# Learning Constructive RBF Networks by Active Data Selection

Sang-Wook Park Byoung-Tak Zhang

Artificial Intelligence Lab (SCAI)
School of Computer Science and Engineering
Seoul National University
Seoul 151-742, Korea
{swpark, btzhang}@scai.snu.ac.kr

## Abstract

*In this paper we propose a learning method for RBF networks that select data actively and increase the number of hidden units to fit the data. In this algorithm, RBF networks iteratively select data where the current networks perform the worst, then train the networks on these data. Experiments have been performed on five real world dataset and performance was compared with the resource-allocating network (RAN). The results show that the proposed RBF networks use smaller data sizes to achieve a similar performance of RAN using total data. We also experiment with CoIL-2000 data set.*

## 1 Introduction

Recently researchers have begun to examine the use of radial basis functions (RBF)[1] for solving function approximation and pattern classification problems. RBF networks are well suited for these problems due to their simple topological structure and their ability to reveal how learning proceeds in an explicit manner [2]. In the classical approach to RBF network implementation, the basis functions are usually chosen as Gaussian and the number of hidden units is fixed a priori based on some properties of the input data. The weights of edges connecting the hidden and output units are estimated by linear least squares methods, e.g., least mean square (LMS) [3]. But it is not easy to select the number of hidden units and decide centers and widths of the hidden units. Another disadvantage with the classical approach is that it is not suitable for sequential learning where new data continuously come because we again decide the new number of hidden units and estimate the parameters of them.

To overcome these drawbacks, Platt developed an algorithm that gets data sequentially and decides the number of hidden units of network and the parameters of hidden units based on the data [4]. This algorithm is based on the idea that the number of hidden units should correspond to the complexity of the underlying function as reflected in the observed data. We call this resulting networks RAN (resource-allocating network). RAN starts with no hidden units and grows by allocating new hidden units based on observations which arrive sequentially [5].

RAN uses sequentially all data only once. However because data have redundant information, we do not need all data in learning [6, 7]. In this paper we introduce ARAN (active RAN) which shows enough performance without learning all data by selecting critical data actively.

Active learning in its most general sense refers to any form of learning wherein the learning algorithm has some degree of control over the examples on which it is trained [13]. There are many heuristics for active learning. We can choose the data which are not learned till now, which learner performs poorly at, which we have low confidence at [8]. Because deterministic algorithm has a problem with local minima, probabilistic active learning methods also are proposed [11]. For learning data, ARAN selects data which current trained neural networks perform poorly at. So we want ARAN to learn efficiently with small training data set.

This paper is organized as follows. Section 2 gives related works for RAN, and then we propose a new algorithm that selects critical data for training data and automatically increases size of RBF neural networks. Section 3 shows experimental results for preprocessed UCI real world data [9] and Coil-2000 data. In section 4, we summarize the conclusions from this study and present some idea for further study.

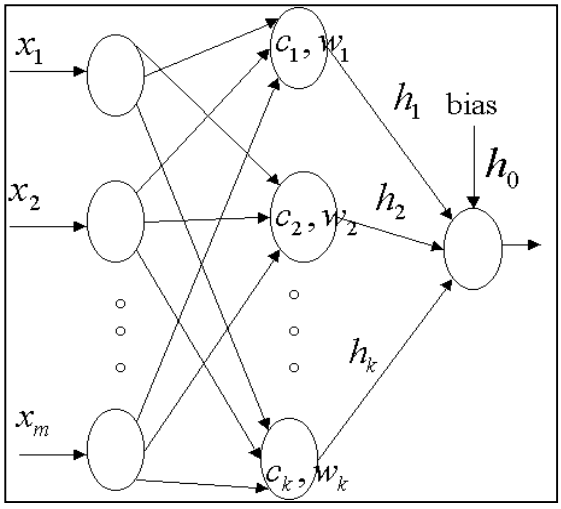
## 2 Algorithm

ARAN algorithm is based on RAN algorithm, which automatically constructs RBF networks. We add active data selection scheme to RAN. First, we review RAN algorithm, then describe ARAN algorithm.

