Institute for Advanced Studies in Basic Sciences – Zanjan
Department of Computer Science and Information Technology
By: Mahdi Vasighi

Artificial Neural Networks

Part 10

Radial Basis Function Networks

Introduction

Cover's Theorem

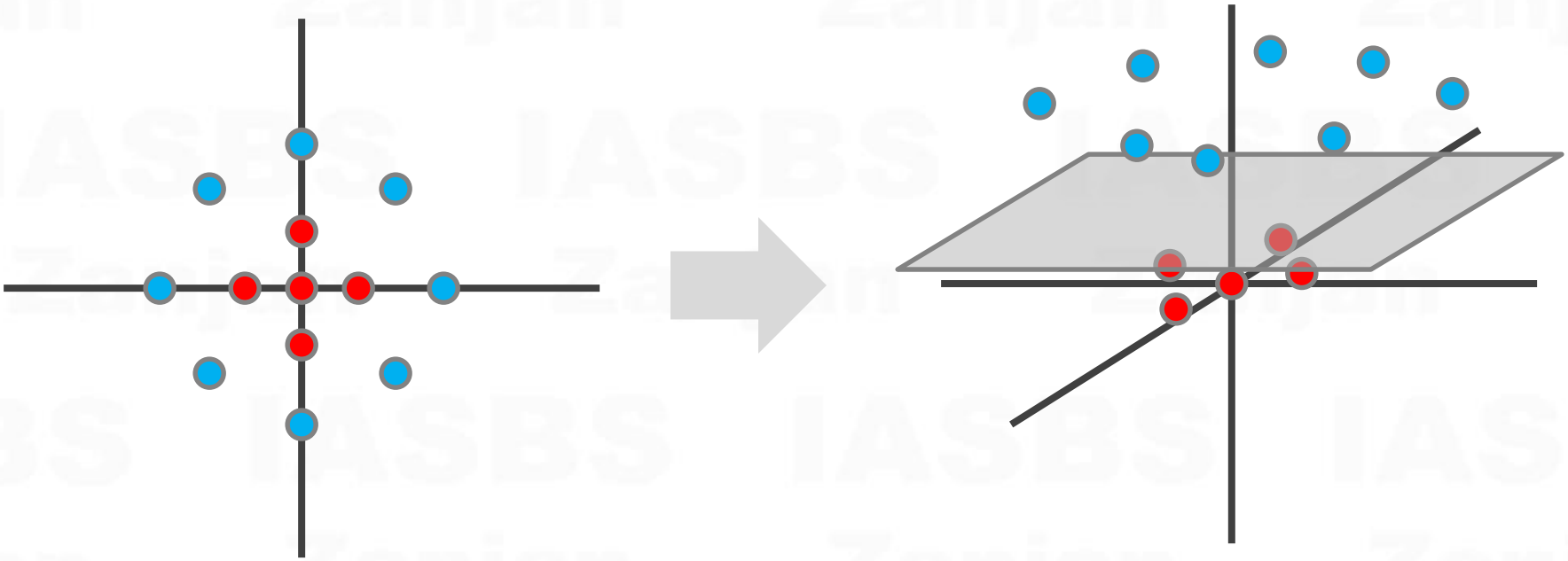
A complex pattern classification problem cast in a high-dimensional space nonlinearly, is more likely to be linearly separable than in a low-dimensional space.

