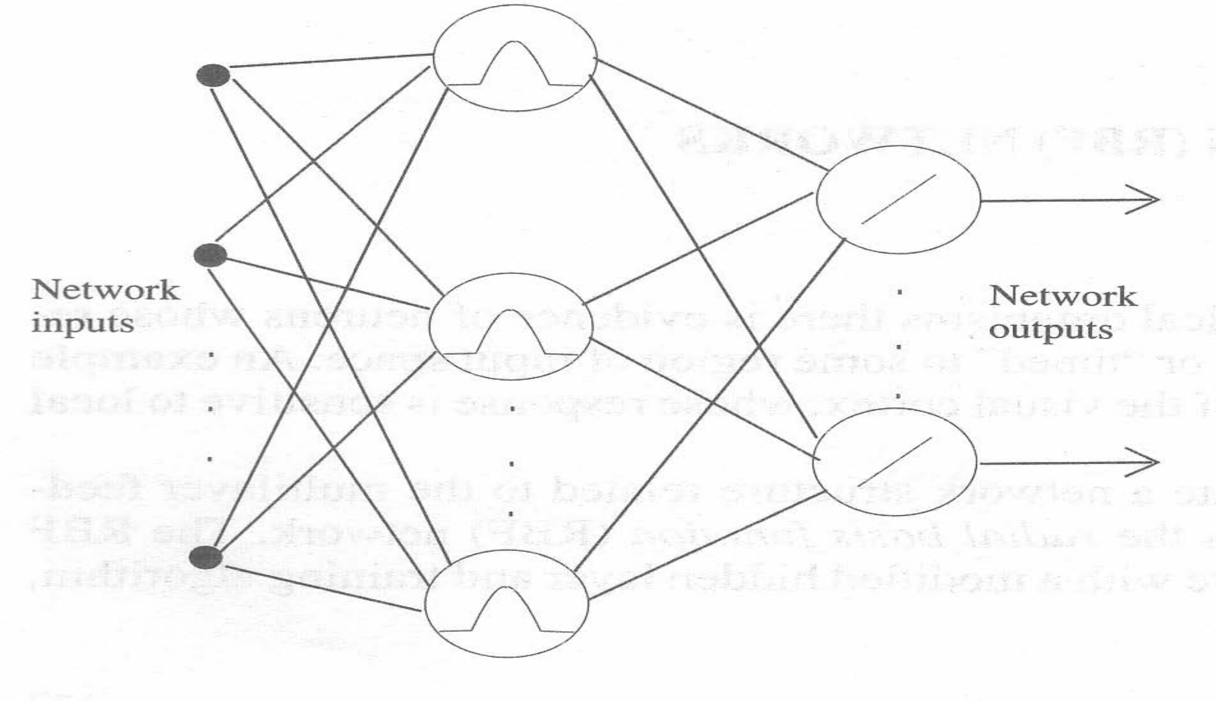
6. Radial Basis Function (RBF) Neural Networks

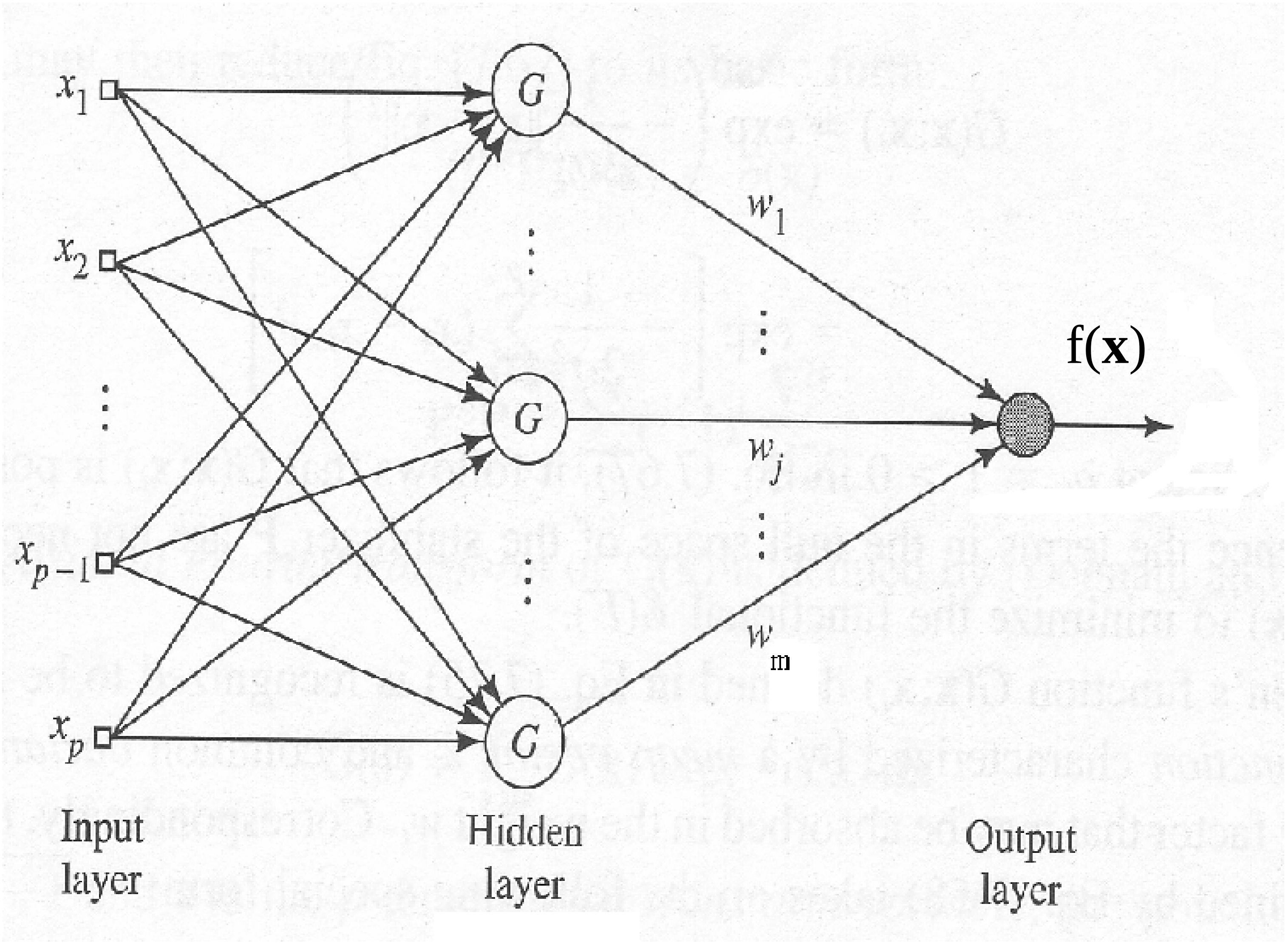
In the nervous system of biological organisms, there is evidence of neurons whose response characteristics are "local" or tuned to some region of input space. An example is the orientation sensitive cells of the visual cortex, whose response is sensitive to local region in the retina.

In this part, we will investigate a network structure related to the multi-layer feed-forward network, known as the radial basis function (RBF) neural network. The RBF neural network has a feed-forward structure with a modified hidden layer and training algorithms.

RBF Neural Network Structure

As mentioned, RBF networks emulate the behavior of certain biological networks. Basically, the hidden layer consists of the locally tuned or locally sensitive neurons, and the output layer consists of linear units. In hidden-layer neurons, the neuron response (output) is localized and decreases as function of the distance of inputs from the neuron's receptive field center.



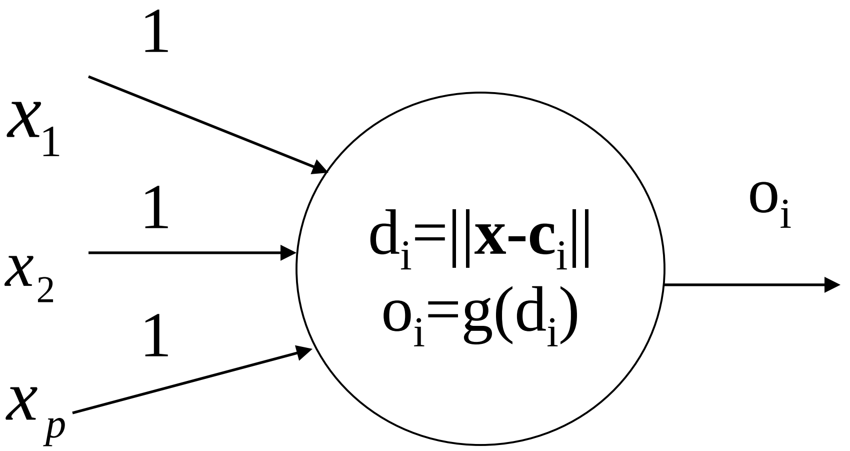


The architecture of RBF with a single output

RBF Neuron Characteristics

Input layer: The same as the input layer of feed-forward network, the input layer neurons do not perform any computation and just distribute the input variables to the hidden layer. But note the weights between input layer neurons and hidden layer neurons in the RBF network are all set to 1.

