Error-Correcting Classification via ML-Techniques and Regularization

Wolfgang Utschick and Josef A. Nossek

Technische Universität München
Lehrstuhl für Netzwerktheorie und Schaltungstechnik
Arcisstr. 21, D-80290 München
wout@nws.e-technik.tu-muenchen.de

Abstract-Instead of *a priori* approaches to Error-Correcting Output Coding (ECOC) an adaptive solution to the definition of codewords in multiclass learning problems is presented. The generation of codewords and the training of parallel subsystems of the overall classification system is matter of an iterative optimization method which takes advantage of a Maximum Likelihood (ML) technique in combination with a regularization principle for penalizing low distances between codewords. Even for a very restricted complexity of outputs the optimized codes outperform the trivial one-per-class output coding. In a practical application, the generalization capability of the infered decision rule was considerably improved.

I. Introduction

Polychotomous classification is a general task in pattern recognition. Designing a classifier is considered to be the approximation of an unknown decision rule which will be applied to a sequence of patterns. The input pattern must be uniquely assigned to one element of the unordered set of class indices $\mathcal{K} = \{1, 2, \ldots, K\}$ on the basis of n observed attributes $(x_1, \ldots, x_N) = \boldsymbol{x} \in \mathcal{R}^N$ from the original pattern. The construction of the decision rule is infered from a set of pre-classified training examples $\mathcal{S} = ((\boldsymbol{x}_1, k_1), (\boldsymbol{x}_2, k_2), \cdots, (\boldsymbol{x}_M, k_M)) \subset \mathcal{X} \times \mathcal{K}$, for which the true class indices $k_m \in \mathcal{K}$ are given (see supervised learning in [1, 2, 3, 4]).

The approximation of the decision rule is generally identified with the construction of a mapping from the feature space $\mathcal{R}^N$ into the decision space $\mathcal{D}$ [3]. The decision process of each input vector is composed in two steps by

$$d : \mathcal{R}^n \rightarrow \mathcal{D} \rightarrow \mathcal{K}, \; \boldsymbol{x} \mapsto \boldsymbol{d} \mapsto k. \qquad (1)$$

A highly important feature between the layers of the classifier is the representation of classes in the decision space. Practitioners often prefer the trivial 1-out-of-K coding where each output of the classifier corresponds to one of all classes. This method directly corresponds to decision rules based on posterior probabilities of classes [5]. Provided the representation of classes is based on binary reference vectors $\boldsymbol{t} \in \{-1, +1\}^J$ embedded in a real-valued decision space $\mathcal{D}$, the estimation of reference vectors (codewords) for classes [6] is equal to the optimal decomposition of polychotomies into dichotomies [7] (e.g. class k is associated with one of the two super classes of the jth dichotomy if $t_{kj} = +1$, and v.v.). Reference vectors can be interpreted as corners of a high-dimensional hyper cube.

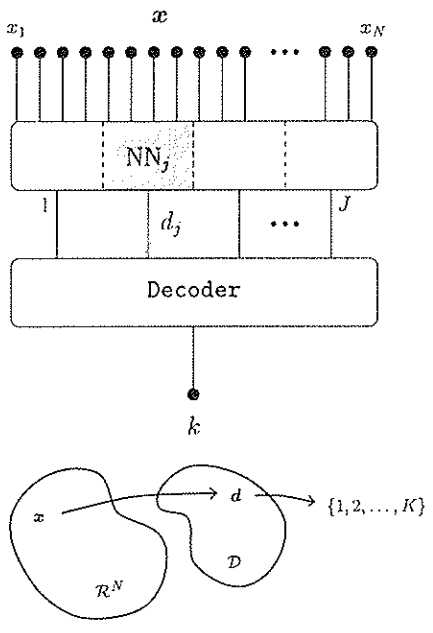

Figure 1: The Classification System.

It has been shown that ECOC proposed by T. Dietterich and G. Bakiri [6, 8] outperforms the one-per-class approach. ECOC is a kind of "neither homogeneous nor non-homogeneous voting" [8], which reduces both the bias error **and** the variance error if the bias errors between the individual two-group classifiers

are uncorrelated. On the other hand, error-correcting coding is supposed to be robust in relation to small sample sizes and in some cases does not depend on the particular assignment of reference vectors to classes [8]. Whereas T. Dietterich's method of ECOC is *a priori*, the decomposition of polychotomies proposed by E. Mayoraz and M. Moreira [7] depends on the underlying training data.

