A Weighted Majority Vote Strategy Using Bayesian Networks

Luigi P. Cordella[1], Claudio De Stefano[2], Francesco Fontanella[2], and Alessandra Scotto di Freca[2]

[1] Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione (DIETI)
Università di Napoli Federico II – Italy
cordel@unina.it
[2] Dipartimento di Ingegneria Elettrica e dell'Informazione (DIEI)
Università di Cassino e del Lazio Meridionale – Italy
{destefano,fontanella,a.scotto}@unicas.it

Abstract. Most of the methods for combining classifiers rely on the assumption that the experts to be combined make uncorrelated errors. Unfortunately, this theoretical assumption is not easy to satisfy in practical cases, thus effecting the performance obtainable by applying any combination strategy. We tried to solve this problem by explicitly modeling the dependencies among the experts through the estimation of the joint probability distributions among the outputs of the classifiers and the true class. In this paper we propose a new weighted majority vote rule, that uses the joint probabilities of each class as weights for combining classifier outputs. A Bayesian Network automatically infers the joint probability distribution for each class. The final decision is made by taking into account both the votes received by each class and the statistical behavior of the classifiers. The experimental results confirmed the effectiveness of the proposed method.

1 Introduction

Classifier combining techniques have been widely used for increasing classification performance in many applications [8,14]. A key issue in this framework is to ensure that the experts to be combined are sufficiently accurate so that the majority of them provide identical answers in case of correct classifications. At the same time, the experts should be sufficiently diverse so as to avoid that the majority of them provide identical answers in case of errors. Under this assumptions, it has been theoretically and experimentally demonstrated that the combination paradigm is more effective than the use of a powerful individual classifier [10].

It should be noted, however, that the theoretical assumption that classifiers to be combined make uncorrelated errors is not easy to be satisfied in practical cases. This is the reason why most of the approaches based on trained combiners assume correlated base classifiers, trying to characterize the dependencies among them through a suitable learning phase. On the contrary, combining methods using voting strategies, which are among the most widely adopted in

A. Petrosino (Ed.): ICIAP 2013, Part II, LNCS 8157, pp. 219–228, 2013.

the literature, assume the independence of classifiers, thereby neglecting possible dependence relationships among them [7,10]. The effect is that the performance of a combining method tends to be degraded and biased in case of highly dependent classifiers, since a correct classification provided by some of them may be overturned by the convergence of other classifiers on the same wrong decision.

In order to guarantee classifier independence, different approaches have been proposed in the literature: classifier diversity may be favored, for instance, either by using different classification schemes or different feature sets, or by using different data during the training. In this context, there has been a great scientific interest in designing multiple classifier systems based on the same classifier model trained on different data subsets or feature subsets. Such systems are commonly denoted as classifier *ensembles*. Simple and weighted majority vote rules are used in popular ensemble learning algorithms, such as Bagging [2] and Boosting [5], and have been successfully applied in many fields [11]. Nonetheless, even when the confidence measures used to weight classification results are jointly computed by evaluating the performance of the whole set of experts, the results may be not satisfactory: experimental studies on the classifier diversity obtained with bagging and boosting have shown that these techniques do not ensure to obtain sufficiently diverse classifiers [9]. In particular, as regards boosting, it has been observed that while at first steps highly diverse classifiers are obtained, as the boosting process proceeds, classifier diversity strongly decreases.

In order to overcome the classifier independence problem, in previous studies [3,4], we proposed to explicitly model the dependencies among the experts to be combined . To this aim, we estimated the joint probability distributions among the outputs of the classifiers and the true class, looking at the set of class labels provided by the classifiers for each sample of the training set. A Bayesian Network (BN) was used to learn such a joint probability. BN's were chosen because of both their effectiveness and the availability of learning tools, which allow us to describe correlations among classifier responses and true class label. Such correlations are then used to infer the probability distribution for each class. BN's were already used for combining classifiers [12] obtaining interesting results. However, the main drawback of these approaches is the need of large data sets in order to effectively estimate the joint probability distributions: in fact, when the set of class labels provided by the experts for an input sample represents a pattern not adequately learned by the probabilistic model, more classes may have similar joint probabilities, increasing the risk of misclassification. Unfortunately, adequately large training sets are not available in many real world applications.

