Artificial Neural Networks

Topic-07: Radial Basis Function Neural Networks

Radial Basis Function Neural Networks

- The Radial-Basis function was first introduced in the solution of the real multivariate interpolation problems
- The RBFNN first performs non-liner transformation from given input space into higher dimension hidden space followed by linear transformation from hidden space to output space.
- *A pattern classification problem cast in a higher dimensional space is more likely to be linearly separable than in a lower dimensional space*- this is the reason for frequently making the dimension of the hidden space of RBF network high.
- Another important point is that *the dimension of the hidden space is directly related to the capacity of the network to approximate a smooth input-output mapping*. So the higher the dimension of the hidden space, the more accurate the approximation will be.

Basic Architecture of RBFNN

The construction of a RBFNN in its most basic form involves three layers with entirely different roles

- **Input Layer**: It contains *n* input (sensory/source) neurons that connects to the network to its external environment
- **Hidden Layer**: This is **only one** hidden layer. Hidden units provide a set of radial-basis function performs a non-linear transformation from the input space to the hidden space. In most applications the hidden space is of high dimensionality than input space.
- **Output Layer**: It contains *m* output neurons and each of which combines in a linear way the activations of the hidden layer. Supplies the response of the network for the activation pattern applied to the input layer

- <u>The connection between input layer and hidden layer have no associated weights.</u>
- The selection of an appropriate RBF depends on the type of problem to be solved by RBFNN.

$$RBF \quad \varphi_k = \varphi\left(\left\| X - c_k \right\|\right) = \varphi_k(r)$$

$$\varphi(r) = r \quad a \quad linear \quad radial \quad function$$

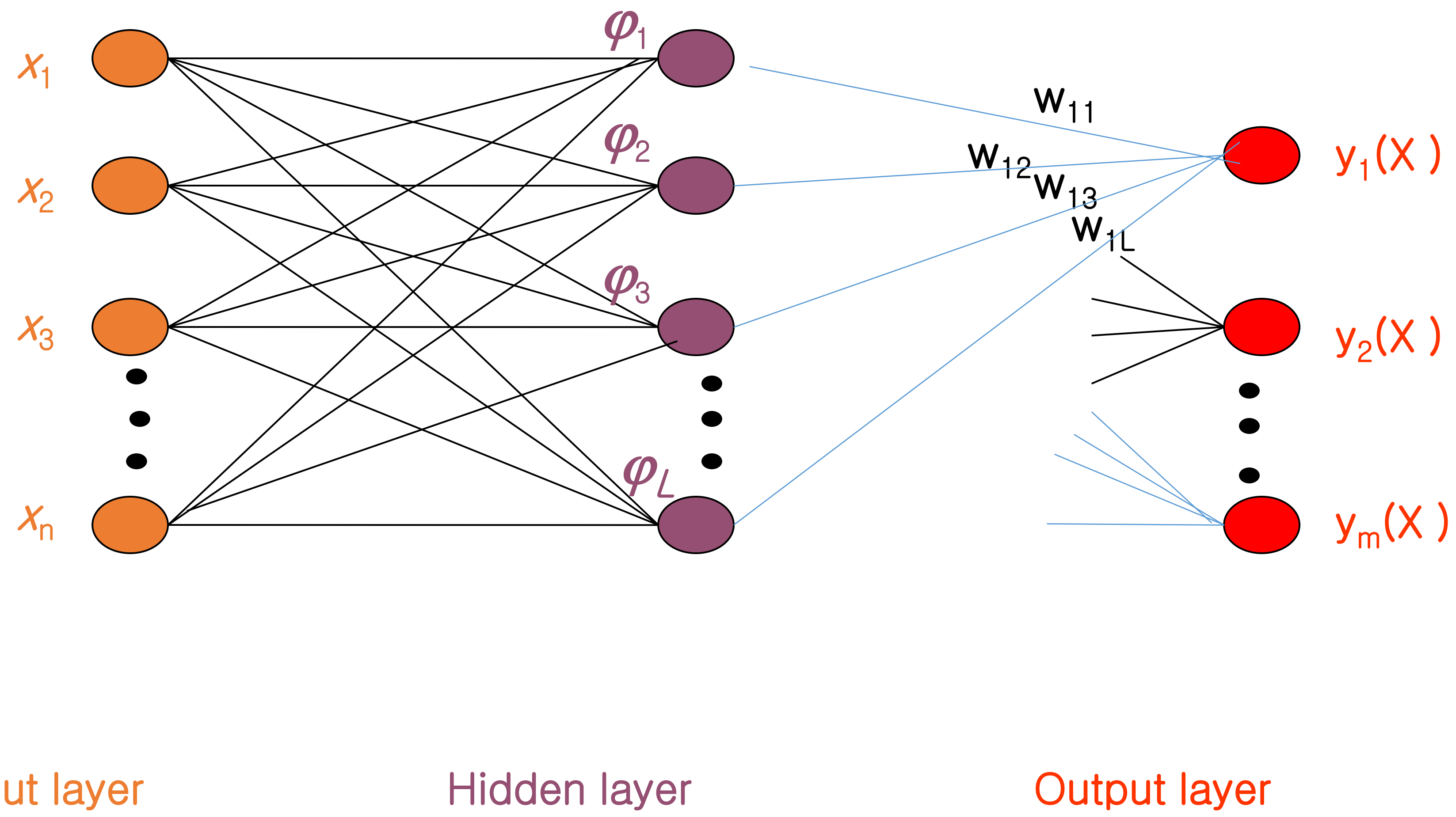
$$\varphi(r) = r^2 \, a \quad quadratic \quad function$$

