1998 Fifth IEEE International Workshop on Cellular Neural Networks and their Applications, London, England, 14-17 April 1998

Classification Systems Based on Neural Networks

Josef A. Nossek, Robert Eigenmann, Georgios Papoutsis, Wolfgang Utschick

Institute for Network Theory and Circuit Design
Munich Institute of Technology
Arcisstr. 21, 80290 Munich, Germany

ABSTRACT: *Classification is a problem that apears in many real life applications. In this paper we describe the general case of multi-class classification, where the task of the classification system is to map an input vector $\boldsymbol{x}$ to one of $K > 2$ given classes. This problem is split in many two-class classification problems, each of them describing a part of the whole problem. These are solved by neural networks, producing an intermediate output in a reference space, which is then decoded to the solution of the original problem. The methods described here are then applied to the handwritten character recognition problem to produce the results described later in the article. It is suspected that they also may be applied successfully in the context of the CNN paradigm and be implemented on a CNN-Universal Machine.*

1 Introduction

The task of classification occurs in a wide range of applications. Typical examples are the discrimination of handwritten symbols for reading Zip Codes on envelopes, reading amounts in check reading systems, etc. Intelligent Character Recognition which is generally a single module in a complex document analysis system [1]. After scannning the document a variety of processes must be performed before classification occurs (see [2]).

Of course a classification system could be used to classify raw input data from the scanned documents. In real applications, such methods will generally produce poor results. It is necessary to transform the input data into a new representation, and the choice of pre-processing is one of the most important factors for the performance of the complete system (Fig. 1).

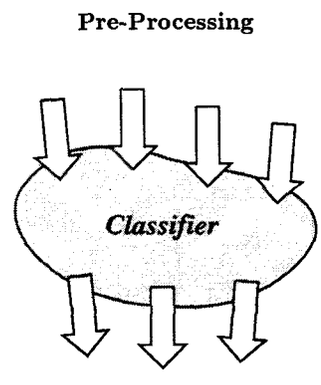

Figure 1: *Modules of an ICR-System: pre-processing, classification, post-processing.*

Pre-processing is generally applied to all training samples before training starts. The extracted features are supplied to the classification system: "Clearly there is a balance to be found in the extent to which data processing is performed in the pre-processing and post-processing stages, and the extent to which it is performed by the network itself." (C.M. Bishop in [3]).

Many of the preprocessing steps (edge and corner detection, detection of connected components, etc.) have also been implemented on the CNN-architecture [4].

0-7803-4867-2/98/$10.00©1998 IEEE

2 The classification system

Designing a classification system is considered to be the construction of a decision rule (1) that will be applied to a sequence of patterns $\boldsymbol{x}$, and that uniquely assigns each pattern to one of a finite set of pre-defined classes $\{C_1, C_2, \ldots, C_K\}$ on the basis of N observed attributes $x_1, \ldots, x_N$ from the original pattern $\boldsymbol{x}^T = (x_1, \ldots, x_N) \in \mathcal{R}^N$.

$$d : \mathcal{R}^N \rightarrow \mathcal{I}_K = \{1, 2, \ldots, K\},\ \boldsymbol{x} \mapsto k. \tag{1}$$

The true class behind any input vector is given by the unknown target function $t : \mathcal{R}^N \rightarrow \mathcal{I}_K,\ \boldsymbol{x} \mapsto k$. The construction of a decision rule from a training set $\mathcal{S} := \{(\boldsymbol{x}_1, k_1), (\boldsymbol{x}_2, k_2)\ , \cdots,\ (\boldsymbol{x}_M, k_M)\}$, taken from the joint probability space, for which the true classes $k_m = t(\boldsymbol{x}_m)$ are known, has been called supervised learning (see [3, 5, 6]).

The optimal decision rule that minimizes the risk of misclassification errors can be obtained by $k = \arg\max_l P[C_l \mid \boldsymbol{x}]$ which is called Bayes classification rule [7, 8, 9, 10].

$$P[C_l \mid \boldsymbol{x}] = \frac{P[\boldsymbol{x} \mid C_l] P[C_l]}{\sum_{i=1}^{K} (P[\boldsymbol{x} \mid C_i] P[C_i])}, \tag{2}$$

$$\sum_{i=1}^{K} P[C_l \mid \boldsymbol{x}] = 1.$$

Unfortunately, the Bayes classifier is not applicable in most practical applications because distributions of classes are generally unknown. The estimation of density functions is not feasible because the task of density estimation is an ill-posed problem and even harder than the original problem of classification. On the contrary, non-parametric approaches (e.g. neural networks) do not need any assumptions about the distributions of the training set. The class of possible decision functions depends on the choice of the classifier.

