Pattern classification *

Thierry Denœux

U.M.R. CNRS 6599 Heudiasyc
Université de Technologie de Compiègne
BP 20529 - F-60205 Compiègne cedex - France
email: Thierry.Denoeux@hds.utc.fr

Abstract

Pattern classification consists in assigning entities, described by feature vectors, to predefined groups of patterns. When the statistical characteristics of the problem under consideration are perfectly known, minimal error probability can be achieved by means of the Bayes decision rule. In practice, however, a sub-optimal classifier has to be constructed from training data. Several neural network approaches to this problem have been proposed. *Nearest-neighbor* models are based on assessing the similarity between the input pattern and a set of reference patterns with known classification. The *regression* approach consists in predicting category from pattern by minimizing a certain error criterion. In the finite sample case, the definition of the structural complexity of these models is shown to have considerable influence on classification error. Finally, a taxonomy of the main neural network and alternative techniques to pattern classification is presented.

1 Introduction

In many application domains such as character recognition, speech understanding, medical diagnosis, process fault detection or financial decision-making, problems arise that consist in classifying entities, represented by feature vectors, into one of several groups of patterns, or *classes*. A classification system is typically composed of two parts [10, 13]. A *preprocessor* transforms raw data produced by sensors or extracted from computer data bases into vectors of *observations* or *features*. Features are defined so as to encode in compact form most of the information needed to discriminate between pattern categories. Feature vectors are then passed to a *classifier* that evaluates the evidence presented and makes a decision regarding the class assignment of the entity under consideration.

Ever since the pioneering work of Rosenblatt [39] and Widrow [49], a large part of connectionist research has been devoted to the development and theoretical analysis of pattern classifiers having neural network-like structure and

*In E. Fiesler and R. Beale, editors, *Handbook of Neural Computation*. Oxford University Press and Institute of Physics Publishing, pages F1.2:1–8, 1996.

learning capabilities. In recent years, the development of several new models with previously unequaled performance in real-world applications [40, 20] has generated a wave of interest in connectionism and pattern recognition in general. Although this enthusiasm was first considered with some skepticism by researchers in mainstream statistical pattern recognition [11], artificial neural networks are now generally seen as particular types of statistical pattern classifiers [47, 41].

In the next section, the basic notations and definitions underlying statistical pattern recognition will first be defined. The main neural network approaches to pattern classification will then be described, with an overview of their asymptotic and small sample properties. In the last section, a taxonomy of statistical and neural network classifiers will be presented.

2 Problem description

We consider a finite number M of populations or classes, $\omega_1, \ldots, \omega_M$. An entity of interest is assumed to belong to one and only one of these populations. Each entity is described by a feature vector $\mathbf{x} \in \mathbb{R}^d$ which is seen as a realization of a random vector $\mathbf{X}$. The probability density function of $\mathbf{X}$ in class ω_i is denoted by $f_X(\mathbf{x}|\omega_i)$. Each entity is generally assumed to be drawn from a mixture of the M populations, in proportions $P(\omega_1), \ldots, P(\omega_M)$, respectively, with $\sum_{i=1}^{M} P(\omega_i) = 1$. The mixture density of X is then:

$$f_X(\mathbf{x}) = \sum_{i=1}^{M} P(\omega_i) f_X(\mathbf{x}|\omega_i) \tag{1}$$

$P(\omega_i)$ can be seen as the prior probability that the entity belongs to ω_i. Having observed feature vector $\mathbf{x}$, the posterior probability $P(\omega_i|\mathbf{x})$ can be computed by applying the Bayes theorem:

$$P(\omega_i|\mathbf{x}) = \frac{f_X(\mathbf{x}|\omega_i) P(\omega_i)}{f_X(\mathbf{x})} \tag{2}$$

If the class-conditional probability distributions and the priors are all known, then an optimal solution to the classification problem is provided by Bayes decision theory. Let us denote by $A = \{\alpha_1, \ldots, \alpha_a\}$ a finite set of actions. α_i is often interpreted as the decision of allocating $\mathbf{x}$ to class ω_i. However, other actions such as ambiguity or distance reject [6, 9] can also be considered in the analysis.