Hidden layer: a general form for hidden layer neurons of the RBF neural network may be described by:



In mathematic terms, we can describe the operation of the hidden layer neuron as:

$$d_j = \|\mathbf{c}_j - \mathbf{x}\|$$

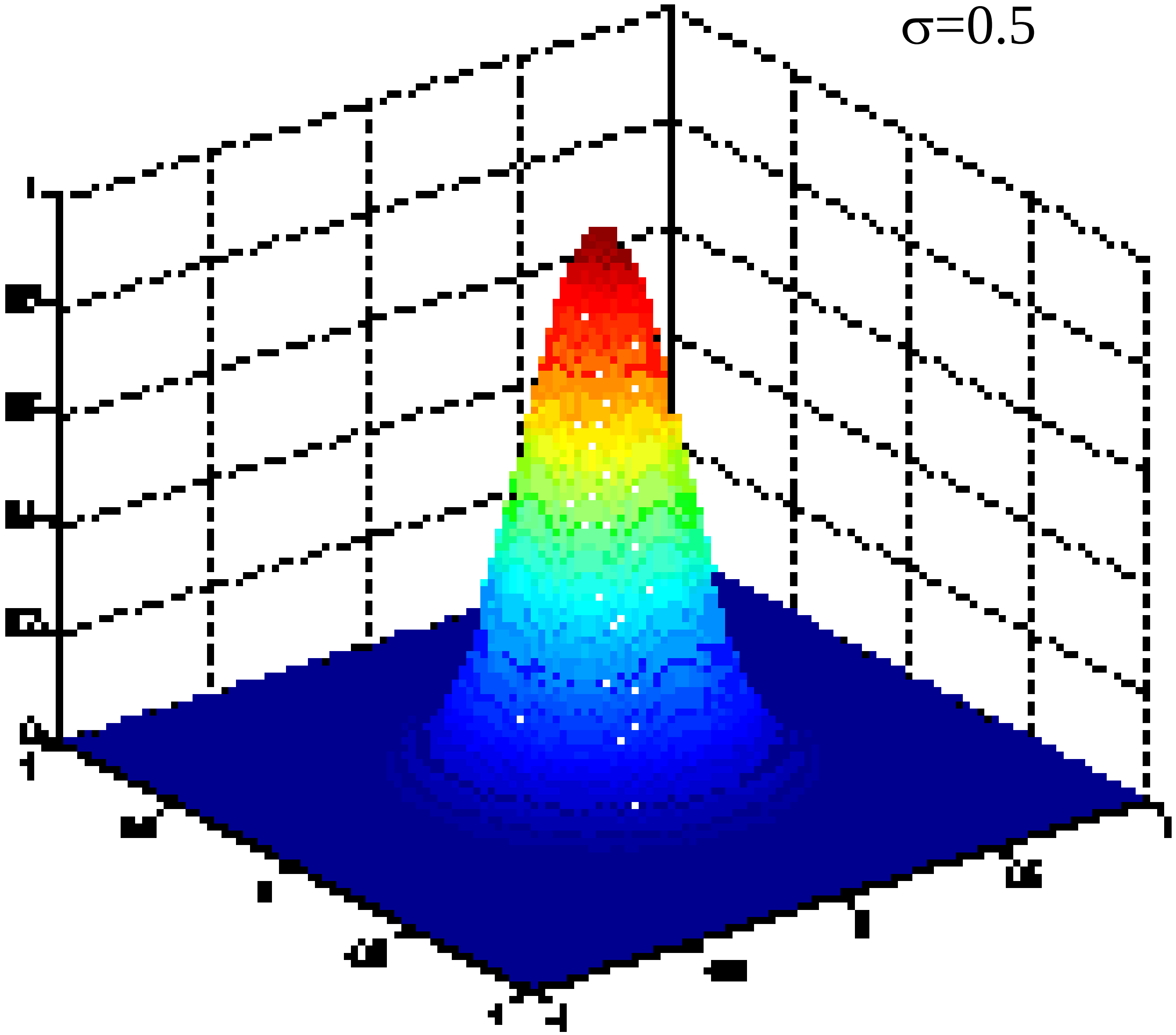
$$o_j(\mathbf{x}) = g(d_j) = g(\|\mathbf{c}_j - \mathbf{x}\|)$$

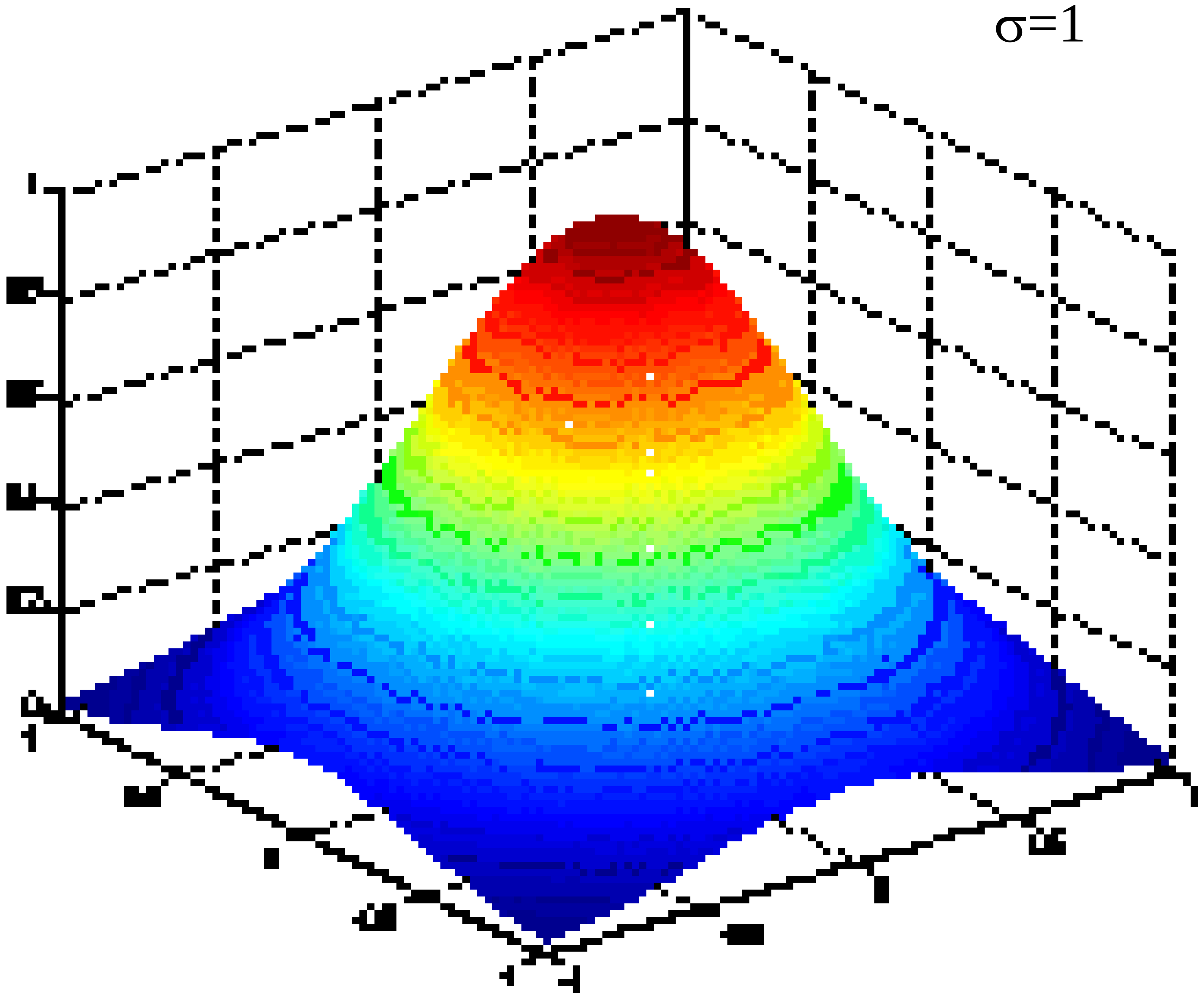
Where $\mathbf{c}_j$ is called center vector of neuron j; g is the radial basis function. Some commonly used basis functions are as follows:

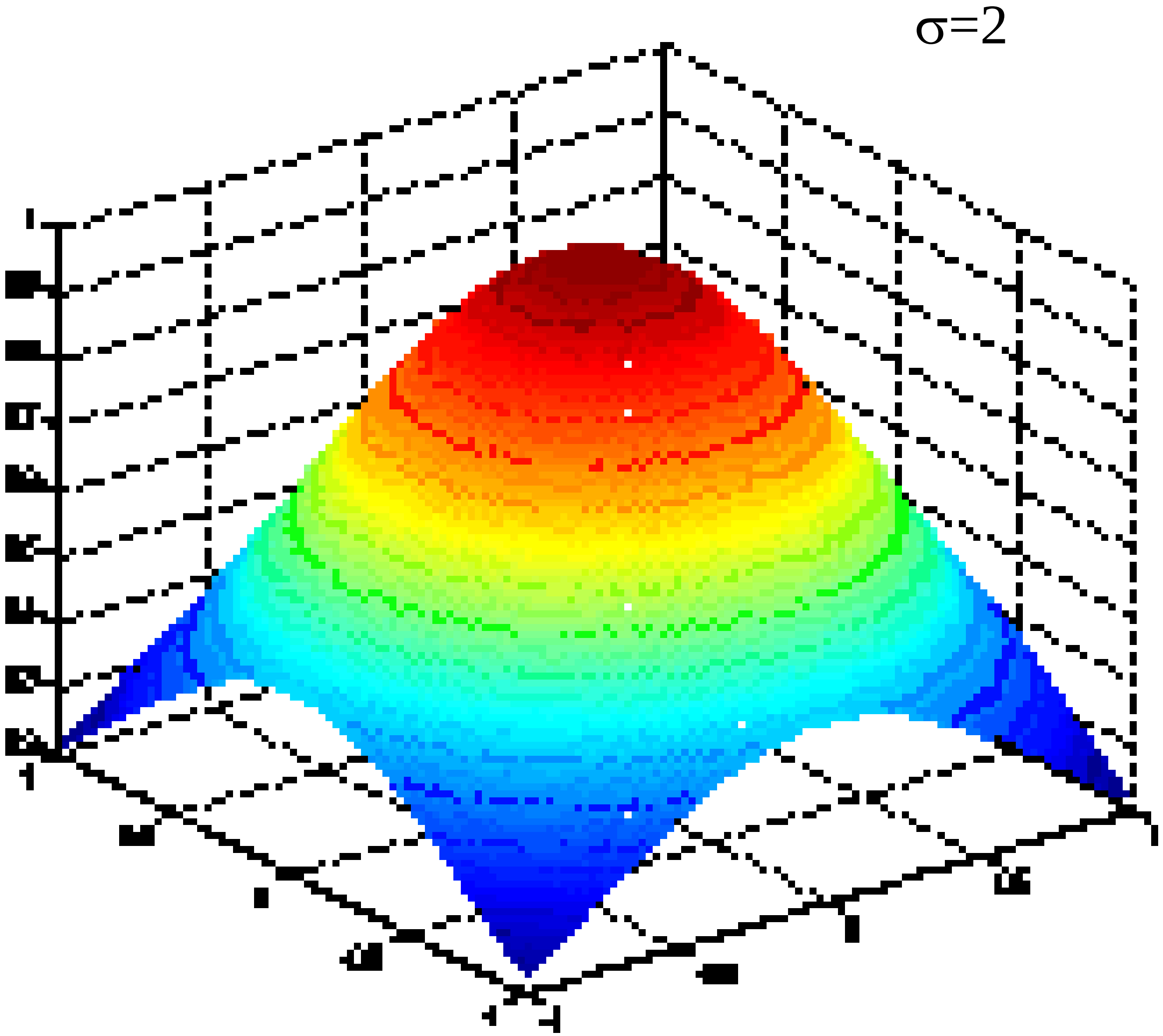
(1) The Gaussian function

$$g(d) = \exp\left(-\frac{d^2}{2\sigma^2}\right)$$

where σ is the width of the basis function. The suitable value of σ is application dependent.


