Int. J. Appl. Math. Comput. Sci., 2012, Vol. 22, No. 4, 855–866
DOI: 10.2478/v10006-012-0063-0

COMBINED CLASSIFIER BASED ON FEATURE SPACE PARTITIONING

MICHAŁ WOŹNIAK, BARTOSZ KRAWCZYK

Department of Systems and Computer Networks
Wrocław University of Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland
e-mail: {michal.wozniak,bartosz.krawczyk}@pwr.wroc.pl

This paper presents a significant modification to the AdaSS (*Adaptive Splitting and Selection*) algorithm, which was developed several years ago. The method is based on the simultaneous partitioning of the feature space and an assignment of a compound classifier to each of the subsets. The original version of the algorithm uses a classifier committee and a majority voting rule to arrive at a decision. The proposed modification replaces the fairly simple fusion method with a combined classifier, which makes a decision based on a weighted combination of the discriminant functions of the individual classifiers selected for the committee. The weights mentioned above are dependent not only on the classifier identifier, but also on the class number. The proposed approach is based on the results of previous works, where it was proven that such a combined classifier method could achieve significantly better results than simple voting systems. The proposed modification was evaluated through computer experiments, carried out on diverse benchmark datasets. The results are very promising in that they show that, for most of the datasets, the proposed method outperforms similar techniques based on the clustering and selection approach.

Keywords: pattern recognition, combined classifier, multiple classifier system, clustering and selection algorithm, evolutionary algorithm.

1. Introduction

There are a number of proposals on how to automate the classification process (Duda *et al.*, 2001). Nevertheless, there is not a single pattern recognition algorithm that is appropriate for all the tasks we are faced with, since each classifier has its own domain of competence (Wolpert, 2001). Usually we can pool different classifiers to solve a given problem. Therefore, methods that can exploit the strengths of individual classifiers are currently the focus of intense research (Jain *et al.*, 2000). It is worth noting that the incompetence area, i.e., the subset of the feature space where all individual classifiers make the wrong decision, is typically small (Polikar, 2006).

The presented approach is called a Multiple Classifier System (MCS), a combined classifier, or a classifier ensemble (Kuncheva, 2004), and its main components are depicted in Fig. 1. In this concept, the greatest effort is concentrated on combining the outputs of elementary classifiers at our disposal for a given classification problem. This concept was first presented by Chow (1965) who proved that the decision of independent classifiers with appropriately defined weights is optimal. Here are some of the advantages of MCSs:

- The design of an MCS does not differ from that of a classical pattern recognition (Giacinto *et al.*, 2000) application. In the standard approach we select the most valuable features and choose the best classification method from the set of those available. The design of a classifier ensemble aims to create a set of complementary/diverse classifiers and assign an appropriate fusion method, which can optimally combine the individual classifiers' outputs.

- Some works report that MCSs can improve the overall performance compared with the best

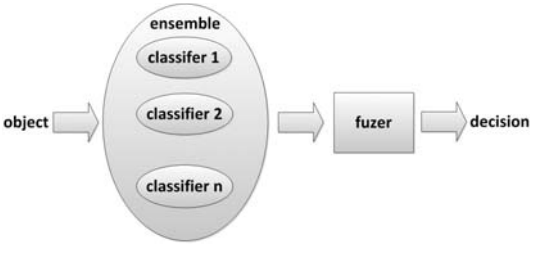

Fig. 1. Overview of a multiple classifier system.

individual classifier, because they are able to exploit unique strengths of each of the individual classifiers. In some cases (e.g., the majority voting by a group of independent classifiers) their characteristics have been proven in an analytical way (Tumer and Ghosh, 1996). Additionally, an MCS protects against selection of the worst classifier for a small sample (Marcialis and Roli, 2003).

- Many machine learning algorithms (e.g., C4.5 based on a top-down induction decision tree concept) are *de facto* heuristic search algorithms, which cannot guarantee that an optimal model is found. Therefore, the combined approach, which could start searching from different points of the search space, seems to be an attractive proposition.

- Combined classifiers could be used in efficient computing environments such as parallel and multithreaded computer architectures (Wilk and Woźniak, 2011). Another attractive area of application are distributed computing systems (P2P, GRID) (Kacprzak *et al*., 2012; Walkowiak, 2010; Chmaj *et al*., 2012), especially in the case of a database that is partitioned for privacy reasons and only the final decision is available at each node of the computer network.

There are a number of important issues that must be taken into consideration when building multiple classifier systems. These can be grouped into the following problems:

- Selecting a pool of diverse and complementary individual classifiers for the ensemble.

- Designing a fuser, aimed at creating a mechanism that can exploit the strengths of the selected classifiers and combine them optimally.

- Proposing the topology, i.e., interconnections between classifiers in the ensemble.

We do not address the last issue because most of the combined classifiers are based on a parallel topology, which has a good methodological background (Kuncheva, 2004) and is used in this work.

Selecting members of the committee with different components seems interesting. Apart from increasing the computational complexity, combining similar classifiers should not contribute much to the MCS under construction. An ideal ensemble consists of classifiers with high accuracy and high diversity, i.e., mutually complementary. First, classifiers must be selected to obtain positive results from their fusion. Many algorithms covering this subject were inspired by guidelines on how to design reliable software, among which Partridge and Krzanowski (1997), Brown and Kuncheva (2010) as well as Smetek and Trawinski (2011) are worth mentioning. A strategy for generating the ensemble should guarantee an improvement in its diversity. There are several proposals on how to enforce the diversity of an individual classifier pool listed below:

- We could use different partitions of a dataset or generate a number of datasets through data splitting, a cross-validated committee, bagging, or boosting (Kuncheva, 2004), in the hope that classifiers trained on different inputs would be complementary. Selected features are then used to train a pool of classifiers to assure the diversity of the pool. There are several propositions based on this principle such as random subspace (Ho, 1998). It is worth pointing out the interesting idea presented by Ting *et al*. (2011), who proposed a hierarchical method of ensemble creation, based on feature space splitting and then assigning binary classifiers (support vector machines) locally.

- We could train each individual classifier to recognize a subset of only predefined classes (e.g., a binary classifier: one class against the rest strategy) and then choose a fusion method that can recover the whole set of classes. Error-correcting output codes (Dietterich and Bakiri, 1995) are a well-known technique for such tasks.

