ELSEVIER

Pattern Recognition Letters 17 (1996) 529–536

Pattern Recognition Letters

A note on comparing classifiers [1]

Robert P.W. Duin [*]

Pattern Recognition Group, Faculty of Applied Physics, Delft University of Technology, P.O. Box 5046, 2600 GA Delft, Netherlands

Received 23 June 1995; revised 22 November 1995

Abstract

Recently many new classifiers have been proposed, mainly based on neural network techniques. Comparisons are needed to evaluate the performance of the new methods. It is argued that a straightforward fair comparison demands automatic classifiers with no user interaction. As this conflicts with one of the main characteristics of neural networks, their flexibility, the question whether they are better or worse than traditional techniques might be undecidable.

Keywords: Automatic classifiers; Benchmarking; Comparisons; Feedforward neural networks

1. Introduction

The large interest in artificial neural networks (ANN) during the last ten years has produced a number of interesting pattern recognition applications with sometimes surprisingly good results (Cheng and Titterington, 1994; Golomb et al., 1991; Kamata et al., 1992; Michie et al., 1994; Prechelt, 1994; Ripley, 1994; Sabourin and Drouhard, 1992; Schmidt, 1994; Sejnowski and Rosenberg, 1988). This at least suggests that neural networks are good for building pattern classifiers. It has to be regretted that originally almost never comparisons were made with traditional techniques like the nearest neighbor rule (NNR). More recently, the interest in such comparisons has grown, e.g. see (Ripley, 1994; Prechelt, 1994). A study by Schmidt et al. (1994) showed that the NNR is equivalent or better in a number of applications, including NETtalk, the text-to-speech recognition problem studied by Sejnowski and Rosenberg (1988) that originally caused much enthusiasm for the neural network possibilities.

A very broad comparison study was organized in the STATLOG project (Michie et al., 1994) using 22 real-world datasets and 23 different classifiers including a number based on ANN. This made clear that there is no unique best classifier. Several methods showed a good performance over a wide range of databases, among which the NNR and the feedforward neural network.

The question whether neural nets can outperform traditional techniques remains an intriguing one for many researchers. It is the topic of several discussions on the Internet, the theme of workshops and competitions (ICMS Workshop, 1995) and it enters

[*] Temporarily with the Vision, Speech and Signal Processing Group, Dept. of Electrical Engineering, Univ. of Surrey, United Kingdom. Email: duin@ph.tn.tudelft.nl

[1] This paper is based on a presentation at the ICMS Workshop on Statistics and Neural Networks, Edinburgh, April 19–20, 1995, titled "The Feedforward Network as an Automatic Classifier". The experiments presented here are new and more extensive.

the discussion sections of many research papers. It is the goal of this paper to discuss this question itself. What do we mean by: "Can neural networks outperform traditional techniques?". How might such a question be answered? What are the pitfalls and traps that should be avoided? We realize that for some researchers our conclusions might not be that surprising. Given the large set of publications, however, in which the authors struggle with the problem of presenting fair comparisons it is clear that it deserves more attention.

2. Comparison problems

At a first glance it may seem that comparing classifiers is as easy as error counting. There is, however, certainly more to say. Even if we skip the issue of computing time then still two other factors are of importance: For what application(s) will the classifier be used and by whom?

It is perfectly clear that performance differences are a function of class distributions and sample sizes and therefore of the application. If one tries to draw conclusions that are application independent and thus distribution free, then only performance bounds can be obtained, e.g. in terms of the Vapnik–Chervonenkis complexity of classifiers (Vapnik, 1982; Devroye, 1988). This is not the issue here. We are interested in the real performance for practical applications. Therefore, an application domain has to be defined. The traditional way to do this is by a diverse collection of datasets. In studying the results, however, one should keep in mind that such a collection does not represent any reality. It is an arbitrary collection, at most showing partially the diversity, but certainly not with any representative weight. It appears still possible that for classifiers showing a consistently bad behavior in the problem collection, somewhere an application exists for which they are perfectly suited.