If, as a result of observing pattern $\mathbf{x}$, we take action α_i while the entity under consideration belongs to class ω_j, we incur a loss $\lambda(\alpha_i|\omega_j)$. The expected loss $R(\alpha_i|\mathbf{x})$ is:

$$R(\alpha_i|\mathbf{x}) = \sum_{j=1}^{M} \lambda(\alpha_i|\omega_j) P(\omega_j|\mathbf{x}) \tag{3}$$

A *decision rule* is a function $\alpha : \mathbb{R}^d \mapsto A$ that prescribes an action $\alpha(\mathbf{x})$ each time an observation vector $\mathbf{x}$ is encountered. The overall risk associated to α

is:

$$R(\alpha) = \int_{\mathbb{R}^d} R(\alpha(\mathbf{x})|\mathbf{x}) f_X(\mathbf{x}) d\mathbf{x} \tag{4}$$

The decision rule that minimizes the risk can be shown to be the *Bayes rule*, which selects for each vector $\mathbf{x}$ the action α_i for which $R(\alpha_i|\mathbf{x})$ is minimum.

In the particular case of a zero-one loss function $\lambda(\alpha_i|\omega_j) = 1 - \delta_{ij}$, where δ is the Kronecker symbol, we have:

$$R(\alpha_i|\mathbf{x}) = 1 - P(\omega_i|\mathbf{x}) \tag{5}$$

and the overall risk is the average probability of misclassification. Consequently, the Bayes rule consists in that case in selecting the class with the highest posterior probability. This rule has optimal classification performance in the sense that it minimizes the average probability of error.

In practice, however, this rule cannot be applied because the exact posterior probabilities are unknown. However, approximations to that rule can be constructed if a training set $\mathcal{T} = \{(\mathbf{x}^{(1)}, \mathbf{t}^{(1)}), \ldots, (\mathbf{x}^{(\ell)}, \mathbf{t}^{(\ell)})\}$ of ℓ patterns with known classification is available. $\mathbf{t}^{(i)}$ denotes a vector of M zero-one indicator variables defining the known class of pattern $\mathbf{x}^{(i)}$:

$$\mathbf{t}_k^{(i)} = 1 \quad \mathbf{x}^{(i)} \in \omega_k \tag{6}$$

$$\mathbf{t}_k^{(i)} = 0 \quad \mathbf{x}^{(i)} \notin \omega_k \tag{7}$$

The construction of allocation rules based on a limited amount of training data is one of the fundamental problems in statistical pattern recognition and connectionism.

3 Neural network classifiers

In the past thirty years, a large number of neural network models have been proposed for performing pattern classification tasks. Although these models are characterized by a variety of architectures and learning rules, most of them can be seen as instances of two main paradigms, the *nearest-neighbor* approach and the *regression* approach, which are summarized in the following sections.

3.1 The nearest-neighbor approach

In the nearest-neighbor approach, the most probable classification of an unknown pattern is determined by assessing its similarity with a set of reference vectors or *prototypes* of each class. The pattern is assigned to the class of the nearest prototype. As a consequence, the surface separating the different decision regions is piecewise linear. In such models, learning is essentially a process of prototype formation and adaptation. Two important models in this category are the *Restricted Coulomb Energy* (RCE) [38] and *Learning Vector Quantization* (LVQ) [20] networks.

In the RCE model, each prototype of a given class is characterized by a weight vector and a receptive field size. The learning algorithm combines two

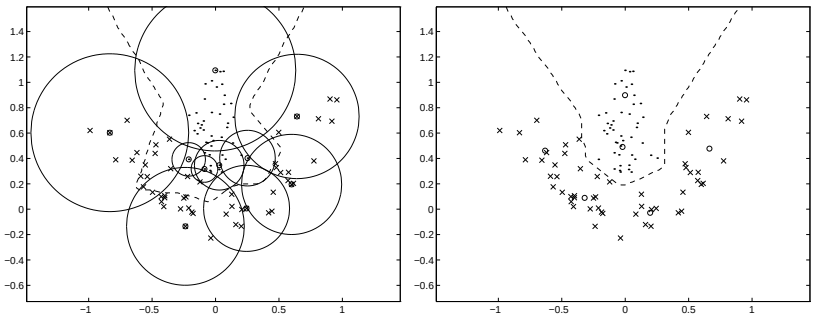

Figure 1: Prototypes (o) and decision boundaries (- -) obtained by RCE (left) and LVQ (right) in a two-class problem. The receptive fields of RCE prototypes are indicated as circles.