In this work we propose a method that estimates the reference vectors of classes using a technique that combines a maximum likelihood (ML) method for density estimation together with the principle of regularization. The presented classification system (Fig. 1) consists of J parallel classifiers (neural networks) and a following decoder that realizes the function from the decision space onto the set of indices. Each parallel network realizes its own two-class problem with respect to the decomposition of the complete classification problem. The $\boldsymbol{d}(\boldsymbol{x}, \boldsymbol{\theta})$ expresses the mapping of the first layer where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_J)$ summarizes the parameters of the parallel components, and $\boldsymbol{T} = (\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_K)^T$ represents the set of desired codewords for classes in form of a matrix. The decoder performs a minium distance based decision in form of

$$k = \arg\min_{l \in \mathcal{K}} \|\boldsymbol{d} - \boldsymbol{t}_l\|_2. \tag{2}$$

Instead of two separate optimization tasks – searching for a set of reference vectors (decomposition into dichotomies) and training of the parallel components – the training of the network layer is split into a number of training epochs. During each training epoch the parameters of the network components and the reference vectors of classes are iteratively estimated by means of expectation maximization (EM) [9, 10].

In contrast to EM in density estimation problems, the transformed input vectors in form of $\boldsymbol{d}(\boldsymbol{x}, \boldsymbol{\theta})$, or in other words the realizations of the random variable $\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_M$ – the given sample which plays the fundamental part in the ML algorithm – are not constant. Therefore the algorithm rather aims at learning the parameters $\boldsymbol{\theta}$ of all parallel classifiers such that the vectors $\boldsymbol{d}_m(\boldsymbol{\theta}) = \boldsymbol{d}(\boldsymbol{x}_m, \boldsymbol{\theta})$, $\boldsymbol{x}_m \in \mathcal{X}$, optimally correspond to mixtures of disjoint distributions $p(\boldsymbol{d}(\boldsymbol{x}, \boldsymbol{\theta}) \mid \boldsymbol{t})$ according to their respective class memberships. The estimation of this mixture distributions is in turn part of the optimization. It is assumed that for a training epoch the representation of each class is distributed over a small subset of vectors $\boldsymbol{t} \in \mathcal{T}_k$. The subset contains neighboring vectors of reference vector $\boldsymbol{t}_k$ which is defined to be the "gravity center" of all $\boldsymbol{t} \in \mathcal{T}_k$ during this epoch. In order to prevent the distributions of classes in decision space from overlapping, the subsets of vectors $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_K$ must be disjoint. Under the assumption of normal distributions around vectors $\boldsymbol{t} \in \mathcal{T}_k$, the center of distributions are implicity given. The covariance of the normal distribution is defined to be diagonal and only depends on a single variance parameter which is reasonably set to > 1 (cf. distance between binary reference vectors).

The characteristic of the EM algorithm is that the optimization of both objectives, namely the optimum mixture of distributions according to $\boldsymbol{t} \in \mathcal{T}_k$ and the embedding of training samples into desired distributions by training of the first layer, is alternately preformed. The algorithm is only indirectly error-correcting! Fig. 2 presents the idea of the algorithm.

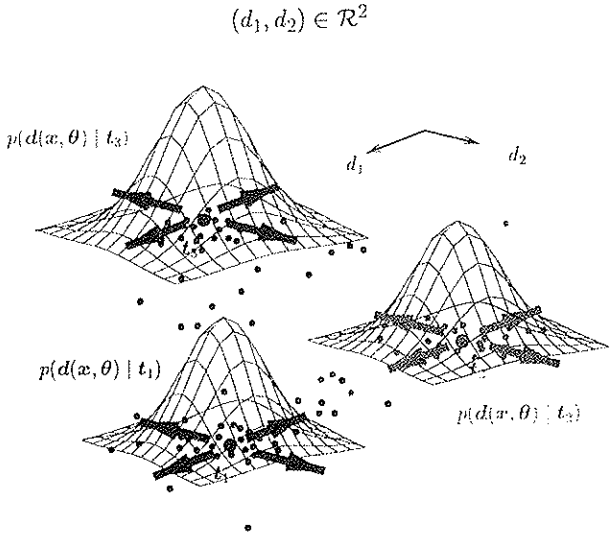

Figure 2: "Gravity Centers" in Decision Space.

