5.5 Regularization Theory

- In regularization techniques, a suitable auxiliary constraint is applied to an ill-posed problem to make it well-posed.
- The constraint includes some prior information about the solution.
- A typical regularization constraint: smoothness condition.
- Assume that we know N pairs of input vectors $\mathbf{x}_i$ and corresponding desired responses d_i, $i = 1, \ldots, N$.
- The desired responses are here one-dimensional for simplicity (not a limitation)
- Let the approximating function be $F(\mathbf{x}) = F(\mathbf{x}, \mathbf{w})$.

- Regularization theory (Tikhonov, 1963) involves basically two terms:

  1. **Standard Error Term.**

  $$\mathcal{E}_s(F) = \frac{1}{2}\sum_{i=1}^{N}(d_i - y_i)^2 = \frac{1}{2}\sum_{i=1}^{N}[d_i - F(\mathbf{x}_i)]^2$$

     - This measures the error between the desired response d_i and the actual response y_i for the training set $i = 1, \ldots, N$.

  2. **Regularizing Term.**

  $$\mathcal{E}_c(F) = \frac{1}{2} \parallel \mathbf{D}F \parallel^2$$

     - Here $\mathbf{D}$ is a *linear differential operator*.

     - It contains prior information about the form of the solution (input-output mapping $F(\mathbf{x})$).

     - The selection of the *stabilizer* $\mathbf{D}$ is problem-dependent.

- The problem can be treated by representing a continuous function as a vector in a function space.

- The function space is here the L_2 space consisting of all real-valued functions $f(\mathbf{x})$ for which $\| f(\mathbf{x}) \|^2$ is integrable.

- We shall not go into mathematical details in this course.

- The quantity to be minimized in regularization theory is

$$\begin{aligned}\mathcal{E}(F) &= \mathcal{E}_s(F) + \lambda \mathcal{E}_c(F) \\ &= \frac{1}{2}\sum_{i=1}^{N}[d_i - F(\mathbf{x}_i)]^2 + \frac{1}{2}\lambda \| \mathbf{D}F \|^2\end{aligned}$$

- Here λ is a positive real number called the *regularization parameter*.

- The minimizer of the *Tikhonov functional* $\mathcal{E}(F)$ is the solution $F_\lambda(\mathbf{x})$ of the regularization problem.

- If $\lambda \to 0$, the problem is unconstrained, and its solution is completely determined by the training examples.

- On the other hand, if $\lambda \to \infty$, the solution depends completely on the smoothness condition.

- In this case, the training examples are regarded unreliable.
- In practice, the regularization parameter λ has a value between these two extremes.
- The regularizing term $\mathcal{E}_c(F)$ represents a *model complexity-penalty function*.
- The rest of Section 5.5 describes how the regularization problem can be solved.
  - We shall skip most of this highly mathematical and advanced theory.
  - Let us summarize some main points very briefly.
- First, so-called Frechet differential is used to differentiate the Tikhonov functional.
- Then Rietz representation theorem is applied for representing the result in a more suitable form.

- After some manipulations, Euler-Lagrange equation can be derived for the Tikhonov functional $\mathcal{E}(F)$.

$$\tilde{\mathbf{D}}\mathbf{D}F_\lambda(\mathbf{x}) = \frac{1}{\lambda}\sum_{i=1}^{N}[d_i - F(\mathbf{x}_i)]\delta(\mathbf{x} - \mathbf{x}_i)$$

- Using so-called Green's functions defined for differential operators, the solution of the regularization problem can be represented in the form

$$F_\lambda(\mathbf{x}) = \frac{1}{\lambda}\sum_{i=1}^{N}[d_i - F(\mathbf{x}_i)]G(\mathbf{x}, \mathbf{x}_i)$$

- In the next subsection, the coefficients of the expansion are determined.
- This yields as the solution of the regularization problem the expansion

$$F_\lambda(\mathbf{x}) = \sum_{i=1}^{N} w_i G(\mathbf{x}, \mathbf{x}_i)$$

- Here w_i is the ith element of the weight vector $\mathbf{w}$ defined by

$$(\mathbf{G} + \lambda \mathbf{I})\mathbf{w} = \mathbf{d}$$

- $G(\mathbf{x}, \mathbf{x}_i)$ is the Green's function defined by the partial differential equation

$$\mathbf{L}G(\mathbf{x}, \xi) = \delta(\mathbf{x} - \xi)$$

- Assume now that the stabilizer (smoothing operator) $\mathbf{D}$ is required to be both translationally and rotationally invariant.

