## 4.18 Supervised Learning Viewed as an Optimization Problem

- The supervised training of a multilayer perceptron (MLP) is now viewed as a *numerical optimization* problem.
- The error surface of a MLP is a highly nonlinear function of the weight vector $\mathbf{w}$.
- Let $\mathcal{E}_{av}(\mathbf{w})$ denote the cost function, averaged over the training sample set.

- Recall now from Section 3.3 the Taylor series expansion of a scalar function $\mathcal{E}(\mathbf{w})$ of (the components of) the vector $\mathbf{w}$:

$$\mathcal{E}(\mathbf{w}+\Delta\mathbf{w}) = \mathcal{E}(\mathbf{w}) + \mathbf{g}^T\Delta\mathbf{w} + \frac{1}{2}(\Delta\mathbf{w})^T\mathbf{H}\Delta\mathbf{w} + \cdots$$

- Here $\Delta\mathbf{w}$ is a (small) correction or update term.

- $\mathbf{g}$ is the gradient vector of $\mathcal{E}(\mathbf{w})$, and $\mathbf{H}$ its Hessian matrix, both evaluated at the point $\mathbf{w}$.

- When applied to $\mathcal{E}_{av}(\mathbf{w})$ with dependences on $n$ included, the Taylor series expansion becomes

$$\begin{aligned}\mathcal{E}_{av}(\mathbf{w}(n)+\Delta\mathbf{w}(n)) &= \mathcal{E}_{av}(\mathbf{w}(n)) + \mathbf{g}^T(n)\Delta\mathbf{w}(n) \\ &+ \frac{1}{2}\Delta\mathbf{w}^T(n)\mathbf{H}(n)\Delta\mathbf{w}(n) + \cdots\end{aligned}$$

- The local gradient $\mathbf{g}(n)$ is defined by evaluating the quantity

$$\mathbf{g}(n) = \frac{\partial \mathcal{E}_{av}(\mathbf{w})}{\partial \mathbf{w}}$$

  at the point $\mathbf{w} = \mathbf{w}(n)$.

- The local Hessian matrix $\mathbf{H}(n)$ is similarly defined by

$$\mathbf{H}(n) = \frac{\partial^2 \mathcal{E}_{av}(\mathbf{w})}{\partial \mathbf{w}^2},$$

  evaluated also at the operating point $\mathbf{w} = \mathbf{w}(n)$.

- Because the ensemble averaged cost function $\mathcal{E}_{av}(\mathbf{w})$ is used here, a batch mode of learning is presumed.

- In the steepest descent method, the adjustment $\Delta\mathbf{w}(n)$ applied to the weight vector $\mathbf{w}(n)$ is defined by the negative gradient vector

$$\Delta\mathbf{w}(n) = -\eta\mathbf{g}(n)$$

  where $\eta$ is the learning-rate parameter.

- An example of this is the back-propagation algorithm in the batch mode.
- In effect, the steepest descent method uses a *linear approximation* of the cost function around the operating point $\Delta\mathbf{w}(n)$.
- There the gradient vector is the only source of local information about the error surface.
- Advantage: simplicity of implementation.
- Drawback: convergence can be very slow in large-scale problems.
- Inclusion of the momentum term is a crude attempt to use some second-order information about the error surface.
- It helps somewhat, but makes the training process more delicate to manage.
- Reason: the designer must "tune" one more parameter.

- For improving significantly the convergence speed of MLP training, one must use *higher-order information*.
- This can be done by using a *quadratic approximation* of the error surface around the current point $\mathbf{w}(n)$.
- It is easy to see that the optimum value of the update $\Delta\mathbf{w}(n)$ of the weight vector $\mathbf{w}(n)$ is

$$\Delta\mathbf{w}^*(n) = \mathbf{H}^{-1}(n)\mathbf{g}(n)$$

- Here it is assumed that the inverse $\mathbf{H}^{-1}(n)$ of the Hessian matrix $\mathbf{H}(n)$ exists.
- The above formula is the essence of *Newton's method* (Section 3.3)
- However, the practical application of Newton's method to supervised training of MLP's is handicapped by the following factors:
  - One must calculate the inverse Hessian $\mathbf{H}^{-1}(n)$, which can be computationally expensive.