### 2.1 RAN

RAN is a two-layer RBF network like Figure 1. The first layer consists of units that respond to only a local region of the space of input values. The second layer

aggregates outputs from these units and creates the function that approximates the input-output mapping over the entire space. [4, 10]



[Figure 1] The structure of RBF network.

The units on the first layer store a particular region in the input space. A simple function that implements a locally tuned unit is a gaussian:

$$z_j = \exp\left(-\frac{\|c_j - x\|^2}{w_j^2}\right)$$

Here, $c_j$ is the center of *j*th hidden unit, $w_j$ is the width of *j*th hidden unit, *x* is input pattern.

Each output of the network *y* is the sum of the first-layer outputs $z_j$, each weighted by the synaptic strength $h_j$ plus a constant vector $\gamma$, which does not depend on the output of the first layer:

$$y = \sum h_j z_j + \gamma$$

The network starts with no hidden unit. As the network get data, the network allocates new hidden unit if two conditions are satisfied. The first condition considers a pattern as new one if the minimal distance between the coordinate *x* and the center is greater than a determined value.

$$\|x - c_{nearest}\| > \delta(t)$$

The second condition is given by

$$\|T_p - O_p\| > \varepsilon$$

which means that new neurons will not be created for smaller corrections. Typically, $\varepsilon$ is a desired accuracy of output of networks. Errors larger than $\varepsilon$ are immediately corrected by the allocation of a new unit, while errors smaller than $\varepsilon$ are gradually repaired using gradient descent. The distance $\delta(t)$ is the scale of resolution that the network is fitting at the $t$ th input presentation. $\delta(t)$ shrinks each training epoch until it reaches $\delta_{\min}$.

If data $x$ is satisfied above two conditions, the neural networks create a new hidden unit like followings

$$h_j = T - y$$
$$w_j = \kappa\|x - c_{nearest}\|$$
$$c_j = x$$

Learning algorithm is shown in Table 1 as pseudocode.

[Table 1] Learning algorithm of RAN.

$\delta = \delta_{\max}$
$\gamma = T_0$
loop over presentations of input-output pair (x, T)
{
  evaluate output of network $y = \sum h_j z_j(x) + \gamma$
  compute error $E = T - y$
  find distance to nearest center $d = \min_j \|c_j - x\|$
  if $\|E\| > \varepsilon$ and $d > \delta$ then
  {
    allocate new unit, $c_{new} = x, h_{new} = E$
    if this is the first unit to be allocated then
      width of new unit = $k\delta$
    else
      width of new unit = $kd$
  }
  else
    perform gradient descent on $\gamma, h_j, c_j$
  if $\delta > \delta_{\min}$
    $\delta = \delta \times \exp(-1/\tau)$
}

## 2.2 Active RAN

ARAN algorithm is an enhanced algorithm of RAN. RAN uses all training data just once sequentially, but ARAN selects training data actively. In order to describe more specifically, we give some terminologies and definitions.

Candidate set$(C)$ is defined as the data set, which is not used till now by neural networks of total data for learning. Training set$(T)$ is defined as the data set, which is used for learning. So, the number of total data (*N*) is the number of data in *C* plus the number of data in *T*. In the beginning, all training data is included in *C* and *T* is empty set.

$(x_m, y_m)$ represents $m$th data, and $f(x_m)$ represents output of current neural networks when $x_m$ is an input of neural networks. Therefore, the error of $m$th datum is as follows.

$$e_m = \frac{1}{\dim(y_m)} \| y_m - f(x_m) \|$$

Here, $dim(y_m)$ is dimension of $y_m$.

In each step, after errors of all data in $C$ are calculated, we select $\lambda$ data with maximum error as new training data.

