

Supervised Training of a Neural Network for Classification via Successive Modification of the Training Data - An Experimental Study*

Mayer Aladjem

Department of Electrical and Computer Engineering
Ben-Gurion University of the Negev, P.O.B. 653, 84105 Beer-Sheva,
Israel, *e-mail*: aladjem@bgu.ac.il

Abstract. A method for training of an ML network for classification has been proposed by us in [3,4]. It searches for the non-linear discriminant functions corresponding to several small local minima of the objective function. This paper presents a comparative study of our method and conventional training with random initialization of the weights. Experiments with a synthetic data set and the data set of an OCR problem are discussed. The results obtained confirm the efficacy of our method which finds solutions with lower misclassification errors than does conventional training.

Keywords: Neural networks for classification, auto-associative network, projection pursuit, structure removal, discriminant analysis, statistical pattern recognition.

1 Introduction

The training of a *multi-layer* (ML) neural network can be formulated in terms of the minimization of an error function $E^{ML}(\mathbf{w})$, which depends on the vector \mathbf{w} comprising adaptive parameters in the network [5]. The error $E^{ML}(\mathbf{w})$ of the network with nonlinear activation functions in the hidden units is a highly nonlinear function. Consequently the minimization of $E^{ML}(\mathbf{w})$ has to be carried out by an iterative optimization algorithm. The primary goal is to find the global minimum of $E^{ML}(\mathbf{w})$. By a naive use of a training algorithm (a local minimizer of $E^{ML}(\mathbf{w})$), the computed value for the observed minimum of $E^{ML}(\mathbf{w})$ can be merely a local minimum. The solution depends strongly on the starting point of the local optimizer.

This paper presents a comparative study of a method for searching for several small local minima of $E^{ML}(\mathbf{w})$ proposed by us in [3,4], and the conventional training with random initialization of the weights [5]. Section 2 describes our method [3,4] and Section 3 contains the results and analyses of the comparison based on a synthetic data set and an OCR application.

2 Training Neural Networks for Classification via Successive Modification of the Training Data

Suppose we are given training data $(\mathbf{x}_1, \mathbf{I}_1), (\mathbf{x}_2, \mathbf{I}_2), \dots, (\mathbf{x}_{N_t}, \mathbf{I}_{N_t})$ 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

*This work has been partially supported by the Paul Ivanier Center for Robotics and Production Management, Ben Gurion University of the Negev, Israel

.....
 $(\mathbf{x}_j \in \mathbb{R}^n, n \geq 2)$ and their associated class-indicator vectors $\mathbf{l}_j, j=1,2,\dots,N_i$. We discuss a two class problem and we require that \mathbf{l}_j is a two-dimensional vector $\mathbf{l}_j=(l_{1j}, l_{2j})^T$ which shows that \mathbf{x}_j belongs to one of the classes ω_1 or ω_2 . The components l_{1j}, l_{2j} are defined to be one or zero according to the class-membership of \mathbf{x}_j , i.e. $l_{1j}=1, l_{2j}=0$ for $\mathbf{x}_j \in \omega_1$ and $l_{1j}=0, l_{2j}=1$ for $\mathbf{x}_j \in \omega_2$. The class-indicator vectors \mathbf{l}_j imply decomposition of the set X_i into two subsets corresponding to the unique classes. We denote by N_{ti} the number of the training observations in class ω_i . Our method for recursive training requires a normalization of the data, called *sphering* [6] (or *whitening* [5]). For the sphered training data X_i 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*.

2.1 Training an SL Network for Classification

Here, we present our method for training a *single-layer* (SL) neural network for classification [2,3]. In the next sections we extend it to training an ML network.

We discuss SL network with linear activation function of the output. It carries out a linear mapping $y=\mathbf{w}^T\mathbf{x}, \mathbf{x} \in \mathbb{R}^n, y \in \mathbb{R}^1, n \geq 2$, with \mathbf{x} an arbitrary n -dimensional observation, and \mathbf{w} a vector containing the weights of the network. We require \mathbf{w} to have unit length, and $y=\mathbf{w}^T\mathbf{x}$ can be interpreted geometrically [5, pp.77-79] as the projection of the observation \mathbf{x} onto vector \mathbf{w} in the \mathbf{x} -space.

