Neural Networks: Radial Basis Function Network

Andrzej Kordecki

Neural Networks (ML.ANK385 and ML.EM05): Lecture 08
Division of Theory of Machines and Robots
Institute of Aeronautics and Applied Mechanics
Faculty of Power and Aeronautical Engineering
Warsaw University of Technology

Table of Contents

Radial-Basis Function Network

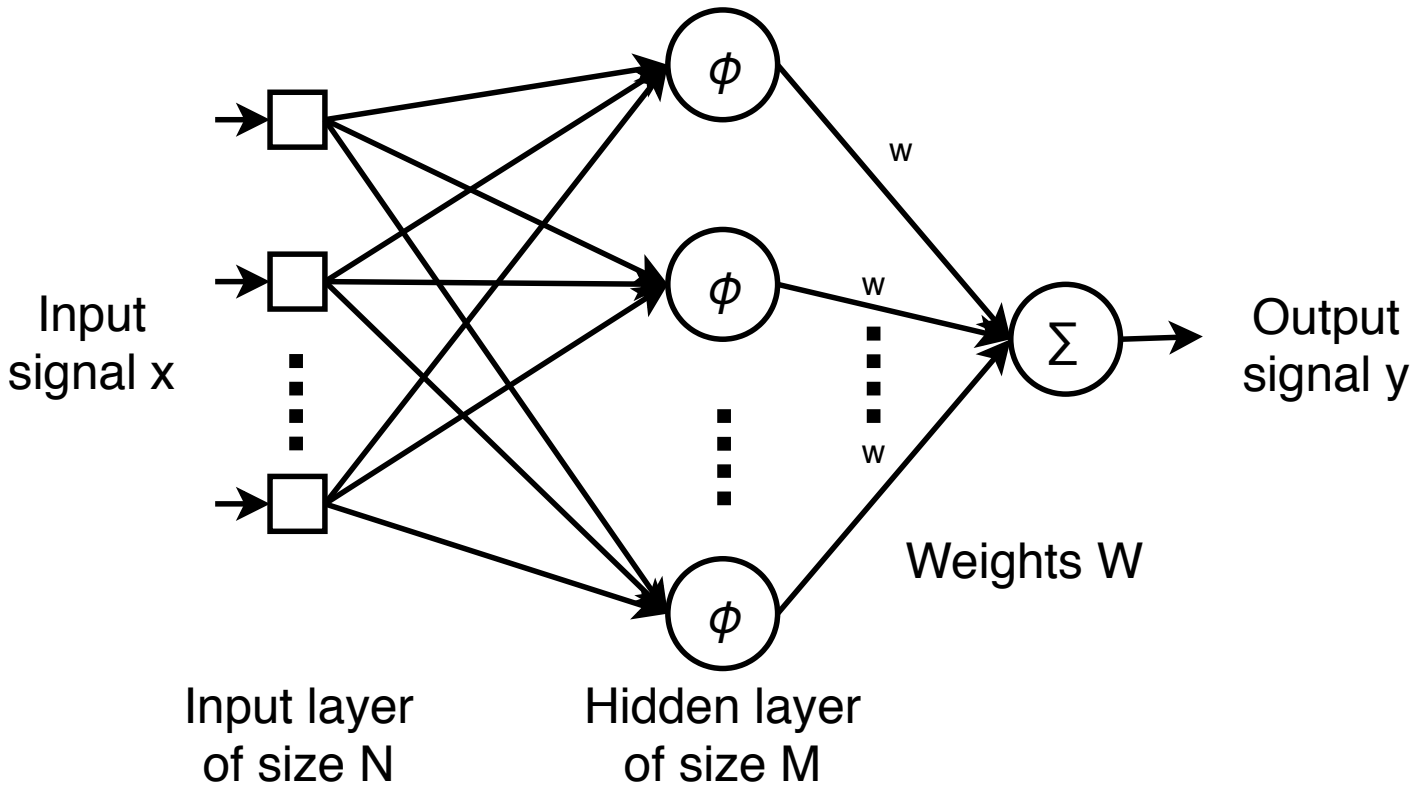
Introduction to Radial Basis Functions

The idea of Radial Basis Function (RBF) Networks derives from the theory of function approximation. The RBF main features are:

- RBF networks have many uses, including function approximation, classification, and system control,
- Represent feedforward network with an input layer, a single hidden layer, and an output layer usually consisting of a single unit.
- Designed to perform a nonlinear mapping from the input space to the hidden space, followed by a linear mapping from the hidden space to the output space,

Radial-Basis Function Network

The feedforward network with an input layer, a single hidden layer, and an output layer consisting of a single unit.



The links between source nodes and the hidden units are direct connections with no weights.

Interpolation

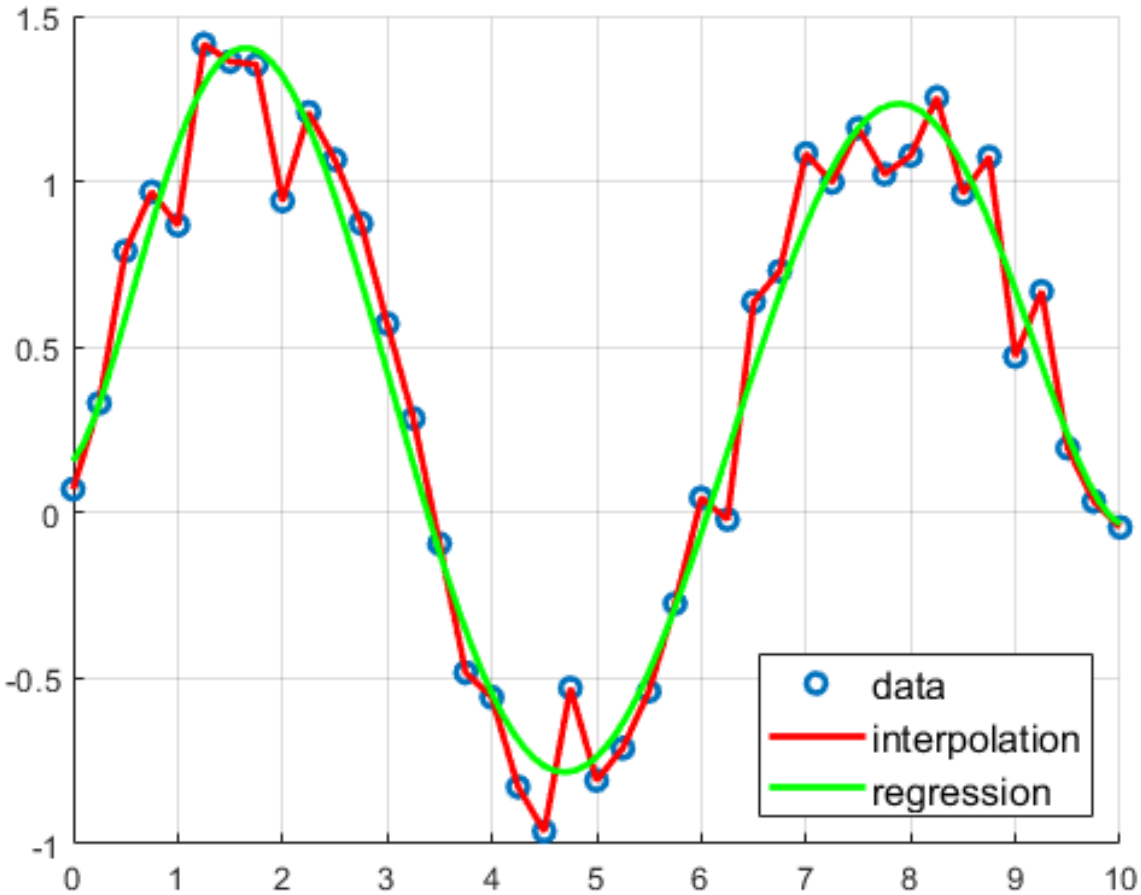
The interpolation of a set of N training data points (x_p, t_p) requires every one of input vectors $\{x_p\}$ to be mapped onto the corresponding set of N real target output $\{t_p\}$. The goal is to find a function $f(x)$ that satisfies the interpolation condition:

$$f(x_p) = t_p, \quad p = 1, 2, ...N$$

Let's define a function $\varphi(x)$ as a hidden non-linear function, which plays a similar role to that of a hidden unit in a feedforward neural network. The space spanned by the set of hidden functions $\{\varphi(x)\}$ is referred to as the feature space.

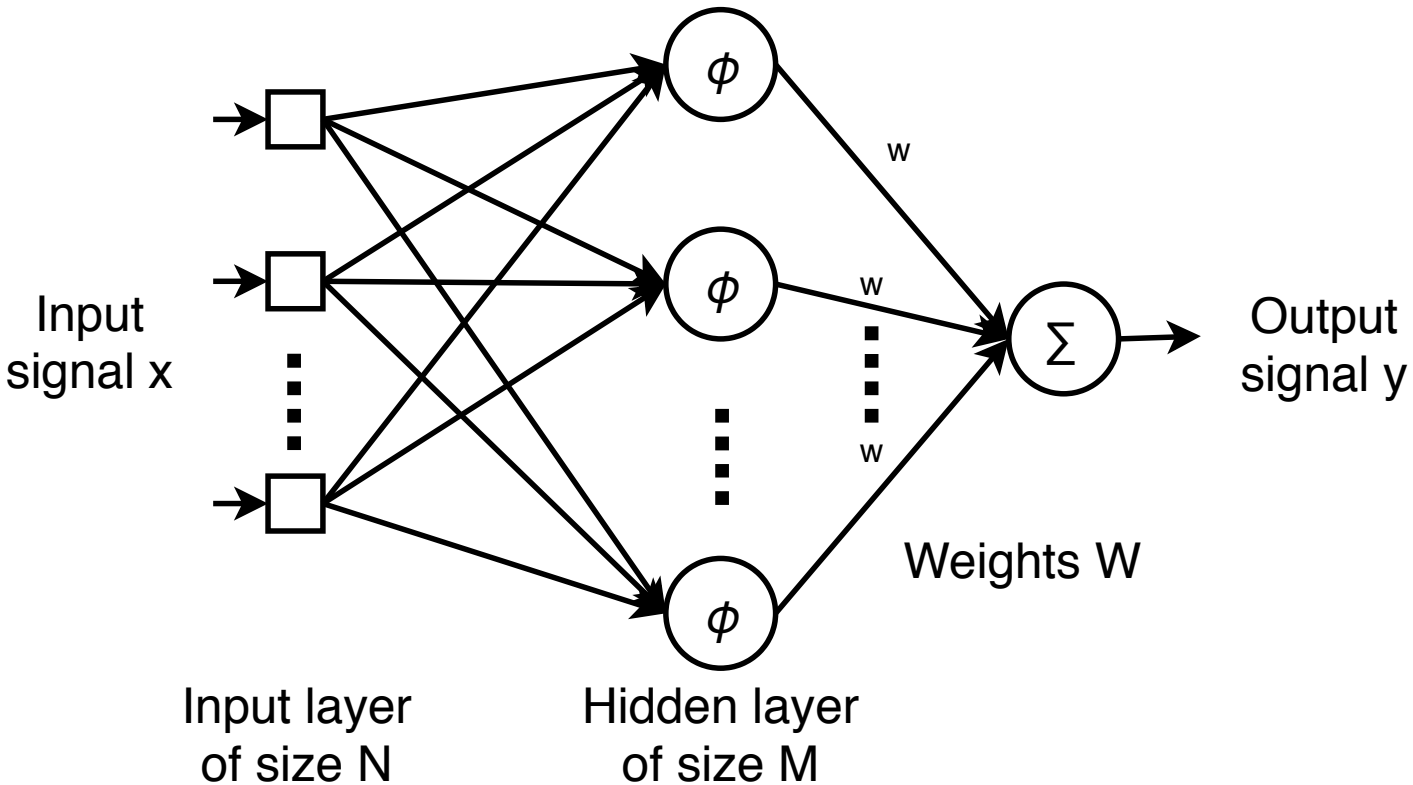
Interpolation

The interpolation is a special case of regression where the output function goes exactly through all the data points.



Radial-Basis Function Network

The model of the RBF have mostly defined structure.



Interpolation

The RBF method consists of choosing a function F that has the form:

$$F(x) = \sum_{i=1}^{M} w_i \varphi(||x - x_i||)$$

where $\varphi(x - x_i)$ is a set of M arbitrary, one for each data point (in interpolation $M = N$), generally nonlinear functions, known as radial-basis functions, and $||x - x_i||$ denotes a norm that is usually Euclidean between x and x_i.