σ=0.5
1
0
1
1
-1
-1
1


σ=1
1
0
1
0
-1
-1
0
1


σ=2

(2) The multi-quadratic function:

$$g(d) = \sqrt{d^2 + \sigma^2}$$

(3) The inverse multi-quadratic function

$$g(d) = 1/\sqrt{d^2 + \sigma^2}$$

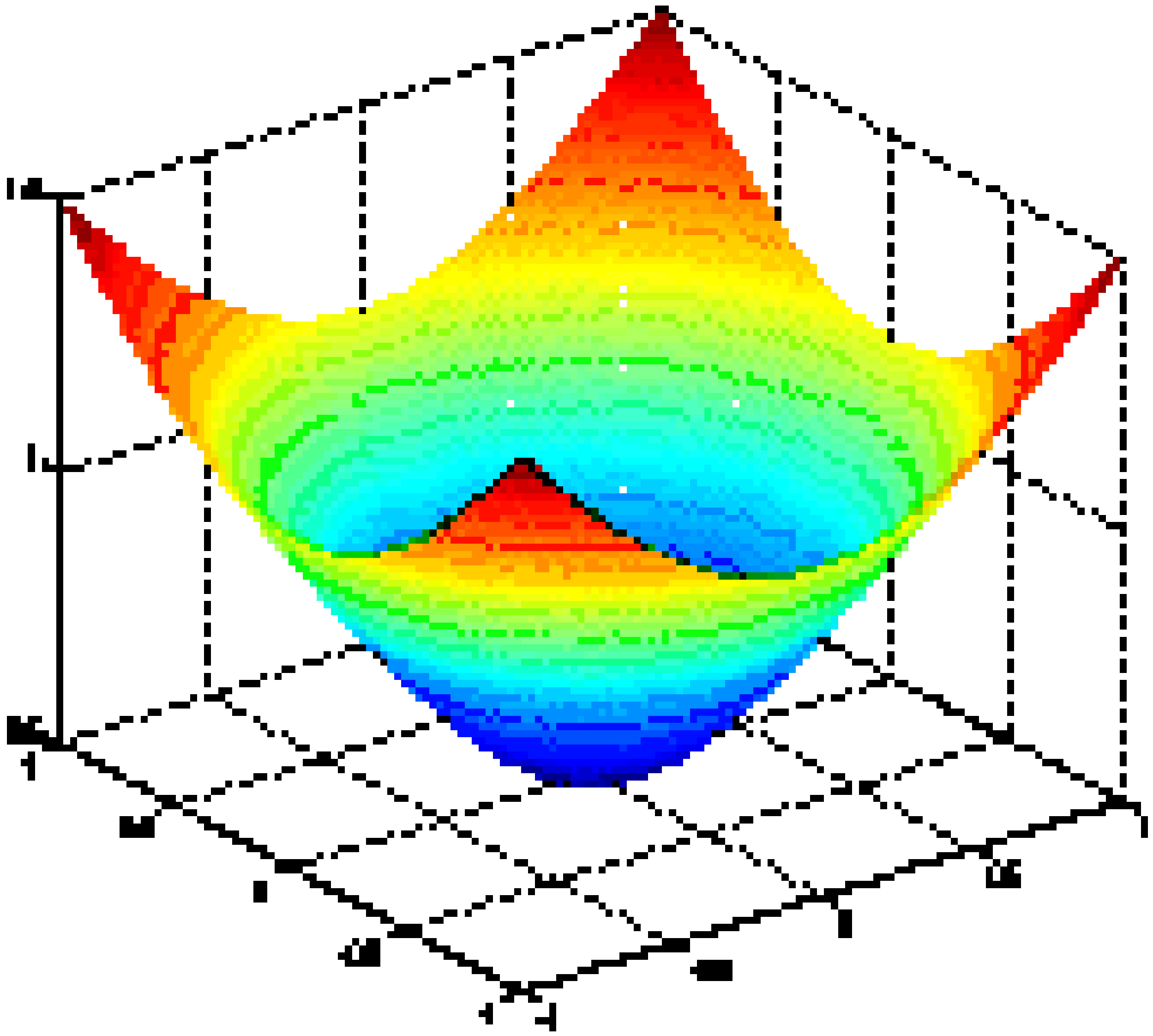
(4) The thin plate spline

$$g(d) = d^2 \log(d)$$

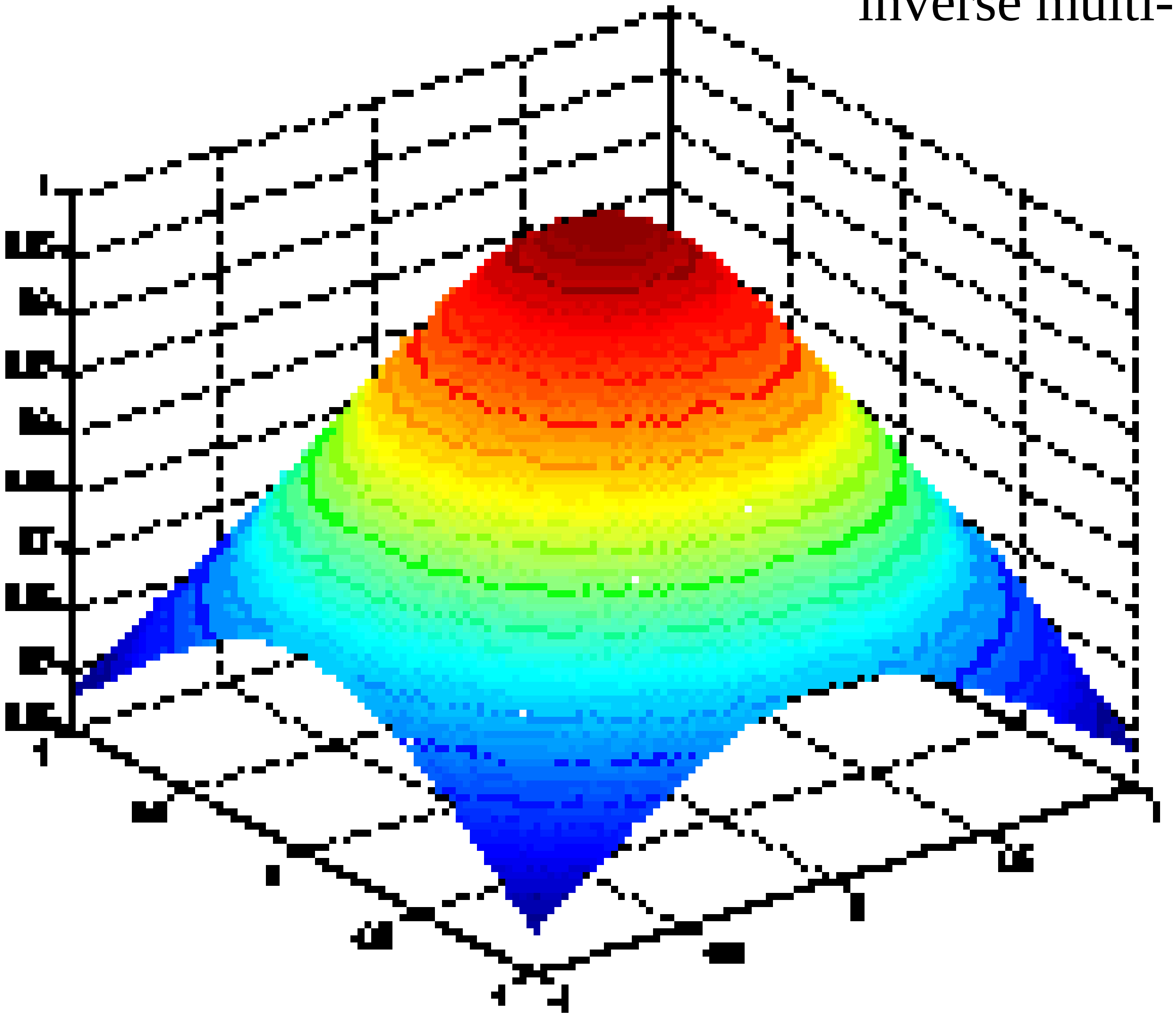
<u>The output layer</u>: The output layer neuron is a linear combiner. The output of the network is the weighted sum of the hidden layer neuron outputs:

$$f(\mathbf{x}) = \sum_{j=1}^{m} w_j o_j(\mathbf{x})$$

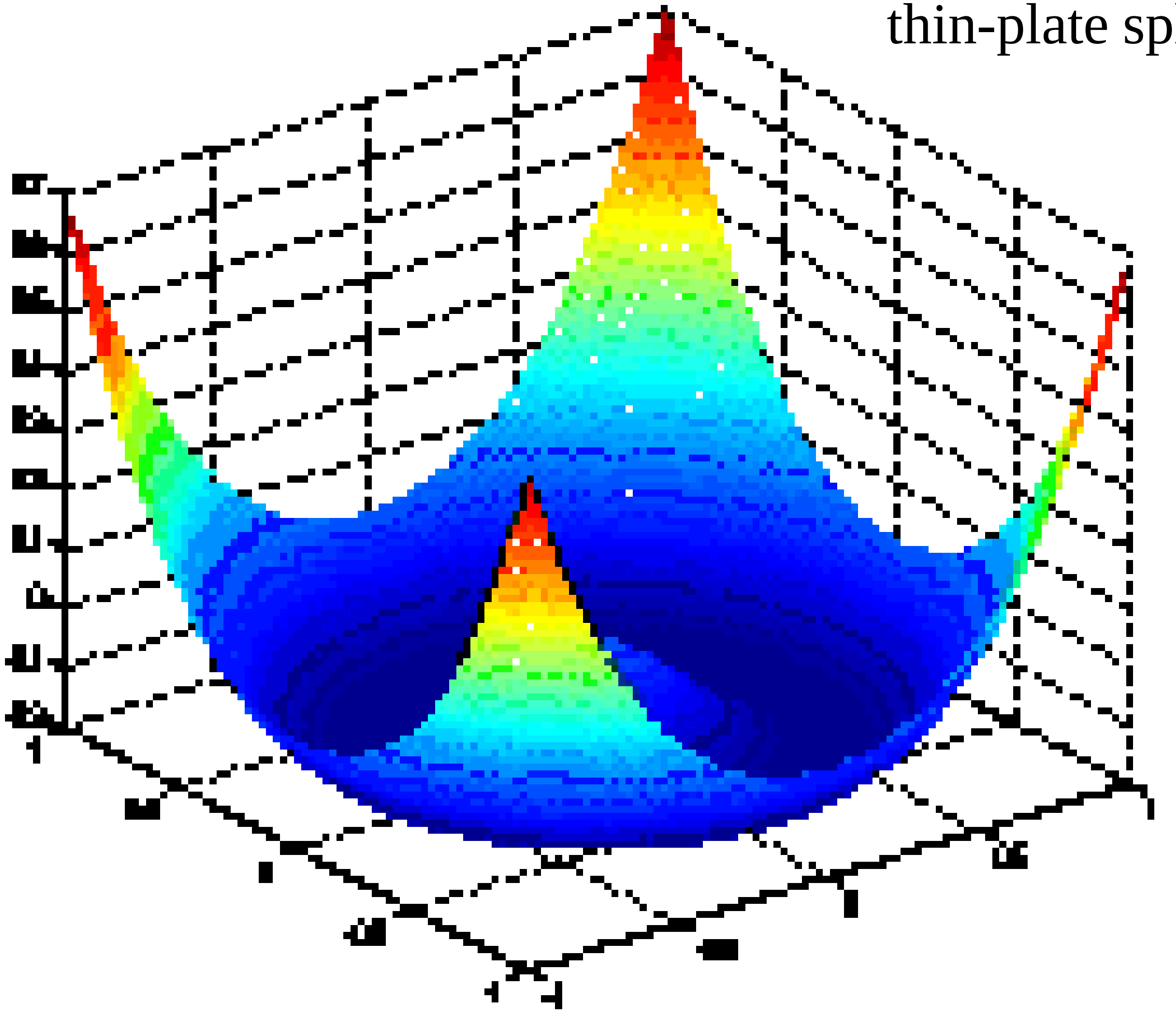
multi-quadratic



inverse multi-quadratic



thin-plate spline



If Gaussian basis function is used, we have:

$$f(\mathbf{x}) = \sum_{j=1}^{m} w_j o_j(\mathbf{x}) = \sum_{j=1}^{m} w_j \exp\left(-\frac{\left\| \mathbf{x} - \mathbf{c}_j \right\|^2}{2\sigma^2} \right)$$

The parameters that determine the RBF are:

(1) The center vectors $\mathbf{c}_j$, j=1,2,…,m.

(2) The weights w_j, j=1,2,…,m.

(3) The width of the basis function σ.

The learning process undertaken by the RBF network may be visualized as follows. The linear weights associated with the output neuron of the network tend to evolve on a different time scale compared with the nonlinear activation functions of the hidden layer neurons.

An import point here is that it is reasonable to separate the optimization of the hidden layer and the output layer by using different techniques. Based on this idea, the training of the RBF neural network can be done using a two-step procedure:

(1) In the first step, the RBF centers are determined.

(2) In the second step, the weights are estimated using some algorithm, with the goal of minimizing the difference between the desired output and the actual output of the RBF neural network.

Weight estimation

It is noted that the output of the RBF neural network has a linear relationship with the output of hidden layer neurons, thus the estimation of weights in the second step can be done using a linear least square estimation algorithm.

Assume there are *N* training samples:

$$\{\mathbf{x}(1), d(1)\}, \{\mathbf{x}(2), d(2)\}, \ldots, \{\mathbf{x}(N), d(N)\}$$

Where d(k) is the target value of **x**(k).

We now assume the centre of each neuron at the hidden layer is known, thus, for input vector **x**(k), we have:

$$f[\mathbf{x}(k)] = \sum_{j=1}^{m} w_j \exp\left(-\frac{\left\| \mathbf{x}(k) - \mathbf{c}_j \right\|^2}{2\sigma^2} \right) = \sum_{j=1}^{m} w_j o_j(k)$$

If we input each of the N sample into the RBF neural network, we can obtain N equations. Summarizing the N equations in a matrix form, yields:

$$\begin{bmatrix} f[\mathbf{x}(1)] \\ f[\mathbf{x}(2)] \\ \vdots \\ f[\mathbf{x}(N)] \end{bmatrix} = \begin{bmatrix} o_1(1) & o_2(1) & \cdots & o_m(1) \\ o_1(2) & o_2(2) & \cdots & o_m(2) \\ \vdots & \vdots & \ddots & \vdots \\ o_1(N) & o_2(N) & \cdots & o_m(N) \end{bmatrix} \times \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}$$

Define:

$$\mathbf{f} = \begin{bmatrix} f[\mathbf{x}(1)] \\ f[\mathbf{x}(2)] \\ \vdots \\ f[\mathbf{x}(N)] \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}$$

$$\mathbf{\Phi} = \begin{bmatrix} o_1(1) & o_2(1) & \cdots & o_m(1) \\ o_1(2) & o_2(2) & \cdots & o_m(2) \\ \vdots & \vdots & \ddots & \vdots \\ o_1(N) & o_2(N) & \cdots & o_m(N) \end{bmatrix}$$

We obtain:

$$\mathbf{f} = \mathbf{\Phi}\mathbf{w}$$

We need to find such weights that the following cost function is minimized:

$$J = \sum_{k=1}^{N} \{ f[x(k)] - d(k) \}^2$$

$$= [\mathbf{f} - \mathbf{d}]^{\mathbf{T}} [\mathbf{f} - \mathbf{d}]$$

$$= [\mathbf{\Phi w} - \mathbf{d}]^T [\mathbf{\Phi w} - \mathbf{d}]$$

Where

$$\mathbf{d} = \begin{bmatrix} d(1) & d(2) & \cdots & d(N) \end{bmatrix}^T$$

Solve

$$\frac{\partial \boldsymbol{J}}{\partial \mathbf{w}} = \mathbf{O}$$

We obtain:

$$\mathbf{w} = (\mathbf{\Phi}^T \mathbf{\Phi})^{-1} \mathbf{\Phi}^T \mathbf{d}$$

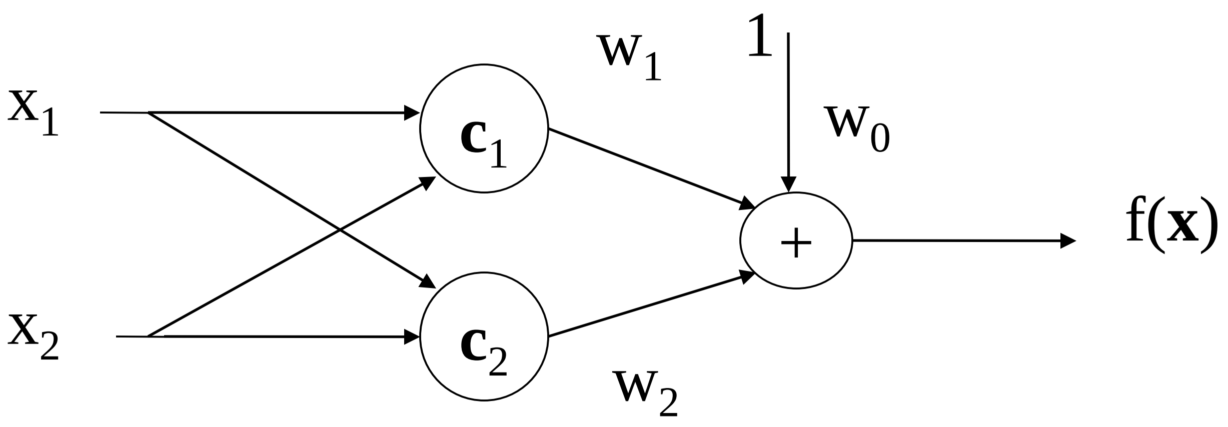
The weights thus obtained are call linear least square estimates.

