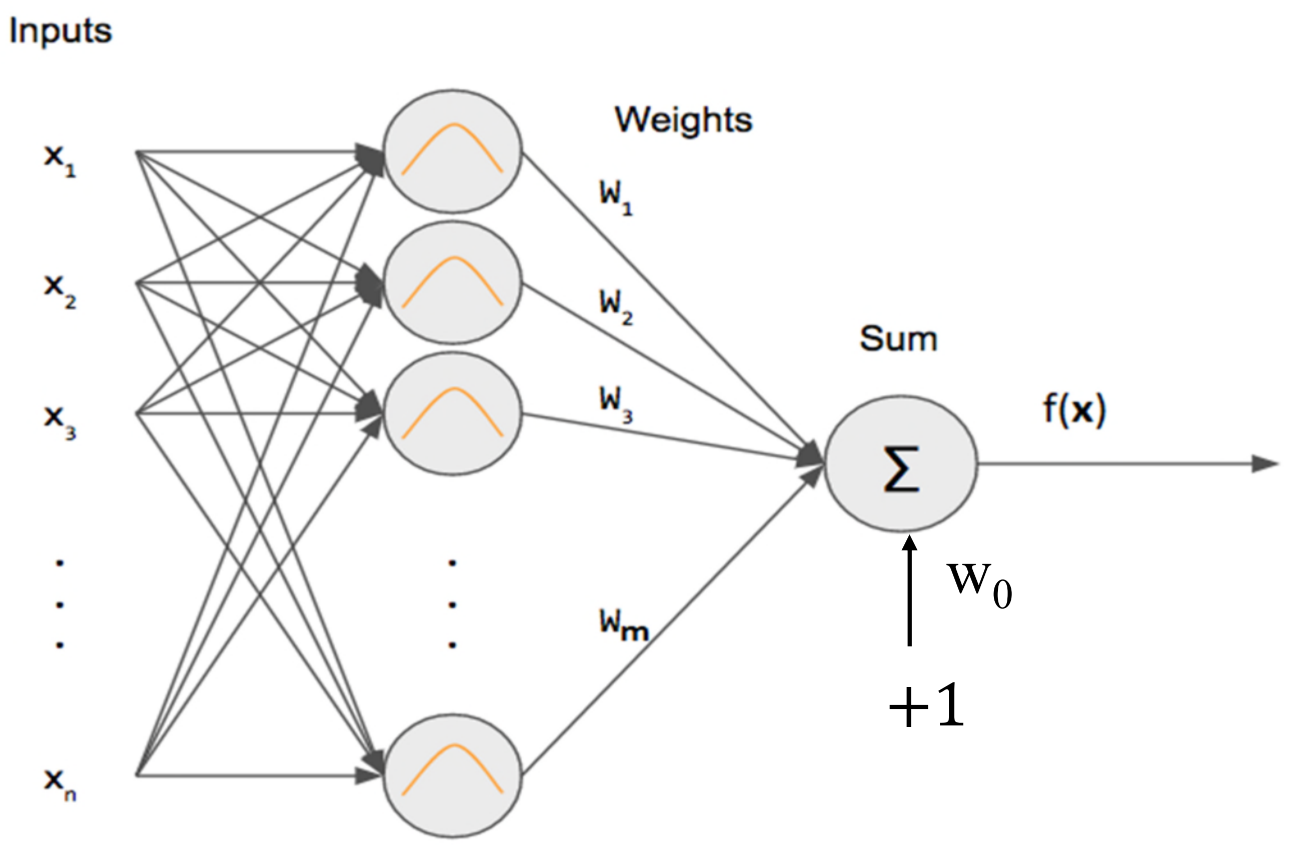
3. Radial Basis Function (RBF) Neural Networks

In the nervous system of biological organisms, there is evidence that responses of some neurons 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 explore a type of neural network inspired by this finding.

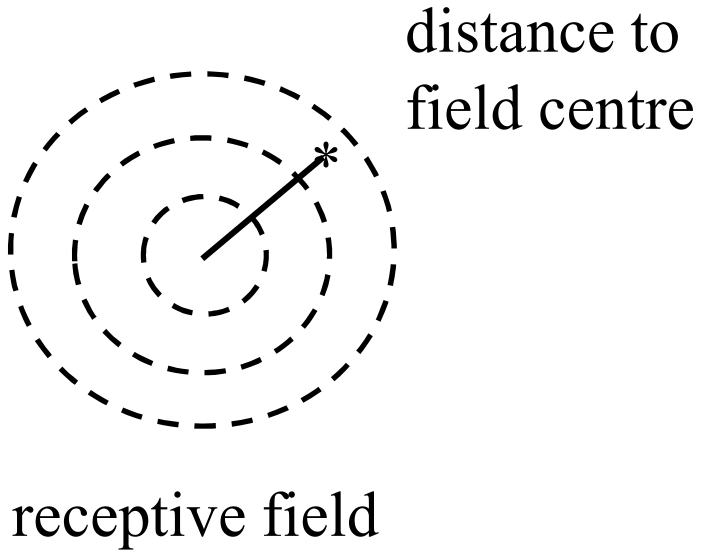
This neural network is known as the radial basis function (RBF) neural network. The RBF neural network has a feed-forward structure with a modified hidden layer and training algorithms.

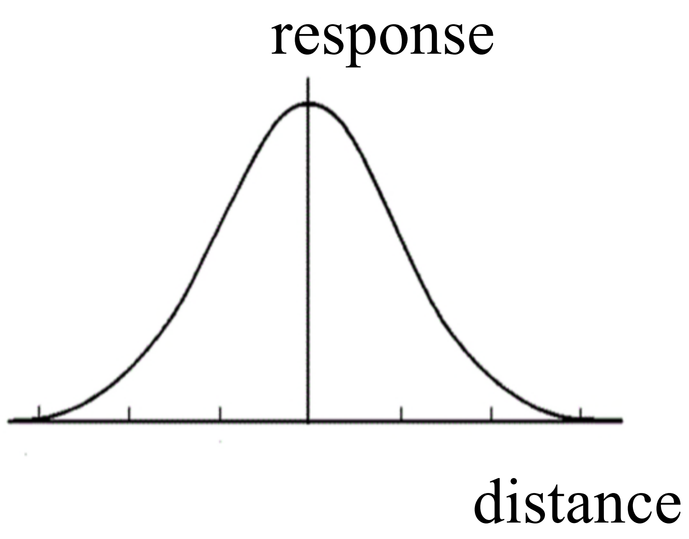
RBF neural network architecture



The architecture of RBF neural network with a single output

RBF networks emulate the behavior of certain biological networks. Basically, the hidden layer consists of the locally tuned or locally sensitive neurons, whose response (output) is localized and decays as a function of the distance from the input to the center of neuron's receptive field as illustrated below:





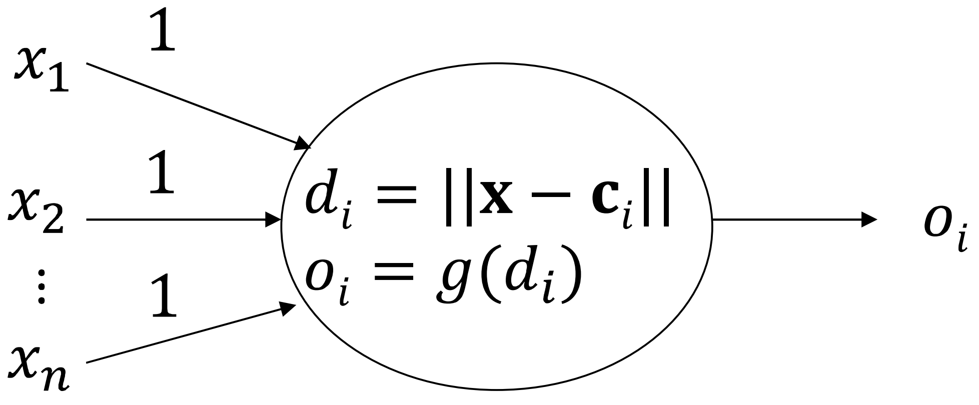
RBF neural network characteristics

(1) The input layer

The input layer neurons do not perform any computation and just distribute the inputs to the hidden layer.

(2) The hidden layer

The weights between input layer neurons and hidden layer neurons in the RBF network are all set to 1. A general form for hidden layer neurons of the RBF neural network is illustrated as:



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

$$d_i = \|\mathbf{x} - \mathbf{c}_i\|$$

$$o_i(\mathbf{x}) = g(d_i) = g(\|\mathbf{x} - \mathbf{c}_i\|)$$

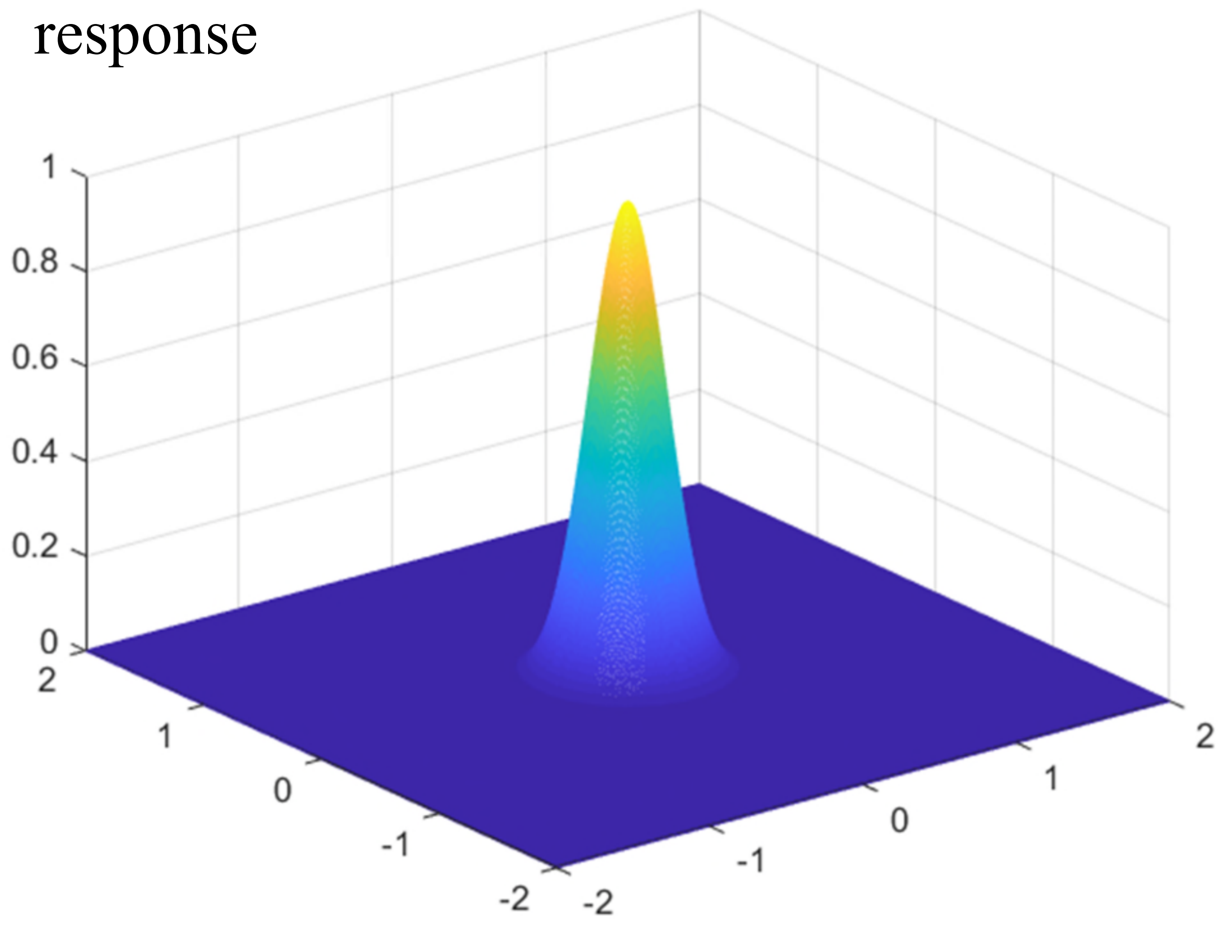
Where $\mathbf{c}_i$ is called center vector, i.e. the center of receptive field, of neuron i; g is the radial basis function. Some commonly used basis functions are as follows:

(1) The Gaussian function

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

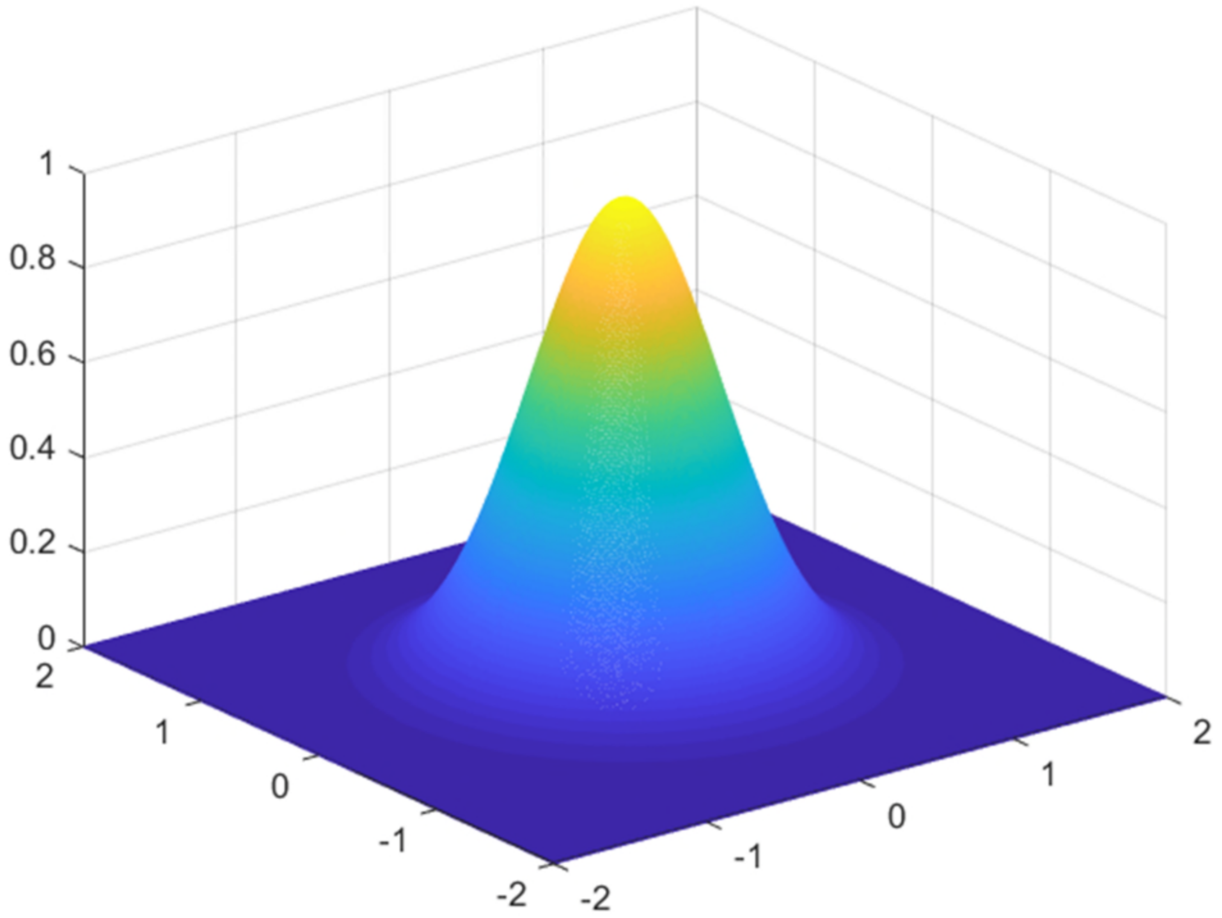
where σ is the width of the basis function, which directly determines the size of the receptive field.

Gaussian basis function with σ=0.2

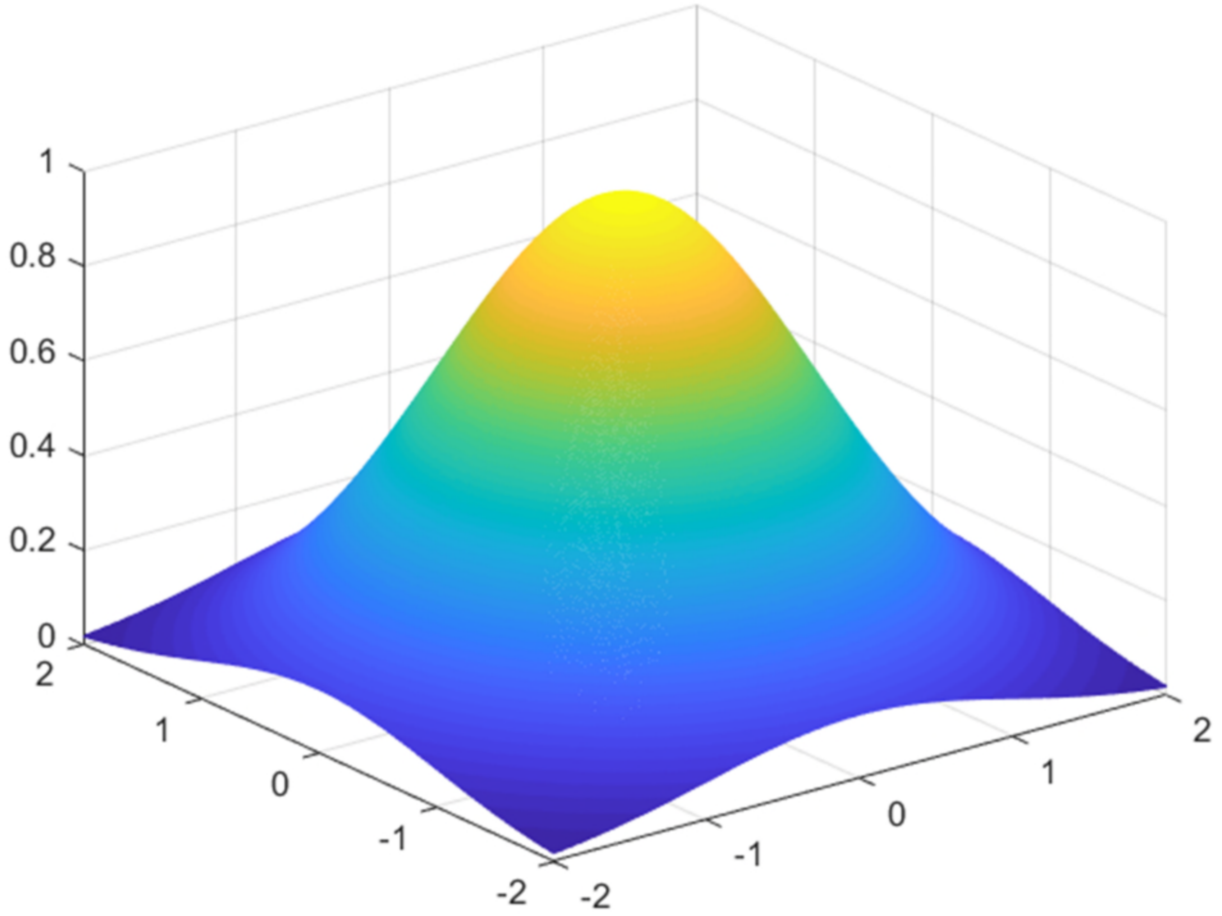


Receptive field centred at (0,0)

