International Journal of Neural Systems, Vol. 0, No. 0 (April, 2000) 00–00
© World Scientific Publishing Company

ON THE INFLUENCE OF PARAMETER θ^- ON PERFORMANCE OF RBF NEURAL NETWORKS TRAINED WITH THE DYNAMIC DECAY ADJUSTMENT ALGORITHM

ADRIANO L. I. OLIVEIRA, ERICLES A. MEDEIROS, THYAGO A. B. V. ROCHA
MIGUEL E. R. BEZERRA and RONALDO C. VERAS
Department of Computing Systems, Polytechnic School of Engineering, Pernambuco State University
Rua Benfica, 455, Madalena, Recife – PE, Brazil, 50.750-410
E-mail: adriano@dsc.upe.br

Received (to be inserted
Revised by Publisher)

The dynamic decay adjustment (DDA) algorithm is a fast constructive algorithm for training RBF neural networks (RBFNs) and probabilistic neural networks (PNNs). The algorithm has two parameters, namely, θ^+ and θ^-. The papers which introduced DDA argued that those parameters would not heavily influence classification performance and therefore they recommended using always the default values of these parameters. In contrast, this paper shows that smaller values of parameter θ^- can, for a considerable number of datasets, result in strong improvement in generalization performance. The experiments described here were carried out using twenty benchmark classification datasets from both *Proben1* and the *UCI* repositories. The results show that for eleven of the datasets, the parameter θ^- strongly influenced classification performance. The influence of θ^- was also noticeable, although much less, on six of the datasets considered. This paper also compares the performance of RBF-DDA with θ^- selection with both AdaBoost and Support Vector Machines (SVMs).

1. Introduction

Radial basis functions network (RBFN) is a popular neural network architecture which has achieved good performance on both regression and classification problems [1,2,3]. There are several training methods for RBFNs in the literature [4,5,6,7]. In more traditional training methods, the number of hidden units of the network must be selected in advance [1,2,3]. Alternatively, a number of constructive training methods have been proposed for RBFNs [4,5,6,7,8,9]. In these methods, the number of hidden units is automatically determined during training. According to Ma and Khorasani, constructive training techniques offer a number of important advantages over other training techniques such as pruning algorithms and regularization-based techniques [10].

The dynamic decay adjustment (DDA) algorithm is a fast constructive algorithm proposed originally for training RBF neural networks [4] and later extended for PNNs [11]. The algorithm has achieved performance comparable to MLPs in a number of classification tasks and has a number of advantages for practical applications [4,11]. It is very fast with training concluding in only four to five epochs and the results do not depend on the initial values of the weights [4,11]. The advantages of DDA have motivated its implementation in hardware [12] as well as the proposal of a number of novel training methods based on it [8,9,13,14,15,16]. One of these methods has achieved generalization performance comparable to support vector machines (SVMs) on some image classification datasets [9,17].

The articles that introduced DDA have argued that the classification performance of the networks generated by the algorithm on test sets has weak dependence on training parameters θ^+ and θ^- [4,11]. Therefore, Berthold and Diamond advocate the use of the default values of these parameters, namely, $\theta^+ = 0.4$ and $\theta^- = 0.1$ ($\theta^- = 0.2$ in the case of PNNs) for training using DDA [4,11].

In contrast with what has been argued previously regarding DDA parameters, we have observed experimentally that for some datasets the value of θ^- considerably influences performance. The important influence of θ^- on performance has been observed on both benchmark image classification datasets from the UCI repository [13,9,8] and novelty detection in time series via classification [14]. DDA training with θ^- smaller than the default produces networks with larger number of RBF units in the hidden layer. This can improve coverage of the input space and thus enhance generalization performance; too small θ^- values can lead to overfitting [13,14,9].

To take advantage of the influence of θ^- on performance, a method has been proposed for improving performance through θ^- selection [13]. This method has indeed achieved much better classification performance than default RBF-DDA, yet at the expense of generating networks with increased number of hidden neurons [13]. This drawback of the method has motivated the proposal of other methods more recently [16,8,9].

One of these methods is RBF-DDA-T, which prunes the network after training by deleting all neurons which cover only one training pattern [16]. RBF-DDA-T significantly decreases complexity, yet at the expense of a slight degradation in classification accuracy with respect to default RBF-DDA.

Alternatively, we have proposed a method, referred to as RBF-DDA-SP, which is able to achieve performance comparable to RBF-DDA with θ^- selection with much smaller networks [8]. Similar results are also obtained by a method that integrates a data reduction algorithm with DDA with θ^- selection [9]. The disadvantage of these last two methods is that both introduce another parameter which must be carefully selected via cross-validation [9,8]. Thus, they have two critical parameters for model selection whereas DDA with θ^- selection has only one (θ^-) [13], which enables faster training, an important advantage in some applications.

This paper provides an experimental study on the influence of parameter θ^- on RBF-DDA generalization performance using twenty benchmark classification datasets from both the *Proben1* [20] and *UCI* [18] repositories. This is in contrast to an earlier work, which has considered only three optical character recognition (OCR) datasets [13]. This study aims to experimentally assess whether the influence of θ^- takes place on other datasets as well. In the experiments we consider a diverse set of classification problems, including image classification, medical problems and wine recognition. Our objective is to investigate whether it is advisable to use the default values of the parameters θ^+ and θ^- in all problems such as recommend by Berthold and Diamond [4,11] or if it is better to select the value of parameter θ^- as in a recently proposed method [13].

This paper is organized as follows. Next section reviews the DDA algorithm focusing on the role of parameters θ^+ and θ^-. Section 3 describes the experiments carried out along with the results and discussion as well as a comparison with AdaBoost and SVM. Finally, section 4 concludes the paper.