At the end of each training epoch, the set of reference vectors $\boldsymbol{T}$ is updated. The update of reference vectors allows to implement the principle of regularization [11]. Using a regularization term, which is based on distances between reference vectors in decision space, already have shown promising results [12].

II. Expectation Maximization

Given the estimated parameters $\boldsymbol{\theta}^*$ and $\boldsymbol{T}^*$ $(P^*[\boldsymbol{t}])$ from the last training epoch, the expectation term (cf. Appendix) of the EM is equal to

$$E = \int_{\mathcal{T}_{k_1}^*} \cdots \int_{\mathcal{T}_{k_M}^*} \sum_{m=1}^{M} \log[p(\boldsymbol{d}_m(\boldsymbol{\theta}), \boldsymbol{t}_m)] \prod_{\mu=1}^{M} p(\boldsymbol{t}_\mu \mid \boldsymbol{d}_\mu(\boldsymbol{\theta}^*))\, d\boldsymbol{t}_1 \cdots d\boldsymbol{t}_M. \tag{3}$$

The reverse order of intergration and summation in Eqn. 3 together with the condition of

$\int_{\mathcal{T}^*_{k_\mu}} p(\boldsymbol{t}_\mu \mid \boldsymbol{d}_\mu(\theta^*))\, d\boldsymbol{t}_\mu = 1$ results in

$$E = \sum_{m=1}^{M} \int_{\mathcal{T}^*_{k_m}} \log[p(\boldsymbol{d}_m(\boldsymbol{\theta}), \boldsymbol{t})]\, p(\boldsymbol{t} \mid \boldsymbol{d}_m(\boldsymbol{\theta}^*))\, d\boldsymbol{t}. \quad (4)$$

Considering the prior knowledge of class membership of vectors d_m, which reduces the variety of $\mathcal{T}^*_{k_m}$ to K disjoint $\mathcal{T}^*_k$, and $\log[p(\boldsymbol{d}_m(\boldsymbol{\theta}), \boldsymbol{t}_m)] = \log[p(\boldsymbol{d}_m(\boldsymbol{\theta}) \mid \boldsymbol{t}_m)] + \log[p(\boldsymbol{t}_m)]$, simplifies the E-step to

$$E = \sum_{k=1}^{K} \int_{\mathcal{T}^*_k} \sum_{m_k=1}^{M_k} \log[p(\boldsymbol{d}_{km_k}(\boldsymbol{\theta}) \mid \boldsymbol{t})]\, p(\boldsymbol{t} \mid \boldsymbol{d}_{km_k}(\boldsymbol{\theta}^*))\, d\boldsymbol{t} + \sum_{k=1}^{K} \int_{\mathcal{T}^*_k} \sum_{m_k=1}^{M_k} \log[p(\boldsymbol{t})]\, p(\boldsymbol{t} \mid \boldsymbol{d}_{km_k}(\boldsymbol{\theta}^*))\, d\boldsymbol{t} \quad (5)$$

subject to

$$\int_{\mathcal{T}^*_k} p(\boldsymbol{t})\, d\boldsymbol{t} = P[k], \quad (6)$$

where $P[k]$ denotes the prior probability of class index k, and $M_1 + M_2 + \ldots + M_K = M$.

Finally, the next estimation of parameters is obtained by maximizing both expectation terms in Eqn. 5. Using binary vectors $\boldsymbol{t}_k \in \mathcal{T}^*_k \subset \{-1, +1\}^J$, $\int$ is replaced by $\sum$, $p(\boldsymbol{t}) \rightarrow P[\boldsymbol{t}]$, and each domain $\mathcal{T}^*_k$ contains a finite number of possible reference vectors for class k.

For the maximization of the expectation term (5), the gradient according to θ is

$$\frac{\partial E}{\partial \theta} = \sum_{k=1}^{K} \sum_{\boldsymbol{t}}^{\mathcal{T}^*_k} \sum_{m_k=1}^{M_k} \frac{P[\boldsymbol{t} \mid \boldsymbol{d}_{km_k}(\boldsymbol{\theta}^*)]}{p(\boldsymbol{d}_{km_k}(\boldsymbol{\theta}) \mid \boldsymbol{t})} \frac{\partial p(\boldsymbol{d} \mid \boldsymbol{t})}{\partial \boldsymbol{d}} \frac{\partial \boldsymbol{d}}{\partial \boldsymbol{\theta}}. \quad (7)$$