Gaussian basis function with σ=0.5



Gaussian basis function with σ=1

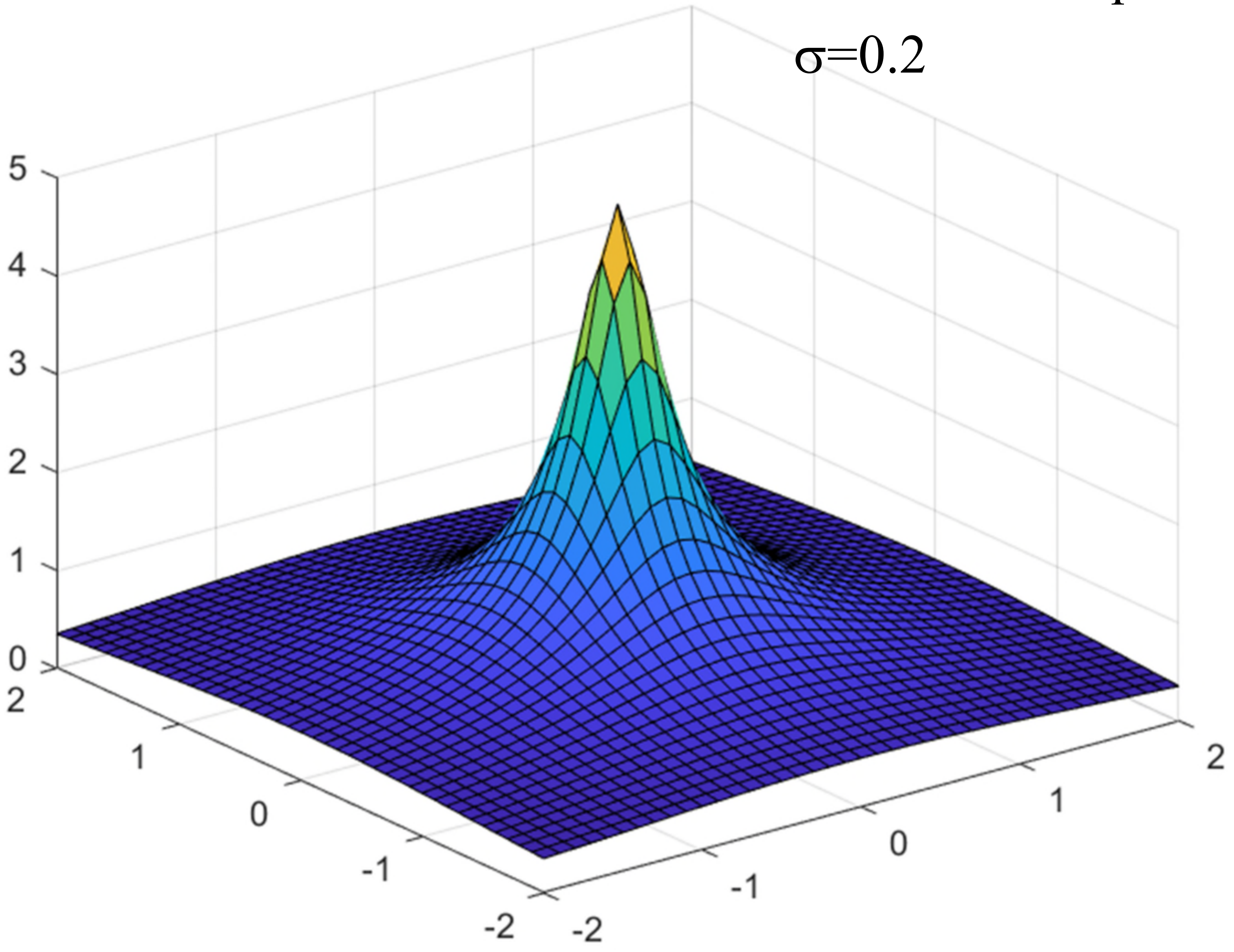


(2) The inverse multi-quadratic function

$$g(d_i) = \frac{1}{\sqrt{{d_i}^2 + \sigma^2}}$$

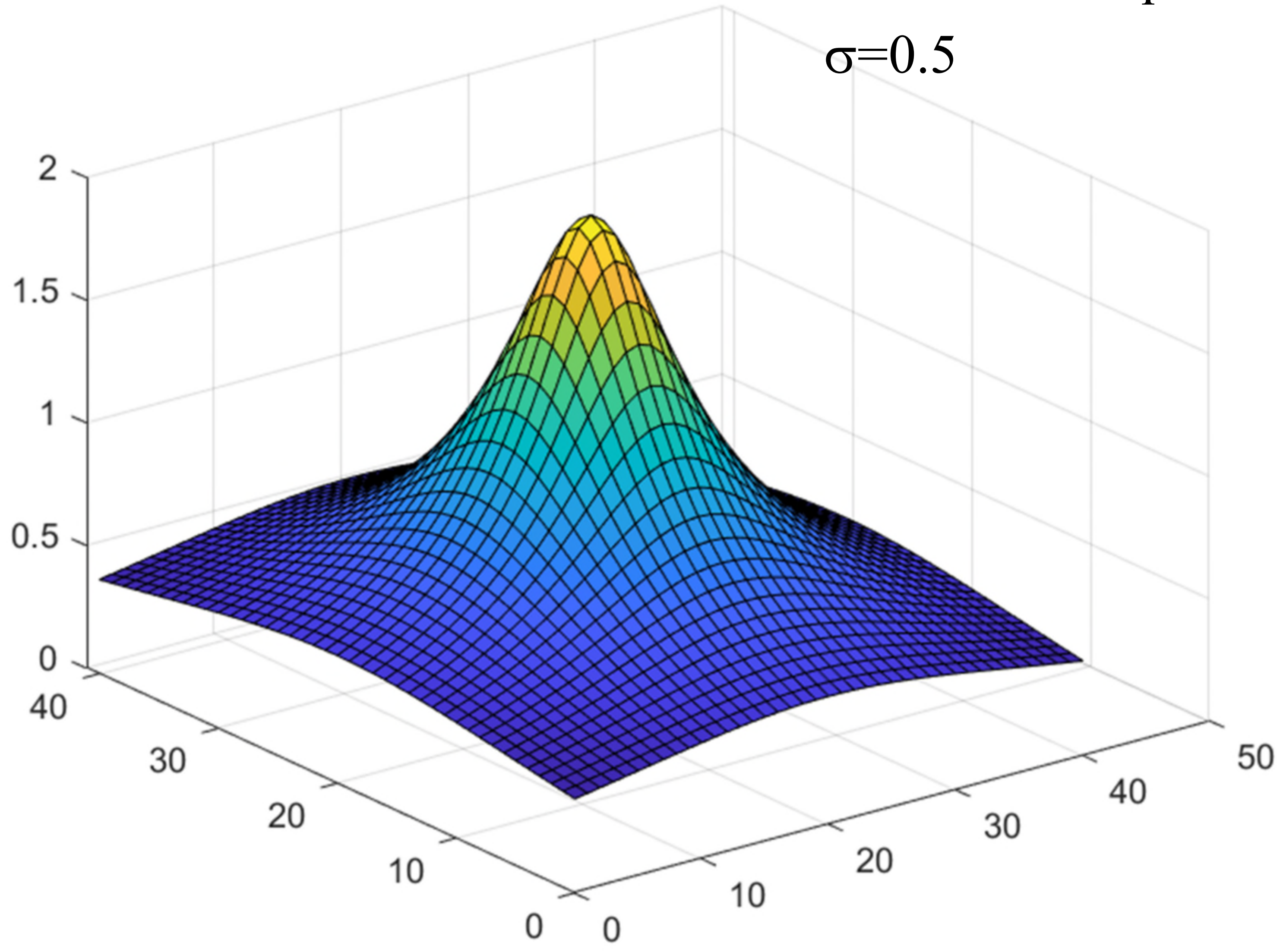
inverse multi-quadratic

σ=0.2



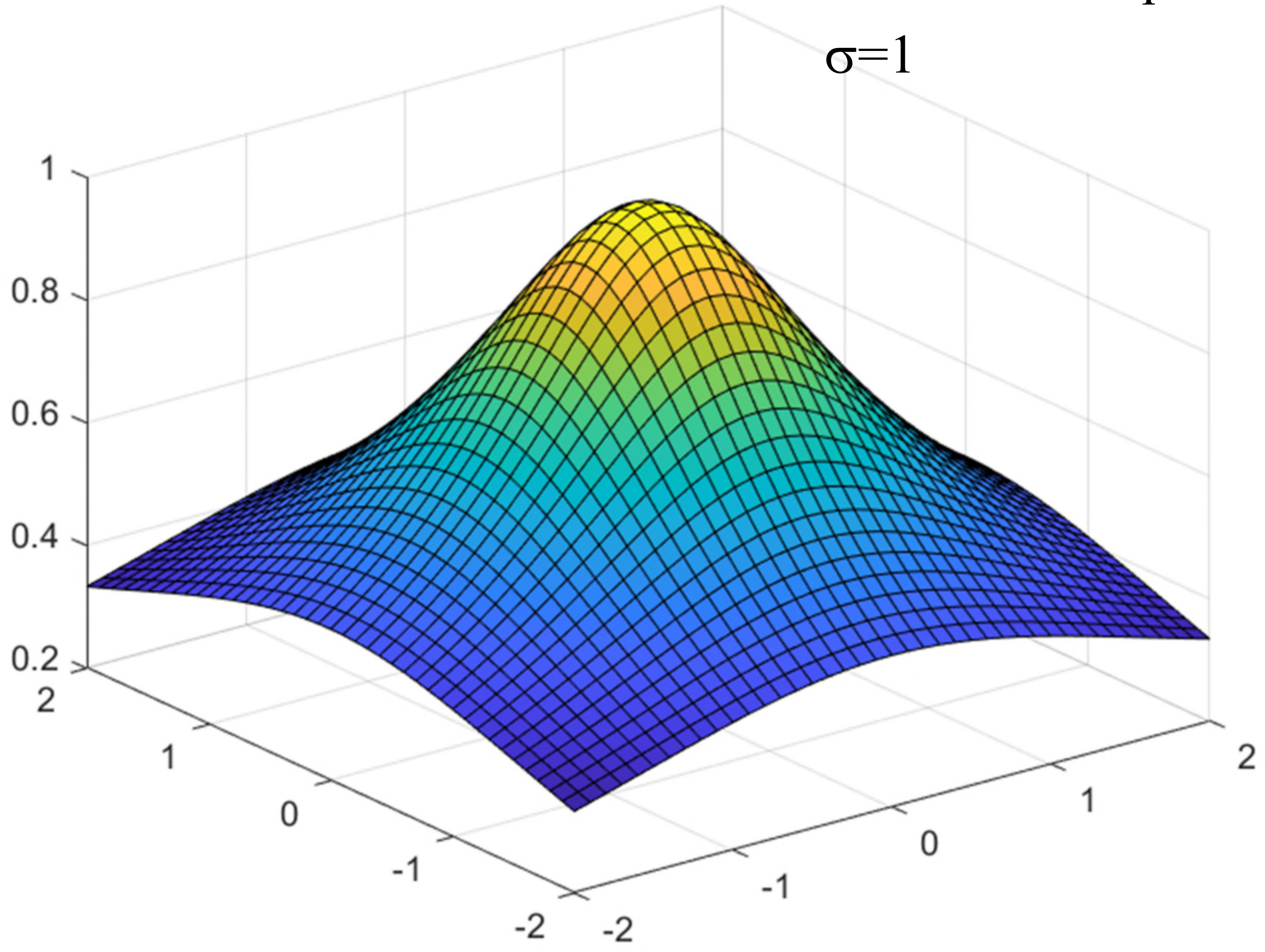
inverse multi-quadratic

σ=0.5



inverse multi-quadratic

σ=1



(3) The output layer

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

$$f(\mathbf{x}) = \sum_{j=1}^{m} w_j \, o_j(\mathbf{x}) + w_0$$

Where m is the number of hidden layer neurons, w_j is the weight from hidden layer neuron j to the output layer neuron, w_0 is the bias term.