Determining the Weights

The output of RBF network is a linear combination of the basis functions. The idea is to find the "weights" w such that the interpolation function goes through the data points in following way:

$$F(x_p) = \sum_{i=1}^{N} w_i \varphi(||x_p - x_i||) = t_p$$

We have a set of simultaneous linear equations for the unknown weights (in interpolation $M = N$):

$$\begin{bmatrix} \varphi_{1,1} & \varphi_{1,2} & \cdots & \varphi_{1,M} \\ \varphi_{2,1} & \varphi_{2,2} & \cdots & \varphi_{2,M} \\ \cdots & \cdots & \cdots & \cdots \\ \varphi_{N,1} & \varphi_{N,2} & \cdots & \varphi_{N,M} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ . \\ w_M \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ . \\ t_N \end{bmatrix}$$

Determining the Weights

We can write this in matrix form by defining the vectors $t = \{t_p\}$ and $w = \{w_p\}$, and the matrix $\Phi = \{\varphi_{p,q} = \varphi(||x_q - x_p||); p, q = 1, 2, ..., N\}$. This simplifies the equation to:

$$\Phi w = t$$

If the inverse of Φ exists, we can use any standard matrix inversion techniques to estimate weights:

$$w = \Phi^{-1} t$$

Radial Basis Functions

The Micchelli theorem prove:

Let $\{x_p\}$ be a set of distinct points in R^{M_0} space. Then the $N \times N$ interpolation matrix Φ, whose pq-th element is $\varphi_{p,q} = \varphi(||x_q - x_p||)$, is nonsingular.

It can be shown that, for a large class of basis functions $\varphi(x)$, the matrix Φ is indeed nonsingular (hence invertable) providing the data points are distinct. There is a large class of radial-basis functions that is covered by Micchelli's theorem.

Radial Basis Functions

Some of the most commonly used basis functions $\varphi(r)$ are:

- Gaussian functions - it is an important member of the class of radial-basis functions:

$$\varphi(r) = \exp(-\frac{r^2}{2\sigma^2}) \quad \text{where} \quad \sigma > 0$$

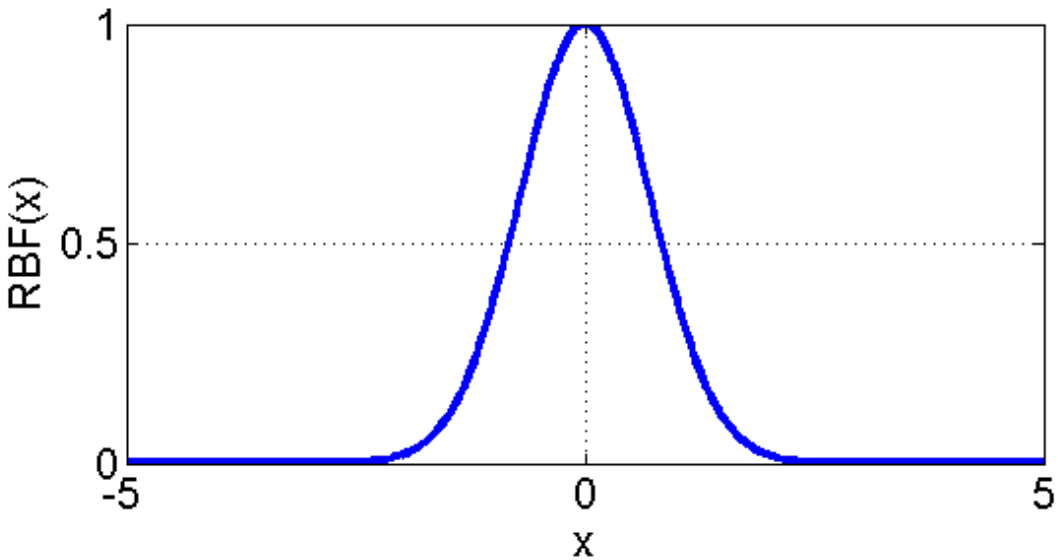
- Cubic function:

$$\varphi(r) = r^3$$

- Linear function:

$$\varphi(r) = r$$

Gaussian Function

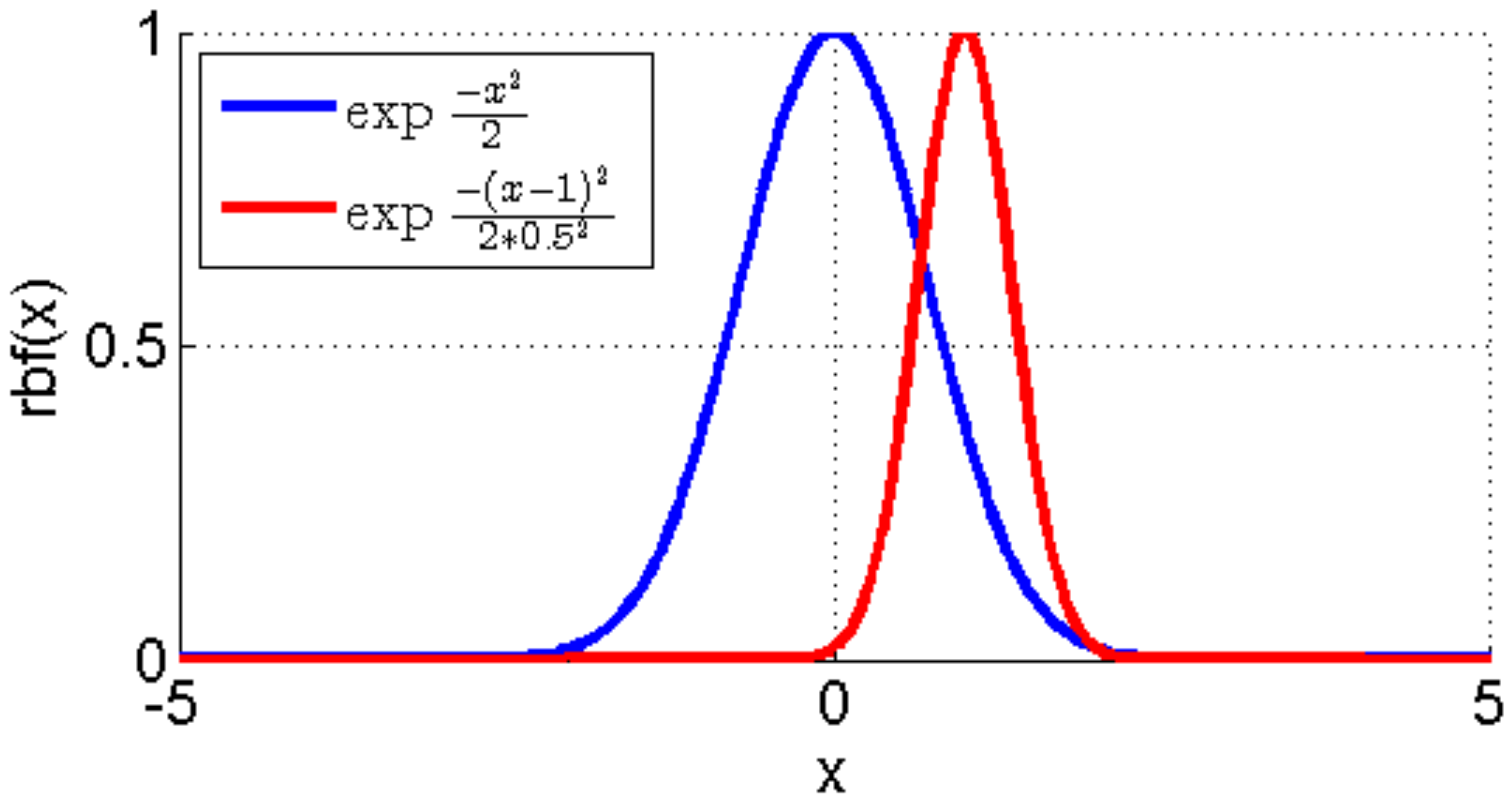


- Normalized neuron output:
  $y = \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{K} (x_i - \mu_i)^2 \right)$

- Continuous,
- differentiable,
- non-monotonic,
- bounded,
- nonlinear

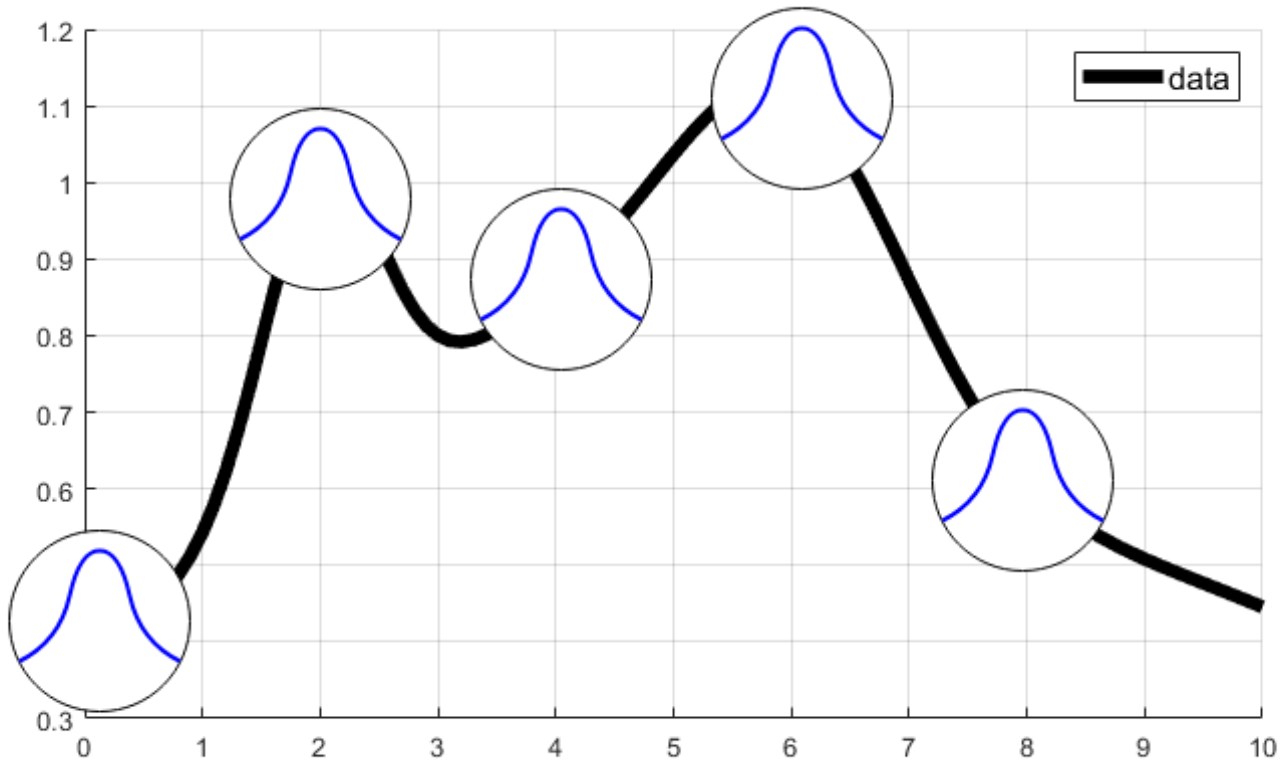
Gaussian Function

Example: Weights impact on Gaussian function for $K = 1$, $\sigma = 1$ and $\sigma = 0.5$.



