ELSEVIER

Pattern Recognition Letters 21 (2000) 385–397

Pattern Recognition Letters

www.elsevier.nl/locate/patrec

Combination of neural and statistical algorithms for supervised classification of remote-sensing images

Giorgio Giacinto [a], Fabio Roli [a,*], Lorenzo Bruzzone [b]

[a] *Department of Electrical and Electronic Engineering, University of Cagliari, Piazza D'Armi, 09123 Cagliari, Italy*
[b] *Department of Civil and Environment Engineering, University of Trento, Via Mesiano, 77, 38050 Trento, Italy*

Received 18 March 1999; received in revised form 5 October 1999

Abstract

Various experimental comparisons of algorithms for supervised classification of remote-sensing images have been reported in the literature. Among others, a comparison of neural and statistical classifiers has previously been made by the authors in (Serpico, S.B., Bruzzone, L., Roli, F., 1996. Pattern Recognition Letters 17, 1331–1341). Results of reported experiments have clearly shown that the superiority of one algorithm over another cannot be claimed. In addition, they have pointed out that statistical and neural algorithms often require expensive design phases to attain high classification accuracy. In this paper, the combination of neural and statistical algorithms is proposed as a method to obtain high accuracy values after much shorter design phases and to improve the accuracy–rejection tradeoff over those allowed by single algorithms.

Keywords: Remote-sensing image classification; Combination of multiple classifiers; Design of classification systems; Accuracy–rejection tradeoff

1. Introduction

Supervised classification of remote-sensing images is currently performed by neural and statistical algorithms (Benediktsson et al., 1990; Serpico and Roli, 1995; Bruzzone et al., 1997; Kanellopoulos et al., 1997; Bruzzone and Serpico, 1997; Bruzzone and Prieto, 1999). An experimental comparison of various neural and statistical algorithms was presented by the authors in (Serpico et al., 1996). Experiments reported by the authors and other researchers have clearly shown that the superiority of one algorithm over another cannot be claimed for remote-sensing image classification. Performances basically depend on the characteristics of the images considered and on the efforts devoted to the 'design phases' of the algorithms used (i.e., choice of classifier architectures, tuning of learning parameters, etc.). For example, the superiority of the k-nearest neighbour (k-nn) classifier over the multilayer perceptron (MLP) neural network, or vice versa, strongly depends on the efforts devoted to the selection of an appropriate value of the k parameter for the k-nn classifier and to the selection of an appropriate architecture and suitable learning parameters for the MLP neural network. In our experiments, we

* Corresponding author. Tel.: +39-070-6755874; fax: +39-070-6755900.
E-mail address: roli@diee.unica.it (F. Roli).

0167-8655/00/$ - see front matter
PII: S0167-8655(00)00006-4

also noticed that a sufficient level of classification accuracy may be reached through a reasonable design effort (Serpico and Roli, 1995; Roli, 1996; Serpico et al., 1996; Bruzzone et al., 1997; Giacinto and Roli, 1997a; Kanellopoulos et al., 1997). Further improvements often require an increasingly expensive design phase. Results reported in (Serpico et al., 1996) pointed out that MLP neural networks easily reached an accuracy of about 80% on the selected test set. By contrast, an accuracy higher than 89% needed a long design phase, involving trials with different architectures and at different learning rates. This conclusion is in close agreement with the ones drawn by other researchers (Beyer and Smieja, 1996; Kanellopoulos et al., 1997).

The above-mentioned experimental analyses have also pointed out the 'complementary' behaviours of neural and statistical algorithms in terms of classification errors. In particular, careful analyses of the results reported in (Serpico et al., 1996; Giacinto and Roli, 1997a; Giacinto et al., 1997b; Kanellopoulos et al., 1997) have shown us that even neural and statistical algorithms reaching similar overall accuracies made a significant number of different classification errors.

The above behaviours of neural and statistical classifiers can be exploited by use of methods for combining multiple classifiers in order to develop classification systems that may attain high accuracies after short design phases. According to our experimental analyses, neural and statistical classifiers providing reasonable but not yet high accuracies can be obtained after short design phases, and, typically, these classifiers make a sufficient number of uncorrelated errors. High accuracies can then be reached by combining such classifiers. Several experiments reported in the literature have shown that the combination of the results yielded by an ensemble of classifiers making 'uncorrelated' errors may give a gain in accuracy, as compared with the accuracies provided by single classifiers (Hansen and Salamon, 1990; Xu et al., 1992; Sharkey, 1996; Giacinto and Roli, 1997a; Giacinto et al., 1997b).

The uncorrelation of errors could also be exploited to facilitate the 'rejection' of misclassifications. It is reasonable to assume that the errors made by a multiple classifier system (MCS) consisting of neural and statistical classifiers are characterized by classification 'reliabilities' lower than the ones of individual classifiers. Therefore, the accuracy–rejection tradeoff allowed by such an MCS should be better than those admitted by single algorithms.

In this paper, the combination of neural and statistical algorithms is proposed as a method to obtain high accuracy values after short design phases and to improve the accuracy–rejection tradeoff over those allowed by single algorithms. Short descriptions of methods used to combine neural and statistical classifiers are provided in Section 2. The concept of accuracy–rejection tradeoff is briefly defined in Section 3, where we also propose two measures of 'classification entropy' aimed at comparing the 'aptitudes' of various classifiers to reject errors without rejecting correct classifications. Experimental results obtained on the same data set as used in (Serpico et al., 1996) are reported and discussed in Section 4. Conclusions are drawn in Section 5.