If Gaussian or inverse multi-quadratic function is used, we have:

$$f(\mathbf{x}) = \sum_{j=1}^{m} w_j \, o_j(\mathbf{x}) + w_0 = \sum_{j=1}^{m} w_j \exp\left(-\frac{\left\|\mathbf{x} - \mathbf{c}_j\right\|^2}{2\sigma^2}\right) + w_0$$

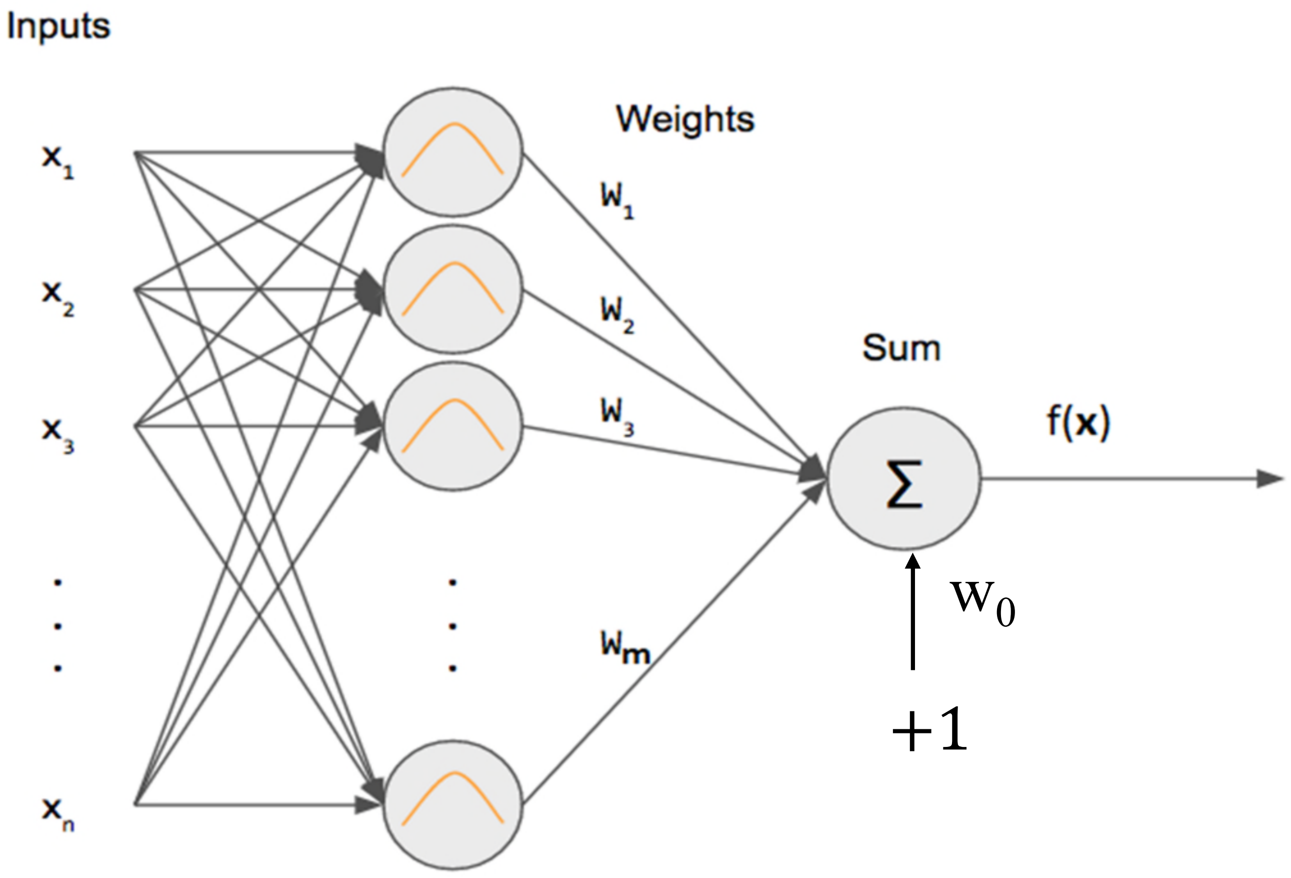
$$f(\mathbf{x}) = \sum_{j=1}^{m} w_j\, o_j(\mathbf{x}) + w_0 = \sum_{j=1}^{m} w_j \left(\left\|\mathbf{x} - \mathbf{c}_j\right\|^2 + \sigma^2\right)^{-\frac{1}{2}} + w_0$$

In either case, the parameters that determine the response of the RBF neural network are:

- ❑ The center vectors $\mathbf{c}_j, j = 1,2, \dots, m$.
- ❑ The weights $w_j, j = 0,1,2, \dots, m$.
- ❑ The width of the basis function σ.

It is noted that the network output is a nonlinear function of σ and $\mathbf{c}_j$ but it has a linear relation with the output of hidden neurons.

Let's have a look of the architecture of RBF neural network again:



The mapping performed by the RBF neural network can be interpreted as the combination of two steps:

- ❑ Step1: a nonlinear mapping from the input space to hidden space
- ❑ Step 2: a linear mapping from the hidden space to the output space

$$\mathbf{x} \longrightarrow \boxed{g(\|\mathbf{x}-\mathbf{c}_i\|)} \xrightarrow{\mathbf{o}} \boxed{\mathbf{w}^T\boldsymbol{o}+w_0} \longrightarrow f(\mathbf{x})$$