In this work, the decision function (Fig. 2) is composed of two steps from the domain of input vectors $\mathcal{R}^N$ into decision space $\mathcal{D}$, and from there to the set of possible class indices $\{1, \ldots, K\}$: $\boldsymbol{d} \circ D : \mathcal{R}^N \rightarrow \mathcal{D} \rightarrow \mathcal{I}_K,\ \boldsymbol{x} \mapsto \boldsymbol{d} \mapsto k$. The first layer is composed of J parallel neural components, mapping the input $\boldsymbol{x}$ to a vector $\boldsymbol{d}$ in the decision space $\mathcal{D} = \mathcal{R}^J$. Each class C_k is represented by a reference vector $\boldsymbol{t}_k$ in the decision space. In the second step, the decoder assignes the vector $\boldsymbol{d}$ to the reference vector $\boldsymbol{t}_k$ with the minimal distance to $\boldsymbol{d}$ in a given metric on $\mathcal{D}$.

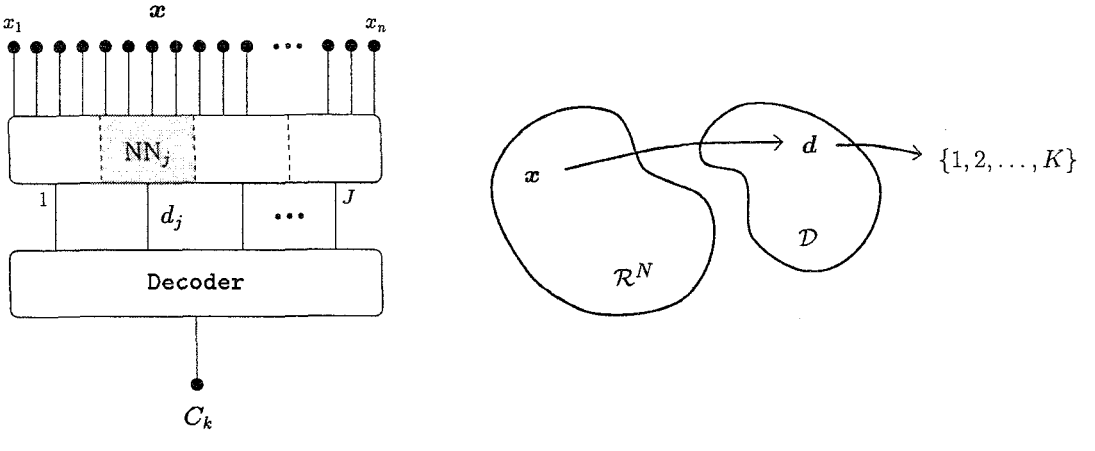

Figure 2: *Mapping of the feature vector by the classification system of J parallel plug-in components. The decoder assigns the output vector $\boldsymbol{d}$ to one of the possible classes C_k by means of minimal distances to reference vectors.*

The set of reference vectors is defined in the so-called coding matrix $\boldsymbol{T}$. The k-th row of this matrix represents the reference vector $\boldsymbol{t}_k$. The following examples show the coding matrix for $K = 10$ classes, for a coding with $J = 7$, the trivial 1-out-of-K coding with $J = 10$ and the trivial pairwise coding with

$J = 45$:

$$\boldsymbol{T} = \begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ +1 & +1 & +1 & +1 & -1 & -1 & -1 \\ +1 & -1 & +1 & -1 & -1 & +1 & -1 \\ -1 & +1 & -1 & +1 & -1 & +1 & -1 \\ -1 & -1 & -1 & -1 & +1 & +1 & +1 \\ -1 & +1 & -1 & +1 & +1 & -1 & +1 \\ -1 & +1 & +1 & -1 & -1 & +1 & +1 \\ -1 & -1 & +1 & +1 & -1 & -1 & +1 \\ -1 & +1 & +1 & -1 & +1 & -1 & -1 \\ -1 & -1 & +1 & +1 & +1 & +1 & -1 \end{pmatrix}, \quad \boldsymbol{T}_{1of10} = \begin{pmatrix} +1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & +1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & +1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & +1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & +1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & +1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & +1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & +1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & +1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & +1 \end{pmatrix},$$

$$\boldsymbol{T}_{pairwise} = \begin{pmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 & 0 & 0 & 0 & & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & +1 & +1 & & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & & +1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & & 0 & +1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & & -1 & -1 \end{pmatrix}.$$

