

Recursive Training of Neural Networks for Classification

Mayer Aladjem

Department of Electrical and Computer Engineering, Ben-Gurion University of the Negev,

P.O.B. 653, 84105 Beer-Sheva, Israel

List of Figures

1	AA network. It is trained to map input vectors into themselves in such a way that $y_j' = y(\mathbf{x}_j'; \mathbf{w}^*)$ has overlapped class-conditional densities.	4
2	Linear mapping $y = \mathbf{w}^T \mathbf{x}$ in a two-dimensional \mathbf{x} -space. Parzen estimators $\hat{p}(\mathbf{w}^T \mathbf{x} \omega_1)$ and $\hat{p}(\mathbf{w}^T \mathbf{x} \omega_2)$ of the class-conditional densities along \mathbf{w}	6
3	Original data X_t into the (x_1, x_2) -plane.	8
4	Transformed data X_t' after the RCS at 64°	8
5	Result of the method of Section 2. $E^{PF}(\mathbf{w})$ for various directions of \mathbf{w} into (x_1, x_2) -plane: original data (path "—"); transformed data after successive RCS at 64° (path "...") and 19° (path "-.-").	8
6	Result of the method [2]. $E^{PF}(\mathbf{w})$ for various directions of \mathbf{w} into (x_1, x_2) -plane: original data (path "—"); transformed data after successive RCS at 64° (path "...") and 160° (path "-.-").	8
7	OCR data. 100 different random initializations of the training.	13

Recursive Training of Neural Networks for Classification

Mayer Aladjem

Department of Electrical and Computer Engineering, Ben-Gurion University of the Negev,
P.O.B. 653, 84105 Beer-Sheva, Israel

Abstract

A method for recursive training of neural networks for classification is proposed. It searches for the discriminant functions corresponding to several small local minima of the error function. The novelty of the proposed method lies in the transformation of the data into new training data with a deflated minimum of the error function and iteration to obtain the next solution. A simulation study and a character recognition application indicate that the proposed method has the potential to escape from local minima and to direct the local optimizer to new solutions.

Index Terms—Neural networks for classification, auto-associative network, linear and non-linear classification functions, projection pursuit, structure removal.

1 Introduction

We discuss the training of neural networks for classification. The training is carried out by minimizing an error function which allows the outputs of the network to represent classification functions [6, page 194]. We are interested in error functions which are highly nonlinear with respect to the adjustable weights of the networks. Such objectives appear in the multi-layer networks for classification [6] and in the networks for non-parametric linear discriminant analysis [2], [3], [7, pages 257-280]. In these cases the training must be carried out by an iterative optimization algorithm. The primary goal is to find the global minimum of the error function. By a naive use of a training algorithm (a local minimizer of the error function) the computed value for the observed minimum can be merely a local minimum. The solution depends strongly on the starting point of the local optimizer.

In this paper we propose a recursive method for searching for several small local minima of the error functions. Our method is not a global minimization method, but rather a tool for escaping from a minimum already found and directing the local optimizer to a new solution.

In Section 2 we propose our method. Sections 3 and 4 contain the results and analysis of the experiments with linear and non-linear classification functions and comparative studies of other methods for the minimization of error functions. Pseudo-code of a computational procedure for our method is presented in an appendix at the end of the paper.

Some preliminary results of our work were presented in [3], [4]. This paper contains a more thorough analysis and more complete results.

2 A Method for Recursive Training

Suppose we are given training data $(\mathbf{x}_1, \mathbf{c}_1), (\mathbf{x}_2, \mathbf{c}_2), \dots, (\mathbf{x}_{N_t}, \mathbf{c}_{N_t})$ comprising a set $X_t = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{N_t}\}$ of N_t training observations in n -dimensional sample space ($\mathbf{x}_j \in R^n$, $n \geq 2$) and their associated class indicator vectors \mathbf{c}_j , $j = 1, 2, \dots, N_t$. We discuss a two class problem and we require that \mathbf{c}_j is a two-dimensional vector $\mathbf{c}_j = (c_{1j}, c_{2j})^T$ which shows that \mathbf{x}_j belongs to one of the classes ω_1 or ω_2 . The components c_{1j} , c_{2j} are defined to be one or zero according to the class-membership of \mathbf{x}_j , i.e. $c_{1j} = 1$, $c_{2j} = 0$ for $\mathbf{x}_j \in \omega_1$ and $c_{1j} = 0$, $c_{2j} = 1$ for $\mathbf{x}_j \in \omega_2$. The class-indicator vectors \mathbf{c}_j imply decomposition of the set X_t into two subsets corresponding to the unique classes. We denote by N_{t_i} the number of the training observations from class ω_i , for $i = 1, 2$. We require a normalization of the training data X_t , called sphering [8] (or whitening [6]). For the sphered X_t the pooled sample covariance matrix becomes the identity matrix and the sample mean vector is a zero vector. In the remainder of the paper, all operations are performed on the sphered data.

We denote $y(\mathbf{x}; \mathbf{w})$, $\mathbf{x} \in R^n$, $y \in R^1$, to be the mapping function of a network for classification, with \mathbf{x} an n -dimensional input vector, and \mathbf{w} a vector comprising the adjustable weights of the network. The goal of the training of the network for classification is to adjust the weights \mathbf{w} in order to get a mapping function $y(\mathbf{x}; \mathbf{w})$ which separates the vectors \mathbf{x} from different classes. The latter is achieved by minimization of an error function which measures the class-overlap of the responses $y_j = y(\mathbf{x}_j; \mathbf{w})$ for the training vectors $\mathbf{x}_j \in X_t$.

We require a normalization which implies zero mean and unit variance of y_j .

Here we propose a method for recursive searching for several small local minima of an error function of a classification network. In this section we do not choose a specific error function because our method can be applied to any error function. We proceed as follows. Using a local optimizer we obtain a vector of weights \mathbf{w}^* related to a local minimum of the error function. Then we transform the training data X_t to X_t' increasing class overlap of the responses $y_j' = y(\mathbf{x}_j'; \mathbf{w}^*)$ for the transformed vectors $\mathbf{x}_j' \in X_t'$ (we deflate the error function at the solution \mathbf{w}^*), and iterate to obtain a new solution using X_t' .