Radial Basis Function Networks

Introduction

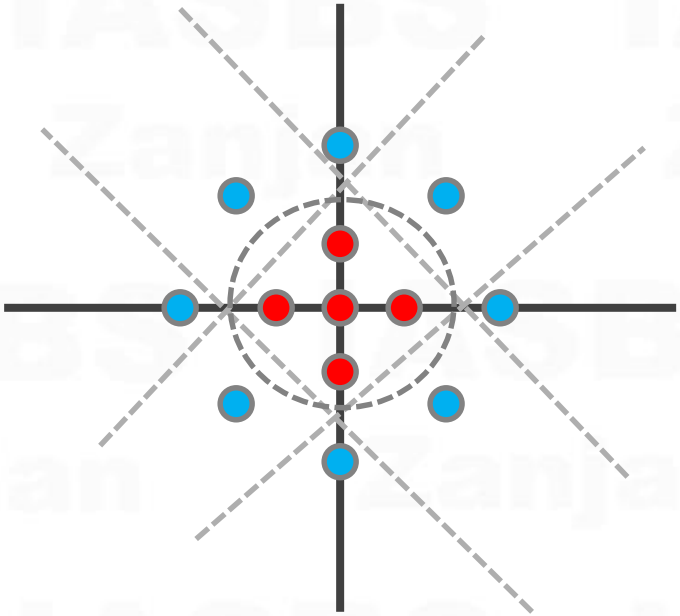
Cover's Theorem

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.

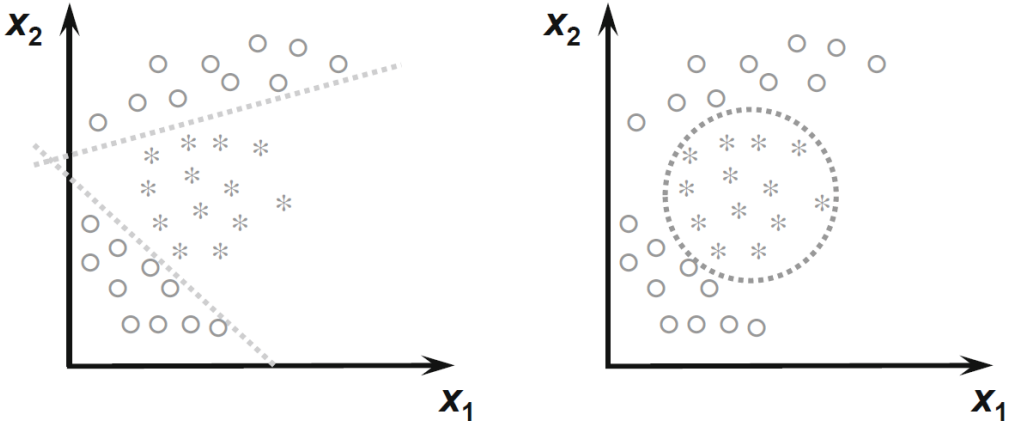


Radial Basis Function Networks

Introduction



With a traditional feed-forward network using sigmoid functions, perhaps four hidden nodes may be required for this problem.



- A function is **radially symmetric** (or is an RBF) if its output depends on the distance of the input sample (vector) from another stored vector.
- Neural networks whose node functions are radially symmetric functions are referred to as **RBF-nets**.