mechanisms of prototype formation and receptive field modification. If input $\mathbf{x}$ belonging to class ω_j does not fall into the receptive field of any prototype of that class, then a new prototype of class ω_j is created at the location of $\mathbf{x}$. If $\mathbf{x}$ falls inside the receptive field of some prototype of class $c \neq \omega_j$, then the receptive field of that prototype is reduced so as to exclude $\mathbf{x}$. This algorithm has been shown experimentally to be able to resolve class boundaries of arbitrary complexity. However, since no adaptation of prototype vectors is performed, the required number of prototypes may grow very large. Also, the learning process usually becomes unstable in regions where there is a strong overlap between classes. Some improvements to this basic model have been proposed [38].

The LVQ model introduced by Kohonen [20, 21] essentially differs from the previous one in that the number of prototypes is fixed, but their weight vectors are continuously updated in the course of the learning process by a *competitive learning* mechanism. Upon presentation of input vector $\mathbf{x}$ of class ω_j, the nearest prototype i is selected. If that prototype belongs to class $c^{(i)}$, its weight vector $\mathbf{p}^{(i)}$ is updated as:

$$\mathbf{p}^{(i)} \leftarrow \mathbf{p}^{(i)} + \eta(t)(\mathbf{x} - \mathbf{p}^{(i)}) \quad \text{if } c^{(i)} = \omega_j \tag{8}$$

$$\mathbf{p}^{(i)} \leftarrow \mathbf{p}^{(i)} - \eta(t)(\mathbf{x} - \mathbf{p}^{(i)}) \quad \text{if } c^{(i)} \neq \omega_j \tag{9}$$

where $\eta(t)$ is a time-decreasing scalar parameter ($0 < \eta(t) < 1$). After training, the prototype vectors acquire values such that classification using the nearest-neighbor principle approximates the Bayes rule with zero-one costs. Variants of this basic scheme have been proposed by Kohonen [21] and others (e.g. [33]).

Simulations performed with both models (RCE and LVQ) on a simple two-class problem are reported in figure 3.1. The LVQ algorithm can be seen to yield a smoother decision boundary with a comparatively smaller number of neurons, as a result of prototype adaptation during training.

Neural-network classifiers based on the nearest neighbor approach present the advantage of being fast during both training and operation. Experimentally, they are generally found to offer good performance as compared to other, more computationally demanding methods [22]. Although it is conjectured that

the methods relying on competitive learning allow approximation to the Bayes rule for large sample sizes [21], the determination of the quality of this approximation is a difficult theoretical problem. When classification is performed by considering the nearest neighbor *among samples*, the asymptotic error rate is known to be bounded between the Bayes error and twice the Bayes error [7]. This result can be seen as a heuristic justification of the good performance of nearest neighbor techniques in large sample problems.

3.2 The regression approach

Classification by regression is certainly the most popular approach in the field of artificial neural networks. A regression classifier attempts to predict category from pattern by minimizing a measure of expected error between output and target patterns [43].

More precisely, let us denote the input-output function implemented by a neural network with specified architecture by:

$$F: \quad \mathbb{R}^d \times \mathbb{R}^W \quad \mapsto \mathcal{O} \tag{10}$$

$$(\mathbf{x}, \mathbf{w}) \quad \to F(\mathbf{x}, \mathbf{w}) \tag{11}$$

where $\mathcal{O}$ is the set of possible output values and $\mathbf{w}$ the vector of weights of size W.