Example

Consider the exclusive OR (XOR) problem again. The input and output of the logic operator are as follows:

Index of the data	inputs	outputs
1	$[1\ 1]^T$	0
2	$[0\ 1]^T$	1
3	$[0\ 0]^T$	0
4	$[1\ 0]^T$	1

Assume two hidden layer neurons are used, the centers of two neurons are set to: $\mathbf{c}_1=[1\ 1]^T$, $\mathbf{c}_2=[0\ 0]^T$, and the width of the Gaussian basis function is set to 0.707.



$$f(\mathbf{x}) = \sum_{i=1}^{2} w_i \exp(-\|\mathbf{x}-\mathbf{c}_i\|^2 / 2\sigma^2) + w_0$$

Substituting all the 4 data points to the above equation, we obtain the following matrix equation:

$$\begin{bmatrix} 1 & 0.1353 & 1 \\ 0.3678 & 0.3678 & 1 \\ 0.1353 & 1 & 1 \\ 0.3678 & 0.3678 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

The linear least square estimates of the weights are :

$$\begin{bmatrix} w_1 \\ w_2 \\ w_0 \end{bmatrix} = \begin{bmatrix} -2.5018 \\ -2.5018 \\ +2.8404 \end{bmatrix}$$

Let's check the performance of the RBF neural network.

If $\mathbf{x}$=[1 1]^T, we have f($\mathbf{x}$)=0.000106

If $\mathbf{x}$=[0 1]^T, we have f($\mathbf{x}$)=1.0001

If $\mathbf{x}$=[0 0]^T, we have f($\mathbf{x}$)=0.000106

If $\mathbf{x}$=[1 0]^T, we have f($\mathbf{x}$)=1.0001

Obviously the RBF neural network with just two neurons could approximate the XOR logic operation perfectly well.

Strategies for neuron center selection

There are a few strategies that we can follow in the design of an RBF network. A few example strategies are introduced next.

Random selection from training samples

The simplest approach is to assume fixed radial-basis function. Specifically, the locations of the centers may be chosen randomly from the training data. This is considered to be a "sensible" approach provided that the training data are distributed in a representative manner for the problem at hand. For the radial-basis functions themselves, we may employ an isotropic Gaussian function whose standard deviation is fixed according to the spread of the centers:

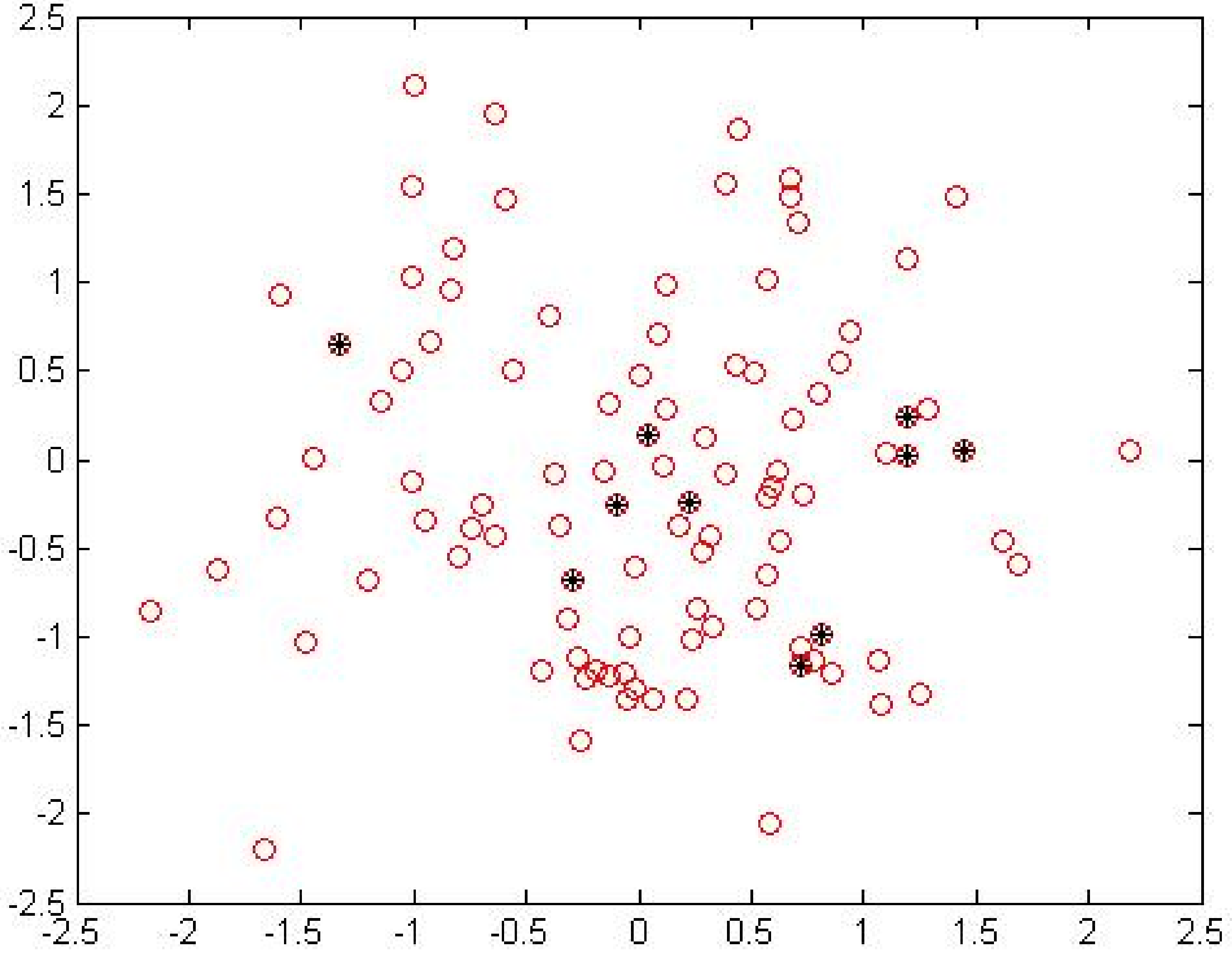
$$\sigma = d_{max} / \sqrt{2m}$$

where d_{max} is the maximum distance between the chosen centers, and m is the number of centers.

The above formula ensures that individual radial-basis functions are not too peaked or too flat (both of the two extreme conditions should be avoided).

As an alternative for basis function width determination, we may use individually scaled centers with broader width in areas of low density, and narrower width in areas of high density.

Random selection method is easy to implement, but the problem with this method is that it cannot guarantee a good coverage of the input space, unless a large number of neurons are used.


2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5

Self-organized selection of centers

The randomly selected centers cannot guarantee to cover the input space well. This problem can be overcome by selecting prototypes of samples as neuron centers. In this approach, the basis functions are permitted to move the locations of their centers in a self-organized fashion. The self-organized component of the learning process serves to allocate network resources in a meaningful way by placing the centers of the RBF functions in only those regions of the input space where significant data are presented. SOM neural networks can be used to fulfill this task.

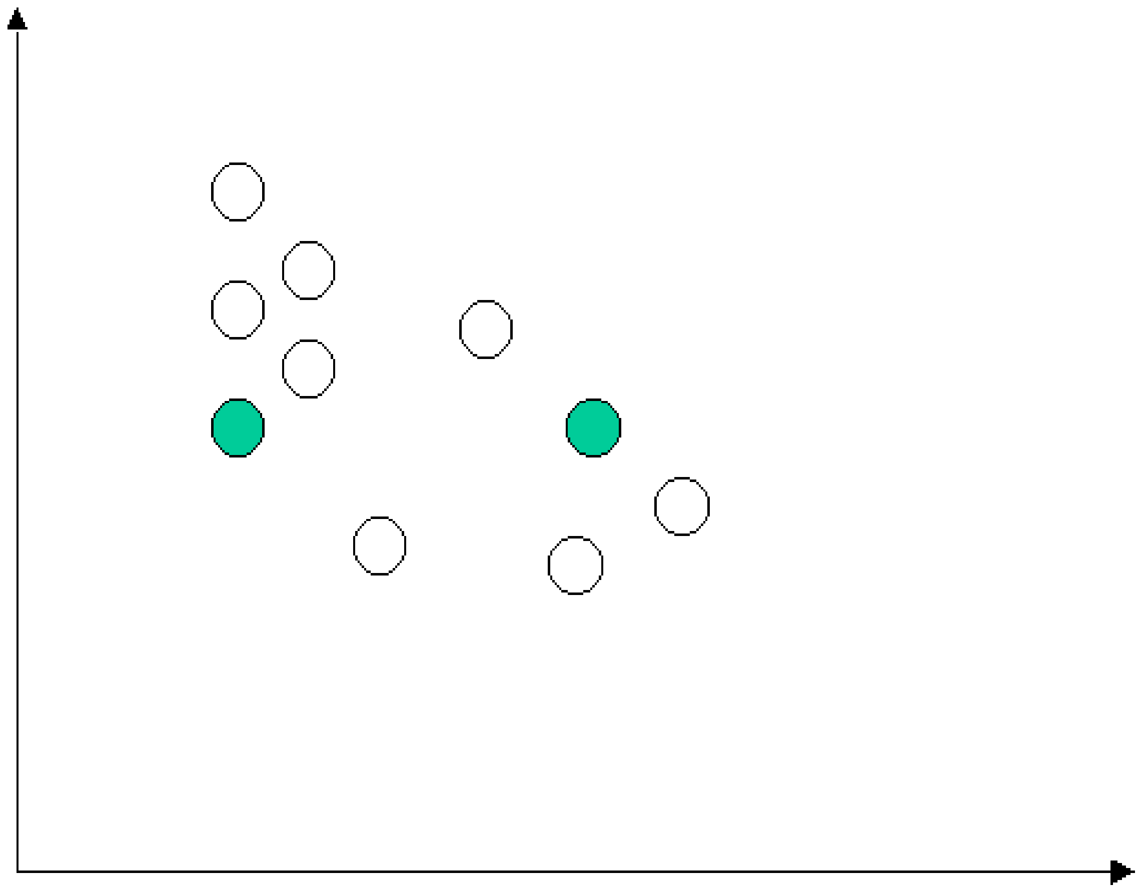
K-means clustering can also be used to select neuron centers for RBF neural networks. K-means clustering algorithm is essentially an iterative procedure described below:

(a) Initialize cluster center positions either by randomly selecting from the samples or by some prior knowledge.