Gaussian Function

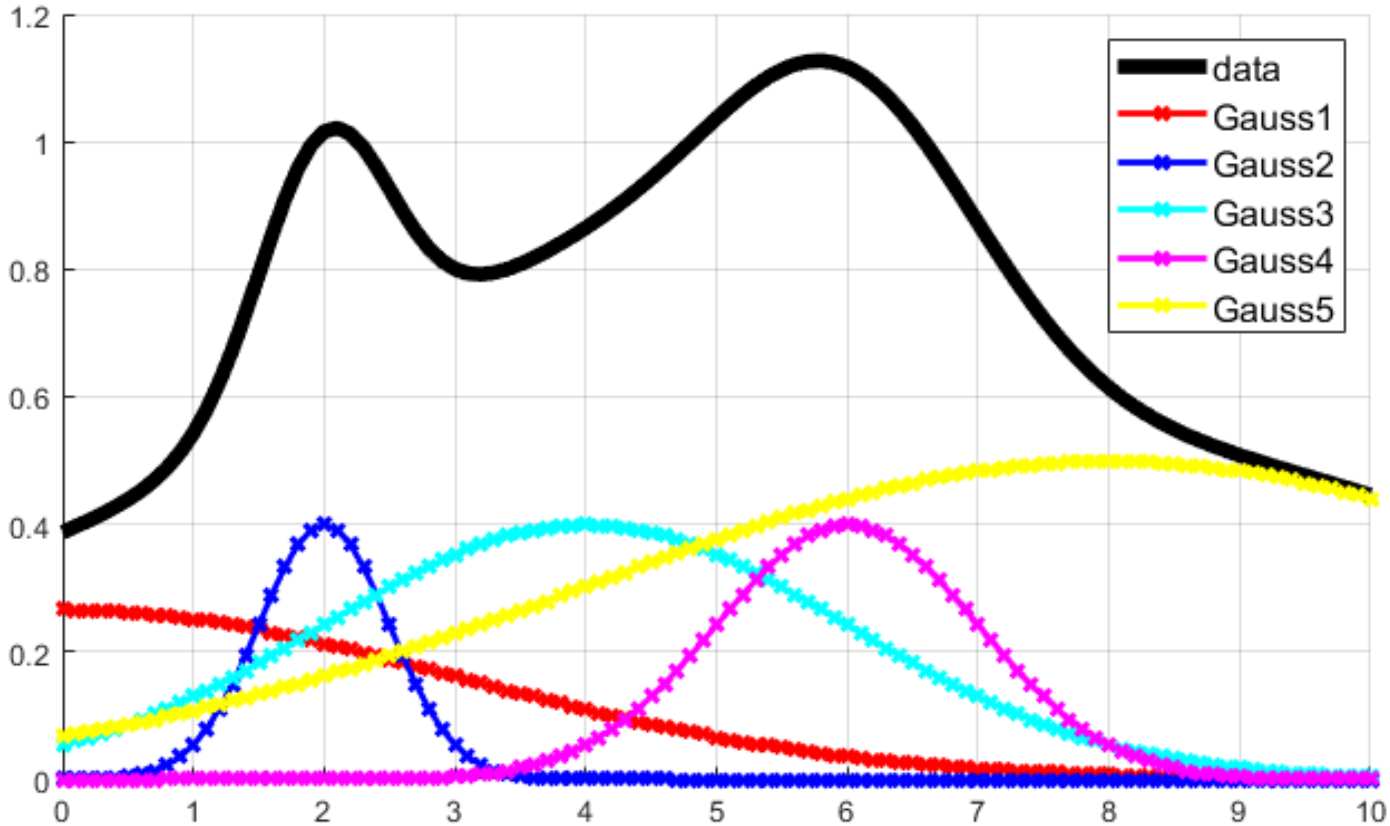
Example: Interpolation with use of linear combination of 5 Gaussian functions.



We need to find Gaussian function parameters and weights.

Gaussian Function

Example: Solution of interpolation.



Radial Basis Functions

- Multiquadrics functions:

$$\varphi(r) = (r^2 + \sigma^2)^{0.5} \quad \text{where} \quad \sigma > 0$$

- Inverse multiquadric functions:

$$\varphi(r) = \frac{1}{(r^2 + \sigma^2)^{0.5}} \quad \text{where} \quad \sigma > 0$$

- Generalized multiquadrics functions:

$$\varphi(r) = (r^2 + \sigma^2)^{\beta} \quad \text{where} \quad \sigma > 0, 1 > \beta > 0$$

- Generalized inverse multiquadric functions:

$$\varphi(r) = \frac{1}{(r^2 + \sigma^2)^{\alpha}} \quad \text{where} \quad \sigma > 0, \alpha > 0$$

Properties of the Radial Basis Functions

The Gaussian and Inverse Multi-Quadric Functions are localized in the sense that:

$$\phi(r) \to 0 \quad \text{as} \quad |r| \to \infty$$

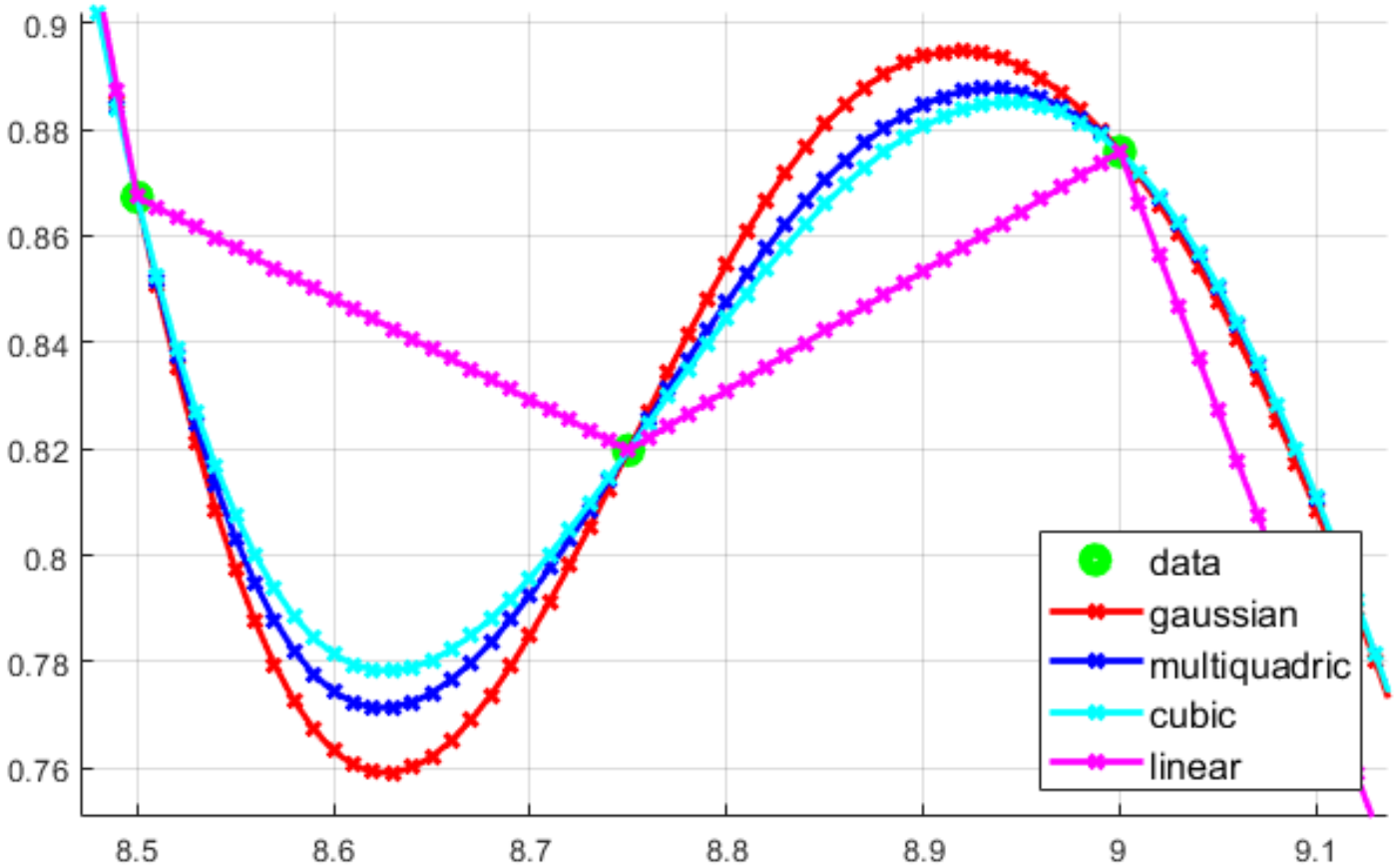
All the other functions above have the property

$$\phi(r) \to \text{inf} \quad \text{as} \quad |r| \to \infty$$

Note that even the Linear Function $\varphi(r) = r = ||x - x_p||$ is still non-linear in the components of x. In one dimension, this leads to a piecewise-linear interpolating function which performs the simplest form of exact interpolation.

Radial Basis Function Networks

RBF interpolating with different basis functions.



Problems with Exact Interpolation

The two problems of exact interpolation networks:

- They perform poorly with noisy data. As we have already seen for MLP, we do not usually want the network's outputs to pass through all the data points when the data is noisy, because that will be a highly oscillatory function that will not provide good generalization.
- They are not computationally efficient. The network requires one hidden unit for each training data pattern, and so for large data sets the network will become very costly to evaluate.

With MLPs we can improve generalization by using more training data – the opposite happens in RBF networks, and they take longer to compute as well.

Improving RBF Networks

We can improve the basic structure of the RBF networks:

- The number M of basis functions (hidden units) do not need to be equal to the number N of training data points. In general it is better to have M much less than N.
- The centers of the basis functions do not need to be defined as the training data input vectors. They can instead be determined by a training algorithm.
- The basis functions do not need to have all the same width parameter σ. It can also be determined by a training algorithm.

The Improved RBF Network

When these changes are made to the exact interpolation formula, and we allow the possibility of more than one output unit, we arrive at the RBF network mapping

$$y = w_0 + \sum_{j=1}^{M} w_j \varphi_j(x)$$

which we can simplify by introducing an extra basis function $\varphi_0 = 1$ to give:

$$y = \sum_{j=0}^{M} w_j \varphi_j(x)$$

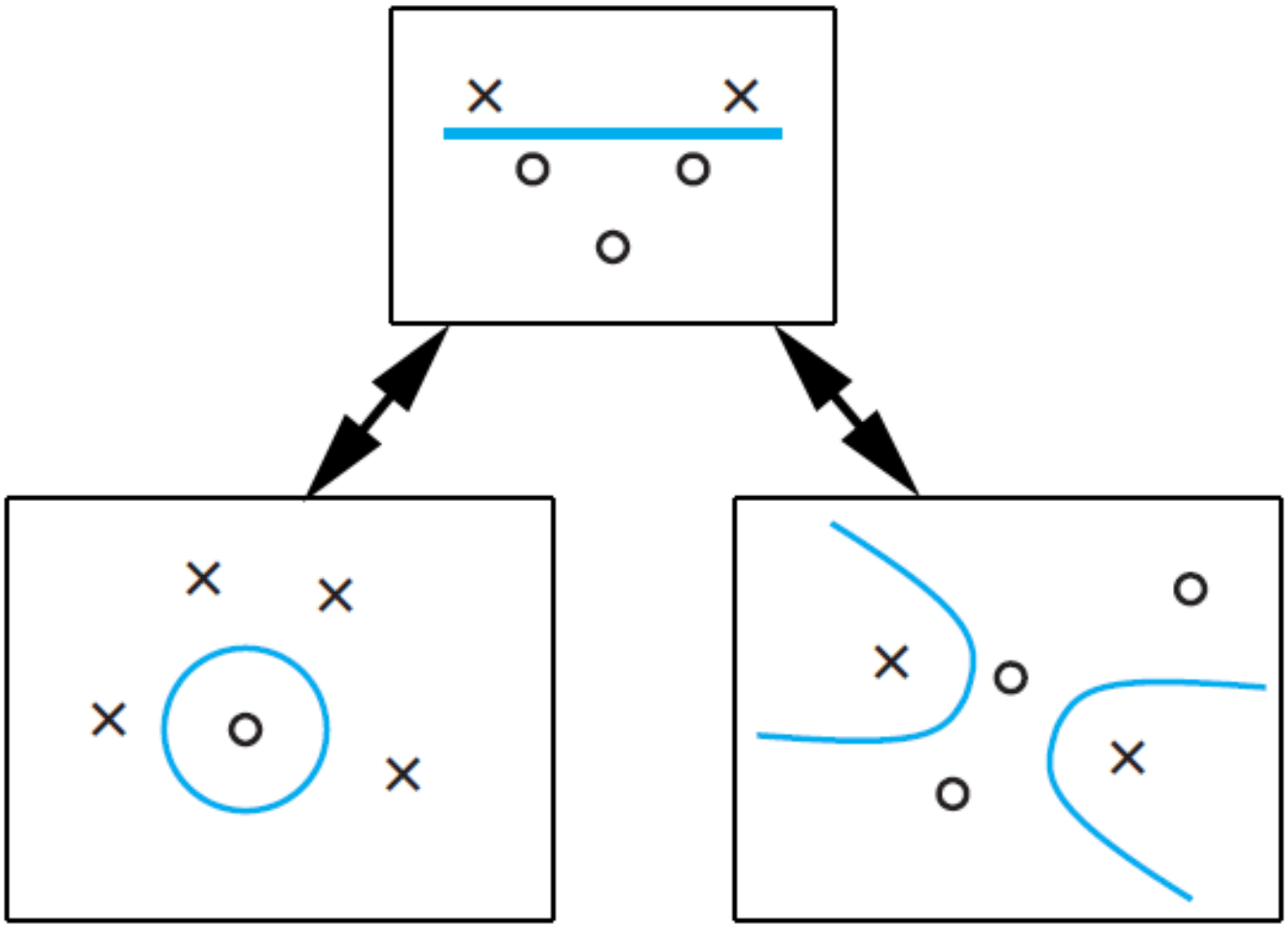
The RBF Networks Training

Classification of nonlinearly separable problem

The problem of classifying nonlinearly separable patterns can be proceed in two stages:

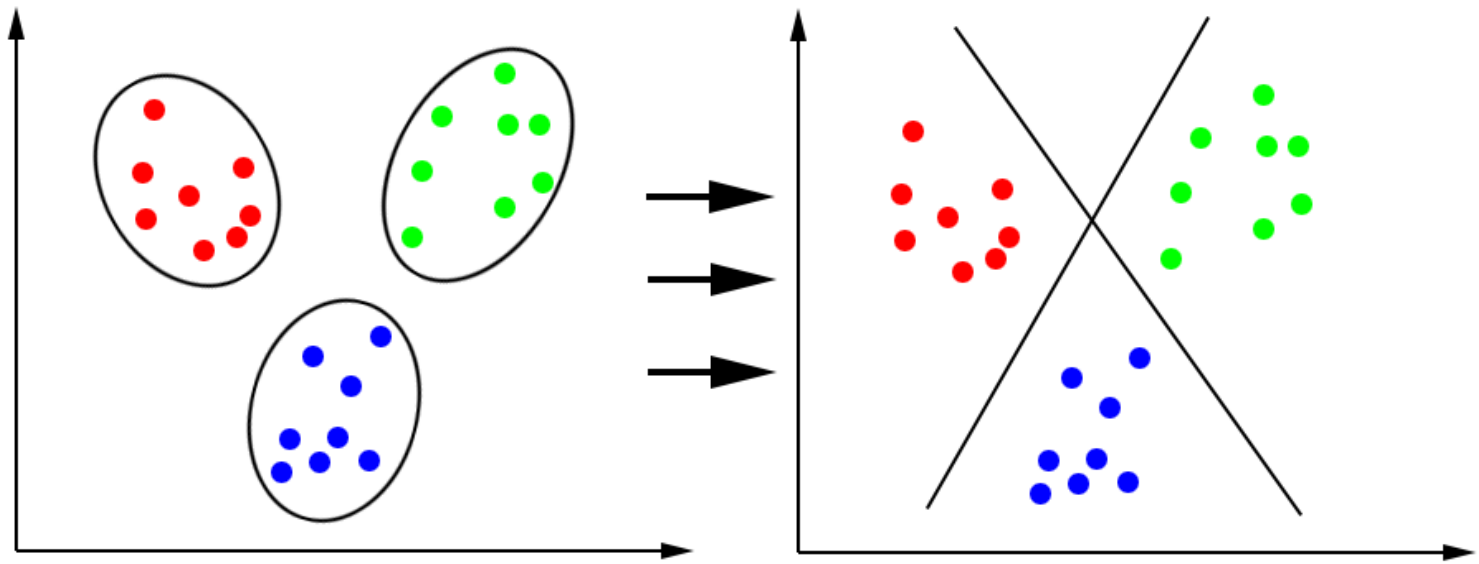
- The first stage transforms a given set of nonlinearly separable patterns into a new set for which, under certain conditions, the likelihood of the transformed patterns becoming linearly separable is high,
- The second stage completes the solution to the prescribed classification problem by using least-squares estimation.

Classification of nonlinearly separable problem



Classification of nonlinearly separable problem

An alternative to MLP hyper-planes classes separation is approach to model the separate class distributions by localized radial basis functions.



Classification

The underlying justification is found in Cover's theorem on the separability of patterns:

A complex pattern-classification problem, cast in a high-dimensional space nonlinearly, is more likely to be linearly separable than in a low-dimensional space, provided that the space is not densely populated.

The problem is basically solved by first transforming it into a high dimensional space in a nonlinear manner and then linear separating the classes in the output layer. The RBF network is used to perform a complex pattern classification task.

Classification

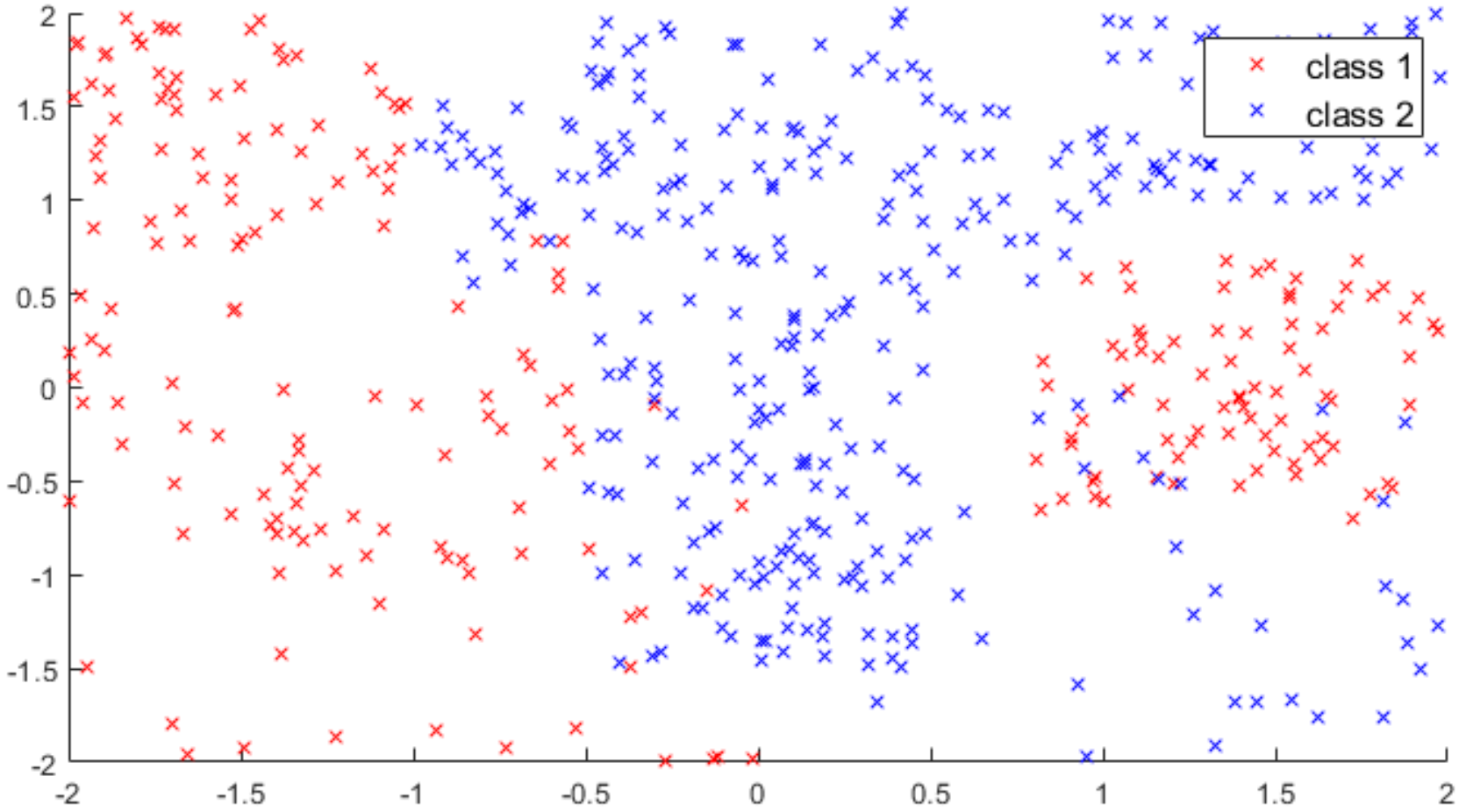
In principle, it is easy to have an RBF network perform classification – we only need training data. The output function $y_k(x)$ for each class k with appropriate targets:

$$t_k^p = \begin{cases} 1 & \text{if pattern belongs to class } k \\ 0 & \text{otherwise} \end{cases}$$

In addition to the RBF network outputting good classifications, it can also provide estimates of the posterior class probabilities.

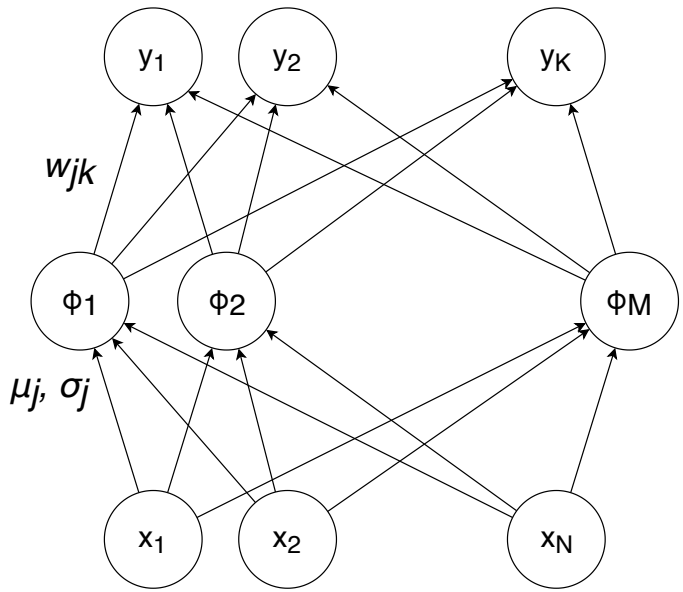
Classification

Problem of classification to 2 categories.



The RBF Network Architecture

In the RBF Network, the hidden to output layer part operates like a standard feed-forward MLP network, with the sum of the weighted hidden unit activations giving the output unit activations.



The hidden unit activations are given by the RBF $\varphi_j(x, \mu_j, \sigma_j)$, which depend on the "weights" μ_j, σ_j and input x.

The RBF Networks Training

The goal is to approximate the some underlying classification functions $f_k(x)$ with functions $y_k(x)$ of the form:

$$y_k = \sum_{j=0}^{M} w_{j,k} \varphi_j(x)$$

We shall concentrate on the case of Gaussian basis functions:

$$\varphi_j(x) = \exp\left(- \frac{||x - \mu_j||^2}{2\sigma_j^2} \right)$$

Network parameters to determine: M, $w_{k,j}$, μ_j and σ_j.

The RBF Networks Training

The RBF Networks Training:

- In designing the RBF network a key issue that needs to be addressed is how to compute the parameters of basis function units that constitute the hidden layer by using unlabeled data. In other words, the computation is to be performed in an unsupervised manner. When the input to hidden "weights" are found, they are kept fixed while the hidden to output weights are learned.