$$\varphi(r) = \exp\left(-r^2 / b^2\right) \quad a \quad gaussian \quad function$$

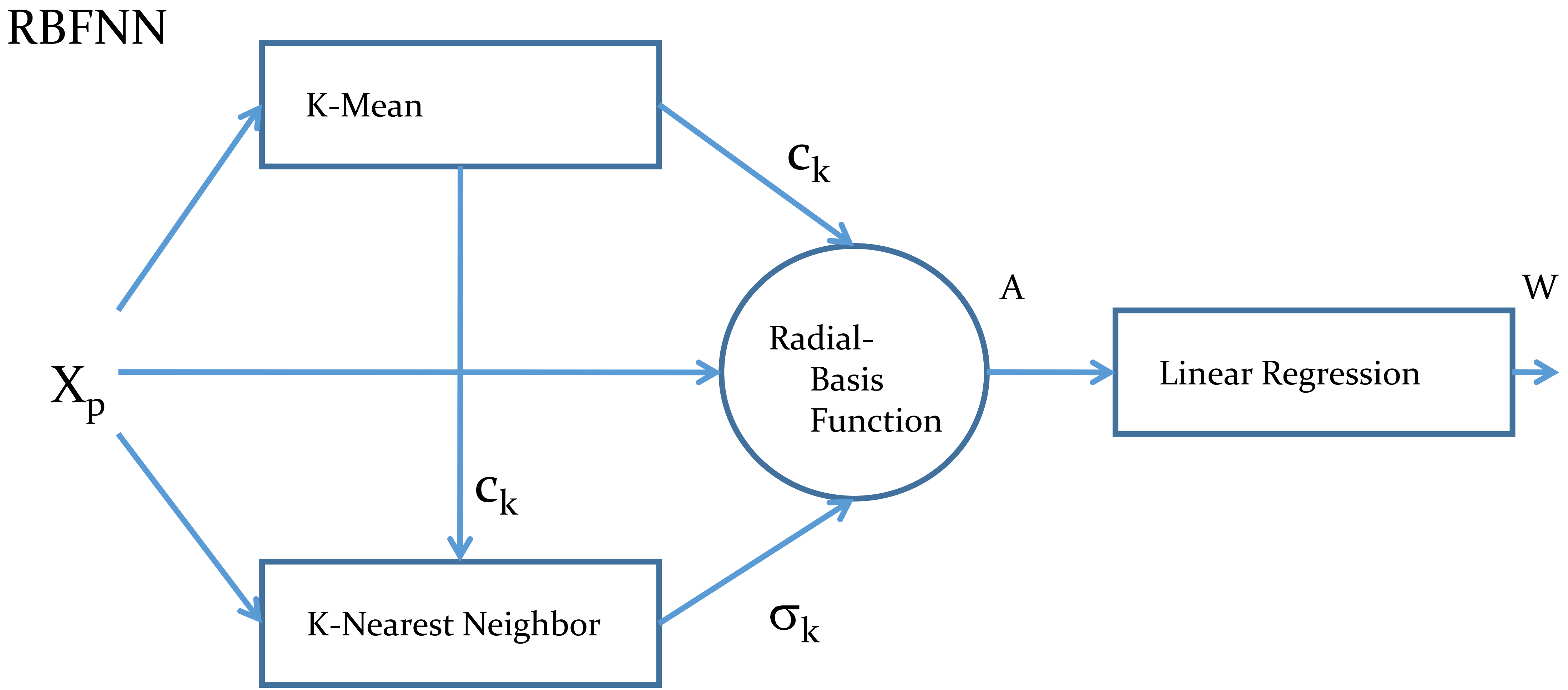
$$\varphi(r) = r^2 \log(r) \quad a \quad thin - plate \quad spline \quad function$$