Let $x^{max}$ be data that has maximum error $(e^{\max})$ of $\lambda$ data. So if following two conditions are satisfied, current RBF networks create a new hidden unit.

$$\| x^{\max} - c_{nearest} \| > \delta(t)$$

$$e^{\max} > \varepsilon$$

If above conditions are satisfied, the neural networks add a new $k$th hidden unit like followings.

$$h_k = y^{\max} - f(x^{\max})$$

$$w_k = \kappa \| x^{\max} - c_{nearest} \|$$

$$c_k = x^{\max}$$

This method makes the hidden unit for the difficult data, so it is possible to learn the data. If above conditions are not satisfied, we just use $\lambda$ data for LMS.

[Table 2] Learning algorithm of ARAN

initialize random neural networks with no hidden unit.
repeat until end condition is satisfied
{
  select $\lambda$ data which have maximum errors by current neural networks in $C$.
  if $\| x^{\max} - c_{nearest} \| > \delta(t)$ and $e^{\max} > \varepsilon$ then

  {
    allocate new hidden unit,
    $c_{new} = x^{\max}$
    $h_{new} = y^{\max} - f(x^{\max})$
    $w_{new} = \kappa \| x^{\max} - c_{nearest} \|$
  }
  else
    perform gradient descent on $\gamma, h_j, c_j$ with selected data
  add selected data to $T$ and subtract selected data to $C$.
  perform gradient descent on the neural networks with $T$.
}

After that, selected $\lambda$ data are added to $T$ and the RBF networks are learned with all data in $T$ by LMS. Simple ARAN algorithm is shown in Table 2

## 3 Experiments

In this section, we describe two sets of experiments demonstrating ARAN performance. In the first set of experiments, we applied ARAN to learning 5 UCI dataset for classification problems. In the second set of experiments, we applied ARAN to learning CoIL-2000 dataset.

### 3.1 UCI data

We experimented with five UCI data of preprocessed 12 classification problems. We compared ARAN with RAN on theses five benchmark dataset. Only horse problem of five problems is 3 classes problem and extra problems are all 2 classes problems. We used 2/3 of all data as training data, and 1/3 as test data for respective problems. Parameters for each problem are tuned properly through many experiments. Especially, $\lambda$ value in ARAN was decided for not bigger value than the number of all training data divided by 100.

Table 3 shows mean values for each problem after 10 times experiments with RAN and ARAN. For each problem, left column shows results of RAN, right column shows results of ARAN. The result of RAN shows the number of hidden units, training performance and generalization performance of final RBF networks after RAN used all data. On the other hand the result of ARAN shows the number of training data, the number of hidden units and performances at the point of good performance after learning by active data selection.

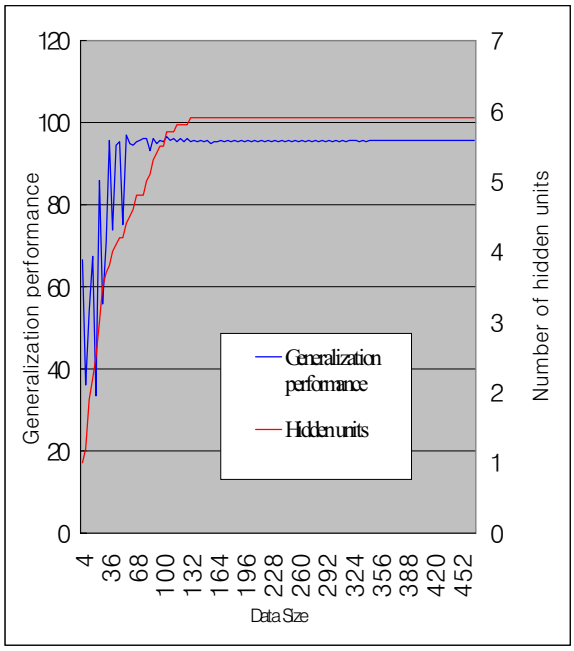
We know that the RBF networks have enough performance without learning by total data for all five problems in table 3. Especially for Cancer data, ARAN has good performance with only 15% data of all. By the way, the result of ARAN shows that training performance is lower than generalization performance. The reason is that ARAN first uses difficult data for training data, and measures training performance for the data. Or, generalization error is higher than training error in ARAN because ARAN measures error about relative difficult data as training error. If ARAN uses all data in the end, training performance is higher than generalization performance.