The RBF Networks Training

The RBF Networks Training:

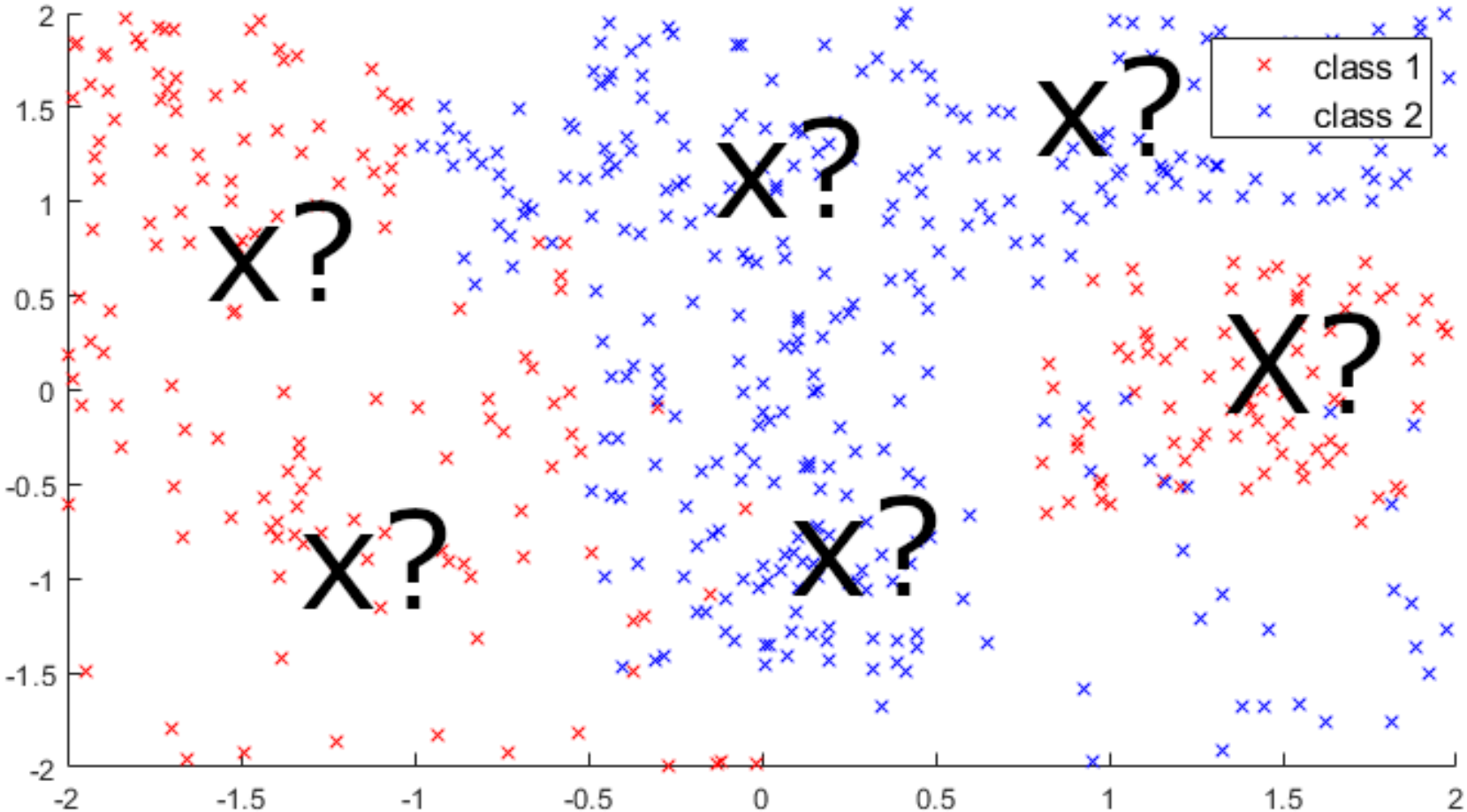
- The second stage of training involves just a single layer of weights $w_{k,j}$ and linear output activation functions. The weights can easily be found analytically by solving a set of linear equations. This can be done very quickly, without the need for a set of iterative weight updates as in gradient descent learning. But, still the use of back-propagation algorithm is one of the option of NN training.

Basis Function Optimization

- One major advantage of RBF networks is the possibility of choosing suitable hidden unit/basis number and function parameters without having to perform a full non-linear optimization of the whole network.
- Clustering is a form of unsupervised learning whereby a set of observations (i.e., data points) is partitioned into natural groupings or clusters of patterns in such a way that the measure of similarity between any pair of observations assigned to each cluster minimizes a specified cost function. The clustering methods:
  - Fixed centers selected at random
  - K-means clustering

Basis Function Optimization

How many hidden units should we use in classification problem? What should be there parameters?



Fixed Centers Selected At Random

The simplest approach to setting the RBF parameters:

- Set the centers (μ) fixed at M points selected at random from the N data points,
- Set all their widths (σ) to be equal and fixed at an appropriate size for the distribution of data points.

Specifically, we can use normalized centers at μ_j defined:

$$\varphi_j(x) = \exp\left(-\frac{||x - \mu_j||^2}{2\sigma_j^2} \right)$$

where $\mu_j \subset x^p$ and σ_j are all related in the same way to the maximum or average distance between the chosen centers μ_j.

K-Means Clustering

Setting the RBF parameters with use of K-Means clustering:

- In the k-means clustering, we need to propose a K number of clusters that data is to be partitioned into a proposed number of subsets, where K is smaller than the number of observations.
- The number of subset must be assumed based on knowledge apriori or found with use of other methods.
- But in general, each cluster is represented by cluster centers (centroids) μ with initial values usually generated randomly,

K-Means Clustering

The procedure to partition the data points x_p into K disjoint sub-sets S_j containing N_j data points to minimize the sum squared clustering function

$$J = \sum_{j=1}^{K} \sum_{p \in S_j} ||x_p - \mu_j||^2$$

where μ_j is the mean/centroid of the data points in set S_j given by

$$\mu_j = \frac{1}{N_j} \sum_{p \in S_j} x_p$$

Once the basis centres have been determined in this way, the widths can then be set according to the variances of the points in the corresponding cluster.

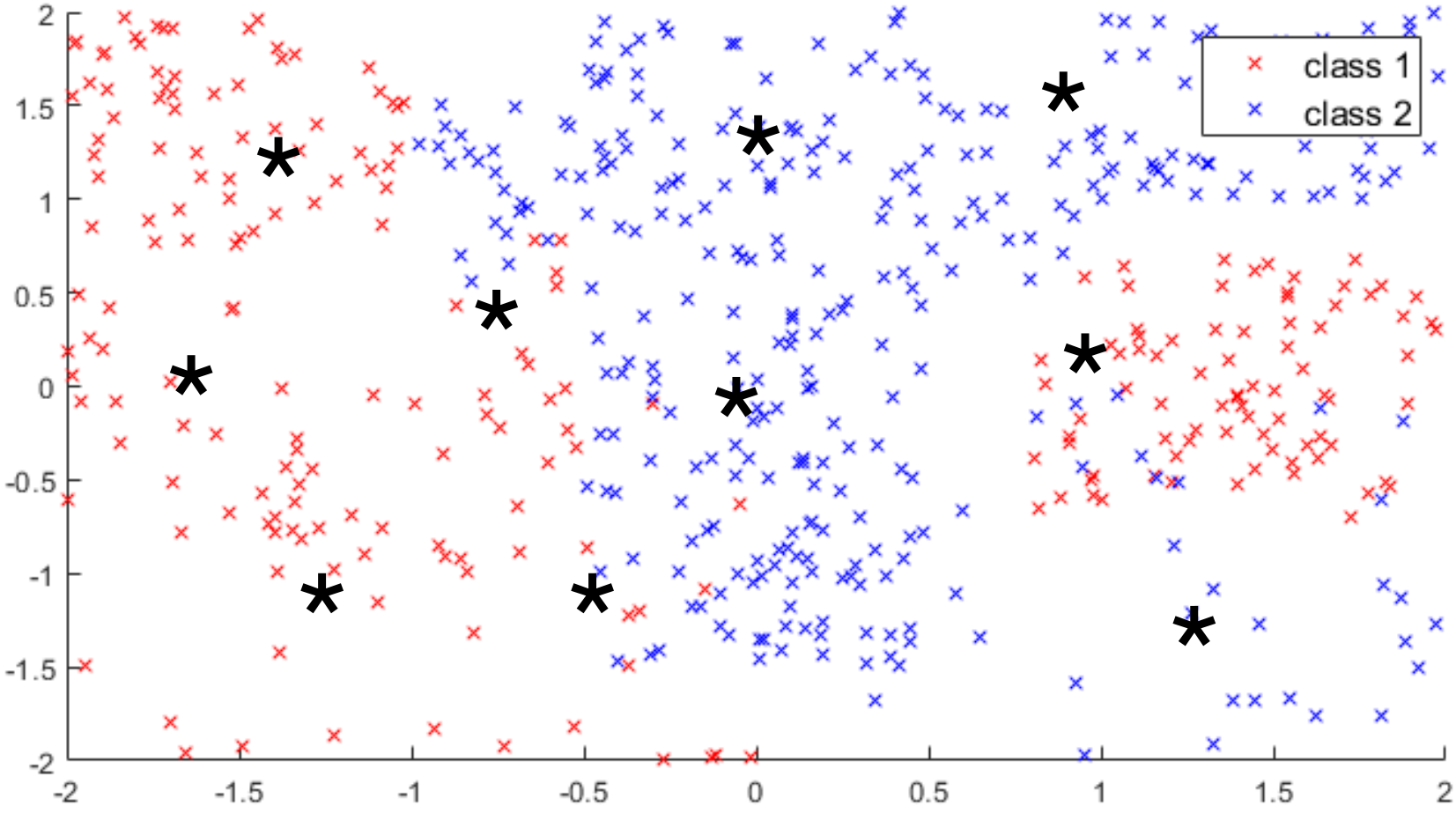
K-Means Clustering

K-Means algorithm:

1. Choose an initial number of clusters and assign to each of them a initial value.
2. Calculate all distances between each data sample and each cluster center.
3. Assign each data sample to the group that has the closest cluster center.
4. When all data samples have been assigned, recalculate the positions of cluster center.
5. Repeat steps 2-4 until thecluster centers no longer move.

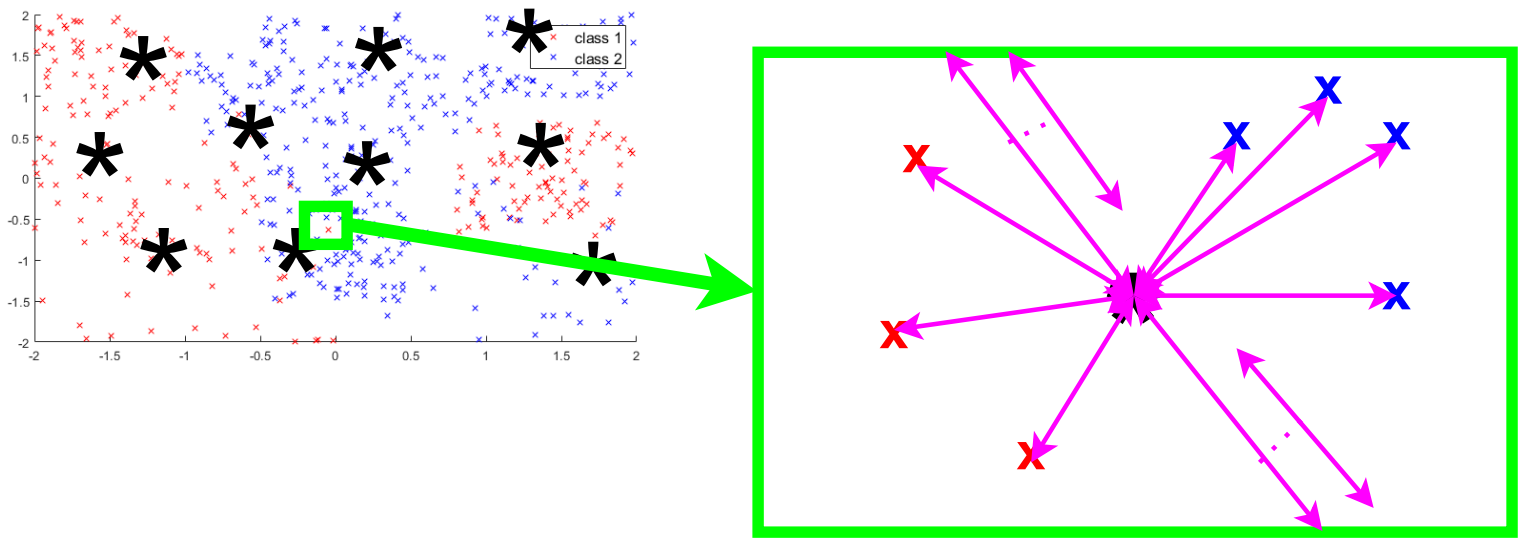
K-Means Clustering

Example: Assume $K = 10$ clusters.