- We could train individual classifiers based on different models or different versions of models.

For classification tasks where the cost of acquiring feature values (which could be interpreted as the price for examination or time required to collect the data for decision making) plays a key role, this must be taken into consideration during the classifier selection step. In the method presented by Krawczyk and Woźniak (2011), the authors suggested a way to utilize the exploitation cost of individual classifiers in the selection process.

Another important concept of classifier selection assumes a local specialization of individual classifiers. According to this proposal, a single classifier that achieves the best results is chosen from a pool for each demarcated partition of the feature space. Its answer is treated as the system answer, for all objects included in the partition. This methodology was described by Rastrigin and Erenstein (1981). Certain proposals based on this idea assume a local specialization of particular classifiers and only search for locally optimal solutions (Baram, 1998; Cordella *et al*., 2000; Giacinto *et al*., 2000; Goebel and Yan, 2004; Ruta and Gabrys, 2005), while other methods propose dividing the feature space and selecting (or training) a classifier for each partition (Kuncheva, 2000; Baruque *et al*., 2011).

Another important issue is the choice of the collective decision making method. We can divide the fusion algorithms mentioned above into two groups:

- methods that make decisions on the basis of outputs (labels) of individual classifiers,
- methods that propose constructing new discriminant functions based on continuous outputs (supports) of individual classifiers.

The former group includes voting algorithms (Biggio *et al*., 2007; Xu *et al*., 1992). Initially only majority voting schemes were implemented, but in later works more advanced methods were proposed. These take the importance of decisions coming from particular committee members into consideration (van Erp *et al*., 2002; Kuncheva *et al*., 2001).

Many known conclusions regarding the classification quality of MCSs have been derived analytically, but these are typically valid only under strong restrictions, such as particular cases of the majority vote (Hansen and Salamon, 1990), or make convenient assumptions, such as a classifier committee consisting only of independent classifiers. Unfortunately, such assumptions and restrictions are in most cases not very useful for solving practical problems. Here, we should mention the works that propose training the weights, which seems to be an attractive alternative method (Woods *et al*., 1997; Woźniak and Jackowski, 2009).

The second group of fusers is based on discriminant analysis. The main form of discriminants is a posterior probability typically associated with probabilistic pattern recognition models, although outputs of neural networks or other functions whose values are used to establish the decision of the classifier (the so-called support functions) could be considered as well. Aggregation methods that do not require learning use simple operators, like minimum, maximum, product, or mean. However, they are typically subject to very restrictive conditions (Duin, 2002), which limit their practical use. Therefore, the design of new fusion classification models, especially trained fusers, is currently the focus of intense research.

Assume that we have n classifiers $\Psi^{(1)}$, $\Psi^{(2)}$, $\ldots$, $\Psi^{(n)}$. For a given object $x \in \mathcal{X}$, each individual classifier decides whether it belongs to class $i \in \mathcal{M} = \{1, \ldots, M\}$ based on the values of discriminants. Let $F^{(l)}(i,\, x)$ denote a function that is assigned to class i for a given value of x, and that is used by the l-th classifier $\Psi^{(l)}$. The combined classifier Ψ uses the following decision rule (Jacobs, 1995):

$$\Psi(x) = i \quad \text{if} \quad \hat{F}(i,\, x) = \max_{k \in M} \hat{F}(k,\, x), \tag{1}$$

where

$$\hat{F}(i,\, x) = \sum_{l=1}^{n} w^{(l)} F^{(l)}(i,\, x), \quad \sum_{i=1}^{n} w^{(l)} = 1. \tag{2}$$

Next we consider the following possibilities for the weight assignment:

1. *Weights dependent on the classifier.* This is the traditional approach where weights are connected with a classifier and each discriminant of the *l*-th classifier is weighted by the same value $w^{(l)}$. The probability error of such a classifier can be estimated, e.g., as in the work of Woźniak (2008).

2. *Weights dependent on the classifier and feature vector.* A weight $w^{(l)}(x)$ is assigned to the *l*-th classifier and for a given x has the same value for each discriminant function used by it. In this type of model, known as a "mixture of experts", parameter estimation is normally used to establish the weights (Jacobs *et al*., 1991).

3. *Weights dependent on the classifier and class number.* A weight $w^{(l)}(i)$ is assigned to the *l*-th classifier and the *i*-th class. Here, the given classifier weights assigned to different classes may differ.

4. *Weights dependent on the classifier, class number, and feature vector.* A weight $w^{(l)}(i, x)$ is assigned to the *l*-th classifier, but for a given x its value may differ for discriminants assigned to each class.

In this work we focus on the third alternative, because, as shown by Woźniak and Zmyślony (2010), this type of fuser achieves fairly good quality and does not require *a priori* knowledge of the weights. This is in contrast to the case where weights are also dependent on the feature values. If weights depend on x, they are *de facto* functions and their estimation is more complicated, usually requiring *a priori* knowledge about them.

We make use of the approach that tries to divide the feature space into subspaces and then assigns a combined classifier to such a partition. This approach improves the AdaSS (*Adaptive Splitting and Selection*) algorithm (Jackowski and Woźniak, 2009) mainly by replacing the fuser based on the majority voting rule by a fuser using discriminant functions. Our approach uses a linear combination of discriminant functions of individual classifiers.

2. Related works

Because the AdaSS algorithm and our proposition are descendants of Kuncheva's CS (*Clustering and Selection*) algorithm, we first recall this method.

2.1. Clustering and selection algorithm. The CS algorithm (Kuncheva, 2000) consists of three main steps:

1. Selecting individual classifiers for a pool.

2. Establishing clustering algorithm parameters and partitioning the learning set according to a given algorithm.

3. Selecting the best individual classifier for each cluster according to its local quality.

The most important features of the CS algorithm are as follows:

- The CS algorithm uses clustering algorithms to divide the feature space, a task that involves separating some subsets of elements from the learning set based on their similarity (Jain *et al.*, 1999). Partitioning does not take into account modified clustering criteria, which find such feature space decisions to ensure that at least one classifier in the pool is able to achieve high accuracy in a given subspace. So there is no restriction on allowing cluster borders to cross the borders separating the fields with objects from particular classes. This is a desired effect, because the clustering considered does not aim to double discriminant functions but to separate fields in which these classifiers achieve a high quality of classification. As an example, consider the binary toy problem presented in Fig. 2. The pool consists of two simple linear classifiers. The output of the clustering algorithm is valid, yet at the same time it does not exploit the full potential of the given classifiers. This leads to a decrease in the overall accuracy of the ensemble. Now consider another partitioning of the feature space presented in Fig. 3, which takes into consideration competencies of the given classifiers, thus leading to better exploitation of these competencies.