A second issue, causing even more problems, is the dependence of the performance of some classifiers on the skill of the analyst who applies them. We will elaborate on that in the following.

Some classifiers are very flexible, with many user-adjustable parameters, others are almost entirely automatic. There will be hardly any discussion on the performance of Fisher's linear discriminant or the 1-NN rule (for a given metric) on a particular dataset. However, if somebody states that he uses an ANN based classifier, or even more specific, a multi-layer perceptron using the backpropagation rule, then there is still a wide range of possibilities. His results will be highly dependent on the architecture, the initialization procedure, his strategy of determining targets, step sizes, momentum terms and the use of weight decay. There is not such a thing as a uniquely defined neural network classifier. Software packages and textbooks give guidelines on how to establish the parameters for a particular application. The result and its performance will be user dependent. Some people might get better neural network classifiers than others on the same problem.

Almost all of the traditional classification techniques are well defined and can be used without much user interaction. This holds for linear and nonlinear discriminant analysis, the nonparametric methods as well as for decision trees. This has several consequences for the issue of comparing classifiers on a particular application.

First, the answer to the question: "Can neural networks outperform traditional techniques?" might be "Yes", if a better classifier has been found. The answer, however, can never be "No", because there is always the possibility that somebody finds a better ANN-based solution.

Second, as the neural network technology is very flexible, it can implement almost any classification function, including the ones found by traditional classifiers. A good experimenter may find this and thus obtain an at least equal performance.

Third, the use of neural networks involves a skill. It is not just the use of an off-the-shelf algorithm for which it is sufficient being able to read the manual in combination with some general experimentation capabilities. It demands training, awareness of all types of pitfalls and a special art for tuning the parameters.

Fourth, as tuning the design parameters involves a lot of experimentation and as datasets are finite, there is always the possibility that a good classifier is found by chance. In order to avoid this, the datasets should be divided into three subsets: one for training, one for tuning and one for testing. For a fair comparison, the last one should be used only once for all classifiers, at the moment they have been established. There will be a strong temptation for the researcher to do some more tuning when he finds out that his neural network performs relatively badly. Papers that emphasize that this has not been done are very rare. On the contrary, one regularly may read that the test performance improved as a result of modifying the algorithm. So the test set is used as a training set and performances estimated by it are biased.

In summary, for an arbitrary application it is to be expected that a neural network-based classification function exists that performs at least as good as traditional classifiers. If this has not been found, it is unlikely that this result will appear in print as it throws doubt on the capabilities of the experimenter.

This might be the end of the comparison issue. We just establish that neural networks are not a classifier but a large and complicated toolset and the result does heavily depend on the analyst or artist using the tools. This is the strength and the weakness of this technique. If you know how to use the tools you can do a lot, if not, better stay away and use traditional automatic tools.

After realizing the above, there may still something to do. We might investigate (1) what an expert can reach using this toolset, (2) what the value of the toolset is for an arbitrary researcher and (3) what automatic tools can be built using this toolset. These three possibilities will be worked out in some more detail below.

If we assume that the result mainly depends on the expert using the toolset and not on what is in the toolset, a *comparison between experts* (or groups of experts) is more appropriate than one between methods. This implies that a set of datasets has to be sent to a number of experts. They are free to use whatever technique they want to find good classifiers. These classifiers are collected and independently tested. A report of the experts explaining what they have done will indicate the usefulness of the particular tools for them. Such competitions have been organized in the past, e.g. during the 3rd International Pattern Recognition Conference in 1976, but often on just a single dataset. Below it is argued that this is not very informative.

In order to find out the *value of particular toolsets*, including their description, possibilities for user interaction, etcetera, the following test might be done. Assume that we have a small number of toolsets, some based on traditional techniques, some on neural networks only and some mixtures. Now each toolset is sent to a number, say ten, researchers, together with the datasets in order to find out what the value of the toolset is for "the average researcher". In this scenario it is necessary that each participant gets a toolset at random, not his favorite one, as this is a type of consumer test on toolsets.

