ISTITUTO DI ANALISI DEI SISTEMI ED INFORMATICA
CONSIGLIO NAZIONALE DELLE RICERCHE

F. Lampariello, M. Sciandrone

EFFICIENT TRAINING OF RBF NEURAL NETWORKS FOR PATTERN RECOGNITION

R. 492 Dicembre 1998

Francesco Lampariello – Istituto di Analisi dei Sistemi ed Informatica del CNR, viale Manzoni 30 - 00185 Rome, Italy. Email: *lampariello@iasi.rm.cnr.it*.

Marco Sciandrone – Istituto di Analisi dei Sistemi ed Informatica del CNR, viale Manzoni 30 - 00185 Rome, Italy. Email: *sciandro@iasi.rm.cnr.it*.

Istituto di Analisi dei Sistemi ed Informatica, CNR
viale Manzoni 30
00185 ROMA, Italy

tel. ++39-06-77161
fax ++39-06-7716461
email: iasi@iasi.rm.cnr.it
URL: http://www.iasi.rm.cnr.it

Abstract

The problem of training a RBF neural network for distinguishing two disjoint sets in R^n is considered. The network parameters can be determined by minimizing an error function that measures the degree of success in the recognition of a given number of training patterns. In this paper, taking into account the specific feature of classification problems, where the goal is to obtain that the network outputs take values above or below a fixed threshold, we propose an approach alternative to the classic one that makes use of the least-squares error function. In particular, the problem is formulated in terms of a system of nonlinear inequalities, and a suitable error function, which depends only on the violated inequalities, is defined. Then, a training algorithm based on this formulation is presented. Finally, the results obtained by applying the algorithm to a set of test problems are compared with those obtained by adopting the commonly used least-squares error function. The results show the effectiveness of the proposed approach in RBF network training for pattern recognition.

Key words: Pattern recognition, neural network training, error functions

1. Introduction

Feedforward neural networks have increasingly been used in many areas for the solution of difficult real world problems. This is due to the approximation capability of these devices, i.e. to the property that any continuous function can be approximated within an arbitrary accuracy by means of a neural network, provided that its topology includes a sufficient number of hidden nodes (see e.g. [1]-[4]). Once the architecture has been defined, the network is determined by performing a training process, which can be viewed as nonlinear optimization problem where the goal is to find the network parameters that minimize a suitable error function. This is done by using a given number of pattern-target pairs, that are samples of the input-output mapping to be approximated.

Our attention in this paper is focused on the problem of training Radial Basis Function (RBF) networks in the field of pattern recognition, and more specifically for classification problems where the task is to assign a label to one of a number of discrete classes or categories. The error function commonly used for this purpose is the *least-squares* function. Different error functions can be adopted, such as the *cross-entropy* [5] or the *exponential* [6] function, in order to avoid some undesirable effects related to the use of the sum of squares. In the minimization of an error function of these kinds, the attempt is to reduce as much as possible the differences between the network outputs corresponding to the given inputs and the label values associated with the training patterns. In classification problems, however, it is sufficient to obtain that the network outputs take values above or below a fixed threshold. During the training process based on the foregoing error functions, even the patterns that are already correctly classified with respect to the threshold value give a contribution to the overall network error. This may imply a poor convergence rate of the optimization algorithm applied, since these patterns will unnecessarily influence both the search direction and the steplength used by the algorithm.

On the basis of these observations, we formulate the training problem simply in terms of a system of nonlinear inequalities and, for solving it, we consider a *threshold* error function to which only the patterns corresponding to the violated inequalities give a contribution. Then, making use of a standard routine for minimizing an error function of this kind, a specific algorithm is designed by introducing two scalar parameters, which represent the upper and lower reference levels for the network outputs with respect to the fixed threshold. The values of these parameters are automatically updated during the training process, according to the progress made in the recognition of the training patterns. In this way, it is possible to avoid that the algorithm will converge to points of the parameter space which do not provide an acceptable solution of the training problem.

Finally, the results obtained by applying the algorithm to a set of test problems are compared with those obtained by adopting the commonly used least-squares error function. The results show the effectiveness of the proposed approach in RBF network training for pattern recognition.

2. Radial basis function networks

A feedforward neural network is a computing device whose processing units (the *nodes*) are distributed in adjacent layers connected through unidirectional links (the *weights*). In particular, a RBF network is a fully connected network with one "hidden" layer, whose nodes have some radially symmetric function as activation function. Such a network implements an input-output mapping $f : R^n \rightarrow R$ according to

$$f(x) = \sum_{i=1}^{n_r} \lambda_i \phi(\|x - c^i\|),$$

where $x \in R^n$ is the input vector, $\phi(\cdot) : R^+ \to R$ is a radially symmetric function, $\|\cdot\|$ denotes the Euclidean norm, $\lambda_i \in R$, $i = 1, \ldots, n_r$ are the weights, $c^i \in R^n$, $i = 1, \ldots, n_r$ are known as the RBF *centers*, and n_r is the number of hidden nodes. Typical choices for the function $\phi(\|x - c^i\|)$ are:

$$\begin{aligned} &\exp(-\|x - c^i\|^2/2\sigma^2) && (\textit{Gaussian function}) \\ &(\|x - c^i\|^2 + \sigma^2)^{1/2} && (\textit{Direct multiquadric function}) \\ &(\|x - c^i\|^2 + \sigma^2)^{-1/2} && (\textit{Inverse multiquadric function}) \end{aligned}$$

where $\sigma > 0$ is the so-called "shift parameter".

The method of radial basis functions has been used in the theory of *multivariable interpolation* in high-dimensional space [7], [8]. It has been shown (see e.g. [2], [3]) that the RBF network is a universal approximator for continuous functions, provided that the number n_r of the hidden nodes is sufficiently large. This property makes the network a powerful tool for dealing with many real world problems. From both theoretical and practical investigations it appears that the performance of the RBF network is not greatly influenced by the choice of the activation function ϕ.

In order to approximate a given nonlinear mapping, the network parameters λ_i, c^i, $i = 1, \ldots, n_r$ have to be determined by using a finite set of input-output data (training process).