2. Methods for combining multiple classifiers

Three methods previously proposed in the handwriting recognition field (Xu et al., 1992) were used to perform the combination of statistical and neural classifiers. In the following sections, such methods are briefly summarized with reference to a classification task for M data classes. Each class is assumed to represent a set of specific patterns, each pattern being characterized by a feature vector $\underline{X}$. K different classification algorithms are also assumed to be available to solve the classification problem, so that ensembles made up of k different classifiers ($k = 1..K$) may be used.

2.1. Combination by voting principle

Let us consider an MCS made up of k different classifiers. Each classifier provides the results in terms of the class labels assigned to the patterns. A given input pattern receives, therefore, k classification labels from the MCS, each label corresponding to one of the M data classes. A simple

method to combine the results of the k classifiers lies in interpreting each classification result as a 'vote' for one of the M data classes. The data class that receives a larger number of votes than a prefixed threshold is taken as the class of the input pattern. Typically, the threshold is equal to half the number of the considered classifiers ('majority' rule). However, more conservative rules can be adopted (e.g., the 'unison' rule).

2.2. Combination by the Bayesian average

It is well-known that some classification algorithms can provide estimates of the posterior probabilities that an input pattern $\underline{X}$ may belong to the data class ω_i

$$P(\underline{X} \in \omega_i \,|\, \underline{X}), \quad i = 1..M. \tag{1}$$

For example, estimates of the postprobabilities can be given by MLP neural networks (Richard and Lippman, 1991; Gish, 1990). Analogously, it is straightforward for the k-nn classifier to compute such estimates (Duda and Hart, 1973). When these kinds of classifiers are used, a simple method of combination lies in the computation of the 'average' posterior probabilities:

$$P_{\text{av}}(\underline{X} \in \omega_i \,|\, \underline{X}) = \frac{1}{K} \sum_{k=1}^{K} P_k(\underline{X} \in \omega_i \,|\, \underline{X}), \quad i = 1..M. \tag{2}$$

The final classification is performed according to the Bayesian criterion, that is, the input pattern $\underline{X}$ is assigned to the data class for which $P_{\text{av}}(\underline{X} \in \omega_i \,|\, \underline{X})$ has the maximum value.

2.3. Combination by belief functions

This method exploits the knowledge of the decisions made by the classifiers forming an MCS on the training set. Such knowledge is extracted by the so-called 'confusion matrix'. For each classifier C_k, $k = 1..K$, it is quite simple to see that the confusion matrix computed on the training set can provide estimates of the following probabilities:

$$P(\underline{X} \in \omega_i \,|\, C_k(\underline{X}) = j_k), \quad i = 1..M,\ k = 1..K,\ j_k \in [1, M], \tag{3}$$

where $C_k(\underline{X}) = j_k$ means that the classifier C_k assigned the training pattern $\underline{X}$ to the class j_k.

On the basis of the above probabilities, the K classifiers can be combined according to the following 'belief' functions:

$$\text{bel}(i) = \eta \prod_{k=1}^{K} P(\underline{X} \in \omega_i \,|\, C_k(\underline{X}) = j_k), \quad i = 1..M, \tag{4}$$

where η is a constant that ensures that $\sum_{i=1}^{M} \text{bel}(i) = 1$. The final classification is then performed by assigning the input pattern $\underline{X}$ to the data class for which bel(i) has the maximum value.

The reader interested in more details about the above combination methods is referred to (Xu et al., 1992).

3. The accuracy–rejection tradeoff

In this section, the concept of accuracy–rejection tradeoff is briefly defined (Section 3.1). Section 3.2 presents two measures of 'classification entropy' aimed at comparing the 'aptitudes' of certain classifiers to reject errors without rejecting correct classifications.

3.1. The concept of accuracy–rejection tradeoff

In many pattern-recognition applications, the accuracy reached by the classification system is often lower than that requested by the end user. A common solution to this problem is to 'reject', i.e., not to classify, the patterns that are the most likely to be wrongly classified and to handle them by more sophisticated procedures (typically, a manual classification process is performed). As an example, the accuracy of a thematic map derived from remote-sensing images can be increased if one entrusts the classification of rejected patterns to a skilled photointerpreter. Classification with a rejection option allows one to obtain the desired accuracy, as potential errors are converted into rejections. However, handling high rejection rates is usually too time-consuming for application

purposes. In addition, correct classifications may also be converted into rejections as the rejection rate increases. Therefore, a tradeoff between accuracy and rejection is mandatory. The formulation of the best accuracy–rejection tradeoff and the related optimal rejection rule can be found in (Chow, 1970). In the following, we briefly summarize them.

Let us consider a classification task for M classes and a classification algorithm that assigns to each pattern $\underline{X}$ estimates of posterior probabilities $P(\underline{X} \in \omega_i | \underline{X})$ such that $\sum_{i=1}^{M} P(\underline{X} \in \omega_i | \underline{X}) = 1$.

According to the optimal rejection rule, a pattern $\underline{X}$ is rejected if

$$\max_i P(\omega_i | \underline{X}) < T, \tag{5}$$

where T is a threshold dependent on the rejection rate fixed by the end user ($0 < T < 1$). (It is worth noting that the rejection rate increases when the threshold increases.) The rationale of the Chow rejection rule becomes evident if one observes that $\max_i P(\omega_i | \underline{X})$ is the conditional probability of classifying a given pattern $\underline{X}$ correctly. Therefore, for a given threshold T and the related rejection rate, the patterns with the highest probabilities to be wrongly classified are rejected. A detailed proof of the optimality of the above rule can be found in (Battiti and Colla, 1994).