Radial Basis Function Networks

Introduction

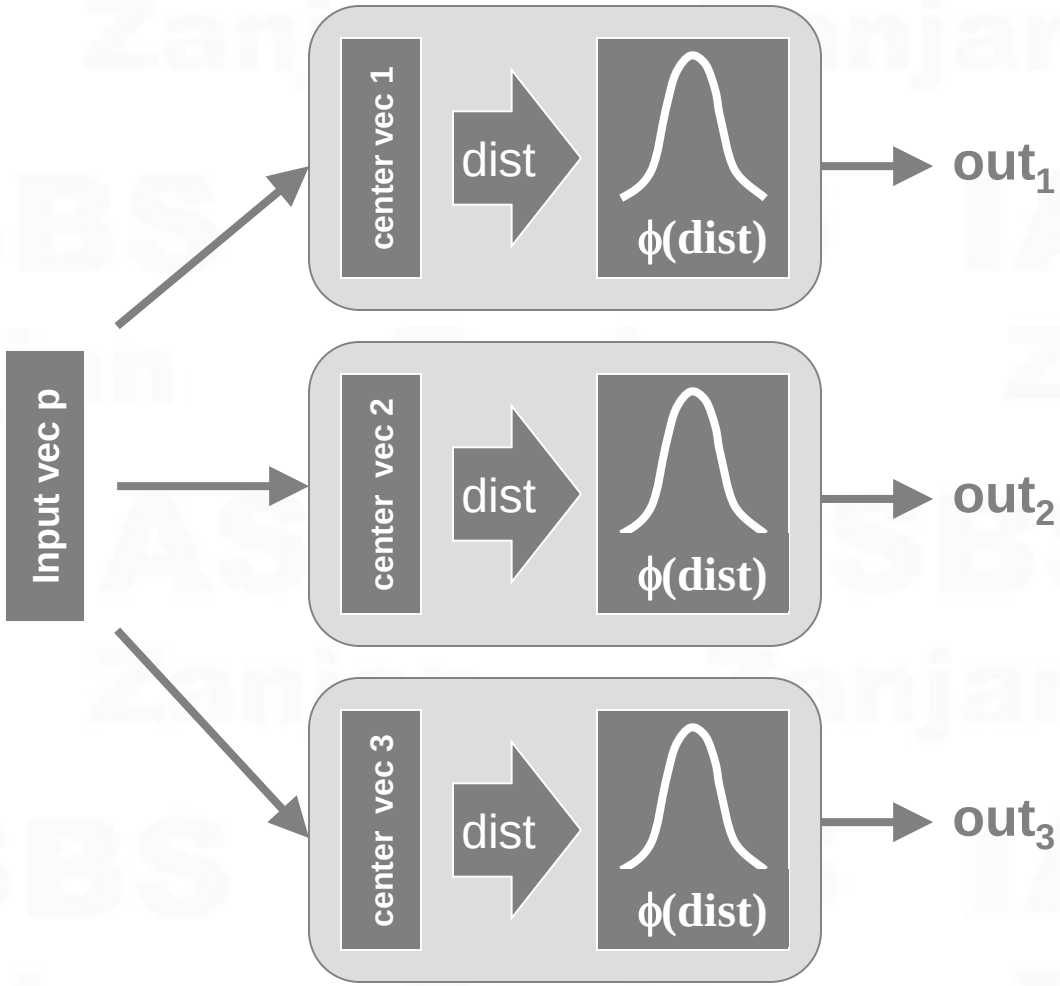
We have seen how MLP networks with a hidden layer of sigmoidal units can learn to approximate functions. Learning in RBF Networks is supervised but in a slightly different approach.

RBF networks have some features:

- Two-layer feed-forward networks.
- The hidden nodes implement a set of radial basis functions (e.g. Gaussian functions).
- The output nodes implement linear summation functions as in an MLP.
- The training is fast.
- The networks are good for interpolation