- Feature space partitioning and selection of the classifiers are carried out sequentially. A natural consequence of this is the lack of feedback between steps. In the second step it is possible to find the best classifiers for the previously defined clusters. However, it is impossible to modify the shape of the clusters to adjust to the competencies of the chosen classifiers assigned to the clusters. Thus, there is no guarantee that the model obtained from the proposed partitioning is the most effective one.

- Finally, a limitation of the CS algorithm is that only one individual classifier is assigned to each cluster. This significantly limits the advantages of the existing pool of classifiers, from which a set of committees could be composed for each cluster, which could improve the achieved results.

2.2. AdaSS. The AdaSS algorithm, proposed by Jackowski and Woźniak (2009), fuses partitioning the feature space and assigning classifiers to each partition into one integrated process, when searching for optimal parameters for the model. The main advantage of this approach is that the training algorithm considers the shape of a region to determine the content of a classifier and, conversely, that the regions adapt to the competencies of the classifiers. Additionally, the majority voting rule is used to establish the decision of each area classifier. The objective of this complex optimizing task is to minimize the classification error. For the example binary toy problem, in AdaSS the clustering is done simultaneously with classifier selection. This provides feedback that forces the optimization procedure to change the form of the clusters.

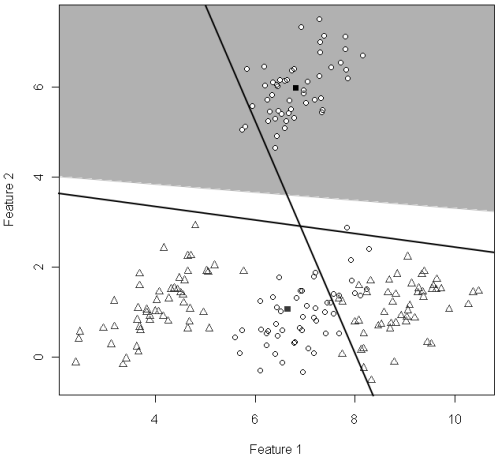

Fig. 2. Example outcome of a CS algorithm for a binary toy problem. One can easily see that the returned clusters do not fully exploit the possibilities offered by the two classifiers.

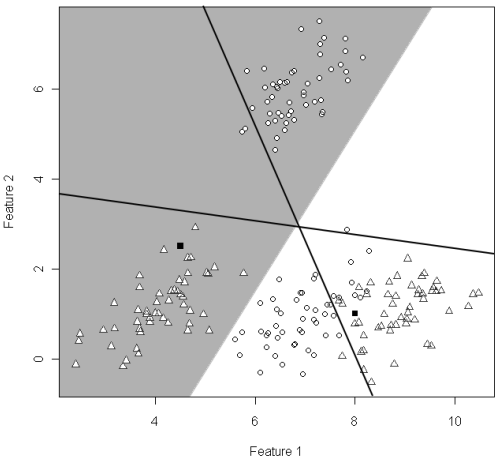

Fig. 3. Modified clusters for the binary toy problem. It is easily seen that here the competency of the classifiers is exploited more efficiently.

In this work we adapt the AdaSS model, noting that its major disadvantage is a fairly simple fuser model using the majority voting rule. This does not guarantee a quality improvement compared with the CS algorithm, especially when there are considerable quality differences between individual classifiers. This observation for a pool of independent classifier was described by Kuncheva *et al*. (2003) and Matan (1996), amongst others. Therefore, we opted to replace the fuser based on majority voting by a more promising method that allows an area classifier to be trained instead of being selected and uses a deterministic decision fusing rule.

2.3. Proposed modification to AdaSS. The main difference between the original AdaSS and this modification lies in the process of constructing the compound classifiers that are later assigned to each partition. We propose using the combined classifiers based on (1), although the common discriminant function is formulated as follows:

$$\hat{F}\left(i,\ x\right) = \sum_{l=1}^{n} w^{(l)}\left(i\right) F^{(l)}\left(i,\ x\right),$$

where

$$\sum_{i=1}^{n} w^{(l)} = 1, \quad \forall i \in \mathcal{M}. \tag{3}$$

Here we focus on the problem of establishing weights dependent only on the classifier and the class number. Because we use an evolutionary approach to train the classifier, we need an appropriate representation to code the fuser parameters.

In the case under consideration, the fuser training task leads to the problem of how to establish the following vector $\mathcal{W}$:

$$\mathcal{W} = \left[\mathcal{W}^{(1)},\ \mathcal{W}^{(2)},\ \ldots,\ \mathcal{W}^{(n)}\right], \tag{4}$$

which consists of weights assigned to each classifier and each class number,

$$\mathcal{W}^{(l)} = \left[w^{(l)}\left(1\right),\ w^{(l)}\left(2\right), \ldots, w^{(l)}\left(M\right)\right]^{\ T}. \tag{5}$$

The aim is to find such a fuser with the lowest misclassification rate of Ψ.

3. Model of a classifier

Assume that the feature space $\mathcal{X}$ is divided into a set of H constituents (Kuratowski and Mostowski, 1976), i.e.,

$$\begin{aligned} &\mathcal{X} = \bigcup_{h=1}^{H} \hat{\mathcal{X}}_h, \\ &\hat{\mathcal{X}}_k \cap \hat{\mathcal{X}}_l = \varnothing, \quad \forall k, l \in \{1, \ldots, H\}, \quad k \neq l, \end{aligned} \tag{6}$$

where $\hat{\mathcal{X}}_h$ denotes the h-th constituent (cluster). Ψ_h is a combined classifier assigned to the h-th cluster and decision according to (1), (3). $\mathcal{W}_h$ denotes its weight vector (4).

The number of clusters, H, is an arbitrarily chosen parameter. On the one hand, a larger number makes a wider exploration of the local competencies of the area classifiers possible, while on the other hand, it may lead to the overtraining of the entire system.