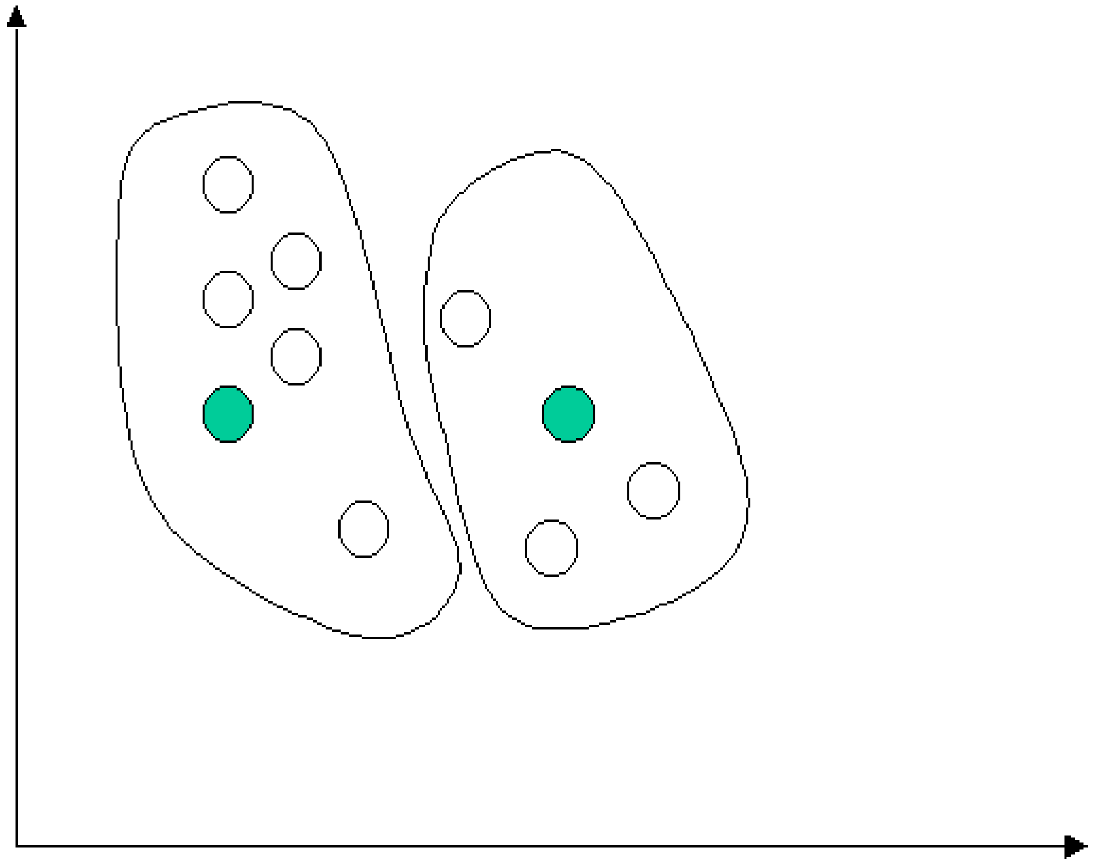
(b) Perform partitioning by assigning samples to the nearest clusters. This is usually done by first calculating the distance between each sample to each cluster center and then assigning samples to the cluster with the smallest distance.

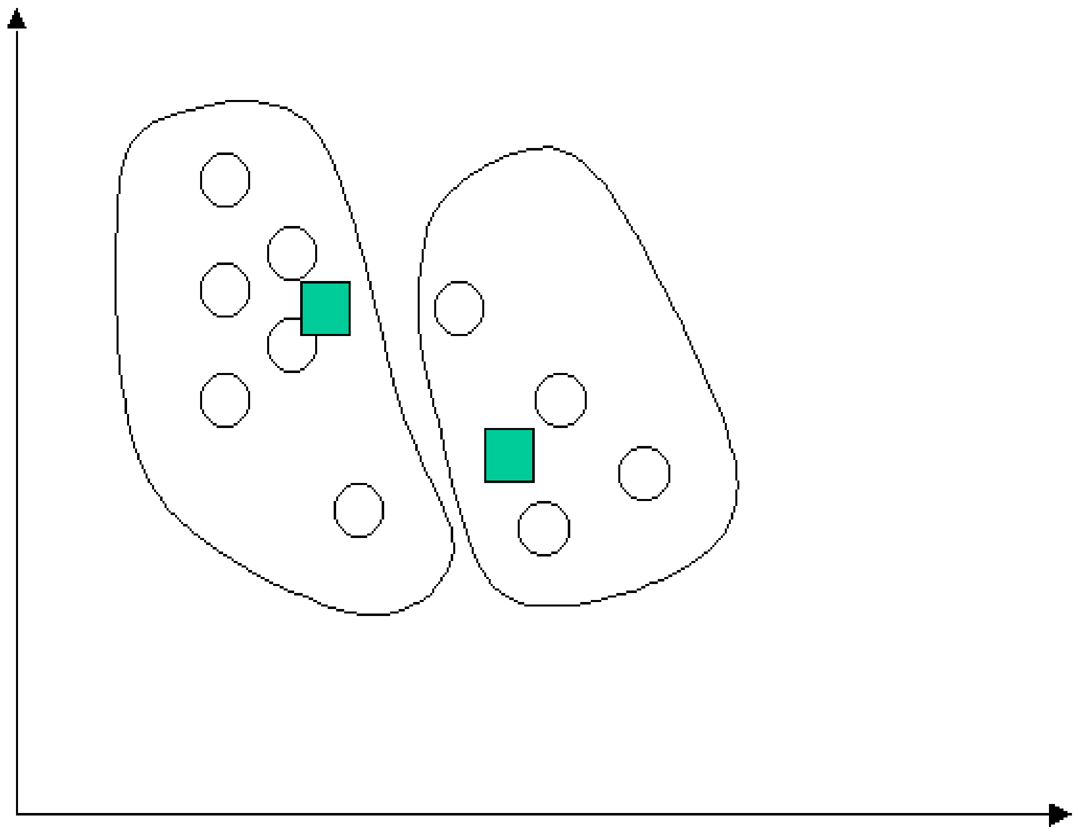
(c) Find new cluster centers by averaging the samples in the same cluster.

(d) Go to step (b) until the cluster center positions and the sample partitioning are no longer changed.

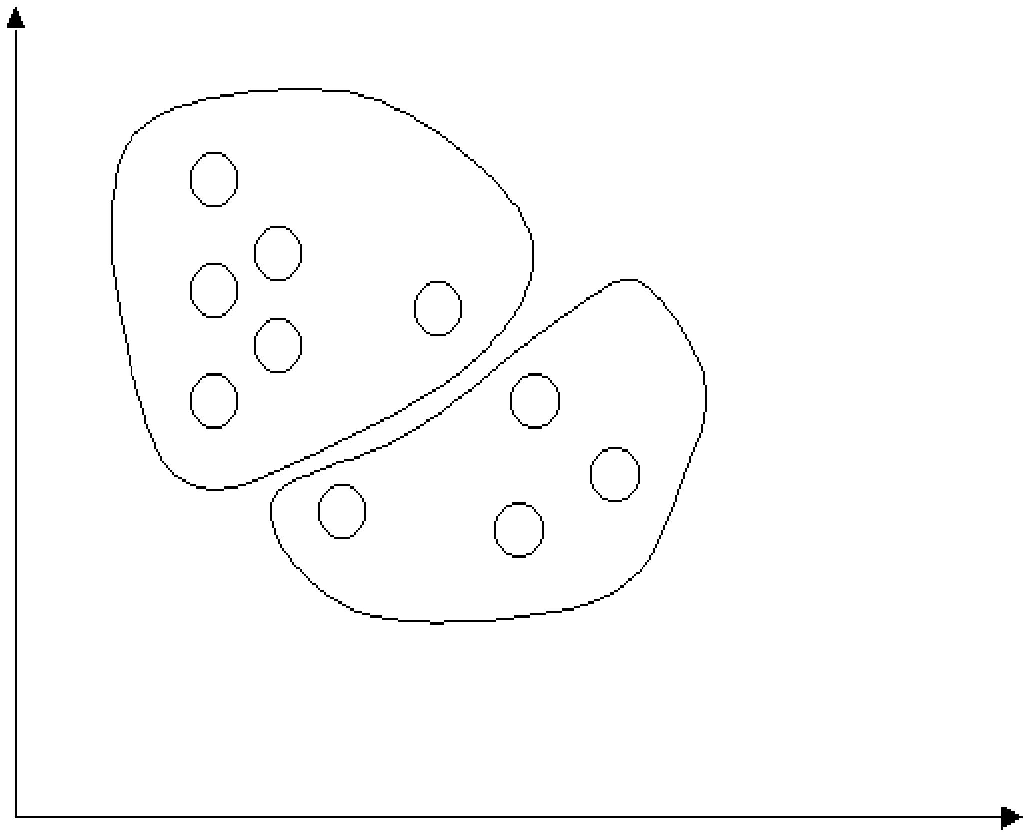


(a) Initialization of cluster centers


(b) Clustering



(c) Finding new cluster centers



(d) Clustering based on new centers.

Center selection as a model term selection problem

For RBF neural network with m neurons, we have:

$$f(\mathbf{x}) = \sum_{j=1}^{m} w_j \exp\left(-\frac{\left\| \mathbf{x} - \mathbf{c}_j \right\|^2}{2\sigma^2} \right) = \sum_{j=1}^{m} w_j o_j(\mathbf{x})$$

f(**x**) is a nonlinear function of **x**, but it has a linear relationship with the output of hidden layer neurons. If these outputs are considered as model terms, the above model has a linear-in-the-parameter structure. Thus, the selection of centers for RBF neural networks can be solved as a linear model selection problem.

