Knowledge Enhancement and Reuse with Radial Basis Function Networks

Joydeep Ghosh Arindam C Nag
Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712

Abstract— This paper presents a technique for enhancing an RBFN when provided with additional information in the form of new features, without retraining or resorting to the original features. The proposed technique improves the learning speed as well as network performance as compared to a network that is trained from scratch. We also present a method of reusing knowledge embedded in an RBFN for initializing another RBFN to be trained on a related problem. Both methods have several real-life applications.

I. Introduction

In many real life situations, additional knowledge or data becomes available after a model (such as a classifier or function approximator) has already been learned based on existing data and knowledge [8], [4]. Such new information can be in the form of [i]additional data samples, [ii]additional data attributes (features), or [iii] new qualitative knowledge about the problem (e.g. expert rules). For example, suppose a network has been trained to predict rainfall patterns based on features derived from a set of sensor readings. Later on, a new type of sensor becomes available for measuring additional environmental information. Features extracted from the new sensor reading can potentially improve the existing prediction. In remote sensing, land cover has been classified for decades using multispectral data with O(10) bands/pixel; but lately extra information is available through new hyperspectral sensors that provide about 200 bands/pixel [6]. In both situations, one wonders if a new network needs to be trained from scratch to include all the existing features along with the new features, or the existing trained network can be modified (enhanced) to accommodate the new features. The latter case may yield a much faster (re)training time for comparable quality of results.

Closely related to the issue of such incremental model enhancement is that of knowledge reuse [5], whereby existing solutions to a set of problems are leveraged to help solve a different but related problem. For example, suppose we have developed a system to identify and classify different types of vehicles in desert-like terrain. If there is a new need to identify the same or similar vehicle types in mountainous terrian, does one need to start from scratch or can one somehow incorporate the existing system to obtain a quicker and better solution to the new but related problem?

In this paper, we study how both situations can be tackled using when the existing models comprise of Radial basis function networks(RBFNs), which are universal approximators and also have useful localization properties [7], [3]. For network reuse, the techniques involved include methods for systematically eliminating input features (reducing input data dimensionality) from a trained network, ways of adding features to a trained network, and ways of modifying the new network's output based on the outputs of existing networks [4]. We note that the problem of eliminating features from a trained network is related to the problem of recall in the presence of missing variables, e.g. [9], [10].

II. Knowledge Enhancement in RBFNs using Additional Features

In this section, we present techniques for knowledge enhancement in radial basis function networks when additional features become available.

A. RBF Network Classifier

One of the main reasons for the popularity of RBF networks is that the training process can be clearly divided into two stages with each stage relatively independent of the other. For classification, we would like to model the posterior probabilities $p(C_k|\mathbf{x})$ for each class C_k, where $\mathbf{x}$ is the input feature vector. Following Bishop [3], if the unconditional density of the input data $p(\mathbf{x})$ is approximated using a Gaussian mixture model:

$$p(\mathbf{x}) = \sum_{j=1}^{M} p(\mathbf{x}|j)P(j). \tag{1}$$

and the hidden units are initialized as the mixture components, j, we get a normalized RBFN:

$$P(C_k|\mathbf{x}) = \sum_{j=1}^{M} w_{kj}\,\phi_j(\mathbf{x}), \tag{2}$$

where the hidden unit activations are given by

$$\phi_j(\mathbf{x}) = \frac{p(\mathbf{x}|j)P(j)}{\sum_{i=1}^{M} p(\mathbf{x}|i)P(i)} = P(j|\mathbf{x}) \tag{3}$$

and the hidden to output unit weights are given by

$$w_{kj} = \frac{P(j|C_k)P(C_k)}{P(j)} = P(C_k|j). \quad (4)$$

In this scenario, the hidden unit centers can be considered as representative feature vectors and hidden unit activations can be interpreted as the posterior probabilities of the presence of the corresponding features in the input space. The weights then represent the posterior probability of a class membership, given these features. The kernel function for each mixture component j is modeled as:

$$p(\mathbf{x}|j) = \frac{1}{(2\pi)^{d/2}\sigma_j^d} e^{-\frac{||\mathbf{x}-\boldsymbol{\mu}_j||^2}{2\sigma_j^2}}, \quad (5)$$

where $\boldsymbol{\mu}$ represents the center of the j^{th} kernel and d is the dimensionality of $\mathbf{x}$.

