ELSEVIER

Pattern Recognition Letters 23 (2002) 593–600

Pattern Recognition Letters

www.elsevier.com/locate/patrec

Generating classifier outputs of fixed accuracy and diversity

Ludmila I. Kuncheva *, Roumen K. Kountchev

School of Informatics, University of Wales Bangor, Dean Street, Bangor, Gwynedd LL57 1UT, UK

Received 6 March 2001; received in revised form 30 July 2001

Abstract

We offer an algorithm for random generation of classifier outputs with specified individual accuracies and pairwise dependencies. The outputs are binary vectors (correct/incorrect classification) for a hypothetical data set. The generated team output can be used to study the majority vote over multiple dependent classifiers.

Keywords: Multiple classifier systems; Ensemble diversity; Dependency; Q statistic; Output generating algorithm

1. Introduction

Combining classifiers is now an established research area known under different names in the literature: committees of learners, mixtures of experts, classifier ensembles, multiple classifier systems, consensus theory, etc. (Ng and Abramson, 1992; Xu et al., 1992).

Majority vote is a popular method for combining classifier outputs and has been studied for the case of independent classifiers (Battiti and Colla, 1994; Lam and Krzyzak, 1994; Lam and Suen, 1997; Xu et al., 1992).

Let $\mathscr{D} = \{D_1, \ldots, D_L\}$ be a team (set, committee, mixture, pool, ensemble) of classifiers such that $D_i : \Re^n \to \Omega$, where $\Omega = \{\omega_1, \ldots, \omega_c\}$ is a set of class labels. For a given input $\boldsymbol{x} \in \Re^n$, the majority vote assigns $\boldsymbol{x}$ the class label supported by the majority of the classifiers D_i.

Finding independent classifiers is one aim in combining classifiers because, theoretically, the majority vote accuracy P_{maj} over independent classifiers, each of accuracy $p > 0.5$, exceeds p. More interestingly, when dependent classifiers are combined, P_{maj} can be higher than p, and also higher than the accuracy of an independent team. However, dependent classifiers may also render $P_{\text{maj}} < p$ (Kuncheva et al., submitted; Kuncheva et al., 2000). One question that is generally not answered yet is how P_{maj} will be affected if classifier accuracies differ by a small fraction or when the pairwise dependencies between classifiers are different. For example, suppose we are interested in the behavior of majority vote of $L = 5$ classifiers when degrading the individual performances of three of them but keeping the diversity of the team the same. This task is hardly feasible if we have to build the classifiers on real-life data. Therefore, for

* Corresponding author. Tel.:+44-1248-383661; fax: +44-1248-361429.

E-mail address: l.i.kuncheva@bangor.ac.uk (L.I. Kuncheva).

0167-8655/02/$ - see front matter
PII: S0167-8655(01)00155-6

studying empirically the relationship between diversity and majority vote, a simulation routine is needed where we specify the target individual accuracies and pairwise relationships and produce the classifier outputs.

In this paper we offer an algorithm for generating randomly a set of L classifier outputs over a hypothetical data set of N elements. The Q statistic for pairwise dependency is introduced in Section 2. The formulas for generating two classifiers of specified accuracies and dependency between them are derived in Section 3. The procedure for L classifiers is discussed in Section 4 and some results are given in Section 5.

2. The Q statistic for pairwise dependency

Let $\boldsymbol{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\}$ be a labeled data set, $\boldsymbol{z}_j \in \Re^n$ coming from the classification problem in question. We can represent the output of any classifier D_i as an N-dimensional binary vector $\boldsymbol{y}_i = [y_{1,i}, \ldots, y_{N,i}]^{\mathrm{T}}$ of *correct classification*, such that $y_{j,i} = 1$, if D_i recognizes correctly $\boldsymbol{z}_j$, and 0, otherwise, $i = 1, \ldots, L$.