The main point of the method is the procedure for transforming X_t to X_t' called *reduction of the class separation* (RCS). We obtain X_t' in the output of an *auto-associative* (AA) network (Fig.1). We train this network by minimizing an error function

$$J(\mathbf{u}) = (1 - \nu)J^{MS}(\mathbf{u}) + \nu\Omega(\mathbf{u}). \quad (1)$$

Here,

$$J^{MS}(\mathbf{u}) = \frac{1}{N_t} \sum_{j=1}^{N_t} (\mathbf{x}_j' - \mathbf{x}_j)^T (\mathbf{x}_j' - \mathbf{x}_j) \quad (2)$$

is the *mean-squared* (MS) error of the AA network, $\Omega(\mathbf{u})$ is the penalty function and ν ($0 \leq \nu \leq 1$) is the parameter controlling the extent to which the penalty term $\Omega(\mathbf{u})$ influences the form of the solution. In (2) $\mathbf{x}_j' = \mathbf{r}(\mathbf{x}_j; \mathbf{u})$ is the AA network mapping as a function of the input vectors \mathbf{x}_j and the vector \mathbf{u} comprises the adjustable weights of the AA network. The penalty term $\Omega(\mathbf{u})$ measures the departure of $y_j' = y(\mathbf{x}_j'; \mathbf{w}^*)$ from the targets q_j having overlapped class-conditional densities. We define $\Omega(\mathbf{u})$ to be the MS error

$$\Omega(\mathbf{u}) = \frac{1}{N_t} \sum_{j=1}^{N_t} [y_j' - q_j]^2. \quad (3)$$

An AA mapping $\mathbf{x}_j' = \mathbf{r}(\mathbf{x}_j; \mathbf{u})$ which provides a good fit of the outputs \mathbf{x}_j' and the inputs \mathbf{x}_j gives a small value for $J^{MS}(\mathbf{u})$ (2), while one which produces output \mathbf{x}_j' with overlapped class-conditional densities for $y_j' = y(\mathbf{x}_j'; \mathbf{w}^*)$ gives a small value for $\Omega(\mathbf{u})$ (3).

Figure 1: AA network. It is trained to map input vectors into themselves in such a way that $y_j' = y(\mathbf{x}_j'; \mathbf{w}^*)$ has overlapped class- conditional densities.

Minimizing the total error $J(\mathbf{u})$ (1) we obtain an AA mapping $\mathbf{x}_j' = \mathbf{r}(\mathbf{x}_j; \mathbf{u})$ which is a compromise between fitting the training data \mathbf{x}_j and increasing the class overlap for $y_j' = y(\mathbf{x}_j'; \mathbf{w}^*)$ (deflating the error function of the classification network at \mathbf{w}^*).

We obtain the targets q_j in the following way. We propagate the training data $\mathbf{x}_j \in X_t$, $j = 1, 2, \dots, N_t$ through the classification network and obtain the corresponding outputs $y_j = y(\mathbf{x}_j; \mathbf{w}^*)$. Then we compute the class-conditional means, variances and (cumulative) distribution functions for the sample y_j , $j = 1, 2, \dots, N_t$. We denote the obtained estimators by $\hat{m}_{y|\omega_i}$, $\hat{\sigma}_{y|\omega_i}^2$, and $\hat{F}(y|\omega_i)$ for $i = 1, 2$. Finally we obtain the targets q_j by the percentile transformation method [9, page 226]

$$q_j = \begin{cases} \text{for } y_j = y(\mathbf{x}_j; \mathbf{w}^*), \mathbf{x}_j \in \omega_1 (c_{1j} = 1) \\ \left[\Phi^{-1} \left(\hat{F}(y_j|\omega_1) \right) \right] (\hat{\sigma}_{y|\omega_1}^2 \pm \Delta\sigma^2)^{1/2} + (\hat{m}_{y|\omega_1} - \Delta m_1) \\ \\ \text{for } y_j = y(\mathbf{x}_j; \mathbf{w}^*), \mathbf{x}_j \in \omega_2 (c_{2j} = 1) \\ \left[\Phi^{-1} \left(\hat{F}(y_j|\omega_2) \right) \right] (\hat{\sigma}_{y|\omega_2}^2 \pm \Delta\sigma^2)^{1/2} + (\hat{m}_{y|\omega_2} - \Delta m_2). \end{cases}, j = 1, 2, \dots, N_t \quad (4)$$

Here, Φ^{-1} is the inverse of the standard normal (cumulative) distribution function Φ .

The targets q_j (4) have approximately normal class-conditional densities $N(\hat{m}_{y|\omega_1} - \Delta m_1, \hat{\sigma}_{y|\omega_1}^2 \pm \Delta\sigma^2)$ for $c_{1j} = 1$ and $N(\hat{m}_{y|\omega_2} - \Delta m_2, \hat{\sigma}_{y|\omega_2}^2 \pm \Delta\sigma^2)$ for $c_{2j} = 1$. Here, $\Delta\sigma^2$

($0 \leq \Delta\sigma^2 \leq 1$), $\Delta m_1, \Delta m_2$ are parameters which control the overlap of these densities. If we set $\Delta m_1 = 0, \Delta m_2 = 0$ and $\Delta\sigma^2 = 0$ we obtain minimal difference between q_j and y_j in the sense of the minimal relative entropy distance between the class-conditional densities of q_j and y_j [8, page 254]. If $(\hat{m}_{y|\omega_i} - \Delta m_i) = 0$ and $(\sigma_{y|\omega_i}^2 \pm \Delta\sigma^2) = 1$, for $i = 1, 2$ we obtain targets q_j with equal (fully overlapped) class-conditional densities $N(0, 1)$.