Moving from these considerations, we propose a new combining rule which tries to solve the main drawbacks of the above mentioned techniques. More specifically, we have defined a new weighted majority vote rule, which associates a weight to each class rather than to each expert. The joint probabilities of each class with the set of responses provided by the experts are used as weights. The rationale of our approach is that of balancing the effects due to both the statistical dependencies among classifiers, and the inconsistencies in the probabilistic

model obtained by the BN. In fact, it may happen that the majority of experts provides the same wrong decision for an input sample, but the whole set of responses is a pattern adequately learned by the BN so that a high probability is assigned to the correct class and, obviously, low probabilities to the others. In these cases, weighting the votes of each class with the probability provided by the BN may overturn a possible wrong decision, favoring the choice of the correct class. On the contrary, when two or more classes have similar high probability values, the weighted majority vote rule tends to select among them the class that has received the highest number of votes. These choice appears more reliable than that based only on the probabilities provided by the BN. Finally, when the set of responses represents a pattern not learned by the BN, all the classes have similar low probability values and the final decision only depends on the votes received by the experts.

The results of the proposed approach have been compared with those obtained by four approaches. The first two are majority voting and weighted majority voting, using as weight the accuracy of the responses provided by the classifiers. The other two approaches are the well known algorithms for building classifier ensembles, i.e. bagging [2] and boosting [5]. Comparison results confirmed the effectiveness of the proposed approach.

The remainder of the paper is organized as follows: Section 2 describes the combining method and the properties of the BN, while Section 3 illustrates and discusses the experimental results. Finally, some conclusions are eventually left to Section 4.

2 The Combiner Architecture

Consider the responses $\mathbf{e} = \{e_1, \ldots, e_L\}$ provided by a set of L experts $E = \{E_1, \ldots, E_L\}$ for an input sample $\mathbf{x}$ in a N-class problem, and assume that such responses constitute the input to the combiner, as shown in Fig.1. Each expert E_i outputs the label of the class assigned to the sample $\mathbf{x}$, in the set $C = \{C_1, \ldots, C_N\}$, while the combiner provides the final classification result. In this stage the BN acts as a higher level classifier that works on a L-dimensional discrete-values feature space.

The proposed system operates in two different modalities. During the learning phase, a training set of samples is used for training the BN. In particular, a supervised learning strategy is adopted, which consists in observing both the responses $\mathbf{e}$, and the "true" class label for each sample of the training set. The Aim of the learning is to compute the joint probability distribution $p(c, e_1, \ldots, e_L)$, where c is the random variable representing the class to be estimated. In the operative phase, the combiner classifies unknown samples by using a weighted voting strategy. In particular, considering the set of responses $\mathbf{e}$ provided by the experts for an unknown sample $\mathbf{x}$, the combiner computes the class $\hat{c}$ of that sample by using the equation:

$$\hat{c} = \max_{1 \leq k \leq N} w_k(\mathbf{e}) \sum_{i=1}^{L} r_{i,k} \tag{1}$$

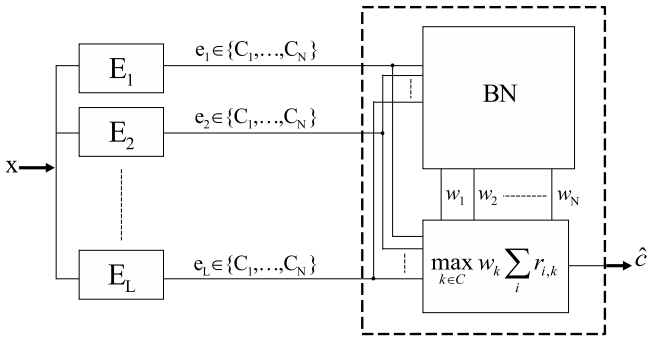

Fig. 1. The Combiner architecture

where w_k is the weight provided by the BN for the k–th class and $r_{i,k}$ is a function whose value is 1 when the classifier E_i classifies the sample $\mathbf{x}$ as belonging to the class C_k, and 0 otherwise. w_k represents the joint probability that the class is C_k when the set of responses provided by the experts is $\mathbf{e}$:

$$w_k(\mathbf{e}) = p\ (c = C_k, \mathbf{e}) \tag{2}$$

An high value of the weight w_k means that the set of responses $\mathbf{e}$ is very frequent in the training set for samples belonging to the class C_k.