As regards the centers c^i, they may be chosen randomly among the input data or fixed at specific locations using some "clustering" technique (see e.g. [9]), i.e. placed in the regions where the input data are more meaningful. Then, the weights λ_i are computed by applying an optimization algorithm for minimizing a suitable cost function.

A different approach [4] is based on a supervised learning process involving both the weights and the centers of the network. Although this approach is more complex and computationally more expensive, it usually leads to an improvement of the network performance, and then it will be adopted in the sequel.

3. Pattern recognition via neural networks

Let us consider two disjoint point sets A and B in R^n. A reference value, say 1, is associated to the points in A and a different value, say 0, to those in B. The classification problem consists in distinguishing the two point sets, i.e., given an arbitrary point $x \in A \cup B$, in recognizing whether x belongs to A or to B. This problem can be dealt with by using a feedforward neural network. In particular, we consider a RBF network that implements the input-output mapping $f(w) : R^n \to R$, where the parameter vector $w \in R^{n_r(n+1)}$ is composed of the weights and the centers λ_i, c^i, $i = 1, \ldots, n_r$. By using a given number N_p of pattern-target pairs (the training set)

$$\text{TS} = \{(x_p, d_p) \in A \cup B \times \{0, 1\}, p = 1, \ldots, N_p\},$$

w is to be determined (network training) in such a way that

$$\begin{aligned} \forall x \in A \Rightarrow \ & f(x; w) = 1 \\ \forall x \in B \Rightarrow \ & f(x; w) = 0. \end{aligned}$$

The network is trained by minimizing an error function $E(w)$ that measures the degree of success in the recognition of the training patterns in TS

$$E(w) = \sum_{(x_p,d_p)\in \mathrm{TS}} E_p(w),$$

where $E_p(w)$ is the contribution of pattern x_p to the overall network error. Then, starting from a prefixed vector $w(0)$, new values of the parameters are iteratively determined to reduce the network error, according to a scheme of the form

$$w(k+1) = w(k) + \alpha(k)d(k)$$

where $\alpha(k)$ is the stepsize (*learning rate*) along the search direction $d(k)$, until a sufficiently small value of $E(w)$ is reached. Thus, a given point x is recognized to belong to A or to B by computing $z = f(x,w) - \tau$, where $\tau \in R$ is a fixed *threshold value*, and using the step function that maps negative numbers z into $\{0\}$ and non-negative numbers z into $\{1\}$.

The commonly used error function is the *least-squares* function

$$E(w) = 1/2 \sum_{(x_p,d_p)\in \mathrm{TS}} \big(d_p - f(x_p,w)\big)^2, \tag{1}$$

or a normalized version of it.

It has been observed that the use of an error function of this kind may have undesirable effects. For instance, it is possible that the training algorithm will converge towards regions of the parameter space where the wide majority of patterns are correctly classified, while a small number of these are severely misclassified. This may imply poor subsequent improvements in the training process, since the contribution of the few misclassified patterns to the overall error function is overcomed by that of the numerous correctly classified ones.

In order to avoid this drawback, some alternative error functions have been proposed. In particular, Solla *et al.* [5] have considered the *cross-entropy* (or *logarithmic*) function

$$E(w) = -\sum_{(x_p,d_p)\in \mathrm{TS}} \ln\Big[\big(f(x_p,w)\big)^{d_p}\big(1-f(x_p,w)\big)^{1-d_p}\Big]. \tag{2}$$

With this function, the contribution to the error gradient due to the misclassified patterns is significantly higher than with the least-squares function, making it easier for the algorithm to escape from bad regions of the parameter space. Note that it is necessary to ensure $0 < f(x_p,w) < 1$, by suitably rescaling the network outputs.

A different approach, proposed by Møller [6], is based on the use of the *exponential* error function

$$E(w) = 1/2 \sum_{(x_p,d_p)\in \mathrm{TS}} \exp\big\{-\alpha\big(f(x_p,w) - d_p + \beta\big)\big(d_p - f(x_p,w) + \beta\big)\big\}, \tag{3}$$

where α and β are positive scalars. The function (3) incorporates the constraint $|d_p - f(x_p,w)| \leq \beta$, $\forall (x_p,d_p) \in \mathrm{TS}$, that establishes an "acceptable" error level β for the network outputs with respect to the reference values. During the training process, the value of β, initially set to a relatively high level, is progressively reduced in size. The contribution of a severely misclassified pattern to this error function is significantly higher (depending on the choice of the parameter α) than those of the patterns for which the constraint is satisfied. In this way, it is unlikely that the training algorithm will converge towards bad regions of the parameter space.

We observe that, in the minimization of (1), (2) and (3), the goal is to reduce as much as possible the differences between the network outputs corresponding to the given inputs and the reference values associated with the training patterns, i.e. the algorithm searches for a vector w such that $f(x_p,w)$ is as close as possible to 1 or to 0.

In pattern recognition problems, however, it is sufficient to obtain that the network outputs take values above or below a fixed threshold, i.e. we have to find w such that $f(x_p,w) \geq \tau$ or $f(x_p,w) < \tau$. Therefore, at a given stage of the training process based on the foregoing error

functions, even the patterns $(x_p, 1)$ for which $f(x_p, w)$ is greater than or equal to τ and those $(x_p, 0)$ for which $f(x_p, w)$ is lower than τ, altough already correctly classified, give a contribution to the overall error function. This may imply a poor convergence rate of the algorithm used for minimizing $E(w)$, since these contributions will unnecessarily influence both the search direction and the steplength, which are calculated from the objective function.

4. An alternative formulation of the training problem for pattern recognition

We propose here an alternative approach that takes into account the specific features of pattern recognition, by defining a suitable error function. In particular, the problem is formulated in terms of a system of nonlinear inequalities as follows.