For real pattern-recognition applications, the designer of the classification system is also interested in analyzing the different accuracy–rejection tradeoffs that a certain algorithm can allow. In addition, the designer often needs to compare the tradeoffs provided by different algorithms (or different 'versions' of the same algorithm) in order to select the classifier most suited to the end-user's requirements (e.g., the classifier reaching the highest accuracy and a rejection rate below a fixed threshold). This kind of analysis can be performed in the so-called accuracy–rejection plane (A–R plane), introduced by Battiti and Colla (1994). In the A–R plane, the accuracy–rejection tradeoffs provided by a given algorithm are described by the curve $A(R)$ connecting the points that represent the accuracy values for different rejection rates.

3.2. Measures of classification entropy for evaluating the 'aptitude' of a classifier for error rejection

The above-mentioned analysis in the A–R plane is necessary whenever a detailed evaluation of the accuracy–rejection tradeoff is requested. In some cases, a simple evaluation of the 'aptitude' of a classifier to reject errors without rejecting correct classifications can be sufficient. As an example, during the design phase, an evaluation of such aptitude can allow the designer to disregard immediately the algorithms poorly suited to providing good accuracy–rejection tradeoffs. The designer can then perform a detailed analysis in the A–R plane for the most promising algorithms.

In the following, we propose two measures of classification entropy aimed at comparing the 'aptitudes' of various classifiers to reject errors without rejecting correct classifications. Such measures allow the designer to evaluate which classifiers are the most likely to provide good accuracy–rejection tradeoffs.

Given a data set with N patterns, let us assume that there exists a classifier that correctly classifies N_c patterns and misclassifies $N_w = N - N_c$ patterns. Let us also assume that such a classifier provides estimates of the class posterior probabilities. In order to evaluate the classifier's aptitude to reject errors without rejecting correct classifications, we have defined the following two measures of classification entropy:

$$H(\text{correct classification}) = -\frac{1}{N_c}\sum_{m=1}^{N_c}\sum_{j=1}^{M} P(\omega_j | \underline{X}_m) \ln\left(P(\omega_j | \underline{X}_m)\right),$$
$$0 \leqslant H \leqslant \ln\frac{1}{M}, \tag{6}$$

$$H(\text{misclassification}) = -\frac{1}{N_w}\sum_{m=1}^{N_w}\sum_{j=1}^{M} P(\omega_j | \underline{X}_m) \ln\left(P(\omega_j | \underline{X}_m)\right),$$
$$0 \leqslant H \leqslant \ln\frac{1}{M}, \tag{7}$$

which we have named the 'entropy of correct classification' and the 'entropy of misclassification'.

It is worth noting that the H(correct classification) and the H(misclassification) are computed on the sets of correctly and wrongly classified patterns, respectively. Therefore, they characterize the average degrees of 'confusion' in the outputs of the classifier for correct and wrong classifications. In particular, the values of such measures are close to 0 as much as all patterns are classified with values of $\max_i P(\omega_i|\underline{X})$ close to 1 (i.e., the classifier performs correct or wrong classifications to high degrees of certainty). On the contrary, the above measures take on values close to $\ln(1/M)$ as much as all patterns are classified with values of $P(\omega_i|\underline{X}),\ i = 1..M$, close to $1/M$ (i.e., the classifier performs correct or wrong classifications that are very 'confused'). It is easy to deduce that a classifier involving a value close to zero for the H(correct classification) and a value close to $\ln(1/M)$ for the H(misclassification) is very apt to reject errors without rejecting correct classifications, when the Chow rule is applied.

It is worth noting that the above two measures work better for comparison purposes than for an 'absolute' evaluation of the classifier's aptitude for error rejection. In addition, they do not provide any precise information on the related accuracy–rejection tradeoff. To this end, an analysis in the A–R plane must be performed.

4. Experimental results

4.1. Data set description

The data set is the same, as considered in (Serpico et al., 1996). It consists of multisensor remote-sensing images related to an agricultural area near the village of Feltwell (UK). A section (250×350 pixels) of a scene acquired by an optical sensor (an Airborne Thematic Mapper scanner) and a radar sensor (a NASA/JPL synthetic aperture radar) was selected. Our experiments were carried out characterizing each pixel by a 15-element feature vector containing the brightness values in six optical bands and over nine radar channels. We selected 10944 pixels belonging to five agricultural classes (i.e., sugar beets, stubble, bare soil, potatoes, and carrots) and randomly subdivided them into a training set (5124 pixels) and a test set (5820 pixels). A detailed description of the data set can be found in (Serpico and Roli, 1995).

4.2. Experiment planning

Experiments were performed to attain the following objectives:

- to show that the combination of neural and statistical classifiers can be used as a method to obtain high classification accuracies after much shorter design phases than the ones required by classification systems based on a single algorithm;
- to show that the combination of neural and statistical classifiers may improve the accuracy–rejection tradeoff over the tradeoffs provided by single algorithms.