Computing the joint probability of many variables is an hard task. As anticipated in the Introduction, in this study we used the Bayesian Network framework, because it offers an effective tool for learning from training data the joint probability distributions. Further details about BN's are presented in the following subsections.

2.1 Bayesian Network Properties

A BN makes it possible to represent the joint probability distribution of a set of random variables through the structure of a Direct Acyclic Graph (DAG). The nodes of the graph are the variables, while the arcs are their statistical dependencies. An arrow from the generic node i to node j means that j is conditionally dependent on i, and we can refer to i as a *parent* of j. For each node, a conditional probability quantifies the effect that the parents have on that node. Considering that a DAG structure describes the statistical dependencies among variables, the conditional probability distribution of a random variable e_i, given all the others, can be simplified as follows:

$$p\ (\ e_i\ |\ pa_{e_i}, np_{e_i}) = p\ (\ e_i\ |\ pa_{e_i}\ ) \tag{3}$$

where pa_{e_i} indicates the set of nodes which are parents of node e_i, and np_{e_i} indicates all the not parents nodes. Eq. (3), known as causal Markov property, allows all the variables corresponding to nodes in np_{e_i} to be discarded, since

such variables are not statistically dependent on e_i. After simple steps, the joint probability of a set of variables $\{c, e_1, \ldots, e_L\}$ can be expressed as:

$$p\,(c, e_1, \ldots, e_L) = p\,(\,c\,|\,pa_c\,) \prod_{e_i \in \mathbf{e}} p\,(\,e_i\,|\,pa_{e_i}\,) \tag{4}$$

In case of a node having no parents, the conditional probability coincides with the a priori probability of that node.

It is worth noticing that the node c may be parent of one or more nodes of the DAG. Therefore, it may be useful to divide the L nodes of the DAG in two groups: the first one, say G_c, contains the nodes having the node c among their parents, while the second one, say $G_{\overline{c}}$, contains the remaining nodes. With this assumption, Eq. (4) can be rewritten as:

$$p\,(c, e_1, \ldots, e_L) = p\,(c|pa_c) \prod_{e_i \in G_c} p\,(e_i|pa_{e_i}) \prod_{e_i \in G_{\overline{c}}} p\,(e_i|pa_{e_i}) \tag{5}$$

Note that the third term in Eq. (5) does not depend on c. This implies that, given a set of responses $\mathbf{e}$, in the computation of the probabilities $p\,(c = C_1, \mathbf{e})$, $\ldots$, $p\,(c = C_N, \mathbf{e})$ the third term in Eq. (5) is constant and thus can be omitted without affecting the results, since all the probabilities become scaled by the same factor. As a consequence, the weight w_k in Eq. (2) can be rewritten as:

$$w_k(\mathbf{e}) = p\,(c|pa_c) \prod_{e_i \in G_c} p(e_i|pa_{e_i}) \tag{6}$$

Hence, each weight w_k can be computed by considering only the Markov Blanket [6] of the node c in the DAG.

2.2 Bayesian Network Learning

Using a BN for combining classifier responses requires that both the network structure and the parameters of the probability distribution, be learned from a training set of examples. The aim of the structural learning is to find the statistical dependencies relations among the variables (graph nodes). It can be seen as an optimization problem, which requires the definition of a search strategy in the graph space in order to maximize a scoring function evaluating the effectiveness of candidate solutions. A scoring function may the posterior probability of the structure, given the training data. Let us assume that a set of N_D labeled samples is available. The training set used for the structural learning is obtained by considering, for each of the N_D samples, both the labels provided by the combining classifiers and the "true" label. Without loosing generality, let us also assume that DAG structures are equally likely. Under these conditions, denoting with S^h the DAG of a candidate BN and with D the training set having N_D elements, the scoring function to be maximized is the likelihood of D given the structure S^h: $score(S^h) = p(D|S^h)$.