Let $\varepsilon > 0$ be an arbitrarily small real number which represents the tolerance in distinguishing whether a network output is above or below a fixed threshold τ. As discussed in the preceding section, the RBF network training problem consists of determining a vector $w \in R^{n_r(n+1)}$ such that the nonlinear inequalities (for simplicity and without loss of generality, we assume $\tau = 0$)

$$\begin{cases} f(x_p, w) \geq \varepsilon, & \forall (x_p, 1) \in \text{TS}; \\ f(x_p, w) \leq -\varepsilon, & \forall (x_p, 0) \in \text{TS} \end{cases} \tag{4}$$

are satisfied. Then, for solving system (4), we consider the following *threshold* error function

$$E(w) = \sum_{(x_p,1)\in \text{TS}} \big(\max\{0, \varepsilon - f(x_p, w)\}\big)^q + \sum_{(x_p,0)\in \text{TS}} \big(\max\{0, \varepsilon + f(x_p, w)\}\big)^q, \tag{5}$$

where $q \geq 2$ is a given integer.

The properties of the function (5) are

(i) $E(w)$ is non-negative;
(ii) $E(w) = 0$ iff w is a solution of system (4);
(iii) $E(w)$ is continuously differentiable $q - 1$ times.

Therefore, by (i) and (ii), the problem becomes that of finding global minima of (5).

It is evident that this function is only dependent on the violated inequalities, so that, during its minimization, the misclassified training patterns only contribute to the overall error.

We observe that, since ε must be small, due to the structure of the function (5), there are regions of the parameter space where the overall error is close to zero and, at the same time, the number of the misclassified patterns may be relatively high. Therefore, the minimization algorithm could be trapped in one of these regions, reaching a point where $E(w) \simeq 0$ with a very low gradient norm, which may not provide an acceptable solution of the training problem, although representing a "good" solution of the optimization problem. A similar situation can not occur when the least-squares function is adopted as error function, since in this case a value of the overall error close to zero implies a high percentage of the correctly classified patterns. In other words, by using least-squares based error functions, a good approximation of a global minimum provides in any case a good solution of the training problem, whereas this correspondence may not hold when the function (5) is adopted. In order to overcome this drawback, we substitute in the threshold error function the fixed tolerance ε with two adjustable scalar parameters $\delta_u > 0$ and $\delta_\ell > 0$, which represent the upper and lower reference levels for the network outputs with respect to the fixed threshold. The values of these parameters are systematically reduced, during the minimization process, according to the progress made in the recognition of the training patterns.

In particular, after a certain number of iterations, the percentage of the patterns which have been correctly classified (classification accuracy), i.e. the percentage of the satisfied inequalities in system (4), is compared with that corresponding to the initial point. Then, the minimization process is continued with the same parameter values δ_u and δ_ℓ or with suitably reduced ones, depending on whether a sufficient increase in the classification accuracy has been obtained or not. This procedure is repeated until either all the training patterns have been correctly classified (or their number is sufficiently large), or both the parameters δ_u and δ_ℓ have been reduced at the prefixed tolerance.

We note that the use of two parameters, instead of a single one, is advantageous since it is possible to better handle the contribution to the overall error of the patterns in each of the two training subsets $\{x_p, 1\}$ and $\{x_p, 0\}$.

Formally, we define the following Threshold Error Training Algorithm (TETA). Let δ_u and δ_ℓ be the adjustable scalar parameters, N_u the number of patterns $(x_p, 1) \in \text{TS}$ and n_u the number of these for which $f(x_p, w) \geq \varepsilon$, N_ℓ the number of patterns $(x_p, 0) \in \text{TS}$ and n_ℓ the number of these for which $f(x_p, w) \leq -\varepsilon$, so that $p_u = n_u/N_u$ and $p_\ell = n_\ell/N_\ell$ are the fractions of the correctly classified patterns in the two subsets, θ the minimum relative increase in the classification fraction required for maintaining the value of δ_u or δ_ℓ unaltered and N the maximum allowed number of iterations performed by a given algorithm used for minimizing the overall error.

The Training Algorithm TETA

Data. Let $w^{(i)} \in R^{n_r(n+1)}$ be the starting parameter vector, $\varepsilon > 0$, $\delta_u > \varepsilon$, $\delta_\ell > \varepsilon$, $\theta > 0$ real numbers and N a positive integer.

Step 0. Compute the fractions $p_u^{(i)}$ and $p_\ell^{(i)}$ corresponding to $w^{(i)}$ of the correctly classified patterns in the two training subsets of TS and the overall classification accuracy $p^{(i)} = \big(p_u^{(i)} N_u + p_\ell^{(i)} N_\ell\big)/N_p$.

Step 1. If $p^{(i)} = 1$ or if $\delta_u \leq \varepsilon$ and $\delta_\ell \leq \varepsilon$ stop.

Step 2. Starting from the point $w^{(i)}$, perform N iterations of a given algorithm for minimizing

$$E(w) = 1/2 \sum_{(x_p,1)\in \text{TS}} \big(\max\{0, \delta_u - f(x_p, w)\}\big)^2 + 1/2 \sum_{(x_p,0)\in \text{TS}} \big(\max\{0, \delta_\ell + f(x_p, w)\}\big)^2, \tag{6}$$

and let $w^\star$ be the reached point. Compute the corresponding fractions $p_u^\star$, $p_\ell^\star$, $p^\star = \big(p_u^\star N_u + p_\ell^\star N_\ell\big)/N_p$, and set $w^{(i)} := w^\star$.

Step 3. If $\dfrac{p^\star - p^{(i)}}{1 - p^{(i)}} > \theta$ go to Step 5.

Step 4. Compute $\Delta_u = \dfrac{p_u^\star - p_u^{(i)}}{1 - p_u^{(i)}}$ and $\Delta_\ell = \dfrac{p_\ell^\star - p_\ell^{(i)}}{1 - p_\ell^{(i)}}$.

If $\Delta_u \leq \theta$ and $\Delta_\ell \leq \theta$, then Update δ_u and Update δ_ℓ.

If $\Delta_u \geq \theta$, then Update δ_ℓ if $0 \leq \Delta_\ell < \theta$, or Update δ_u if $\Delta_\ell < 0$.