There are various statistics to assess the similarity of two classifier outputs $\boldsymbol{y}_i$ and $\boldsymbol{y}_k$ (Sneath and Sokal, 1973). The Q statistic (Yule, 1900) for two classifiers is

$$Q_{i,k} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}}, \tag{1}$$

where N^{ab} is the number of elements $\boldsymbol{z}_j$ of $\boldsymbol{Z}$ for which $y_{j,i} = a$ and $y_{j,k} = b$ (see Table 1).

For statistically *independent* classifiers (and $N \to \infty$), $Q_{i,k} = 0$. Q varies between -1 and 1. Classifiers that tend to recognize *the same* objects correctly will have positive values of Q, and those which commit errors on different objects will render Q negative.

Shown below are four 2×2 tables with the respective Q's and individual accuracies $\hat{p}_1 = \hat{p}_2 = p$ for $N = 100$ objects.

$Q = -1$		$Q = -0.5$		$Q = 0$		$Q = 1$	
0	50	30	30	36	24	50	0
50	0	30	10	24	16	0	50
$p = 0.5$		$p = 0.6$		$p = 0.6$		$p = 0.5$	

Our previous studies (Kuncheva and Whitaker, 2001) picked the Q statistic from a set of 10 measures of classifier diversity for the following reasons: Q depends to a less degree on the individual accuracies than the other 9 measures do, and Q has a specific value 0 for statistically independent classifiers. A formal relationship between Q and the limits of the majority vote has been shown in (Kuncheva et al., submitted).

The aim is to generate L binary vectors of length N, $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_L$. The input is the number of data points N, a vector with desired individual accuracies $\boldsymbol{p} = [p_1, \ldots, p_L]^{\mathrm{T}}$, and a symmetric matrix $Q = [Q_{i,k}]$, where $Q_{i,k}$ is the desired dependency between classifiers D_i and D_k, $i, k \in \{1, \ldots, L\}$.

Generating *independent* classifiers of accuracies in $\boldsymbol{p}$ is easy. However, when a certain dependency has to be modeled, the generation is not trivial. Below we offer a solution to this problem without claiming that this is the best or the only possible one.

3. The generating algorithm for two classifiers

Consider first *two classifiers*, D_i and D_k, and their respective output vectors $\boldsymbol{y}_i$ and $\boldsymbol{y}_k$.

Let D_i have accuracy A, so that approximately $N \times A$ elements of $\boldsymbol{y}_i$ are 1s and $N \times (1 - A)$ elements are 0s. Assume that we alternate each

Table 1
A 2×2 table of the relationship between a pair of classifiers

	D_k correct (1)	D_k wrong (0)
D_i correct (1)	N^{11}	N^{10}
D_i wrong (0)	N^{01}	N^{00}

Total, $N = N^{00} + N^{01} + N^{10} + N^{11}$.

element 1 of $\boldsymbol{y}_i$ with probability P_1 and each element 0 with probability P_2. The result will be a new output vector, say, $\boldsymbol{y}_k$, whose accuracy A_{new} can be calculated from A, P_1, P_2, and N. The number of ones in $\boldsymbol{y}_k$ will be those that did not change, approximately $N \times A \times (1-P_1)$, and also those coming from previous zeros, i.e., approximately $N \times (1-A) \times P_2$. Thus, the new accuracy will be

$$\begin{aligned} A_{\text{new}} &= \frac{1}{N}(NA(1-P_1) + N(1-A)P_2) \\ &= A(1-P_1) + (1-A)P_2. \end{aligned} \tag{2}$$