We train the network by minimizing an error function $E^{SL}(\mathbf{w})$, which is the negative *Patrick-Fisher distance* [2]. $E^{SL}(\mathbf{w})$ measures the overlap of the class-conditional densities along vector \mathbf{w} . $E^{SL}(\mathbf{w})$ is a nonlinear function with respect to \mathbf{w} . In order to search for several small local minima of $E^{SL}(\mathbf{w})$ we proposed a method for recursive minimization of $E^{SL}(\mathbf{w})$ [2]. We obtain a vector of weights \mathbf{w}^* related to a local minimum of $E^{SL}(\mathbf{w}^*)$ and then we transform the data along \mathbf{w}^* into data with greater overlap of the class-conditional densities (deflated minimum of $E^{SL}(\mathbf{w})$ at the solution \mathbf{w}^*), and iterate to obtain a new vector of weights. The main point of the method is the procedure for deflating the local minimum of $E^{SL}(\mathbf{w})$ called "*reduction of the class separation*".

2.1.1 Procedure for "reduction of the class separation"

In order to deflate $E^{SL}(\mathbf{w})$ at \mathbf{w}^* (to increase class overlap along \mathbf{w}^*), we transform class-conditional densities along \mathbf{w}^* to normal densities. For this purpose, we rotate the data applying the linear transformation

$$\mathbf{r}=\mathbf{U}\mathbf{x} \tag{1}$$

with \mathbf{U} an orthonormal ($n \times n$) matrix. We denote the new coordinates as r_1, r_2, \dots, r_n ($\mathbf{r}=(r_1, r_2, \dots, r_n)^T$). We require that the first row of \mathbf{U} is \mathbf{w}^* , which results in a rotation such that the new first coordinate of an observation \mathbf{x} is the output of the SL network having weight vector \mathbf{w}^* ($r_1=y=(\mathbf{w}^*)^T\mathbf{x}$). Assume that $p(y|\omega_i), i=1,2$ are the

class-conditional densities of $y=(\mathbf{w}^*)^T \mathbf{x}$ and, $m_{y|\omega_i}$, $\sigma_{y|\omega_i}^2$ their means and variances.

We transform $p(y|\omega_i)$ to normal densities and leave the coordinates r_2, r_3, \dots, r_n unchanged. Let \mathbf{q} be a vector function with components q_1, q_2, \dots, q_n that carries out this transformation: $r_1'=q_1(y)$ with r_1' having normal class-conditional distributions and $r_i'=q_i(r_i)$, $i=2,3,\dots,n$ each given by the identity transformations. The function q_1 is obtained by the percentile transformation method [2,6]:

- for observations \mathbf{x} from class ω_1 :

$$q_1(y) = [\Phi^{-1}(F(y|\omega_1))](\sigma_{y|\omega_1}^2 \pm \Delta\sigma^2)^{1/2} + (m_{y|\omega_1} - \Delta m_1); \quad (2)$$

- for observations \mathbf{x} from class ω_2 :

$$q_1(y) = [\Phi^{-1}(F(y|\omega_2))](\sigma_{y|\omega_2}^2 \pm \Delta\sigma^2)^{1/2} + (m_{y|\omega_2} - \Delta m_2). \quad (3)$$

Here, $\Delta\sigma^2$, Δm_1 , Δm_2 are user-supplied parameters, $F(y|\omega_i)$ is the class-conditional (cumulative) distribution function of $y=(\mathbf{w}^*)^T \mathbf{x}$ for $i=1,2$ and Φ^{-1} is the inverse of the standard normal distribution function Φ . Finally,

$$\mathbf{x}' = \mathbf{U}^T \mathbf{q}(\mathbf{U}\mathbf{x}) \quad (4)$$

transforms the class-conditional densities of the output of the SL network to give normal densities $p(r_1'|\omega_i)=N(m_{y|\omega_i} - \Delta m_i, \sigma_{y|\omega_i}^2 \pm \Delta\sigma^2)$ leaving all directions orthogonal to \mathbf{w}^* unchanged.