In order to keep the transformed data \mathbf{x}_j' as close to the original data $\mathbf{x}_j, j = 1, 2, \dots, N_t$ as possible, we search for the smallest values of the parameters ν and $\Delta\sigma^2$ which deflate the error function at \mathbf{w}^* . We choose the sign (+ or -) of the change ($\pm\Delta\sigma^2$) in order to approach $(\sigma_{y|\omega_i}^2 \pm \Delta\sigma^2)$ to 1. We assign the latter value to 1 if it crosses 1. For each $\Delta\sigma^2$ we compute the values of Δm_1 and Δm_2 using the normalization requirements for the responses $y_j' = y(\mathbf{x}_j'; \mathbf{w}^*)$:

-Zero pooled sample mean

$$\frac{N_{t1}}{N_t}(\hat{m}_{y|\omega_1} - \Delta m_1) + \frac{N_{t2}}{N_t}(\hat{m}_{y|\omega_2} - \Delta m_2) = 0; \quad (5)$$

-Unit pooled sample variance

$$\frac{N_{t1}}{N_t} [(\hat{\sigma}_{y|\omega_1}^2 \pm \Delta\sigma^2) + (\hat{m}_{y|\omega_1} - \Delta m_1)^2] + \frac{N_{t2}}{N_t} [(\hat{\sigma}_{y|\omega_2}^2 \pm \Delta\sigma^2) + (\hat{m}_{y|\omega_2} - \Delta m_2)^2] = 1. \quad (6)$$

Finally, we require the AA network to have non-linear activation functions (Fig.1) in order to fit the targets q_j which depend on \mathbf{x}_j non-linearly (4).

3 Experiments with Linear Classification Functions

In order to visualize and study the proposed method we apply it to a network which represents a linear classification function $y = y(\mathbf{x}; \mathbf{w}^*) = \mathbf{w}^{*T} \mathbf{x}$ with \mathbf{x} an arbitrary n -dimensional observation and \mathbf{w}^* an n -dimensional vector, called the discriminant vector. We require \mathbf{w}^* to have unit length which implies zero mean and unit variance for the responses $y_j = \mathbf{w}^{*T} \mathbf{x}_j$ for the sphered training data $\mathbf{x}_j \in X_t$.

Here we summarize a geometrical interpretation (Fig.2) of our method:

- The linear mapping $y = \mathbf{w}^{*T} \mathbf{x}$ can be interpreted as the projection of an arbitrary point

Figure 2: Linear mapping $y = \mathbf{w}^T \mathbf{x}$ in a two-dimensional \mathbf{x} -space. Parzen estimators $\hat{p}(\mathbf{w}^T \mathbf{x} | \omega_1)$ and $\hat{p}(\mathbf{w}^T \mathbf{x} | \omega_2)$ of the class-conditional densities along \mathbf{w} .

\mathbf{x} onto a directional vector \mathbf{w} in \mathbf{x} -space. As previously, we require \mathbf{w} to have unit length.

- The discriminant vector \mathbf{w}^* implies the minimal class-overlap of the projections $y_j = \mathbf{w}^{*T} \mathbf{x}_j$.
- The percentile transformation (4) reorganizes the projections y_j into targets q_j having normal class-conditional densities along \mathbf{w}^* .
- The AA network reorganizes the original training points \mathbf{x}_j into new points \mathbf{x}_j' increasing the overlap of the classes along \mathbf{w}^* .

In the following sections we study discriminant vectors \mathbf{w}^* which correspond to the local minima of an error function (*negative Patrick-Fisher distance* [7, pages 275-280])

$$E^{PF}(\mathbf{w}) = - \left\{ \int_{R^n} \left[\frac{N_{t1}}{N_t} \hat{p}(\mathbf{w}^T \mathbf{x} | \omega_1) - \frac{N_{t2}}{N_t} \hat{p}(\mathbf{w}^T \mathbf{x} | \omega_2) \right]^2 d\mathbf{x} \right\}^{1/2}. \quad (7)$$

Here, $\hat{p}(\mathbf{w}^T \mathbf{x} | \omega_1)$ and $\hat{p}(\mathbf{w}^T \mathbf{x} | \omega_2)$ denote the Parzen estimators of the class-conditional densities along an arbitrary vector \mathbf{w} in \mathbf{x} -space.

3.1 Synthetic Data

We ran experiments for a synthetic data having a highly nonlinear shape for $E^{PF}(\mathbf{w})$ (7).

The data contains 50 points per class drawn from two dimensional normal mixtures:

$$\text{for class } \omega_1: p(x_1, x_2 | \omega_1) = \frac{1}{3}N([-1.5 \ 0]^T, \Sigma) + \frac{1}{3}N([0.5 \ 3]^T, \Sigma) + \frac{1}{3}N([-1 \ -3]^T, \Sigma),$$

for class ω_2 : $p(x_1, x_2|\omega_2) = \frac{1}{3}N([-0.5 \ 3]^T, \Sigma) + \frac{1}{3}N([3 \ 0]^T, \Sigma) + \frac{1}{3}N([0.5 \ -3]^T, \Sigma)$. Here, $N([\mu_1 \ \mu_2]^T, \Sigma)$ denotes bivariate normal density with a mean vector $[\mu_1 \ \mu_2]^T$ and a diagonal covariance matrix $\Sigma = \text{diag}(0.1, 0.2)$. Fig.3 presents the data after sphering. For this data we computed $E^{PF}(\mathbf{w})$ (7) for the vectors \mathbf{w} directed under different angles with respect to x_1 -axis (Coordinate 1 in Fig.3) using a smoothing parameter $h = 0.1$ for the Gaussian kernels of $\hat{p}(\mathbf{w}^T \mathbf{x}|\omega_1)$ and $\hat{p}(\mathbf{w}^T \mathbf{x}|\omega_2)$ [7, page 277]. The solid path "—" in Fig.5 presents $E^{PF}(\mathbf{w})$. We observe local minima at angles 19° , 64° , 105° , 128° and 162° .