Figure 2 is a result of ARAN experiments for Cancer data. It shows relations between training data size and generalization performance, size of hidden units. Because we take mean value of 10 times experiments, the size of hidden units is not increased as integer. In the beginning as training data increase, generalization error and size of hidden unit increases but after some times both of them change little. This shows the RBF networks do not need all data to solve Cancer problem, and find the number of hidden units properly. So if we control the parameters of ARAN,

[Table 3] Results of RAN and ARAN for UCI data.

| Task | Cancer | | Diabetes | | Heart | | Card | | Horse | |
|---|---|---|---|---|---|---|---|---|---|---|
| | RAN | ARAN | RAN | ARAN | RAN | ARAN | RAN | ARAN | RAN | ARAN |
| Data used in training | 466 | 72.4 | 512 | 300 | 614 | 487 | 460 | 290 | 243 | 112 |
| Number of hidden units | 5.0 | 4.7 | 10.4 | 8.5 | 5.0 | 6.0 | 11.1 | 10.2 | 5.4 | 7.1 |
| Train (%) performance | 97.1 | 83.7 | 76.4 | 64.1 | 83.1 | 79.0 | 81.1 | 83.9 | 59.7 | 63.0 |
| Test (%) performance | 97.0 | 97.0 | 74.5 | 74.9 | 79.4 | 80.4 | 85.1 | 88.3 | 70.8 | 72.1 |

the RBF networks can learn by minimal training data and we can find proper size of the hidden units.




[Figure 2] Result of ARAN experiment for cancer problem.

### 3.2 CoIL-2000 data

CoIL–2000 data are for CoIL challenge 2000. This dataset is owned and supplied by the Dutch datamining company Sentient Machine Research, and is based on real world business data. This dataset is information about potential customers of caravan insurance. Information about customers consists of 86 variables includes product usage data and socio-demographic data derived from zip area codes. The training dataset contains 5822 descriptions of customers, including the information of whether or not they have a caravan insurance policy. A test set contains 4000 customers. In CoIL-2000 challenge, participants have to predict whether a customer is interested in a caravan insurance policy from other data about the customers [12].

In training dataset, only 348 data are positive. It is about 6%. Test dataset are similar to training dataset. Only 238 data of whole 4000 test data are positive. It is difficult to learn the data because this data set is very unbalanced. But participants of CoIL challenge 2000 do not have to predict whether customers have the caravan insurance or not exactly. The problem is to find the subset of customers with a probability of having a caravan insurance policy above some boundary probability. The boundary depends on the costs and benefits such as of the costs of mailing and benefit of selling insurance policies. To approximate this problem, participants have to find the set of 800 customers in the test set that contains the most caravan policy owners.

Table 4 shows results of CoIL challenge 2000. Best case selection can choose all 238 caravan policy owners in 800 selected customers, and Random selection can choose 42. The winner of this competition found 121 customers of 238 total policy owners in 800 selected customers. The second found 115 customers, the third found 112 customers.

We experimented for this data set with ARAN and RAN. First, we used all attributes in data set. In the second experiment, we used 28 attributes that had different statistical property between positive examples and negative examples. In third experiment, we used only 4 important attributes which consist of car policy, fire policy, trailer policy and education level. Table 5 shows results. For each case we performed many experiments with various parameters, then we selected best results in table 5.

[Table 4] Result of CoIL Challenge 2000.

| Case | Result (caravan policy owners in 800 selected customers) |
|---|---|
| Best case selection | 238 |
| Random selection | 42 |
| The first | 121 |
| The second | 115 |
| The third | 112 |

[Table 5] Result of ARAN for 3 different feature selection data.

| Attributes used | Result (caravan policy owners in 800 selected customers) | |
|---|---|---|
| | ARAN | RAN |
| 85 | 104 | 100 |
| 28 | 97 | 95 |
| 4 | 115 | 109 |

The parameters for the first experiments chosen are $\delta_{\max} = 6, \delta_{\min} = 3, \varepsilon = 0.3, \lambda = 1$, learning rate=0.01. ARAN use only 537 training data of total training data 5822, and number of hidden units was 1. As ARAN used more data over 537, its performance became worse.