Analogously,

$$\frac{\partial E}{\partial P[\boldsymbol{t}]} = \sum_{m_k=1}^{M_k} \frac{P[\boldsymbol{t} \mid \boldsymbol{d}_{km_k}(\boldsymbol{\theta}^*)]}{P[\boldsymbol{t}]} + const. \quad (8)$$

for all $\boldsymbol{t} \in \bigcup_{k \in \mathcal{K}} \mathcal{T}^*_k$, where *const.* represents the Lagrangian multiplier from the additional linear constraint $P[k = 1] + P[k = 2] + \ldots + P[k = K] = 1$ together with Eqn. 6. Setting Eqn. 8 to zero, the summation over all $\boldsymbol{t} \in \mathcal{T}^*_k$ yields $const. = -M_k$, and for all $k \in \mathcal{K}$ and $\boldsymbol{t} \in \mathcal{T}^*_k$ the estimation of $P[\boldsymbol{t}]$ is obtained by

$$P[\boldsymbol{t}] = \frac{1}{M_k} \sum_{m_k=1}^{M_k} P[\boldsymbol{t} \mid \boldsymbol{d}_{km_k}(\boldsymbol{\theta}^*)], \quad (9)$$

where $P[\boldsymbol{t} \mid \boldsymbol{d}_{km_k}(\boldsymbol{\theta}^*)]$ is infered from the estimated parameters of the last training epoch:

$$P[\boldsymbol{t} \mid \boldsymbol{d}_{km_k}(\boldsymbol{\theta}^*)] = \frac{p(\boldsymbol{d}_{km_k}(\boldsymbol{\theta}^*) \mid \boldsymbol{t})\, P^*[\boldsymbol{t}]}{\sum_{\boldsymbol{\tau}}^{\mathcal{T}^*_k} p(\boldsymbol{d}_{km_k}(\boldsymbol{\theta}^*) \mid \boldsymbol{\tau})\, P^*[\boldsymbol{\tau}]}. \quad (10)$$

III. Regularization

In order to prevent the distributions of classes in decision space from overlapping, and for the benefit of ECOC, increasing the minimum distance between codewords of classes is highly desirable. In the following, we apply a regularization principle during the learning process for generation of reference vectors. The regularization term is defined as the logarithm of the product of all distances between reference vectors, or

$$R = \sum_{k=1}^{K-1} \sum_{l=k+1}^{K} \log \|\boldsymbol{t}_k - \boldsymbol{t}_l\|_2, \quad (11)$$

and favours the generation of higher Hamming distances between reference vectors. Hence, the cost function of the overall optimization is defined as

$$F = E(\mathcal{X}, \mathcal{K}, \boldsymbol{\theta}, \boldsymbol{T}^*) + \lambda \cdot R(\boldsymbol{T}^*), \quad (12)$$

where λ is called "regularization parameter".

After each training epoch the next potential reference vector of class k is calculated by

$$\boldsymbol{t}_k = \boldsymbol{t}^*_k \oplus (\nabla_k E + \lambda \cdot \nabla_k E). \quad (13)$$

The ∇_k is called pseudo-gradient operator because in the theory of the differential calculus there is no definition of gradients in $\{-1, +1\}^J$. The definition in Eqn. 13 roughly follows the rules of numerical differentiation. The feasible candidates for $\boldsymbol{t}_k$ are elements from the $\mathcal{N}_1(\boldsymbol{t}^*_k)$ neighborhood ($\boldsymbol{t} \in \mathcal{T}^*_k$ which maximally differ from $\boldsymbol{t}^*_k$ in one component) of the current reference vector. The operator $\oplus$ is defined as $\boldsymbol{t}_k \oplus \boldsymbol{v} := \arg\min_{\boldsymbol{t} \in \mathcal{T}^*_k} \|\boldsymbol{t}_k + \boldsymbol{v} - \boldsymbol{t}\|_2$ for all $\boldsymbol{v} \in \mathcal{R}^J$. The restriction to $\mathcal{N}_1(\boldsymbol{t}_k)$ neighborhoods of possible candidates is important. Otherwise, large modifications of the reference vectors would change the composition of the corresponding dichotomies such dramatically that the after-training meaning of the learned parameters from the previous training epoch would be destroyed.