If $\Delta_\ell \geq \theta$, then Update δ_u if $0 \leq \Delta_u < \theta$, or Update δ_ℓ if $\Delta_u < 0$.

Step 5. Set $p_u^{(i)} := p_u^\star$, $p_\ell^{(i)} := p_\ell^\star$, $p^{(i)} := p^\star$, and go to Step 1.

Update δ_u**.** Let S_u be the set: $S_u = \{(x_p, 1) \in \text{TS} : \varepsilon < f(x_p, w^{(i)}) < \delta_u\}$ and n_{S_u} the number of patterns in S_u. If $S_u \neq \emptyset$, set

$$\delta_u = 1/n_{S_u} \sum_{(x_p,1)\in S_u} f(x_p, w^{(i)}).$$

Update δ_ℓ. Let S_ℓ be the set: $S_\ell = \{(x_p, 0) \in \text{TS} : -\delta_\ell < f(x_p, w^{(i)}) < -\varepsilon\}$ and n_{S_ℓ} the number of patterns in S_ℓ. If $S_\ell \neq \emptyset$, set

$$\delta_\ell = 1/n_{S_\ell} \sum_{(x_p,0)\in S_\ell} f(x_p, w^{(i)}).$$

We observe first that, the initial value of the parameters δ_u and δ_ℓ should be chosen sufficiently large (with respect to the tolerance ε), in order to obtain that the network output values corresponding to the patterns $(x_p, 1)$ become well separated from those corresponding to the patterns $(x_p, 0)$. On the other hand, the initial values should be of the same order of magnitude as the network outputs. Therefore, it appears reasonable to take, for δ_u and δ_ℓ, the absolute mean values of the outputs corresponding to the patterns that are initially already correctly classified in the two training subsets.

Starting from the initial point $w^{(i)}$, the optimization algorithm performs at most N iterations. The number N should be reasonably large in order to ensure a significant progress in the minimization of $E(w)$, so that the choice of N should be made taking into account the convergence properties of the algorithm used. After N iterations, if the overall classification accuracy has not been sufficiently increased or, possibly, it has been at once reduced, it is likely that the contribution to the overall error of the correctly classified patterns for which $\varepsilon < f(x_p, w) < \delta_u$ and of those for which $-\delta_\ell < f(x_p, w) < -\varepsilon$ has been prevailing over that of the misclassified patterns. In this case a reduction of the parameter δ_u or δ_ℓ or both is beneficial, since some correctly classified patterns will no more give a contribution and hence, that of the misclassified patterns will be reinforced. In particular, we distinguish whether the classification accuracy has not been sufficiently increased in both or in one of the two training subsets only. In the first case, both the parameters δ_u and δ_ℓ are reduced. In the second case, if in both the training subsets the classification accuracy has been increased, only the parameter is reduced corresponding to the subset in which the increase was insufficient, while if in one subset the classification accuracy is decreased, the parameter is reduced corresponding to the other subset. As regards the updating rule of δ_u and δ_ℓ, it is important that the their values be reduced gradually. Then, we take as new values, the mean of the outputs corresponding to the patterns in the sets S_u and S_ℓ, provided that these are not empty. In this last case, the corresponding value remains unchanged.

Finally, it is possible that both the parameters δ_u and δ_ℓ become lower than or equal to ε and the overall classification accuracy obtained is still unsatisfactory for the problem considered. In this case, since it is unlikely that a further progress could be obtained, the training algorithm should be restarted by performing a greater number N of iterations or with a suitably reduced value of the parameter θ.

We remark that an error function similar to (6) has been considered by Sontag and Sussmann [10], where, with reference to multilayer perceptron networks, the properties of the local minima have been analyzed. However, in [10], the parameters whose role is analogous to that of our adjustable parameters δ_u and δ_ℓ are taken fixed at the reference values d_p associated with the training patterns. In this connection, we observe that the reduction of δ_u and δ_ℓ during the training process and their updating rule, represent the outstanding characteristic of our approach.

5. Experimental results and conclusion

In this section we report the results obtained by applying the training algorithm TETA defined above to a set of classification problems taken from the literature. In particular, two of these are classical test problems, while the others consist of real world data taken from the PROBEN1 benchmark site [11]. The aim of all experiments is to show the efficiency of the approach proposed here, by comparing our results with those obtained by adopting the commonly used least-squares error function.

The algorithm TETA has been implemented by applying, for minimizing the overall error function (6), a preconditioned limited memory quasi-Newton conjugate gradient method (E04DGF routine, NAG library). Taking into account the effectiveness of this standard routine, we have set the maximum number of iterations at the value $N = 100$. As regards the other parameters, the values chosen are $\varepsilon = 10^{-6}$ and $\theta = 0.01$.

In the starting parameter vector, the weights λ_i, $i = 1, \ldots, n_r$, are randomly chosen in the interval $[-0.5, 0.5]$, whereas the centers c^i, $i = 1, \ldots, n_r$ are randomly chosen among the input vectors x_p.

As regards the training algorithm that minimizes the least-squares error function (LSTA), we used again the E04DGF routine, allowing at most 10,000 function evaluations and starting from the same initial points as our algorithm.

The RBF networks that we used, are composed of hidden nodes having the direct multiquadric function as activation function and the shift parameter is set to $\sigma = 0.1$. The number n_r of hidden nodes has been chosen, for each problem, sufficiently large so as to ensure the correct classification of the set of training patterns.

1. **Two Spirals** problem

The task is to discriminate between two sets of points that are arranged in two interlocking spirals in the plane.

The training set consists of $N_p = 720$ patterns. The input space dimension is $n = 2$ and the network is composed of $n_r = 15$ hidden nodes.

2. **Two Concentric Classes** problem

The aim of this problem is that of distinguishing points in two nested circular non overlapping domains.

The training set consists of $N_p = 2,000$ patterns. The input space dimension is $n = 2$ and the network is composed of $n_r = 15$ hidden nodes.

3. **Cancer** problem [11]

This problem deals with the diagnosis of breast cancer. The task is to classify a tumor as either benign or malignant based on cell descriptions obtained by microscopic examination. This data set was created for the "Breast Cancer Wisconsin" of the University of Wisconsin Hospitals, Madison [12].