$$\varphi(r) = \sqrt{\left(r^2 - b^2\right)} \quad a \quad multiquadratic \quad function$$

RBFNN Architecture



Learning Process of RBFNN



- RBF is a kind of **supervised** neural networks
- Design of NN as *curve-fitting* problem
- **Learning**: find surface in multidimensional space best fit to training data by determining $\boldsymbol{w_i}$, $\boldsymbol{\sigma_i}$ and $\mathbf{c}_i$ separately
  - RBF networks solve this problem by dividing the learning into two independent processes.
    - Center and spread learning (or determination)
    - Output layer Weights Learning
- **Generalization**: Use of this multidimensional surface <u>to interpolate the test data</u>

- The response characteristics of the k^{th} hidden unit is given by

$$\varphi_k(X) = \varphi\left(\frac{\|X - c_k\|}{\sigma_k^2}\right)$$

- Where $\phi_k(.)$ is strictly positive radial symmetric function with a unique maximum at k^{th} center c_k and which drop off rapidly to zero away from the center.
- The parameter is the width of the receptive field in the input space for the unit *k*.
- In other words functions σ_k are defined in areas of the corresponding points c_k which causes their sensitive receptive field parameter σ_k that defines the geometric size of the k^{th} receptive field in the input space for unit *k*.

Finding the Weight

- c_i can be find by using k-means algorithm
- The width σ can be find by using k- nearest neighbor rule
- The weights can be determined as follows

$$\begin{bmatrix} y_1(X) \\ y_2(X) \\ \bullet \\ \bullet \\ \bullet \\ y_m(X) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \bullet & \bullet & \bullet & a_{1L} \\ a_{21} & a_{22} & \bullet & \bullet & \bullet & a_{2L} \\ \bullet & \bullet & \bullet & & & \bullet \\ \bullet & \bullet & & \bullet & & \bullet \\ \bullet & \bullet & & & \bullet & \bullet \\ a_{m1} & a_{m2} & \bullet & \bullet & \bullet & a_{mL} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \bullet \\ \bullet \\ \bullet \\ w_L \end{bmatrix}$$

$$Y = AW$$

$$W = A^{-1}Y$$

Finding c_ks by Using k-means Algorithm

Step1: K initial clusters are chosen randomly from the samples to form K groups.

Step2: Each new sample is added to the group whose mean is the closest to this sample.

Step3: Adjust the mean of the group to take account of the new points.

Step4: Repeat step2 until the distance between the old means and the new means of all clusters is smaller than a predefined tolerance.

Outcome: There are K clusters with means representing the centroid of each clusters.

Advantages: (1) A fast and simple algorithm.

(2) Reduce the effects of noisy samples.

Finding the RBF function width σ by Using *K* Nearest Neighbor Rule

- **The objective is to cover the training points so that a smooth fit of the training samples can be achieved**

$$\sigma_i = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\left\| c_k - c_i \right\|^2}$$

k^{th} nearest neighbor of c_i

Conclusion

- The objective is to cover the training points so that a smooth fit of the training samples can be achieved
- The hidden layer RBFNN does not have corresponding weights and threshold.