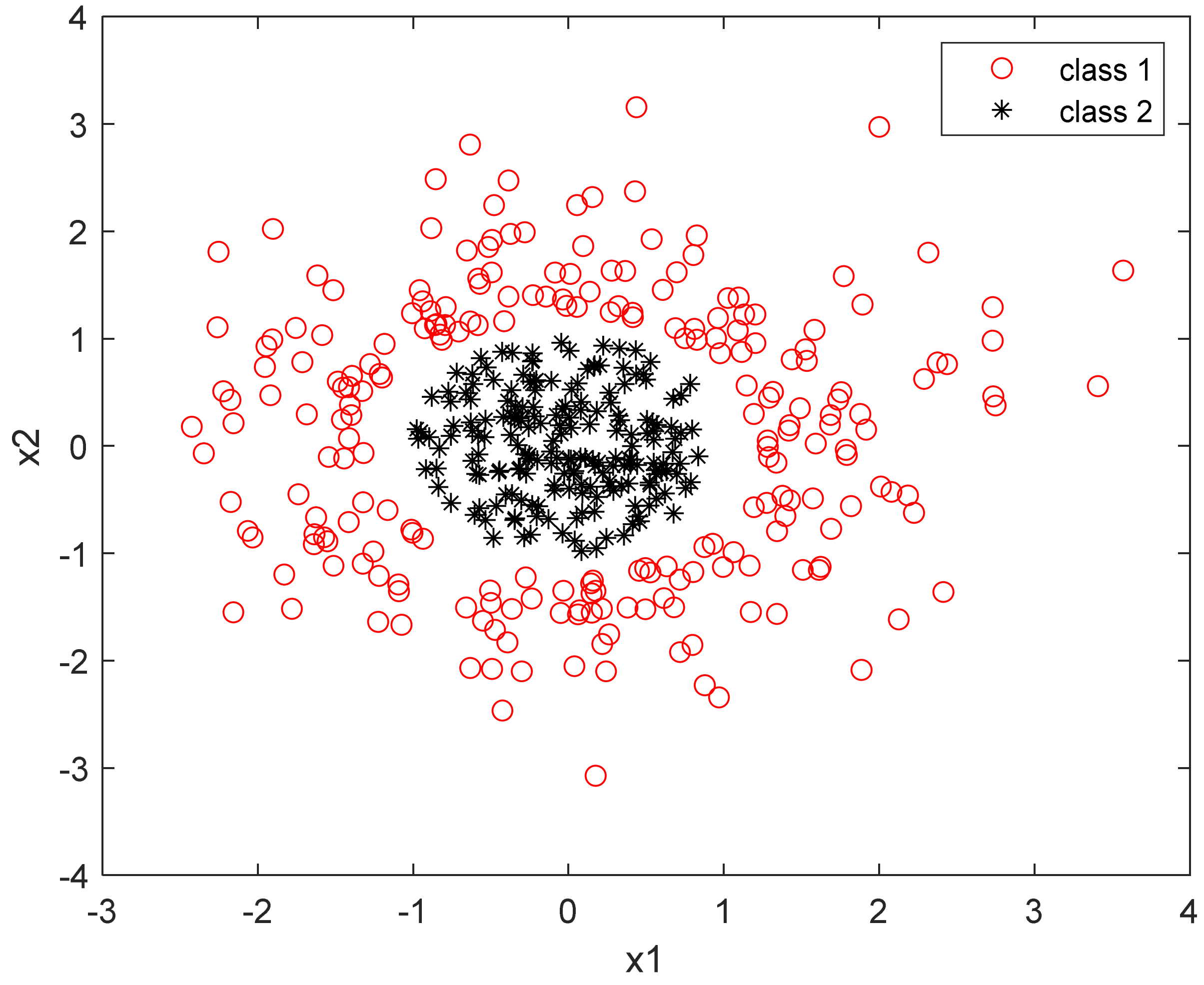
Nonlinear mapping Linear mapping

Where

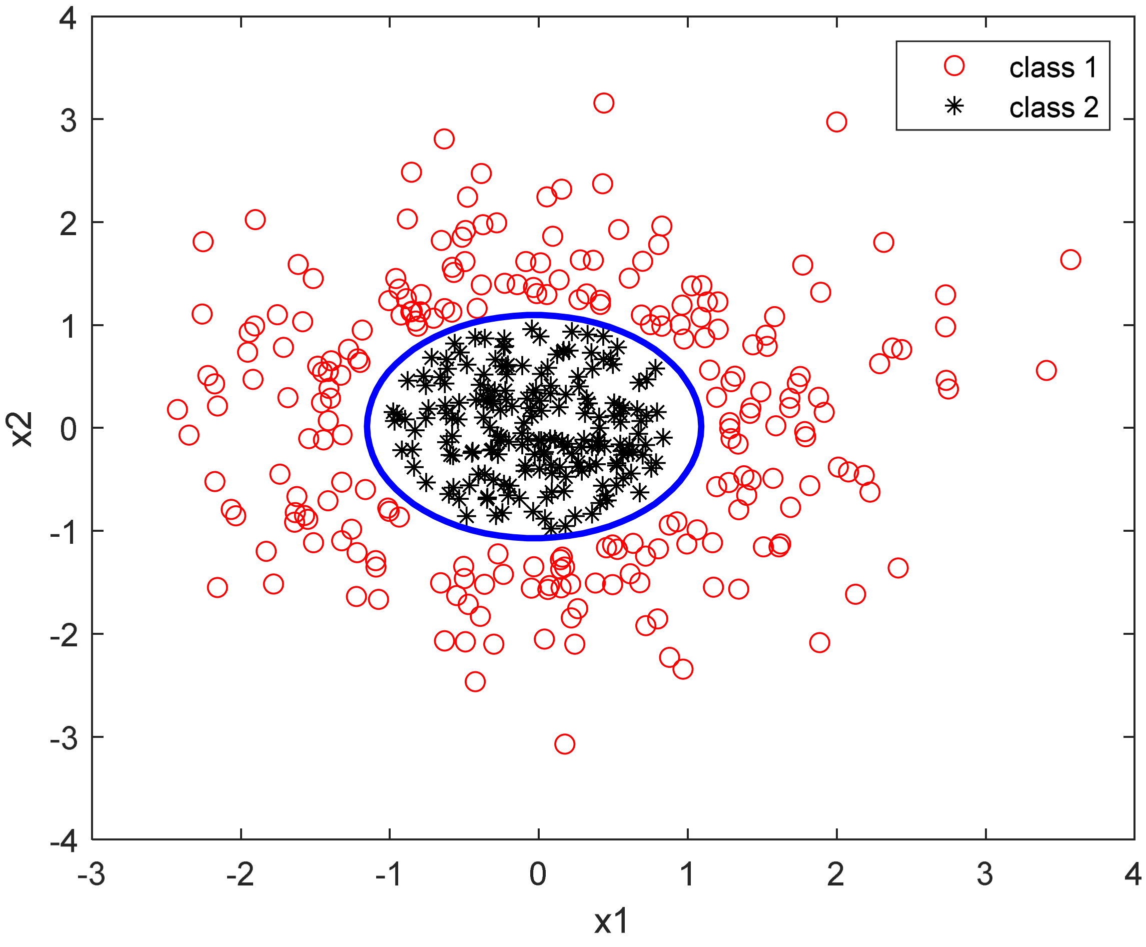
$$\mathbf{w} = [w_1 \quad w_2 \quad \cdots \quad w_m]^T$$

$$\mathbf{o} = [o_1 \quad o_2 \quad \cdots \quad o_m]^T$$

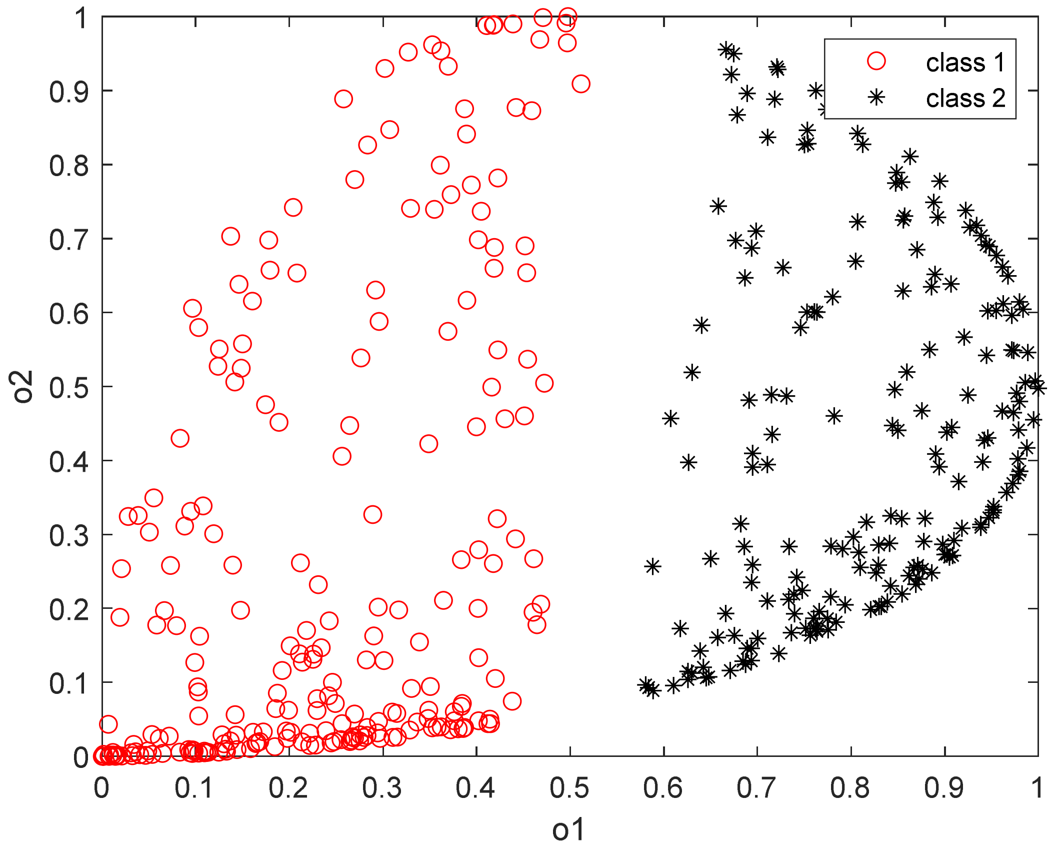
Consider a nonlinear classification problem shown below:



Our goal is to design an RBF network classifier (the blue curve is the decision boundary) to separate the samples of the two classes:

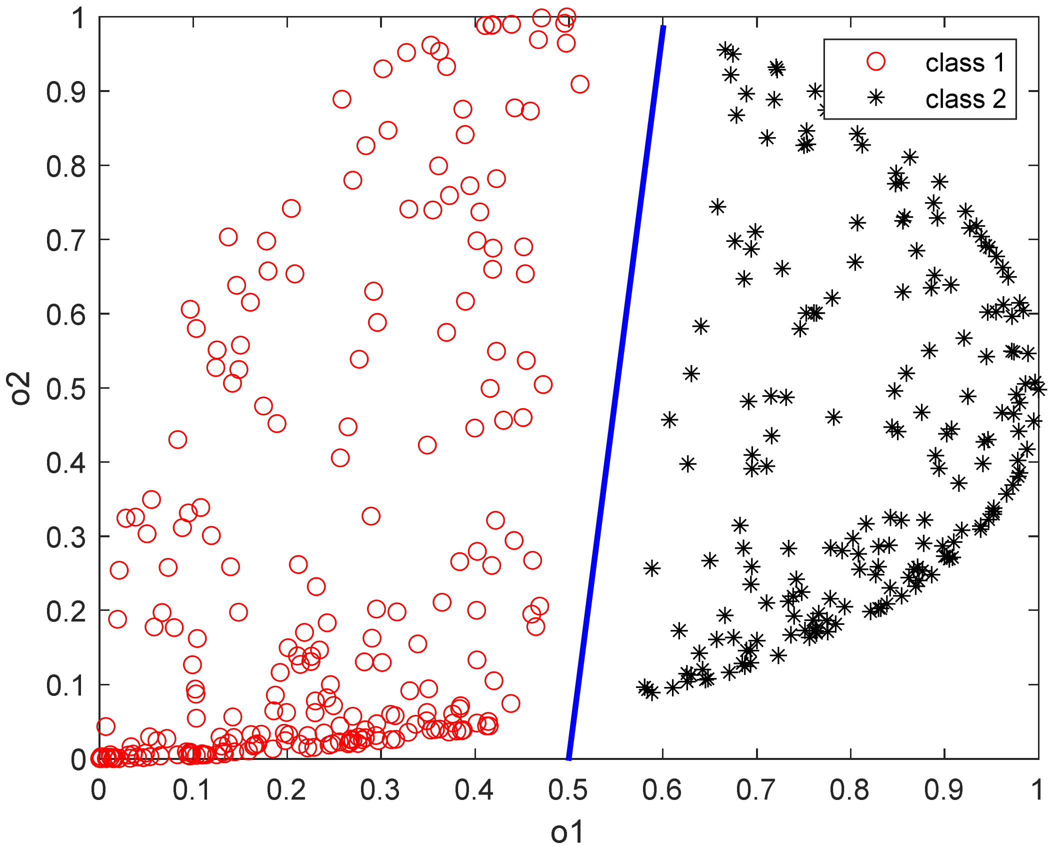


This can be done by two steps. Step 1: mapping from input space to hidden space where samples are linearly separable as shown below:



Step 2: linear classification in the hidden space

$$f = w_1 o_1 + w_2 o_2 + w_0$$



The above interpretation of the RBF neural networks suggests linear weights associated with the output neuron and the centre vectors of the hidden layer neurons may be learned separately:

(1) In the first step, the RBF hidden neuron center vectors are determined.

(2) In the second step, the weights are estimated using linear estimation method which minimizes the difference between the desired output and the actual output of the RBF neural network.

Question:

Any intuitive ideas on the centre vector determination?

(1) Methods for neuron center determination

There are a few methods that we can follow in the determination of hidden layer neuron center vectors. A few example methods are introduced next.

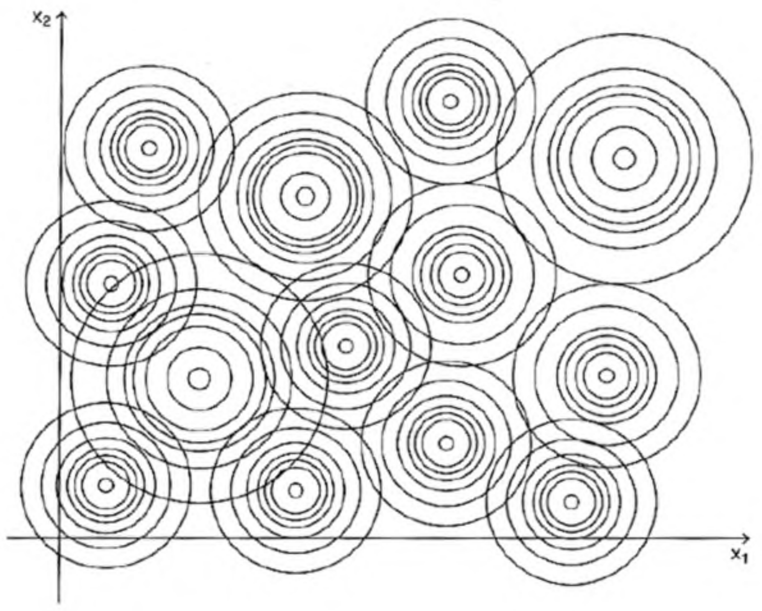
❑ **Random selection from training samples**

This is the simplest approach. The center vectors are chosen randomly from the training data or generated randomly from the region of the data. This is considered to be a sensible approach provided that the randomly selected training sample are distributed in a representative manner for the training data.