In this representation, the j-th column of $\boldsymbol{T}$ represents the classification task of the j-th neural component in the first layer, by splitting the set $\mathcal{C} = \{C_1, \ldots, C_K\}$ in the two subsets:

$$\mathcal{C}_j^+ = \{C_k \in \mathcal{C} : t_{kj} = +1\}, \qquad \mathcal{C}_j^- = \{C_k \in \mathcal{C} : t_{kj} = -1\}. \tag{3}$$

which are to be discriminated by that component.

Trivial codes like the 1-out-of-K code or the pairwise code (see [11]) have the advantage of using simple dichotomies for the classification task of the first layer, thus achieving good single output results. This is though compensated by the need of more components in the first layer and the lack of an error correction capability. A non trivial code on the other hand may give worse classification results on a single output level because of the not so well arranged dichotomies in the pattern space, but an increased Hamming distance between the reference vectors makes error detection, or even error correction feasible, which can compensate for the single output errors.

The main differences of the presented classification system to the standard approaches of Error Correcting Output Coding [12, 13, 14] are the substantially smaller number of parallel plug-in classifiers $J \leq K$, the introduction of an error function Φ, and above all, the combination of the optimization of the coding matrix $\boldsymbol{T}$ and the training process of the parallel classifiers in a mutual training process [15, 16].

The design of non-trivial error-correcting codes is based on the maximization of a cost function Φ, where the cost function is given by a trade-off between the empirical risk on training samples and a regularization term in the decision space:

$$\Phi(\boldsymbol{t}_1, \ldots, \boldsymbol{t}_K) := \underbrace{\sum_{k=1}^{K} P[\boldsymbol{d} = \boldsymbol{t}_k \mid C_k] \cdot P[C_k]}_{\Phi_1} + \lambda \cdot \underbrace{\log \prod_{k=1}^{K-1} \prod_{l=k+1}^{K} \|\boldsymbol{t}_k - \boldsymbol{t}_l\|_1}_{\Phi_2}. \tag{4}$$

The regularization is given by the logarithm of the product of distances between the reference vectors of all classes. The error term Φ_1 obviously corresponds to the empirical risk P_{err}^{trn}, whereas Φ_2 represents the regularizing term for the design of reference vectors. The impact of the regularizer Φ_2 complicates the generation of trivial two-class problems (dichotomies) of each parallel classifier[1]. Hence, the regularization approach increases the requirements on the complexity of the models. This may be interpreted as a type of Structural Risk Minimization (SRM) by configuration of the training data and not by model selection.

For the solution of the discrete optimization problem $\max \Phi$, the training of the parallel components is split into a sequence of S training epochs. After each training epoch of the parallel neural networks, the optimization of reference vectors $\boldsymbol{t}_k \in \mathcal{T}$ for optimal error-correcting coding is continued. The training of the complete classifier periodically alternates two steps: training of parallel subsystems and optimizing reference vectors. There is no analytical solution to the discrete optimization problem of (4). Therefore, the search for optimal reference vectors is performed iteratively. After each iteration step (Step 1) s, the next potential reference vector of class k is calculated by a so-called pseudo-gradient (see [16]).

[1] The simplest dichotomy would consist of only one class in $\mathcal{C}^+$ and all other classes in the complementary set $\mathcal{C}^-$.

3 Neural Components

This section concentrates on the realization of a 2-class classifier with neural networks applied to each dichotomy $j = 1, \ldots, J$ of classes in the K-class problem. The index j is not used in the subsequel.

Generally, Bayes theorem (2) defines the optimal decision rule, deciding for the class with the maximum posterior probability

$$d = \begin{cases} +1 & \text{for} \quad P[\mathcal{C}^+ \mid \boldsymbol{x}] > P[\mathcal{C}^- \mid \boldsymbol{x}] \\ -1 & \text{else} \end{cases} .$$

Intelligent classification methods perform an estimation of the posterior probability in a region around the optimal decision boundary $\mathcal{B}(\boldsymbol{x}) : P[\mathcal{C}^+ \mid \boldsymbol{x}] = P[\mathcal{C}^- \mid \boldsymbol{x}] = 0.5$, avoiding a density estimation in the high dimensional feature space.