2. RBF-DDA Neural Networks

The DDA algorithm was inspired by an older constructive training algorithm called RCE (*Restricted Coulomb Energy*) [19]. Experimental results have shown that DDA outperforms RCE and other training algorithms for RBF networks in a number classification tasks [4].

An RBF trained by DDA is referred to as RBF-DDA. The number of units in the input layer represent the dimensionality of the input space. The input layer is fully connected to the hidden layer. RBF-DDAs have a single hidden layer. The number of hidden units is automatically determined during training. Hidden units use Gaussian activation functions. The activation $R_i(\overrightarrow{x})$ of a hidden unit i is given by

$$R_i(\overrightarrow{x}) = \exp\left(-\frac{||\overrightarrow{x} - \overrightarrow{r_i}||^2}{\sigma_i^2}\right) \qquad (1)$$

where $\overrightarrow{x}$ is the input vector and $||\overrightarrow{x} - \overrightarrow{r_i}||$ is the Euclidean distance between the input vector $\overrightarrow{x}$ and the Gaussian center $\overrightarrow{r_i}$. $\overrightarrow{r_i}$ and σ_i values are determined by the DDA training algorithm.

RBF-DDA uses 1-of-n coding in the output layer, with each unit of this layer representing a class. Classification uses a winner-takes-all approach, whereby the unit with the highest activation gives the class. Each hidden unit is connected to exactly one output unit. Each of these connections has a weight A_i. Output units use linear activation functions with values computed by $f(\overrightarrow{x}) = \sum_{i=1}^{m} A_i \times R_i(\overrightarrow{x})$, where m is the number of RBFs connected to that output.

The DDA training algorithm is constructive, starting with an empty hidden layer, with units being added to it as needed. The centers of RBFs, $\overrightarrow{r_i}$, and their widths, σ_i are determined by DDA during training. The values of the weights of connections between hidden and output layers are also given by the DDA algorithm.

The DDA algorithm relies on two parameters in order to decide about the introduction of new prototypes (RBF units) in the networks. One of these parameters is a *positive threshold* (θ^+), which must be overtaken by an activation of a prototype of the same class so that no new prototype is added. The other is a *negative threshold* (θ^-), which is the upper limit for the activation of conflicting classes [4,11]. The use of two thresholds results in improved classification in areas where the algorithm did not introduce new prototypes.

A trained RBF-DDA network holds the following two equations for every training pattern $\overrightarrow{x}$ of class c [4,11]:

$$\exists i : R_i^c(\overrightarrow{x}) \geq \theta^+ \tag{2}$$

$$\forall k \neq c, 1 \leq j \leq m_k : R_j^k(\overrightarrow{x}) < \theta^- \tag{3}$$

Notice that the above conditions do not guarantee the correct classification of all training patterns, because they hold for hidden units, not for output units.

The DDA algorithm for one training epoch is shown in Figure 1 [4]. This algorithm is executed until no changes in the values of the parameters of the network (number of hidden units and their respective parameters and weights values) are detected. This usually takes place in only four to five epochs [4]. This natural stopping criterion leads to networks that naturally avoid overfitting training data [4,11]. Notice that, in each epoch, the algorithm starts by setting all weights to zero because otherwise they would accumulate duplicate information about training patterns.

```
1: // reset weights:
FORALL prototypes $p_i^k$ DO
    $A_i^k = 0.0$
ENDFOR
2: // train one complete epoch
FORALL training pattern $(\overrightarrow{x}, c)$ DO
    IF $\exists p_i^c : R_i^c(\overrightarrow{x}) \geq \theta^+$ THEN
3: // sample covered by existing neuron of the same class
        $A_i^c += 1.0$
    ELSE
4: // "commit": introduce new prototype
        add new prototype $p_{m_c+1}^c$ with:
        $\overrightarrow{r}_{m_c+1}^c = \overrightarrow{x}$
        $A_{m_c+1}^c = 1.0$
        $m_c += 1$
5: // adapt radii
        $\sigma_{m_c+1}^c = \max_{k \neq c \wedge 1 \leq j \leq m_k} \{\sigma : R_{m_c+1}^c(\overrightarrow{r}_j^k) < \theta^-\}$
    ENDIF
6: // "shrink": adjust conflicting prototypes
    FORALL $k \neq c, 1 \leq j \leq m_k$ DO
        $\sigma_j^k = \max\{\sigma : R_j^k(\overrightarrow{x}) < \theta^- \}$
    ENDFOR
ENDFOR
```

Fig. 1. DDA algorithm (one epoch) for RBF training

During training, the DDA algorithm creates a new prototype for a given training pattern $\overrightarrow{x}$ only if there is no prototype of *the same class* in the network whose output $R_i(\overrightarrow{x}) \geq \theta^+$. Otherwise, the algorithm only increments the weight A_i of the connection associated with one of the RBFs (of the same class of the training pattern) which gives $R_i(\overrightarrow{x}) \geq \theta^+$ (line 3 of the DDA algorithm). When a new prototype is introduced in the network, its center will have the value of the training vector $\overrightarrow{x}$; the weight of its connection to the output layer is set to 1 (line 4 of the DDA algorithm). The width of the Gaussian will be chosen in such a way that the outputs produced by the new prototype for existing prototypes of conflicting classes is smaller than θ^- (line 5 of the DDA algorithm). Finally, there is a *shrink* *phase*, in which the widths of conflicting prototypes are adjusted to produce output values smaller than θ^- for the training pattern $\overrightarrow{x}$ (line 6 of the DDA algorithm).