In the case of *multilayer layer perceptrons* (MLPs) [40] with one hidden layer and logistic activation function in the hidden layer, the k-th component $F_k(\mathbf{x}, \mathbf{w})$ of output vector $F(\mathbf{x}, \mathbf{w})$ is defined as:

$$F_k(\mathbf{x}, \mathbf{w}) = \sum_{j=1}^{N_2} w_{kj}^{(2)} \sigma \left(\sum_{i=1}^{d} w_{ji}^{(1)} x_i + \theta_j^{(1)} \right) + \theta_k^{(2)} \tag{12}$$

where $w_{ji}^{(1)}$ is the weight from input unit i to hidden unit j, $\theta_j^{(1)}$ the bias of hidden unit j, $w_{kj}^{(2)}$ the weight from hidden unit j to output unit k, $\theta_k^{(2)}$ the bias of output unit k, N_2 the size of the hidden layer, and σ a sigmoid function.

In the case of *radial basis function* (RBF) networks [32, 14], the output from hidden unit j is defined as a function of the Euclidean distance between input $\mathbf{x}$ and a prototype vector $\mathbf{p}^j$. As in the previous model, output units compute a weighted sum of the outputs from the hidden layer. The output $F_k(\mathbf{x}, \mathbf{w})$ from unit k is given by:

$$F_k(\mathbf{x}, \mathbf{w}) = \sum_{j=1}^{N_2} w_{kj} \exp \left(-\frac{1}{2\sigma_j^2} \|\mathbf{x} - \mathbf{p}^{(j)}\|^2 \right) \tag{13}$$

where w_{kj} is the weight from hidden unit j to output unit k, σ_j is a parameter defining the size of the receptive field of prototype j, and N_2 is defined as above.

An important distinction between MLPs and RBF networks concerns the nature of the internal representation of input patterns. In MLPs, an input signal may activate an arbitrary number of hidden units, resulting in *distributed* representation. In RBF networks, one input predominantly activates the hidden

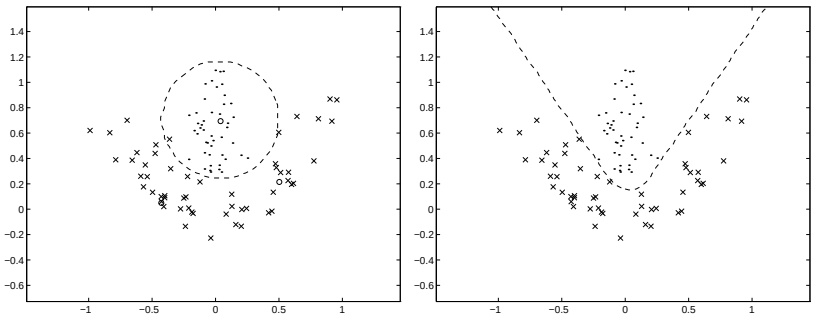

Figure 2: Decision boundaries (- -) obtained by a RBF network with three prototype units (left) and by a MLP with two hidden units (right) in a two-class problem.

unit with the closest weight vector, which creates a *local* representation. From this point of view, RBF networks are related to the nearest-neighbor classifiers described in the previous section. A comparison of both models on the same two-class problem as above is shown in figure 3.2.

Both MLPs and RBF networks share the fundamental property of being universal approximators, that is, given enough hidden units, they can approximate any continuous mapping with arbitrary accuracy [32, 18].

In the MLP and RBF network models, training is performed by optimizing the performance on a training set $\mathcal{T} = \{(\mathbf{x}^{(1)}, \mathbf{t}^{(1)}), \ldots, (\mathbf{x}^{(\ell)}, \mathbf{t}^{(\ell)})\}$, using some iterative procedure [40]. Performance is assessed by computing the mean of some error measure between the classifier output and target values. Different output coding schemes and error measures have been proposed. Typically, the desired output for training vector $\mathbf{x}^{(i)}$ is taken as $\mathbf{t}^{(i)}$, and the error for that pattern is defined as $\|\mathbf{t}^{(i)} - F(\mathbf{x}^{(i)}, \mathbf{w})\|^2$. The empirical performance on the training set is then:

$$J_\ell(\mathbf{w}) = \frac{1}{\ell} \sum_{i=1}^{\ell} \|\mathbf{t}^{(i)} - F(\mathbf{x}^{(i)}, \mathbf{w})\|^2 \tag{14}$$