A well studied approach is the classification using feedforward neural networks. The general architecture consists of several layers $l = 1, \ldots, L$ of $h^{<l>}$ neurons, whereas each neuron performs the nonlinear mapping

$$x_i^{<l>} = \Psi(\boldsymbol{w}_i^{\mathrm{T}<l>} \boldsymbol{x}^{<l-1>}).$$

The activation function Ψ is monotonically increasing and bounded, usually defined as $\Psi(a) = \tanh(\beta a) \in (-1, +1)$.

The weights are obtained in a gradient descent optimization process, minimizing the sum of the squared divergence E of the targets t^j and the performed outputs $d(\boldsymbol{x}^j)$ [17]

$$E = \sum_{j=1}^{M} (d(\boldsymbol{x}_j) - t_j)^2.$$

Although this approach is known to yield good results in a large field of applications, there is no theory about the optimal number of layers and neurons within each layer. A high degree of freedom results in poor generalization, signifying memorization of training patterns instead of extracting a decision rule. Another disadvantage is the slow convergence towards a local optimum and high computational effort to evaluate the gradient in an sequential process.

Using a hard-limiting activation function $\Psi(a) = \mathrm{sign}(a) \in \{-1, +1\}$ simplifies the general feedforward architecture significantly, thus reducing the problem to find an appropriate complex architecture to the determination of only one neural layer. Additionally, training does not employ gradient information, reducing the computational effort.

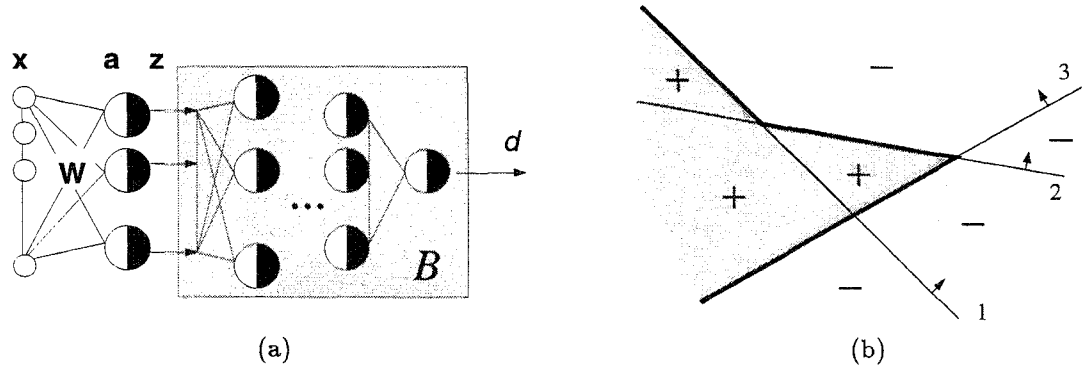

Figure 3: *2-class classifier. Architecture (a), piecewise-linear decision boundary in feature space (b).*

The first layer consisting of $h = h^{<1>}$ neurons performs a mapping

$$W : \mathcal{R}^N \to \{-1, +1\}^h, \ \boldsymbol{x} \mapsto \boldsymbol{z} ,$$

and the subsequel layers $l = 2, \ldots, L$ can be interpreted as a Boolean mapping

$$B : \{-1, +1\}^h \to \{-1, +1\}, \ \boldsymbol{z} \mapsto d ,$$

which requires the storage of 2^h binary values $b_{\boldsymbol{c}}$ for each possible internal representation $\boldsymbol{c} \in \{-1,+1\}^h$, see Fig. 3(a). Geometrically interpreted, each neuron $i = 1,\ldots,h$ defines a hyperplane $\boldsymbol{w}_i^T\boldsymbol{x} = 0$ in the feature space and h neurons manage a partitioning of the feature space into 2^h convex regions, called cells. As the weights $\boldsymbol{w}_i$ define the location of the hyperplanes, the binary values $b_{\boldsymbol{c}}$ correspond to the output d, given an input $\boldsymbol{x}$ located in cell $\boldsymbol{c}$, see Fig. 3(b).