The parameters of the mixture model can be obtained using an iterative process called the Expectation Maximization (EM) algorithm [2] The Expectation step involves the computation of the conditional probability of the j^{th} kernel given the n^{th} sample:

$$P(j|\mathbf{x}^n) = \frac{P(\mathbf{x}^n|j)P(j)}{P(\mathbf{x}^n)}. \quad (6)$$

The Maximization step involves the computation of the new values for the model parameters $\boldsymbol{\mu}_j$, σ_j^2 and $P(j)$

$$\boldsymbol{\mu}_j^{new} = \frac{\sum_n P(j|\mathbf{x}^n)\mathbf{x}}{\sum_n P(j|\mathbf{x}^n)} \quad (7)$$

$$(\sigma_j^{new})^2 = \frac{1}{d}\frac{\sum_n P(j|\mathbf{x}^n)||\mathbf{x}^n - \boldsymbol{\mu}_j^{new}||^2}{\sum_n P(j|\mathbf{x}^n)} \quad (8)$$

$$P(j)^{new} = \frac{1}{N}\sum_n P(j|\mathbf{x}^n) \quad (9)$$

where N is the total number of samples. The Expectation and Maximization steps are alternately performed till the solution converges. This EM algorithm to find the hidden unit centers and widths constitutes the first stage training. The second stage training consists of determining the weights w_{kj} using a one-step pseudo-inverse computation.

B. Theory for Accommodating Additional Features

Now say, additional features $\mathbf{x}_a$ need to be incorporated into the classification process to enhance it. One way of doing so would be to train another network from scratch such that during the first stage training (determining basis function parameters), the completed data samples consisting of previous features and augmented with the new features, are used for the mixture modeling process. Alternatively one can develop a method whereby a separate mixture model is created for inputs consisting of the new features only. This model can then be combined with the hidden units of the existing classifier to give the new hidden unit parameters for the enhanced classifier. Let us assume that a set of kernels S_a is used for representing the unconditional density of $\mathbf{x}_a$ using mixture modeling:

$$p(\mathbf{x}_a) = \sum_{i \in S_a} p(\mathbf{x}_a|i)P(i) \quad (10)$$

where

$$p(\mathbf{x}_a|i) = \frac{1}{(2\pi)^{c/2}\sigma_i^c} e^{-\frac{||\mathbf{x}_a-\boldsymbol{\mu}_i||^2}{2\sigma_i^2}}, \quad (11)$$

and c is the dimensionality of $\mathbf{x}_a$. Then,

$$P(C_k|\mathbf{x}, \mathbf{x}_a) = \sum_{l \in S_{joint}} w_{lk} P(l|\mathbf{x}, \mathbf{x}_a) \quad (12)$$

where S_{joint} is the set of kernels used to approximate the unconditional density $p(\mathbf{x}, \mathbf{x}_a)$, i.e.

$$p(\mathbf{x}, \mathbf{x}_a) = \sum_{l \in S_{joint}} p(\mathbf{x}, \mathbf{x}_a|l)P(l). \quad (13)$$

In the case where $\mathbf{x}$ and $\mathbf{x}_a$ are independent, we have:

$$p(\mathbf{x}, \mathbf{x}_a) = p(\mathbf{x})p(\mathbf{x}_a) \quad (14)$$

$$= \sum_{j \in S}\sum_{i \in S_a} p(\mathbf{x}|j)p(\mathbf{x}_a|i)P(j)P(i) \quad (15)$$

where Eq. 1 and Eq. 10 were substituted for $p(\mathbf{x})$ and $p(\mathbf{x}_a)$ and S represents the set of kernels used in the mixture modeling of $p(\mathbf{x})$. Comparing Eq. 13 and Eq. 15, we have

$$S_{joint} = S \bigotimes S_a. \quad (16)$$

Therefore,

$$p(\mathbf{x}, \mathbf{x}_a|l) = p(\mathbf{x}|j)p(\mathbf{x}_a|i) \quad (17)$$

$$= \frac{e^{-\frac{1}{2}\left(\frac{||\mathbf{x}-\boldsymbol{\mu}_j||^2}{\sigma_j^2} + \frac{||\mathbf{x}_a-\boldsymbol{\mu}_i||^2}{\sigma_i^2}\right)}}{(2\pi)^{(d+c)/2}\sigma_j^d\sigma_i^c} \quad (18)$$

$$P(l) = P(j)P(i). \quad (19)$$

Thus the kernels of set S_{joint} are formed by combining kernels from the two mixture densities $p(\mathbf{x})$ and $p(\mathbf{x}_a)$ and we can avoid having to perform EM algorithm on the completed data set thereby reducing the training time. The advantage becomes pronounced when the dimensionality of $\mathbf{x}$ is much larger than that of $\mathbf{x}_a$.