The training set consists of $N_p = 699$ patterns. The input space dimension is $n = 9$ and the network is composed of $n_r = 15$ hidden nodes.

4. **Heart** problem [11]

The task is to predict heart disease, i.e. to decide whether at least one of four major vessels is reduced in diameter by more than 50%. The decision is made based on personal data and results of various medical examination.

The training set consists of $N_p = 303$ patterns. The input space dimension is $n = 35$ and the network is composed of $n_r = 25$ hidden nodes.

5. **Credit Card** problem [11]

The task is to predict the approval or non-approval of a credit card for a customer.

The training set consists of $N_p = 690$ patterns. The input space dimension is $n = 51$ and the network is composed of $n_r = 30$ hidden nodes.

The results obtained for these classification problems, starting from three different initial points, are reported in the Figures 1-5. In each figure, the classification accuracy obtained by the two algorithms during the network training is plotted versus the number of function evaluations.

We observe first that, in all cases, the classification accuracy obtained by Algorithm TETA after a given number of function evaluations is greater than that obtained by Algorithm LSTA. Moreover, from the figures it is evident that, after the initial progress, the training algorithm that uses the least-squares error function does not obtain significant improvements in spite of an extended minimization process, whereas Algorithm TETA continues to obtain a progress. This confirms that, due to the contribution to the overall error of the patterns that have been already correctly classified, the behaviour of a training algorithm may be unsatisfactory. Note that this aspect may result of particular importance in the case where the network training is to be continued using new available data.

In all problems, except in the last one, Algorithm TETA obtains a high percentage of correctly classified patterns by performing a relatively small number of function evaluations. In the Credit Card problem, which is a large dimensional one, the required number of function evaluations near 10,000 is due to the large number of network parameters (the optimization problem has 1581 variables).

Finally, we observe that, in the problems considered here, the entire set of available data has been used as training set, since our aim was to show the effectiveness of the algorithm proposed in the recognition of the given patterns. However, we remark that, in order to ensure a satisfactory generalization property of the trained network, it is necessary to establish when the training process can be considered successfully achieved, for instance by monitoring the classification accuracy on further data (test set). In this connection, we note that the choice of the number of hidden nodes plays an important role in order to avoid overfitting or underfitting effects.

In conclusion, although the numerical experience reported here is not particularly extensive and does not include the generalization aspect, on the basis of the results obtained it appears that the use of a threshold-type error function and the definition of a suitable algorithmic scheme for handling its minimization represent an attractive approach in RBF network training for pattern recognition.

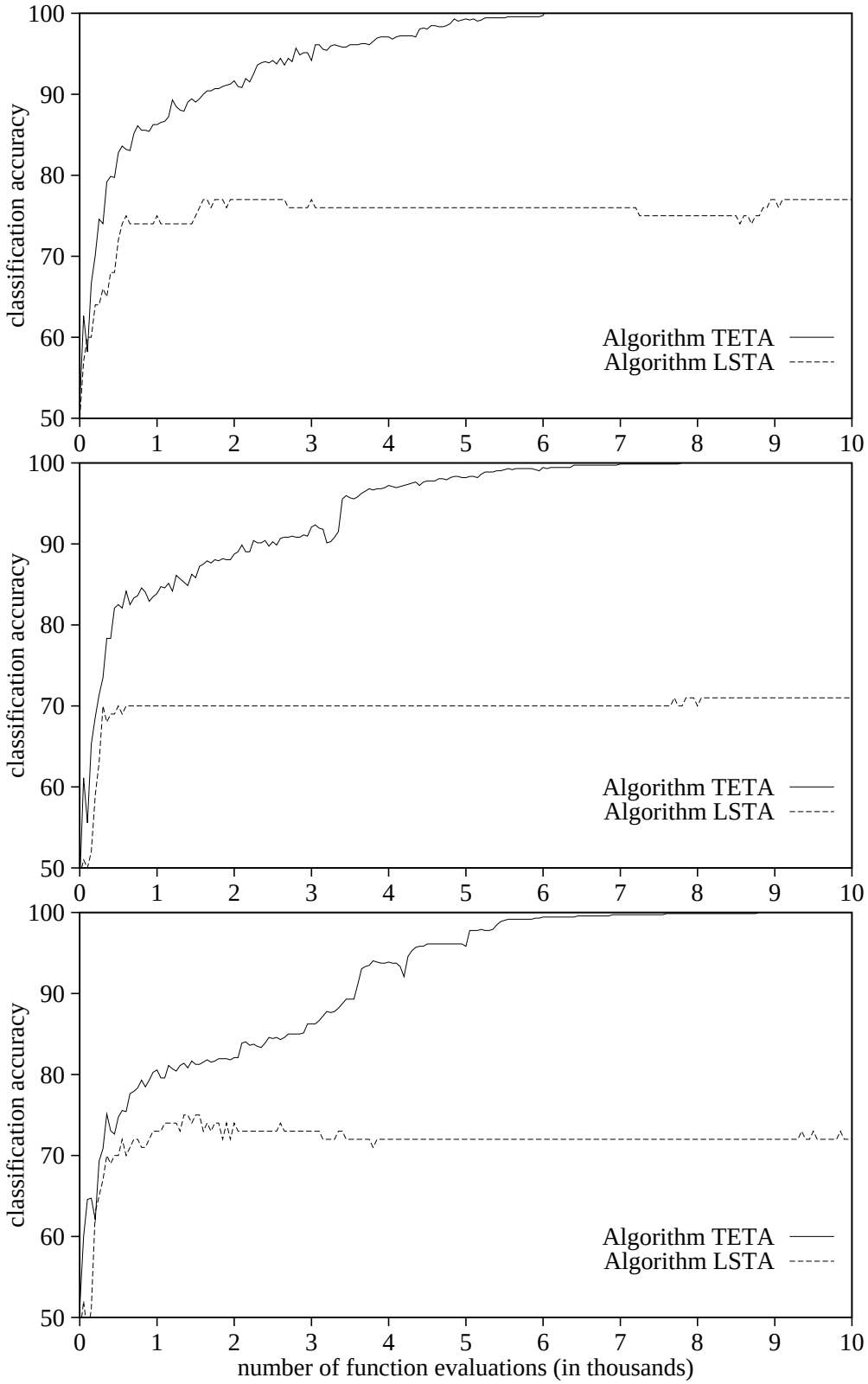

Fig. 1. Performance of the training algorithms for the Two Spirals problem, starting from three different initial points.