The parameters for the second experiments chosen are $\delta_{\max} = 3, \delta_{\min} = 0.1, \varepsilon = 0.2, \lambda = 50$, learning rate= 0.05. ARAN used only 150 training data for learning, and number of hidden units was 1.

The parameters for the third experiments chosen are $\delta_{\max} = 2, \delta_{\min} = 1, \varepsilon = 0.3, \lambda = 1$, learning rate=0.1. ARAN used only 150 training data, and number of hidden units was 1. ARAN continued to learn till it used all data, its performance decreased to 102 from 115.

## 4 Conclusions

In this paper, we proposed RBF networks that select data actively and increase size of networks in necessity. For active data selection method, data were selected on which current RBF networks perform most poorly. Figure 2 shows that even though ARAN does not use all data, its performance is similar to that of ARAN using all data. Experiments showed that for unbalanced data like CoIL challenge 2000 data, using all training data is worse than using proper number of data.

In this paper, we proposed the constructive learning algorithm that creates a hidden neuron to fit the data, but when new data are selected continuously, some existing hidden neurons could be unnecessary. In this case, we can limit overgrowth of neural networks size [2]. For the further study, we have to add pruning algorithm to ARAN, so we expect to have a good performance with smaller size.

## Acknowledgment

This research was supported by Brain Science and Engineering Research Program sponsored by Korea Ministry of Science and Technology.

## References

[1] D. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks", *Complex Systems*, Vol. 2, pp. 321-355, 1988.

[2] K. Tao, "A closer look at the radial basis function (RBF) networks," in Conf. Rec. 27th Asilomar Conf. Signals, *Systems and Computers*, pp.401-405, 1993.

[3] J. Moody and C. J. Darken, "Fast learning in network of locally-tuned processing units," *Neural Computation*, Vol. 1, pp.281-294, 1989.

[4] J. Platt, "A resource allocating network for function interpolation," *Neural Computation*, vol 3, pp. 213-225, 1991.

[5] L. Yingwei , N. Sundararajan, P. Saratchandran, "Performance Evaluation of a Sequential Minimal Radial Basis Function (RBF) Neural Network Learning Algorithm," *IEEE Transactions on Neural Networks*, Vol. 9 No.2, pp. 308-318, 1998.

[6] B.T. Zhang, "Accelerated Learning by Active Example Selection," *International Journal of Neural Systems*, 5(1), pp. 67-75, 1994.

[7] B.T. Zhang, "Learning by Incremental Selection of Critical Examples," Arbeitspapere der GMD, No 735, German National Research Center for Computer Science (GMD), St. Augustin/Bonn, 1993.

[8] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active Learning with Statistical Models," *Journal of Artificial Intelligence Research*, 4, pp. 129-145, 1996.

[9] Lutz Prechelt, "PROBEN1-A set of neural network benchmark problems and benchmarking rules," Fakultat fur Informatick, Univ. Karlsruhe, Germany, Tech. Rep. 21/94, 1994.

[10] Gustavo Deco and Jurgen Ebmeyer, "Coarse Coding Resource-Allocating Network," *Neural Computation*, Vol 5, no 1, pp. 105-114, 1993.

[11] Kenji Fukumizu, "Statistical Active Learning in Multilayer Perceptrons," *IEEE Transactions on Neural Networks*, Vol. 11 No.1, pp.17-26, 2000.

[12] http://www.dcs.napier.ac.uk/coil/challenge/.

[13] Ray Liere and Rrasad Tadepalli, "Active Learning with Committees for Text Categorization", *Proceedings of AAAI-97*, pp. 591-596, 1997.