Here, we present the classification rule for the compound classifier Ψ, which returns the decision of the classifier assigned to the cluster to which the given object belongs:

$$\Psi(x) = i \iff \Psi_h(x) = i \text{ and } x \in \hat{\mathcal{X}}_h. \tag{7}$$

4. Training algorithm

The main idea of the learning procedure was based on the AdaSS algorithm (Jackowski and Woźniak, 2009), which uses an evolutionary approach (Ashlock, 2006; Troć and Unold, 2010), but we introduced several important changes such as the structure of the chromosome and improvement of some of the steps mainly associated with establishing mutation probabilities and protecting against overfitting.

To simplify the presentation of the algorithm, we assume that we are dealing with continuous features only. If needed, the presented method can be easily adapted to account for discrete attributes.

4.1. Representation. A chromosome Ch represents the model for compound classifier parameters. Its structure consists of two components. The first one embodies a set of centroids $\mathcal{C}$ and represents feature space partitioning into H clusters (5). The other includes definitions of the combined classifiers for each of the clusters (4),

$$Ch = [\mathcal{C}, \mathcal{W}], \tag{8}$$

where

$$\mathcal{C} = \{C_1, C_2, \ldots, C_H\}$$

and

$$\mathcal{W} = \{\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_H\}.$$

Each C_h is represented by a centroid,

$$\mathcal{C}_h = \{c_h^{(1)}, c_h^{(2)}, \ldots, c_h^{(d)}\}, \tag{9}$$

where d is the feature space dimension. Each $\mathcal{W}_h$ consists of weights assigned to each individual classifier (4),

$$\mathcal{W}_h = \begin{pmatrix} w_h^1(1) & w_h^2(1) & \ldots & w_h^n(1) \\ w_h^1(2) & w_h^2(2) & \ldots & w_h^n(2) \\ \vdots & \vdots & \ddots & \vdots \\ w_h^1(M) & w_h^2(M) & \ldots & w_h^n(M) \end{pmatrix}, \tag{10}$$

which is converted to a vector,

$$\mathcal{W}_h = [w_h^1(1), \ldots, w_h^1(M), w_h^2(1), \ldots, w_h^n(M)].$$

Let $\mathrm{member}(C, x)$ denote the function that returns the cluster index to which a given x belongs,

$$\mathrm{member}(C, x) = \arg\min_{h=1}^{H} \mathrm{dist}(x, C_h), \qquad (11)$$

where 'dist' denotes the Euclidean metric. In the case of a tie, i.e., when x is the same distance from more than one centroid, the lowest class number is returned. We should point out that when dealing with discrete attributes, a different metric should be defined. According to the definition (11), we can reformulate Eqn. (7) as follows:

$$\Psi(x) = \hat{\Psi}_{\mathrm{member}(\mathcal{C}, x_n)}(x_n). \qquad (12)$$

4.2. Learning material. The classifier training procedure requires adequate learning material, denoted as the learning set LS. We assume that it consists of K learning objects,

$$\mathcal{LS} = \{(x_1, j_1), (x_2, j_2), \ldots, (x_K, j_K)\}, \qquad (13)$$

where x_i denotes observations described in the i-th object, and j_i denotes its correct class label.

Usually $\mathcal{LS}$ is divided into two subsets: $\mathcal{TS}$, called the training set, is used during training, while the second subset $\mathcal{VS}$, called the validation set, is used to protect trained classifiers against overfitting (Alpaydin, 2010),

$$|\mathcal{TS}| = N, \quad |\mathcal{VS}| = K - N. \qquad (14)$$

4.3. Criterion. As the optimization criterion, we propose the frequency of correct decisions that $\hat{\Psi}$ makes on $\mathcal{TS}$:

$$Q(\hat{\Psi}) = \frac{1}{n} \sum_{n=1}^{N} (\delta(\hat{\Psi}_{\mathrm{member}(C, x_n)}(x_n), j_n)), \qquad (15)$$

where δ denotes Kronecker's delta.

4.4. Algorithm. As mentioned in the beginning, we use an evolutionary approach to find the best solution. Any procedure that is performed on the chromosome must take into account the fact that each of its parts has quite a different nature. Therefore, we enforce the rule that information will not be exchanged between parts of the chromosomes processed by evolutionary operators. An overview of the algorithm is presented as pseudocode in Algorithm 1.

The control parameters for the algorithm are as follows:

1. N_c: the upper limit of algorithm cycles,
2. N_p: the population size,
3. β: the probability of mutation,
4. γ: the probability of crossover,
5. Δ_m: the mutation range factor,
6. V: the upper limit of algorithm iterations with decreasing quality ($V < N_c$).

Algorithm 1. Overview of the training algorithm.

```
input:
  TS: training set
  VS: validation set
  H: number of clusters
Initialization;
V_c = 0
for (t = 1; t ≠ N_c; t++)
  {
  Mutation;
  Crossover;
  Selection_and_Reproduction;
  Protecting_against_overfitting;
  if Q_vs(t) < Q_vs(t − 1)
    {
    V_c = V_c + 1;
    if V_c = V;
    break;
    }
    else
    V_c = 0;
  }
Return_the_best_chromosome;
```

Given below is a detailed explanation of selected steps of the algorithm:

- *Initialization*
  Initialization involves setting the parameters for the algorithm and randomly creating the first generation of chromosomes. Each of the chromosomes in a generation is evaluated according to its fitness function value (15) to determine the elite chromosomes.

- *Selection_and_Reproduction*
  This step generates a set of members preserving all the constraints and implications resulting from the logic of the model and the values of the input parameters. Chromosomes for the next generation are selected using the roulette wheel selection scheme (Goldberg, 1989) to implement proportional random selection according to their fitness function values (15). The training set is

exploited for this purpose. To avoid losing ground in finding the highest-scoring chromosome, elitism (Srinivas and Patnaik, 1994) has been implemented, i.e., the highest scoring chromosome of the current generation is placed into the descendant population, without allowing any changes to its structure, such as cross-over, or being subjected to mutation.