Radial Basis Function Networks

Structure



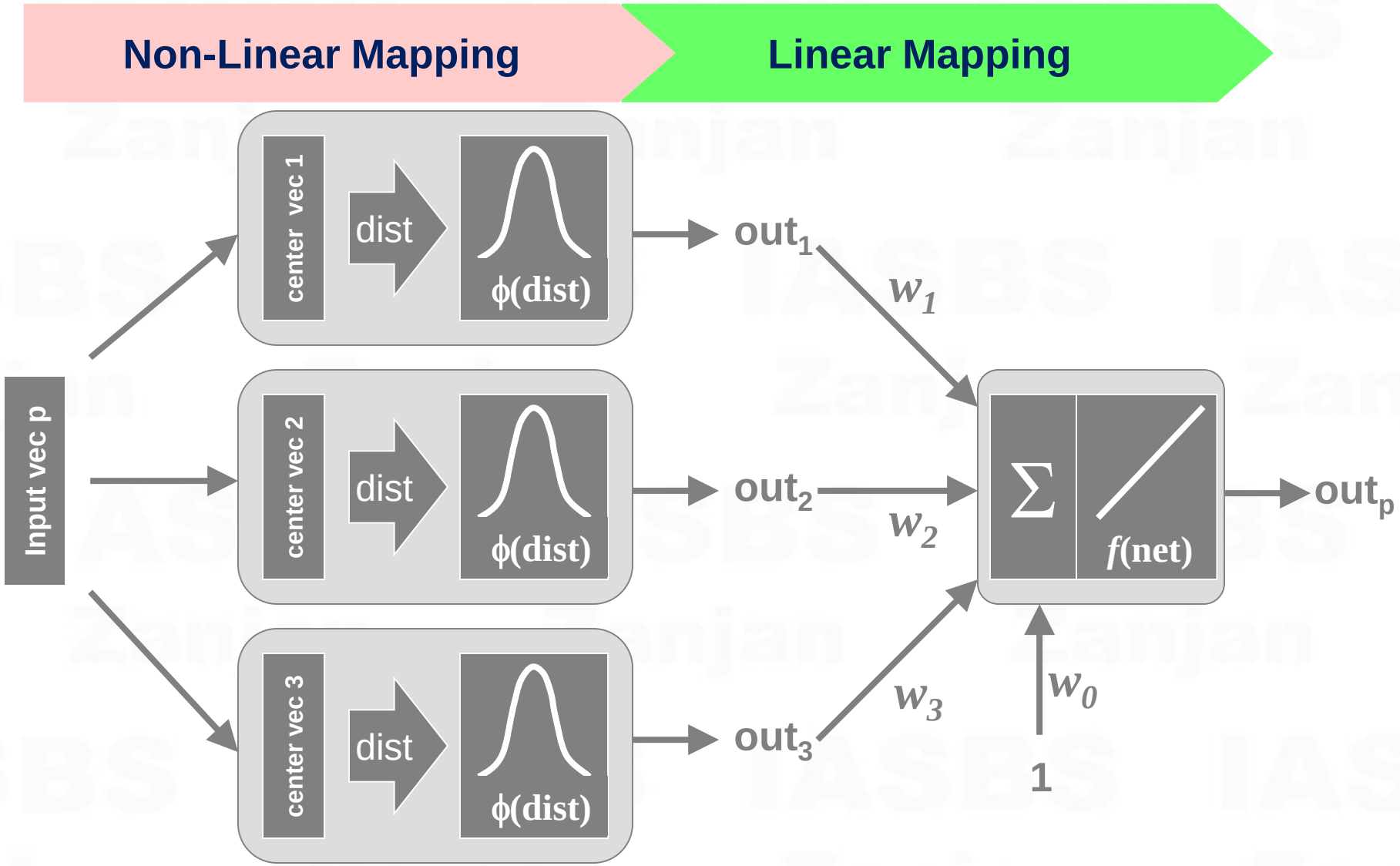
$$\phi(dist) = \phi(\|\vec{x} - \vec{c}_j\|)$$