Finally, it can be argued that if a toolset is good, one should also be able to build an *automatic classifier* with it. This, of course, eliminates the possibility for the user to incorporate his skills and a priori knowledge of the problem. Nevertheless, it is still interesting to investigate what can be reached by such an automatic classifier, as it might help to find out and understand what types of approaches are useful for what problems. Most neural network packages have as a default one or more of these classifiers. This will be discussed further in the next section.

Of these three possibilities, the expert contest, the consumer test and the automatic classifier comparison, only the last one can be done by a single researcher. The other two have to be organized at large. None of them will straightforwardly answer the question whether neural nets are better than traditional techniques. They will, however, all give a contribution to a final answer.

One of the largest comparisons recently undertaken, the STATLOG project (Michie et al., 1994), has already been mentioned shortly in the introduction. Due to the large number of methods included, the wide variability of datasets and the careful procedures followed, and the extensive report of the results, this is a very valuable experiment. One can find out for which types of problems what methods score well and which don't and see that some are almost always bad and some are very often among the good performers. In terms of the above three possibilities, this project is a mixture of an expert contest and an automatic classifier comparison. Basically each method was run by another expert. As a guideline they were supposed to adjust as few parameters as possible. In the discussions, however, one can read that especially for neural networks this is still a subtle process that may heavily influence the results. Consequently, the results may be different if the methods are rotated over the experts.

3. The automatic classifier

For a given labeled dataset, an automatic classifier finds without any user interaction a discriminant function. It can thus be used by anybody. It has of course the disadvantage that it will probably perform worse than a skilled analyst using a good toolset. Automatic classifiers may also be of value as a reference to such analysts. For less demanding problems they give a fast and easily obtained result.

We are here interested in automatic classifiers as their performance in a given application area or on a given set of problems can easily and objectively be obtained. If one wants to use the result of such a comparison for the scientific study of classification algorithms and strategies, then it is useful to add one more demand: an automatic classifier should be well defined on a clear concept and should not include disputable parameter choices. If it does not fulfill this condition, the result of the comparison might not be generally accepted and will not clearly show the value of particular concepts.

Examples of automatic classifiers are the nearest mean classifier, the linear and quadratic discriminants based on normal density assumptions, the 1-nearest neighbor rule, the k-nearest neighbor rule with optimization of k, using the leave-one-out error estimator and the Parzen density classifier using leave-one-out maximum likelihood optimization of the smoothing parameter for each class. Some of these classifiers are metric dependent. It has therefore to be assumed that the metric is given. Automatic scaling methods should be studied separately and in addition to the classification.

Decision trees are somewhat more problematic. For binary decision trees, well defined criteria are available, such as maximum information gain or the gini criterion. The problem, however, is the size of the tree. If a tree is grown until its natural end of zero error, it is oversized (over-trained) and might perform badly on an independent test set. Pruning or early stopping is necessary. There is a lot of discussion possible on the best strategies.

Neural networks are very problematic for designing clear automatic classifiers. A large part of the research in this area is focused on strategies for the selection of an architecture and regularization during training. The training rules themselves show also a very large variability: fixed or variable step sizes, targets, momentum terms, weight decay, first- or second-order techniques, addition of noise in various places and averaging results. For the moment each proposal for an automatic classifier based on neural networks seems disputable. In (Schmidt et al., 1994) we experimented with a conjugated gradient descent technique, avoiding the choice for step size and momentum term. Stopping was defined by a fixed number of line optimizations. The only free parameter was the number of hidden units. This classifier showed bad results compared with the nearest neighbor classifier. During various presentations it was remarked that for the presented datasets much better neural network classification results than ours existed. This is true. It should be emphasized, however, that we want to have a constant classifier over a set of problems and that this classifier is not to be changed after the comparison started.