2.1. *Influence of Parameter* θ^-

It was originally assumed that parameters θ^+ and θ^- would not influence the generalization performance of RBF-DDA neural networks [4,11]. This would be very interesting for practical applications, since the model selection phase would not be needed and training would take place always using the default values of the parameters, namely $\theta^+ = 0.4$ and $\theta^- = 0.1$, as proposed by Berthold and Diamond [4,11]. Conversely, it was observed that for some optical character recognition problems, the value of parameter θ^- strongly influences generalization performance [4,11]. Therefore, model selection via adjusting parameter θ^- is very important for improving RBF-DDA performance on such datasets.

Training RBF-DDA networks with θ^- smaller than the default produces networks with larger number of hidden RBF units. This can improve coverage of the input space, thereby improving performance for some datasets. Nonetheless, care must be taken since very small values of θ^- can produce networks that heavily overfit training data, thereby degrading generalization performance.

The influence of parameter θ^- on the number of hidden RBF units can be observed in Figures 2 and 3. Both figures illustrate the training phase of DDA in a two classes problem. In both cases, the network has already introduced two RBF units, one for each class. Figure 2 depicts the case of training with $\theta^- = 0.1$ (the default) value, whereas in Figure 3 training is carried out using $\theta^- = 0.01$. Note that, in both figures, the output of the Gaussian on the pattern of opposite class is slightly smaller than θ^-, as indicated in lines 5 and 6 of the DDA algorithm (Figure 1). The result of employing smaller values of θ^- is that the Gaussians introduced by DDA are narrower than in the case of default θ^- ($\theta^- = 0.1$), as can be observed comparing Figures 2 and 3.

Figures 2 and 3 show the phase of DDA training where one Gaussian of each class has already been introduced in the network and a new pattern X of class B is being examined. Notice that in the case of default θ^- the activation of both Gaussians for the new pattern X are above the positive threshold, that is, $R_A(X) \geq \theta^+$ and $R_B(X) \geq \theta^+$. Therefore, the DDA algorithm will not introduce a new RBF unit, since $R_B(X) \geq \theta^+$ (see lines 2 and 3 in Figure 1). Conversely, for $\theta^- = 0.01$, $R_B(X) < \theta^+$, as shown in Figure 3. Hence, in this case, a novel RBF unit will be introduced in the network by the DDA algorithm (see line 4 in Figure 1). This example has shown, therefore, that the value of θ^- directly influences the introduction of RBF units during training.

Thus, it can be very important in some problems to carefully select the value of θ^- to be used, since the number of RBF hidden units introduced by DDA is determined by this parameter. This can be done via simple validation or via cross-validation. In simple validation the original dataset is further divided into a *reduced training set* and a *validation set*. Training is carried out using only the reduced training set. The validation set is used to assess the generalization performance and therefore select the value of θ^- which provides the best performance. This approach was used in a previous work, which showed that performance on the validation set can be used (in optical character recognition datasets) to select the optimal θ^- [13]. Subsequently, the optimal θ^- is used to train using the complete (original) training set and to test using the test set [13].

The training method just described is presented in figure 4. Our method tests a number of θ^- values using the validation set, starting with the default value, $\theta^- = 0.1$. Next, θ^- is decreased by $\theta^- = \theta^- \times 10^{-1}$ (line 5 of algorithm 3). This is done because we have observed that performance does not change significantly for intermediate values of θ^- [13]. θ^- is decrease until the validation error starts to increase, since smaller values lead to overfitting [13]. The near optimal θ^- found by this procedure is used to train using the complete training set [13].

It is important to stress that the method introduced here maintains two important characteristics of the original DDA algorithm: (a) the constructive nature of the algorithm and (b) the effective use of all training data to adjust the parameters of the model (RBF network).

```
1: $\theta^+ = 0.4$
2: $\theta^-_{opt} = \theta^- = 10^{-1}$
3: Train one RBF-DDA with $\theta^-$ using the reduced
```

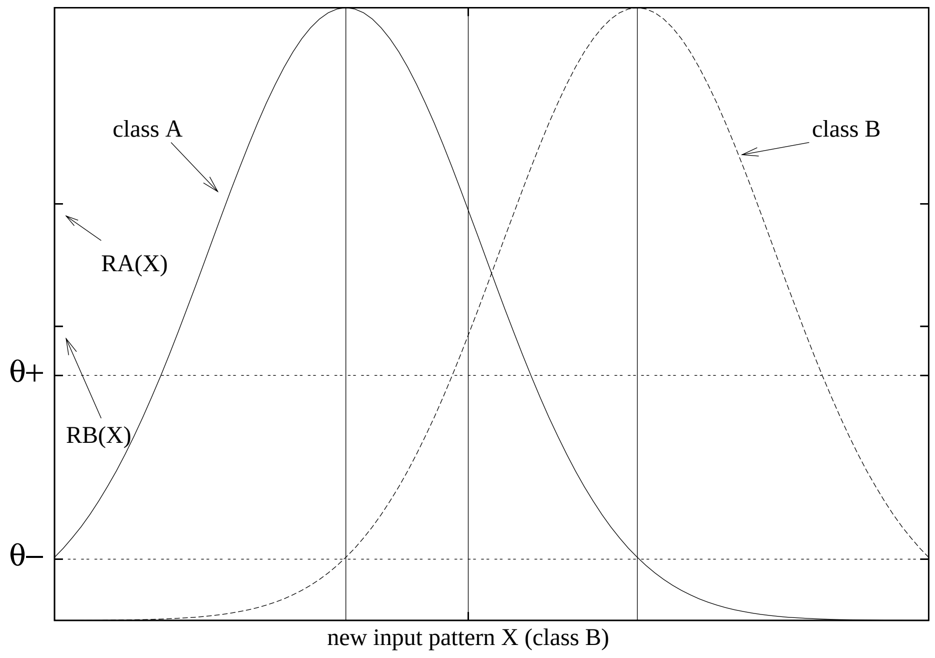

Fig. 2. Building an RBF-DDA network with $\theta^- = 0.1$ (default). Introducing a new pattern of class B during training.