3.2 Experiments with the RCS of Section 2

As we said, the main point of our method is the procedure for reduction of the class separation (RCS) explained in Section 2. We ran the RCS along the direction (\mathbf{w}^*) under 64° which corresponds to a local minimum $E^{PF}(\mathbf{w}^*) = -0.65$ (see "—", Fig.5). Here we study a situation which is highly unfavorable to our procedure because we deflate a local minimum at 64° which has a small value and is located in an area having a sharp oscillation of $E^{PF}(\mathbf{w})$. In this situation we must reorganize the data heavily in order to eliminate the minimum at 64° .

Here we present some details for the RCS at 64° . We set $\Delta\sigma^2 = 1$ in (4) and $\nu = 0.7$ in (1) and trained a two layer AA network having 50 hidden units with sigmoid activation functions by minimizing $J(\mathbf{u})$ (1). Then we propagated the original data X_t (Fig.3), through the AA network and obtained a transformed data X_t' (Fig.4). Comparing X_t' and X_t , we observe that a significant class overlap was gained for X_t' (Fig.4) along the direction under 64° . This is a desired result because our goal is to eliminate the local minimum of $E^{PF}(\mathbf{w})$ at 64° .

We calculated $E^{PF}(\mathbf{w})$ for X_t' for different \mathbf{w} and show its path "... " in Fig.5. We observe that the RCS eliminated the minimum at 64° and succeeded in preserving the local minima at 19° , 128° and 162° which is desirable for our method. We obtained some increase of the values of the transformed $E^{PF}(\mathbf{w})$ at the latter locations, which is a consequence of the reorganization of X_t' .

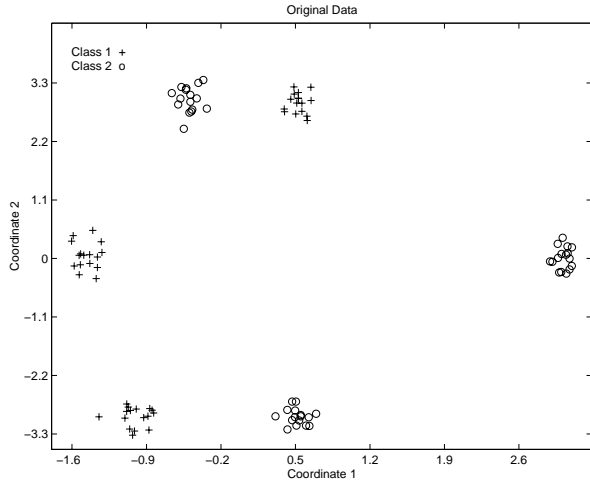


Figure 3: Original data X_t into the (x_1, x_2) -plane.

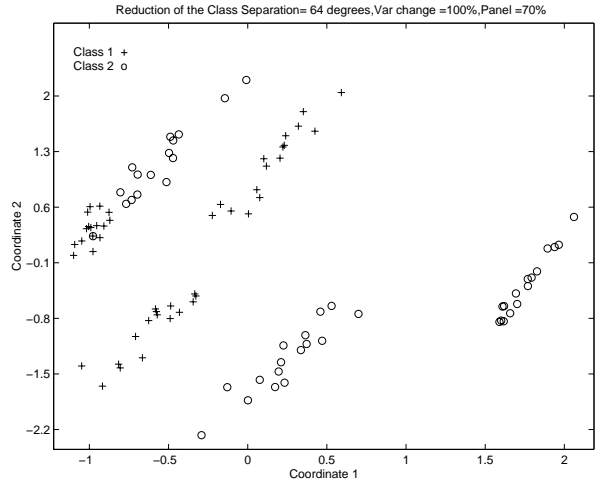


Figure 4: Transformed data X'_t after the RCS at 64° .

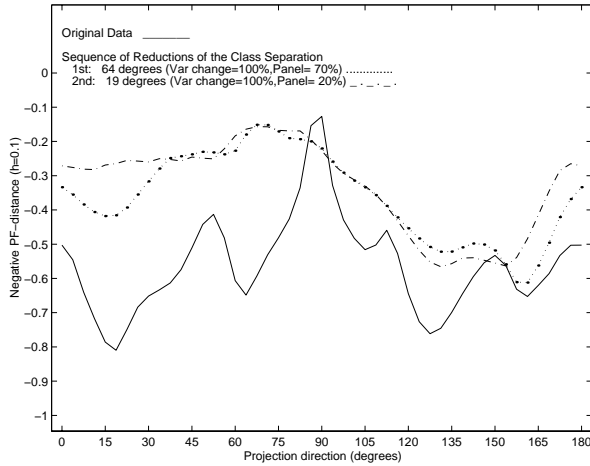


Figure 5: Result of the method of Section 2. $E^{PF}(\mathbf{w})$ for various directions of \mathbf{w} into (x_1, x_2) -plane: original data (path "—"); transformed data after successive RCS at 64° (path "...") and 19° (path "-.-").

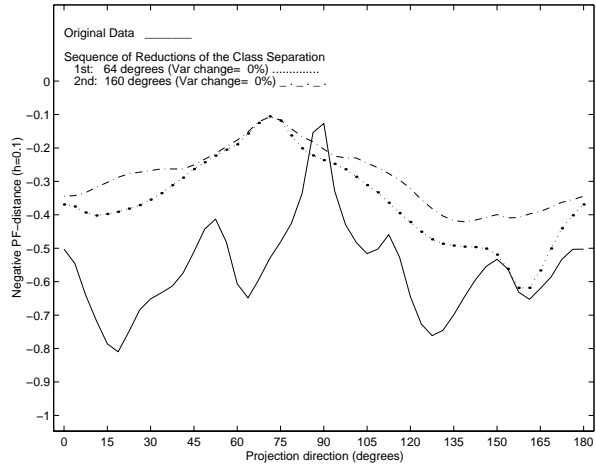


Figure 6: Result of the method [2]. $E^{PF}(\mathbf{w})$ for various directions of \mathbf{w} into (x_1, x_2) -plane: original data (path "—"); transformed data after successive RCS at 64° (path "...") and 160° (path "-.-").