In the next section an example is presented for which we used 'trainbpx', one of the neural network classifiers of the Neural Network Toolbox of the MATLAB package. We have good experiences with this backpropagation feedforward classifier (Hoekstra et al., 1995) and used it here with its default parameter setting as defined by MATLAB. This method is not exactly what we mean by an automatic classifier. It has an adaptive learning rate that often produces fast and good results but that appears rather 'engineered' and is certainly not a clearly defined and implemented concept. We use it in our example because it is a 'state-of-the-art' method and is widely used with the MATLAB package.

Two choices for this classifier are still open: the architecture and the stopping rule. We decided to use a single hidden layer with a fixed number of 20 hidden units. For most problems with a moderate feature size this will be sufficient. For small sample sizes, the danger exists that the networks will be overtrained. In order to prevent this an early stopping rule is used based on an artificial tuning set having a size of five times the training set. This tuning set is generated from Parzen density estimates for the classes (Duin, 1976). With this set the neural network performance is evaluated during training using the error count. Training is stopped when the iteration number is twice that during which the last improvement occurred.

4. An example

We will now present a small comparison of six classifiers. It is not our purpose to draw any definite conclusions from this. The goal of this experiment is mainly to serve as an illustration. The following classifiers are used in a MATLAB implementation. They were entirely developed, including the automatic choice of parameters, before they were run on the datasets used for the comparison.

-*NMean*, the nearest mean rule. Each object is assigned to the class of the nearest class mean computed from the training set. This is a very simple basic rule, very fast to compute and use. It does not, however, take into account any other distribution information than the mean. Moreover, it cannot be made dependent on the a priori class probabilities, nor on differences in class frequencies in the training set.

-*Norm*, the Bayes rule assuming normal densities and an equal covariance matrix for all classes. This covariance matrix is estimated as the average of the within-class covariance matrices. This method can handle different a priori class probabilities. We estimated these probabilities by the observed class frequencies in the training set.

-*1-NNR*, the 1-nearest neighbor rule. Each incoming object is assigned to the class of its nearest neighbor in the training set. This procedure is sensitive for different class frequencies in the training set, which, however, cannot be taken into account.

-*k-NNR*, the k-nearest neighbor rule. In this case the highest frequency of the class labels of the k-nearest neighbors is used for assigning a label to a new object. We estimate k automatically by selecting the best result over all k of the leave-one-out error estimates for the training set.

-*DTree*, binary decision tree. We use the information gain criterion for growing an error-free tree for the training set and then prune it by the pessimistic pruning technique, see (Quinlan, 1986, 1987). This procedure is sensitive for different class frequencies in the training set.

-*ANN*, The neural network classifier described above, which is also sensitive for different class frequencies in the training set.

The following datasets were used.

-IRIS, Fisher's Iris Set (Fisher, 1936).

-IMOX, extracted from Munson's handprinted character set, see (Jain and Ramaswami, 1988).

-80X, also extracted from Munson's handprinted character set, see (Jain and Ramaswami, 1988).

-BLOOD, as published by the American Statistical Association, see (Cox et al., 1982). We used the patient's age as a feature and removed all samples with missing data fields.

-SONAR, a collection of sonar signals bounced off a metal cylinder and a roughly cylindrical rock (Gorman and Sejnowski, 1988).

-GLASS, a collection of glass fragments of different origin used for forensic investigations. We regrouped the classes as proposed in (Ripley, 1994).

-DNORM, an artificial 30-dimensional dataset of two normally distributed classes. The classes are initially generated with means $\mu_A = (0, 0, 0, \ldots, 0)$ and $\mu_B = (3, 3, 0, 0, \ldots, 0)$ and with a diagonal covariance matrix having $(1, 40, 1, 1, \ldots, 1)$ on its diagonal. Subsequently the dataset is rotated in $\mathbb{R}^{30}$ obtaining high feature correlations. As the vector between the class means is not perpendicular to one of the eigenvectors of the covariance matrix, simple procedures like the nearest mean rule fail. This example is constructed such that most classifiers yield a much larger error than the class overlap of 0.064. See also (Duin, 1995).