Typical methods for model selection are forward selection and backward elimination. The forward selection is a bottom-up method, starting from an empty set. The inclusion of a neuron is based on the approximation error reduction after including the neuron in the network. The approximation error based forward neuron selection procedure can be summarized as follows:

(1) In the first step, consider all training samples (vectors) as the candidate center of the first neuron to be selected:

$$\mathbf{c}_1^j = \mathbf{x}_j \qquad\qquad j=1,2,\ldots N$$

Construct N 1-neuron neural networks using the N candidate neurons respectively, estimate the weight and calculate the approximation error. The neuron that leads to minimum approximation error is selected as the first neuron $\mathbf{c}_1$.

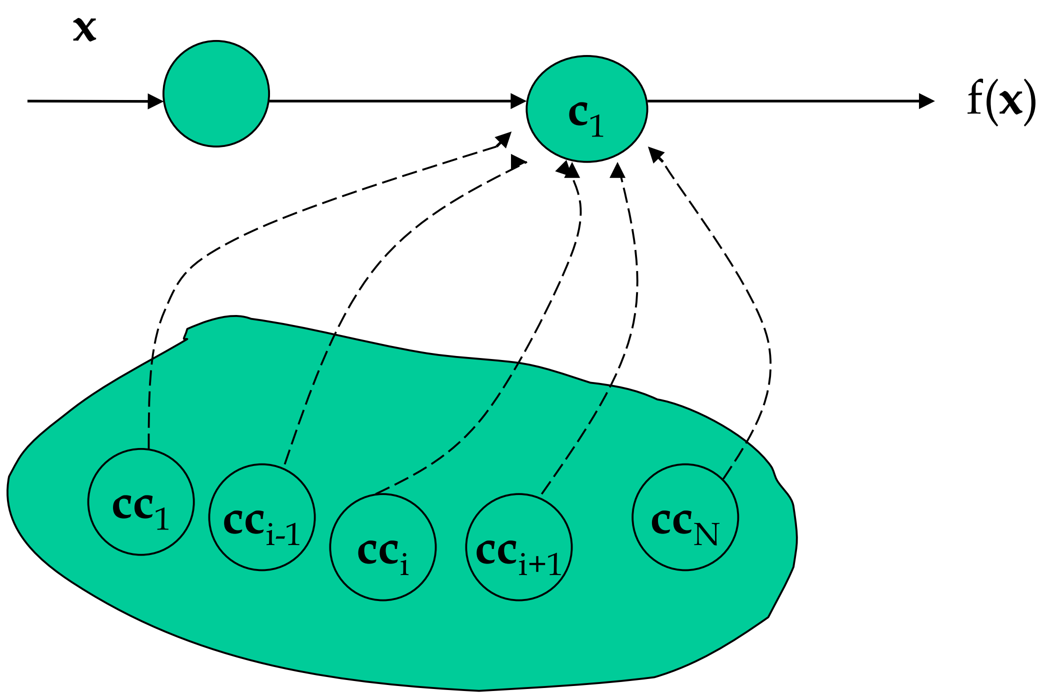
(2) At the second step, consider each of the remaining (N-1) samples as a candidate of center to be secondly selected, construct N-1 2-neuron neural networks using each of the candidate center together with the selected neuron center c_1.

Estimate the weights, calculate the approximation error. The candidate that leads to the minimum approximation error is selected as the second center.

(3) The above procedure is continued until the stopping criterion is satisfied. The stopping criterion can either be a certain number of neurons having been selected, or the approximation error being smaller than a pre-specified value.

At Step 1:

Consider each sample as a center of the neurons to construct a 1-neuron RBF neural network;



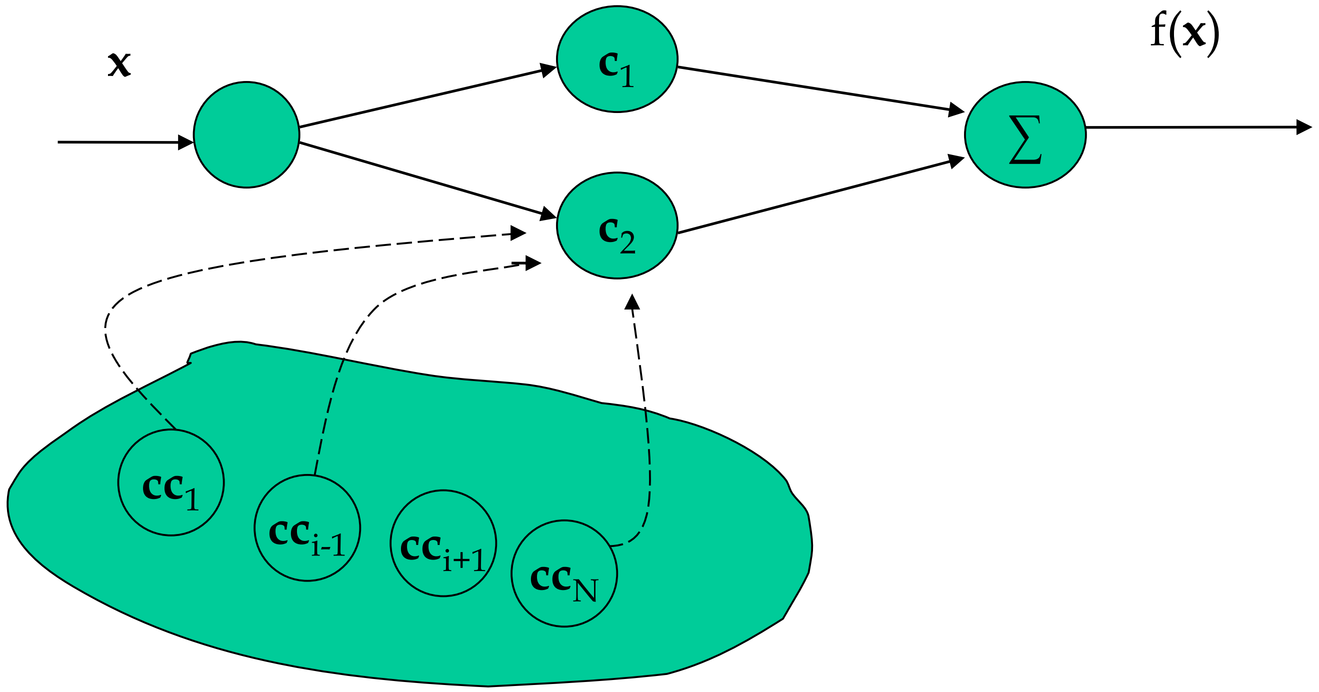
Pool of candidate neuron centers

(1) Calculate the response of the neuron to all training samples;

(2) Estimate the weight and calculate the error. Select the one having minimum error, say $\mathbf{c}_i$, as the first center.

At Step 2:

Consider each neuron center in the pool as the one to be selected next, and construct 2-neuron RBF neural networks.

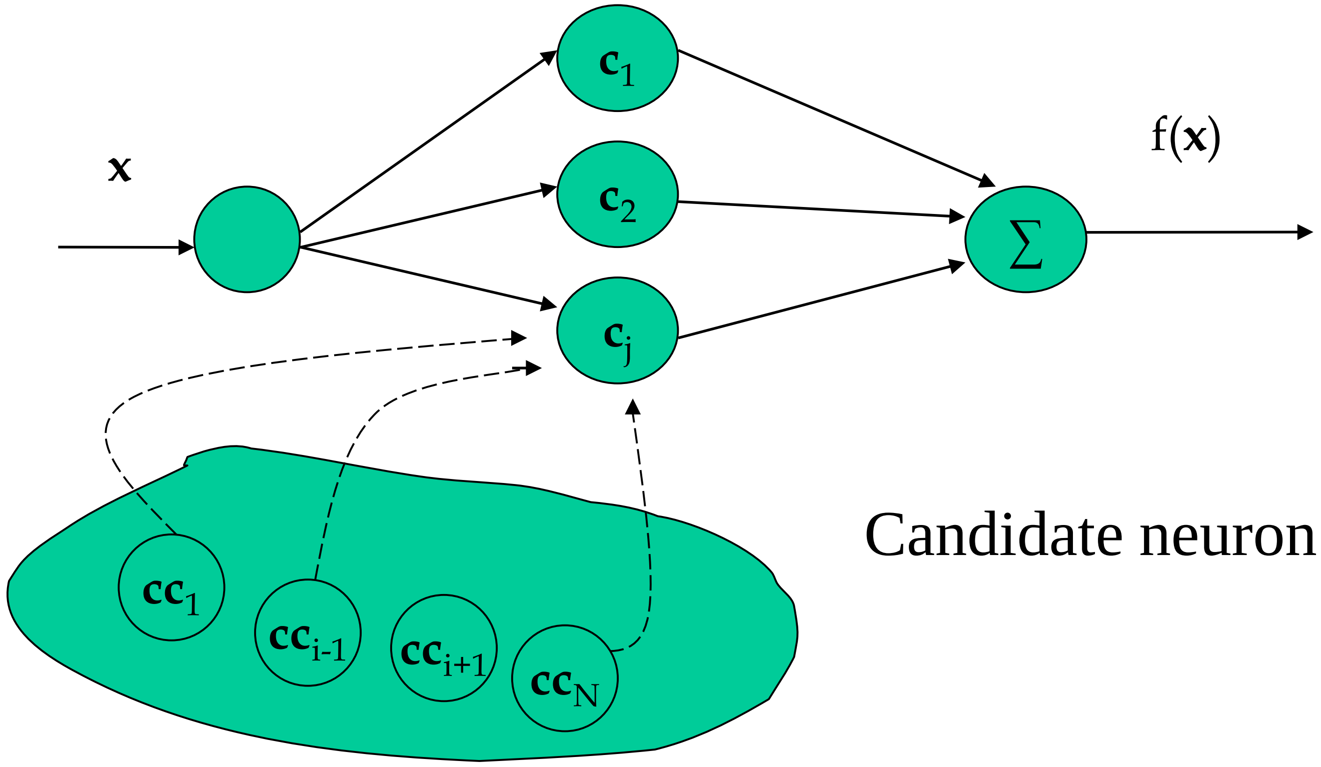


Pool of candidate neuron centers

(1) Calculate the response of the neuron to all training samples;

(2) Estimate the weights and calculate the error. Select the one having minimum error, say $\mathbf{c}_{\mathbf{k}}$, as the first center.

