-out-training and leave-one-out-testing hidden Markov models for a handwritten numeral recognizer: The implication of a single classifier and multiple classifications. *IEEE Transactions on Pattern analysis and Machine Intelligence, 31*(12), 2168–2178.

Ko, A. H. R., Sabourin, R., & Britto, A. S. Jr., (2008). From dynamic classifier selection to dynamic ensemble selection. *Pattern Recognition, 41*(5), 1718–1731.

Kohavi, R., & Wolpert, D. H. (1996). Bias plus variance decomposition for zero-one loss functions. In: Proceedings of the international machine learning conference (ICML 1996) (pp. 275–283).

Kuncheva, L. I. (2002). A theoretical study on six classifier fusion strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24*(2), 281–286.

Kuncheva, L. I., Skurichina, M., & Duin, R. P. W. (2002). An experimental study on diversity for bagging and boosting with linear classifiers. *International Journal of Information Fusion, 3*(2), 245–258.

Kuncheva, L. I., & Whitaker, C. J. (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning, 51*(2), 181–207.

Kuncheva, L. I., & Rodriguez, J. J. (2007). Classifier ensembles with a random linear oracle. *IEEE Transactions on Knowledge and Data Engineering, 19*(4), 500–508.

Martinez-Munoz, G., Hernandez-Lobato, D., & Suarez, A. (2009). An analysis of ensemble pruning techniques based on ordered aggregation. *IEEE Transactions on Pattern Analysis Machine Intelligence, 31*(2), 245–259.

Opitz, D., & Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research, 11*, 169–198.

Poggio, T., & Vetter, T. (1992). Recognition and structure from one 2D model view: Observations on prototypes, object classes and symmetries. A.I. Memo No. 1347, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Partridge, D., & Krzanowski, W. (1997). Software diversity: Practical statistics for its measurement and exploitation. *Information and Software Technology, 39*, 707–717.

Pekalska, E., Skurichina, M., & Duin, R. P. W. (2004). Combining dissimilarity-based one-class classifiers. In: International workshop on multiple classifier systems (MCS 2004) (pp. 122–133).

Polikar, R., DePasquale, J., Mohammed, H. S., Brown, G., & Kuncheva, L. I. (2010). Learn++ MF: A random subspace approach for the missing feature problem. *Pattern Recognition* (43), 3817–3832.

Ruta, D., & Gabrys, B. (2001). Analysis of the correlation between majority voting error and the diversity measures in multiple classifier systems. In: Proceedings of the 4th international symposium on soft computing.

Ruta, D., & Gabrys, B. (2005). Classifier selection for majority voting. *International Journal of Information Fusion*, 63–81.

Santos, E. M. D., Sabourin, R., & Maupin, P. (2008). A dynamic overproduce-and-choose strategy for the selection of classifier ensembles. *Pattern Recognition, 41*(10), 2993–3009.

Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics, 26*(5), 1651–1686.

Shipp, C. A., & Kuncheva, L. I. (2002). Relationships between combination methods and measures of diversity in combining classifiers. *International Journal of Information Fusion, 3*(2), 135–148.

Tax, D. M. J., Van Breukelen, M., Duin, R. P. W., & Kittler, J. (2000). Combining multiple classifiers by averaging or by multiplying. *Pattern Recognition, 33*(9), 1475–1485.

Ulas, A., Semerci, M., Yildiz, O. T., & Alpaydin, E. (2009). Incremental construction of classifier and discriminant ensembles. *Information Sciences* (179), 1298–1318.

Woloszynski, T., & Kurzynski, M. (2010). A measure of competence based on randomized reference classifier for dynamic ensemble selection. In: 2010 international conference on pattern recognition.

Zouari, H., Heutte, L., Lecourtier Y., & Alimi, A. (2004). Building diverse classifier outputs to evaluate the behavior of combination methods: The case of two classifiers. In: International workshop on multiple classifier systems (MCS 2004) (pp. 273–282).