There exists a large variety of training algorithms for the proposed type of neural network, working with a fixed Boolean function [18, 19] or including the adaptation of the values $b_{\boldsymbol{c}}$ in the training process [20]. Very powerful constructive training algorithms have been published, fitting the network complexity to the classification problem my means of varying the number of neurons [21, 22] during training.

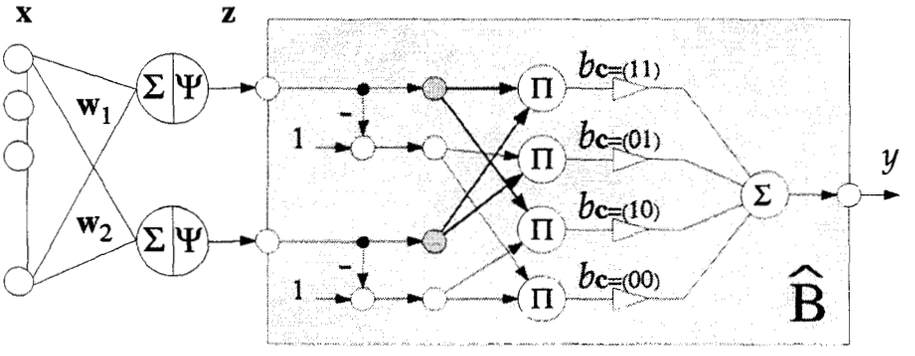

Figure 4: $\Sigma\Pi\Sigma$ *network: Modified architecture for* $h = 2$ *neurons*

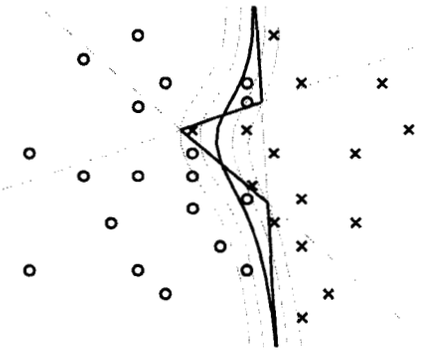

Figure 5: *Effect of the* $\Sigma\Pi\Sigma$ *architecture on the original piecewise-linear decision boundary*

As the output d of the hard-limiting neural network is binary, no information about the reliability of the decision is supported. Although classification is performed with high accuracy, professional systems require the option to reject uncertain decisions in order to decrease errors. Certainty of the decision is related to the distance of a pattern to the decision boundary [20] or can be evaluated with the expectation value of the output d [23]. Latter leads to a modified architecture (Fig. 4), considering the statistical distribution of the neural activation in a modified norm of the weight vectors. The neurons of the hidden layer are changed to have a sigmoidal activation function

$$\Psi(a_i) = \frac{1}{1+\mathrm{e}^{-a_i}} ,$$

and the Boolean function is replaced by a soft Boolean function $\hat{B}(\boldsymbol{z})$, managing real valued inputs and providing an output

$$y = \hat{B}(\boldsymbol{z}) = \sum_{\boldsymbol{c} \in \{0;1\}^h} \left(b_{\boldsymbol{c}} \cdot \prod_{c_i = 1} z_i \cdot \prod_{c_i = 0} (1 - z_i) \right) . \tag{5}$$

The $\Sigma\Pi\Sigma$ network has shown to increase generalization [23] due to the regularization property on the former piecewise linear decision boundary (Fig. 5).

4 Pre-processing and feature extraction in OCR

In general, the complexity of any classification task increases with the dimension of the feature space. Motivated by this property, a pre-processing $\mathcal{P}$ of the data in order to decrease the dimensionality of the input vectors is presumed in intelligent classifiers.

$$\mathcal{P} : \mathcal{R}^N \rightarrow \mathcal{R}^{\widetilde{N}},\ \boldsymbol{x} \mapsto \widetilde{\boldsymbol{x}}\ .$$

An universal pre-processing strategy is given by the Karhunen-Loéve transformation or principal component analysis (pca) [24], projecting the input vectors onto the eigenvectors corresponding to the $\widetilde{N}$ largest eigenvalues.

Given a real-life classification application, applying prior knowledge about the problem can significantly simplify the task. Considering an Optical Character Recognition (OCR) problem, the raw data is presented in a $m \times n$ pixel bitmap image. An appropriate pre-processing should normalize the image regarding invariance to the stroke width, image size, translation and tilt of the character.