The components of pseudo-gradient $\nabla_k E$ would be consequently defined by

$$\left.\frac{\partial E}{\partial t_{kj}}\right|_{\boldsymbol{t}^*_k} = \frac{E(\boldsymbol{t}^*_k + \epsilon \cdot \boldsymbol{e}_j) - E(\boldsymbol{t}^*_k)}{|\epsilon|}, \quad (14)$$

where all e_{ji} of $\boldsymbol{e}_j$ are equal to δ_{ji}, and $\epsilon = -2t^*_{kj}$ because of the binary properties of codewords. Hence, the calculation of $\nabla_k E$ would require multiple repeated runs of the last training epoch based on slightly modified reference vectors.

A definition of $\nabla_k E$ with lower complexity can be obtained by utilizing the probabilities of vectors $t \in \mathcal{T}^*_k$ from the previous training epoch (see Eqn. 9). The proposed definition is equal to

$$\left.\frac{\partial E}{\partial t_{kj}}\right|_{\boldsymbol{t}^*_k} = \frac{P[\boldsymbol{t}^*_k + \epsilon \cdot \boldsymbol{e}_j] - P[\boldsymbol{t}^*_k]}{|\epsilon|}. \quad (15)$$

The pseudo-gradient of the regularization term is analogously derived from the rules of numerical differentiation:

$$\frac{\partial}{\partial t_{kj}} R \Big|_{t_k^*} = \tag{16}$$

$$\sum_{l=1}^{k-1} \frac{1}{\|t_k - t_l\|_2} \cdot \frac{\partial}{\partial t_{kj}} \|t_k - t_l\|_2 \Big|_{t_k^*}$$

$$+ \sum_{l=k+1}^{K} \frac{1}{\|t_k - t_l^*\|_2} \cdot \frac{\partial}{\partial t_{kj}} \|t_k - t_l^*\|_2 \Big|_{t_k^*}.$$

The differential ($k \neq l$) is again defined as

$$\frac{\partial}{\partial t_{kj}} \|t_k - t_l^*\|_2 \Big|_{t_k^*} = \tag{17}$$

$$\frac{\|t_k^* + \epsilon \cdot e_j - t_l^*\|_2 - \|t_k^* - t_l^*\|_2}{|\epsilon|}.$$

In order to guarantee that $F(\mathcal{X}, \mathcal{K}, \boldsymbol{\theta}, \boldsymbol{T}) \geq F(\mathcal{X}, \mathcal{K}, \boldsymbol{\theta}^*, \boldsymbol{T}^*)$, the iterative procedure (13) of the discrete optimization must be performed semi-implicitly and the next set of possible candidates of reference vectors $\boldsymbol{T}$ must be chosen properly. This is because of the non-local properties of the pseudo-gradients. The calculation of $\nabla_k R$ for the k-th reference vector depends on the already-adapted reference vectors $t_1 \ldots t_{k-1}$, and the still-unadapted vectors $t_k^* \ldots t_K^*$. The order of the sequence of updates influences the optimization result of the iteration step. A random choice of the order avoids systematical errors.

If the pseudo-gradient points into the direction of a possible candidate for t_k, it simply depends on a threshold whether the corresponding bit of the original reference vector changes from $+1 \rightarrow -1$ or $-1 \rightarrow +1$, or does not change. The threshold is practically determined by a stepsize.

There might be an interesting interpretation of the proposed cost function approach: structural risk minimization (SRM) and regularization. SRM [13] is a result of Statistical Learning Theory and emphasizes the minimization of the classification error on training samples as well as the generalization on test samples by optimal model selection. The impact of the regularizer R complicates the generation of trivial two-class problems (dichotomies) of each parallel classifier (the simpliest dichotomy would consist of purely one class separated from all other classes). Hence, regularization increases the expressivity requirements which are imposed on the selected model by the composed classification task. This may be interpreted as a type of SRM by configuration of the training data and not by model selection.

IV. Example

The following example is based on the data sets from a realistic practical application – the recognition of handwritten digits ($K = 10$) taken from a mixture of the NIST databases [14]. The choice of the classification task is especially relevant because of its importance on the generation of optimal dichotomies according to their data structure. The training was not applied to the pure bitmap of scanned images but to the transformed representation of the data. Each image of a character was transformed into a high-dimensional vector ($N = 194$) by a feature extraction method [15]. The feature extraction is performed by matching the original input images with a variety of artificial patterns, and each of this patterns is generated as a grid of equidistant lines or concentrical circles. The matching scores are equal to the input vector $\boldsymbol{x}$.