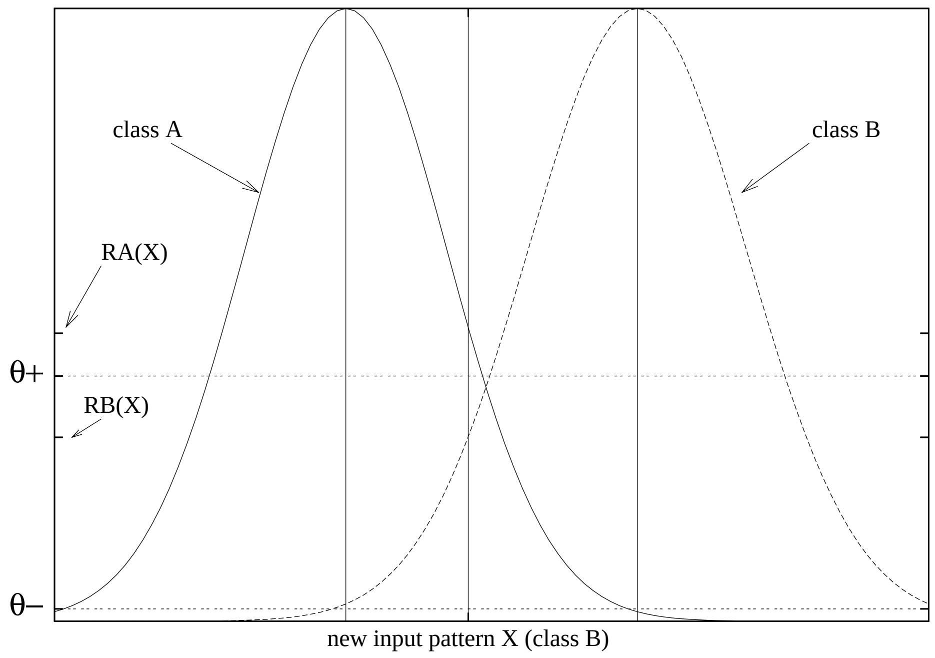

Fig. 3. Building an RBF-DDA network with $\theta^- = 0.01$. Introducing a new pattern of class B during training.

```
training set and test on the validation
    set to obtain $ValError = MinValError$
4: **REPEAT**
5:     $\theta^- = \theta^- \times 10^{-1}$
6:     Train one RBF-DDA with $\theta^-$ using the reduced training
           set and test on the validation set to obtain $ValError$
8:     **IF** $ValError < MinValError$
9:         $MinValError = ValError$
10:        $\theta^-_{opt} = \theta^-$
11:    **ENDIF**
12: **UNTIL** $ValError > MinValError$ **OR** $\theta^- = 10^{-10}$
13: Train one RBF-DDA with $\theta^-_{opt}$ using the complete training set
14: Test the RBF-DDA trained with $\theta^-_{opt}$ on the test set
```

Fig. 4. Improving RBF-DDA through θ^- selection

3. Experiments

3.1. *Datasets*

In this study we have used a diverse range of benchmark classification datasets from both Proben1 [20] and the UCI repository of machine learning databases [18]. The characteristics of the datasets used in the experiments are shown in table 1. We have used twenty datasets in total, with ten coming from each repository, as shown in table 1.

The simulations were carried out using *holdout*, whereby the dataset is divided into disjoint training and test sets. Training is carried out using only data from the training set. The test set is used to assess the generalization performance of the classifier. We have used the standard division of the datasets into training and test sets provided by the repositories (Proben1 and UCI). Each dataset from Proben1 comes with three different partitionings (permutations), since for small datasets results may vary significantly for different partitionings [20]. For instance the problem *cancer* is available in three datasets *cancer1*, *cancer2*, *cancer3*, which differ only in the ordering of the examples [20].

All experiments were carried out with the implementation of RBF-DDA available in the Stuttgart Neural Network Simulation (SNNS) [21].

3.2. *Results and Discussion*

For each dataset, the simulations were carried out by using the following values of θ^-: 0.2; 0.1; 10^{-2}; 10^{-3}; 10^{-4}; 10^{-5}; 10^{-6}; 10^{-7}; 10^{-8}; 10^{-9}; 10^{-10}. These values were also considered in a previous work [13] and were selected because we have observed that intermediate values do not significantly change classification performance.

Figures 5 and 6 depict classification error and number of hidden RBF units, respectively, as a function of parameter θ^- for the *soybean2* dataset. Figure 5 depicts both the error on the test set, obtained by training the networks using the *complete training set*, which comprise all training data, and testing using the test set. The same figure also shows the validation error, obtained by training the networks using a *reduced training set*, which has 75% of the training examples, and tested using a validation set which has the remaining 25% of the training examples.

Figure 5 shows that both test and validation errors decrease with decreasing values of θ^- (up to $\theta^- = 10^{-5}$), then the networks start to overfit training data leading to an increase in the errors. It is also important to observe that the performance on the validation set can be used to select the optimal value of θ^- (which gives the smaller error on the test set) as proposed in [13]. On the other hand, Figure 6 shows that the number of hidden RBF units increases as θ^- decreases. Thus, the improvement in performance achieved with smaller values of θ^- comes with a considerable increase in the complexity of the networks.