Example. Consider a binary output vector $\boldsymbol{y}_i = [0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 1]^{\mathrm{T}}$ of size 10, coming from classifier D_i with accuracy $A = 0.6$. If $P_1 = 0$ and $P_2 = 0$, then the new output $\boldsymbol{y}_k$ that we produce using $\boldsymbol{y}_i$ will be identical to $\boldsymbol{y}_i$. Now assume that $P_1 = 0.3$ and $P_2 = 0.1$. To form the new vector, $\boldsymbol{y}_k$, we generate 10 random numbers, $r_1, \ldots, r_{10}$, one for each component of $\boldsymbol{y}_i$. The jth component of $\boldsymbol{y}_k$ is obtained by comparing r_j with P_1 if $y_{j,i}$ is 1, and with P_2 if $y_{j,i}$ is 0. If r_j is greater than the respective probability, then we rewrite the jth bit in the new vector. Otherwise, we alternate it. The table below shows an example of producing $\boldsymbol{y}_k$ from the above $\boldsymbol{y}_i$

$\boldsymbol{y}_i$	=	0	0	1	0	1	1	0	1	1	1
r_j's		0.30	0.54	0.15	0.70	0.38	0.86	0.85	0.59	0.50	0.90
$\boldsymbol{y}_k$	=	0	0	0	0	1	1	0	1	1	1

The only bit that changed was $y_{3,i}$. The (predicted) new accuracy, (2), is

$$A_{\text{new}} = 0.6 \times (1-0.3) + (1-0.6) \times 0.1 = 0.46.$$

Indeed, the observed value of A_{new} from this example is 0.5. We need two values, P_1 and P_2, so that not only the new accuracy is targeted but also the dependency between the outputs of the basic classifier and the new one.

We need P_1 and P_2 so that D_k is derived by this alternating procedure from D_i, and the desired accuracies are $p_i = A$ and $p_k = A_{\text{new}}$ while also fixing the desired $Q_{i,k}$. Noticing that approximately:

$$\begin{aligned} N^{11} &= NA(1-P_1), \\ N^{10} &= NAP_1, \\ N^{01} &= N(1-A)P_2, \\ N^{00} &= NAP_2, \end{aligned} \tag{3}$$

and applying (1), we obtain through algebraic manipulations

$$\begin{aligned} Q_{i,k} &= \frac{(1-P_1)(1-P_2) - P_1P_2}{(1-P_1)(1-P_2) + P_1P_2} \\ &= \frac{1-P_1-P_2}{1-P_1-P_2+2P_1P_2}. \end{aligned} \tag{4}$$

Substituting $p_i = A$ and $p_k = A_{\text{new}}$ in (2) and solving simultaneously (2) and (4) for P_1 and P_2, we obtain

$$\begin{aligned} P_1 &= 1 - P_2 - \frac{p_k}{p_i} + \frac{P_2}{p_i}, \\ P_2 &= \frac{-(1-Q_{i,k}+2Q_{i,k}(p_i-p_k)) \pm \sqrt{\text{Discr}}}{4Q_{i,k}(1-p_i)}, \end{aligned} \tag{5}$$

where

$$\begin{aligned} \text{Discr} = {} & (1-Q_{i,k}+2Q_{i,k}(p_i-p_k))^2 \\ & - 8Q_{i,k}(1-p_i)p_k(Q_{i,k}-1). \end{aligned}$$

For $Q_{i,k} = 0$ (independent D_i and D_k), $P_1 = 1 - A_{\text{new}}$ and $P_2 = A_{\text{new}}$.

Using the above equations, 50 pairs of classifiers D_i, D_k were generated with desired values $p_i = 0.6$, $p_k = 0.8$, $Q_{i,k} = -0.3$, and $N = 1000$. The mean and the standard deviations found through the experiment are

$$\hat{p}_i = 0.6024 \pm 0.0136, \quad \hat{p}_k = 0.7991 \pm 0.0131,$$
$$\hat{Q} = -0.2946 \pm 0.0803.$$

It is important to keep in mind that the derived relationships are only approximate. The accuracy of the approximation will depend on the chosen sample size N. Small values of N might lead to spurious results. At a glance, the size of N might not look to be too much of a problem. As we are