In practice, there may not be a need to include all the kernels of the set S_{joint}. Kernels with low prior probabilities $P(l)$ may be eliminated.

Now consider the case where $\mathbf{x}$ and $\mathbf{x}_a$ are dependent. Again the joint density can be approximated as

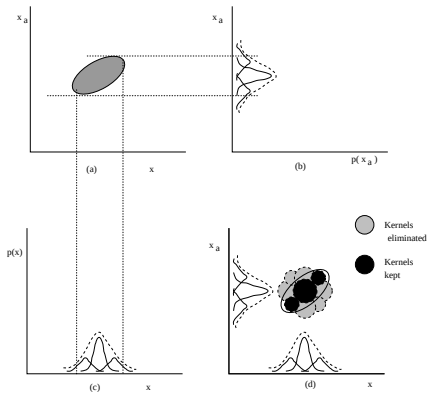

Fig. 1. A dependent distribution. x and x_e are dependent variables.

$$p(\mathbf{x}, \mathbf{x}_a) \approx \sum_{l \in S_{joint}} p(\mathbf{x}, \mathbf{x}_a | l) P(l). \quad (20)$$

with

$$p(\mathbf{x}, \mathbf{x}_a | l = i, j) = \frac{e^{-\frac{1}{2}\left(\frac{||\mathbf{x}-\boldsymbol{\mu}_j||^2}{\sigma_j^2} + \frac{||\mathbf{x}_a-\boldsymbol{\mu}_i||^2}{\sigma_i^2}\right)}}{(2\pi)^{(d+c)/2}\sigma_j^d \sigma_i^c} \quad (21)$$

However in this case $P(l) \neq P(j)P(i)$ and needs to be computed. The prior probabilities can be directly computed using Eq. 9, where $P(l|\mathbf{x}^n, \mathbf{x}_a^n)$ is given by:

$$P(l|\mathbf{x}^n, \mathbf{x}_a^n) = \frac{p(\mathbf{x}^n, \mathbf{x}_a^n | l)}{\sum_{k \in S_{joint}} p(\mathbf{x}^n, \mathbf{x}_a^n | k)}. \quad (22)$$

Consider the example shown in Figure 1. To facilitate the illustration $\mathbf{x}$ and $\mathbf{x}_a$ represented as one dimensional x and x_a respectively. The joint probability density is shown by the shaded region in Figure 1a and is assumed to be Gaussian. If the samples are projected along the two dimensions and separate mixture models are used for modeling the densities along each of the dimensions, we get models as shown in Figures 1b and 1c. Figure 1d shows the combination of the kernels from the two mixture densities. Some of the kernels will be eliminated since they will have low values for $P(l|\mathbf{x}^n, \mathbf{x}_a^n)$ and therefore will have low prior probabilities $P(l)$. This procedure is akin to the CLIQUE clustering technique recently proposed in the data mining literature [1], which is very efficient and easily implementable.

C. Enhancement Algorithm

Consider a RBF network (Net_K) that has been trained to classify samples $\mathbf{x}$, each having K features $x_1 \ldots x_K$. Suppose, we need to enhance the classification by accommodating new features $K+1 \ldots L$, given we have additional samples $\mathbf{x}'$, each being a vector of size L. We first train a new network(Net_{KL}) using only features $K+1 \ldots L$ of the new samples $\mathbf{x}'$. Candidate units for the enhanced network(Net_L) are then obtained by combining the hidden units from both the above networks(Net_K and Net_{KL}) via a cross product. The variance σ_L^2 of the candidate units is approximated as: $\sigma_L^2 = \left(\sigma_K\sqrt{\frac{K}{L}} + \sigma_{KL}\sqrt{\frac{L-K}{L}}\right)^2$, where σ_K^2 and σ_{KL}^2 are the variances of the hidden units of the networks. Prior probabilities of these potential units are then computed using Eq. 9 and Eq. 22. These units are then sorted in descending order of their prior probabilities over the samples $\mathbf{x}'$. Potential networks are formed by choosing increasing numbers of candidate units from the sorted list and training their second layer weights using the samples $\mathbf{x}'$. To avoid inspection of a large number of networks, the number of hidden units that can be used for a network is restricted. Among the potential networks the one with the least validation error is chosen as the enhanced network ('Validation set method'). Another method ('Probability fraction method') of forming the new network is to choose the first m units from the sorted candidate pool such that the sum of the prior probabilities of these m units exceeds some fraction, f. Such a method avoids the need to inspect many networks with increasing numbers of hidden units with a validation set and hence is more efficient. However the performance is dependent on the choice of f. From the following experiments, it can be seen that the above approaches yield a lower training time than having to train the new network from scratch, and also provides lower generalization error.