In a very first trial, the expectation term (5) was substituted by means of the empirical error of the decision rule (see [12]). Then

$$E \approx \sum_{k=1}^{K} P[\boldsymbol{d}(\boldsymbol{x}, \boldsymbol{\theta}) = t_k \mid \boldsymbol{x} \in \mathcal{X}_k] \cdot P[k] \tag{18}$$

is inspired from the supposed similarity between maximizing E and minimizing the empirical error of the decision rule under the assumption of error-correcting training of the classifier, and $\mathcal{T}_k = \mathcal{N}_e(t_k)$ where $e = \lfloor \frac{h_{min}}{2} \rfloor$, and h_{min} expresses the minimal Hamming distance between any codewords of $\boldsymbol{T}$.

The following results are based on the minimization of the empirical error instead of applying Eqn. 7. The employed learning algorithm is an extended version of the classical error-correcting Madaline algorithm [16] (see [12]) which is separately applied to the single dichotomies of each parallel network and exclusively depends on the binary reference values $t_{kj} \in \{-1, +1\}$ of components. The one per class coding (1-out-of-10) served as a startcode for the iterative optimization process. The number of parallel components is equal to the number of classes ($J = 10$). Each parallel network consists of 5 neurons and the number of free parameters is equal to $5 \times 194 \approx 1000$. The sample size of $\mathcal{S}$ and the size of an i.i.d. set of test examples is 10×2000 respectively.

The influence of the regularization parameter λ is presented in Figure 3. It shows averaged results (incl. standard deviation) from 10 repeated training processes. The error rate of the constant 1-out-of-10 coding serves as a reference. The polynomial least squares approximation of the error rates versus λ is given by $\lambda \mapsto 2.5707 - 0.9489 \cdot \lambda + 0.6436 \cdot \lambda^2$. For the given configurations the optimal value of parameter was found to be $\lambda_{opt} \approx 0.8$. For larger values of λ, the generation of reference vectors would have resulted in ma-

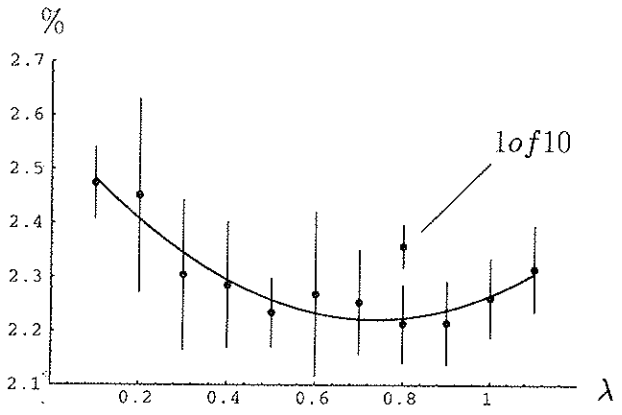

Figure 3: Error Rates of Test Samples after Training vs. Regularization Parameter.

ximal Hamming distances, however almost independently from the underlying database, and the generalization capabilities would have become worse.

Eqn. 19 presents the codematrix after 18 training epochs of 2000 online parameter adaptations of $\boldsymbol{\theta}$ respectively:

$$\boldsymbol{T}_{opt} =$$

$$\begin{pmatrix} +1 & -1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 & -1 \\ -1 & +1 & -1 & -1 & +1 & -1 & -1 & -1 & +1 & +1 \\ -1 & -1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 & -1 \\ +1 & -1 & -1 & +1 & -1 & -1 & +1 & -1 & -1 & +1 \\ -1 & -1 & -1 & -1 & +1 & +1 & +1 & -1 & -1 & -1 \\ -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 & -1 \\ +1 & +1 & +1 & -1 & -1 & -1 & +1 & -1 & -1 & -1 \\ +1 & +1 & -1 & -1 & +1 & -1 & -1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 & -1 & +1 & -1 & -1 & +1 & -1 \\ -1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 & -1 & +1 \end{pmatrix}. \quad (19)$$

Although the probability of error-free decisions – correct decisions at any component – became even worse, the generalization error of the decision rule – exploiting the error-correcting properties of $\boldsymbol{T}_{opt}$ – was reduced about 12% from 2.35% (**1-out-of-10**) to 2.07%. The training results of the overall classification rule and its components have always met 0.00% classification error.