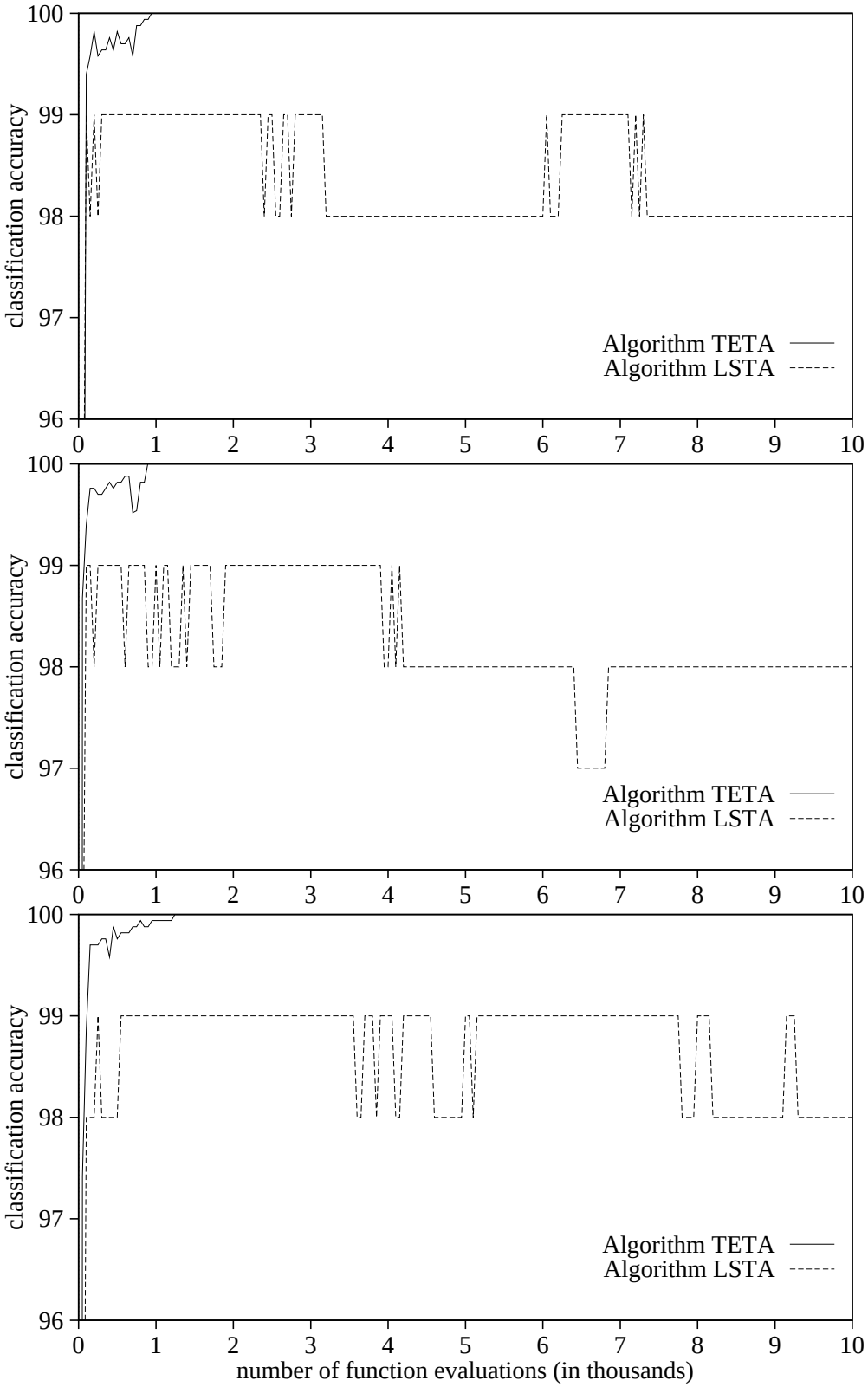

Fig. 2. Performance of the training algorithms for the Two Concentric Classes problem, starting from three different initial points.