- *Mutation*
  The mutation operator alters the member being processed by adding some random changes to its chromosome. Each component of the chromosome is processed separately and can be altered with a certain probability, that is, alteration together with an optimization progress according to

$$P_c(t) = \beta\frac{t}{N_c}, \quad P_w(t) = \beta - P_c(t), \tag{16}$$

  where t is the iteration index of the algorithm, $P_c(t)$ is the mutation probability of the centroid vector during the t-th step, and $P_w(t)$ is the mutation probability of the weight vector during the t-th step. According to the schema, in the early phases of optimization, special emphasis is placed on searching for a possible feature space partitioning. Over the course of the learning progress, attention is shifted onto the classifiers assigned to the partitions and their fine adjustments. Mutation involves adding a vector of numbers randomly generated according to the normal density distribution (with mean equal to 0 and standard deviation set to Δ_m).

- *Crossover*
  All the chromosomes (with the exception of the elite chromosome) are paired up, and with probability γ, crossed over according to the two-point rule (Goldberg, 1989).

- *Protecting_against_overfitting*
  The main purpose of this procedure is to protect the classifier against overfitting. The procedure uses $\mathcal{VS}$ to calculate the fitness of the elite chromosome, Q_{VS}, in the same way as for a regular population assessment (15). The procedure terminates the optimization process if deterioration of the result obtained by the highest scoring chromosome is observed in the course of V subsequent learning cycles.

5. Experiments

The main objective of the experiments was to examine the behavior of the proposed method and to compare it with the original AdaSS and CS algorithms.

5.1. Setup. All experiments were carried out in the R environment (Team, 2008), and computer implementations of the classification and optimization methods used were taken from dedicated packages built into the above-mentioned software. This ensured that results achieved the best possible efficiency and that performance was not diminished by a bad implementation.

A pool of five individual classifiers consisting of slightly undertrained neural networks (for which the training process was deliberately stopped early) was used in each of the experiments to ensure diversity of the simple classifiers, thereby allowing their local competencies to be exploited. The details of the neural networks used are as follows:

- activation function: sigmoidal,
- learning method: backpropagation,
- architecture: five neurons in the hidden layer and the last layer composed of a number of neurons equal to the number of classes in each of the datasets considered.

To each of the clusters there was assigned either a single best classifier or a committee of five individual classifiers. The parameter values used while training AdaSS and the proposed modification are presented in Table 1.

Additionally, for comparison, the *clustering and selection* method was implemented, in which the k-means algorithm (MacQueen, 1967) was used as the clustering method and classifiers were selected from the same pool used by AdaSS and the proposed modification.

5.2. Datasets. The idea behind the choice of datasets was to examine the behavior of the selected methods on as diverse a set of benchmarks as possible. Therefore, we chose some high-dimensional sets, some very large sets with a small number of features, and also some typical/balanced ones, which allowed us to cover a wide range of real-life possibilities and make our tests more practically oriented. All datasets come from the UCI Machine Learning Repository (Frank and Asuncion, 2010).

Table 1. Parameters for the training phase.

Parameter	Values
H	1;3;5;7
N_c	200
N_p	100
β	0.7:0.3
γ	0.3:0.7
Δ_m	0.2

Table 2. Details of datasets used in the experimental investigation.

No.	Name	Objects	Features	Classes
1	Splice-junction Gene Sequences	3190	61	3
2	Ozone Level	2536	73	2
3	Vehicle Silhouettes	946	18	4
4	Letter Recognition	20000	16	26
5	Abalone	4177	8	28
6	Gamma Telescope	19020	11	2
7	Dermatology	366	33	6
8	SPECTF Heart	267	44	2
9	Promoter Gene Sequences	106	58	2
10	Audiology	226	69	23
11	Mammographic Mass	961	6	2
12	Musk (version 2)	6598	168	2
13	Internet Advertisements	3279	1558	2
14	Wine	178	13	3
15	Parkinsons	197	23	2
16	Horse Colic	368	27	11
17	Ecoli	336	8	8
18	Pima Indian Diabetes	768	8	2
19	Breast Cancer	286	9	2
20	Iris	150	4	3

The selected datasets with their respective names and numbers of objects, features, and classes are listed in Table 2.

5.3. Results. To facilitate the comparison, the accuracy of the individual classifiers is presented in Table 3, where 'No.' denotes the identification number for the dataset under consideration, and Cx denotes the classifier number x (e.g., C1 is the first classifier from the pool).

Table 4 presents comparative detailed experimental results for AdaSS and the proposed modification denoted as AD and MAD, respectively, tested on the 20 selected databases.

The combined 5×2 cv F test (Alpaydin, 1999) for statistical significance was carried out to compare the obtained results for the same size of the classifier committee only. If there is a statistical significance, the result of the better model is bolded.

Results of the experiments confirmed that the idea of adaptive clustering and selection is a promising direction for MCS design. For each dataset the compound classifiers outperformed the individual ones. What is worth stressing is that the proposed modification usually achieved better results than the original algorithm. It was also proved that statistically significant differences exist for several databases. The native version of the algorithm outperformed its modification only for one dataset (No. 13), although it should be mentioned that this dataset is characterized by a huge number of features and there was probably insufficient information for fuser training. Other dependencies were as expected. The quality of classification was improved according to the number of competence areas and the size of the committee, confirming what was stated by Jackowski and Woźniak (2009). For a few datasets only (Nos. 10, 11, 17, 19, and 20), we observed that the best results were achieved with $H = 5$, which could mean that overtraining occurs with higher numbers of competence areas.

Table 3. Accuracy of individual classifiers. No. denotes the dataset identifier according to Table 2.

No.	C1	C2	C3	C4	C5
1	63,124	63,124	61,457	60,290	54,850
2	61,005	61,005	61,005	57,900	56,900
3	81,260	81,260	81,260	78.500	77,900
4	82,035	80,000	80,000	78,500	75,250
5	84,980	82,180	80,500	76,500	72,000
6	81,240	81,240	81,240	75,250	72,500
7	82,950	82,000	81.000	80.000	77.300
8	71,370	70,200	68,950	65,500	65,000
9	52,950	52,950	50,150	50,000	50,000
10	48,670	48,670	48,670	47,300	46,500
11	62,450	62,000	61,600	60,000	56,400
12	73,405	73,405	71,600	70,405	69,810
13	57,500	56,360	55,000	53,689	52,689
14	90,125	90,125	90,125	90,125	88,500
15	85,090	85,090	83,200	82,200	78,800
16	77,605	74,605	72,000	69,100	68,605
17	77,890	77,890	77,890	77,890	76,000
18	60,350	59,200	57,450	53,000	53,000
19	88,450	88,450	88,450	84,150	84,150
20	92,750	92,750	92,750	92,250	91,500

An interesting conclusion is drawn from the comparative analysis of the CS algorithm and the

Table 4. Accuracy of AdaSS (AD) and its modification (MAD) for different values of H (number of clusters: 1, 3, 5, 7) and number of classifiers per cluster (1 or 5) compared with standard clustering and selection (CS).