- This case is important in practice.

- Then the Green's function must be a *radial-basis function*:

$$G(\mathbf{x}, \mathbf{x}_i) = G(\| \mathbf{x} - \mathbf{x}_i \|)$$

- The regularized solution

$$F_\lambda(\mathbf{x}) = \sum_{i=1}^{N} w_i G(\mathbf{x}, \mathbf{x}_i)$$

  becomes in this case

$$F_\lambda(\mathbf{x}) = \sum_{i=1}^{N} w_i G(\| \mathbf{x} - \mathbf{x}_i \|)$$

- This solution has a similar form as the standard RBF solution given by:

$$F(\mathbf{x}) = \sum_{i=1}^{N} w_i \varphi(\| \mathbf{x} - \mathbf{x}_i \|)$$

- Both the solutions use a strict interpolation technique.
- There are as many (N) basis functions as training data points.
- However, the first solution is regularized while the latter one is not.

- The two solutions become the same only when the regularization parameter $\lambda = 0$.

- A translationally and rotationally invariant Green's function is the *multivariate Gaussian function* defined by

$$G(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\frac{1}{2\sigma_i^2} \parallel \mathbf{x} - \mathbf{x}_i \parallel^2\right)$$

- For this important choice, the regularizing solution becomes a linear superposition of multivariate Gaussian functions:

$$F_\lambda(\mathbf{x}) = \sum_{i=1}^{N} w_i \exp\left(-\frac{1}{2\sigma_i^2} \parallel \mathbf{x} - \mathbf{x}_i \parallel^2\right)$$

- The weights w_i are solved from the equation:

$$(\mathbf{G} + \lambda\mathbf{I})\mathbf{w} = \mathbf{d}$$

- $\mathbf{G}$ is the *Green's matrix* with elements $G(\mathbf{x}_i, \mathbf{x}_j)$.

$$\mathbf{G} = \left[\begin{array}{cccc} G(\mathbf{x}_1, \mathbf{x}_1) & G(\mathbf{x}_1, \mathbf{x}_2) & \cdots & G(\mathbf{x}_1, \mathbf{x}_N) \\ G(\mathbf{x}_2, \mathbf{x}_1) & G(\mathbf{x}_2, \mathbf{x}_2) & \cdots & G(\mathbf{x}_2, \mathbf{x}_N) \\ \vdots & \vdots & & \vdots \\ G(\mathbf{x}_N, \mathbf{x}_1) & G(\mathbf{x}_N, \mathbf{x}_2) & \cdots & G(\mathbf{x}_N, \mathbf{x}_N) \end{array} \right]$$

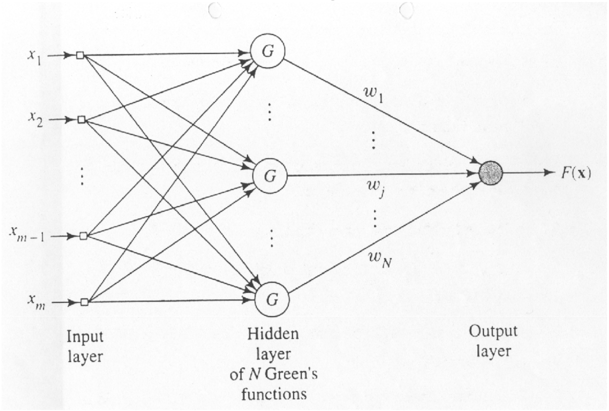
- The Gaussians have often the same variances $\sigma_i = \sigma$; such RBF networks are still universal approximators.

5.6 Regularization Networks

- The regularized approximating function

$$F_\lambda(\mathbf{x}) = \sum_{i=1}^{N} w_i G(\mathbf{x}, \mathbf{x}_i)$$

can be implemented using the network structure shown below.



- This three-layer network is called a *regularization network*.

- In the first layer the components of the input vector $\mathbf{x}$ are inputted to the network.

- The second (hidden) layer has nonlinear units connected directly to all the nodes of the input layer.

- There is one hidden unit for each data point $\mathbf{x}_i$, $i = 1, \ldots, N$.