$$\phi(dist) = \exp\left(\frac{-\|\vec{x} - \vec{c}_j\|^2}{2\sigma^2}\right)$$

Gaussian basis functions can be used as a radial basis function

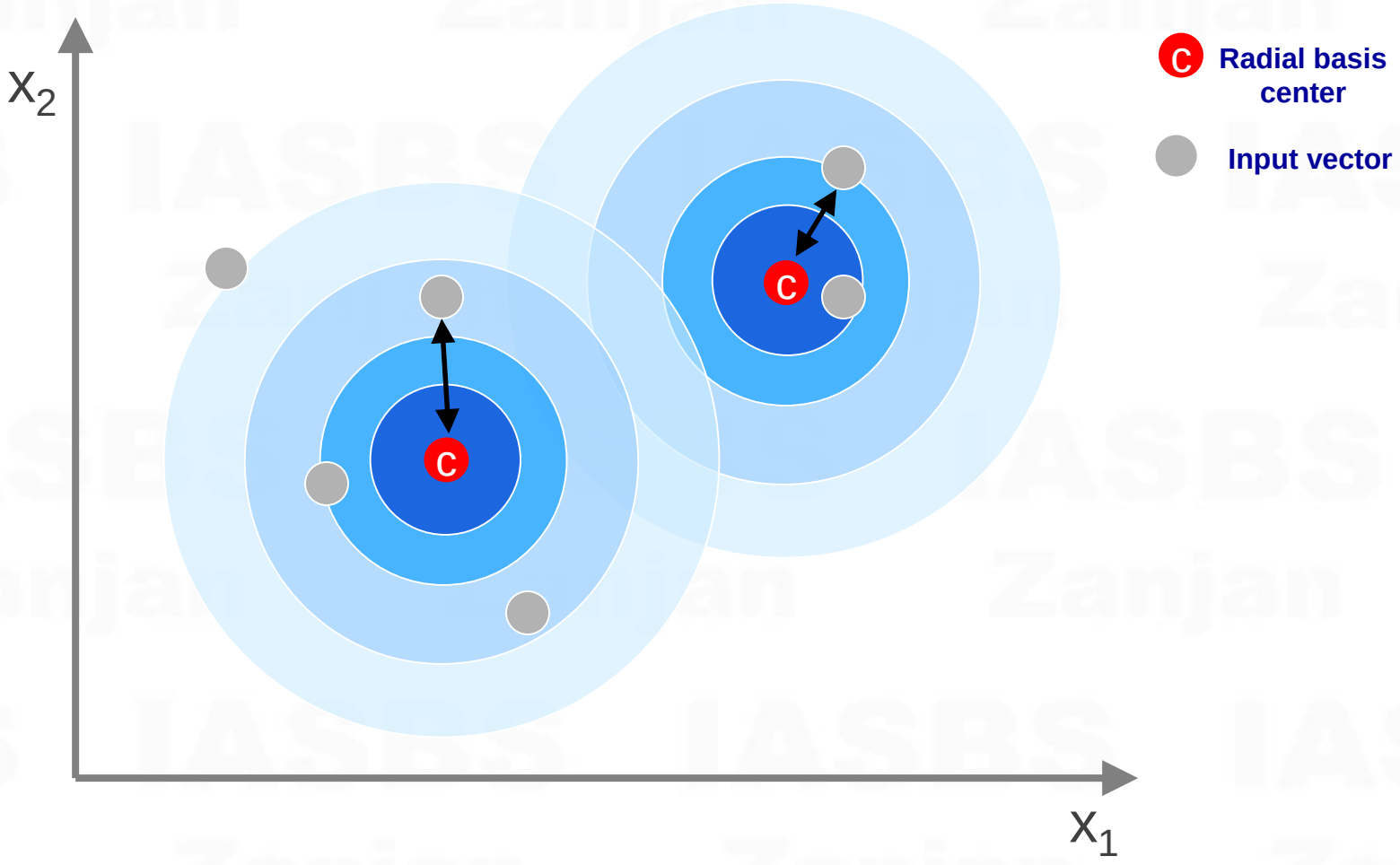
Radial Basis Function Networks

Structure



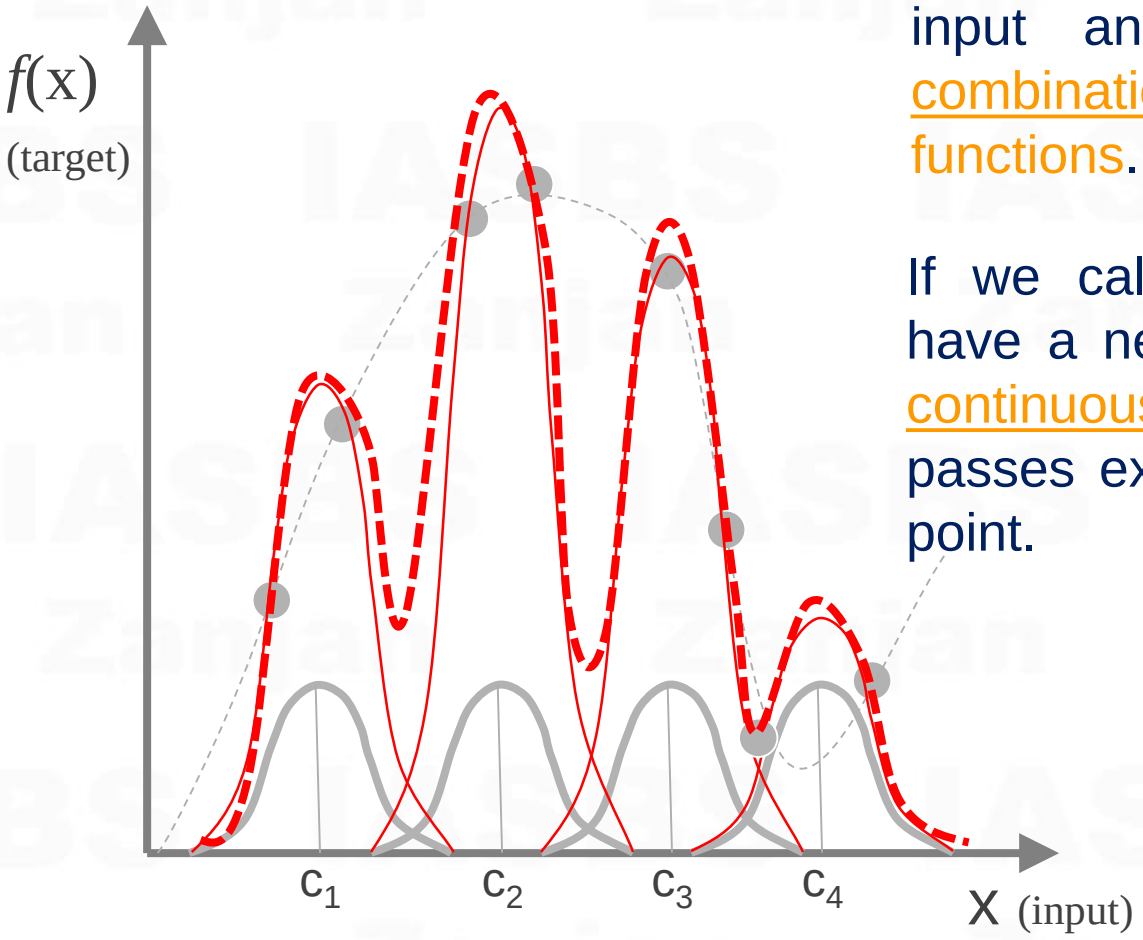
Radial Basis Function Networks

Structure



Radial Basis Function Networks

Structure



The simplified story is that RBF tries to learn relationship between input and target using linear combination of radial basis functions.