No.	Mod	N	1	3	5	7	No.	Mod	N	1	3	5	7
1.	CS	–	63,124	64,350	66,650	68,000	11.	CS	–	62,450	66,130	68,000	68,000
	AD	1	63,124	68,658	68,956	69,300		AD	1	62,450	65,760	66,500	66,500
		5	65,950	69,100	70,005	70,005			5	62,450	66,100	66,100	67,200
	MAD	1	63,124	68,910	69,250	70,250		MAD	1	62,450	**68,012**	**69,350**	**68,785**
		5	65,250	**72,335**	**72,985**	**72,985**			5	62,700	**68,540**	**69,975**	**69,320**
2.	CS	–	61,005	68,020	70.500	70.000	12.	CS	–	73,405	75,080	75,080	78,110
	AD	1	61,005	69,120	72,500	72,500		AD	1	73,405	76,780	79,110	80,245
		5	62,765	70,300	74,000	74,000			5	73,405	77,125	80,015	80,540
	MAD	1	61,005	70,000	74,800	74,985		MAD	1	73,900	**80,305**	**84,050**	**85,900**
		5	**64,950**	**72,128**	**76,015**	**76,015**			5	73,900	**81,240**	**85,090**	**86,210**
3.	CS	–	81,260	88,500	88,500	89,225	13.	CS	–	57,500	59,500	63,170	63,170
	AD	1	81,260	88,500	88,500	89,225		AD	1	57,500	**64,120**	**66,670**	**67,040**
		5	81,430	89,125	89,125	90,005			5	56,240	**64,500**	**65,670**	**67,090**
	MAD	1	81,260	88,100	88,369	88,915		MAD	1	57,500	63,905	64,990	63,960
		5	87,430	89,015	89,050	89,050			5	55,689	65,140	63,905	63,120
4.	CS	–	82,035	86,250	88,750	88,750	14.	CS	–	90,125	93,450	93,450	92,680
	AD	1	82,035	90,125	91,068	91,068		AD	1	90,125	93,450	93,450	92,680
		5	85,005	90,125	91,068	91,068			5	90,125	93,450	93,450	92,680
	MAD	1	82,035	90,125	91,068	91,068		MAD	1	90,125	**94,800**	**95,425**	**95,425**
		5	85,760	90,125	91,068	91,068			5	90,125	**94,480**	**95,425**	**95,425**
5.	CS	–	84,980	86,975	88,013	89,250	15.	CS	–	85,090	90,000	90,000	92,090
	AD	1	84,980	86,975	87,005	89,013		AD	1	85,090	88,450	90,000	91,230
		5	85,345	87,390	88,012	89,500			5	85,560	89,180	90,500	91,230
	MAD	1	84,980	86,780	86,928	88,980		MAD	1	85,090	88,670	90,670	**91,289**
		5	85,500	87,200	88,450	89,250			5	**88,345**	91,500	92,005	**93,990**
6.	CS	–	81,240	84,234	86,120	88,500	16.	CS	–	77,605	82,000	82,000	80,430
	AD	1	81,240	84,234	86,120	88,500		AD	1	77,605	80,100	79,900	83,350
		5	82,300	85,015	86,950	89,170			5	77,340	78,450	79,115	82,430
	MAD	1	81,240	84,735	**88,730**	**90,005**		MAD	1	77,605	**82,900**	**83,500**	**85,190**
		5	**84,000**	86,005	**89,032**	**91,200**			5	77,340	**81,450**	**83,500**	**85,930**
7.	CS	–	82,950	89,600	92,440	94,009	17.	CS	–	77,890	78,125	76,211	74,980
	AD	1	82,950	89,600	92,440	94,009		AD	1	77,890	78,125	76,211	74,980
		5	85,012	90,450	92,440	95,989			5	78,600	80,216	79,350	79,350
	MAD	1	82,950	91,780	93,005	94,230		MAD	1	77,890	78,560	76,900	75,005
		5	84,900	90,345	94,218	96,560			5	77,005	79,560	79,560	78,945
8.	CS	–	71,370	73,430	76,454	76,454	18.	CS	–	60,350	64,230	64,230	64,230
	AD	1	71,370	75,530	78,908	78,908			1	60,350	64,230	64,230	67,600
		5	72,015	76,025	79,780	79,780			5	62,900	67,025	69,450	69,450
	MAD	1	71,370	**78,120**	**80,900**	**80,900**		MAD	1	60,350	**67,000**	**67,300**	**70,350**
		5	**75,050**	**80,300**	**81,000**	**82,450**			5	**64,000**	**69,300**	70,300	**72,010**
9.	CS	–	52,950	59,600	62,440	64,009	19.	CS	–	88,450	89,230	90,230	92,544
	AD	1	52,950	59,600	62,440	64,009		AD	1	88,450	90,560	92,340	91,500
		5	55,012	60,450	62,440	65,989			5	89,200	90,560	93,150	93,005
	MAD	1	52,950	**62,780**	**65,005**	**67,230**		MAD	1	88,450	89,911	92,544	92,020
		5	54,900	**63,345**	**66,218**	**68,560**			5	89,120	91,008	93,300	92,890
10.	CS	–	48,670	48,670	51,350	52,340	20.	CS	–	92,750	95,000	95,000	95,000
	AD	1	48,670	53,500	58,230	56,105		AD	1	92,750	95,000	96,005	94,250
		5	50,110	54,340	60,650	57,890			5	93,000	95,250	96,000	95,000
	MAD	1	48,670	**55,900**	**59,233**	**58,670**		MAD	1	92,750	95,000	95,750	94,00
		5	50,995	**59,110**	61,748	**60,157**			5	93,000	95,500	96,250	95,500

algorithms based on AdaSS. In most cases the proposed method outperformed the standard CS approach. This is due to the adaptive nature of the proposed method; the size of the clusters can be modified through the evolutionary process to take full advantage of the classifiers available in the pool. In the standard CS algorithm, the clustering step

is done independently of classifier selection, which is the main weakness of this approach. Only for four datasets (Nos. 3, 5, 19, and 20) did AdaSS not perform better than CS, but in all these cases it returned similar results.