In [2] we proposed a procedure for defining the values of the control parameters $\Delta\sigma^2$, Δm_1 , Δm_2 and the sign (+ or -) of the change $\pm\Delta\sigma^2$ in order to direct the local optimizer to a new minimum of $E^{SL}(\mathbf{w})$, and to keep the class-conditional densities of \mathbf{x}' (4) as close to the densities of the original data \mathbf{x} as is possible.

2.1.2 Neural Network Reduction of the Class Separation (NN_RCS)

In [3] we proposed a neural network implementation of the procedure for “reduction of the class separation”, called NN_RCS. We use an auto-associative network having non-linear activation functions in the hidden units (Fig.1). The targets used to train the network are the input vectors themselves, so that the network is attempting to map each input vector onto itself. We train the network by minimizing an error function of the form

$$E(\mathbf{u}) = (1-\nu)E^{AA}(\mathbf{u}) + \nu\Omega^{SL}(\mathbf{u}), \quad 0 \leq \nu \leq 1, \quad (5)$$

with

$$E^{AA}(\mathbf{u}) = \frac{1}{N_t} \sum_{j=1}^{N_t} [\mathbf{r}(\mathbf{x}_j; \mathbf{u}) - \mathbf{x}_j]^T [\mathbf{r}(\mathbf{x}_j; \mathbf{u}) - \mathbf{x}_j], \quad (6)$$

$$\Omega^{SL}(\mathbf{u}) = \frac{1}{N_t} \sum_{j=1}^{N_t} [(\mathbf{w}^*)^T \mathbf{r}(\mathbf{x}_j; \mathbf{u}) - q_1((\mathbf{w}^*)^T \mathbf{x}_j)]^2. \quad (7)$$

Here, $E^{AA}(\mathbf{u})$ is the standard mean-square (MS) error of the auto-associative network, $\Omega^{SL}(\mathbf{u})$ is the penalty function and v is the parameter controlling the extent to which the penalty term $\Omega^{SL}(\mathbf{u})$ influences the form of the solution. In (6) and (7) $\mathbf{r}(\mathbf{x}_j; \mathbf{u})$ represents the output vector $\mathbf{x}'_j = \mathbf{r}(\mathbf{x}_j; \mathbf{u})$ of the auto-associative network (Fig.1) as a function of the input training vectors \mathbf{x}_j , $j=1, 2, \dots, N_t$ and vector \mathbf{u} comprising the adjustable weights of the network; $q_1((\mathbf{w}^*)^T \mathbf{x}'_j)$ is the function which transforms the class-conditional densities of $y = (\mathbf{w}^*)^T \mathbf{x}'_j$ to the normal densities (see (2) and (3)).

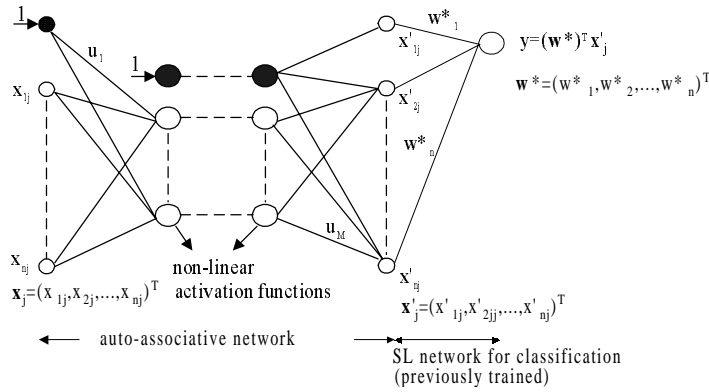


Fig.1. Auto-associative network. It is trained to map input vectors into themselves in such a way that the data along \mathbf{w}^* have normal class-conditional densities.