1. Input $\mathbf{p}$ (of size L), the matrix Q (of size $L \times L$), and N (number of data points).
2. Calculate $P_1(i,k)$ and $P_2(i,k)$ for all $i,k = 1,\ldots,L,\ i \neq k$ using (5).
3. For $j = 1 : N$,
   (a) Choose a random permutation $\{i_1,\ldots,i_L\}$ of $\{1,2,\ldots,L\}$.
   (b) Set the jth output of classifier D_{i_1}, y_{j,i_1}, to 1 with probability p_{i_1}, and to 0, otherwise.
   (c) For $t = 2 : L$,
      i. Using $D_{i_{t-1}}$ as the basic classifier, set the jth output of D_{i_t}, y_{j,i_t}, according to the probabilities $P_1(i_{t-1},i_t)$ or $P_2(i_{t-1},i_t)$.
      ii. End t
   (d) End j.
4. Return $\mathbf{y}_1,\ldots,\mathbf{y}_L$.

Fig. 1. A pseudo-code for generating the array with the classifier outputs from the vector of accuracies $\boldsymbol{p}$ and the matrix of pairwise dependencies Q.

generating the data, we can afford to generate a series of ensembles, and then judge and pick the closest approximation to our target values. The argument against this is that sampling from real data will hardly produce small sets of classifier outputs with an exact representation of the accuracy and dependency. Thus, our (small) data set with perfect representation of the target values will not be an adequate model of a real-life problem of the same size. It is recommended therefore to use large values of N.

4. The generating algorithm for L classifiers

When more than two classifiers are needed, the generation is not straightforward. The random alternating of the output of one classifier (called the *basic* classifier) to get the output of another classifier (called the *subsequent* classifier) has to be "shared" between all L classifiers. If we first generate D_i and use it to obtain D_k through the alternating procedure described in Section 3, and then use either of D_i or D_k to produce another classifier D_l, there is no guarantee that the classifier left aside and D_l will have the desired Q value. The idea proposed here only partly solves the problem. First, for each pair of classifiers $D_i, D_k \in \mathscr{D}$, we calculate $P_1(i,k)$ and $P_2(i,k)$ using (5). [1] Next, for each data point $\boldsymbol{z}_j \in \boldsymbol{Z}$, a random permutation of the integers from 1 to L is generated. It is used to pick the order in which the classifiers will be selected as the *basic* and the *subsequent* ones. For example, take the permutation $(3,5,2,1,4)$ for 5 classifiers. First, classifier D_3 is nominated as the basic one and D_5 as the subsequent one. The output value of D_3 for $\boldsymbol{z}_j$ is generated randomly: 1 with probability p_3 or 0 with probability $1 - p_3$. Next, D_3 is used to set the output value for D_5 (for

[1] These are P_1 and P_2 from the two-classifier case. They now need to be indexed with respect to the two classifiers, as we consider a pool of L classifiers with $L(L-1)/2$ possible pairs.

this particular $\boldsymbol{z}_j$) using the probabilities in $P_1(3,5)$ or $P_2(3,5)$. Further, D_5 is taken to be the basic classifier and D_2 is picked as the subsequent one. Using $P_1(5,2)$ or $P_2(5,2)$ we set the value of D_2. The final pair for this example will be D_1 and D_4. This generation process is repeated N times, so that L binary column vectors of length N are obtained. The random nomination of the basic classifiers spreads the "responsibility" over the whole $\mathscr{D}$ assuming that eventually every two classifiers will have enough entries in the respective output vectors obtained from one another, so that the desired Q between them is approached.

The pseudo-code for the algorithm is shown in Fig. 1.

5. Simulation results

Case 1. The proposed algorithm was run 100 times with $L = 3$ classifiers and input parameters as follows: $Q_{i,k} = b\ \forall i, k = 1,2,3, i \neq k$, and $(Q_{i,i} = 1)$, where $b \in \{-1, -0.9, \ldots, 1\}$; and $\boldsymbol{p} = [a,a,a]^{\mathrm{T}}$, for $a \in \{0.6, 0.7, 0.8, 0.9\}$. Thus 21×4 tables were constructed and the results in each cell were calculated as an average of 100 experiments (total of 8400 generations of data sets). In all experiments, the data set generated contained $N = 200$ points.