We carried out RCS for a grid of $(\Delta\sigma^2, \nu)$ -values defined by the outer product of $\Delta\sigma^2 = (0.0, 0.25, 0.5, 0.75, 1.0)$ and $\nu = (0.1, 0.2, \dots, 0.9)$ and did not find better preservation of the shape of the $E^{PF}(\mathbf{w})$ than the result (path "...") shown in Fig.5. Purposely, we trained a large AA network which can store any rearrangement of the training points perfectly. This is a situation of over-fitting which is undesirable in view of the generalization ability of the network [6, Chapter 9]. In our method the purpose of the AA network is to fit (not to generalize) the input vectors and we think of the over-fitting as desirable. Finally the path "... in Fig.5 presents the result of the best fit of the original data for the deflated $E^{PF}(\mathbf{w})$ at 64° .

We iterated the RCS, choosing to eliminate the local minimum at 19° for the transformed $E^{PF}(\mathbf{w})$. The path "-.-" in Fig.5 represents the best result obtained for $\Delta\sigma^2 = 1$ and $\nu = 0.2$. We observe that we eliminated the local minima at 64° and 19° , while preserving the local minima at 128° for $E^{PF}(\mathbf{w})$ which is desirable for our method.

3.3 Experiments with the RCS in [2]

In [2] we proposed a method for RCS for linear classification functions. This transforms the training points into new points which have overlapped normal class-conditional densities along \mathbf{w}^* leaving all directions orthogonal to \mathbf{w}^* unchanged. We ran this method for the synthetic data set used in the previous Section 3.2. We carried out two successive RCS along the directions (\mathbf{w}^*) under 64° and 160° . Fig.6 presents the result obtained. We observed that the first RCS (at 64°) smoothed the $E^{PF}(\mathbf{w})$ (path "...") in the range $0^\circ - 150^\circ$ eliminating the local minima 19° and 128° , and the second RCS (at 160°) smoothed the $E^{PF}(\mathbf{w})$ totally eliminating all local minima. Comparing Fig.5 and Fig.6 we conclude that the proposed RCS in Section 2 preserved $E^{PF}(\mathbf{w})$ in the entire space better than method [2].

3.4 Remarks

Here, we summarize our observations from the experiments with linear classification functions.

- We observed that our new method (Section 2) preserved the shape of the $E^{PF}(\mathbf{w})$ better than method [2] for the situations of small local minima in the area of a sharp oscillation of $E^{PF}(\mathbf{w})$ (such situations were discussed in Sections 3.2 and 3.3). In other situations these methods produce the same results. Considering the significant small computational complexity of the method [2] we recommend carrying out an extensive search for the local minima of $E^{PF}(\mathbf{w})$ by [2] (see for such experiments [5]) and then choosing some of the solutions corresponding to the small values of $E^{PF}(\mathbf{w})$, and finally running the new method (Section 2) at these solutions only. Proceeding in this way we could find extra local minima which have been missed by method [2] in situations of sharp oscillation of $E^{PF}(\mathbf{w})$.

- The value of $E^{PF}(\mathbf{w})$ at a local minimum \mathbf{w}^* increases monotonically by enlarging the values of the control parameter ν and $\Delta\sigma^2$ for the RCS. We found that ν influences the change of $E^{PF}(\mathbf{w})$ more strongly than $\Delta\sigma^2$ does. This stimulated us to implement the following procedure for choosing the appropriate values for ν and $\Delta\sigma^2$. We set $\Delta\sigma^2 = 1$ and ran RCS for progressively increased values for ν starting from $\nu = 0$. For each trial we evaluated the deflation of $E^{PF}(\mathbf{w})$ (see Appendix A and [2]) and stopped the trials when reaching a value for ν which deflates $E^{PF}(\mathbf{w})$. Then we set the latter value for ν and ran RCS for progressively increased values for $\Delta\sigma^2$ starting from $\Delta\sigma^2 = 0$ until we deflated $E^{PF}(\mathbf{w})$. Finally, we used the latter values of ν and $\Delta\sigma^2$ for the RCS of the training data in the recursive optimization.

In some applications we compared the values ν and $\Delta\sigma^2$ found by the exhaustive search for a grid of $(\nu, \Delta\sigma^2)$ -values defined by the outer product of $\nu = (0.1, 0.2, \dots, 0.9)$ and $\Delta\sigma^2 = (0.0, 0.25, 0.5, 0.75, 1.0)$ (9×5 runs of the RCS) and by the above-described procedure increasing ν and $\Delta\sigma^2$ by steps $\varepsilon_\nu = 0.1$ and $\varepsilon_{\Delta\sigma^2} = 0.25$. We found similar values by these procedures. Considering the significantly smaller number of iterations of the latter

procedure (on average 5 + 3 runs of the RCS) we recommend it for the implementations.

In Appendix A we present a pseudo-code of this procedure in the general case including the nonlinear discriminant functions. From experience, we found that $\varepsilon_\nu = 0.1$ and $\varepsilon_{\Delta\sigma^2} = 0.25$ are suitable values for data having dimension $n = 2 \div 20$. For higher dimensions some experimentation is needed for setting ε_ν and $\varepsilon_{\Delta\sigma^2}$.

- The purpose of the AA network in our method is to fit (not to generalize) the input vectors and we think the over-fitting to be desirable. Consequently the size of the AA network can be set to be as large as the run time limitations and computational resources allow.

- In Section 3.2 we deflated the $E^{PF}(\mathbf{w})$ at 64° and 19° using a large AA network. Nevertheless we observed a significant increase of the values of the transformed $E^{PF}(\mathbf{w})$ in the neighborhood of 64° and 19° (Fig.5) which is undesirable for our method. It seems reasonable to search for other (non-normal) density functions of q_j and/or other error functions of the AA network (non-MS error functions) which deflate $E^{PF}(\mathbf{w})$ and cause less reorganization of its shape. This is the object of our current research and it is beyond the scope of this work.