- The Hessian matrix $\mathbf{H}(n)$ must be nonsingular so that $\mathbf{H}^{-1}(n)$ exists. This condition does not necessarily always hold.
- When the cost function $\mathcal{E}_{av}(\mathbf{w})$ is nonquadratic, there is no guarantee for the convergence of Newton's method.

- Some of these difficulties can be overcome by using a *quasi-Newton* method.
- This requires an estimate of the gradient vector $\mathbf{g}$ only.
- However, even the quasi-Newton methods are computationally too expensive except for the training of very small-scale neural networks.
- They are described at the end of Section 4.18; skipped in our course.
- Another class of second-order optimization methods: *conjugate-gradient methods*.
- Somewhat intermediate between the steepest descent and Newton's method.

- They need not the Hessian matrix, avoiding the difficulties associated with its evaluation, storage, and inversion.
- Essential idea of conjugate-gradient methods: a more sophisticated update direction than the gradient is used.
- Conjugate-gradient methods are applicable also to large-scale problems involving hundreds or thousands of adjustable parameters.
- They are well suited for the training of a MLP network, too.

## Nonlinear Conjugate Gradient Algorithm for the Training of an MLP

- Initialization
- Computation

1. For $\mathbf{w}(0)$, use back-propagation to compute the gradient vector $\mathbf{g}(0)$.
2. Set $\mathbf{s}(0) = \mathbf{r}(0) = -\mathbf{g}(0)$.
3. At time step $n$, use a line search to find $\eta(n)$ that minimizes $\mathcal{E}_{av}(\eta)$ sufficiently, representing $\mathcal{E}_{av}(\eta)$ expressed as a function of $\eta$ for fixed $\mathbf{w}$ and $\mathbf{s}$.
4. Test to determine if Euclidean norm of the residual $\mathbf{r}(n)$ has fallen below a specified value, that is a small fraction of the initial value.
5. Update the weight vector:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta(n)\mathbf{s}(n)$$

6. For $\mathbf{w}(n+1)$, use backprop to compute the updated gradient vector $\mathbf{g}(n+1)$.

7. Set $\mathbf{r}(n+1) = -\mathbf{g}(n+1)$.
8. Use Polak-Ribiére method to calculate $\beta(n+1)$:

$$\beta(n+1) = \text{max}\left\{ \frac{\mathbf{r}^T(n+1)(\mathbf{r}(n+1) - \mathbf{r}(n))}{\mathbf{r}^T(n+1)\mathbf{r}(n)}, 0 \right\}$$

9. Update the direction vector:

$$\mathbf{s}(n+1) = \mathbf{r}(n+1) + \beta(n+1)\mathbf{s}(n)$$

10. Set $n = n+1$, and go back to step 3.

- Stopping criterion. Terminate the algorithm when the following condition is satisfied:

$$||\mathbf{r}(n)|| \le \varepsilon ||\mathbf{r}(0)||$$

where $\varepsilon$ is a prescribed small number.

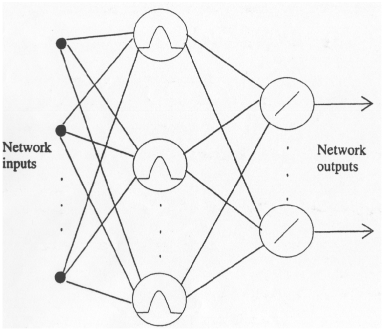
- Note: The Gauss-Newton method discussed in Section 3.3 is available in MATLAB for training MLP networks.
- The stabilized Gauss-Newton formula (3.23) is called there Levenberg-Marquardt's algorithm.

# 5. Radial-Basis Function Networks

## 5.1 Introduction

- Back-propagation algorithm may be viewed as a *stochastic approximation* technique for supervised training of a MLP network.
- Now a completely different viewpoint is taken on the design of a neural network.
- It is treated as a *curve-fitting (approximation) problem* in a high-dimensional space.
- Learning is then equivalent to finding a surface in a multidimensional space that provides the best fit to the training data (in some statistical sense).