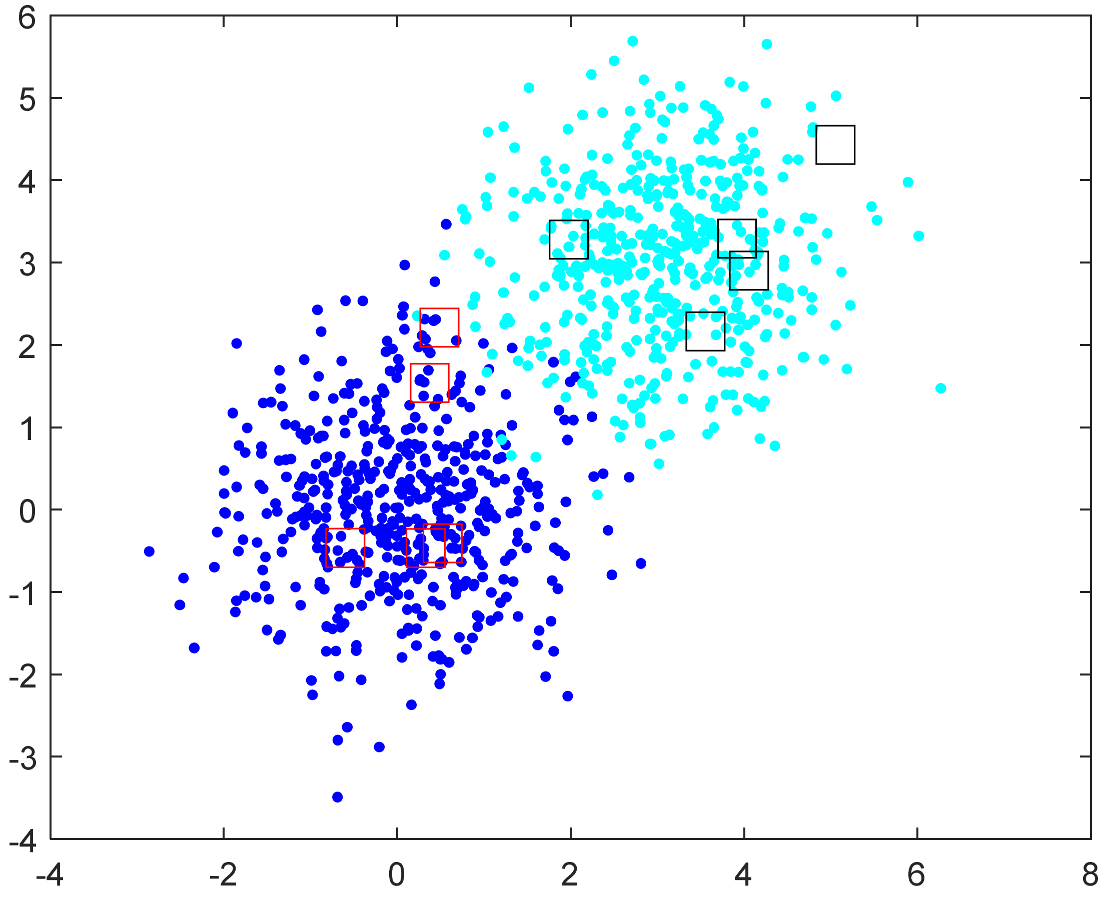
For the radial basis function, we may employ Gaussian function whose width is set as follows:

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

where d_{max} is the maximum distance between the chosen center vectors, and m is the number of center vectors (i.e. number of hidden layer neurons). The above setting will ensure that the radial basis functions are not too peaked or too flat. We may also use individually scaled center vectors with broader width in areas of low density, and narrower width in areas of high density.



Random selection/generation method is easy to implement, but this method cannot guarantee a good coverage of the input space if a small number of neurons are used as shown below:

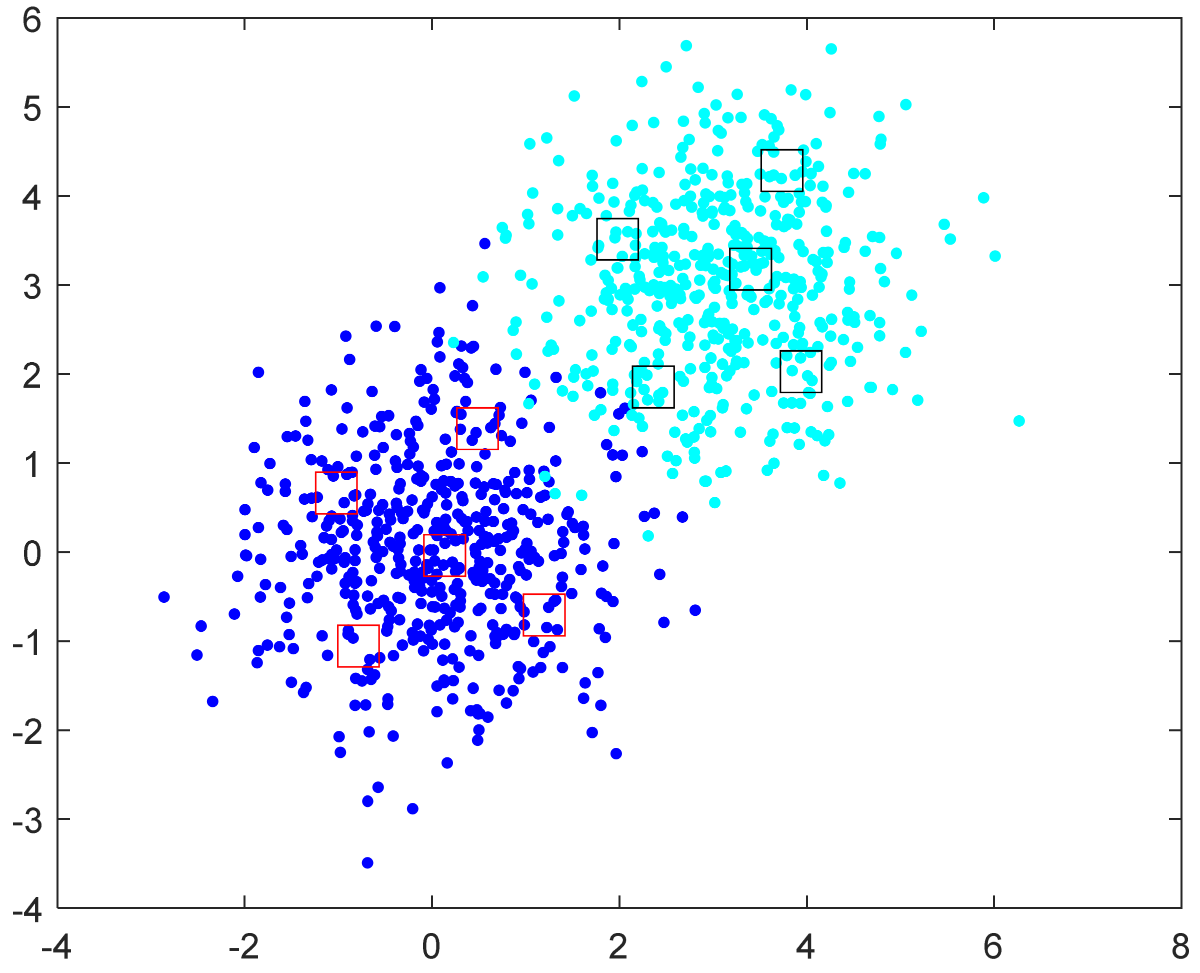


❑ Prototypes of training samples as center vectors

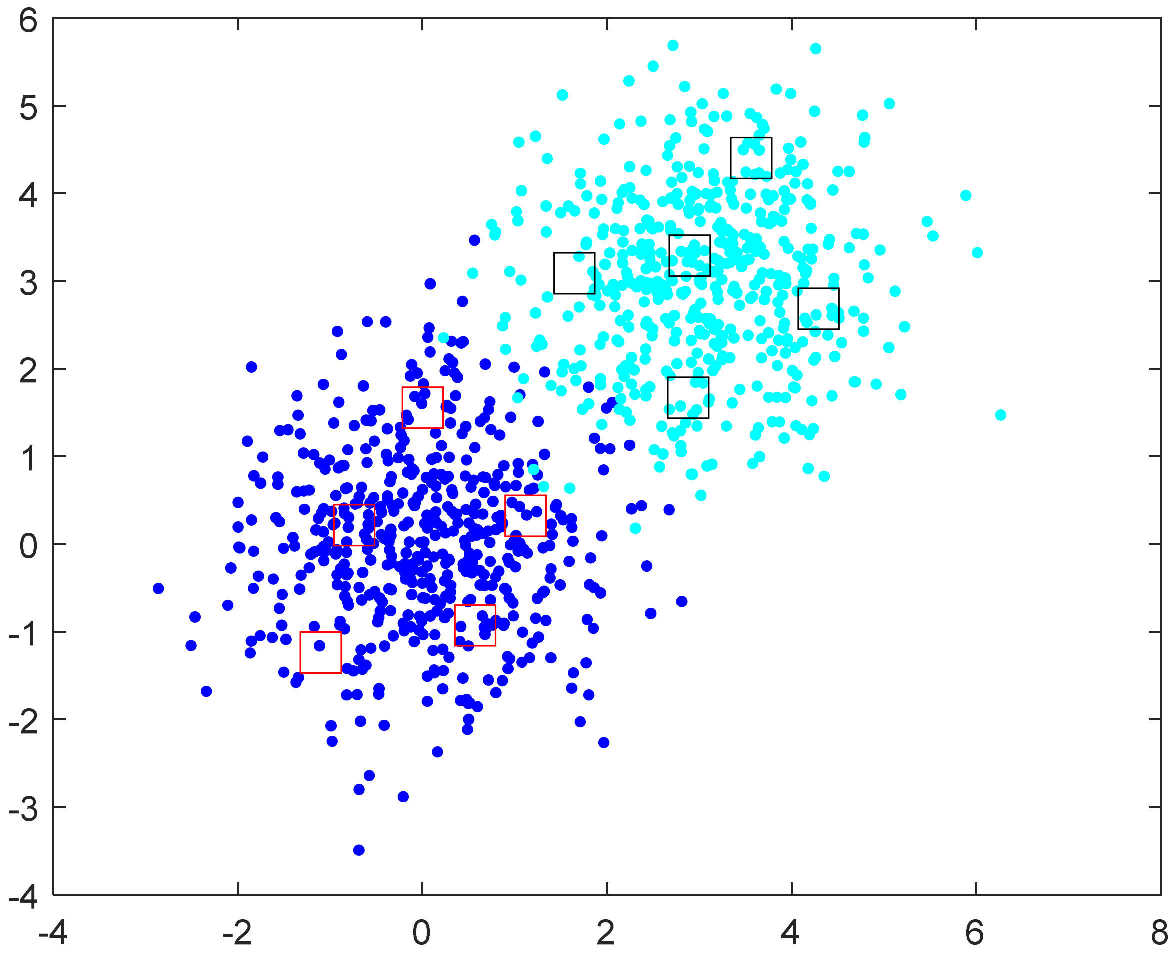
The randomly selected center vectors may not provide a good coverage of the samples. Then a natural solution to this problem is to select and use prototypes of training samples as neuron center vectors. In this approach, the basis functions are permitted to place the center vectors of the RBF hidden layer neurons in those regions of the input space where significant data are presented. The self-organizing map (SOM) neural networks introduced in previous chapter can be used to fulfill this task.