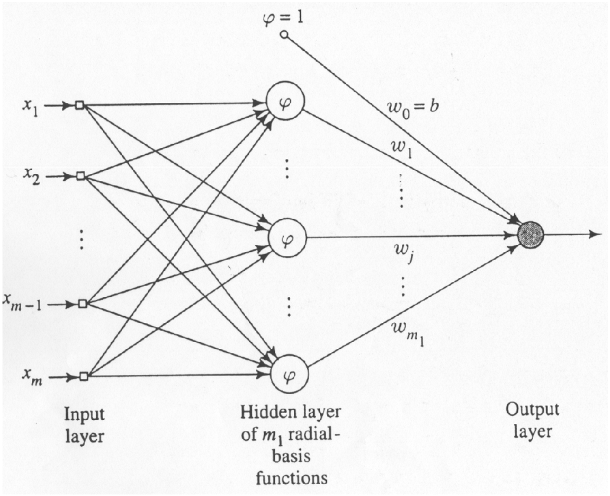
- The output of the ith hidden unit is defined by the Green's function $G(\mathbf{x}, \mathbf{x}_i)$.

- The output layer consists of a single linear unit fully connected to the hidden layer.

- The weights of the output layer are the unknown coefficients of the expansion.

- The architecture of figure can easily be generalized for any desired number of outputs.

- If the Green's function $G(\mathbf{x}, \mathbf{x}_i)$ is *positive definite* for all i, following RBF network can be used.



- This holds for example for the multivariate Gaussian function.

- The regularization network has three desirable properties:

  1. *Universal approximator*: it can approximate arbitrarily well any multivariate continuous function if there are enough hidden units.
  2. *Best-approximation property*: the output layer is linear with respect to the unknown coefficients.
  3. *Optimality*: it minimizes the Tikhonov cost functional.

5.7 Generalized Radial-Basis Function Networks

- If the number N of the training vectors $\mathbf{x}_i$ grows large, the basic regularization network may become prohibitively expensive to implement.
- Reason: there are as many hidden nodes as training points.
- The weights of the regularization network are solved from the equation:

$$\mathbf{w} = (\mathbf{G} + \lambda\mathbf{I})^{-1}\mathbf{d}$$

- Here the element (i, j) of the $N \times N$ matrix $\mathbf{G}$ is $G(\mathbf{x}_i, \mathbf{x}_j)$.
- $\mathbf{I}$ is the $N \times N$ unit matrix.
- The desired response vector $\mathbf{d} = [d_1, d_2, \ldots, d_N]^T$, and the weight vector $\mathbf{w} = [w_1, w_2, \ldots, w_N]^T$.
- The computational load needed for inversing the matrix grows roughly cubically (as N^3) with N.

- The complexity of the network can be reduced by approximating the regularized solution.

$$F_\lambda(\mathbf{x}) = \sum_{i=1}^{N} w_i G(\mathbf{x}, \mathbf{x}_i)$$

- This can be done by using *Galerkin's method*.
- There the approximated solution $F^*(\mathbf{x})$ is expanded on a finite basis:

$$F^*(\mathbf{x}) = \sum_{i=1}^{m_1} w_i \varphi_i(\mathbf{x})$$

- Typically, the number m_1 of the basis functions $\varphi_i(\mathbf{x})$ is less than the number N of the data points.
- With radial-basis functions in mind, we set

$$\varphi_i(\mathbf{x}) = G(\| \mathbf{x} - \mathbf{t}_i \|), \;\; i = 1, 2, \ldots, m_1$$

- The set of centers $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_{m_1}$ should be determined.

- It turns out that the optimal weight vector for the approximated solution $F^*(\mathbf{x})$ is obtained by solving the equation

$$(\mathbf{G}^T\mathbf{G} + \lambda\mathbf{G}_0)\mathbf{w} = \mathbf{G}^T\mathbf{d}$$

- The non-square $N \times m_1$ matrix $\mathbf{G}$ is defined:

$$\mathbf{G} = \begin{bmatrix} G(\mathbf{x}_1, \mathbf{t}_1) & G(\mathbf{x}_1, \mathbf{t}_2) & \cdots & G(\mathbf{x}_1, \mathbf{t}_{m_1}) \\ G(\mathbf{x}_2, \mathbf{t}_1) & G(\mathbf{x}_2, \mathbf{t}_2) & \cdots & G(\mathbf{x}_2, \mathbf{t}_{m_1}) \\ \vdots & \vdots & & \vdots \\ G(\mathbf{x}_N, \mathbf{t}_1) & G(\mathbf{x}_N, \mathbf{t}_2) & \cdots & G(\mathbf{x}_N, \mathbf{t}_{m_1}) \end{bmatrix}$$

  its element (i, j) is $G(\mathbf{x}_i, \mathbf{t}_j)$.