The means and the standard deviations of the individual accuracies $\hat{a}$ obtained from the generated sets of classifier output, regardless of the target Q, are:

$$p = 0.6, \quad \hat{p} = 0.600\ (\pm 0.021),$$

$$p = 0.7, \quad \hat{p} = 0.701\ (\pm 0.020),$$

$$p = 0.8, \quad \hat{p} = 0.800\ (\pm 0.018),$$

$$p = 0.9, \quad \hat{p} = 0.900\ (\pm 0.013).$$

Fig. 2 shows the deviations of the individual accuracies from the target value, $\hat{p} - p_{\text{target}}$, for target

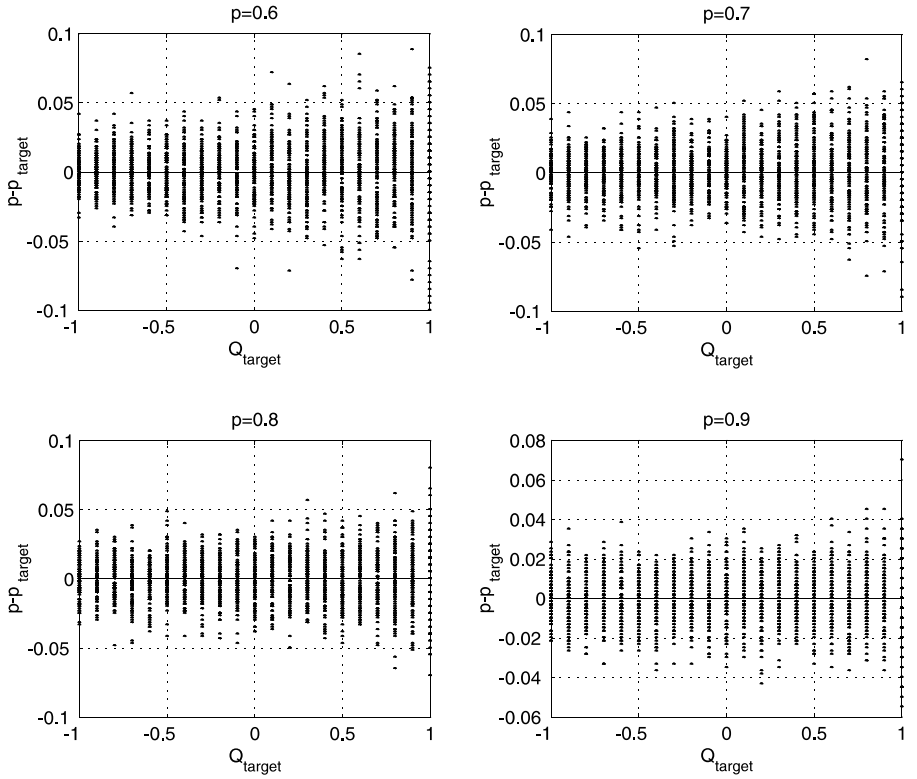

Fig. 2. Deviations of the individual accuracies from the target values 0.6, 0.7, 0.8 and 0.9 versus the target values of the diversity Q.

values of diversity $Q_{\text{target}} \in \{-1.0, \ldots, 1.0\}$. The figure shows that the individual accuracies are slightly less precise toward bigger values of the Q_{target}.

Shown in Table 2 are the averaged pairwise $\hat{Q}$'s, and their standard deviations for the generated data.

Plotted in Fig. 3 are the deviations of the generated Q's from the target values. If the generation was perfect, all the points should have been on the diagonal line. However, as the graphs show, the generation is biased. First, higher values of Q are more closely modeled than lower values. Second, the standard deviations in Table 2 and the graphical output in Fig. 3 indicate that the variability of $\hat{Q}$ increases with p.