6. Final remarks

This paper deals with compound pattern recognition algorithms based on the clustering and selection approach. A new method was derived from the AdaSS algorithm, but using a slightly different training method and a more sophisticated fusion block. The proposed approach was evaluated and compared with both the CS and native AdaSS algorithms through computer experiments. The experimental investigations were carried out on a wide range of benchmark datasets. The results confirmed the high quality of performance of the proposed modification, which mostly outperformed both the native algorithm and the standard CS approach. However, we must emphasize that the training time for the proposed method is significantly longer than the training phase for the native AdaSS. On the other hand, this undesirable feature of the algorithm is not an obstacle in its implementation in real decision problems because the classifier is trained only once. The decision time is the same for both AdaSS and its modification.

Acknowledgment

This work is supported by the Polish Ministry of Science and Higher Education under a grant for the years 2010–2013, and by the Polish National Science Center under a grant for the period 2011–2014.

References

Alpaydin, E. (1999). Combined 5 x 2 cv f test for comparing supervised classification learning algorithms, *Neural Computation* **11**(8): 1885–1892.

Alpaydin, E. (2010). *Introduction to Machine Learning*, 2nd Edn., The MIT Press, London.

Ashlock, D. (2006). *Evolutionary Computation for Modeling and Optimization*, 1st Edn., Springer, New York, NY.

Baram, Y. (1998). Partial classification: The benefit of deferred decision, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20**(8): 769–776.

Baruque, B., Porras, S. and Corchado, E. (2011). Hybrid classification ensemble using topology-preserving clustering, *New Generation Computing* **29**(3): 329–344.

Biggio, B., Fumera, G. and Roli, F. (2007). Bayesian analysis of linear combiners, *Proceedings of the 7th International Conference on Multiple Classifier Systems, MCS'07, Prague, Czech Republic*, pp. 292–301.

Brown, G. and Kuncheva, L.I. (2010). "Good" and "bad" diversity in majority vote ensembles, *9th International Workshop on Multiple Classifier Systems, MCS 2010, Cairo, Egypt*, pp. 124–133.

Chmaj, G., Walkowiak, K., Tarnawski, M. and Kucharzak, M. (2012). Heuristic algorithms for optimization of task allocation and result distribution in peer-to-peer computing systems, *International Journal of Applied Mathematics and Computer Science* **22**(3): 733–748, DOI: 10.2478/v10006-012-0055-0.

Chow, C.K. (1965). Statistical independence and threshold functions, *IEEE Transactions on Electronic Computers* **EC-14**(1): 66–68.

Cordella, L., Foggia, P., Sansone, C., Tortorella, F. and Vento, M. (2000). A cascaded multiple expert system for verification, *in* J. Kittler and F. Roli (Eds.), *Multiple Classifier Systems*, Lecture Notes in Computer Science, Vol. 1857, Springer, Berlin/Heidelberg, pp. 330–339.

Dietterich, T.G. and Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes, *Journal of Artificial Intelligence Research* **2**: 263–286.

Duda, R.O., Hart, P.E. and Stork, D.G. (2001). *Pattern Classification*, 2nd Edn., Wiley, New York, NY.

Duin, R. (2002). The combining classifier: To train or not to train?, *16th International Conference on Pattern Recognition, Quebec, Canada*, Vol. 2, pp. 765–770.

Frank, A. and Asuncion, A. (2010). UCI machine learning repository, `http://archive.ics.uci.edu/ml`.

Giacinto, G., Roli, F. and Fumera, G. (2000). Design of effective multiple classifier systems by clustering of classifiers, *15th International Conference on Pattern Recognition, Barcelona, Spain*, Vol. 2, pp. 160–163 .

Goebel, K. and Yan, W. (2004). Choosing classifiers for decision fusion, *Proceedings of the 7th International Conference on Information Fusion, Stockholm, Sweden*, pp. 563–568.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st Edn., Addison-Wesley Longman Publishing Co., Inc., Boston, MA.

Hansen, L. and Salamon, P. (1990). Neural network ensembles, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **12**(10): 993 –1001.

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

Jackowski, K. and Woźniak, M. (2009). Algorithm of designing compound recognition system on the basis of combining classifiers with simultaneous splitting feature space into competence areas, *Pattern Analysis and Applications* **12**(4): 415–425.

Jacobs, R.A. (1995). Methods for combining experts' probability assessments, *Neural Computation* **7**(5): 867–888.

Jacobs, R.A., Jordan, M.I., Nowlan, S.J. and Hinton, G.E. (1991). Adaptive mixtures of local experts, *Neural Computation* **3**(1): 79–87.

Jain, A., Duin, R. and Mao, J. (2000). Statistical pattern recognition: A review, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(1): 4–37.

Jain, A.K., Murty, M.N. and Flynn, P.J. (1999). Data clustering: A review, *ACM Computing Surveys* **31**(3): 264–323.

Kacprzak, T., Walkowiak, K. and Woźniak, M. (2012). Optimization of overlay distributed computing systems for multiple classifier system—Heuristic approach, *Logic Journal of the IGPL* **20**(4): 677–688.

Krawczyk, B. and Woźniak, M. (2011). Designing cost-sensitive ensemble genetic approach, *in* R. Choras (Ed.), *Image Processing and Communications Challenges 3*, Advances in Intelligent and Soft Computing, Vol. 102, Springer, Berlin/Heidelberg, pp. 227–234.

Kuncheva, L. (2000). Clustering-and-selection model for classifier combination, *4th International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies, Brighton, UK*, Vol. 1, pp. 185–188.

Kuncheva, L., Bezdek, J.C. and Duin, R.P.W. (2001). Decision templates for multiple classifier fusion: An experimental comparison, *Pattern Recognition* **34**(2): 299–314.

Kuncheva, L.I. (2004). *Combining Pattern Classifiers: Methods and Algorithms*, Wiley-Interscience, Hoboken, NJ.