The auto-associative network is trained by minimizing the total error function $E(\mathbf{u})$ (5) with respect to \mathbf{u} . A function $\mathbf{r}(\mathbf{x}_j; \mathbf{u})$ which provides a good fit to the training data \mathbf{x}_j , $j=1, 2, \dots, N_t$ will give a small value for $E^{AA}(\mathbf{u})$ (6), while one which produces data with the normal densities along \mathbf{w}^* will give a small value for $\Omega^{SL}(\mathbf{u})$ (7). Minimizing $E(\mathbf{u})$ (5) we obtain the network mapping $\mathbf{r}(\mathbf{x}_j; \mathbf{u})$ which is a compromise between fitting the training data \mathbf{x}_j and reducing the class separation along \mathbf{w}^* (deflating $E^{SL}(\mathbf{w})$ at \mathbf{w}^* for suitable $\Delta\sigma^2$, Δm_1 and Δm_2).

NN_RCS overcomes the use of the orthonormal matrix \mathbf{U} in the transformation (1). This makes it possible to apply NN_RCS for the non-linear classification functions $y(\mathbf{x})$. We have just to obtain function $q_1(y)$ which transforms the class-conditional densities of $y(\mathbf{x})$ to normal densities and to use the non-linear mapping $y(\mathbf{x}'_j)$ instead of the linear one $(\mathbf{w}^*)^T \mathbf{x}'_j$, for $\mathbf{x}'_j = \mathbf{r}(\mathbf{x}_j; \mathbf{u})$ in the penalty function (7). In the next section we apply NN_RCS for reduction of the class separation of the non-linear classification functions obtained by the ML network.

2.2 Recursive Training ML Network for Classification

We discuss an ML network for classification with nonlinear activation functions in the hidden units. We denote, $y(\mathbf{x}; \mathbf{w})$ to be the mapping function of the network with vector \mathbf{w} comprising the adjustable weights of the network. We train the

network minimizing the MS error function $E^{ML}(\mathbf{w})$ which allows the network to represent classification functions

$$E^{ML}(\mathbf{w}) = \frac{1}{N_t} \sum_{j=1}^{N_t} [y(\mathbf{x}_j; \mathbf{w}) - (l_{1j} - l_{2j})]^2. \quad (8)$$

In (8) the targets are $(l_{1j} - l_{2j}) = 1$ for training vectors $\mathbf{x}_j \in \omega_1$ and $(l_{1j} - l_{2j}) = -1$ for $\mathbf{x}_j \in \omega_2$.

Here we present a procedure for recursive training ML network. It involves a modification of the training data during the training process. We denote by \tilde{X}_t the current training set. We start with $\tilde{X}_t = X_t = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{N_t}\}$, where X_t is the original training set.

Step 1 Initial training classification network: We train the classification network using the original training vectors $\mathbf{x}_j \in X_t$. After convergence of the training algorithm to a local minimum $E^{ML}(\mathbf{w}^*)$ we save the solution \mathbf{w}^* .

Step 2 Initializing NN_RCS: We set $v = 1/n$, $\Delta\sigma^2 = 0$ and $\Delta m_i = 0$, for $i = 1, 2$. This setting implies minimal changes of the data during NN_RCS [2,3].

Step 3 NN_RCS: We propagate $\tilde{\mathbf{x}}_j \in \tilde{X}_t$ (current training data) through the previously trained classification network and obtain the corresponding outputs $y_j = y(\tilde{\mathbf{x}}_j; \mathbf{w}^*)$, $j = 1, 2, \dots, N_t$. Then we estimate the class-conditional means, variances and (cumulative) distribution functions of the network output using the sample y_j , $j = 1, 2, \dots, N_t$. We denote the estimates by \hat{m}_i , $\hat{\sigma}_i^2$ and $\hat{F}(y|\omega_i)$ for ω_i , $i = 1, 2$. Then we compute N_t numbers

$$q_j = \begin{cases} [\Phi^{-1}(\hat{F}(y_j|\omega_1))](\hat{\sigma}_1^2 \pm \nabla\sigma^2)^{1/2} + (\hat{m}_1 - \nabla m_1) & \text{for } l_{1j} = 1 \\ [\Phi^{-1}(\hat{F}(y_j|\omega_2))](\hat{\sigma}_2^2 \pm \nabla\sigma^2)^{1/2} + (\hat{m}_2 - \nabla m_2) & \text{for } l_{2j} = 1 \end{cases} \quad (9)$$