K-Means Clustering

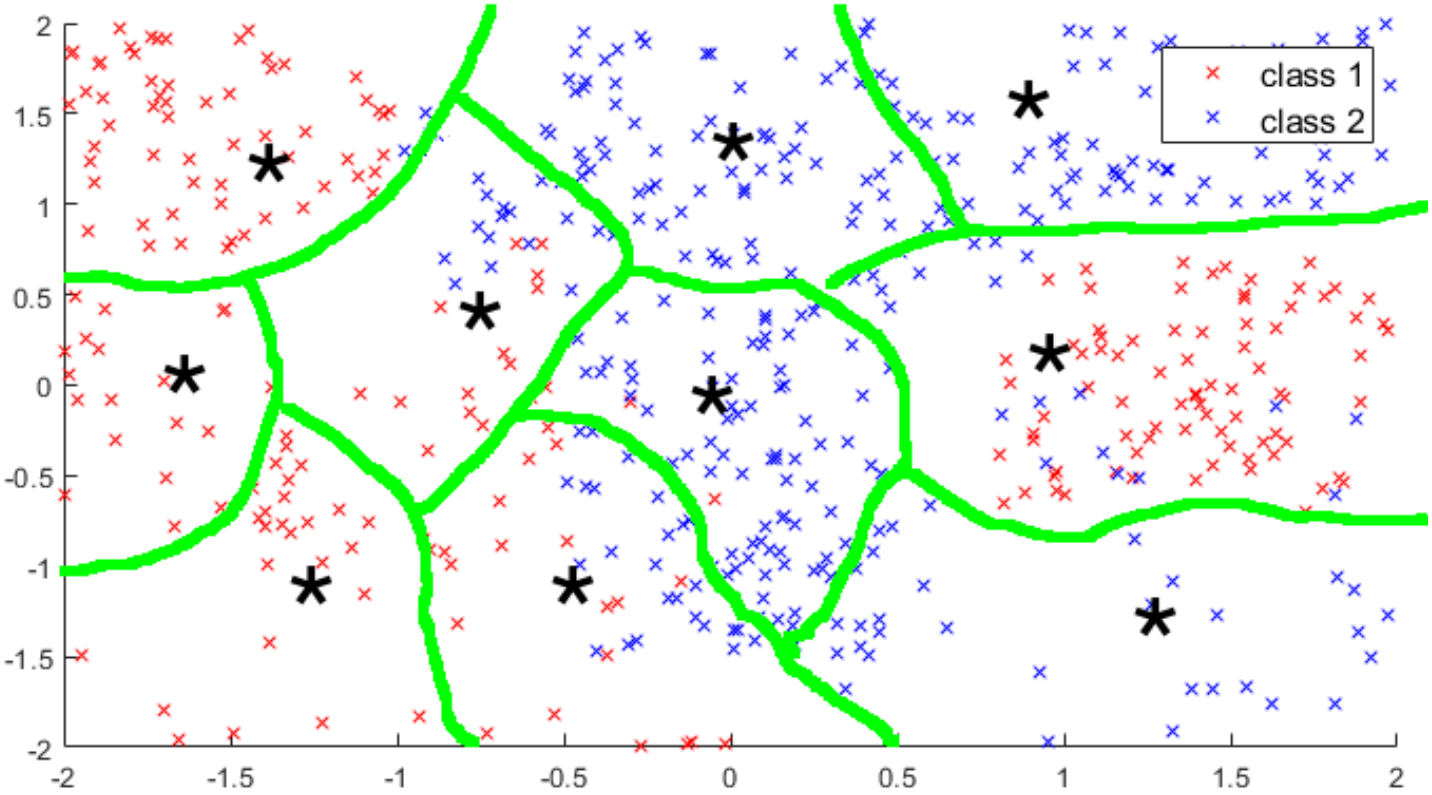
Example: Calculate all distances between each data sample and each cluster center.



The procedure is measuring of similarity between every pair of vectors x_p and μ_j, which is denoted by $d(x_i, \mu_j)$.

K-Means Clustering

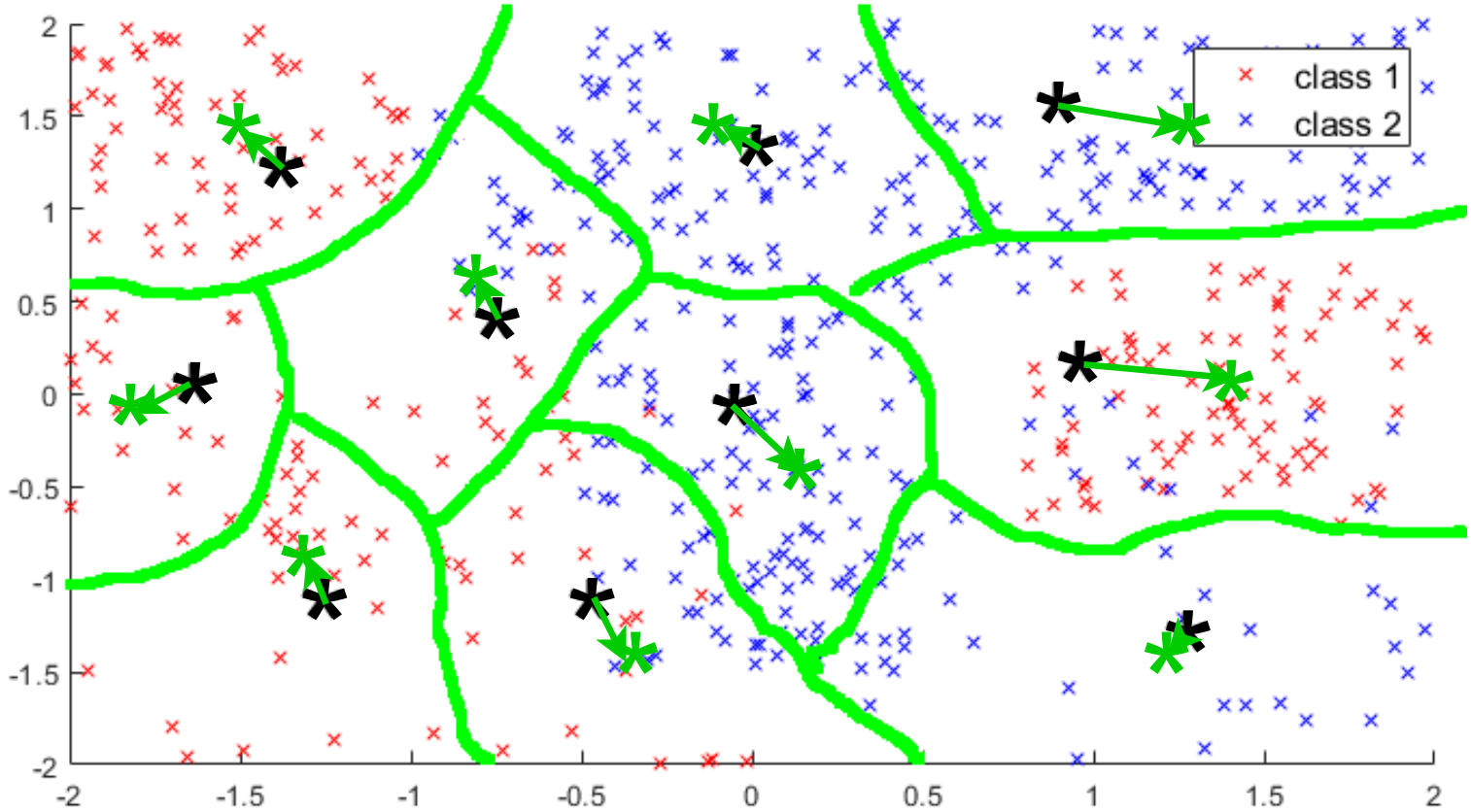
Example: Assign each data sample to the group that has the closest cluster center.



When the measure $d(x_i, \mu_j)$ is small enough (smaller than other centroids), the x_i is assigned to the j cluster.

K-Means Clustering

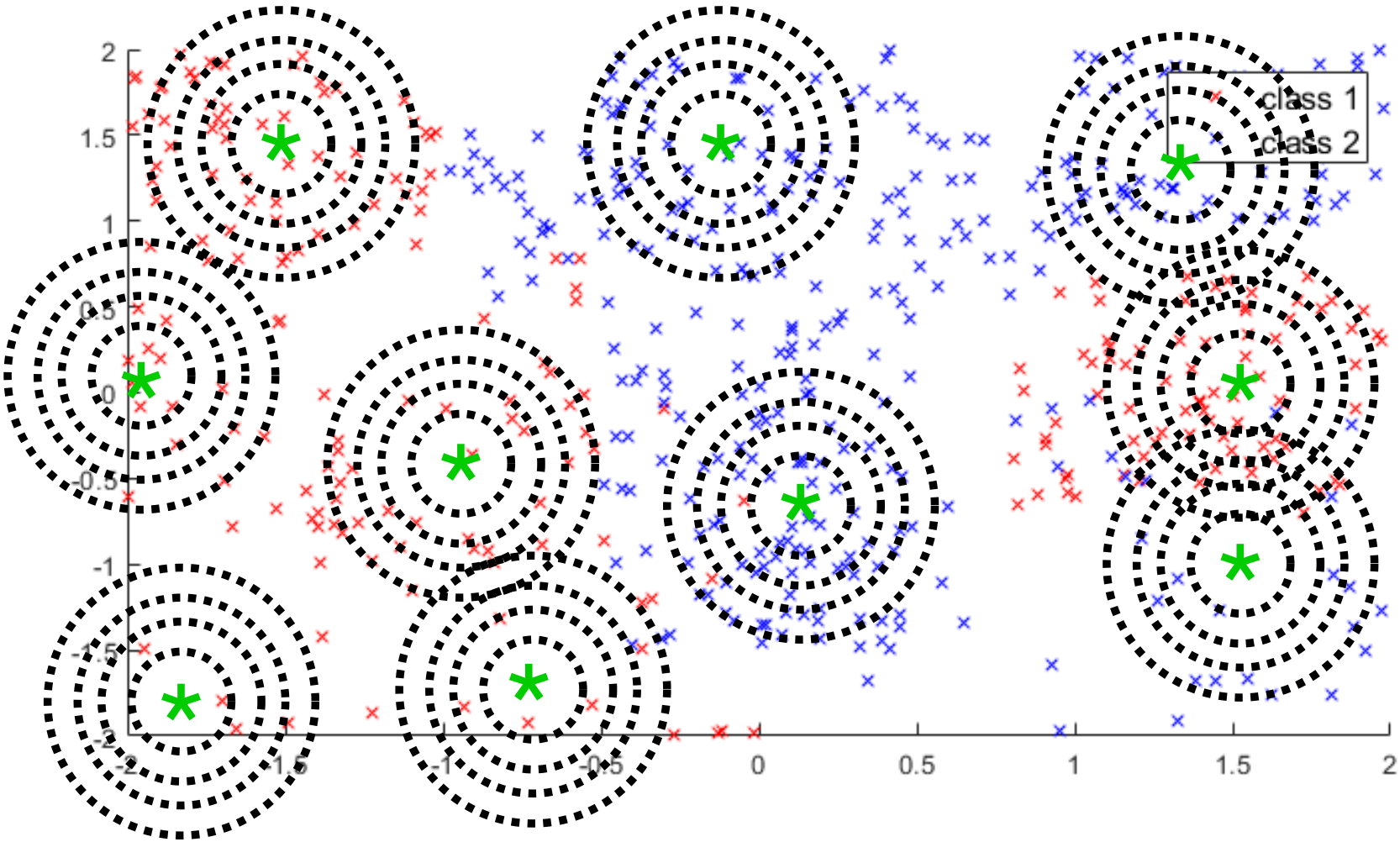
Example: New centroid (centres) coordinates.



The assigned samples will define a prototype of the new cluster.

K-Means Clustering

Example: What range of the data values should each cluster represent?.