According to the chain rule property of random variables in a DAG, this score can be factorized in the following way[1]:

$$score(S^h) = \prod_{i=1}^{L+1} localscore(i) \tag{7}$$

where the function $localscore(i)$ measures how much the variable e_i is statistically dependent on the set of its parent nodes pa_{e_i}, according to the examples of the training set D [6]. It is worth noticing that any change in S^h requires that only the local scores of the nodes affected by that change be updated for computing the new $Score(S^h)$. The function $localscore(i)$ is computed as follows:

$$localscore(i) = \prod_{m=1}^{S_{pa_{e_i}}} \frac{\Gamma(\alpha_{im})}{\Gamma(\alpha_{im} + N_{im})} \prod_{k=1}^{N} \frac{\Gamma(\alpha_{imk} + N_{imk})}{\Gamma(\alpha_{imk})}. \tag{8}$$

where $\Gamma(.)$ is the Euler Gamma function and $S_{pa_{e_i}}$ is the total number of states of pa_{e_i}. Considering that in our case each expert has N possible states, each corresponding to one of the classes in C, if the expert e_i has q parents, then $S_{pa_{e_i}} = N^q$. This implies that, for each response provided by the expert e_i, a vector of q terms representing the answers of the parent nodes of e_i must be analyzed. The term N_{imk} represents how many times pa_{e_i} is in the state m and the expert e_i is in the state k. The term N_{im}, instead, represents how many times pa_{e_i} is in the state m independently from the response provided by the expert e_i. The terms α_{im} and α_{imk} are normalization factors. Once the DAG structure S_h has been determined, the parameters of the conditional probability distributions are computed from D.

As regards the computational complexity of the function $localscore(i)$, its evaluation requires to compute the number of occurrences in D of each possible state of e_i associated to each possible state of pa_{e_i}. Thus, denoting again with q the number of parents of e_i, the computational complexity of $localscore(i)$ can be expressed as $\mathcal{O}(NS_{pa_{e_i}}N_D) = \mathcal{O}(N_D N^{q+1})$. In order to limit the exponential growth of the complexity of this function, we experimentally set to three the maximum number of parents that a node may have. Under these assumptions, the computational complexity of the function $score(S^h)$ (Eq. 7) is, in the worst case, $\mathcal{O}(LN_D N^4)$.

The exhaustive search for the BN structure which maximizes the scoring function is a NP-hard problem. For this reason, greedy algorithms are used to search for suboptimal solutions by maximizing a local scoring function at each step, which takes into account only the local topology of the DAG. In our case the adopted suboptimal structural learning algorithm is the one used in [3]. For the sake of clarity, it has been summarized in Fig.2. Note that the inserting an arc between two nodes, or inverting its direction, may produce a structure not satisfying the acyclicity property. Thus, *insert* and *invert* operations are performed only after checking that the acyclicity property is preserved.

[1] Note that if L is the number of considered classifiers, the DAG to be searched for consists of $L+1$ nodes, where the $(L+1)^{th}$ node represents the true class c.