Concerning the first objective, the design phase of a classification system based on a single algorithm usually involves 'training and testing' different classification algorithms in order to evaluate the most effective for the data set at hand. For each algorithm, the designer of the classification system performs a certain number of trials using different values of the 'design parameters' (e.g., different learning parameters, different classifier architectures, etc.). The number of trials carried out depends on various factors (e.g. the designer's expertise in the classification task at hand, the time allocated to the design phase, the computer performances, etc.). In our experiments, a number of trials aimed at simulating a very long design phase were performed. Five neural and statistical algorithms were trained and tested on the described data set, using many values of the related design parameters (Section 4.3). The purposes of such experiments were to evaluate the 'distribution' of the accuracies exhibited by the classifiers generated during very long design phases and, in particular, to establish the maximum accuracy value achievable. It is worth noting that the methods commonly used by designers of classification systems were adopted.

In order to quantitatively evaluate the complexity of a given design phase and to compare

different design phases, we defined the following measure of 'design complexity' (DC):

$$\text{Design complexity} = \text{card}\{\text{version space}\} \quad (8)$$

Such a measure is based on the concept of 'version space' recently proposed by Partridge and Yates (1996) in the field of MCSs. This space includes the different 'versions' of a classification algorithm generated during the design phase by varying the values of the design parameters of the algorithm. In Eq. (8), we define the cardinality of the version space as a measure of design complexity. As an example, the design phase of a MLP neural network was carried out for two different values of the learning rate, and 10 training trials were made using different 'starting random weights'. The design phase exhibited a DC value equal to 20, as the version space contained 20 networks.

4.3. Results and comparisons

Five neural and statistical classifiers were trained and tested on the selected data set: a Bayes classifier, a k-nn classifier, an MLP neural network, a Radial Basis Functions (RBF) neural network, and a Probabilistic Neural Network (PNN). A long design phase was performed for each classifier, except for the Bayes classifier, which does not need a design phase, and for the PNN. For the latter, an a priori fixed value of the smoothing parameter equal to 0.1 was selected (Serpico et al., 1996).

Forty-six trials for different values of the k parameter, from $k = 1$ up to $k = 91$ (by steps equal to two), were carried out for the k-nn classifier (DC = 46). Experiments using five architectures with one or two hidden layers and various numbers of neurons per layer were performed for the MLP neural network. We considered the same architectures as previously tested by Serpico et al. (1996). All the networks had 15 input units and five output units as the numbers of input features and data classes, respectively (Section 4.1). For each architecture, 20 trials for different starting random weights were performed. On the basis of the experience gained by Serpico et al. (1996), a value equal to 0.01 was fixed for the learning rate. Consequently, the design phase for the MLP neural network exhibited a DC value equal to one hundred. For the RBF neural network, the design phase involved different trials carried out using a very simple strategy based on the k-means clustering algorithm for defining the network architecture. Thirty-four different values of the parameter related to the 'number of clusters' were used (DC = 34).

The global DC of the above design phase was equal to 182, as 182 'versions' of the considered classifiers were created. It is worth noting that, according to the results reported in the literature, this can be considered a very long design phase.

The performances of the classifiers generated during the above design phase are summarized in Table 1. For each kind of classifier, they are characterized by the minimum, mean, and maximum accuracy values exhibited in the version space, as suggested in (Lawrence et al., 1997). An analysis of the distribution of such accuracies led to conclusions that are in agreement with those drawn in our previous work: an improvement in accuracy above a certain level requires a very expensive design phase. As an example, in (Serpico et al., 1996), MLP neural networks provided an accuracy of 89.6% and a DC = 10. Table 1 points out that an improvement in accuracy of just 0.15% was obtained for a DC = 100. It should be noted that, in some cases, classifiers generated during a specific design phase exhibited a unimodal and Gaussian-like distribution of accuracies. As an example, the distribution of the accuracies for the 20 versions of the MLP neural network with a 15–5–5 architecture (i.e., a network with one hidden layer of five neurons) was of the Gaussian type. The maximum accuracy value obtained after this very long design phase (DC = 182) was provided by the k-nn classifier for $k = 25$ (89.80% accuracy).

In order to evaluate the accuracies provided by the combination of various neural and statistical classifiers, we carried out several experiments (Giacinto et al., 1997a; Giacinto and Roli, 1997a). In particular, we focused on the evaluation of the accuracies provided by MCSs made up of three classifiers obtained after very short design phases (i.e., DC = 3). To this end, different ensembles of three classifiers were generated by using a priori fixed values of the related design parameters. All

Table 1
The test-set accuracies exhibited by the classifiers generated during the design phase are summarized[a]

Classifier	Minimum accuracy (%)	Mean accuracy (%)	Maximum accuracy (%)	DC
Bayes	79.37	79.37	79.37	1
k-nn	86.63	88.36	*89.80*	46
MLP	73.45	81.60	89.75	100
RBF	71.40	78.95	86.51	34
PNN	88.66	88.66	88.66	1

[a] For each kind of classifier, the minimum, mean and maximum accuracies are given, as well as the DCs of such phases. The best performance on the test set is given in italics (*k*-nn classifier with $k = 25$).

the ensembles consisted of one *k*-nn classifier and two MLP neural networks. The results yielded by two of these MCSs are reported later on. In order to realistically simulate the accuracies provided by MCSs designed with DC = 3, we used the following approach to choosing the classifiers forming such MCSs:

- An a priori fixed value of the *k* parameter equal to the square root of the training set size was used to design the *k*-nn classifier (DC = 1). This choice of *k* was in agreement with a rule of thumb commonly applied by practitioners to design a *k*-nn classifier very quickly.
- We selected three MLP neural-networks architectures from among the ones considered. For each architecture, we chose the network that had exhibited the accuracy closest to the mean value during the related design phase. This choice allowed us to realistically simulate the accuracies provided by MLP neural networks generated by performing a 'unique' trial based on starting random weights (DC = 1), according to the Gaussian-like distributions of accuracies pointed out by our experiments.