$$\{h_{kl}\} = \begin{pmatrix} & 6 & 5 & 6 & 7 & 6 & 4 & 4 & 4 & 4 \\ 6 & & 5 & 6 & 5 & 6 & 6 & 4 & 6 & 6 \\ 5 & 5 & & 5 & 4 & 5 & 5 & 5 & 5 & 5 \\ 6 & 6 & 5 & & 5 & 6 & 4 & 6 & 4 & 6 \\ 7 & 5 & 4 & 5 & & 5 & 5 & 5 & 5 & 5 \\ 6 & 6 & 5 & 6 & 5 & & 6 & 4 & 4 & 4 \\ 4 & 6 & 5 & 4 & 5 & 6 & & 4 & 6 & 6 \\ 4 & 4 & 5 & 6 & 5 & 4 & 4 & & 6 & 6 \\ 4 & 6 & 5 & 4 & 5 & 4 & 6 & 6 & & 6 \\ 4 & 6 & 5 & 6 & 5 & 4 & 6 & 6 & 6 & \end{pmatrix}. \quad (20)$$

The bit-distances between the reference vectors of (19) are displayed in Eqn. 20. Whereas the minimal Hamming distance $h_{min} = 4$, the averaged Hamming distance between reference vectors $\frac{1}{(K-1)\cdot K/2} \sum_{k=1}^{K-1} \sum_{l=k+1}^{K} h_{kl}$ is even larger than 5. Although the advanced ECOC principally benefits the classification system, the increased correlations between errors of different parallel components for non-trivial coding ($\neq$ one per class) remain a problem. Eqn. 23 shows the error correlations between NN_j, $j = 1, 2, 3$ and 4, according to 1-out-of-10 output coding. Analogously, Eqn. 24 holds for the optimized codematrix in Eqn. 19. The correlation matrices are calculated by $\boldsymbol{R} = \frac{1}{100} \sum_{m=1}^{M} \frac{1}{M} \boldsymbol{r}(\boldsymbol{x}_m)\boldsymbol{r}(\boldsymbol{x}_m)^T$ where

$$r_j(\boldsymbol{x}) := \begin{cases} 1 & , \text{ if } d_j(\boldsymbol{x}_m) \neq t_{k_m j} \\ 0 & , \text{ otherwise} \end{cases}. \quad (21)$$

$$\boldsymbol{R}_{1of10} = \begin{pmatrix} 0.310 & 0.010 & 0.020 & 0.010 & \cdots \\ 0.010 & 0.330 & 0.030 & 0.015 & \cdots \\ 0.020 & 0.030 & 0.695 & 0.100 & \cdots \\ 0.010 & 0.015 & 0.050 & 0.880 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \quad (22)$$

$$\boldsymbol{R}_{opt} = \begin{pmatrix} 2.010 & 0.510 & 0.670 & 0.435 & \cdots \\ 0.510 & 2.685 & 0.440 & 0.335 & \cdots \\ 0.670 & 0.440 & 3.420 & 0.485 & \cdots \\ 0.435 & 0.335 & 0.485 & 1.825 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \quad (23)$$

The definition of error-correcting output codes therefore can not be a matter of *a priori* choices. The presented example shows the superiority of optimized reference vectors even for a considerably higher correlation between errors of parallel components and a rather low complexity of outputs. Further experiments confirmed the earlier results [8] that non-trivial coding is more advantageous in cases where overfitting is likely [12].

V. Conclusion

Theoretical results in ECOC as well as practical applications have generally considered the case where the number of outputs is much larger than the number of classes [6, 8, 17]. Results about using error-correcting codes with small code length for classification problems are not known to the author. On the other hand, the use of $J \gg K$ is not recommendable because of the growing size of the classification system and the increased computational costs for the decision process.

We have shown that starting with a trivial 1-out-of-K code, more efficient codes can be produced by iterative optimization. Even for the case of a restricted number of parallel components $J = K$, the one-per-class output coding is not optimal. A more efficient code has been generated by an iterative optimization method based on a maximum likelihood technique combined with a regularization prinicple for ECOC. The generation of reference vectors is not based on *a priori* appreciated codes but rather depends on an iterative method which considers both the classification

system and the structure of data from the underlying classification task. In the example, the generalization of the classifier for handwritten characters was improved about more than 10%.

Appendix

The EM algorithm is a general method for solving ML estimation problems given incomplete data. Let

$$L(\boldsymbol{\theta}) = \sum_{m=1}^{M} \log p(\boldsymbol{x}_m; \boldsymbol{\theta})$$

denote the log-likelihood function of the training set S where $p(\boldsymbol{x}; \boldsymbol{\Theta})$, which depends on a set of parameters $\boldsymbol{\theta}$, denotes the density function of the probability of input patterns $\boldsymbol{x}$.