generate at random a DAG structure S^h having $L+1$ nodes;
for $i = 0$ to L **do**
  compute $s[i] = localscore(i)$
compute $score(S^h)$
denote with G_0, G_1 and G_{-1} the local gains respectively obtained when an arc connecting two nodes is removed, inserted from the first to the second node, or in the opposite direction
do
  for $i = 0$ to $(L-1)$ **do**
    for $j = i+1$ to L **do**
      if (there is no arc between i and j) **then**
        $G_0 = s[j] \cdot s[i]$
        insert the arc $i \rightarrow j$ and **compute** $G_1 = localscore(j) \cdot s[i]$
        insert the arc $i \leftarrow j$ and **compute** $G_{-1} = localscore(i) \cdot s[j]$
      if (the arc $i \rightarrow j$ exists) **then**
        $G_1 = s[j] \cdot s[i]$
        remove the arc and **compute** $G_0 = localscore(j) \cdot s[i]$
        invert the arc and **compute** $G_{-1} = localscore(i) \cdot localscore(j)$
      if (the arc $i \leftarrow j$ exists) **then**
        $G_{-1} = s[j] \cdot s[i]$
        remove the arc and **compute** $G_0 = localscore(i) \cdot s[j]$
        invert the arc and **compute** $G_1 = localscore(i) \cdot localscore(j)$
      evaluate the maximun gain G and preserve the corresponding arc
      update $s[i]$ or/and $s[j]$ if needed
while (any $s[i]$ changes)

Fig. 2. The BN structural learning algorithm

3 Experimental Results and Discussion

The proposed method was tested on four real world datasets: Multiple Feature (Mfeat), Optical Recognition of Handwritten Digits (Opto), Pen-Based Recognition of Handwritten Digit (Pen) and Landsat Satellite (Sat). These datasets are publicly available from the UCI machine learning repository. The first three datasets describe images of handwritten digits, while the fourth one has been generated from Landsat multi-spectral scanner image data.

For each dataset, we have randomly extracted three equally sized sets of data: TR1, TR2 and TS. TR1 was used for training each single classifier, TR2 for the learning phase of the BN, while TS was used for the whole system performance evaluation. The BN was trained on data different from those used for training the classifiers in order to avoid biasing when it evaluates their responses in test phase.

In our experiments, we considered two different neural network architectures: Learning Vector Quantization (LVQ) and Multilayer Perceptron (MLP). The LVQ networks were trained by using the Frequency Sensitive Competitive

Learning algorithm (FSCL) [1], while the MLP nets were trained by using the Back Propagation algorithm. In order to induce diversity among the experts to be combined, different network parameters were used in the training phase. For the LVQ experts we varied the number of neurons in the output layer, while for the MLP ones the number of neurons in the hidden layer was changed. In practice, for each dataset, 50 classifiers $(e_1, e_2, \ldots, e_{50})$ were trained, using a different random initialization and a diverse number of neurons for each of them. As regards the LVQ nets, the number of neurons in the output layer was varied from 50 to 1 per class, while for the MLP nets, the number of hidden neurons was varied from 204 to 4, with step 4.

From the 50 experts of each pool, one every 10 was extracted to make up a further pool of 5 experts, and one every 5 to obtain a further pool of 10 experts. In this way, the diversity among the experts in each pool was maximized. After all, three pools of 50, 10 and 5 experts were obtained for each network architecture. These six pools were used for training and testing the proposed system, as discussed in Section 2. Since the learning procedure involves a random initialization phase, for each pool the results were averaged over 20 runs.

The average accuracy rates, obtained by the devised approach on the test set (TS), are shown in the last column of Table 1 (BNWMV header). In the same column, the standard deviations are also reported (in brackets). The table also shows the results achieved on the same test set by the four approaches taken into account for comparison, and by the BN combiner alone.

The first two comparing approaches are based on well known voting strategies: majority vote (MV header) and weighted majority vote (WMV header) [13]. The recognition rates obtained on the training set TR2 were used as weights for applying the weighted majority vote rule.

The other two considered approaches are the already quoted algorithms for building classifier ensembles: bagging and boosting. Also in this case, LVQ and MLP were used as base classifier architectures with the same parameter values adopted in the other considered methods, except for the number of neurons. In fact, since bagging and boosting algorithms have specific mechanisms to induce diversity, they require that the underlying architecture remain fixed. Hence, on the basis of preliminary trials, the number of MLP hidden neurons was set to 100, and the number LVQ output neurons was set to 10 per class. In every experiment, the results were averaged over 20 runs, each performed by randomly choosing the initial conditions. The standard deviations are also reported (in brackets) in the corresponding columns (Bag. and Boost. header, respectively).