The three combination methods described in Section 2 were applied to the above two ensembles. Table 2 shows the results yielded by such MCSs in terms of percent classification accuracies and of Kappa coefficient values. The performances of the classifiers forming the MCSs are also presented for the sake of comparison. Moreover, in order to assess the degrees of statistical significance of the differences in accuracy between the MCSs and the single classifiers that form them, we computed the values of the Zeta statistics (see Table 3).

The following conclusions can be drawn from the results in Tables 2 and 3:

- The combination of statistical and neural classifiers is an efficient method to obtain high accuracies after very short design phases. It is worth noting that the accuracies of the two MCSs characterized by DC = 3 are often higher than the one provided by the best classifier generated after a design phase with DC = 182 (see Table 1).

Table 2
The performances of the two MCSs on the test-set[a]

MCS	Single classifier		Majority rule		Bayesian average		Belief functions	
	%Accuracy	Kappa	%Accuracy	Kappa	%Accuracy	Kappa	%Accuracy	Kappa
MLP (15–30–5)	85.50	0.806	89.96	0.867	89.95	0.867	89.95	0.867
MLP (15–15–5)	87.25	0.830						
k-nn ($k = 71$)	88.40	0.840						
MLP (15–30–15–5)	83.35	0.777	89.7	0.863	89.59	0.862	89.64	0.863
MLP (15–15–5)	87.25	0.830						
k-nn ($k = 71$)	88.4	0.847						

[a] The results obtained by the majority-rule, Bayesian-average, and belief-functions combination methods are given in terms of percent classification accuracies and of Kappa coefficient values. The performances of the single classifiers making up the MCSs are also shown. (For each MLP network, the number of neurons on each layer is given within brackets.)

Table 3
The values of the Zeta statistics related to the test-set performances are provided[a]

Zeta statistics	Majority rule MCS	Bayesian average MCS	Belief functions MCS
MLP (15–30–5)	7.62	7.65	7.64
MLP (15–15–5)	4.78	4.80	4.79
k-nn ($k = 71$)	3.58	3.60	3.59
MLP (15–30–15–5)	10.33	10.22	10.35
MLP (15–15–5)	4.24	4.11	4.24
k-nn ($k = 71$)	2.11	1.98	2.11

[a] Such values point out the degrees of significance of the differences in accuracy between the MCSs and the single classifiers that form them. (For each MLP network, the number of neurons on each layer is given within brackets.)

- The combination of a small set of statistical and neural classifiers allows one to improve the accuracies of single classifiers.
- According to the results of the Zeta test, the differences in accuracy between the MCSs and the single classifiers that form them are very significant (see Table 3). (We recall that such differences exhibit degrees of significance higher than 95%, if the values of the Zeta statistics are larger than 1.96, whereas the degrees of significance are higher than 99%, if the values of the Zeta statistics exceed 2.58.)

The best classifier obtained by the design phase summarized in Table 1 (i.e., the k-nn classifier with $k = 25$) and the above two MCSs were also applied to the whole image of 250×350 pixels (Section 4.1). Fig. 1 shows the reference map for this image. The k-nn classifier provided an accuracy equal to

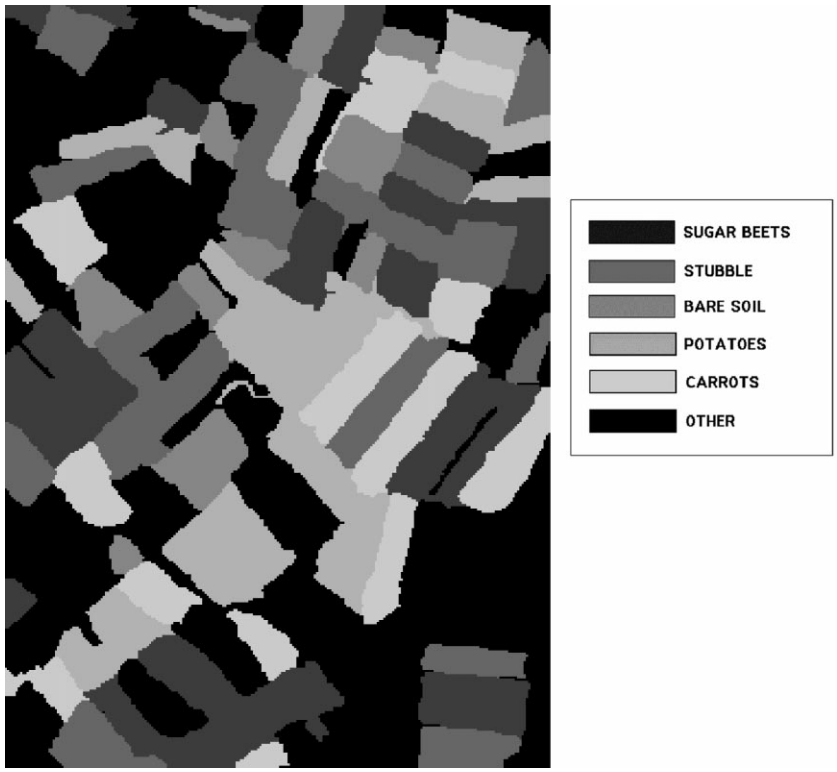

Fig. 1. The reference map for the whole image.