During training, one seeks a weight vector $\mathbf{w}_\ell$ solution of the problem:

$$\min_{\mathbf{w}} J_\ell(\mathbf{w}) \tag{15}$$

However, in most cases, the ultimate goal of learning is in fact to minimize the overall performance for any possible input vector, which can be measured by:

$$J(\mathbf{w}) = E(\|\mathbf{T} - F(\mathbf{X}, \mathbf{w})\|^2) \tag{16}$$

where $\mathbf{X}$ is a random input vector and $\mathbf{T}$ the corresponding random target vector. For large ℓ, $J_\ell(\mathbf{w})$ can be seen as an approximation to $J(\mathbf{w})$, and $\mathbf{w}_\ell$ approximates the solution $\mathbf{w}^*$ to:

$$\arg\min_{\mathbf{w}} J(\mathbf{w}) \tag{17}$$

If the training set is now seen as a realization of a random sample

$$((\mathbf{X}^{(1)}, \mathbf{T}^{(1)}), \ldots, (\mathbf{X}^{(\ell)}, \mathbf{T}^{(\ell)})) \tag{18}$$

then $J_\ell(\mathbf{w})$ and $\mathbf{w}_\ell$ become realizations of random variables $\hat{J}_\ell(\mathbf{w})$ and $\hat{\mathbf{w}}_\ell$, respectively. White [48] discusses conditions on which the sequence of real-valued random variables $\hat{\mathbf{w}}_\ell$ converges, in some strict mathematical sense, to $\mathbf{w}^*$.

So far, we have assumed performance to be assessed by a measure of the distance between desired and obtained output patterns. Intuitively, a classifier whose outputs are close to target values for each $\mathbf{x}$ can be expected to have low error probability. As the number ℓ of training vector becomes infinitely large, it is interesting to study the relationships of this approach with the Bayes rule. This has been done by many authors [48, 15, 24, 43]. The main result is that $\mathbf{w}^*$ minimizes:

$$\int_{\mathbb{R}^d} \|E(\mathbf{T}|\mathbf{x}) - F(\mathbf{x}, \mathbf{w})\|^2 d\mathbf{x} \tag{19}$$

By definition of $\mathbf{T}$, $E(T_j|\mathbf{x}) = P(T_j = 1|\mathbf{x}) = P(\omega_j|\mathbf{x})$. Consequently, $F_j(\mathbf{w}^*, \mathbf{x})$ is a mean-squared approximation to the posterior probability $P(\omega_j|\mathbf{x})$. A classifier trained by minimization of the mean-squared error criterion therefore approximates the Bayes rule asymptotically in ℓ. This result has been extended to other error functions in [15] and to more general output coding schemes in [43]. Note however that the quality of this approximation depends on the architecture of the network under consideration, as well as on the training procedure employed, which may not be able to reach a global minimum of the error function.

3.3 Small sample problems

As remarked by Raudys and Jain [35], the asymptotic classification error of a regression classifier, assuming a perfect training algorithm, cannot be increased by introducing new hidden units. If the classifier is made more complex, this can only result in closer approximation to the Bayes classification rule. This remark also applies, to some extent, to nearest neighbor classifiers, since increasing the number of prototypes results in closer approximation to the 1-NN classifier, which is known to have near-optimal asymptotic performance.

In practice however, one is always in a situation where only a finite number of training samples is available. In such a case, numerical simulations reveal the existence of the so-call *peaking phenomenon* [35]. As the complexity of the classifier increases, classification error initially drops, then attains a minimum, and then begins to increase. Intuitively, this is due to the fact that inexact estimation of additional parameters increases classification error. At some point, this effect becomes larger than the gain resulting from greater flexibility of the classifier. For that reason, the design of patterns classifiers with optimal complexity is of the utmost importance in practical applications, and the development of heuristic methods for automatic determination of a near-optimal

number of hidden neurons has been the subject of very intensive research. A variety of techniques have been proposed, which can be categorized as relying on *destructive*, *constructive* or *direct* strategies. In the *destructive* approach, the complexity of the network is gradually reduced either by penalizing complexity through addition of a bias term to the error function, or by pruning the least relevant units in the course of the training process [37]. In the *constructive* strategy, a small initial network is gradually expanded until the task is considered to be solved. Examples of such techniques are described in [12, 16, 26, 27, 31, 25]. The *direct* approach consists in using prior information, acquired through preprocessing or readily available from domain knowledge, to design a neural net that can then be further trained using a standard learning procedure such as back-propagation [42, 8, 19].