If we calculate the weights, we have a network that represents a continuous differentiable line that passes exactly through each data point.

Radial Basis Function Networks

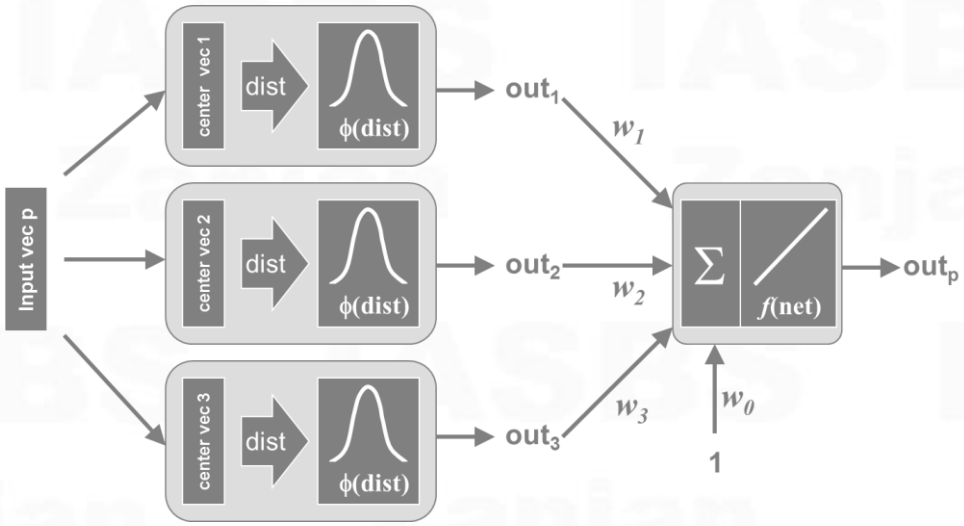
Structure

We can take the basic structure of the RBF networks that perform exact interpolation and improve upon them in a number of ways:

- The number basis functions (M hidden units) need not equal the number of training data points (N). 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 need not all have the same width parameter σ. These can also be determined by a training algorithm.