Tables 2 and 3 compare RBF-DDA trained with default and selected θ^- in the UCI and Proben1 datasets considered, respectively. These tables present, for each classifier, both the classification error on the test set and the number of hidden RBF units of the network. In the case of Proben1 datasets, results are provided for each of the three different partitionings. The results of these tables show that the selection of an adequate value of θ^- lead to a significant improvement in performance for eleven of the datasets considered (*Dermatology Database*,

Table 1. Datasets used in the experiments. (^{1}) from Proben1, (^{2}) from UCI

Dataset	**#patterns**	**#inputs**	**#classes**
Cancer (^{1})	699	9	2
Card (^{1})	690	51	2
Diabetes (^{1})	768	8	2
Gene (^{1})	3175	120	3
Glass (^{1})	214	9	6
Heart (^{1})	920	35	2
Horse (^{1})	364	58	3
Mushroom (^{1})	8124	125	2
Soybean (^{1})	683	82	19
Thyroid (^{1})	7200	21	3
Contraceptive Method Choice (^{2})	1473	9	3
Dermatology Database (^{2})	366	34	6
Haberman's Survival Data (^{2})	306	3	2
Iris Plant Database (^{2})	150	4	3
Wine Recognition Database (^{2})	178	13	3
Optdigits (^{2})	5620	64	10
Pendigits (^{2})	10992	16	10
Letter (^{2})	20000	16	26
Satimage (^{2})	6435	36	6
Segment (^{2})	2310	19	7

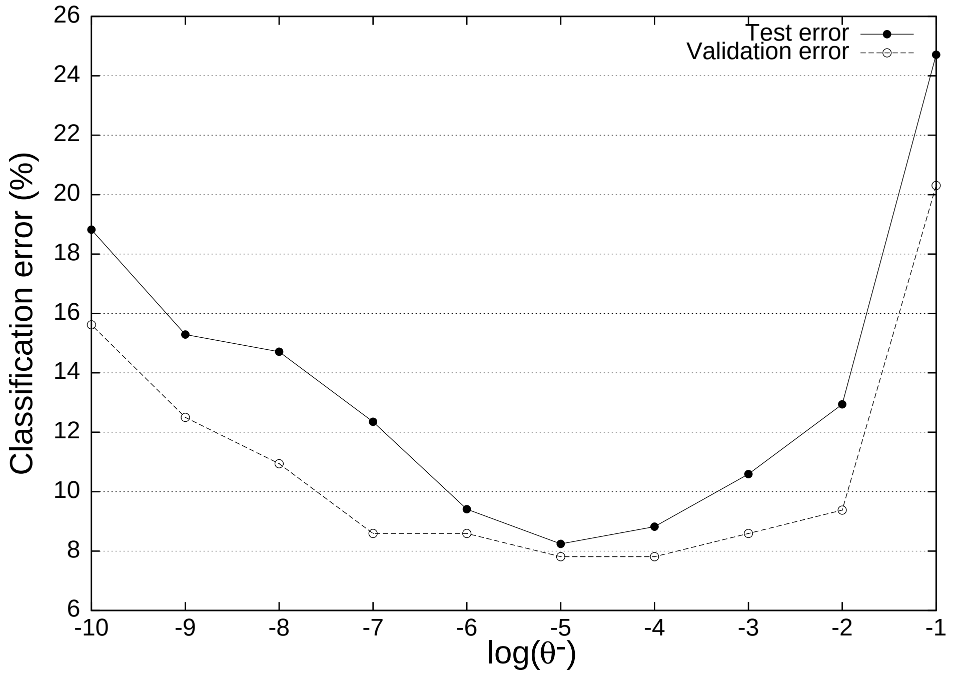

Fig. 5. Classification error on test and validation sets for *soybean2* dataset

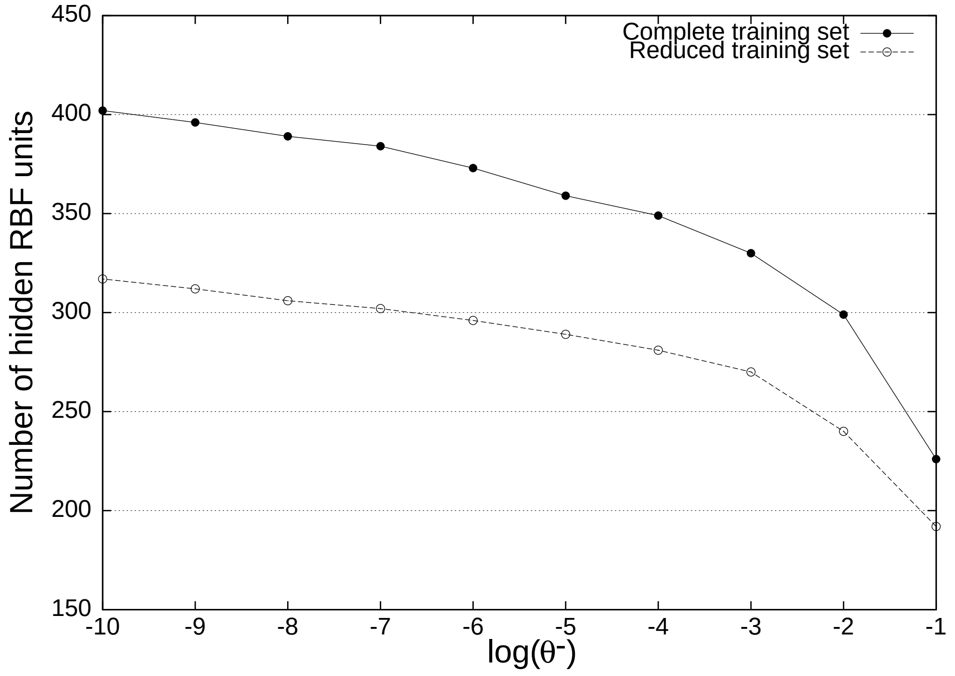

Fig. 6. Number of hidden RBFs for complete and reduced training sets for *soybean2* dataset