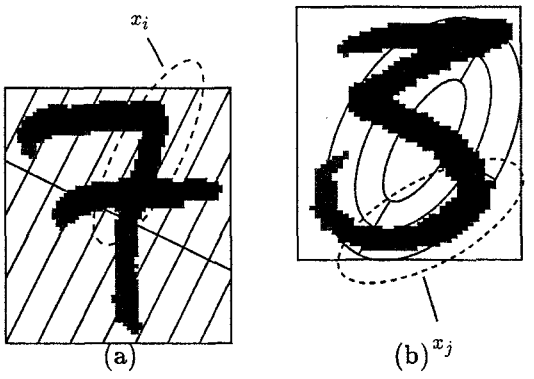

Figure 6: *(a) Line primitives, (b) ellipsoid primitives*

Higher accuracy can be achieved, if some discriminating features can be defined and extracted from the input data. In OCR problems, the decomposition of images into several primitives of line and ellipsoid segments has turned out to support effective features. Here, each element of the feature vector $\boldsymbol{x}$ is related to the matching of one specific primitive to the image, see Fig. 6.

5 Experimental Results

We considered several different sets of target vectors to represent the set of classes $\mathcal{C} = \{0, 1, \ldots 9\}$ of handwritten numerals, i.e. the standard 1-out-of-10 code $\boldsymbol{T}_{1of10}$, the pairwise code $\boldsymbol{T}_{pairwise}$ and a 10-bit non-trivial code $\boldsymbol{T}_{opt}$, generated during the training process (4). Corresponding to Section 4, a vector of 196 features was extracted from the bitmap images.

As mentioned in section 2, the simple dichotomies of trivial codes cause good classification results considering the J two-class problems, whereas non-trivial codes enable an error-correction in the decoding level. Table 1 shows the number of neurons h used in each two-class classifier NN_j, $j = 1, \ldots, J$, the mean bit error $\bar{\eta} = \mathrm{E}[\eta_j]$, the error of the overall classification system ε at 0% rejection and the required training time t on a Sparc 10. The classifiers were trained using 240.000 instances of the MNIST database [25] and a disjoint database of 60.000 patterns was applied for the evaluation of the error rates.

Due to the very simple dichotomies, the pairwise code performed the lowest bit errors and trainig required only little time. In contrast to the pairwise coding scheme, the non-trivial code design had high computaional costs and high bit errors. Due to the increased hamming distance of the non-trivial code, the overall error on the test set can be reduced by means of error correction.

Rejection can be controlled by thresholding the confidence of the first choice, decreasing misclassification, see Fig. 7.

Code	J	h	$\bar{\eta}$ [%]	ε [%]	t [hours]
$\boldsymbol{T}_{pairwise}$	45	1	0.2441	1.45	≈ 2.3
$\boldsymbol{T}_{1of10}$	10	5	0.4351	1.42	≈ 5.4
$\boldsymbol{T}_{opt}$	10	5	1.31	1.56	≈ 39

Table 1: *Classification results for different coding schemes*

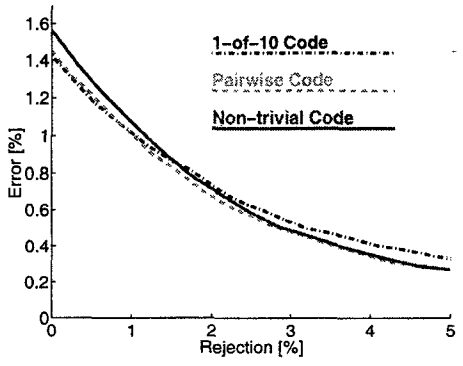

Figure 7: *Error and Rejection for different codes.*

6 Conclusions

In this article we have presented a method for solving classification problems by dividing the classification process in two stages. In the first stage we use many neural networks for solving different parts of the whole problem. These produce an output in a reference space, where each class is also represented by a reference vector. The nearest reference vector to the output describes thus the solution of the whole problem.

Different methods for dividing the problem into simpler ones have been presented. Some of them need few neural networks in the first stage, but divide the set of classes in complex dichotomies, which makes them more difficult to learn. Others need far more networks, each of which having very simple tasks to accomplish, which reduces considerably their complexity.

By applying these methods in real world problems, special attention has to be paid to the preprocessing of the data, as using knowledge of the structure of the data for the feature extraction may make the classification task much simpler. Examples of these methods for the character recognition problem have produced very good results.