At Step 3:



Candidate neuron center pool

The above processing until the stopping criterion is satisfied.

The stopping criterion:

1) pre-defined number of neurons;

2) Pre-defined accuracy.

Optimization Based Selection of Centers

In this approach, the centers of the RBF and all other parameters of the network undergo a supervised learning process. In other words, the RBF network takes on its most generalized form. A natural candidate for such a process is error-correction learning, which is most conveniently implemented using a gradient-descent procedure.

The first step in such a procedure is to define the criterion to be optimized:

$$E = \frac{1}{2}\sum_{j=1}^{N} e_j^2$$

where N is the number of training samples, e_j is the error signal between the desired output d(j) and the network output f[**x**(j)]:

$$e_j = f[\mathbf{x}(j)] - d(j)$$

By minimizing the cost function E using gradient-descent, we can obtain all parameters of the RBF neural network:

$$\mathbf{c}_j, w_j, \sigma = \arg\min(E)$$

The gradient-descent based optimization can be summarized as follows:

(1) Weight estimation:

$$\frac{\partial E(n)}{\partial w_i(n)} = \sum_{j=1}^{N} e_j(n) \exp\left(\frac{\left\| \mathbf{x}(j) - \mathbf{c}_i(n) \right\|^2}{2\sigma^2} \right)$$

$$w_i(n+1) = w_i(n) - \eta_1 \frac{\partial E(n)}{\partial w_i(n)}$$

(2) Center location estimation

$$\frac{\partial E(n)}{\partial \mathbf{c}_i(n)} = -2w_i(n)\sum_{j=1}^{N} e_j(n)\exp\left(-\frac{\left\|\mathbf{x}(j)-\mathbf{c}_i(n)\right\|^2}{2\sigma^2}\right)\frac{\mathbf{x}(j)-\mathbf{c}_i(n)}{\sigma^2(n)}$$

$$\mathbf{c}_\mathrm{i}(\mathrm{n}+1) = \mathbf{c}_\mathrm{i}(\mathrm{n}) - \eta_2 \frac{\partial \mathrm{E}(\mathrm{n})}{\partial \mathbf{c}_\mathrm{i}(\mathrm{n})}$$

(3) Width estimation

$$\frac{\partial E(n)}{\partial \sigma(n)} = -2\sum_{j=1}^{N} w_j(n) e_j(n)\sum_{i=1}^{m}\exp\left(-\frac{\left\|\mathbf{x}(j)-\mathbf{c}_i(n)\right\|^2}{2\sigma^2}\right)\frac{\left\|\mathbf{x}(j)-\mathbf{c}_i(n)\right\|^2}{\sigma^3(n)}$$

$$\sigma(\mathrm{n}+1) = \sigma(\mathrm{n}) - \eta_3 \frac{\partial \mathrm{E}(\mathrm{n})}{\partial \sigma(\mathrm{n})}$$

where η_1, η_2, and η_3 are step-size. The iteration is repeated until parameters have no noticeable changes between two iterations.

Application example of RBF neural networks in data mining

Direct mailings to a potential customers can be a very effective way to market a product or a services. However, as we all know, many of this junk mails are of no interest to the people to receive it. And most of the mails are thrown away. This wastes and costs money.

If the company has a better understanding of who their potential customers are, they can know more accurately who to send mails. Thus, the cost can be reduced.

We can understand customers by analyzing their personal data:

(1) Select a small groups of customers of various background, and collect their responds (training data).

(2) Train a pattern classifier using the above small group of data.

(3) Predict the interest of other customers and select those that have high possibility to be interested.

An insurance company wishes to launch a caravan insurance plan. The problem under study is to predict which customers are potentially interested in the caravan insurance policy.

(1) Training data set.

The training data set contains 5822 customers. The record of each customer consists of 86 attributes (features, variables) including socio-demographic data and product ownership. The socio-demographic data is derived from zip-codes. All customers living in the same areas with the same zip code have the same socio-demographic attributes. In the training data, the information of whether or not the customers have a caravan policy (class labels) is also include.

Among the 5822 customers, only 348 customers have a caravan policy.

(2) Test data set

The test data contains 4000 customers, of whom the information of whether they have a caravan policy is NOT included. The problem here is to select 800 customers that are most interested in the policy from the 4000 customers.

First, we need to train a classifier using the training data. Among the 5822 customers, 348 have a policy (class 1), while 5474 do not (class 2). The data exhibits an unbalanced problem (5474:348). So in the classifier training, we select 500 customers from the 5474 customers of class 2 using SOM neural networks.

The another problem with the training data is that, among the 85

Attributes available (excluding class labels), many are irrelevant, insignificant or redundant. So, before the classifier training, we need to perform attribute selection (feature selection) by eliminating the irrelevant, insignificant and redundant variables, and keeping the useful attributes.

The usefulness of an attribute subset can be evaluated based on its ability to provide large class separation. Based on this criterion, we select 12 attributes, which are used as input variables to the RBF neural network classifier.

After feature selection, we next determine the number of neurons at the hidden layer and the center locations for each neuron.

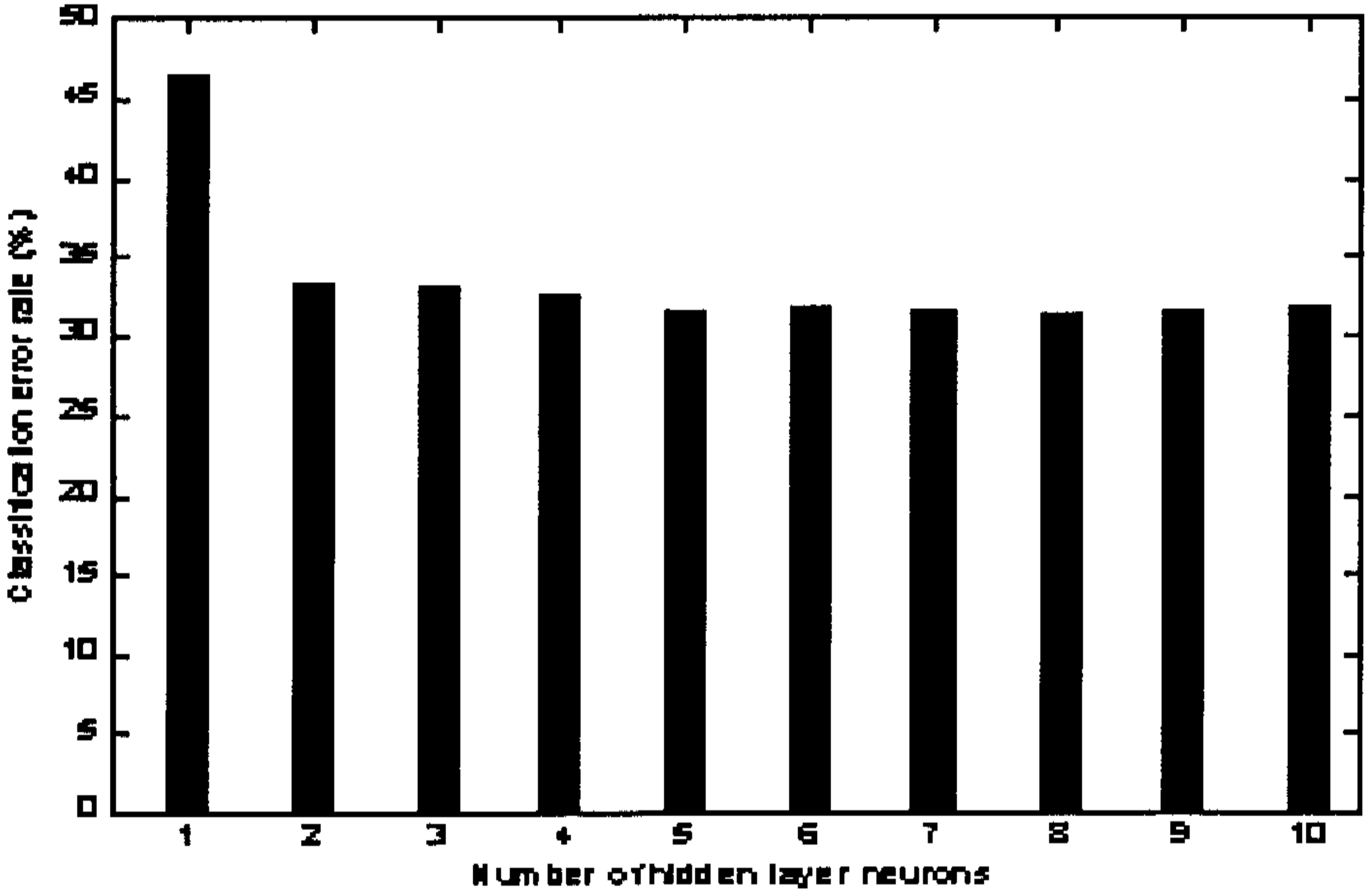
To select neuron center locations, we first consider each of the 848 (348+500) training samples as candidates, and then use the bottom-up method to select.

The RBF network classifier has a structure is 12×7×1. The weights that connect hidden layer neurons and the output layer neuron are estimated using a linear least algorithm which minimizes the following cost function:

$$J = \sum_{k=1}^{848} \{ f[\mathbf{x}(k)] - d(k) \}^2$$

Where d(k) is the class label of the kth training sample, having a value of 1 or -1. f[**x**(k)] is the network output for the kth training sample.

With the RBF classifier learned, we select a set of 800 customers that most possibly have a caravan insurance policy. The number of correctly identified policy owner is 105. Actually, there are 238 policy owners among the 4000 testing customers.



Classification error rate of the 848 training samples

If we randomly select 800 customers from the 4000, the correctly identified policy owner would be

$$\frac{800}{4000} \times 238 \approx 46$$

The advantage of data mining is obvious.