Finally, as regards the classification results provided by the BN combiner (BN header), the maximun a-posteriori probability (MAP) rule was used: the class label provided in output by the combiner for each unknown input sample, is the one exhibiting the highest joint probability with the expert's responses. Also in this case, the standard deviations are reported (in brackets) in the corresponding column.

From Table 1 it can be seen that the BNWMV rule always obtains the best results. In particular, the results of our method are significantly better than

Table 1. Experimental results

Data	Alg.	Ens.	MV	WMV	Bag.	Boost.	BN	BNWMV
Mfeat	MLP	5	97.38	97.54	98.06 (0.04)	98.02 (0.11)	97.53 (0.33)	98.77 (0.00)
		10	97.54	97.85	98.12 (0.34)	98.02 (0.20)	97.35 (0.52)	98.92 (0.00)
		50	98.00	98.00	98.20 (0.11)	98.16 (0.08)	97.67 (0.73)	99.54 (0.00)
	LVQ	5	97.85	97.85	97.84 (0.86)	97.95 (0.34)	97.59 (0.12)	98.92 (0.00)
		10	97.38	97.54	98.09 (0.75)	97.98 (0.91)	97.18 (0.21)	99.54 (0.00)
		50	97.38	97.54	97.83 (0.98)	98.15 (0.61)	97.21 (0.18)	99.85 (0.00)
Opto	MLP	5	95.88	96.10	96.09 (0.08)	96.14 (0.14)	95.28 (0.13)	97.16 (0.00)
		10	95.83	95.88	96.07 (0.20)	96.10 (0.37)	95.21 (0.12)	97.59 (0.02)
		50	95.94	95.99	96.23 (0.11)	96.30 (0.44)	95.85 (0.20)	97.99 (0.05)
	LVQ	5	96.55	96.88	97.19 (0.50)	97.26 (0.43)	95.98 (0.22)	98.24 (0.04)
		10	96.88	96.99	97.61 (0.15)	97.72 (0.23)	95.51 (0.15)	99.09 (0.04)
		50	97.16	97.22	98.05 (0.35)	98.37 (0.55)	96.99 (0.20)	99.24 (0.07)
Pen	MLP	5	92.40	92.48	92.93 (0.40)	92.68 (0.09)	93.79 (0.06)	93.85 (0.00)
		10	92.40	92.48	93.91 (0.24)	93.88 (0.85)	95.93 (0.09)	97.55 (0.04)
		50	92.40	92.40	95.04 (0.18)	95.54 (1.10)	93.81 (0.11)	98.28 (0.01)
	LVQ	5	96.34	96.37	96.56 (0.05)	96.48 (0.14)	96.02 (0.09)	97.50 (0.11)
		10	96.68	96.88	96.89 (0.74)	96.79 (0.80)	96.42 (0.12)	98.15 (0.06)
		50	97.06	97.06	97.60 (0.25)	96.84 (0.88)	97.01 (0.04)	98.77 (0.02)
Sat	MLP	5	88.38	88.67	89.16 (0.90)	89.18 (0.65)	88.37 (1.34)	91.72 (0.06)
		10	89.14	89.52	90.56 (0.95)	90.53 (0.80)	88.45 (0.91)	93.22 (0.21)
		50	89.52	89.71	90.68 (0.55)	91.28 (1.20)	88.72 (0.47)	94.54 (0.08)
	LVQ	5	87.90	88.67	90.51 (0.82)	90.72 (0.61)	87.21 (0.95)	92.33 (0.21)
		10	87.90	88.95	91.90 (1.30)	92.04 (0.78)	87.36 (0.82)	94.90 (0.28)
		50	88.95	89.05	92.46 (0.82)	92.88 (0.25)	88.07 (0.96)	95.83 (0.19)

those provided by bagging and boosting, which are among the most effective and widely used methods in real world applications, and are often used as point of reference for evaluating classification performance. It is interesting to observe that, for all the datasets, LVQ networks always obtain better performances than MLP ones. This trend suggests that the adopted procedure for inducing expert diversity is more effective for the LVQ pools than for the MLP pools. It is also worth noticing that the results of BNWMV method exhibit very small standard deviations (values less than 10^{-3} have been approximated with 0.00), showing that the recognition rate varies very little with respect to the random starting point of the learning procedure. These values are significantly lower than those exhibited by the comparing methods, thus confirming the robustness of the proposed approach.