87.83% for the whole image. Table 4 gives the accuracies reached by the two MCSs in terms of percent classification accuracies and of Kappa coefficient values. Table 5 shows the values of the Zeta statistics related to the degrees of statistical significance of the differences in accuracy between the MCSs of Table 4 and the single classifiers that form them.

Fig. 2(a) shows the classification map of the k-nn classifier for the whole image. As an example, the maps related to the two MCSs based on the Bayesian average are also presented in Figs. 2(b) and (c). The classification results obtained for the whole image confirmed that the combination of statistical and neural classifiers is a suitable method to obtain high accuracies after short design phases. The accuracies of the two MCSs were higher than the one provided by the k-nn classifier generated after a design phase with DC = 182. In addition, according to the results of the Zeta test, it is worth noting that the differences in accuracy between the MCSs and the single classifiers that form them are very significant (Table 5).

The second objective of our experiments was to show that the combination of neural and statistical classifiers may improve the accuracy–rejection tradeoff over the tradeoffs allowed by single algorithms. To this end, the accuracy–rejection tradeoffs of the above two MCSs were analyzed. We used the MCSs based on the Bayesian-average combination method, as such a method provides the estimates of postprobabilities, as required by our two measures of classification entropy and by the Chow rule.

First of all, our measures of classification entropy were computed for the two MCSs and for the related classifiers. Table 6 shows the results obtained. It is worth noting that the combination approach allows an increase in entropy of misclassification, whereas the entropy of correct classification does not exceed the one of the classifier with the maximum entropy value. Therefore, it is

Table 4
The performances of the two MCSs on the whole image[a]

MCS	Single classifier		Majority rule		Bayesian average		Belief functions	
	%Accuracy	Kappa	%Accuracy	Kappa	%Accuracy	Kappa	%Accuracy	Kappa
MLP (15–30–5)	88.76	0.856	89.38	0.864	88.93	0.858	89.12	0.86
MLP (15–15–5)	88	0.846						
k-nn ($k = 71$)	85.51	0.814						
MLP (15–30–15–5)	88.47	0.852	89.50	0.865	89.26	0.862	89.25	0.862
MLP (15–15–5)	88.00	0.846						
k-nn ($k = 71$)	85.51	0.814						

[a] The results obtained by the majority-rule, Bayesian-average, and belief-functions combination methods are given in terms of percent classification accuracies and of Kappa coefficient values. The performances of the single classifiers making up the MCSs are also shown. (For each MLP network, the number of neurons on each layer is given within brackets.)

Table 5
The values of the Zeta statistics related to the performances on the whole image are provided[a]

Zeta statistics	Majority rule MCS	Bayesian average MCS	Belief functions MCS
MLP (15–30–5)	3.34	0.83	1.67
MLP (15–15–5)	7.42	4.92	5.76
k-nn ($k = 71$)	19.74	17.29	18.13
MLP (15–30–15–5)	4.32	3.32	3.32
MLP (15–15–5)	7.85	6.60	6.60
k-nn ($k = 71$)	20.19	18.96	18.96

[a] Such values point out the degrees of significance of the differences in accuracy between the MCSs and the single classifiers that form them. (For each MLP network, the number of neurons on each layer is given within brackets.)