- Generalization is equivalent to using the found surface to interpolate the test data.
- Such a philosophy lies behind the method of radial-basis functions.
- The hidden units provide a set of “functions” that constitute an arbitrary basis for the input vectors (patterns).
- These functions are called *radial-basis functions*.
- The radial basis functions were first introduced in the solution of the real multivariate interpolation problem.
- They are currently one of the main fields of research in numerical analysis.



- A basic *radial-basis function (RBF) network* consists of three layers having entirely different roles:

    1. Input layer is made up of source nodes (sensory units).
    2. The hidden layer applies a nonlinear transformation from the input space to the hidden space.

       - RBF networks have only one, often high-dimensional hidden layer.
    3. A linear output layer.

- The hidden space is usually chosen high-dimensional because of two reasons:

  1. Pattern vectors are more likely to be linearly separable in a high-dimensional space.
  2. The approximation ability of the network is the better the more there are hidden units.

- In this course, we concentrate on basic RBF networks.
- We go through the first and last sections of Chapter 5.
- Most of the regularization theory (Sections 5.5-5.9) will be skipped.

## 5.2 Cover's Theorem on the Separability of Patterns

- Consider the use of a RBF network for a complex pattern classification task.
- The problem is basically solved by transforming it into a high-dimensional space in a nonlinear manner.
- Justification: *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.*
- If the patterns are linearly separable, the classification problem is fairly easy to solve.
- In the following, the separability of patterns is studied.
- This yields a lot of insight into the operation of RBF networks.
- Consider a family of surfaces.

- Each surface divides an input space into two regions.
- $\mathcal{H} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ is a set of $N$ pattern vectors.
- Each pattern vector belongs to one of the two classes $\mathcal{H}_1$ or $\mathcal{H}_2$.
- This kind of binary partition is called a *dichotomy*.
- A dichotomy is called separable with respect to a family of surfaces if there exists a surface separating the points in class $\mathcal{H}_1$ from those in class $\mathcal{H}_2$.
- Let $\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \ldots, \varphi_{m_1}(\mathbf{x})$ be a set of $m_1$ real-valued functions.
- Using these functions, we can define for each pattern $\mathbf{x} \in \mathcal{H}$ the vector

$$\mathbf{f}(\mathbf{x}) = \ [\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \ldots, \varphi_{m_1}(\mathbf{x})]^T$$

- Assume now that $\mathbf{x}$ is a $m_0$-dimensional vector.
- Then the function $\mathbf{f}(\mathbf{x})$ maps points in $m_0$-dimensional input space into corresponding points in a new space of dimension $m_1$.

- $\varphi_i(\mathbf{x})$ is referred to as a *hidden function*.
- The space spanned by the functions $\varphi_1(\mathbf{x}), \ldots, \varphi_{m_1}(\mathbf{x})$ is called the *hidden space* or *feature space*.
- The hidden functions have a similar role as hidden units in an MLP network.
- A dichotomy $[\mathcal{H}_1, \mathcal{H}_2]$ of $\mathcal{H}$ is said to be $\varphi - separable$ if there exists an $m_1$-dimensional vector $\mathbf{w}$ satisfying the condition

$$\begin{array}{rcll} \mathbf{w}^T\mathbf{f}(\mathbf{x}) & > & 0, & \mathbf{x} \in \mathcal{H}_1 \\ \mathbf{w}^T\mathbf{f}(\mathbf{x}) & < & 0, & \mathbf{x} \in \mathcal{H}_2 \end{array}$$

- The hyperplane defined by the equation

$$\mathbf{w}^T\mathbf{f}(\mathbf{x}) = 0$$

describes the separating surface in the hidden $\varphi$-space.

- The inverse image of this subspace, that is,

$$\mathbf{x}: \quad \mathbf{w}^T\mathbf{f}(\mathbf{x}) = 0$$

  defines the *separating surface* in the input space.

- Consider a class of mappings defined by a linear combination of $r$-wise products of the pattern vector coordinates.
  (Käsitellään kuvausta, jonka määrittelee $r$:n hahmovektorin koordinaattien tulojen lineaarikombinaatiot)

- The separating surface corresponding to such a mapping is given by the $r$th degree homogenous equation

$$\sum_{0 \le i_1 \le \ldots \le i_r \le m_0} a_{i_1 i_2 \ldots i_r} x_{i_1} x_{i_2} \ldots x_{i_r} \; = 0$$

- Here $x_i$ is the $i$th component of the input vector $\mathbf{x}$ having $m_0$ components.