- The element (i, j) of the symmetric square $m_1 \times m_1$ matrix $\mathbf{G}_0$ is respectively $G(\mathbf{t}_i, \mathbf{t}_j)$.

- The mathematical derivation is skipped here (see Haykin, pp. 279-280, for details).

- Consider the limiting case where the regularization parameter λ approaches zero.

- Then the weight vector $\mathbf{w}$ converges to the pseudoinverse solution

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

- This is the optimal solution of the overdetermined least-squares data fitting problem where $m_1 < N$.

Weighted Norm

- Instead of the standard Euclidean norm, it is sometimes more appropriate to use a more general weighted norm in radial-basis functions.

- This is defined by the quadratic form

$$\| \mathbf{x} \|_C^2 = (\mathbf{C}\mathbf{x})^T(\mathbf{C}\mathbf{x}) = \mathbf{x}^T\mathbf{C}^T\mathbf{C}\mathbf{x}$$

- Here $\mathbf{C}$ is $m_0 \times m_0$ norm weighting matrix, and m_0 is the dimension of the input vector $\mathbf{x}$.

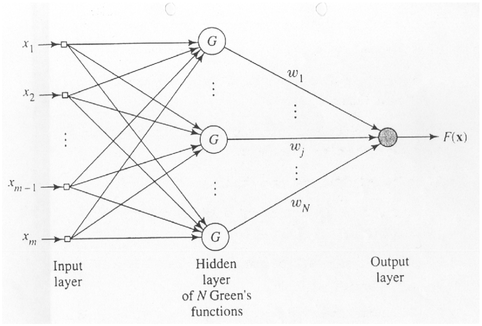
- For Gaussian radial-basis functions, the use of the weighted norm means that we replace the basis function

$$G(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\frac{1}{2\sigma_i^2} \| \mathbf{x} - \mathbf{x}_i \|^2\right)$$

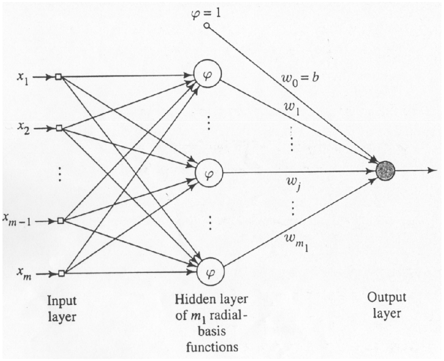
  by

$$G(\| \mathbf{x} - \mathbf{t}_i \|_C) = \exp[-(\mathbf{x} - \mathbf{t}_i)^T\mathbf{C}^T\mathbf{C}(\mathbf{x} - \mathbf{t}_i)]$$

- Here $\mathbf{C}^T\mathbf{C} = \frac{1}{2}\mathbf{\Sigma}^{-1}$.
- Using this notation, $G(\| \mathbf{x} - \mathbf{t}_i \|_C)$ represents a multivariate Gaussian distribution with mean vector $\mathbf{t}_i$ and covariance matrix $\mathbf{\Sigma}$.
- The approximative solution obtained using Galerkin's method provides the framework for the *generalized radial-basis function (RBF) network*.
- Structurally, the generalized RBF network is similar to the regularization network (except for the bias term).
- However, they differ from each other in two important ways:



Regularized network



Generalized RBF network

1. The number of nodes m_1 in the hidden layer is smaller than the number N of training vectors in the generalized RBF network.

   - In the regularized RBF network these numbers are the same.

2. In the regularized RBF network, the only unknown parameters are the linear weights of the output layer.

   - In the generalized RBF network, the center positions of the radial-basis functions as well as the norm weighting matrix are also unknown in addition to the weight vector.