4 Experiments with Non-Linear Classification Functions

We ran an experiment with a data set containing 150 upper-case handwritten letters "M" and 450 lower-case handwritten letters "m". Nine numerical results from a feature extraction procedure (developed in a company in Israel) were recorded for each case. We divided the data into a training set and a test set. The training set contained $N_{t_1} = 75$ M-cases and $N_{t_2} = 225$ m-cases and the rest of the data was used for validation.

Using this data we trained a two layer classification network having three hidden units with sigmoid activation functions. We obtained the weights \mathbf{w}^* of the network minimizing

the MS error function [6, Chapter 6]

$$E^{MS}(\mathbf{w}) = \frac{1}{N_{t_1} + N_{t_2}} \sum_{j=1}^{N_{t_1} + N_{t_2}} [y(\mathbf{x}_j; \mathbf{w}) - (c_{1j} - c_{2j})]^2. \quad (8)$$

Here, $y(\mathbf{x}_j; \mathbf{w})$ denotes the mapping function of the network and the class indicators are $c_{1j} = 1$ and $c_{2j} = 0$ for \mathbf{x}_j representing the M -training cases, and $c_{1j} = 0$ and $c_{2j} = 1$ for the m -cases.

We compared the classification qualities of the networks trained by minimization of $E^{MS}(\mathbf{w})$ by our recursive method (Section 2) and by the conventional method with random initialization. We evaluated the classification quality by the percentage of the wrongly allocated test observations by the network [6, pages 222-226].

We carried out 100 runs of our method (Appendix A, *Mode = Local*), starting from different initial weights \mathbf{w}_0 of the classification network. They were drawn at random by a method proposed in [6, pages 260-262]. In Step 1 of our procedure (Appendix A) we carried out the conventional minimization of $E^{MS}(\mathbf{w})$ while in Step 5 we employed our recursive method. We used a two layer AA network having 6 hidden units with sigmoid activation functions. We carried out three successive RCS ($N_{RCS} = 3$) setting $\Delta\sigma^2 = 1$ and $\nu = 0.5$. We chose $\Delta\sigma^2 = 1$ which implies full overlap of the class-conditional densities of the targets q_j (4) and ensures the deflation of $E^{MS}(\mathbf{w})$ for a sufficiently large value of ν . We set $\nu = 0.5$ which was proved to be appropriate by a preliminary test. We minimized $E^{MS}(\mathbf{w})$ (8) and $J(\mathbf{u})$ (1) by a local optimizer E04UCF from the NAG library setting the number of major iterations to 50. In order not to favor our procedure by expanding the number of iterations in the minimization of $E^{MS}(\mathbf{w})$, we re-ran E04UCF with an extended number of the major iterations $50 \times N_{RCS}$ in Step 1. In the comparison we used the smallest error rate obtained.

Fig.7 shows the obtained result. Our recursive minimization (solid path) outperforms the conventional minimization (dashed path) for most of the initializations. The dots at the bottom of Fig.7 indicate the sequential number of the RCS with the smallest test error rate (. - first RCS, .. - second RCS and ... - third RCS). The dots which are missing indicate the random initializations for which the conventional training gave a smaller error

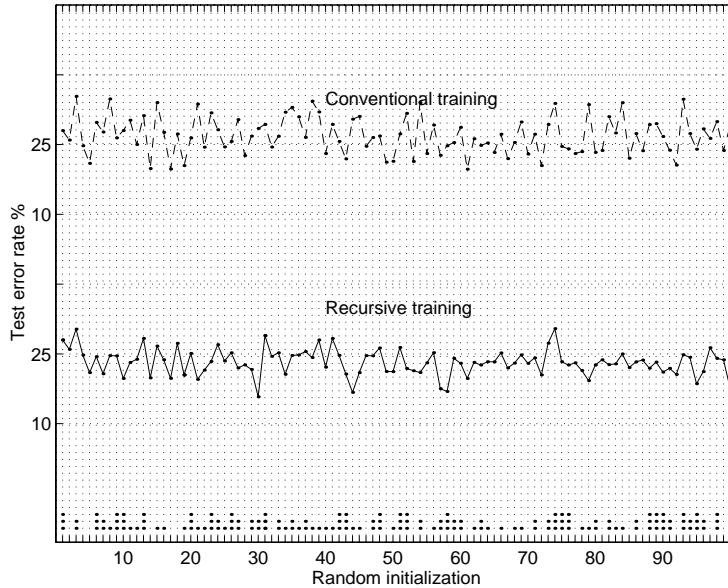


Figure 7: OCR data. 100 different random initializations of the training.

rate than our procedure.

We calculated the averaged difference of the test error rates of networks trained by the conventional minimization and our recursive minimization, which was 3%. We evaluated the significance of this difference by the paired t-test. The 99 percent confidence interval for the population mean difference was $3 \pm 1.4\%$ which confirms a decrease of the test error rate of the network trained by our recursive procedure. Also, our procedure gave results with error rates $\approx 15\%$ (random initializations 30, 44, 57, 58 and 100) which are significantly less than the smallest error rates $\approx 20\%$ of the conventional optimization. From the obtained results we conclude that our method has the potential to escape from local minima and to direct the local optimizer to new solutions.

The computational complexity of our method is higher than that of the conventional minimization and we do not think that our method is better than the conventional method with random initialization. We view it as a tool which can be combined with conventional optimization. In Appendix A we propose a computational procedure which implements the following options:

- *Mode = Local*: By our method we search for extra local minima in the neighborhood

of an "interesting" point \mathbf{w}_0 previously found by conventional optimization.

- *Mode = Global*: We apply our method for certain solutions found by conventional minimization. We carry out an extensive search by conventional optimization, choose the smallest local minimum, deflate the the error function for this minimum and iterate the extensive search by conventional optimization.