D. Experiments

In this section we present the results obtained by applying the knowledge enhancement technique discussed in the previous section to an automatic target recognition problem. All the experiments were performed on a dual PentiumII 300MHz workstation with 512MB RAM.

D.1 Automatic Target Recognition

This problem involves the recognition of 6 different types of military vehicles based on 47 features which were obtained by processing FLIR images. The total number of samples is 398. In all the experiments, 300 samples were used for training/validation and the remaining 98 were used for testing, unless mentioned otherwise. The 300 samples were employed via 10 fold cross-validation for NET1 through NET9, and via 5 fold cross-validation for networks NET10 through NET14.

In the first experiment, a RBF network(NET1) was trained on 300 randomly chosen samples, using a set of 30 features(out of 47). Now, say we need to include the remaining 17 features to enhance the classification performance. We tested two approaches. In the first approach, a new RBF network(NET2) was trained from scratch using the same 300 samples and all the 47 fea-

tures. In the other approach, the knowledge enhancement techniques, viz. the 'validation set' method and the 'probability fraction' method, as explained in the previous section were used. Firstly the 300 samples were used to train a network(NET3) using the 17 features only. The hidden unit centers from both the networks, NET1 and NET3, were combined via a cross product to give candidate hidden units for a new enhanced network. These RBFs were then sorted in the descending order of their prior probabilities. Using the 'validation set' method, the actual hidden units for an enhanced network(NET4) were obtained by choosing the first k [1] RBFs which provide the least validation set error, after being trained for the second layer weights using the training set. NET5 was formed using the 'probability fraction' method where the first m units were chosen from the same candidate pool, such that their combined prior probability was less than 0.4.

Similarly another RBF network(NET6) was trained using 40 features and combined with a RBF network(NET7),trained on the remaining 7 features. NET8 was obtained using the 'validation set' combination method and NET9 was obtained using the 'probability fraction' method, where the fraction f was chosen as 0.4.

In yet another experiment, a network(NET10) was trained using 40 features and 250 samples(200 for training and 50 for validation). To accommodate the remaining 7 features, another network(NET11) was trained using a different set of 50 samples where 40 were used for training and 10 were used for validation. The same 50 samples were then used for combining NET10 and NET11 to form the enhanced network, NET12, using 'validation set method'. NET13 was obtained using 'probability fraction' method where the fraction f was again chosen as 0.4. NET12 and NET13 were compared with NET14, which was trained using these 50 samples and all the 47 features together.

The comparison of all the above networks with respect to their training times and generalization errors is provided in Table I[2].

E. Discussion

One of the main advantages of the enhanced algorithm is that the time taken to train the network using this method is smaller than that required to train the network from scratch. As can be seen in Table I, the training times for networks NET4, NET5, NET8 and NET9 are less than the training time for NET2. Also, as the fraction of the number of features needed to be accommodated compared to the number of existing features reduces, the difference in the training time becomes more pronounced.

The performance of the networks trained using the enhancement techniques are better than the usual training method. The test set performances of NET4, NET5 and NET8 are better than that of NET2. Such an improvement in the performance is also due to the fact that large number of hidden units could be used for the networks trained using the enhancement techniques. This could not be done with the normal training technique. Not only does the time taken to train increase drastically, the technique also avoids the increasing unreliability of finding the psuedo-inverse solution for the weight matrix with increase in number of units.

Another major advantage is that the performance of the enhancement technique is better than the normal algorithm when the number of samples with completed data(containing values for additional features) is small. NET12 and NET13 not only require a smaller training time than NET14, they also perform better.