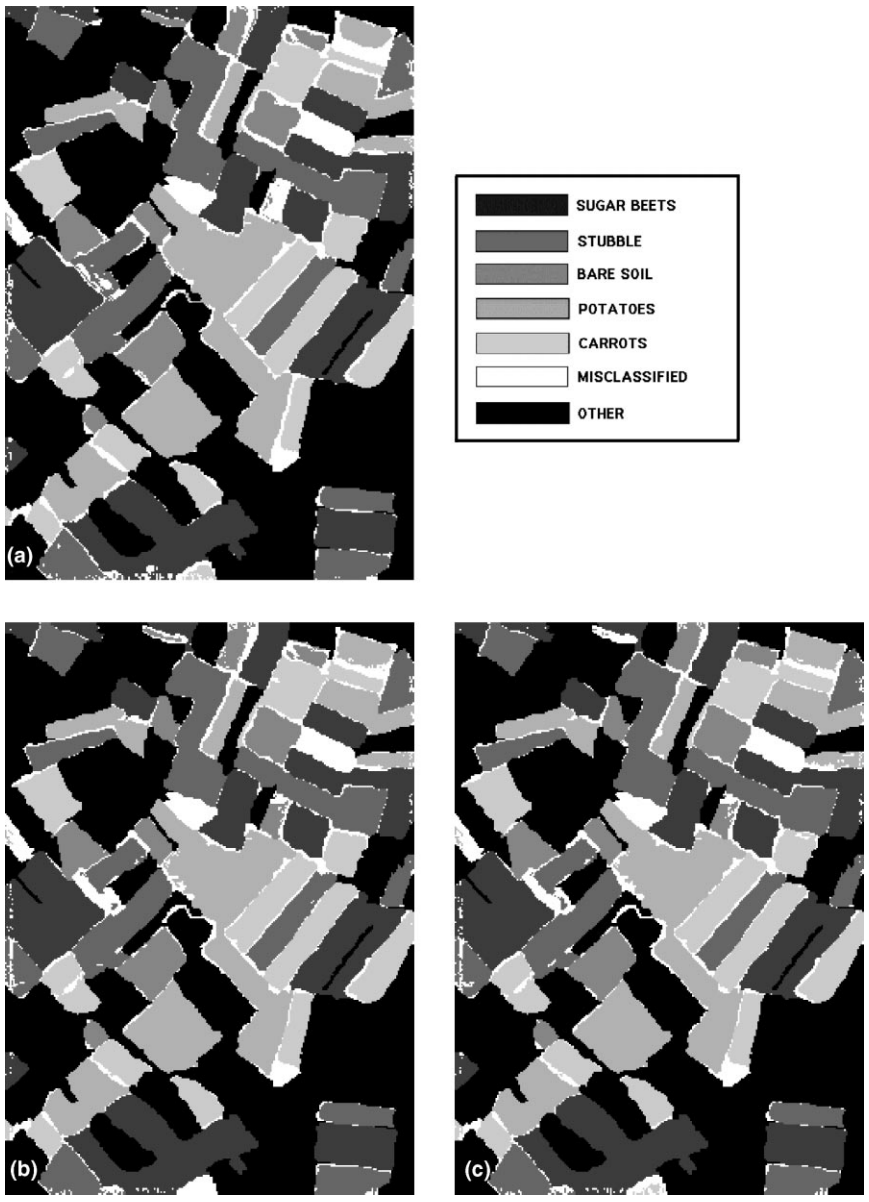

Fig. 2. (a) Classification map obtained by the k-nn classifier ($k = 25$); (b) classification map provided by the first MCS (based on the Bayesian average) in Tables 2 and 3; (c) classification map obtained by the second MCS (based on the Bayesian average) in Tables 2 and 3.

reasonable to hypothesize that the combination method is suited to improving the aptitudes of single classifiers to reject errors without rejecting correct classifications. This conclusion was confirmed by the detailed analysis of the accuracy–rejection tradeoff performed on the A–R plane by using the Chow rule. Figs. 3 and 4 show the accuracy–rejection tradeoffs on the A–R plane for the two MCSs in Table 2 and for the related classifiers. Except for few values of the rejection rate, the accuracies of MCSs are always higher than the ones of single classifiers. It should be

Table 6
Measures of classification entropy for the two MCSs considered and for the related classifiers

MCS	H (correct classification)	H (misclassification)
MLP (15–30–5)	0.21	0.48
MLP (15–15–5)	0.32	0.64
k-nn ($k = 71$)	0.20	0.65
Bayesian average	0.32	0.71
MLP (15–30–15–5)	0.16	0.39
MLP (15–15–5)	0.32	0.64
k-nn ($k = 71$)	0.20	0.65
Bayesian average	0.32	0.70

stressed that the accuracies of MCSs are higher for a range of rejection rates from 0% to 12–14%. Such a range is usually the most significant for application purposes. Therefore, we can say that the combination allows us to improve the accuracy–rejection tradeoff over the tradeoffs provided by single classifiers.

5. Conclusions

In this paper, we have shown that the combination of neural and statistical algorithms is an efficient method to obtain high accuracy values after short design phases and to improve the ac-

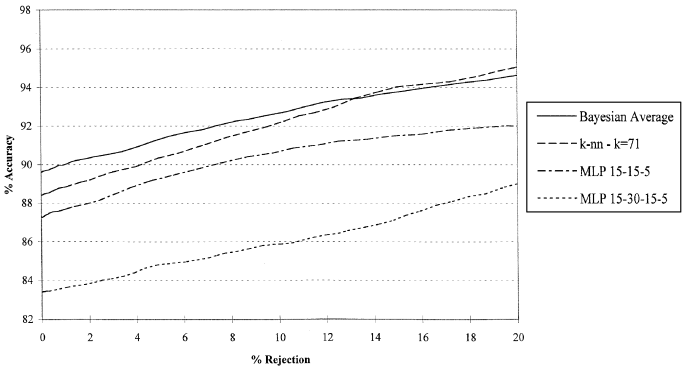

Fig. 3. The accuracy–rejection tradeoffs of the first MCS in Table 2 are represented on the AR plane for values of the rejection rate ranging from 0% to 20%. The MCS is based on the Bayesian average combination method. The tradeoffs of the related classifiers are also given for the sake of comparison.

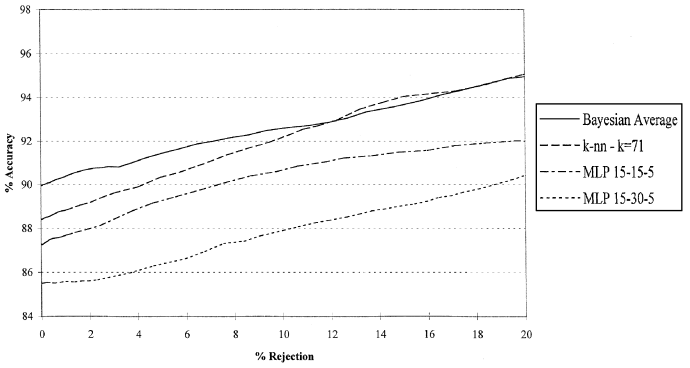

Fig. 4. The accuracy-rejection tradeoffs of the second MCS in Table 2 are represented on the A–R plane for values of the rejection rate ranging from 0% to 20%. The MCS is based on the Bayesian average combination method. The tradeoffs of the related classifiers are also given for the sake of comparison.

curacy–rejection tradeoff over the tradeoffs provided by single algorithms. The reported results have confirmed and further developed the conclusions we drew in (Serpico et al., 1996). In particular, as compared with our previous work, we have also demonstrated the effectiveness of combination methods to improve the accuracy–rejection tradeoff. The pattern-recognition application considered has been remote-sensing image classification; however, on the basis of the experiments we are currently performing, we think that the conclusions of this paper may also be valid for other applications, provided that the uncorrelation of the errors made by neural and statistical classifiers can be assumed. There remains the problem of designing a set of classifiers that make the largest number of different errors, or of selecting the most 'uncorrelated' classifiers from a given set (Sharkey, 1996). Our present research is tackling this problem (Giacinto and Roli, 1997b; Giacinto and Roli, 1999).