Besides SOM neural network, other clustering algorithms such as K-means clustering can also be used to select prototypes of the training samples.

5 prototypes for each class found by SOM neural network



5 prototypes for each class found by K-means clustering

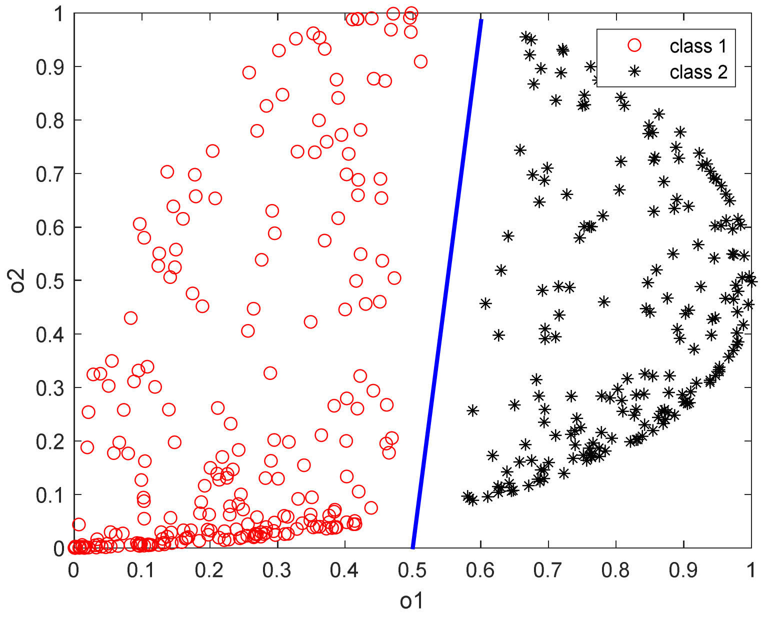


❑ Center vector selection as a feature selection problem

The hidden layer neurons map the samples from input space to hidden space where samples are expected to be linearly separable.

$$\begin{tikzcd} {} \arrow[r, "\mathbf{x}"] & \boxed{g(\|\mathbf{x}-\mathbf{c}_i\|)} \arrow[r, "\mathbf{o}"] & \boxed{\mathbf{w}^T\boldsymbol{o}+w_0} \arrow[r, "f(\mathbf{x})"] & {} \end{tikzcd}$$

Nonlinear mapping   Linear classifier



The hidden layer neuron with center vector $\mathbf{c}_j$ maps an input $\mathbf{x}(k)$ to the hidden space:

$$o_j(k) = \exp\left(-\frac{\left\|\mathbf{x}(k) - \mathbf{c}_j\right\|^2}{2\sigma^2}\right)$$

If we use all of the available training samples $\mathbf{x}(k), k = 1,2, N$, as center vectors, then an arbitrary sample $\mathbf{x}(k)$ will be mapped from the original n-dimensional input space to the N-dimensional hidden space: $\mathbf{x}(k) \in R^n \rightarrow \mathbf{o}(k) \in R^N$:

$$\mathbf{o}(k) = [o_1(k) \quad o_2(k) \quad \cdots \quad o_N(k)]^T$$

Where $o_1, o_2, \cdots, o_N$ are the new features in the hidden space. Our goal is to select m features (each of which corresponds to a centre vector) from the N features that produce the largest separability in the hidden space.

Assume *N* labelled training samples are given as:

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

Where $d(k)$ is the target value, i.e. class label in tasks of pattern classification, of $\mathbf{x}(k)$.

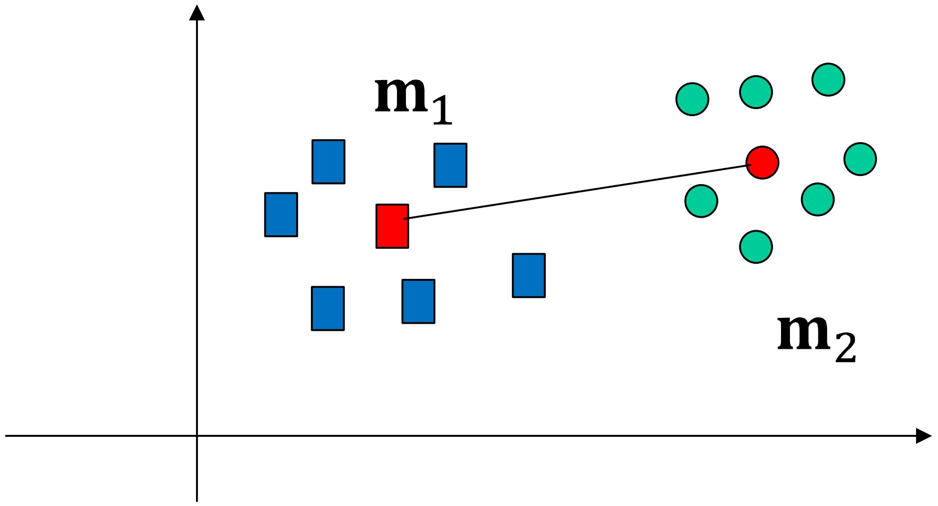
If all training samples are used as center vectors, then in the hidden space, we obtain the following labelled training samples:

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

There are a couple of metrics for evaluating the separability. Two commonly used separability measures are:

(i) Mahalanobis distance-based separability measure
(ii) Scatter-based separability measure

(i) Mahalanobis distance-based separability measure



The separability can be measured by Mahalanobis distance between two mean vectors in the space of the feature set:

$$J = (\mathbf{m}_1 - \mathbf{m}_2)^T \mathbf{C}^{-1} (\mathbf{m}_1 - \mathbf{m}_2)$$

Where $\mathbf{m}_1$ and $\mathbf{m}_2$ are mean vectors of the two classes, **C** is the covariance matrix.

(ii) Scatter-based separability measure

$$J = \mathrm{Tr}(\mathbf{S}_W^{-1}\mathbf{S}_B)$$

or

$$J = \frac{\mathrm{Tr}(\mathbf{S}_B)}{\mathrm{Tr}(\mathbf{S}_W)}$$

Where $\mathbf{S}_B$ and $\mathbf{S}_W$ are the between-class and within-class scatter matrices. $\mathrm{Tr}(\cdot)$ denotes the trace of a matrix defined as the sum of elements on the main diagonal of the matrix.

$$\boldsymbol{S}_B = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T$$

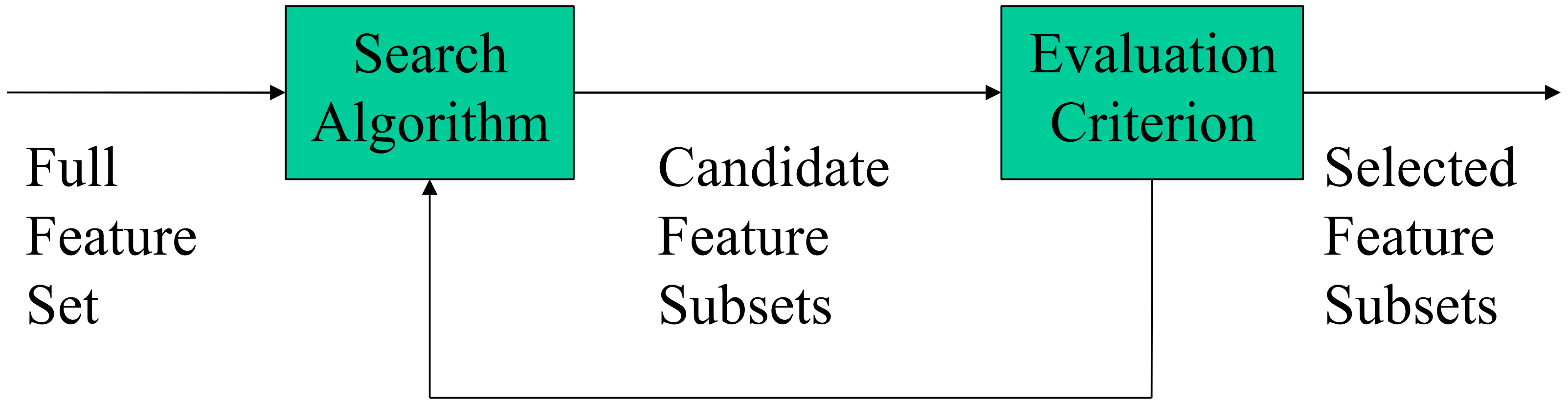
$$\boldsymbol{S}_W = \boldsymbol{S}_1 + \boldsymbol{S}_2$$

$$\mathbf{S}_i = \sum_{\mathbf{o} \in class_i} (\mathbf{o} - \mathbf{m}_i)(\mathbf{o} - \mathbf{m}_i)^T$$

A feature subset selection algorithm consists of two main components:

- Search algorithm
- Evaluation criterion

The goal of the search algorithm is to generate candidate feature subsets, while the evaluation criterion aims to evaluate the goodness of the candidate feature subsets generated by the search algorithm as illustrated below:



There are two typical search algorithms in feature subset selection, namely top-down and bottom-up.

- ❑ **Bottom-up**

The bottom-up method starts with an empty set, and build up the feature subset step by step, by adding the feature that contribute the most to the separability measure at each step. The sequential forward selection method belongs to this strategy.

- ❑ **Top-down method**

In this method, we start with a full set of features and then remove the features that do not contribute significantly to the separability measure step by step. The sequential backward elimination method belongs to this method

We next combine the feature evaluation measures with the search algorithms to form feature (i.e. centre vector) selection algorithms.

Sequential forward feature selection algorithm

We next combine the sequential forward search and the scatter-based separability measure to form a feature selection algorithm.