An intuitive justification of these options can be based on the phenomenon known as the weight-space symmetries of multi-layer networks [6, page 133]. Consider a network having K hidden layers, with $L_i, i = 1, 2, \dots, K$ hidden units for each layer, and full connectivity between layers. The number of the weight-space symmetries of this network is $S = \prod_{i=1}^K L_i! 2^{L_i}$ which states the number of all different local minima \mathbf{w}^* that give rise to the same mapping function $y_j = y(\mathbf{x}_j; \mathbf{w}^*)$ and the same value of $E(\mathbf{w}^*)$ respectively. Consequently by running our method for a certain solution \mathbf{w}^* we transform the training data and eliminate all symmetric minima at the same time. By this means we release the conventional minimization from these S minima.

More thorough theoretical and experimental study of the combination of our method and the conventional minimization is the object of our current research and is beyond the scope of this work.

5 Summary and Conclusion

We proposed a method for recursive training of neural networks for classification. It searches for the discriminant functions corresponding to several small local minima of the error function. In Section 3 we showed that for the linear classification functions corresponding to the error function with a sharp oscillation our method is better than method [2]. In Section 4 we demonstrated that for non-linear classification functions our method has the potential to escape from the local minima of the error function and to direct the local optimizer to new solutions.

Our method extends Friedman's [8] procedure for recursive optimization, called "structure removal". We summarize the main features of this extension:

- Friedman’s procedure is oriented to unsupervised training while our method is a tool for supervised training of classification networks.

- Friedman’s ”structure removal” is a linear method while our procedure is oriented towards non-linear classification functions. The price that we pay for the non-linear extension is an increased computational complexity of our procedure. We train an auto-associative network, while in the linear case [1], [2], [5] [8], [10] a simple rotation of the data is carried out.

- As is the case with Friedman’s procedure [8], [10], we may miss some small local minima including the global one. Therefore we do not think of our procedure as a global optimization method, but rather as a tool for escaping from the local minima already found and directing the local optimizer to new solutions.

- Taking into account the computational complexity of our method, we proposed in Section 3.4 a combination of our method and method [2], and in Section 4 a combination of our method and the conventional minimization with random initialization.

A Appendix: A Procedure for Recursive Minimization of $E(\mathbf{w})$

In Section 2 we proposed a method for recursive minimization of an error function $E(\mathbf{w})$ of a classification network. The main idea is to modify $E(\mathbf{w})$ by means of successive transformations of the training data (RCS) during the minimization. Here we present a computational procedure for this method. In order to formalize this procedure we introduce the following nomenclature:

- X_t denotes the original training data.
- \mathbf{w}_0^* and \mathbf{w}_I^* , for $I = 1, 2, \dots, N_{RCS}$ are some of the local minima of $E(\mathbf{w})$.
- N_{RCS} is the number of the successive transformations of the training data by RCS.

We use $N_{RCS} = 1 \div 6$.

- \tilde{X}_t denotes the current training data.
- $\tilde{\mathbf{w}}$ is a local minimum of the current $E(\mathbf{w})$ ($E(\mathbf{w})$ computed for \tilde{X}_t).
- \tilde{X}_t' denotes a transformation of \tilde{X}_t by RCS.
- $\tilde{\mathbf{w}}'$ is a local minimum of $E(\mathbf{w})$ computed for \tilde{X}_t' .
- \mathbf{u}_0 denotes a vector which comprises the initial weights of the AA network. We choose

\mathbf{u}_0 to be the minimum of $J(\mathbf{u})$ (1) for $\nu = 0$.

- \mathbf{w}_0 is the vector which comprises the initial weights of the classification network. In our comparative study in Section 4 we set \mathbf{w}_0 at random. In the exploratory examination of the neighborhood of \mathbf{w}_0 we set \mathbf{w}_0 to be the local minimum of a previous conventional minimization of $E(\mathbf{w})$.

- ε_ν and $\varepsilon_{\Delta\sigma^2}$ are steps for increasing the control parameters ν and $\Delta\sigma^2$. From experience, we set $\varepsilon_\nu = 0.1$ and $\varepsilon_{\Delta\sigma^2} = 0.25$ for data $\mathbf{x}_j \in R^n$ having dimension $n = 2 \div 20$.

- *Mode = Local or Global* is the mode of setting $\tilde{\mathbf{w}}$. "Local" sets $\tilde{\mathbf{w}}$ in the neighborhood of \mathbf{w}_0 . "Global" sets $\tilde{\mathbf{w}}$ by an extensive minimization of $E(\mathbf{w})$ using a conventional method.

We propose the following computational procedure:

Step 1. Initialization of the RCS: We set the current training data $\tilde{X}_t = X_t$, and the control parameters $\Delta\sigma^2 = 1$ and $\nu = \min\{0.1, 1/n\}$.

If *Mode = Local* we run a local minimizer for $E(\mathbf{w})$ computed for \tilde{X}_t ($\tilde{X}_t = X_t$). We start the minimization from \mathbf{w}_0 . We set $\tilde{\mathbf{w}}$ to be the local minimum found.

If *Mode = Gobal* we minimize $E(\mathbf{w})$ many times starting from different initial weight initialization. We set $\tilde{\mathbf{w}}$ to be the smallest local minimum found in the trials performed.

We save $\tilde{\mathbf{w}}$ into \mathbf{w}_0^* ($\mathbf{w}_0^* = \tilde{\mathbf{w}}$).

for $I = 1 : N_{RCS}$ {Successive runs of the RCS}

Set control variables *Change_of_nu = Yes, Restore = No*.

Step 2. Computing the targets q_j : We propagate $\tilde{\mathbf{x}}_j \in \tilde{X}_t$, $j = 1, 2, \dots, N_t$ (current training vectors) through the classification network having weights $\tilde{\mathbf{w}}$ and obtain the corresponding outputs $y_j = y(\tilde{\mathbf{w}}; \tilde{\mathbf{x}}_j)$. Then we compute q_j using (4),(5) and (6) for the current $\Delta\sigma^2$.