In all cases, the classification error of the classifier has to be either estimated, or derived from theoretical considerations. Raudys and Jain [36] discuss several methods or error estimation including the re-substitution, hold-out, cross-validation and bootstrap methods. The hold-out method consists in dividing the available data into a training set and a test set used for independent error estimation. This method provides an unbiased error estimate, but it has the disadvantage of preventing the use of all the data for the learning process. The cross-validation and bootstrap methods are more efficient, but also more computationally demanding.

As an alternative, an idea of the generalization performance of a classifier can sometimes be gained as a result of some kind of theoretical analysis. Recent investigations based on the Vapnik-Chervonenkis theory [46] and the PAC learning model [45] have led to the derivation of bounds for the true and estimated classification errors of minimum empirical error classifiers [2, 1, 23]. However, these results are based on a worst-case analysis and lead to very pessimistic estimates of the number of samples needed to train a classifier [34]. Nevertheless, some of the most recent results are already applicable with approximations as an initial aid to neural network design [17]. Further improvements are expected from the consideration of specific input distributions and training algorithms in this analysis [23].

4 Alternative approaches

Since the 1950's, substantial progress has been achieved in the design of statistical classifiers from empirical data. According to [36], the number of classification methods already published exceeds two hundred. These methods are described in a number of standard text books such as [10], [13] and [29].

A useful taxonomy of classification techniques, including statistical and neural network approaches, has been proposed by Lippmann [28]. Pattern classifiers can be seen as belonging to three main categories. *Probability Density Function* classifiers estimate class-conditional probability densities separately for each class. They include parametric normal classifiers with different forms of covariance matrices, and non parametric methods of density estimation such as the Parzen-window approach. *Posterior probability classifiers* estimate the

posterior probability of each class, using simultaneously all the available data. Examples of such methods are MLPs and RBF networks, and the voting k-NN rule. The third category of classification method includes techniques for directly partitioning the feature space into decision regions, using binary indicator outputs. Examples of such *boundary forming* methods are the nearest-neighbor methods such as RCE or LVQ, and tree-structured classifiers [4].

A further distinction can be drawn between *model-based* and *data-driven* approaches. In the model-based approach, a particular classifier is chosen among a pre-defined family of functions, or *model*. Parametric classifiers, MLPs and LVQ classifiers fall in this category. On the contrary, the form of data-driven classifiers is not fixed in advance, but determined by the data. This is the case for Parzen-Window, k-NN and tree-structures classifiers, as well as for ontogenic neural networks that adapt their structure during the learning process.

This multiplicity of classification techniques obviously poses a serious problem to the practitioner. Many comparative studies have been made to assess the strengths and weaknesses of various methods (e.g. [44, 30, 5, 3]). In general, comparable error rates are achieved by several techniques, provided they are properly tuned. As noted by Ng and Lippmann [30], the selection of a classifier for a particular task should primarily be guided by practical considerations such as training and classification time, and memory storage requirements. Neural network classifiers usually offer a good compromise between performance and practical applicability.

References

[1] M. Anthony. Probabilistic analysis of learning in artificial neural networks: The PAC model and its variants. Technical Report NC-TR-94-3, Royal Holloway, University of London, Egham, Surrey TW20 0EX, England, 1994.

[2] E. B. Baum and D. Haussler. What size net gives valid generalization. *Neural Computing*, 1(1):151–160, 1989.

[3] J. L. Blue, G. T. Candela, P. J. Grother, R. Chellappa, and C. L. Wilson. Evaluation of pattern classifiers for fingerprint and OCR applications. *Pattern Recognition*, 27(4):485–501, 1994.