Basis function parameter

Assign to all the Gaussian functions σ_j:

- Maximum distance d_{max} (maximum distance between centers)

$$\sigma_j = \frac{d_{max}}{\sqrt{2M}}$$

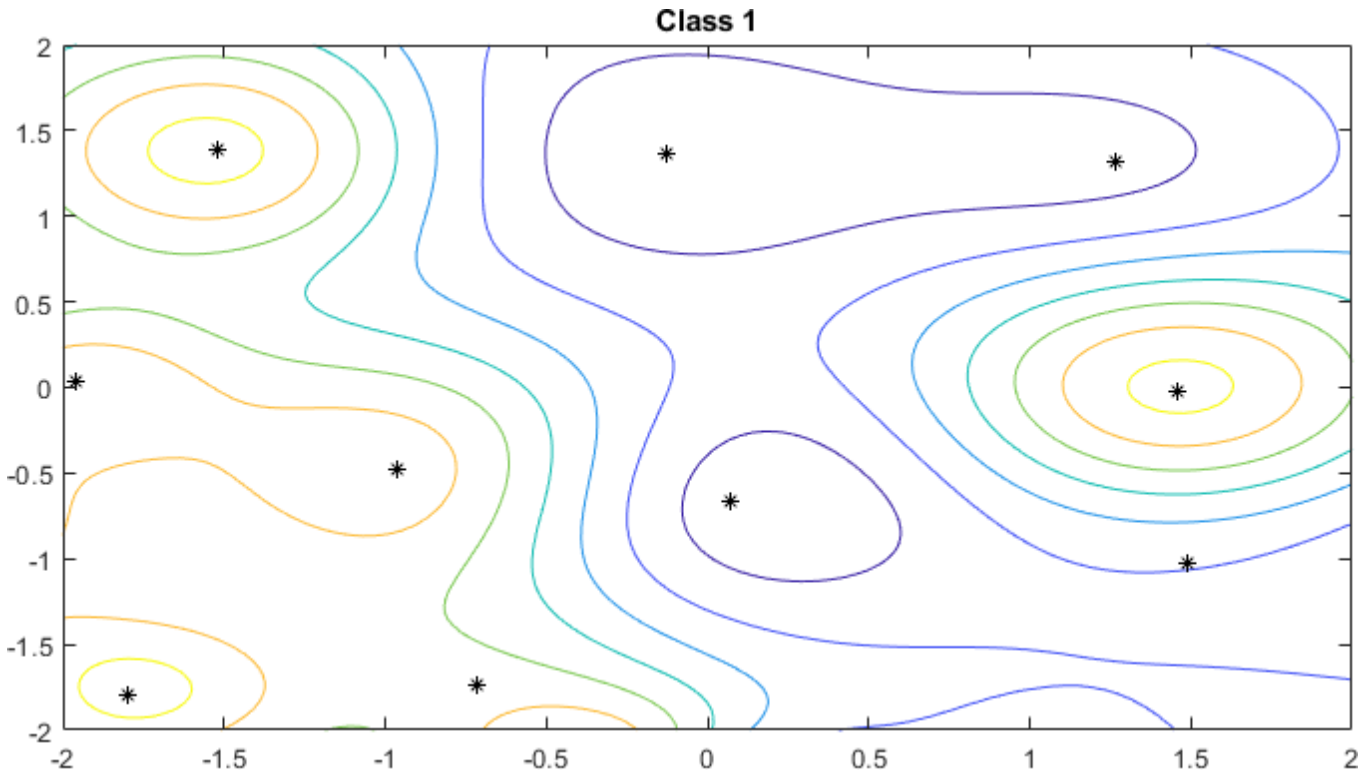
- Average distance $2d_{average}$ (average distance between centers)

$$\sigma_j = 2d_{average}$$

This approach gives reasonable results for large training sets, which ensure that the individual RBFs are neither too wide, nor too narrow.

Basis function parameter

The parameters of each hidden units have direct influence on classification.

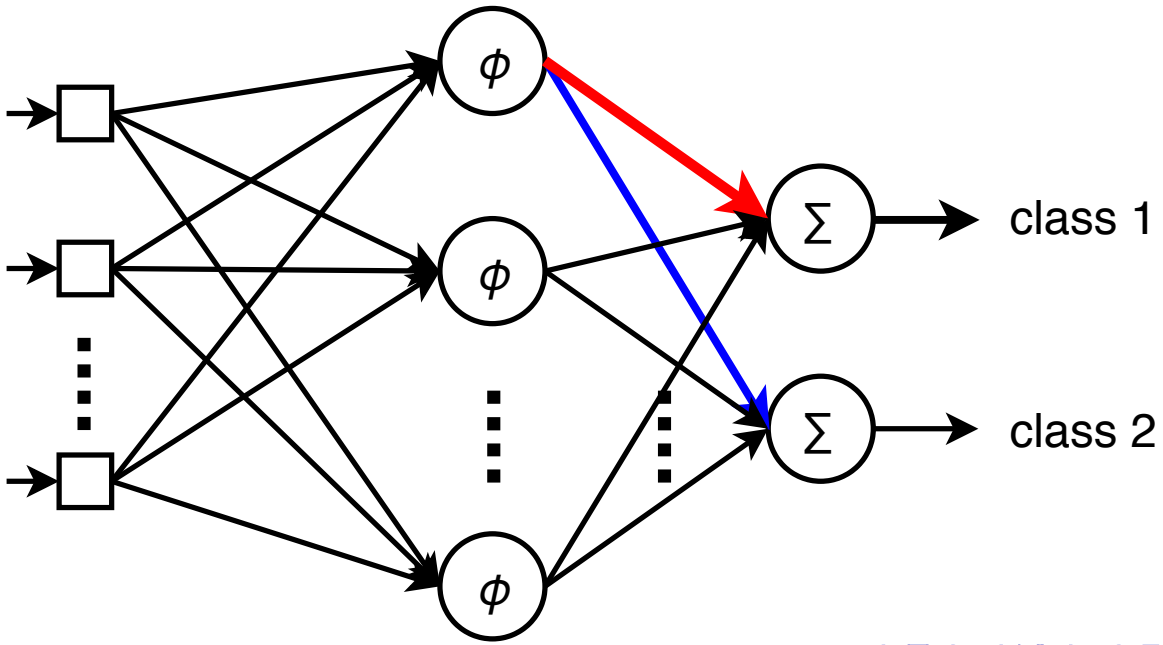


Question: Is k-means method enough for data classification? Have we missed something important?

Basis function parameter

Each RBF neuron (cluster in k-mean) have a different influence on each class. Example of influence of first RBF neuron:

- Red connection should have positive impact on class 1: $w > 0$,
- Blue connection should have negative impact on class 1: $w \leq 0$,



Computing the Output Weights

Given the hidden unit activations $\{\varphi_j(x, \mu_j, \sigma_j)\}$ are fixed while we determine the output weights $\{w_{j,k}\}$, we essentially only have to find the weights that optimize a single layer linear network with use of sum-squared output error:

$$E = \frac{1}{2} \sum_p \sum_k (t_{p,k} - y_k(x_p))^2$$

but here the outputs are a simple linear combination of the hidden unit activations, i.e.

$$y_k(x_p) = \sum_{j=0}^{M} w_{j,k} \varphi_j(x_p)$$

Computing the Output Weights

The back-propagation algorithm:

$$\frac{dE}{dw} = \frac{dE}{dy}\frac{dy}{dw}$$

At the minimum of E the gradients with respect to all the weights w_j will be zero:

$$\frac{\partial E}{\partial w_{j,k}} = \sum_p \left((t_{p,k} - \sum_{j=0}^{M} w_{j,k}\varphi_j(x_p))\varphi_j(x_p) \right)$$

The linear equations like this can be also easily solved analytically.

Computing the Output Weights

The weights are most conveniently written in matrix form by defining matrices with components $W = \{w_{j,k}\}$, $\Phi = \{\varphi_j(x_p)\}$, and $T = \{t_p\}$. This gives

$$\Phi^T(\Phi W^T - T) = 0$$

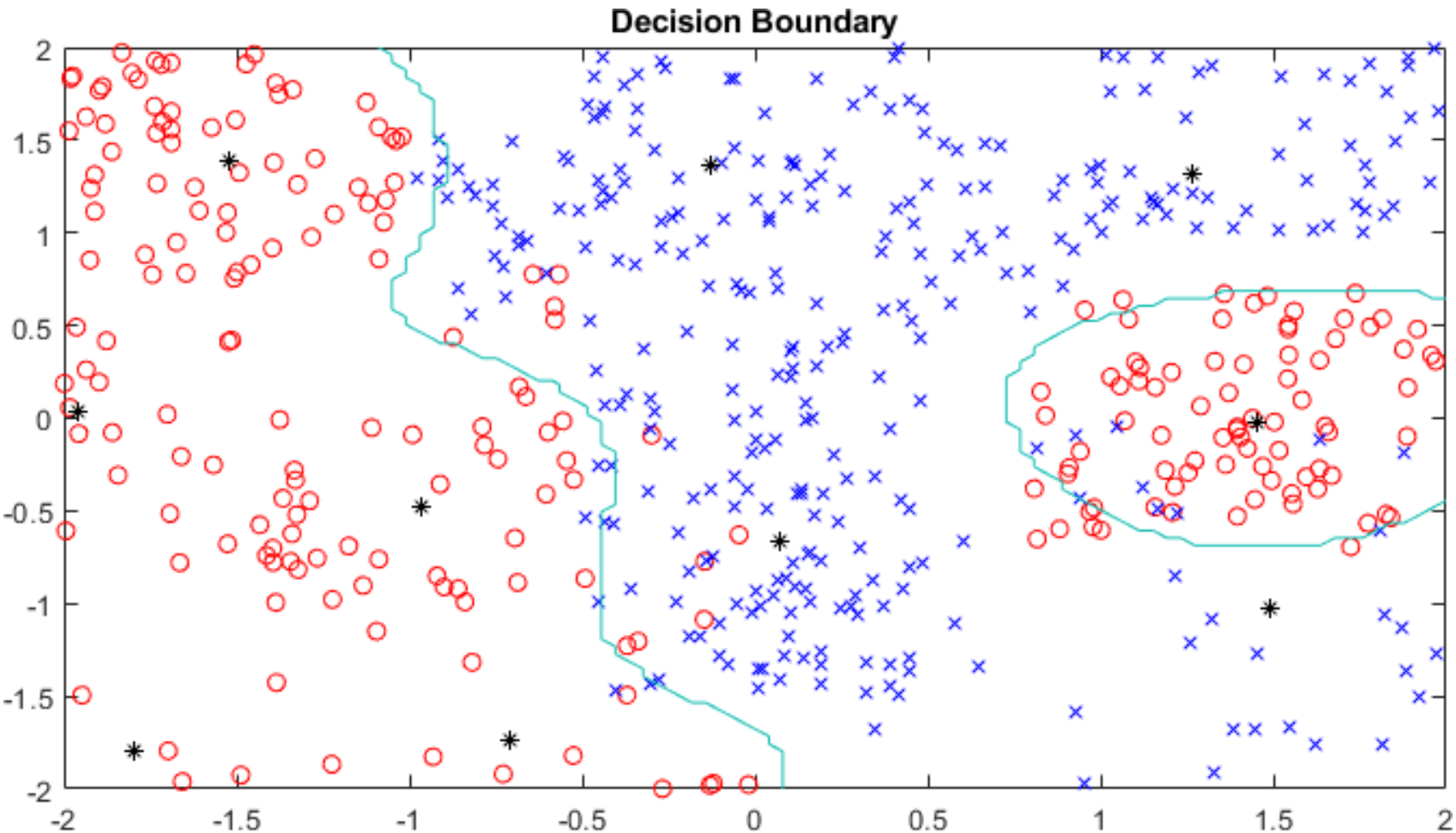
and the formal solution for the weights is

$$W^T = (\Phi^T\Phi)^{-1}\Phi^T T$$

We see that the network weights can be computed by fast linear matrix inversion techniques. In practice we tend to use singular value decomposition (SVD) to avoid possible ill-conditioning of Φ, i.e. $\Phi^T\Phi$ being singular or near singular.