The dataset sizes, their number of classes and features are listed in Table 1. For each dataset the following procedure is followed. The dataset is divided at random in two equally sized subsets. No special care is taken that classes are represented evenly. From this training set an artificial tuning set is generated of five times the size of the training set. This is done from a pseudo-maximum likelihood Parzen density estimation of the training set (Duin, 1976). This tuning set is used for stopping the training of the neural network. The test set is used for

Table 1
Feature and sample sizes for the datasets

Dataset	No. of features	No. of classes	Total no. of samples	No. of samples per class
IRIS	4	3	150	50,50,50
IMOX	8	4	192	48,48,48,48
80X	8	3	45	15,15,15
BLOOD	5	2	194	127,67
GLASS	9	4	214	70,76,17,51
SONAR	60	2	208	97,111
DNORM	30	2	200	100,100

estimating the error rates. The entire procedure is repeated 10 times.

In Table 2, the averaged results are shown as well as the standard deviations over the 10 runs. The best result for each dataset is underlined. Results that deviate significantly from the best result significantly based on the standard deviations are listed in italics. If this holds for all other results the best result is printed in bold. Note that judging the significance of differences on the basis of the standard deviations is in fact a too simple procedure, since the error estimates are based on the same testsets and are therefore dependent.

The first striking point in studying Table 2 is that although the datasets have all relatively small sample and feature sizes, the results are very different. There is no uniformly best procedure. The nearest mean method and the decision tree perform badly on these datasets. It is, however, easy to construct artificial examples for which they do well. If, for instance, the correlation in the DNORM problem is removed by rotating the data, the decision tree is best while the other methods keep the same performance. If we shift the class mean difference to one of the eigenvectors, the nearest mean method performs very well.

Table 2
Averaged error rates and standard deviations over 10 runs

Dataset	NMean	Norm	*k*-NNR	1-NNR	DTree	ANN
IRIS	*0.077*	**0.025**	*0.048*	*0.053*	*0.071*	*0.052*
	0.019	0.010	0.019	0.017	0.031	0.026
IMOX	*0.115*	*0.102*	0.086	0.071	0.092	0.088
	0.027	0.026	0.018	0.023	0.045	0.031
80X	0.114	0.123	0.077	0.082	*0.255*	0.118
	0.054	0.074	0.083	0.088	0.099	0.078
BLOOD	*0.163*	0.125	0.131	*0.153*	*0.158*	0.123
	0.034	0.035	0.035	0.041	0.048	0.033
GLASS	*0.569*	*0.431*	0.303	0.286	*0.334*	*0.380*
	0.049	0.098	0.040	0.045	0.052	0.075
SONAR	*0.352*	*0.315*	0.194	0.188	*0.307*	*0.236*
	0.072	*0.061*	0.050	0.044	0.043	0.034
DNORM	*0.334*	*0.151*	*0.185*	*0.212*	*0.344*	**0.121**
	0.045	0.041	0.045	0.038	0.045	0.017

If we compare the neural network procedure with the nearest neighbor methods, then there is no dataset where the neural network performs significantly better, except for the artificial DNORM dataset. This was deliberately constructed such that almost any method would perform badly. It is interesting that we have thus at least one example for which the neural network classifier is significantly better than all other classifiers. The training effort necessary for the neural network classifier is about 10 to 100 times larger than for any other classifier.

5. Discussion

The above example illustrates and confirms what has already been found by others (Michie et al., 1994), i.e., that there is not such a thing as a best classifier. Moreover, for almost any classifier a problem can be selected or constructed for which it does well. Researchers presenting new or modified classifiers should realize that. As a consequence, their comparisons should include a diverse set of alternative classifiers and should cover a number of real-world datasets, preferably sets already studied in the literature.

The pitfalls that might show up when one is too sloppy with this can be illustrated by results in Table 3, where we compare the above-used neural network classifier with a modification. This modified classifier has one additional step in which the network is pruned using the same tuning set used for the early stopping procedure. In this pruning step all hidden units are removed for which the classification error on this tuning set decreased after removal. If we look at the overall results, this modification is not an improvement. For two datasets, GLASS and SONAR it is even significantly worse.