- $x_0$ is set to unity for expressing the equation in homogenous form.

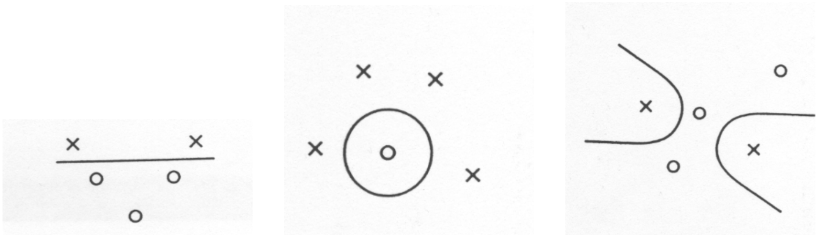
- Such surfaces are called $r$th-order rational varieties.

- An $r$th order product $x_{i_1}x_{i_2}\dots x_{i_r}$ of entries $x_i$ of $\mathbf{x}$ is called a monomial.

- Examples of rational varieties are:

  - hyperplanes (first-order rational varieties)
  - quadrics (second-order rational varieties)
  - hyperspheres (second-order rational varieties with certain linear constraints on the coefficients).

- Three examples of $\varphi$-separable dichotomies of different sets of five points in 2D.



(a) Linearly (b) spherically (c) quadrically
separable dichotomy

- In general, linear separability implies spherical separability which implies quadric separability.

- The converses are not necessarily true.
- Consider now the probability that a particular dichotomy picked at random is $\varphi$-separable.
- The higher is the dimension $m_1$ of the hidden space, the closer is this probability to unity.
- The required assumptions and the result are described somewhat in more detail on pp. 259-260 in Haykin's book.
- Even though here the hidden-unit surfaces have a polynomial form, the result is generally applicable.

- Summarizing, Cover's theorem on the separability of patterns has two basic ingredients:

  1. Nonlinear formulation of the hidden functions $\varphi_i(\mathbf{x})$, $i = 1, 2, \ldots, m_1$.
  2. High dimensionality of the hidden space compared to the input space.

- Sometimes the use of nonlinear mapping alone without increasing the dimensionality is sufficient for producing linear separability.

## Example 5.1: The XOR Problem

- For illustrating the importance of $\varphi$-separability, consider again the simple yet important XOR problem.

- Four points (patterns) (1,1), (0,1), (0,0), and (1,0) in a two-dimensional input space .

- Requirement: construct a binary classifier with output:
  - 0 for the inputs (1,1) or (0,0)
  - 1 for the inputs (1,0) or (0,1).

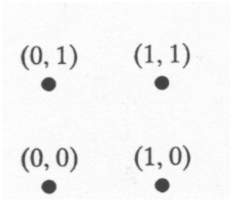
- Recall that the XOR problem is not linearly separable in the original input space.

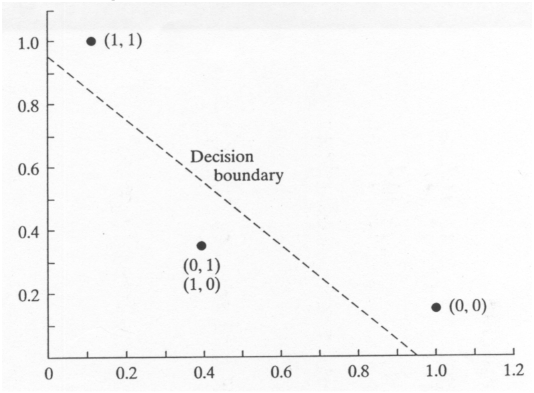
- Define a pair of Gaussian hidden functions

$$\varphi_1(\mathbf{x}) = \exp(- \| \mathbf{x} - \mathbf{t}_1 \|^2), \quad \mathbf{t}_1 = [1, 1]^T$$

$$\varphi_2(\mathbf{x}) = \exp(- \| \mathbf{x} - \mathbf{t}_2 \|^2), \quad \mathbf{t}_2 = [0, 0]^T$$