(1) In Step 1, consider each feature as a candidate,

$$F(i) = \{o_i\}, \qquad i = 1,2,\ldots,N$$

evaluate the separability measure:

$$J^{F(i)} = \frac{\mathrm{Tr}(\mathbf{S}_B^{F(i)})}{\mathrm{Tr}(\mathbf{S}_W^{F(i)})}$$

The feature that has the largest separability is selected as the first feature:

$$k = \mathrm{argmax}(J^{F(i)})$$

$$z_1 = o_k$$

(2) In Step 2, consider each of the remaining $N-1$ features as the candidate for the second feature,

$$F(i) = \{z_1, o_i\}, i = 1,2, \cdots N, i \neq k$$

evaluate the separability measure:

$$J^{F(i)} = \frac{\mathrm{Tr}(\mathbf{S}_B^{F(i)})}{\mathrm{Tr}(\mathbf{S}_W^{F(i)})}$$

The feature that leads to the largest separability is selected as the second feature.

(3) The above process is continued until m features are selected.

(4) The centre vectors corresponding to the m selected hidden space features are selected as the centre vectors of the m neurons in the RBF neural network hidden layer.

Sequential backward feature elimination algorithm

(1) In Step 1, consider each feature as the candidate to be eliminated:

$$F(i) = \{o_1, o_2, \dots, o_N\} - o_i, \qquad i = 1,2,\dots,N$$

evaluate corresponding separability measure:

$$J^{F(i)} = \frac{\mathrm{Tr}(\mathbf{S}_B^{F(i)})}{\mathrm{Tr}(\mathbf{S}_W^{F(i)})}$$

Identify the feature whose removal affects the separability measure the least as the first feature to be removed.

$$k = \mathrm{argmax}(J^{F(i)})$$

$$z_1 = o_k$$

(2) In Step 2, consider each of the remaining $N-1$ features as the candidate for the second feature to be removed:

$$F(i) = \{o_1, o_2, \dots, o_N\} - z_1 - o_i\ , i = 1,2,\cdots, N, i \neq k$$

evaluate the separability measure:

$$J^{F(i)} = \frac{\mathrm{Tr}(\mathbf{S}_B^{F(i)})}{\mathrm{Tr}(\mathbf{S}_W^{F(i)})}$$

The feature whose removal least affects the separability measure is selected as the second feature to be removed.

(3) The elimination process is continued until m features are left.

(4) The centre vectors corresponding to the m hidden space features are selected as the centre vectors of m neurons in the RBF neural network.

(2) 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 couple of methods such as linear discriminant analysis, linear support vector machine etc. Next, we use linear least square estimation algorithm to estimate the weights

Assume the *N* training samples are given as follows:

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

Where $d(k)$ is the target value (or class label in tasks of pattern classification) of $\mathbf{x}(k)$.

The center vector of each neuron at the hidden layer is already determined in previous step, thus for input vector $\mathbf{x}(k)$, we have:

$$f[\mathbf{x}(k)] = \sum_{j=1}^{m} w_j \exp\left(-\frac{\left\|\mathbf{x}(k) - \mathbf{c}_j\right\|^2}{2\sigma^2}\right) + w_0 = \sum_{j=1}^{m} w_j o_j(k) + w_0$$

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

$$\begin{bmatrix} f[\mathbf{x}(1)] \\ f[\mathbf{x}(2)] \\ \vdots \\ f[\mathbf{x}(N)] \end{bmatrix} = \begin{bmatrix} 1 & o_1(1) & o_2(1) & \cdots & o_m(1) \\ 1 & o_1(2) & o_2(2) & \cdots & o_m(2) \\ 1 & \vdots & \vdots & \ddots & \vdots \\ 1 & o_1(N) & o_2(N) & \cdots & o_m(N) \end{bmatrix} \times \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_m \end{bmatrix}$$

Define:

$$\mathbf{f} = \begin{bmatrix} f[\mathbf{x}(1)] \\ f[\mathbf{x}(2)] \\ \vdots \\ f[\mathbf{x}(N)] \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_m \end{bmatrix}$$

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

We have:

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

We wish to find such **w** that the following loss function is minimized:

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

Where

$$\mathbf{d} = [d(1) \quad d(2) \quad \cdots \quad d(N)]^T$$

Solve

$$\frac{\partial J}{\partial \mathbf{w}} = 0$$

We obtain:

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

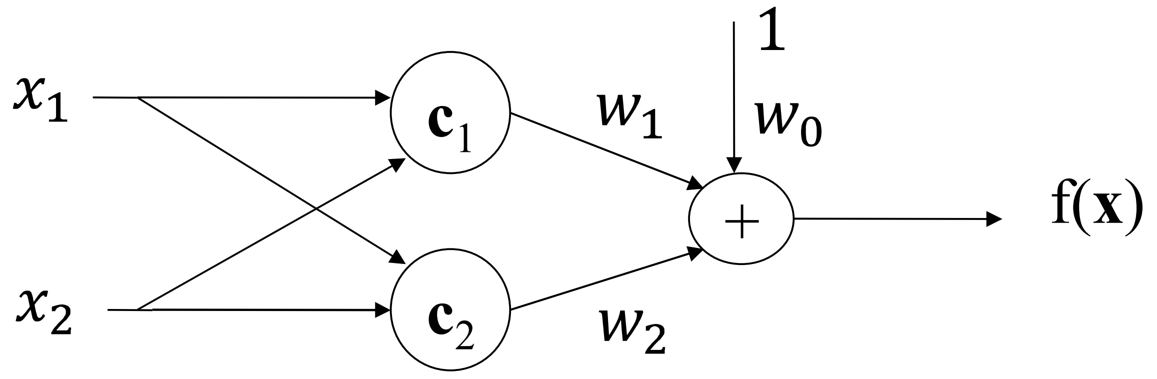
The weights thus obtained are called linear least square estimates.

Example

Consider the exclusive OR (XOR) problem. 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 center vectors of the 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.



$$f[\mathbf{x}(k)] = \sum_{j=1}^{2} w_j \exp\left(-\frac{\left\|\mathbf{x}(k) - \mathbf{c}_j\right\|^2}{2\sigma^2}\right) + w_0$$

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

$$\begin{bmatrix} f[\mathbf{x}(1)] \\ f[\mathbf{x}(2)] \\ f[\mathbf{x}(3)] \\ f[\mathbf{x}(4)] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0.1353 \\ 1 & 0.3678 & 0.3678 \\ 1 & 0.1353 & 1 \\ 1 & 0.3678 & 0.3678 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \mathbf{\Phi w}$$

Define

$$\mathbf{d} = \begin{bmatrix} d(1) \\ d(2) \\ d(3) \\ d(4) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

Then the weights are obtained as follows:

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = (\mathbf{\Phi}^T \mathbf{\Phi})^{-1} \mathbf{\Phi}^T \mathbf{d}$$

$$= \left(\begin{bmatrix} 1 & 1 & 0.1353 \\ 1 & 0.3678 & 0.3678 \\ 1 & 0.1353 & 1 \\ 1 & 0.3678 & 0.3678 \end{bmatrix}^T \begin{bmatrix} 1 & 1 & 0.1353 \\ 1 & 0.3678 & 0.3678 \\ 1 & 0.1353 & 1 \\ 1 & 0.3678 & 0.3678 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 1 & 0.1353 \\ 1 & 0.3678 & 0.3678 \\ 1 & 0.1353 & 1 \\ 1 & 0.3678 & 0.3678 \end{bmatrix}^T \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2.8404 \\ -2.5018 \\ -2.5018 \end{bmatrix}$$

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

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

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

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

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

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

(3) Nonlinear optimization-based center vector and weight estimation

In this approach, the center vectors, weights and width of the radial basis function undergo a single learning process. A natural candidate for such a process is the gradient-descent procedure.

The loss function is defined as:

$$E = \frac{1}{2}\sum_{k=1}^{N} e_k^2$$

where e_k is the error signal between the desired output $d(k)$ and the network output $f[\mathbf{x}(k)]$:

$$e_k = f[\mathbf{x}(k)] - d(k)$$

By minimizing the loss 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 method is summarized as follows:

(1) Weight estimation:

$$\frac{\partial E(n)}{\partial w_j(n)} = \sum_{k=1}^{N} e_k(n) \exp\left(-\frac{\left\|\mathbf{x}(k) - \mathbf{c}_j(n)\right\|^2}{2\sigma^2(n)}\right), j = 1,2, \dots, m.$$

$$\frac{\partial E(n)}{\partial w_0(n)} = \sum_{k=1}^{N} e_k(n)$$

$$w_j(n+1) = w_j(n) - \eta_1 \frac{\partial E(n)}{\partial w_j(n)}, j = 0,1,2 \dots, m.$$

Where η_1 is the step-size.

(2) Center vector estimation

$$\frac{\partial E(n)}{\partial \mathbf{c}_j(n)} = w_j(n) \sum_{k=1}^{N} e_k(n) \exp\left(- \frac{\left\| \mathbf{x}(k) - \mathbf{c}_j(n) \right\|^2}{2\sigma^2(n)} \right) \frac{\mathbf{x}(k) - \mathbf{c}_j(n)}{\sigma^2(n)}$$

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

(3) Width estimation

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

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

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

Application example of RBF neural networks

Direct mailings to potential customers can be very effective to market a product or a service. However, many of this mails are of no interest to the people to receive it. And most of the mails are thrown away, which 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 solve this problem by using the machine learning method as follows:

(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 (test data) and select those that have high possibility to be interested.

An insurance company wishes to launch a caravan insurance plan. The task is to predict which customers are interested.

(1) Training data set

The training data set contains 5822 customers. The record of each customer consists of 85 attributes (features) including socio-demographic data and caravan policy ownership (class label). The socio-demographic data is derived from zip-codes. 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 provided. The problem here is to select 800 customers that are most interested in the policy from the 4000 customers.

The key task here is 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), which is detrimental to classifier training. To address this problem, we can select 500 typical customers from the 5474 customers of class 2 by using SOM neural networks.

Another problem with the training data is that, among the 85 attributes available, many are irrelevant, insignificant or redundant. We may perform **feature selection** by eliminating the irrelevant, insignificant and redundant attributes, and keeping the useful ones only. Finally, 12 attributes are selected (details are omitted here).

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

To select center vectors, we first consider each of the 848 (348+500) training samples as candidates, and then use the sequential forward selection method to select 7 neurons (why 7 here? Because the separability measure has trivial improvement after 7 neurons are selected).

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 training sample $\mathbf{x}(k)$, with a value of 1 (class 1) or -1 (class 2). $f[\mathbf{x}(k)]$ is the RBF neural network output for $\mathbf{x}(k)$.

With the RBF neural network classifier trained, we predict the possibility of each of the 4000 test customers and select 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.

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

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

The above example clearly shows the advantage of using machine learning method to address business problems.