for y_j , $j = 1, 2, \dots, N_t$. Note that (9) is a sample version of (2) and (3) and consequently q_j has distribution $N(\hat{m}_1 - \Delta m_1, \hat{\sigma}_1^2 \pm \Delta\sigma^2)$ for $l_{1j} = 1$ and $N(\hat{m}_2 - \Delta m_2, \hat{\sigma}_2^2 \pm \Delta\sigma^2)$ for $l_{2j} = 1$. In (9) we choose the sign (+ or -) of the change ($\pm\Delta\sigma^2$) in order to approach $\hat{\sigma}_i^2 \pm \Delta\sigma^2$ to 1 (to increase the overlap of the class-conditional densities of q_j).

3.1 Training the auto-associative network: We train the auto-associative network (Fig.2) by an iterative minimization the error function

$$\tilde{E}(\mathbf{u}) = (1-v)E^{AA}(\mathbf{u}) + v\Omega^{ML}(\mathbf{u}). \quad (10)$$

Here, $E^{AA}(\mathbf{u})$ is the MS error of the auto-associative network and $\Omega^{ML}(\mathbf{u})$ is the penalty function which measures the departure of the responses of the classification network from the numbers q_j , $j = 1, 2, \dots, N_t$ (9). We start the iterative minimization from some random initialization of \mathbf{u} . In each iteration we evaluate $E^{AA}(\mathbf{u})$ and $\Omega^{ML}(\mathbf{u})$ for the current \mathbf{u} , as follows: We propagate the training vectors $\tilde{\mathbf{x}}_j \in \tilde{X}_t$, $j = 1, 2, \dots, N_t$ through the currently trained auto-associative network (current \mathbf{u}) and obtain

their responses $\mathbf{x}'_j = \mathbf{r}(\tilde{\mathbf{x}}_j; \mathbf{u})$ with $\mathbf{r}(\tilde{\mathbf{x}}_j; \mathbf{u})$ the mapping function of the auto-associative network (see Fig.2). Then we compute

$$E^{AA} = \frac{1}{N_t} \sum_{j=1}^{N_t} [\mathbf{x}'_j - \tilde{\mathbf{x}}_j]^T [\mathbf{x}'_j - \tilde{\mathbf{x}}_j].$$

In order to evaluate $\Omega^{ML}(\mathbf{u})$ we propagate \mathbf{x}'_j , $j=1, 2, \dots, N_t$ (the modified training data) through the previously trained classification network and obtain their responses (see Fig.2) $y'_j = y(\mathbf{x}'_j; \mathbf{w}^*)$. Then we compute

$$\Omega^{ML} = \frac{1}{N_t} \sum_{j=1}^{N_t} (y'_j - q_j)^2.$$

We denote by \mathbf{u}^* the vector comprising the weights of the trained auto-associative network. We save \mathbf{u}^* .

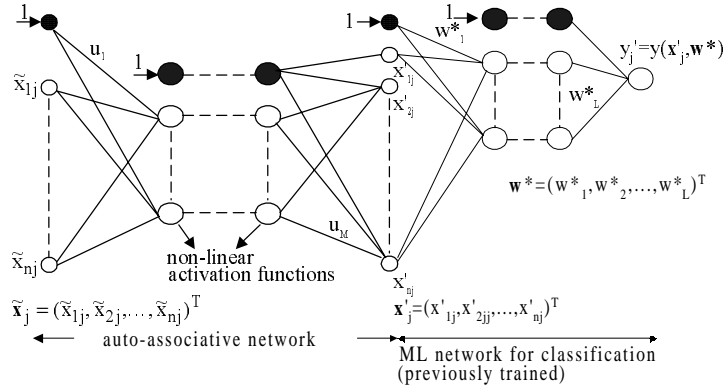


Fig.2. Auto-associative network. It is trained to map input vectors into themselves in such a way that the non-linear discriminant function $y(\mathbf{x}', \mathbf{w}^*)$ has normal class-conditional densities.