- The 4 input patterns are mapped onto the $\varphi_1$ - $\varphi_2$ plane as shown in the table and figure on the right.





| Input Pattern $\mathbf{x}$ | First Hidden Function $\varphi_1(\mathbf{x})$ | Second Hidden Function $\varphi_2(\mathbf{x})$ |
|---|---|---|
| (1,1) | 1 | 0.1353 |
| (0,1) | 0.3678 | 0.3678 |
| (0,0) | 0.1353 | 1 |
| (1,0) | 0.3678 | 0.3678 |

- In this hidden space, the XOR problem becomes linearly separable.

- In this simple problem, the hidden space has the same dimensionality as the input space.

## Separating Capacity of a Surface

- Assume that we have a set of randomly assigned pattern vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ in a multidimensional space.
- In this subsection, the expected maximum number of linearly separable patterns is studied.
- Using probability theory and the definition of a negative binomial distribution, the following result can be derived.
- *The expected maximum number of randomly assigned vectors that are linearly separable in a space of dimensionality $m_1$ is equal to $2m_1$.*
- This is the celebrated asymptotic corollary of Cover's theorem (1965).

- See Haykin, pp. 261-262 for a derivation.
- Think of a family of decision surfaces having $m_1$ degrees of freedom.
- The corollary suggests that $2m_1$ is a natural definition of the *separating capacity* of this family.

## 5.3 Interpolation Problem

- The important point emerging from Cover's theorem:
- One can often gain practical benefit by mapping the input space nonlinearly into a sufficiently high-dimensional space.
- In this way, a nonlinearly separable classification problem can be transformed into a linearly separable one.
- Similarly, we may use a nonlinear mapping in filtering.
- A difficult nonlinear filtering problem can possibly be handled using linear filtering in a higher-dimensional space.
- Consider now a feedforward network with an input layer, a single hidden layer, and an output layer.
- The output layer has only one neuron for simplicity.
- The network is designed to perform a *nonlinear mapping* from the input space to the hidden space (layer).

- This is followed by a *linear mapping* from the hidden space to the output space (layer).

- The overall input-output mapping

$$s : \mathcal{R}^{m_0} \rightarrow \mathcal{R}^1$$

  is from $m_0$-dimensional input space to one-dimensional output space.

- The map $s$ can be thought as a *hypersurface* $\Gamma$ in a $m_0+1$ dimensional space.

- Example: a function $s : \mathcal{R}^1 \rightarrow \mathcal{R}^1$, where $s(x) = x^2$, is parabola in $\mathcal{R}^2$ space.

- The hypersurface $\Gamma$ is a multidimensional plot of the output as a function of the input vector.

- In a practical situation, $\Gamma$ is usually unknown, and the training data are noisy.

- Then the learning process can be viewed as follows:

- In the training phase, the surface $\Gamma$ is fitted to the known input-output pairs (training data).
- A suitable optimization technique is used in fitting.
- The generalization phase corresponds to interpolation between the data points.
- The interpolation is performed on the found optimal approximation of the true surface $\Gamma$.
- This procedure leads to the theory of *multivariate interpolation* in high-dimensional space.
- In the *strict* sense the interpolation problem can be stated:
- *Given a set of $N$ different $m_0$-dimensional points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ and a corresponding set of $N$ real numbers $d_1, d_2, \ldots, d_N$.*
- *Find a function $F: \mathcal{R}^{m_0} \rightarrow \mathcal{R}^1$ satisfying the interpolation condition:*

$$F(\mathbf{x}_i) = d_i, \quad i = 1, 2, \ldots, N$$

- Hence in strict interpolation no error is permitted.

- That is, the interpolating surface $F$ must pass through all the training data points.

- The *radial-basis functions* (RBF) technique uses a function $F$ with the following form:

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

- Here $\varphi(\| \mathbf{x} - \mathbf{x}_i \|)$, $i = 1, \ldots, N$, is a set of arbitrary nonlinear *radial-basis functions*.

- Usually $\| . \|$ is the Euclidean norm.

- The known data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ are taken as the *centers* of the radial-basis functions.