The imprecision of the generating algorithm is unpleasant but is not a serious problem. If we want to generate the outputs of, say, M, classifier teams with a prespecified Q, we can use the algorithm as it is and accept only those teams whose Q and p are sufficiently close to the desired values. The algorithms should be run until we reach the number M of such teams. As the figure shows, it will not always be possible to keep both Q and p precise and generate *all* values of Q at the same time. The time needed to generate M teams will depend on the assignment of the target parameters. Values of Q close to 0 will be easily reached whereas values close to -1 might need a lot of time. Some modifications of the code led us to more accurate values for Q but this destroyed severely the precision of p and therefore these modifications were dismissed.

With bigger number of data points N, the variance of the estimate decreased but the Q values were similarly biased.

Case 2. While in Case 1 the classifiers were of the same accuracy and dependency, in Case 2 different values were tried. The proposed algorithm was run 100 times with input parameters

$$\boldsymbol{p}_{\text{target}} = [0.6, 0.9, 0.8, 0.7]^{\mathrm{T}},$$

$$Q_{\text{target}} = \begin{bmatrix} 1 & -0.2 & 0.5 & -0.6 \\ -0.2 & 1 & -0.2 & 0.1 \\ 0.5 & -0.2 & 1 & -0.4 \\ -0.6 & 0.1 & -0.4 & 1 \end{bmatrix}, \tag{6}$$

$$N = 1000.$$

Table 2
Means and standard deviations of the pairwise $\hat{Q}$'s of the generated classifier outputs

Target $Q(b)$	$p(a)$			
	0.6	0.7	0.8	0.9
−1.0	−0.579 (±0.032)	−0.550 (±0.059)	−0.541 (±0.115)	−0.651 (±0.234)
−0.9	−0.526 (±0.042)	−0.497 (±0.062)	−0.501 (±0.107)	−0.562 (±0.245)
−0.8	−0.460 (±0.051)	−0.444 (±0.069)	−0.437 (±0.116)	−0.521 (±0.236)
−0.7	−0.409 (±0.053)	−0.407 (±0.075)	−0.417 (±0.118)	−0.462 (±0.254)
−0.6	−0.342 (±0.069)	−0.361 (±0.078)	−0.359 (±0.111)	−0.464 (±0.232)
−0.5	−0.304 (±0.067)	−0.304 (±0.085)	−0.323 (±0.114)	−0.408 (±0.251)
−0.4	−0.249 (±0.058)	−0.248 (±0.094)	−0.271 (±0.123)	−0.300 (±0.257)
−0.3	−0.178 (±0.080)	−0.183 (±0.089)	−0.209 (±0.129)	−0.283 (±0.262)
−0.2	−0.129 (±0.073)	−0.140 (±0.079)	−0.139 (±0.133)	−0.225 (±0.262)
−0.1	−0.062 (±0.078)	−0.082 (±0.092)	−0.074 (±0.125)	−0.187 (±0.258)
0.0	0.006 (±0.088)	0.005 (±0.102)	−0.025 (±0.134)	−0.149 (±0.278)
0.1	0.080 (±0.078)	0.056 (±0.097)	0.048 (±0.130)	−0.061 (±0.252)
0.2	0.147 (±0.094)	0.121 (±0.103)	0.118 (±0.136)	0.075 (±0.225)
0.3	0.220 (±0.091)	0.211 (±0.099)	0.197 (±0.131)	0.154 (±0.208)
0.4	0.291 (±0.091)	0.300 (±0.104)	0.302 (±0.120)	0.209 (±0.217)
0.5	0.381 (±0.085)	0.375 (±0.093)	0.364 (±0.120)	0.310 (±0.194)
0.6	0.475 (±0.083)	0.480 (±0.077)	0.488 (±0.108)	0.419 (±0.174)
0.7	0.606 (±0.076)	0.581 (±0.082)	0.591 (±0.087)	0.563 (±0.134)
0.8	0.706 (±0.057)	0.689 (±0.067)	0.702 (±0.075)	0.677 (±0.112)
0.9	0.846 (±0.040)	0.844 (±0.040)	0.829 (±0.046)	0.828 (±0.075)
1.0	1.000 (±0.000)	1.000 (±0.000)	1.000 (±0.000)	1.000 (±0.000)