Radial Basis Function Networks

Learning



$$\text{out}_j = \Phi_j\left(\left\|\vec{x}_p - \vec{c}_j^{(1)}\right\|\right)$$

The output of the network is then taken to be a linear combination of the basis functions

$$\text{out}_k = w_{k0}^{(2)} + \sum_j w_{jk}^{(2)}\Phi_j\left(\left\|\vec{x}_i - \vec{c}_j^{(1)}\right\|\right) \qquad \text{out}_k = \sum_{j=0}^{M} w_{jk}^{(2)}\Phi_j(x)$$

For one sample with *j* radial basis neuron we can write:

$$\begin{bmatrix}\varphi_{11} & \varphi_{12} & \cdots & \varphi_{1j}\end{bmatrix}\begin{bmatrix}w_{11}\\ w_{21}\\ \vdots\\ w_{jk}\end{bmatrix} = \begin{bmatrix}t_1\end{bmatrix}$$

Radial Basis Function Networks

Learning

For M sample with *j* radial basis neuron we can write:

$$\begin{bmatrix} \varphi_{11} & \varphi_{12} & \cdots & \varphi_{1j} \\ \varphi_{21} & \varphi_{22} & & \varphi_{2j} \\ \vdots & & \ddots & \vdots \\ \varphi_{M1} & \varphi_{M2} & \cdots & \varphi_{Mj} \end{bmatrix} \begin{bmatrix} w_{11} \\ w_{21} \\ \vdots \\ w_{j1} \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_M \end{bmatrix}$$

This simplifies the equation to

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

Weights can be calculated as follow:

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

Radial Basis Function Networks

Learning

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

$$\mathbf{\Phi}^T\mathbf{\Phi}\ \mathbf{w} = \mathbf{\Phi}^T\mathbf{t}$$

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

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

pseudo inverse

$$\mathbf{w} = \mathbf{\Phi}^{+}\mathbf{t}$$

The weights can be computed by fast linear matrix inversion techniques!

Radial Basis Function Networks

Learning

There are several ways of choosing proper values for parameters of hidden nodes (weight vectors & spread):

1. Random selection of fixed centers

The simplest and quickest approach to setting the RBF parameters is to have their centers fixed at M points selected at random from the N data points, and to set all their widths (spread) to be equal and fixed at an appropriate size for the distribution of data points.

The widths (spread) of radial basis function can be related to the maximum or average distance between the chosen centers.

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

Radial Basis Function Networks

Learning

There are several ways of choosing proper values for parameters of hidden nodes (weight vectors & spread):

2. subset selection

- **Forward selection**
  - Start with an empty subset of centers
  - Added one basis at a time (the one that most reduces the error)
  - Until some chosen criteria
- **Backward elimination**
  - Start with a full subset of centers
  - Remove one basis function at a time (the one that least increases the error)
  - Until the chosen criterion stops decreasing

Radial Basis Function Networks

Learning

There are several ways of choosing proper values for parameters of hidden nodes (weight vectors & spread):

3. K-means clustering

The K-Means Clustering Algorithm picks the number K of centers in advance, and then follows a simple re-estimation procedure to partition the data points into K disjoint sub-sets.

Cluster centroids can be selected as radial basis centers and the widths (spread) can then be set according to the variances of the points in the corresponding cluster.

With these approaches, determining a good value for M remains a problem. It will generally be appropriate to follow the same kind of validation/cross-validation methodology used for optimizing MLPs.