Step 3. Training the AA network using current ν and q_j : Starting from \mathbf{u}_0 we minimize $J(\mathbf{u})$ (1) and obtain the minimum \mathbf{u}^* . Then we update the starting point $\mathbf{u}_0 = \mathbf{u}^*$.

Step 4. Transformation the training data: We propagate $\tilde{\mathbf{x}}_j \in \tilde{X}_t$ (current training vectors) through the AA network having weights \mathbf{u}^* , and obtain the transformed training data $\tilde{X}_t' = \{\tilde{\mathbf{x}}_j' : \tilde{\mathbf{x}}_j' = \mathbf{r}(\tilde{\mathbf{x}}_j; \mathbf{u}^*)\}$.

Step 5. Training the classification network: Starting from $\tilde{\mathbf{w}}$ we run the local minimizer for the $E(\mathbf{w})$ computed for \tilde{X}_t' and obtain the local minimum $\tilde{\mathbf{w}}'$. Then starting from $\tilde{\mathbf{w}}'$ we re-run the local minimizer for the original $E(\mathbf{w})$ ($E(\mathbf{w})$ computed for X_t) and save the solution into \mathbf{w}_I^* .

Step 6. Updating $\Delta\sigma^2$, ν , \tilde{X}_t and $\tilde{\mathbf{w}}$: We compare the solutions \mathbf{w}_{I-1}^* and \mathbf{w}_I^* saved in Step 5 and for the first trial in Steps 1 and 5. We examine the error rates $E(\mathbf{w}_{I-1}^*)$ and $E(\mathbf{w}_I^*)$, and the class-conditional densities of the network outputs $y_j^{I-1} = y(\mathbf{w}_{I-1}^*; \mathbf{x}_j)$ and $y_j^I = y(\mathbf{w}_I^*; \mathbf{x}_j)$, $\mathbf{x}_j \in X_t$.

If the error rates and the class-conditional densities are approximately equal for \mathbf{w}_{I-1}^* and \mathbf{w}_I^* , we make RCS stronger in order to escape from \mathbf{w}_{I-1}^* . We increase ν or $\Delta\sigma^2$.

if *Change_of_* $\nu = Yes$

$\nu \leftarrow \nu + \varepsilon_\nu$

If $\nu < 1$ then we repeat Steps 3-6, otherwise set *Restore* = *Yes* and continue.

elseif *Change_of_* $\nu = No$

$\Delta\sigma^2 \leftarrow \Delta\sigma^2 + \varepsilon_{\Delta\sigma^2}$

If $\Delta\sigma^2 \leq 1$ then we repeat Steps 2-6, otherwise set *Restore* = *Yes* and continue.

end

end If the error rates

If *Restore* = *Yes* or the error rates and the class-conditional densities are different for \mathbf{w}_{I-1}^* and \mathbf{w}_I^* (different local minima \mathbf{w}_{I-1}^* and \mathbf{w}_I^* have been obtained) then we proceed as follows.

if *Restore* = *No* and *Change_of_* $\nu = Yes$

$\Delta\sigma^2 = 0$, *Change_of_* $\nu = No$ and repeat Steps 2-6

else {We restore ν , $\Delta\sigma^2$ and update $\tilde{X}_t, \tilde{\mathbf{w}}$ }

$\Delta\sigma^2 = 1$, $\nu = \min\{0.1, 1/n\}$ and $\tilde{X}_t = \tilde{X}_t'$

if *Mode* = *Local*

$\tilde{\mathbf{w}} = \tilde{\mathbf{w}}'$

elseif *Mode* = *Global*

We minimize $E(\mathbf{w})$ for \tilde{X}_t many times starting from different weight initialization. We set $\tilde{\mathbf{w}}$ to be the smallest local minimum found.

end

end

end If *Restore* = *Yes*

end for $I = 1 : N_{RCS}$ {Successive runs of the RCS}

Output: Vectors $\mathbf{w}_0^*, \mathbf{w}_1^*, \mathbf{w}_2^*, \dots, \mathbf{w}_{N_{RCS}}^*$ saved in Steps 1 and 5.

- For *Mode* = *Local* these vectors comprise the local minima in the neighborhood of the starting point \mathbf{w}_0 .
- For *Mode* = *Global* they comprise the smallest local minima from many trials with a local minimizer.

References

- [1] M.E.Aladjem, "Linear discriminant analysis for two-classes via removal of classification structure", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol.19, pp.187-192, 1997.
- [2] M.E.Aladjem, "Non-parametric discriminant analysis via recursive optimization of Patrick-Fisher distance", *IEEE Trans. on Syst., Man, Cybern.*, vol.28B, pp.292-299, 1998.

- [3] M.E.Aladjem, "Linear discriminant analysis for two classes via recursive neural network reduction of the class separation", In A.Amin, D.Dori, P.Pudil and H.Freeman (eds.), *Lecture Notes in Computer Science: Advances in Pattern Recognition*, vol.1451, pp.775-784, 1998.
- [4] M.E.Aladjem, "Training of an ML neural network for classification via recursive reduction of the class separation", *Proc. 14th Int. Conf. on Pattern Recognition*, Brisbane, vol.I, pp.450-452, 1998.
- [5] M.E.Aladjem, "Non-parametric linear discriminant analysis by recursive optimization with random initialization", *Lecture Notes in Computer Science: Advances in Intelligent Data Analysis*, vol. 1642, pp.223-234, 1999.
- [6] C.M.Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press Inc., New York, 1995.
- [7] P.A.Devijver, J.Kittler, *Pattern Recognition: A Statistical Approach*, Prentice-Hall International Inc., London, 1982.
- [8] J.H.Friedman, "Exploratory projection pursuit", *Journal of the American Statistical Association*, vol.82, pp.249-266, 1987.
- [9] A.Papoulis, *Probability, Random Variables, and Stochastic Processes*, Third Edition, McGraw-Hill In., New York, 1991.
- [10] J.Sun, "Some practical aspects of exploratory projected pursuit", *SIAM J. Sci. Comput.*, vol.14, pp.68-80, 1993.