Receptive Field

- The covariance matrix $\mathbf{\Sigma}$ determines the receptive field of the Gaussian radial-basis function

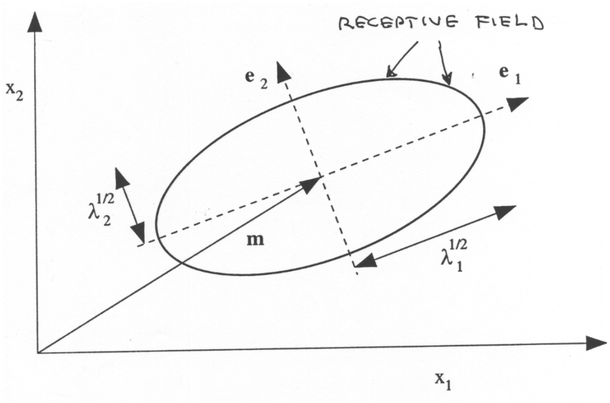
$$\begin{aligned} G(\| \mathbf{x} - \mathbf{t}_i \|_C) &= \exp[-(\mathbf{x} - \mathbf{t}_i)^T \mathbf{C}^T \mathbf{C} (\mathbf{x} - \mathbf{t}_i)] \\ &= \exp[-\tfrac{1}{2} (\mathbf{x} - \mathbf{t}_i)^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \mathbf{t}_i)] \end{aligned}$$

- The receptive field of $G(\| \mathbf{x} - \mathbf{t}_i \|_C)$ is the domain of the input vectors $\mathbf{x}$ for which

$$G(\| \mathbf{x} - \mathbf{t}_i \|_C) > \alpha$$

  where α is a positive constant.

- An illustration of a general Gaussian radial basis function and its receptive field.



- m is the mean vector of the Gaussian
- the eigenvectors e_1 and e_2 of the covariance matrix $\mathbf{\Sigma}$ determine the direction of the hyperellipsoid.
- the lengths of the axes are determined by the eigenvalues λ_1 and λ_2 of the covariance matrix.
- now $\lambda_1 = \sigma_1^2 \rightarrow \lambda_1^{\frac{1}{2}} = \sigma_1$, similarly $\lambda_2^{\frac{1}{2}} = \sigma_2$ (standard dev.).

- We can specify three different types of the covariance matrix $\mathbf{\Sigma}$ and the respective receptive field:

  1. $\mathbf{\Sigma} = \sigma^2\mathbf{I}$, where σ^2 is a common variance.

     - The receptive field is a hypersphere centered at $\mathbf{t}_i$ with a radius σ.

  2. $\mathbf{\Sigma} = \text{diag}(\sigma_1^2, \sigma_2^2, \dots, \sigma_{m_0}^2)$. Thus the covariance matrix $\mathbf{\Sigma}$ is diagonal with different variances σ_i^2.

     - The receptive field is a hyperellipse; the variances determine the length of the axes.

  3. $\mathbf{\Sigma}$ is a nondiagonal positive definite matrix.

     - Diagonalization using a similarity transformation shows that the situation is essentially similar than for a diagonal covariance matrix, but the orientation of $\mathbf{\Sigma}$ is rotated.

5.8 XOR Problem (Revisited)

- In this section, solution of the XOR problem using an RBF network is studied.

- Network uses Gaussian basis functions

$$G(\| \mathbf{x} - \mathbf{t}_i \|) = \exp(-\| \mathbf{x} - \mathbf{t}_i \|^2), \mathsf{i} = 1{,}2$$

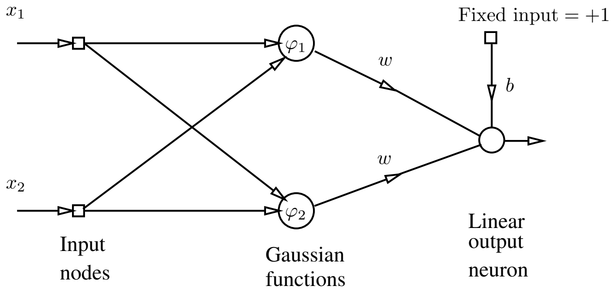
  where centers $\mathbf{t}_1$ and $\mathbf{t}_2$ are

$$\begin{aligned} \mathbf{t}_1 &= [1, 1]^T \\ \mathbf{t}_2 &= [0, 0]^T \end{aligned}$$

- The output unit uses *weight-sharing*, justified by the symmetry of the problem.

- Therefore, the weight vector $\mathbf{w}$ must be computed using the pseudoinverse solution.

- The network uses also a bias term.