Wine Database, *Optdigits*, *Pendigits*, *Letter*, *Satimage*, *Segment*, *Cancer*, *Gene*, *Mushroom*, *Soybean*). For others six datasets a less significant improvement was observed (*Contrac. Method Choice*, *Diabetes*, *Glass*, *Heart*, *Horse*, *Thyroid*). In each table, the datasets for which an improvement in performance took place are depicted in bold. Notice that whenever the improvement in performance is due to the use of θ^- smaller than the default, the network produced is larger than that produced by default RBF-DDA. For instance, the results for the *Dermatology Database* shown in table 2 show that with $\theta^- = 0.1$ (the default value) the classification error on the test set is 28.26%. This error decreases significantly for selected θ^- (in this case, $\theta^- = 10^{-3}$) to 3.26%. This remarkable improvement in performance, nevertheless, comes with an increase in the number of hidden RBF units from 149 to 256.

3.3. *Comparison with AdaBoost and Support Vector Machines*

This section compares the generalization performance of RBF-DDA with θ^- selection with AdaBoost and support vector machines (SVMs) on some image recognition datasets. The results of AdaBoost and SVMs were obtained in recent papers available in the literature [22,23,24,25]. The datasets considered in the comparison are *optdigits*, *pendigits*, *letter* and *satimage.*

Table 4 compares performance of the proposed method for improving RBF-DDA with AdaBoost and SVM. The first column of this table shows the best AdaBoost results from MLP ensembles starting from two MLP components to 100 MLP components [22]. The second column of the table shows the best AdaBoost results reported in a previous work that compared a number of decoding methods [23]. The results shown here are the best obtained in that study, corresponding to AdaBoost with loss-based decoding using the exponential loss function and all-pairs output codes [23].

It can be seen that our method for training RBF networks outperforms AdaBoost on *optdigits*, *letter*, and *satimage.* Our method obtains results comparable to AdaBoost on the *pendigits* dataset. The method proposed in this paper also outperformed a recent ensemble classifier named DIvote on the *pendigits* datasets [24]. This classifier obtained 3.60% classification error on the test set [24] whereas RBF-

Table 2. Results on the UCI datasets: classification error on test sets and number of hidden neurons. θ^- in parenthesis.

Dataset	**RBF-DDA** $\theta^- = 0.1$	**RBF-DDA selected** θ^-
Contrac.	**53.8% [920]**	**53.5% [878]** (0.2)
Dermat.	**28.26% [149]**	**3.26% [256]** (10^{-3})
Haberman	29.87% [133]	29.87% [133] (0.1)
Iris Plant Database	2.63% [33]	2.63% [33] (0.1)
Wine	**8.89% [78]**	**4.44% [121]** (10^{-2})
Optdigits	**10.18% [1953]**	**2.78% [3812]** (10^{-5})
Pendigits	**8.12% [1427]**	**2.92% [5723]** (10^{-5})
Letter	**15.60% [7789]**	**5.30% [12861]** (10^{-4})
Satimage	**14.95% [2812]**	**8.55% [4099]** (10^{-4})
Segment	**25.14% [110]**	**14.19% [172]** (10^{-5})

DDA with θ^- selection achieved 2.92%.

The SVM results shown in table 4 are the best results for these datasets obtained in two previous works [23,25]. This table shows SVM results using both one-vs-all (OVA) and sparse output coding [23,25]. The *optdigits*, *letter* and *satimage* results were obtained in the work of Rifkin and Klautau [25] whereas results for *pendigits* were obtained in the work of Allwein et al. [23].

Note that our method achieves comparable results to SVMs in the *optdigits*, *pendigits* and *satimage* datasets. In the *letter* dataset, SVM obtains better generalization performance. It is important to stress that other output coding methods for SVMs considered in previous works lead to worse performance compared to our RBF-DDA methods [23,25].

Support vector machines are a powerful class of machine learning algorithms shown to achieve good performance on many classification and regression tasks. However, a recent study showed that simpler methods can be very competitive with SVMs on a number of datasets [26]. The training method proposed in this paper has built RBF networks that can achieve generalization performance comparable to SVMs in a number of tasks. Moreover, it can be faster to train then SVMs, which can be an important advantage in practical applications.

3.4. *Comparison with RBF-DDA-T and RBF-DDA-SP*

Finally, in this section we compare the method proposed in this article with both RBF-DDA-T [16] and RBF-DDA-SP [8], two training methods for RBF networks recently proposed in the literature. We have used four UCI datasets for this comparison. The simulation results are shown in table 5.

Notice that RBF-DDA-T is able to produce much smaller networks than RBF-DDA trained with default parameters, yet at the cost of considerable degradation in classification performance. For example, for the Letter dataset, RBF-DDA with default parameters creates a network with 7789 neurons with classification error of 15.60%, whereas the network produced by RBF-DDA-T has only 2837 neurons, yet the classification error increases to 25.32%. On the other hand, the results obtained by the method proposed in this article (RBF-DDA with θ^- selection) show that it significantly outperforms both RBF-DDA and RBF-DDA-T regarding classification performance (for instance, it obtained 5.30% classification error for the Letter dataset). Nonetheless, the disadvantage of the proposed method is that it produces much larger networks in compari-

Table 3. Results on the Proben1 datasets: classification error on test sets and number of hidden neurons. θ^- in parenthesis.