Step 4 Training the classification network using the modified training data: $X_t^{(mod)} = \{ \mathbf{x}_j^{(mod)} : \mathbf{x}_j^{(mod)} = \mathbf{r}(\tilde{\mathbf{x}}_j; \mathbf{u}^*), j=1, 2, \dots, N_t \}$. We denote the solution $\mathbf{w}^{*(mod)}$.

Step 5 Retraining (adjusting) the classification network by means of the original training set X_t : Starting the iterative optimization procedure from $\mathbf{w}^{*(mod)}$ (saved in Step 4) we minimize $E^{ML}(\mathbf{w})$ (8) by using the original training vectors $\mathbf{x}_j \in X_t$, $j=1, 2, \dots, N_t$. After convergence of the training algorithm to a local minimum $E^{ML}(\mathbf{w}^*)$ we save the solution \mathbf{w}^* .

Step 6 Updating the control parameters $v, \Delta\sigma^2, \Delta m_1, \Delta m_2$ and the training set \tilde{X}_t : We examine the misclassification error of the classification network ($y=y(\mathbf{x}; \mathbf{w}^*)$) for the last two solutions \mathbf{w}^* saved in Step 5 and for the first trial in Step 1.

(a) If the error rates are approximately equal for the last two solutions \mathbf{w}^* , we increase $\Delta\sigma^2, \Delta m_1, \Delta m_2$ (increase the overlap of the class-conditional densities of q_j)

and/or increase ν (increase the influence of the penalty term $\Omega^{SL}(\mathbf{u})$ to the form of the solution).

(b) If the error rates are different for the last two solutions \mathbf{w}^* (different local minima of $E^{ML}(\mathbf{w})$ (1) have been obtained) we update the training set ($\tilde{X}_t = X_t^{(mod)}$) and restore the initial values of the control parameters $\nu=1/n$, $\Delta\sigma^2=0$, $\Delta m_1=0$, $\Delta m_2=0$.

Then we *repeat Steps 3-6* using a new random initialization in Steps 3 and 4. We stop the iterations if several vectors \mathbf{w}^* corresponding to classification networks with different error rates are obtained. We regard the vectors \mathbf{w}^* corresponding to the smallest misclassification errors as "*interesting*" solutions.

3 Experiments

Here we compare the discrimination qualities of the classification networks trained by our recursive procedure (Section 2.2) and those trained by a *conventional training* with a random initialization of the weights [5]. We ran experiments with a synthetic data and an OCR problem starting from different initial weights of the classification network in Step 1 of our procedure. The weights were drawn from $N(0, \sigma^2)$ with σ^2 defined by the method of Bishop [5, pp.260-263].

We used two layer networks with sigmoid activation functions, and 3 and 6 hidden units for the classification and the auto-associative networks, respectively. We train these networks minimizing error functions $E^{ML}(\mathbf{w})$ (8) and $\tilde{E}(\mathbf{u})$ (10) by a sequential quadratic programming method (routine E04UCF in the NAG Mathematical Library). We set the number of the batch (major) iterations of the optimization routine E04UCF to 50 for the classification networks (Steps 1, 4 and 5) and 100 for the auto-associative network (Step 3.1). This setting was proved to be appropriate by a preliminary test.

We evaluated the classification quality of the networks by the resulting *test error rates*. We calculated them by the percentage of the wrong allocated test (extra, validation) observations $\mathbf{x}_j^v \notin X_t$, $j=1, 2, \dots, N_v$ by a plug-in Bayes allocation rule [7] using a sample estimate of the a priori probabilities of the classes ($\hat{P}(\omega_i) = N_{it}/N_t$) and the Parzen estimators [7] of the class-conditional densities of the output of the classification network based on the training data. We compute *the test error rate* as the percent of the wrong allocated \mathbf{x}_j^v .

We compared the classification quality of the networks in Step 1 and Step 5 for the synthetic and the OCR data. In Step 1 we carried out the conventional training with random initialization of the weights while in Step 5 we employed our recursive training. For each data set we carried out 100 runs of our procedure, starting from different initial weights of the classification network in Step 1.