Table 3
Results for a pruned neural network classifier (averaged error rates and standard deviations over 10 runs)

Dataset	ANN	Pruned ANN
IRIS	0.052	0.047
	0.026	0.032
IMOX	0.088	0.090
	0.031	0.030
80X	0.118	0.100
	0.078	0.074
BLOOD	0.123	0.162
	0.033	0.066
GLASS	**0.380**	*0.547*
	0.075	0.119
SONAR	**0.236**	*0.350*
	0.034	0.129
DNORM	0.121	0.114
	0.017	0.023

If we imagine that we just present the results on IRIS, 80X and DNORM only, the pruned method seems better and if we 'forget' the standard deviations or run a number of additional trials a 'real' improvement might show up. No conscious researcher, of course, would do such a thing, selecting just the results that show improvements. But as hundreds of researchers and students are constantly trying to improve existing methods, these things might easily happen by 'accident', due to the random process of selecting datasets, training sets and, in case of neural networks, initializations. One way to prevent this is to use standard benchmark datasets. Here again a trap might arise. In the long run such datasets wear out as they start to become training sets for new methods. A general tendency will be that the seemingly best methods belong to the most complicated ones, as these are heavily and frequently tested and modified thereby adapting the most to the datasets. As a consequence, a standard benchmark should not only include a collection of diverse problems (in order to cover all aspects of real-world applications), and be large (in order to get significant results), but it should also be renewed from time to time. The collections of datasets gathered for the STATLOG project (Michie et al., 1994) and in (Prechelt, 1994) may be good starting points for defining benchmarks.

6. Conclusions

In comparing classifiers one should realize that some classifiers are valuable because they are heavily parameterized and thereby offer a trained analyst a large flexibility in integrating his problem knowledge in the classification procedure. Other classifiers, on the contrary, are very valuable because they are entirely automatic and do not demand any user parameter adjustment. As a consequence they can be used by anybody. It is therefore difficult to compare these types of classifiers in a fair and objective way. Three possibilities are proposed:

1. We accept that classifiers are user dependent and we focus on a comparison of experts rather than of classifiers. In this proposal a collection of problems is sent to each of a set of experts. The results compare their skills. Their reports, describing which classifiers they have used and in what way, may show which techniques and procedures are the most valuable ones.

2. We compose toolsets containing classifiers and user instructions and send each of them to a different set of nonexperts, accompanied with always the same collection of problems. These toolsets have a different character, e.g. one using density estimators, one based on decision trees and a neural network-based toolset. As the results are averaged over sets of users, they may show us which toolset is for what problem the best to be used by an arbitrary user.

3. We restrict ourselves to automatic classifiers, realising that this neglects some possibilities of some classifiers. A comparison can now be performed by a single researcher on a benchmark set of problems provided that:

(a) the classifiers are defined entirely before they are run on the benchmark;

(b) the benchmark set contains a variety of problems;

(c) the benchmark set is large enough to make results significant;

(d) benchmarks are used sparsely and then renewed.

As a consequence of the first demand, modifications of classifiers performing badly, should not be developed on the benchmark. As a consequence of the last demand, modifications have to be proposed sparsely.

Each of these three possible types of comparisons will contribute in its own way to the evaluation of the value of neural networks for classification purposes. Because of the intrinsic complexity and flexibility of the technique, a direct, straightforward answer cannot be obtained. For the time being the problem whether neural networks are better than traditional techniques might be undecidable.

Acknowledgements

This paper was prepared while the author visited the Department of Electrical Engineering, University of Surrey, United Kingdom in the spring of 1995. The author likes to thank Prof. Josef Kittler and the members of the Vision, Speech and Signal Processing Group most cordially for their hospitality during his stay. The author also likes to thank Wouter Schmidt for stimulating discussions and for preparing most of the datasets.

References

Cheng, B. and D.M. Titterington (1994). Neural networks: a review from statistical perspective. *Statistical Sci.* 9 (1), 2–54.