After simple manipulations and taking the conditional expectation $[\ldots; \boldsymbol{\theta}^*]$ (at parameters $\boldsymbol{\theta}^*$) with respect to suitably introduced data $(\boldsymbol{y}_m \mid \boldsymbol{x}_m)$, one obtains $L(\boldsymbol{\theta}) = E(\boldsymbol{\theta}, \boldsymbol{\theta}^*) + H(\boldsymbol{\theta}, \boldsymbol{\theta}^*)$ where

$$\begin{aligned} E(\boldsymbol{\theta}, \boldsymbol{\theta}^*) &= [\sum_{m=1}^{M} \log p(\boldsymbol{x}_m, \boldsymbol{y}_m; \boldsymbol{\theta}) \; ; \; \boldsymbol{\theta}^*] \\ H(\boldsymbol{\theta}, \boldsymbol{\theta}^*) &= [\sum_{m=1}^{M} \log p(\boldsymbol{y}_m \mid \boldsymbol{x}_m; \boldsymbol{\theta}) \; ; \; \boldsymbol{\theta}^*] \end{aligned}$$

It can be shown that it suffices to maximize $E(\boldsymbol{\theta}, \boldsymbol{\theta}^*)$ if one aims to maximize the log-likelihood function $L(\boldsymbol{\theta})$.

The idea of the EM method is to produce a simpler ML estimation problem by any appropriate choice of "complete data". The most important prerequisite for the practicability of the EM algorithm is that the original likelihood function of the "incomplete data" $\boldsymbol{x}_m$ can be obtained by the likelihood function of the unknown "complete data" $(\boldsymbol{y}_m \mid \boldsymbol{x}_m)$ in form of "marginalization" which involves integrating out the so-called "complete data" [9, 10].

It follows the outline of the EM algorithm: the algorithm starts with an arbitrary initial guess and denotes by $\boldsymbol{\theta}^*$ the current estimate of the parameters $\boldsymbol{\theta}$ after a number of training epochs. The next iteration step consists of two steps, namely the E-step (expectation):

$$\begin{aligned} E(\boldsymbol{\theta}, \boldsymbol{\theta}^*) &= \int_{\mathcal{Y}_1} \cdots \int_{\mathcal{Y}_M} \sum_{m=1}^{M} \log p(\boldsymbol{x}_m, \boldsymbol{y}_m; \boldsymbol{\theta}) \cdot \\ &\quad \prod_{m=1}^{M} p(\boldsymbol{y}_m \mid \boldsymbol{x}_m; \boldsymbol{\theta}^*) \, d\boldsymbol{y}_1 \cdots d\boldsymbol{y}_M, \end{aligned}$$

and the M-Step (maximization): $\max_{\boldsymbol{\theta}} E(\boldsymbol{\theta}, \boldsymbol{\theta}^*)$.

If $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^*)$ is continuous in both $\boldsymbol{\theta}$ and $\boldsymbol{\theta}^*$, it has been proven that the algorithm converges to a local maximum of the log-likelihood function $L(\boldsymbol{\theta})$ [9, 10].

References

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

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

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

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

[5] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

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

[7] E. Mayoraz and M. Moreira. On the decomposition of polychotomies into dichotomies. In *Proceedings of the 14th International Conference on Machine Learning*, pages 219–226, 1996.

[8] 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.

[9] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B*, 39(1), 1977.

[10] R.A. Redner and H.F. Walker. Mixture densities, maximum likelihood and the EM algorithm. *Siam Review*, 26(2):195–239, 1984.

[11] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.

[12] W. Utschick. *Error-Correcting Classification Based on Neural Networks.* PhD thesis, Technische Universität München, 1998. to appear.

[13] V.N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, 1995.

[14] The state-of-the-art in OCR is subject of NIST conference. *Intelligence*, 9(6), 1992.

[15] W. Utschick, P. Nachbar, C. Knobloch, A. Schuler, and J.A. Nossek. The evaluation of feature extraction criteria applied to neural network classifiers. In *Proceedings of the 3th International Conference on Document Analysis and Recognition*, pages 315–318. IEEE Computer Society Press, 1995.

[16] B. Widrow and M.A. Lehr. 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. *Proceedings of the IEEE*, 78(9):1415–1441, 1990.

[17] 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.