[4] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, CA, 1984.

[5] D. E. Brown, V. Corruble, and C. L. Pittard. A comparison of decision tree classifiers with backpropagation neural networks for multimodal classification problems. *Pattern Recognition*, 26(6):953–961, 1993.

[6] C. K. Chow. On optimum recognition error and reject tradeoff. *IEEE Trans. Inform. Theory*, IT-16:41–46, 1970.

[7] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Trans. Inform. Theory*, IT-13(1):21–27, 1967.

[8] T. Denœux and R. Lengellé. Initializing back-propagation networks with prototypes. *Neural Networks*, 6:351–363, 1993.

[9] B. Dubuisson and M. Masson. A statistical decision rule with incomplete knowledge about classes. *Pattern Recognition*, 26(1):155–165, 1993.

[10] R. O. Duda and P. E Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, New-York, 1973.

[11] R. P.W. Duin. Superlearning and neural network magic. *Pattern Recognition Letters*, 15:215–217, 1994.

[12] S.E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 524–532. Morgan Kaufmann, San Mateo, CA, 1990.

[13] K. Fukunaga. *Introduction to statistical pattern recognition (second edition).* Academic Press, 1990.

[14] F. Girosi. Regularization theory, radial basis functions and networks. In J. H. Friedman V. Cherkassky and H. Wechsler, editors, *From Statistics to Neural Networks*, pages 166–187. Springer-Verlag, Berlin, 1994.

[15] J. B. Hampshire and B. Pearlmutter. Equivalence proof for multilayer perceptron networks and the bayesian discriminant function. In D. S. Touretzky, J. L. Elman, T. J. Sejnowski, and G. E. Hinton, editors, *Coonectionist models. Proceedings of the 1990 summer school.*, pages 159–172. Morgan Kaufmann, San Mateo, Ca, 1991.

[16] Y. Hirose, K. Yamashita, and S. Hijiya. Back-propagation algorithm which varies the number of hidden units. *Neural Networks*, 4(1):61–66, 1991.

[17] S. B. Holden and M. Niranjan. On the practical applicability of VC dimension bounds. Technical Report CUED/F-INFENG/TR.155, Cambridge University Engineering Department, Cambridge CB2 1 PZ, 1994.

[18] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4:251–257, 1991.

[19] M. Karouia, R. Lengellé, and T. Denœux. Performance analysis of a MLP weight initialization algorithm. In *Proc. of ESANN'95, European Symposium on Artificial Neural Networks*, pages 347–352, Brussels, April 1995. D facto publications.

[20] T. Kohonen. *Self organisation and associative memory (second edition).* Springer Verlag, Berlin, 1987.

[21] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[22] T. Kohonen, G. Barna, and R. Chrisley. Statistical pattern recognition with neural networks: Benchmarking studies. In *Proccedings of ICNN'88, Int. Conf. on Neural Networks, vol. I*, pages 61–68. IEEE Computer Society Press, 1988.

[23] M. A. Kraaijveld. *Small sample behavior of multi-layer feedforward network classifiers: Theortetical and practical aspects.* PhD thesis, Delft University, Delft, The Netherland, 1993.

[24] D.-S. Lee, S. N. Srihari, and R. Gaborski. Bayesian and neural network pattern recognition: a theoretical connection and empirical results with handwritten characters. In I. K. Sethi and A. K. Jain, editors, *Artificial Neural networks and Statistical Pattern Recognition*, pages 89–108. Elsevier Science B. V., Amsterdam, 1991.

[25] S. Lee. Supervised learning with gaussian potentials. In B. Kosko, editor, *Neural networks for signal processing*, pages 189–227. Prentice-Hall, Englewood Cliffs, N-J, 1992.

[26] R. Lengellé and T. Denœux. Optimizing multilayer networks layer per layer without back-propagation. In I. Aleksander and J. Taylor, editors, *Artificial Neural Networks II*, pages 995–998. North-Holland, Amsterdam, 1992.