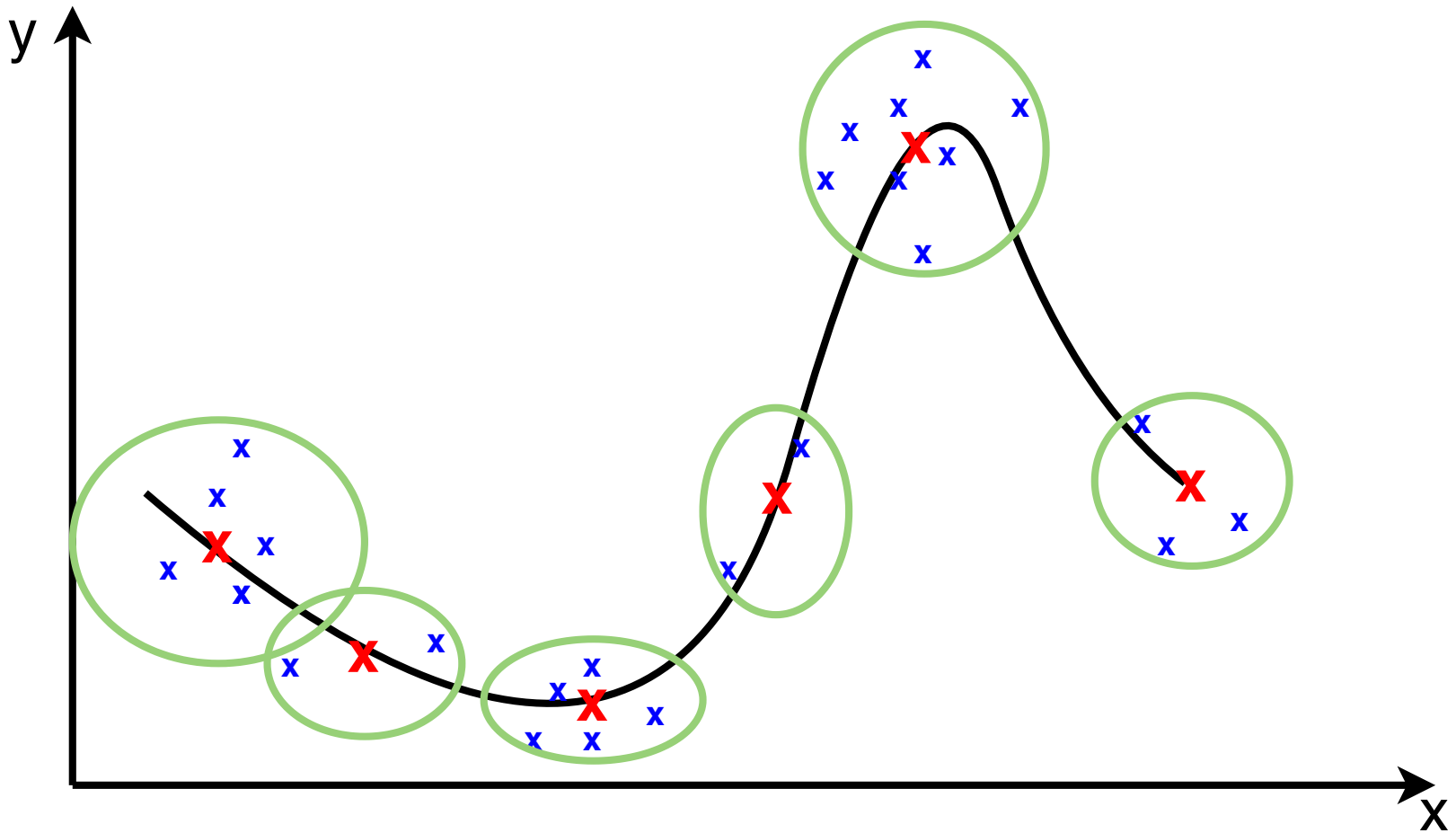
Classification

The Result of classification is hyperspace - decision boundary.



Approximation

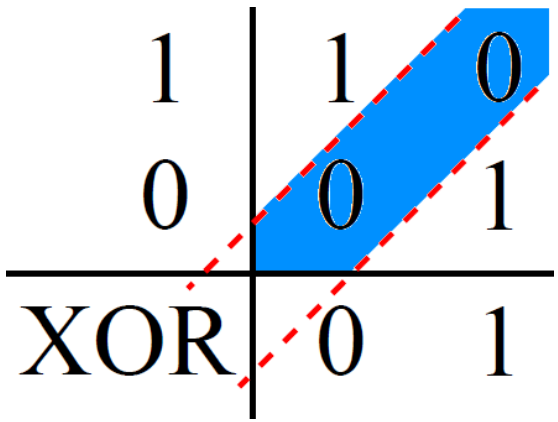
The approximation problems can use the same algorithm ($M < N$).



The XOR Problem

Revision of non-linearly separable XOR problem

p	x_1	x_2	t
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0



To deal with this problem using Perceptrons we needed to either change the activation function, or introduce a non-linear hidden layer to MLP network.

The XOR Problem

To perform the XOR classification in an RBF network, we start by deciding how many basis functions we need. Given there are four training patterns and binary values. We then need to assume the basis function centers for two basis functions $M = 2$:

$$\varphi_1(x) = \exp\left(-||x - \mu_1||^2\right) \quad \mu_1 = [0, 0]^T$$
$$\varphi_2(x) = \exp\left(-||x - \mu_2||^2\right) \quad \mu_2 = [1, 1]^T$$

Because: $\sigma_j = \frac{d_{max}}{\sqrt{2M}} = \frac{\sqrt{2}}{2}$ and $2\sigma_j^2 = 2 * \frac{2}{4} = 1$. The input patterns are mapped onto the (φ_1, φ_2) plane. This will transform the problem into a linearly separable form.

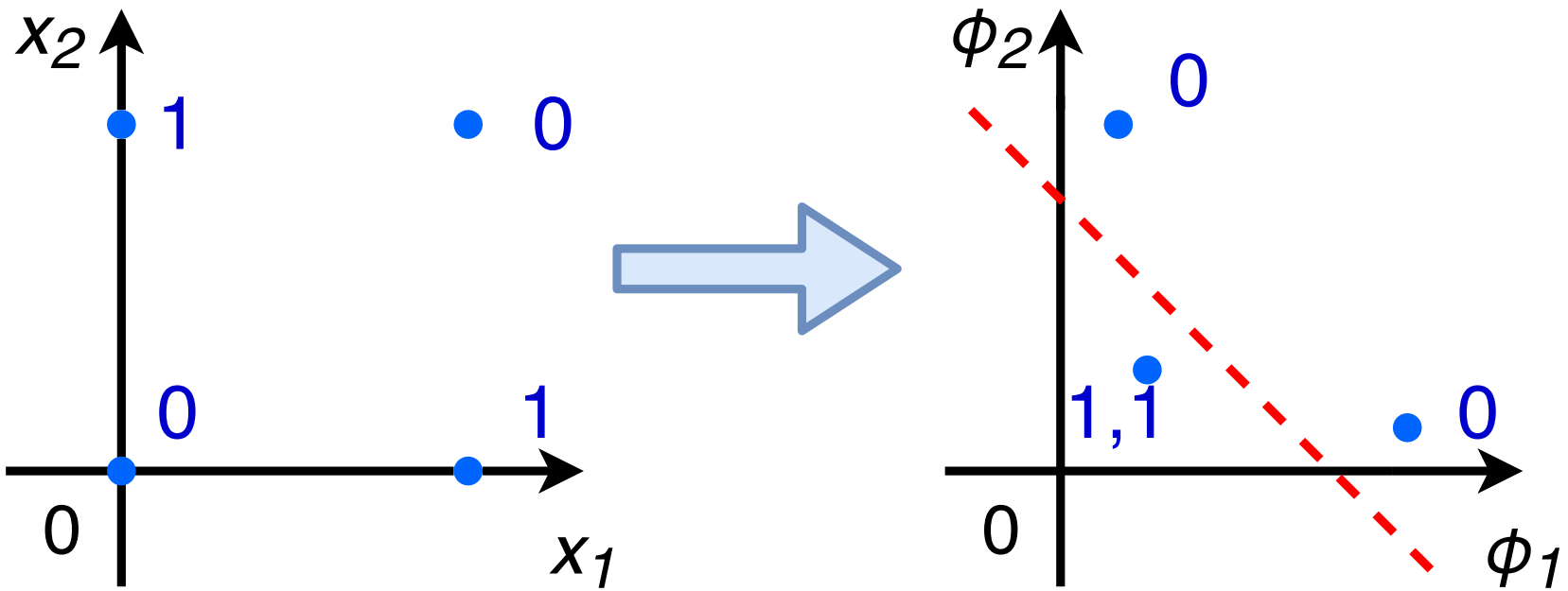
The XOR Problem

The input patterns have been transformed:

p	x_1	x_2	t	φ_1	φ_2
1	0	0	0	1.0000	0.1353
2	0	1	1	0.3678	0.3678
3	1	0	1	0.3678	0.3678
4	1	1	0	0.1353	1.0000

The input patterns (0, 1) and (1, 0) are linearly separable from the remaining input patterns (1, 1) and (0, 0). Thereafter, the XOR problem may be readily solved by using the functions $\varphi_1(x)$ and $\varphi_2(x)$ as the inputs to a linear classifier such as the perceptron.

The XOR Problem



The XOR Problem

The separation line can be calculated for one output $y(x)$, with one weight w_j to each hidden unit j, and one bias $-\theta$ with use of simple equations:

$$y(x) = w_1\varphi_1(x) + w_2\varphi_2(x) - \theta$$

if we want the outputs $y(x_p)$ to be equal to the targets t^p:

$$\begin{aligned} 1.0000w_1 + 0.1353w_2 - \theta &= 0 \\ 0.3678w_1 + 0.3678w_2 - \theta &= 1 \\ 0.3678w_1 + 0.3678w_2 - \theta &= 1 \\ 0.1353w_1 + 1.0000w_2 - \theta &= 0 \end{aligned}$$

Three are different variables can be solve:

$$w_1 = w_2 = -2.5018 \quad , \quad \theta = -2.8404$$

Characteristic of RBF Networks

Supervised RBF Network Training

The supervised RBF Network Training use gradient descent method on a sum squared output error function in the same as in previous lectures. The error function would be:

$$E = \frac{1}{2}\sum_p \sum_k (y_k(x_p) - t_{p,k})^2 = \frac{1}{2}\sum_p \sum_k (w_{j,k}\varphi_j(x_p, \mu_j, \sigma_j) - t_{p,k}^2)$$

and we would iteratively update the weights and basis function parameters using

$$\Delta w_{j,k} = -\eta_w \frac{\partial E}{\partial w_{j,k}} \quad \Delta \mu_j = -\eta_\mu \frac{\partial E}{\partial \mu_j} \quad \Delta \sigma_j = -\eta_\sigma \frac{\partial E}{\partial \sigma_j}$$

We have all the problems of choosing the learning rates η, avoiding local minimum and so on, that we had for training MLPs by gradient descent.

Regularization of RBF Networks

Regularization add an extra term to the error measure which penalizes mappings which are not smooth. If we have network outputs $y_k(x^p)$ and sum squared error measure, we can introduce some appropriate differential operator P:

$$E = \frac{1}{2}\sum_p \sum_k (y_k(x_p) - t_{p,k})^2 + \lambda \sum_k \int |Py_k(x)|^2 dx$$

where λ is the regularization parameter which determines the relative importance of smoothness compared with error.

Regularization of RBF Network

There are many possible forms for P, but the general idea is that mapping functions $y_k(x)$ which have large curvature should have large values of $|Py_k(x)|^2$ and hence contribute a large penalty in the total error function:

$$\lambda \sum_k \int |Py_k(x)|^2 dx = \lambda \sum_k \left(\sum_p \sum_i \frac{1}{2} \frac{\partial^2 y_k(x^p)}{\partial x_i^2} \right)$$

certainly penalizes large output curvature, and minimizing the error function E now leads to linear equations for the output weights.

Regularization of RBF Network

The weight can be found by solving:

$$MW^T - \Phi^T T = 0$$
$$M = \Phi^T \Phi$$

In this we have defined the same matrices with components $W_{k,j} = w_{k,j}$, $\Phi_{p,j} = \varphi_j(x_p)$, and $T_{p,k} = t_{p,k}$ as before, and also the regularized version of $\Phi^T \Phi$:

$$M = \Phi^T \Phi + \lambda \sum_i \left(\frac{\partial^2 \Phi^T}{\partial x_i^2} \right)^2$$

Clearly for $\lambda = 0$ this reduces to the un-regularized results.

RBF vs MLP

Similarities between RBF and MLP:

- both universal approximators,
- both are used in similar application areas.

Difference between RBF and MLP:

- An RBF network has a single hidden layer, whereas MLPs can have any number of hidden layers.
- In RBF networks the hidden nodes (basis functions) operate very differently, and have a very different purpose, to the output nodes, whereas MLPs in different layers share a common neuronal model.

RBF vs MLP

Difference between RBF and MLP:

- In RBF networks, the argument of each hidden unit activation function is the distance between the input and the "weights" (RBF centres), whereas in MLPs it is the inner product of the input and the weights.
- MLPs are usually trained with a single global supervised algorithm, whereas RBF networks are usually trained one layer at a time with the first layer unsupervised.
- In RBF networks, the argument of each hidden unit activation function is the distance between the input and the weights (RBF centres), whereas in MLPs it is the inner product of the input and the weights.

Questions