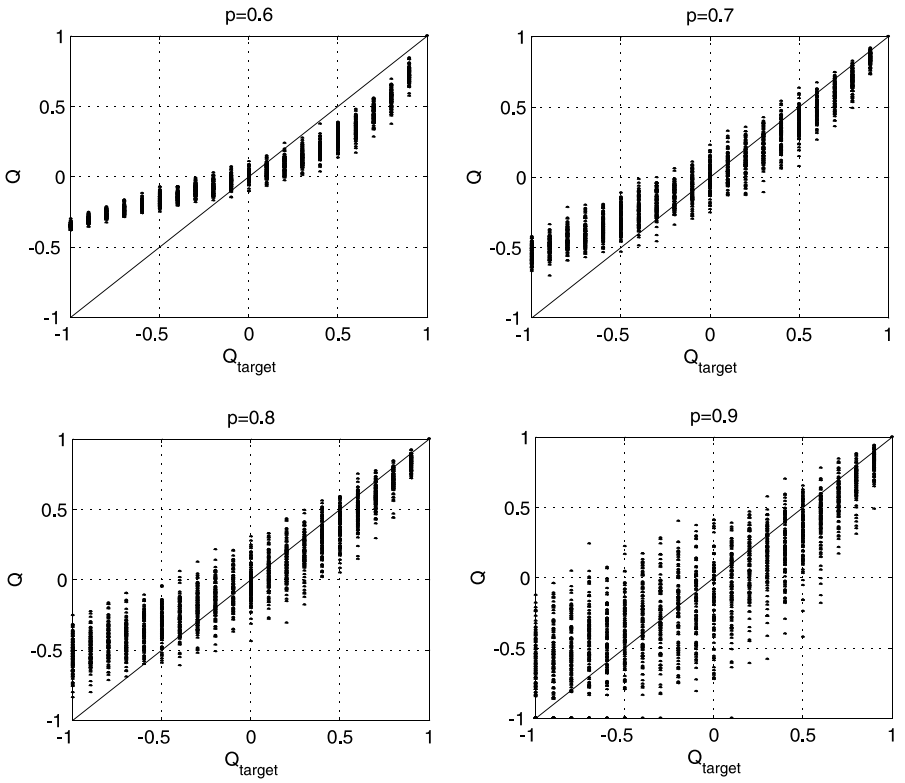

Fig. 3. Deviations of the diversity $\hat{Q}$ from the target values Q_{target} for individual accuracies $a \in \{0.6, 0.7, 0.8, 0.9\}$.

Table 3
The diversity matrices: the averaged over 100 experiments ($\hat{Q}$), the best matrix found ($\hat{Q}^*$) and the target (Q_{target})

Averaged ($\hat{Q}$)				Best ($\hat{Q}^*$)				Target (Q_{target})			
1.00	−0.09	0.28	−0.32	1.00	−0.17	0.40	−0.40	1.00	−0.20	0.50	−0.60
−0.09	1.00	−0.13	0.07	−0.17	1.00	−0.08	0.09	−0.20	1.00	−0.20	0.10
0.28	−0.13	1.00	−0.22	0.40	−0.08	1.00	−0.29	0.50	−0.20	1.00	−0.40
−0.32	0.07	−0.22	1.00	−0.40	0.09	−0.29	1.00	−0.60	0.10	−0.40	1.00