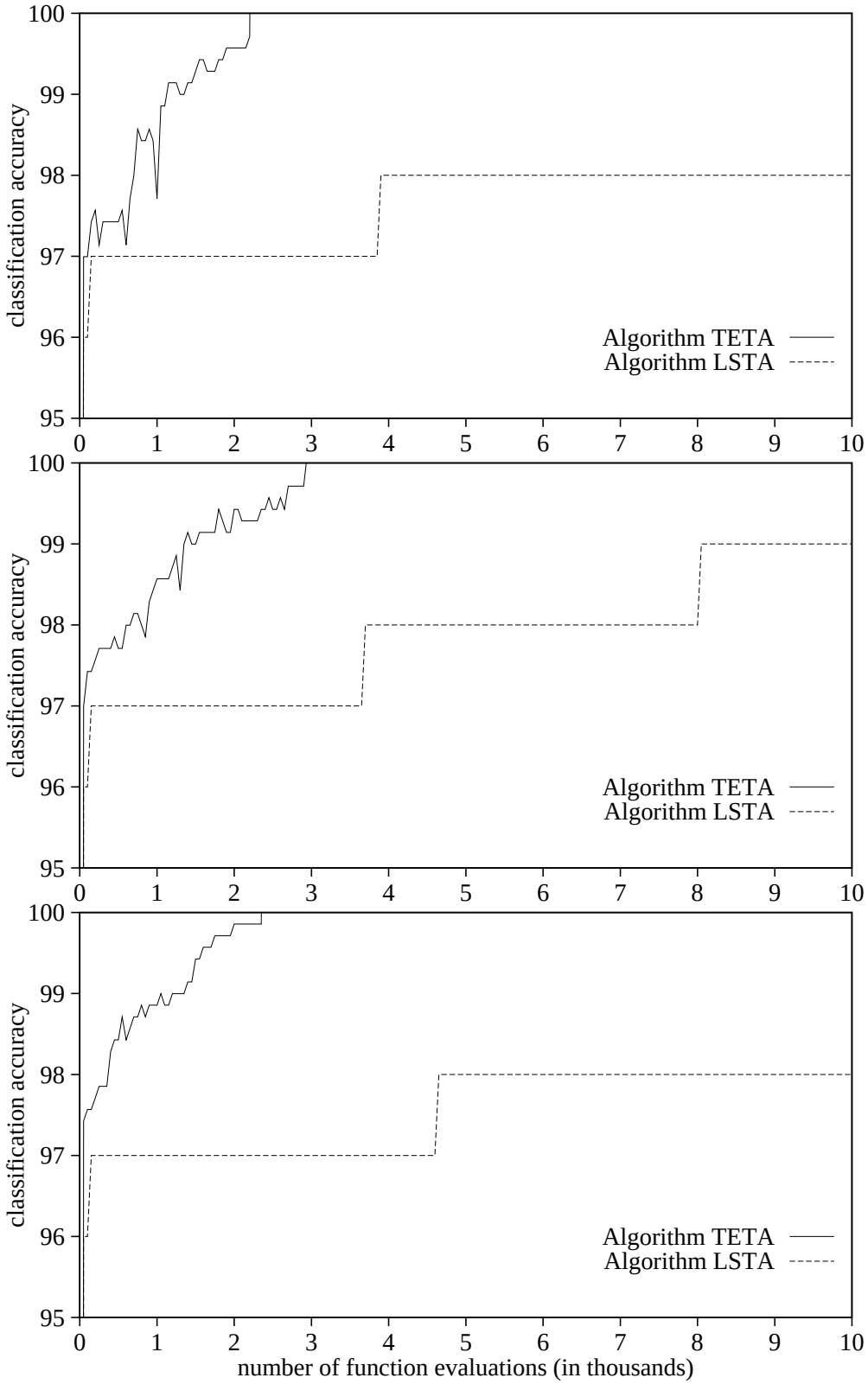

Fig. 3. Performance of the training algorithms for the Cancer problem, starting from three different initial points.

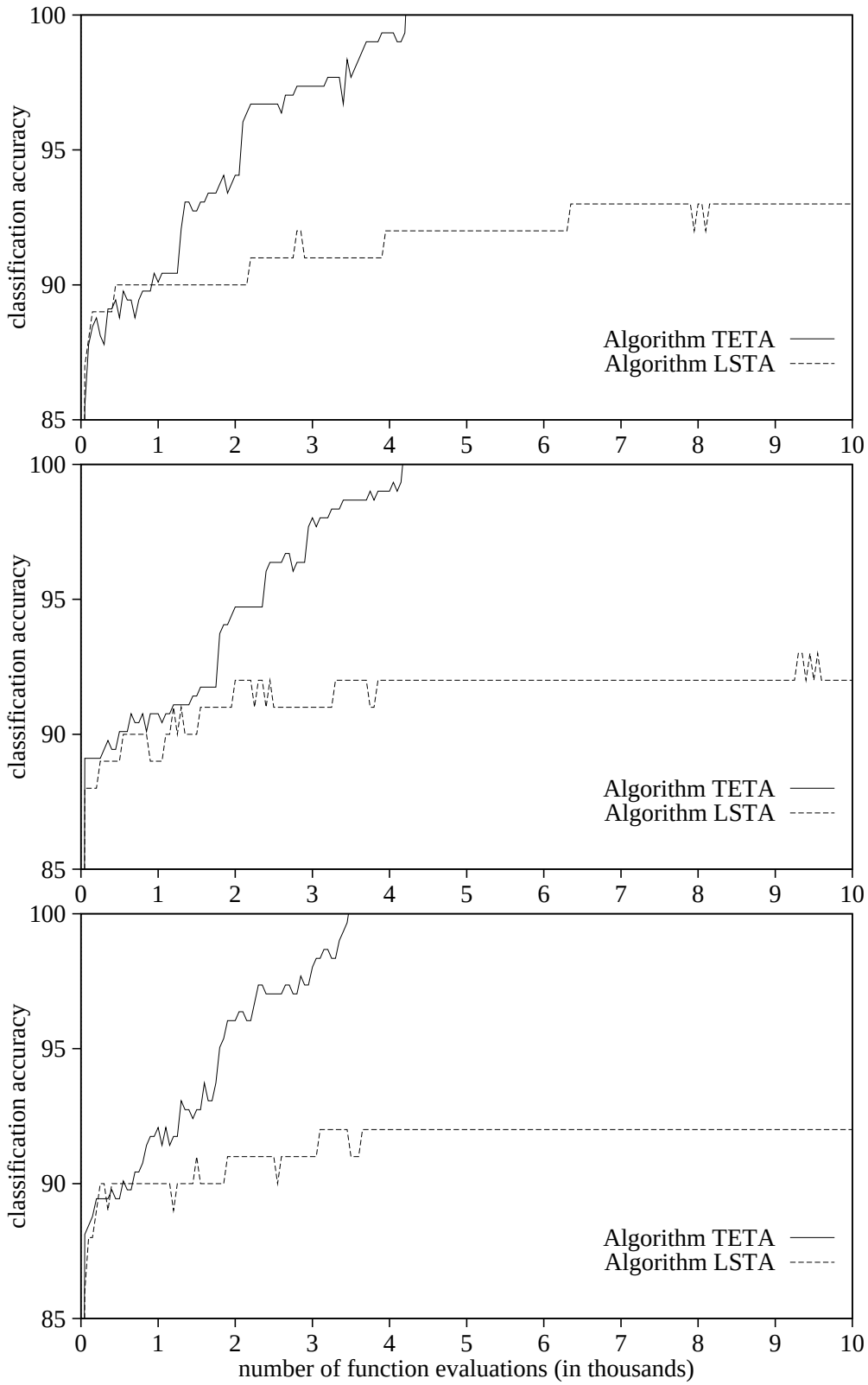

Fig. 4. Performance of the training algorithms for the Heart problem, starting from three different initial points.