Further work is currently being done in combining different codings for increasing further the classification rate of the whole system.

References

[1] H. Bunke and P.S.P. Wang, editors. *Handbook on Optical Character Recognition and Document Analysis.* World Scientific Publishing Company, 1996.

[2] J.Schürmann, N. Bartneck, T. Bayer, J. Franke, E. Mandler, and M. Oberländer. Document analysis - from pixels to contents. *Proceedings of the IEEE*, 80(7), 1992.

[3] C. M. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press, 1995.

[4] L. O. Chua and T. Roska. The CNN Paradigm. *IEEE Transactions on Circuits and Systems - I*, 40(3):147–156, 1993.

[5] B.D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, 1996.

[6] J. Schürmann. *Pattern Classification, A Unified View Of Statistical And Neural Approaches.* Wiley, 1996.

[7] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, 1972.

[8] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* Wiley, 1973.

[9] T.Y. Young and T.W. Calvert. *Classification, Estimation And Pattern Recognition.* American Elsevier Publishing Co., Inc., 1974.

[10] P.A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach.* Prentice Hall, Englewood Cliffs, 1982.

[11] G. Papoutsis and J.A. Nossek. Combination Methods fo Pairwise Classifiers for Multi-Class-Classification. In *Proc. of the European Conference on Circuit Theory and Design, ECCTD, Budapest*, pages 1114–1117, 1997.

[12] T.G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[13] E.B. Kong and T.G. Dietterich. Error-Correcting Output Coding Corrects Bias and Variance. In *Proceedings of the 12th International Conference on Machine Learning*, pages 313–321. Morgan Kaufmann, 1995.

[14] J. Gareth and T. Hastie. Error coding and PaCT's, 1997. Winning paper in the ASA student paper competition for the Statistical Computing Section, available at http://playfair.Stanford.EDU/~trevor/ASA/winners.html.

[15] W. Utschick, H.-P. Veit, and J.A. Nossek. Non-trivial codes for polychotomous classification in pattern recognition. In *Proceedings of the International Conference on Engineering Applications of Neural Networks*, pages 25–28, 1997.

[16] W. Utschick. A regularization method for non-trivial codes in polychotomous classification. *International Journal of Pattern Recognition and Artificial Intelligence*, 1998. to appear.

[17] D.E. Rumelhart and J.L. McClelland. Learning internal representations by error propagation,. In *Parallel Distributed Processing*, volume 1, chapter 8. The MIT Press, 1986.

[18] J.A. Nossek, P. Nachbar, and A.J. Schuler. Comparison of algorithms for feedforward multilayer neural nets. In *Proceedings of the International Conference on Circuits and Systems*, volume 3. IEEE Computer Society Press, 1996.

[19] B. Widrow, R.G. Winter, and R.A. Baxter. Layered neural nets for pattern recognition. *Transactions on Acoustics, Speech, and Signal Processing*, 36(7), 1988.

[20] W. Utschick and J.A. Nossek. Hybrid optimization of feedforward neural networks for handwritten character recognition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 147–150. IEEE Computer Society Press, 1997.

[21] Marc Mézard and Jean-Pierre Nadal. Learning in feedforward layered networks: the tiling algorithm. *J. Phys. A: Math. Gen.*, 22:2191–2203, 1989.

[22] R. Eigenmann and J.A. Nossek. Constructive and Robust Combinations of Perceptrons. In *Proc. of the 13th International Conference on Pattern Recognition, Vienna*, volume IV, pages 195–198. IAPR, 1996.

[23] R. Eigenmann and J.A. Nossek. Modification of Hard-Limiting Multilayer Neural Networks for Confidence Evaluation. In *Proc. of the 4th International Conference on Document Analysis and Recognition, ICDAR, Ulm, Germany*, pages 1087–1091, 1997.

[24] I.T. Jollife. *Principal Component Analysis.* Springer, 1986.

[25] L. Bottou, C. Cortes, J.S. Denker, H. Drucker, I. Guyon, L.D. Jackel, Y. LeCun, E. Sackinger, P. Simard, V.N. Vapnik, and U.A. Miller. Comparison of classifier methods: A case study in handwritten digit recognition. In *Proceedings of the 12th International Conference on Pattern Recognition and Neural Network*, pages 77–82, 1994.