Kuncheva, L., Whitaker, C., Shipp, C. and Duin, R. (2003). Limits on the majority vote accuracy in classifier fusion, *Pattern Analysis and Applications* **6**(1): 22–31.

Kuratowski, K. and Mostowski, A. (1976). *Set Theory: With An Introduction to Descriptive Set Theory*, 2nd Edn., North-Holland Pub. Co., Amsterdam.

MacQueen, J.B. (1967). Some methods for classification and analysis of multivariate observations, *in* L.M.L. Cam and J. Neyman (Eds.), *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1, University of California Press, Berkeley, CA, pp. 281–297.

Marcialis, G.L. and Roli, F. (2003). Fusion of face recognition algorithms for video-based surveillance systems, *in* G.L. Foresti, C.S. Regazzoni and P.K. Varshney (Eds.), *Multisensor Surveillance Systems: The Fusion Perspective*, Dordrecht, The Netherlands, pp. 235–250.

Matan, O. (1996). On voting ensembles of classifiers (extended abstract), *Proceedings of the AAAI-96 Workshop on Integrating Multiple Learned Models, Portland, OR, USA*, pp. 84–88.

Partridge, D. and Krzanowski, W. (1997). Software diversity: Practical statistics for its measurement and exploitation, *Information and Software Technology* **39**(10): 707–717.

Polikar, R. (2006). Ensemble based systems in decision making, *IEEE Circuits and Systems Magazine* **6**(3): 21–45.

Rastrigin, L. and Erenstein, R.H. (1981). *Method of Collective Recognition*, Energoizdat, Moscow.

Ruta, D. and Gabrys, B. (2005). Classifier selection for majority voting, *Information Fusion* **6**(1): 63–81.

Smetek, M. and Trawinski, B. (2011). Selection of heterogeneous fuzzy model ensembles using self-adaptive genetic algorithms, *New Generation Computing* **29**(3): 309–327.

Srinivas, M. and Patnaik, L.M. (1994). Genetic algorithms: A survey, *Computer* **27**(6): 17–26.

Team, R.D.C. (2008). *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna.

Ting, K., Wells, J., Tan, S., Teng, S. and Webb, G. (2011). Feature-subspace aggregating: Ensembles for stable and unstable learners, *Machine Learning* **82**(3): 375–397.

Troć, M. and Unold, O. (2010). Self-adaptation of parameters in a learning classifier system ensemble machine, *International Journal of Applied Mathematics and Computer Science* **20**(1): 157–174, DOI: 10.2478/v10006-010-0012-8.

Tumer, K. and Ghosh, J. (1996). Analysis of decision boundaries in linearly combined neural classifiers, *Pattern Recognition* **29**(2): 341–348.

van Erp, M., Vuurpijl, L. and Schomaker, L. (2002). An overview and comparison of voting methods for pattern recognition, *8th International Workshop on Frontiers in Handwriting Recognition, Ontario, Canada*, pp. 195–200.

Walkowiak, K. (2010). Anycasting in connection-oriented computer networks: Models, algorithms and results, *International Journal of Applied Mathematics and Computer Science* **20**(1): 207–220, DOI: 10.2478/v10006-010-0015-5.

Wilk, T. and Woźniak, M. (2011). Complexity and multithreaded implementation analysis of one class-classifiers fuzzy combiner, *in* E. Corchado, M. Kurzynski and M. Wozniak (Eds.), *Hybrid Artificial Intelligent Systems*, Lecture Notes in Computer Science, Vol. 6679, Springer, Berlin/Heidelberg, pp. 237–244.

Wolpert, D.H. (2001). The supervised learning no-free-lunch theorems, *6th Online World Conference on Soft Computing in Industrial Applications*, pp. 25–42.

Woods, K., Kegelmeyer Jr., W.P. and Bowyer, K. (1997). Combination of multiple classifiers using local accuracy estimates, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**(4): 405–410.

Woźniak, M. (2008). Experiments on linear combiners, *in* E. Pietka and J. Kawa (Eds.), *Information Technologies in Biomedicine*, Advances in Soft Computing, Vol. 47, Springer, Berlin/Heidelberg, pp. 445–452.

Woźniak, M. and Jackowski, K. (2009). Some remarks on chosen methods of classifier fusion based on weighted voting, *in* E. Corchado, X. Wu, E. Oja, A. Herrero and B. Baruque (Eds.), *Hybrid Artificial Intelligence Systems*, Lecture Notes in Computer Science, Vol. 5572, Springer, Berlin/Heidelberg, pp. 541–548.

Woźniak, M. and Zmyślony, M. (2010). Combining classifiers using trained fuser—Analytical and experimental results, *Neural Network World* **13**(7): 925–934.

Xu, L., Krzyzak, A. and Suen, C. (1992). Methods of combining multiple classifiers and their applications to handwriting recognition, *IEEE Transactions on Systems, Man and Cybernetics* **22**(3): 418–435.

Michał Woźniak is a professor of computer science at the Department of Systems and Computer Networks, Wrocław University of Technology, Poland. He received an M.Sc. degree in biomedical engineering from the Wrocław University of Technology in 1992, and Ph.D. and D.Sc. (habilitation) degrees in computer science in 1996 and 2007, respectively, from the same university. His research focuses on machine learning, distributed algorithms, and teleinformatics. Professor Woźniak has published over 160 papers and two books, and edited eight books. He has been involved in several research projects related to the above-mentioned topics and has been a consultant of several commercial projects for well-known Polish companies and public administration. Professor Woźniak is a senior member of the IEEE and a member of the International Biometric Society.

Bartosz Krawczyk received a B.Sc. engineering degree in computer science in 2011 and an M.Sc. degree with distinctions in 2012 from the Wrocław University of Technology, Poland. He was awarded as the best M.Sc. graduate by the the Rector of the Wrocław University of Technology. Mr. Krawczyk is currently a Ph.D. student in the Department of Systems and Computer Networks of the same university. His research is focused on machine learning, multiple classifier systems, one-class classifiers, class imbalance, and interdisciplinary applications of these methods. So far, he has published more than 30 papers in international journals and conferences. Mr Krawczyk is a chairman of the Symposium on *Advances in Artificial Intelligence and Applications* and a member of the Student Editorial Board for the *Methods of Information in Medicine* journal. He has already served as a member of program committees for over 20 international conferences and as a reviewer for over a dozen journals.

Received: 14 April 2012