Cox, L.H., M.M. Johnson and K. Kafadar (1982). Exposition of statistical graphics technology. *ASA Proc. Statistical Computation Section*, 55–56.

Devroye, L. (1988). Automatic pattern recognition: a study of the probability of error. *IEEE Trans. Pattern Anal. Mach. Intell.* 10 (4), 530–543.

Duin, R.P.W. (1976). On the choice of the smoothing parameters for Parzen estimators of probability density functions. *IEEE Trans. Comput.* 25 (11), 1175–1179.

Duin, R.P.W. (1995). Small sample size generalization. *SCIA'95, Proc. 9th Scandinavian Conf. on Image Analysis* (Uppsala, Sweden, June 6–9, 1995), Vol. 2, 957–964.

Fisher, R.A. (1936). The use of multiple measurements in taxonomic problems. *Ann. Eugenics* 7, 280–322.

Golomb, B.A., D.T. Lawrence and T.J. Sejnowski (1991). Sexnet: A neural network identifies sex from human faces. In: Lippmann et al., Eds., *Advances in Neural Information Processing Systems 3*. Morgan Kaufman, San Mateo, CA, 572–577.

Gorman and T.J. Sejnowski (1988). Learned classification of sonar targets using massively parallel network. *IEEE Trans. Acoust. Speech Signal Process.* 36 (7), 1135–1140.

Hoekstra, A., S.A. Tholen and R.P.W. Duin (1995). Confidence Value Estimation in Neural Networks. Internal report.

ICMS Workshop (1995). *ICMS Workshop on Statistics and Neural Networks*. Edinburgh, April 19–20, 1995.

Jain, A.K., and M.D. Ramaswami (1988). Classifier design with Parzen windows. In: E.S. Gelsema and L.N. Kanal, Eds., *Pattern Recognition and Artificial Intelligence*. North-Holland, Amsterdam, 211–228.

Kamata, S., R.O. Eason, A. Perez and E. Kawaguchi (1992), A neural network classifier for LANDSAT image data. *Proc. 11th Internat. Conf. on Pattern Recognition*, Vol. 2, 573–576.

Michie, D., D.J. Spiegelhalter and C.C. Taylor (1994). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, New York.

Prechelt, L. (1994). A Study of Experimental Evaluations of Neural Network Learning Algorithms: Current Research Practice. Technical Report 19/94.

Quinlan, J.R. (1987). Induction of decision trees. *Machine Learning* 1, 81–106.

Quinlan, J.R. (1987). Simplifying decision trees. *Internat. J. Man-Machine Studies*, 27, 221–234.

Ripley, B.D. (1994). Neural networks and related methods for classification. *J. Roy. Statist. Soc. B* 56 (3), 409–456.

Sabourin, R. and J.P. Drouhard (1992). Off-line signature verification using directional PDF and neural networks. *Proc. 11th Internat. Conf. on Pattern Recognition*, Vol 2, 321–325.

Schmidt, W.F. (1994). *Neural Pattern Classifying Systems*. Ph.D. Thesis, Pattern Recognition Group, Department of Applied Physics, Delft University of Technology, ISBN 90-9006716-7.

Schmidt, W.F., D.F. Levelt and R.P.W. Duin (1994). An experimental comparison of neural classifiers with traditional classifiers. In: E.S. Gelsema and L.N. Kanal, Eds., *Pattern Recognition in Practice IV: Multiple Paradigms, Comparative Studies and Hybrid Systems*. North-Holland, Amsterdam, 391–402.

Sejnowski, T.J. and C.R. Rosenberg (1988). NETtalk: a parallel network that learns to read aloud. The Johns Hopkins University Electrical Engineering and Computer Science Technical Report JHU/EECS-86/01, 1986. Reprinted in: J.A. Anderson and E. Rosenfeld, Eds., *Neurocomputing: Foundations of Research*. MIT Press, Cambridge, MA.

Vapnik, V. (1982). *Estimation of Dependences based on Empirical Data*. Springer, New York.