[27] R. Lengellé and T. Denœux. Training multilayer perceptrons layer by layer using an objective function for internal representations. *Neural Networks (to appear)*, 1995.

[28] R. P. Lippmann. Neural networks, Bayesian a posteriori probabilities, and pattern classification. In J. H. Friedman V. Cherkassky and H. Wechsler, editors, *From Statistics to Neural Networks*, pages 83–104. Springer-Verlag, Berlin, 1994.

[29] G. J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition.* John Wiley and Sons, New-York, 1992.

[30] K. Ng and R. P. Lippmann. A comparative study of the practical characteristics of neural networks and conventional pattern classifiers. In R. L. Lippman, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 970–976. Morgan Kaufmann, San Mateo, CA, 1991.

[31] J. C. Platt. Learning by combining memorization and gradient descent. In R. P. Lippman, J. E. Moody, and D. S. Touretzky, editors, *Neural Information Processing 3*, pages 714–720. Morgan Kaufmann, San Mateo, CA, 1991.

[32] T. Poggio and F. Girosi. A theory of networks for approximation and learning. Technical Report A.I. Memo No. 1140, M.I.T, 1988.

[33] F. Poirier and A. Ferrieux. DVQ: Dynamic vector quantization - an incremental LVQ. In T. Kohonen, M. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks, Vol. 2*, pages II–1333–1336. Elsevier Science B. V., Amsterdam, 1991.

[34] S. J. Raudys. Why do multilayer perceptrons have favorable small sample properties. In E. S. Gelsema and L. N. Kanal, editors, *Pattern Recognition in Practice IV*, pages 287–298. Elsevier Science B. V., Amsterdam, 1994.

[35] S. J. Raudys and A. K. Jain. Small sample problems in designing artificial neural networks. In I. K. Sethi and A. K. Jain, editors, *Artificial Neural networks and Statistical Pattern Recognition*, pages 33–50. Elsevier Science B. V., Amsterdam, 1991.

[36] S. J. Raudys and A. K. Jain. Small sample size effects in statistical pattern recognition: Recommendations for practitioners. *IEEE Transactions on Pattern Analysis and Machine INtelligence*, 13(3):252–264, 1991.

[37] R. Reed. Pruning algorithms: a survey. *IEEE Transactions on Neural Networks*, 4(5):740–747, 1993.

[38] D. L. Reilly, L. N. Cooper, and C. Elbaum. A neural model of category learning. *Biological Cybernetics*, 45:35–41, 1982.

[39] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[40] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. McClelland, editors, *Parallel Distributed Processing*. MIT Press, Cambridge, MA, 1986.

[41] W. F. Schmidt. *Neural Pattern Classifying Systems*. PhD thesis, Delft University, Delft, The Netherland, 1993.

[42] I. K. Sethi. Entropy nets: From decision trees to neural networks. *Proceeding of the IEEE*, 78(10):1605–1613, 1990.

[43] D. S. Thomas and A. Mitiche. Asymptotic optimality of pattern recognition by regression analysis. *Neural Networks*, 7(2):313–320, 1994.

[44] A. C. Tsoi and R. A. Pearson. Comparison of three classification techniques, CART, C4.5 and multi-layer perceptrons. In R. L. Lippman, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 963–969. Morgan Kaufmann, San Mateo, CA, 1991.

[45] L. G. Valiant. A theory of the learnable. *Coomunications of the ACM*, 27(11):1134–1142, 1984.

[46] V. N. Vapnik. *Estimation of dependences based on empirical data*. Springer series in statistics. Springer-Verlag, 1982.

[47] P. J. Werbos. Links between artificial neural networks and statistical pattern recognition. In I. K. Sethi and A. K. Jain, editors, *Artificial Neural networks and Statistical Pattern Recognition*, pages 11–31. Elsevier Science B. V., Amsterdam, 1991.

[48] H. White. Learning in artificial neural networks: a statistical perspective. *Neural computation*, 1:425–464, 1989.

[49] B. Widrow and M. A. Lehr. 30 years of adaptive neural networks: Perceptrons,madaline and backpropagation. *Proceedings of the IEEE*, 78(9):1415–1442, 1990.