Dataset	**RBF-DDA** $\theta^- = 0.1$	**RBF-DDA** **selected** θ^-
Cancer1	**1.72% [160]**	**1.15% [277]** (10^{-4})
Cancer2	**4.02% [150]**	**2.87% [250]** (10^{-3})
Cancer3	3.45% [150]	3.45% [214] (10^{-2})
Card1	17.44% [364]	17.44% [364] (0.1)
Card2	18.02% [350]	18.02% [391] (10^{-2})
Card3	23.84% [355]	23.84% [420] (10^{-3})
Diabetes1	27.6% [490]	27.6% [420] (0.2)
Diabetes2	**27.6% [497]**	**27.08% [434]** (0.2)
Diabetes3	**23.96% [500]**	**22.4% [432]** (0.2)
Gene1	**47.8% [2147]**	**25.7% [2247]** (10^{-6})
Gene2	**47.8% [2155]**	**23.8% [2248]** (10^{-5})
Gene3	**49.2% [2148]**	**26.4% [2248]** (10^{-6})
Glass1	28.3% [121]	28.3% [121] (0.1)
Glass2	33.96% [116]	33.96% [132] (10^{2})
Glass3	**33.96% [122]**	**28.3% [139]** (10^{-2})
Heart1	19.13% [475]	19.13% [475] (0.1)
Heart2	21.74% [444]	21.74% [444] (0.1)
Heart3	**25.65% [434]**	**24.78% [383]** (0.2)
Horse1	**30.77% [256]**	**27.47% [261]** (10^{-2})
Horse2	35.16% [254]	35.16% [254] (0.1)
Horse3	**37.36% [255]**	**32.97% [263]** (10^{-2})
Mushroom1	**2.46% [389]**	**0.00% [3645]** (10^{-3})
Mushroom2	**2.26% [389]**	**0.00% [3633]** (10^{-3})
Mushroom3	**2.90% [378]**	**0.00% [3611]** (10^{-3})
Soybean1	**32.94% [221]**	**10.59% [366]** (10^{-5})
Soybean2	**24.71% [226]**	**8.24% [359]** (10^{-5})
Soybean3	**34.12% [220]**	**12.94% [256]** (10^{-4})
Thyroid1	**7.28% [2030]**	**6.50% [3928]** (10^{-4})
Thyroid2	**7.22% [2025]**	**6.61% [3493]** (10^{-3})
Thyroid3	**7.94% [2037]**	**6.50% [3862]** (10^{-5})

Table 4. Classification errors on test sets: comparison with AdaBoost and SVMs. (1) results from [22] and (2) results from [23]

Dataset	AdaBoost (1)	AdaBoost (2)	SVM (OVA)	SVM (Sparse)	RBF-DDA (θ^- selection)
optdigits	4.67%	-	2.73%	3.01%	2.78%
pendigits	3.17%	2.90%	2.50%	2.70%	2.92%
letter	19.91%	7.10%	2.75%	3.55%	5.30%
satimage	-	11.40%	7.80%	8.85%	8.55%

son with RBF-DDA trained with default parameters and RBF-DDA-T.

Table 5 shows RBF-DDA-SP results considering three different percentages of pruning (30%, 40% and 50%) [8]. Observe that the method proposed here outperforms RBF-DDA-SP in all datasets and considering the three percentages of pruning applied to RBF-DDA-SP. In fact, RBF-DDA with θ^- selection is the method which achieved the best classification performance in the experiments reported in table 5. RBF-DDA-SP is a method based on both pruning and model selection [8]. Therefore, it has two critical parameters, namely, θ^- and the percentage of pruning, p. On the other hand, RBF-DDA with θ^- has only one critical parameter. This means that model selection with RBF-DDA with θ^- selection is much faster than with RBF-DDA-SP, which can be an important advantage in practical applications.

4. Conclusions

We have presented an experimental study on the influence of parameter θ^- on the generalization performance of RBF-DDA. The papers that introduced the dynamic decay adjustment (DDA) algorithm assumed that the value of parameter θ^- would not significantly influence performance and therefore they recommended using the default value of this parameter [4,11]. In contrast, it was recently observed that for some optical character recognition datasets values of θ^- smaller than the default produce much better networks in terms of generalization performance [13].

We have carried out simulations using twenty benchmark classification datasets from both the *proben1* and *UCI* repositories. The datasets come from a large variety of problems such as image classification, wine recognition and medical problems. The results have shown that for eleven of these datasets the parameter θ^- strongly influences classification accuracy. For others six of these datasets a smaller improvement in performance was also observed. The simulation results have also shown that smaller values of θ^- produce bigger networks. In the case of the remaining three datasets, the default value of θ^- ($\theta^- = 0.1$) achieved the best performance. This demonstrates that for a given classification problem it is very important to carefully select the value of parameter θ^- via cross-validation in order to boost performance of RBF-DDA networks as in [13].

Acknowledgments

The authors would like to thank CNPq (Brazilian Research Agency) and Pernambuco State University for their financial support. The authors would also like to thank the anonymous referees for their helpful comments and suggestions.