Acknowledgements

The authors would like to thank Prof. S.B. Serpico for helpful comments and suggestions. This work was supported by the Italian Space Agency within the framework of the project 'Metodologie innovative di integrazione, gestione, analisi di dati da sensori spaziali per l'osservazione della idrosfera, dei fenomeni di precipitazione e del suolo'.

References

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

Benediktsson, J.A., Swain, P.H., Ersoy, O.K., 1990. Neural network approaches versus statistical methods in classification of multisource remote-sensing data. IEEE Transactions on Geoscience and Remote Sensing 28, 540–552.

Beyer, U., Smieja, F., 1996. Learning from examples, agent teams and the concept of reflection. International Journal of Pattern Recognition and Artificial Intelligence 10 (3), 251–272.

Bruzzone, L., Conese, C., Maselli, F., Roli, F., 1997. Multisource classification of complex rural areas by statistical and neural-network approaches. Photogrammetric Engineering and Remote Sensing 63 (5), 523–533.

Bruzzone, L., Serpico, S.B., 1997. Classification of imbalanced remote-sensing data by neural networks. Pattern Recognition Letters 18 (11–13), 1323–1328.

Bruzzone, L., Prieto, D.F., 1999. A technique for the selection of kernel-function parameters in RBF neural networks for classification of remote-sensing images. IEEE Transactions on Geoscience and Remote Sensing Sensing 37 (2), 1179–1184.

Chow, C.K., 1970. On optimum recognition error and reject tradeoff. IEEE Transactions on Information Theory 16, 41–46.

Duda, R.O., Hart, P.E., 1973. Pattern Classification and Scene Analysis. Wiley, New York.

Giacinto, G., Roli, F., 1997a. Ensembles of neural networks for soft classification of remote sensing images. In: Proceedings of the European Symposium on Intelligent Techniques, Bari, Italy, pp. 166–170.

Giacinto, G., Roli, F., 1997b. Adaptive selection of image classifiers. In: Proceedings of the Ninth International Conference on Image Analysis and Processing, Florence, Italy, pp. 38–45.

Giacinto, G., Roli, F., Vernazza, G., 1997a. Comparison and combination of statistical and neural network algorithms for remote-sensing image classification. In: Kanellopoulos, Wilkinson, Roli, Austin (Eds.), Neurocomputation in Remote Sensing Data Analysis, Springer, Berlin, pp. 117–124.

Giacinto, G., Paolucci, R., Roli, F., 1997b. Application of neural networks and statistical pattern recognition algorithms to earthquake risk evaluations. Pattern Recognition Letters 18, 1353–1362.

Giacinto, G., Roli, F., 1999. Automatic Design of Multiple Classifier Systems by Unsupervised Learning. In: Proceedings of the International Workshop on Machine Learning and Data Mining in Pattern Recognition, Vol. 1715, Leipzig, Germany, Springer, Berlin, pp. 131–143.

Gish, H., 1990. A probabilistic approach to the understanding and training of neural network classifiers. In: Proceedings of the 1990 International Conference on Acoustic, Speech, and Signal Processing, 3–6 April, pp. 1361–1364.

Hansen, L.K., Salamon, P., 1990. Neural Network Ensembles. IEEE Transactions on Pattern Analysis Machine Intelligence 12, 993–1001.

Kanellopoulos, I., Wilkinson, G., Roli, F., Austin J., (Eds.) 1997. Neurocomputation in Remote Sensing Data Analysis, Springer, Berlin.

Lawrence, S., Back, A.D., Tsoi, A.C., Giles, C.L., 1997. On the distribution of performance from multiple neural-network trials. IEEE Transactions on Neural Networks 8, 1507–1517.

Partridge, D., Yates, W.B., 1996. Engineering multiversion neural-net systems. Neural Computation 8, 869–893.

Richard, M., Lippman, R., 1991. Neural network classifiers estimate Bayesian a posteriori probabilities. Neural Computation 3, 461–463.

Roli, F., 1996. Multisensor image recognition by neural networks with understandable behaviour. International Journal of Pattern Recognition and Artificial Intelligence 10 (8), 887–917.

Serpico, S.B., Bruzzone, L., Roli, F., 1996. An experimental comparison of neural and statistical non-parametric algorithms for supervised classification of remote-sensing images. Pattern Recognition Letters 17, 1331–1341.

Serpico, S.B., Roli, F., 1995. Classification of multi-sensor remote-sensing images by structured neural networks. IEEE Transactions on Geoscience and Remote Sensing 33, 562–578.

Sharkey, A.J.C. (Ed.), 1996. Combining artificial neural nets: ensemble approaches. Connection Science (Special Issue) 8.

Xu, L., Krzyzak, A., Suen, C.Y., 1992. Methods for combining multiple classifiers and their applications to handwriting recognition. IEEE Transactions on Systems Man and Cybernetics 22, 418–435.