In order not to favor our procedure we initialized the classification network in Step 4 with the weights used for initialization in Step 1, and we started the training the auto-associative network in Step 3.1 from a fixed point for all trials.

We ran the experiments in a batch mode which does not search for the suitable values of the parameters $\Delta\sigma^2$, Δm_1 , Δm_2 and ν in each run of the reduction of the

class separation. We set $\hat{m}_i - \Delta m_i = 0$ and $\hat{\sigma}_i^2 \pm \Delta \sigma^2 = 1$ for $i=1,2$ in (9), and $v=0.5$ in (10) for all runs. This setting implies that the class-conditional densities of q_j (9) are $N(0,1)$ which results in a modified training data $X_t^{(\text{mod})}$ with an approximately full overlap of the classes for the previously defined discriminant functions. This certainly eliminates the local minimum of $E^{\text{ML}}(\mathbf{w})$ at the previous solutions but it causes large destructuring of $X_t^{(\text{mod})}$ which is highly unfavorable to our procedure.

We carried out three successive runs of Steps 3-6. In order not to favor our procedure by expanding the number of training iterations we re-ran Step 1 with an extended number of the batch (major) iterations of the optimization routine E04UCF. We set it to $50 \times 2 \times N_{(\text{steps}3-6)}$, with $N_{(\text{steps}3-6)}$ the number of repetitions of Steps 3-6. In the comparison we used the smallest test error rate obtained in Step 1.

3.1 Batch Run of the Synthetic Data

The synthetic observations were drawn from six dimensional distributions $p(\mathbf{x}|\omega_i) = p(x_1, x_2|\omega_i)p(x_3, x_4|\omega_i)p(x_5|\omega_i)p(x_6|\omega_i)$ for $i=1,2$. Here the densities $p(x_1, x_2|\omega_i)$ and $p(x_3, x_4|\omega_i)$ were mixtures of the normal distributions (see [3]) and $p(x_5|\omega_i) = p(x_6|\omega_i) = N(0,1)$. The classes were totally overlapped in the (x_5, x_6) -plane, partially overlapped in the (x_1, x_2) -plane and totally separated in the (x_3, x_4) -plane. We set $N_{i1} = N_{i2} = 150$; $N_{v1} = N_{v2} = 500$.

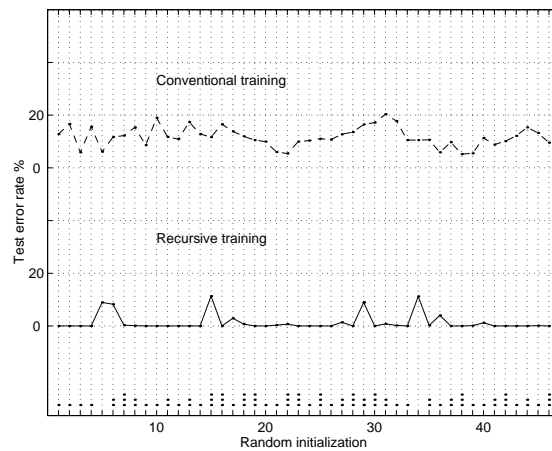


Fig.3. Synthetic data. Random initializations in which conventional training failed with test error rate larger then 2.5%.

We studied the situations (initial weights) in which the conventional training failed with test error rates larger then 2.5% in Step 1 (dashed path in Fig.3). The solid path in Fig.3 presents the results obtained by our procedure. The dots in the bottom of Fig.3 indicate the sequential number of the reduction of the class separation with the smallest test error rate (‘ - first, ‘‘ - second and ‘’’ - third reduction). The dots which are missing indicate the cases (random initializations 4

and 34) for which the conventional training was better than our procedure. We missed the perfect solution (test error rate 0%) for the random initializations 5,6, 15, 29, 34 and 36, and succeeded for the rest of the cases (40 initializations). Consequently we do not view our procedure as a global optimization method, but as a tool which searches for several solutions with small local minima and ensures that the local minimum already found will not be chosen again at a later stage.