- The input-output relation of the network is defined by

$$y(\mathbf{x}) = \sum_{i=1}^{2} wG(\| \mathbf{x} - \mathbf{t}_i \|) + b$$

Input-output transformation computed for XOR Problem

Data Point j	Input Pattern, $\mathbf{x}_j$	Desired output, d_j
1	(1,1)	0
2	(0,1)	1
3	(0,0)	0
4	(1,0)	1

- To fit training data, we require that

$$y(\mathbf{x}_j) = d_j \ , \ \ j = 1, 2, 3, 4$$

where $\mathbf{x}_j$ is an input vector and d_j corresponding desired output.

- Let

$$g_{ji} = G(\| \mathbf{x}_j - \mathbf{t}_i \|) \ , \ \ j = 1, 2, 3, 4; \, i = 1, 2$$

- Then the equations can be written in matrix form

$$\mathbf{G}\mathbf{w} = \mathbf{d}$$

where

$$\mathbf{G} = \begin{bmatrix} 1 & 0.1353 & 1 \\ 0.3678 & 0.3678 & 1 \\ 0.1353 & 1 & 1 \\ 0.3678 & 0.3678 & 1 \end{bmatrix}$$

$$\mathbf{d} = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix}^T$$

$$\mathbf{w} = \begin{bmatrix} w & w & b \end{bmatrix}^T$$

- Problem is overdetermined in the sense that we have more data points than free parameters.
  - That is why $\mathbf{G}$ is not square.
  - No unique inverse exists for $\mathbf{G}$.
- Problem can be solved using pseudoinverse solution.

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

- $\mathbf{G}^T\mathbf{G}$ is a square matrix with unique inverse.

- After substitutions we get

$$\mathbf{G}^{+} = \begin{bmatrix} 1.8292 & -1.2509 & 0.6727 & -1.2509 \\ 0.6727 & -1.2509 & 1.8292 & -1.2509 \\ -0.9202 & 1.4202 & -0.9202 & 1.4202 \end{bmatrix}$$

and

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

5.10 Approximation Properties of RBF Networks

- Multilayer perceptrons have the universal approximation property.
- Also the family of RBF networks can uniformly approximate any continuous function on a compact set.
- Formally, let $G : \mathcal{R}^n \to \mathcal{R}$ be integrable, continuous, and bounded function satisfying the condition

$$\int_{\mathcal{R}^n} G(\mathbf{x}) d\mathbf{x} \neq 0$$

- Let $\mathcal{F}_G$ denote the family of RBF networks consisting of functions $F : \mathcal{R}^n \to \mathcal{R}$

$$F(\mathbf{x}) = \sum_{i=1}^{m_1} w_i G\left(\frac{\mathbf{x} - \mathbf{t}_i}{\sigma}\right)$$

- Here $\sigma > 0$, $w_i \in \mathcal{R}$ and $\mathbf{t}_i \in \mathcal{R}^n$ for $i = 1, 2, \ldots, m_1$.

The universal approximation theorem for RBF networks:

- For any continuous input-output mapping function f(**x**) there is an RBF network with
  - a set of centers $\mathbf{t}_i$, $i = 1, 2, \ldots, m_1$ and a common width σ such that
  - the input-output mapping function $F(\mathbf{x})$ realized by the RBF network is close to $f(\mathbf{x})$ in the L_p norm, $p \in [1, \infty)$.
- Note that the kernel $G : \mathcal{R}^n \rightarrow \mathcal{R}$ need not be radially symmetric.
- The theorem provides a theoretical basis for using RBF networks in practical applications.

5.11 Comparison of RBF Networks and Multilayer Perceptrons

- Both RBF and MLP networks are nonlinear layered networks having universal approximation properties.

- The most important differences between them are:

  1. An RBF network has a single hidden layer, while an MLP can have several hidden layers.
  2. The computational nodes in the MLP network are similar in various layers, while in the RBF network they are quite different in the output and hidden layers.
  3. In the RBF network, the output layer is linear, while it is usually nonlinear in an MLP network.
  4. In each hidden node, the activation function of RBF network computes an *Euclidean distance*, while in MLP networks an *inner product* between the input and the weight vector is computed.

5. MLPs construct *global* approximations, while RBF networks approximate *locally* nonlinear input-output mappings.

- MLP may require less parameters than the RBF network for achieving the same accuracy.