4 Conclusions

A new weighted majority vote rule, which associates a weight to each class rather than to each expert, is proposed. In this rule, the joint probabilities of each class with the set of responses provided by the experts are used as weights. Such joint probabilities are automatically inferred by suitably training a Bayesian Network.

The final decision is obtained by taking into account both the votes received by each class and the statistical behavior of the classifiers.

The experimental results confirm the effectiveness of the proposed method. In particular, the comparison among the results of our method and those obtained by separately using either the MV rule, or the BN combiner, shows that in each experiment our method always obtains better results, significantly increasing the recognition rate. This behavior provides experimental evidence that the basic idea of our approach, i.e. that of exploiting the strengths of both MV rule and BN combiner, trying to overcome at the same time their weaknesses, represents an effective solution for classifier combination and allowed us to obtain very promising results.

References

1. Ahalt, S.C., Krishnamurthy, A.K., Chen, P., Melton, D.E.: Competitive learning algorithms for vector quantization. Neural Netw. 3(3), 277–290 (1990)
2. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
3. De Stefano, C., D'Elia, C., Scotto di Freca, A., Marcelli, A.: Classifier combination by bayesian networks for handwriting recognition. Int. J. of Pattern Rec. and Artif. Intell. 23(5), 887–905 (2009)
4. De Stefano, C., Fontanella, F., Marrocco, C., di Freca, A.S.: A hybrid evolutionary algorithm for bayesian networks learning: An application to classifier combination. In: Di Chio, C., Cagnoni, S., Cotta, C., Ebner, M., Ekárt, A., Esparcia-Alcazar, A.I., Goh, C.-K., Merelo, J.J., Neri, F., Preuß, M., Togelius, J., Yannakakis, G.N. (eds.) EvoApplicatons 2010, Part I. LNCS, vol. 6024, pp. 221–230. Springer, Heidelberg (2010)
5. Freund, Y., Shapire, R.: Experiments with a new boosting algorithm. In: Proceedings of ICML 1996, pp. 148–156 (1996)
6. Heckerman, D.: A tutorial on learning with bayesian networks. Tech. rep., Learning in Graphical Models (1995)
7. Kang, H.J., Lee, S.W.: Combination of multiple classifiers by minimizing the upper bound of bayes error rate for unconstrained handwritten numeral recognition. Int. J. of Pattern Rec. and Artif. Intell. 19(3), 395–413 (2005)
8. Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.: On combining classifiers. IEEE Transactions on PAMI 20(3), 226–239 (1998)
9. Kuncheva, L., Skurichina, M., Duin, R.P.W.: An experimental study on diversity for bagging and boosting with linear classifiers. Information Fusion 3(4), 245–258 (2002)
10. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience (2004)
11. Oza, N., Tumer, K.: Classifier ensembles: Select real-world applications. Information Fusion 9(1), 4–20 (2008)
12. Sierra, B., Serrano, N., Larraaga, P., Plasencia, E.J., Inza, I., Jimnez, J.J., Revuelta, P., Mora, M.L.: Using bayesian networks in the construction of a bi-level multi-classifier. a case study using intensive care unit patients data. Artificial Intelligence in Medicine 22(3), 233–248 (2001)
13. Xu, L., Krzyzak, A., Suen, C.Y.: Methods of combining multiple classifiers and their applications to handwriting recognition. IEEE Trans. on Systems, Man, and Cybernetics 22(3), 418–435 (1992)
14. Zhou, Z., Wu, J., Tang, W.: Ensembling neural networks: many could be better than all. Artificial Intelligence 137(1-2), 239–263 (2002)