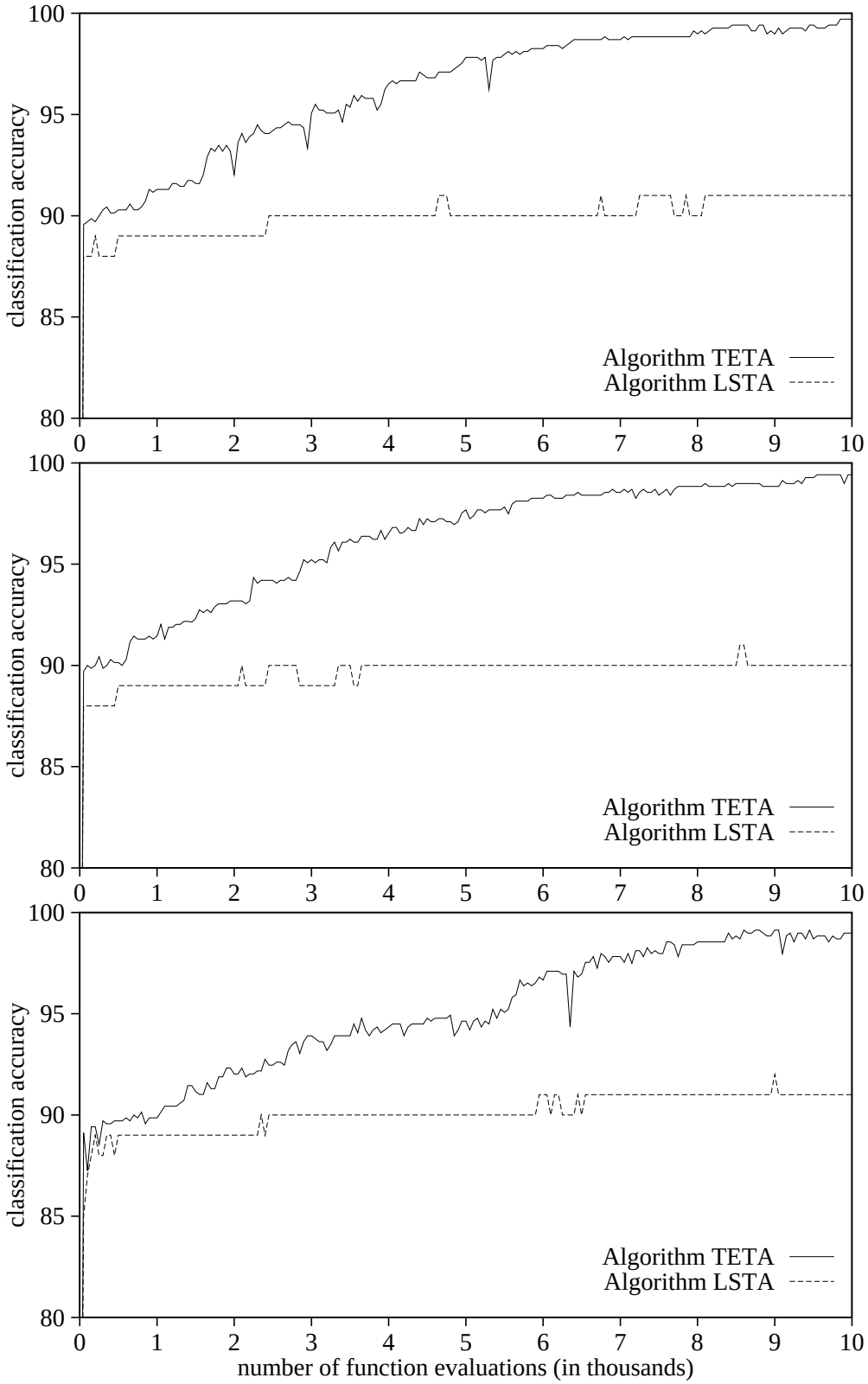

Fig. 5. Performance of the training algorithms for the Credit Card problem, starting from three different initial points.

References

[1] Hornik K., Stinchcombe M., White H.: Multilayer feedforward networks are universal approximators. Neural Networks 2: 359-366, 1989.

[2] Hartman E.J., Keeler J.D., Kowalsky J.M.: Layered neural networks with gaussian hidden units as universal approximators. Neural Computation 2: 210-215, 1990.

[3] Girosi F., Poggio T.: Networks and the best approximation property. Biological Cybernetics 63: 169-176, 1990.

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

[5] Solla S.A., Levin E., Fleisher M.: Accelerated learning in layered neural networks. Complex Systems 2: 625-640, 1989.

[6] Møller M.: Efficient training of feed-forward neural networks. Ph.D. Thesis, Daimi PB-464, Computer Science Department, Aarhus University, 1993.

[7] Micchelli C.A.: Interpolation of scattered data: distance matrices and conditionally positive definite function. Construct. Approx. 2: 11-22, 1986.

[8] Powell M.J.D.: Radial basis function approximations to polynomials. Proc. 12th Biennial Numerical Analysis Conf. (Dundee): 223-241, 1987.

[9] Musavi M.T., Ahmed W., Chan K.h., Faris K.B., Hummels D.M.: On the training of radial basis function classifiers. Neural Networks 5: 595-603, 1992.

[10] Sontag E.D., Sussmann H.J.: Backpropagation separates where perceptrons do. Neural Networks 4: 243-249, 1991.

[11] Prechelt L.: PROBEN1-A set of neural network benchmark problems and benchmarking rules. Fakultät für Informatik, Universität Karlsruhe, Tech. Rep. 21/94, Karlsruhe, Germany, 1994.

[12] Mangasarian O.L., Wolberg W.H.: Cancer diagnosis via linear programming. SIAM news 23: 1-18, 1990.