III. Knowledge Reuse in RBFNs

In this section we propose techniques for knowledge reuse in the context of RBF networks. In general, the method used for recall in RBF networks in the presence of missing variables can be combined with the method for adding features to provide a unified framework of direct transfer of knowledge across RBF networks.

A. Elimination of Features

Say a network has been trained to distinguish between different kinds of passenger cars based on features such as mileage, engine type, number of doors, etc. If another network needs to be trained which distinguishes different sports cars, the new network may be initialized using the features common between passenger and sports cars. This can be done by using the technique followed for performing recall in the presence of missing features [9]. If a network has been trained using EM algorithm for deciding parameters for the hidden units, the new network can be initialized by projecting the Gaussian hidden units of the existing network along the common dimensions of the two problems. In the above example mileage and engine type may be common features for the two classification problems, whereas number of doors is not a distinguishing feature among sports cars. Thus the Gaussians used as hidden units in the network for classifying passenger cars can be projected along the dimensions representing mileage and engine type. This projection is taken by simply neglecting the dimension associated with number of doors.

[1] For all the experiments involving 'validation set' method, the maximum number of hidden units that were tested was limited to the larger of 100 and the cardinality of the cross product.

[2] The training times of the combined networks consist of the time taken to train the network on the new features and time taken to combine. The time taken to combine is given in brackets. e.g. Time taken to train NET4 = Time taken to train NET3 + Time to combine NET1 and NET3. Also the time shown here is the time taken to train the networks on the number of hidden units shown in the previous column.

B. Addition of Features

Now say an additional feature such as wheel-base needs to be incorporated in the classification of sports cars and was not used for passenger cars. This accommodation of new features can be done using the method discussed in the context of knowledge enhancement in the previous chapter, i.e. train another network using only the new feature and combine this network with that obtained after elimination to give the consolidated network. Such a training technique may not only be faster than training a new network from scratch but may also lead to better performance as shown in the following experiments.

C. Experiments

The automatic target recognition dataset introduced in the previous section is used to demonstrate the efficacy of the reuse algorithm over the normal training technique. This dataset consists of 6 different kinds of military vehicles, 2 each of tanks, trucks and carriers. The 47 input features include 23 Zernike moments, 7 standard moments and 17 other features such as grey-level information, width, height, etc. The dataset has 281 samples of trucks and tanks and 117 samples of carriers.

In the first experiment, an RBF network(TRNET1) was trained to distinguish the 4 different kinds of trucks and tanks. Only 23 Zernike moments and 7 standard moments were used as input features for this problem. 250 samples were used for training TRNET1 and the remaining 31 samples were used for testing. We now wanted to train a RBF network to differentiate the 2 different kinds of carriers using the 23 Zernike moments and the 17 other features not used for training TRNET1. Since the 23 features constituting the Zernike moments were common between the two problems, using the reuse algorithm, the hidden units of TRNET1 were projected along these 23 dimensions to give network TRNET2. A new network, TRNET3 was then trained on 80 samples of carriers on the remaining 17 features to distinguish the two types of carriers. TRNET2 and TRNET3 were then combined using the 'probability fraction' method discussed in the previous chapter to form the final network, TRNET4. The fraction f was chosen as 0.4, the same value used for the experiments in the previous chapter. The training time and performance of this network was compared with another RBF network(TRNET5), trained from scratch(using the traditional approach) to distinguish carriers using the 40 features. A 10 fold cross validation was performed on networks TRNET3, TRNET4 and TRNET5.

D. Discussion

As can be seen from table II , the time taken to train TRNET4 is smaller than that of TRNET4. Also the performance of the network is better than its traditional counterpart. The advantages become more pronounced as the number of training samples and common features increase and the fraction of new features needed to be accommodated reduces.

IV. Conclusion

Knowledge enhancement and reuse are becoming increasingly important especially in the context of rapidly building a series of models for analyzing large collections of data being continuously acquired from remote sensing, satellite imagery, customer retailing, web logs etc. In this paper, we presented a novel enhancement algorithm for accommodating new features within a trained network. Such a technique is more efficient and performs considerably better than training a new network trained from scratch. We also presented a knowledge reuse framework that can be used to transfer knowledge across Radial Basis Function Networks.

V. Acknowledgements

This work was supported by NSF grant ECS-9900353, and by a research grant from Intel Corp.