References

1. Howlett, R. L., Jain, L. C. (Eds.), Radial Basis Function Networks 1 - Recent Developments in Theory and Applications. Vol. 66 of Studies in Fuzziness and Soft Computing. Springer-Verlag (2001).
2. Theodoridis, S., Koutroumbas, K.,. Pattern Recognition, 2nd Edition. Elsevier (2003).
3. Nabney, I.. Netlab: Algorithms for Pattern Recognition. Advances in Pattern Recognition. Springer (2001).
4. M. R. Berthold and J. Diamond. Boosting the performance of RBF networks with dynamic decay adjustment. In G. T. et al, editor, *Advances in Neural Information Processing*, volume 7, pages 521–528. MIT Press (1995).
5. Oyang, Y.-J., S.-C.Hwang, Y.-Y.Ou, Chen, C.-Y., Chen, Z.-W. Data classification with radial basis function networks based on a novel kernel density estimation algorithm. IEEE Transactions on Neural Networks 16 (1), 225–236 (2005).
6. Lee, S.-J., Hou, C.-L., An ART-Based construction of RBF networks. IEEE Transactions on Neural Networks 13 (6), 1308–1321 (2002).
7. Simon, D., Training radial basis neural networks with the extended kalman filter. Neurocomputing 48, 455–475 (2002).
8. A. L. I. Oliveira, B. J. M. Melo, and S. R. L. Meira. Improving constructive training of RBF networks through selective pruning and model selection. *Neurocomputing*, 64:537–541 (2005).
9. A. L. I. Oliveira, B. J. M. Melo, and S. R. L. Meira. Integrated method for constructive training of radial basis functions networks. *Electronics Letters*, 41(7):429–430 (2005).
10. Ma, L., Khorasani, K., New training strategies for constructive neural networks with application to regression problems. Neural Networks 17, 589–609 (2004).
11. M. R. Berthold and J. Diamond. Constructive training of probabilistic neural networks. *Neurocomputing*, 19:167–183 (1998).
12. M. Habib and A. Mourad. Architecture and design methodology of the RBF-DDA neural network. In

Table 5. Comparison of constructive training methods for RBF networks. The table shows classification errors on test sets and number of hidden RBFs.

Method	Optdigits	Pendigits	Letter	Satimage
RBF-DDA (default)	10.18% [1953]	8.12% [1427]	15.60% [7789]	14.95% [2812]
RBF-DDA-T (default)	14.75% [655]	8.43% [978]	25.32% [2837]	24.75% [662]
RBF-DDA (θ^- sel.)	2.78% [3812]	2.92% [5723]	5.30% [12861]	8.55% [4099]
RBF-DDA-SP (30%, θ^- sel.)	3.13% [2672] (0.23%)	3.04% [4344] (0.14%)	6.54% [9358] (0.18%)	9.18% [2934] (0.27%)
RBF-DDA-SP (40%, θ^- sel.)	3.30% [2292] (0.28%)	3.17% [3884] (0.18%)	7.10% [8191] (0.12%)	9.59% [2546] (0.32%)
RBF-DDA-SP (50%, θ^- sel.)	3.57% [1912] (0.26%)	3.29% [3424] (0.19%)	8.00% [7023] (0.25%)	10.05% [2157] (0.40%)

Proc. of IEEE Int. Symposium on Circuits and Systems (ISCAS'98), volume 3, pages 199–202 (1998).

13. A. L. I. Oliveira, F. B. L. Neto, and S. R. L. Meira. Improving RBF-DDA performance on optical character recognition through parameter selection. In *Proc. of the 17th International Conference on Pattern Recognition (ICPR'2004)*, volume 4, pages 625–628. IEEE Computer Society Press (2004).
14. A. L. I. Oliveira, F. B. L. Neto, and S. R. L. Meira. Improving novelty detection in short time series through RBF-DDA parameter adjustment. In *Proc. of International Joint Conference on Neural Networks (IJCNN'2004)*, volume 3, pages 2123–2128, Budapest, Hungary, 2004. IEEE Press (2004).
15. J. Paetz. Knowledge-based approach to septic shock patient data using a neural network with trapezoidal activation functions. *Artificial Intelligence in Medicine*, 28:207–230 (2003).
16. J. Paetz. Reducing the number of neurons in radial basis function networks with dynamic decay adjustment. *Neurocomputing*, 62:79–91 (2004).
17. A. L. I. de Oliveira. *Neural Networks Forecasting and Classification-Based Techniques for Novelty Detection in Time Series.* PhD thesis, Center for Informatics, UFPE, Brazil (2004). Available at http://adriano.dsc.upe.br.
18. C. Blake and C. Merz. UCI repository of machine learning databases. [http://www.ics.uci.edu/~mlearn/MLRepository.html], (1998).
19. M. J. Hudak. RCE classifiers: Theory and practice. *Cybernetics and Systems*, 23:483–515 (1992).
20. L. Prechelt. Proben1 – a set of neural networks benchmark problems and benchmarking rules. Technical Report 21/94, Universität Karlsruhe, Germany (1994).
21. A. Zell. *SNNS - Stuttgart Neural Network Simulator, User Manual, Version 4.2.* University of Stuttgart and University of Tubingen (1998).
22. C. Kaynak and E. Alpaydin. Multistage cascading of multiple classifiers: One man's noise in another man's data. In *Proc. of the 17th International Conference on Machine Learning*, pages 455–462 (2000).
23. E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *J. Machine Learning Research*, 1:113–141 (2000).
24. N. V. Chawla, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer. Learning ensembles from bites: A scalable and accurate approach. *J. Machine Learning Research*, 5:421–451 (2004).
25. R. Rifkin and A. Klautau. In defense of one-vs-all classification. *J. Machine Learning Research*, 5:101–141 (2004).
26. D. Meyer, F. Leisch, and K. Hornik. The support vector machine under test. *Neurocomputing*, 55:169–186 (2003).