3.2 Batch Run of the OCR Data

The real data set concerning an OCR problem contains 600 cases: 150 upper-case handwritten letters “M” and 450 lower-case handwritten letters “m”. Twenty-three numerical results from a feature extraction procedure (developed in a company in Israel) were recorded for each case. We used 75 M-cases and 225 m-cases for training and the rest of the data for validation. As previously, we ran 100 replications of our procedure starting from different initial weights of the network in Step 1. We summarize the overall shape of the test error rates over the 100 replications by the boxplots shown in Fig.4. The boxplot in the left presents the error rates of the network trained by the conventional training, the central boxplot illustrates the errors of the network trained by our recursive procedure, and the boxplot in the right presents the paired difference of these errors. In Fig.4 the boxes show the error rates between “quartiles”; the lines represent the medians of the error rates; “whiskers” go out to the extremes of the error rates, and an extreme error rate is shown by “+” (for more details see [8]). We observe that the errors of our procedure tend to be smaller than the errors of the conventional training. Also, our procedure gained the best result indicated by “+”. We calculated the averaged difference of the test error rates of the networks trained by conventional training and our training procedure, which was 3%. We evaluated the significance of this difference by the paired t-test and obtained the 99 percent confidence interval 3 ± 1.4 % which confirms a significant decrease of the test error rate of the network trained by our procedure.

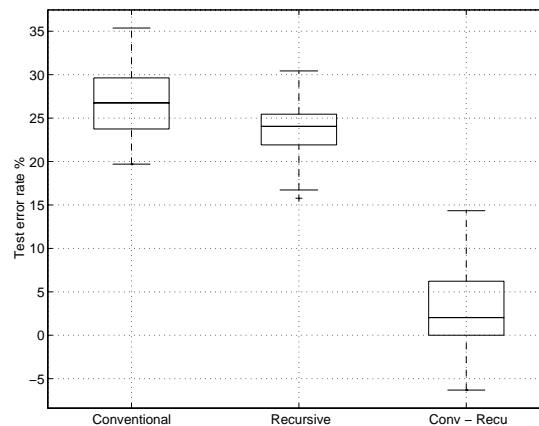


Fig.4. OCR data. Boxplots of the test error rates of the networks trained by the conventional and the recursive training.

4 Summary and Conclusion

In this paper we carried out an experimental study of our method for training an ML network for classification [3,4]. It was more successful than the conventional training with random initialization of the weights for a synthetic data set and an OCR application considered in Section 3. Our method extends Friedman's [6] procedure for recursive optimization, called "*structure removal*". 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 the high computational complexity of the NN_RCS (Section 2.1.2). In NN_RCS we use an intensive non-linear optimization technique for training the auto-associative network, while in the linear case [1,2,6] a simple rotation of the data is carried out.

Our procedure, as is the case with the stochastic smoothing algorithms, may miss some small local minima including the global one. Consequently we do not view our procedure as a global optimization method, but as a tool which searches for several solutions with small local minima and ensures that the local minimum already found will not be chosen again at a later stage. If the goal is the global minimization of the objective function, then more complicated algorithms, like simulated annealing and other global optimizers, may be used. Unfortunately they require many evaluations of the objective function which leads to long run times.

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, "Nonparametric discriminant analysis via recursive optimization of Patrick-Fisher distance", *IEEE Trans. on Syst., Man, Cybern.*, vol.28B, No 1, 1998.
3. M.E. Aladjem, " Recursive training of an ML neural network for pattern recognition", *IEEE Trans. on Neural Networks* (submitted).
4. M.E. Aladjem, "Training of an ML neural network for classification via recursive reduction of the class separation", *14th Int. Conf. on Pattern Recognition*, Brisbane, Australia, 17-20 August, 1998 (submitted).
5. C.M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press Inc., New York, 1995.
6. J.H. Friedman, "Exploratory projection pursuit", *Journal of the American Statistical Association*, vol. 82,pp. 249-266, 1987.
7. G.J. McLachlan, *Discriminant Analysis and Statistical Pattern Recognition*, John Wiley & Sons, Inc., New York, 1992.
8. W.N.Venables and B.D.Ripley, *Modern Applied Statistics with S-Plus*, Springer-Verlag, New York, 1994.