The means and the standard deviations of the individual accuracies are as follows:

$$\hat{\boldsymbol{p}} = [0.602, 0.900, 0.801, 0.701]^{\mathrm{T}},$$

$$\text{std} = [0.015, 0.009, 0.012, 0.012]^{\mathrm{T}}.$$

For each of the 100 classifier teams, we calculated the squared error between the entries in the target matrix Q_{target} and the obtained $\hat{Q}$. The smallest value of the squared error (0.1537) was identified and the respective classifier team was taken as the best approximation. Table 3 shows the averaged matrix $\hat{Q}$, the best matrix, $\hat{Q}^*$ and Q_{target} for comparison. The individual accuracies for the best team are $\hat{\boldsymbol{p}} = [0.635, 0.902, 0.806, 0.700]^{\mathrm{T}}$.

The accuracy of the approximated $Q_{i,k}$ depends on how many times D_i and D_k have been picked as the pair (*basic* and *subsequent*) classifiers. For each point in the data set, L of the possible $L(L-1)/2$ pairwise relationships is being fixed. With the proposed algorithm, the remaining relationships tend

to "drag" the team toward independence (Q values closer to 0). Therefore, for smaller L we could expect more accurate estimates of $Q_{i,k}$ than for larger L. Designing a probabilistic model similar to that for the case of 2 classifiers is not straightforward when $L > 2$. As mentioned before, a natural option to overcome the imperfect generation will be to pick only those teams whose parameters are close to the targets and dismiss the rest.

6. Conclusions

This paper proposes an algorithm for generating L classifier outputs for a hypothetical data set of N elements. The outputs are binary and indicate correct/incorrect classification. We derive formulas according to which two classifiers can be generated with specified accuracies and dependency Q between them. Next, an algorithm for generating multiple classifiers is proposed based on the formulas. The degree of dependency of the generated classifier outputs is smaller by absolute value than the prespecified one which is a flaw of the generation strategy. A selection procedure can be applied to remedy this. The algorithm can be used for simulation of classifier outputs in studies of the majority vote accuracy of a pool of *dependent* classifiers.

References

Battiti, R., Colla, A.M., 1994. Democracy in neural nets: Voting schemes for classification. Neural Networks 7, 691–707.

Kuncheva, L.I., Whitaker, C.J., 2001. Ten measures of diversity in classifier ensembles: limits for two classifiers. In: Proc. IEE Workshop on Intelligent Sensor Process., Birmingham. IEE, London, pp. 10/1–10/6.

Kuncheva, L.I., Whitaker, C.J., Shipp, C.A., Duin, R.P.W., Limits on the majority vote accuracy in classifier fusion (submitted).

Kuncheva, L.I., Whitaker, C.J., Shipp, C.A., Duin, R.P.W., 2000. Is independence good for combining classifiers? In: Proc. 15th Internat. Conf. on Pattern Recognition, Barcelona, Spain, Vol. 2, pp. 169–171.

Lam, L., Krzyzak, A., 1994. A theoretical analysis of the application of majority voting to pattern recognition. In: Proc. 12th Internat. Conf. on Pattern Recognition, Jerusalem, Israel, pp. 418–420.

Lam, L., Suen, C.Y., 1997. Application of majority voting to pattern recognition: an analysis of its behavior and performance. IEEE Trans. Syst., Man Cybern. 27 (5), 553–568.

Ng, K.-C., Abramson, B., 1992. Consensus diagnosis: a simulation study. IEEE Trans. Syst., Man Cybern. 22, 916–928.

Sneath, P.H.A., Sokal, R.R., 1973. In: Numerical Taxonomy. Freeman, New York, p. 1973.

Xu, L., Krzyzak, A., Suen, C.Y., 1992. Methods of combining multiple classifiers and their application to handwriting recognition. IEEE Trans. Syst., Man Cybern. 22, 418–435.

Yule, G.U., 1900. On the association of attributes in statistics. Philos. Trans. A 194, 257–319.