References

[1] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *ACM SIGMOD*, pages 94–105, 1998.

[2] S.-I. Amari. The EM algorithm and information geometry in neural network learning. *Neural Computation*, 7:13–18, 1995.

[3] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, New York, 1995.

[4] Kurt D. Bollacker and Joydeep Ghosh. Effective supra-classifiers for knowledge base construction. *Pattern Recognition Letters*, 20(11-13):1347–52, November 1999.

[5] Kurt D. Bollacker and Joydeep Ghosh. Knowledge transfer mechanisms for characterizing image datasets. In S.K. Pal, A. Ghosh, and M.K. Kundu, editors, *Soft Computing and Image Processing.* Physica-Verlag, Heidelberg, 2000.

[6] S. Kumar, J. Ghosh, and M. M. Crawford. Best-bases feature extraction algorithms for classification of hyperspectral data. *IEEE Trans. Geoscience and Remote Sensing*, 39(7):1368–79, 2001.

[7] J. Park and I.W. Sandberg. Universal approximation using radial basis function networks. *Neural Computation*, 3(2):246–257, Summer 1991.

[8] S. Thrun and L.Y. Pratt. *Learning To Learn*. Kluwer Academic, Norwell, MA, 1997.

[9] V. Tresp, S. Ahmad, and R. Neuneier. Training neural networks with deficient data. In *Advances in Neural Information Processing Systems-6*, pages 128–135, 1994.

[10] V. Tresp, R. Neuneier, and S. Ahmad. Efficient methods for dealing with missing data in supervised learning. In D.S. Touretzky G. Tesauro and T.K. Leen, editors, *Advances in Neural Information Processing Systems-7*, pages 687–696, 1995.

Algo	Features	Train Set	Hidden Units	Train Time (sec.)	Training Rate (% correct)	Test Rate (% correct)
NET1	30	300	14.5±0.53	4.04±0.12	71.13±2.63	67.04±2.85
NET2	47	300	13.1±1.19	4.51±0.03	70.7±1.53	66.73±2.85
NET3	17	300	14.2±0.42	3.46±0.09	69.83±1.81	66.73±3.09
NET4	**47**	**300**	**76.5 ±12.58**	**4.46(1) ±0.22**	**95.77 ±1.32**	**88.06 ±2.15**
NET5	**47**	**300**	**28.3 ±0.95**	**4.26(0.08) ±0.13**	**82.33 ±1.81**	**76.84 ±3.33**
NET6	40	300	13.1±1.52	4.15±0.38	71.27±2.25	66.53±3.9
NET7	7	300	12.2±2.25	2.74±0.43	54.83±5.55	45.1±5.52
NET8	**47**	**300**	**59.8 ±16.4**	**3.46(0.72) ±0.62**	**92.1 ±3.69**	**85.31 ±4.25**
NET9	**47**	**300**	**28.3 ±3.13**	**3.44(0.7) ±0.58**	**81.3 ±3.44**	**74.89 ±3.34**
NET10	40	250	13.8±1.64	3.65±0.35	76.47±3.39	67.14±4.17
NET11	7	50	2.6±0.89	0.49±0.03	88.87±0.77	18.57±1.68
NET12	**47**	**50**	**15 ±8.57**	**0.55(0.06) ±0.04**	**96.13 ±2.10**	**55.31 ±5.09**
NET13	**47**	**50**	**8 ±1.41**	**0.54(0.05) ±0.04**	**93.33 ±1.03**	**47.47 ±7.72**
NET14	47	50	6±2.24	0.8±0.39	92.33±2.9	44.29±6.65

TABLE I

Automatic Target Recognition Results: Train time of combined network = train time of network with new features + time taken to combine kernels(shown in brackets).

Algo	Features	Train Set	Hidden Units	Train Time (sec.)	Training Rate (% correct)	Test Rate (% correct)
TRNET1	30	250	13	3.18	75.6	70.97
TRNET3	17	80	3.2±0.42	0.64±0.03	91±5.52	92.43±6.08
TRNET4	**40**	**80**	**8.6 ±0.84**	**0.71(0.07) ±0.04**	**94.5 ±2.96**	**96.76 1.14**
TRNET5	40	80	3.6±0.69	0.79±0.05	87.13±7.62	87.03±9.52

TABLE II

REUSE IN AUTOMATIC TARGET RECOGNITION.