- The interpolation conditions yield for solving the coefficients (weights) $w_1, w_2, \ldots, w_N$ of RBF's linear equations

$$\begin{bmatrix} \varphi_{11} & \varphi_{12} & \cdots & \varphi_{1N} \\ \varphi_{21} & \varphi_{22} & \cdots & \varphi_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ \varphi_{N1} & \varphi_{N2} & \cdots & \varphi_{NN} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix}$$

- These equations can be written conveniently in the matrix-vector form

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

- Here

$$\mathbf{w} = [w_1, w_2, \ldots, w_N]^T$$

  is the *linear weight vector*, and

$$\mathbf{d} = [d_1, d_2, \ldots, d_N]^T$$

  is the *desired response vector*.

- $N$ is the *size of the training sample.*

- The elements $\varphi_{ji}$ of the $N \times N$ *interpolation matrix* $\mathbf{\Phi}$ are defined by

$$\varphi_{ji} = \varphi(\| \mathbf{x}_j - \mathbf{x}_i \|)$$

- Assuming that $\mathbf{\Phi}$ is nonsingular, the weight vector can be solved easily:

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

- Errors in Haykin's formulas (5.15) and (5.16): $\mathbf{x}$ must be replaced by $\mathbf{d}$: $\mathbf{\Phi w} = \mathbf{d}$, $\mathbf{w} = \mathbf{\Phi}^{-1}\mathbf{d}$

## Michelli's Theorem

- The interpolating matrix $\mathbf{\Phi}$ is nonsingular for a large class of radial-basis functions.
- Michelli's theorem (1986):
- *Let* $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ *be a set of distinct points in* $\mathcal{R}^{m_0}$*.*
- *Then the* $N \times N$ *interpolating matrix* $\mathbf{\Phi}$*, whose* $ji$*-th element is* $\varphi(\| \mathbf{x}_j - \mathbf{x}_i \|)$*, is nonsingular.*

- Michelli's theorem covers the following functions used often in RBF networks:

  1. *Multiquadrics:*

  $$\varphi(r) = (r^2 + c^2)^{1/2} \text{ for some } c > 0 \text{ and } r \in \mathcal{R}$$

  2. *Inverse multiquadrics:*

  $$\varphi(r) = \frac{1}{(r^2 + c^2)^{1/2}} \text{ for some } c > 0 \text{ and } r \in \mathcal{R}$$

  3. *Gaussian functions:*

  $$\varphi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right) \text{ for some } \sigma > 0 \text{ and } r \in \mathcal{R}$$

- The inverse multiquadrics and Gaussian function are *localized* functions: $\varphi(r) \rightarrow 0$ as $r \rightarrow \infty$.

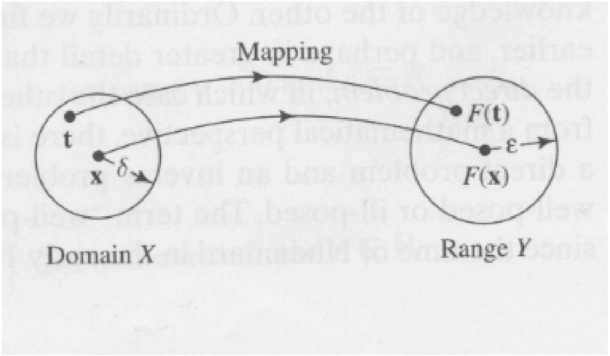
- For both functions, the interpolation matrix $\mathbf{\Phi}$ is positive definite.

- In contrast, the multiquadrics are *nonlocal* since $\varphi(r)$ becomes unbounded as $r \to \infty$.
- The interpolating matrix is not positive definite but anyway nonsingular.
- Radial-basis functions that *grow* at infinity approximate better a smooth input-output mapping.
  - Compared with those using a positive definite interpolation matrix.
  - A remarkable result.

## 5.4 Supervised Learning as an Ill-Posed Hypersurface Reconstruction Problem

- The strict interpolation strategy described in Section 5.3 is not always a good strategy for training RBF networks.
- Reason: we must use as many radial-basis functions as data points.
- If there are many data points, this leads easily to overfitting.
- Then the RBF network models also the noise in the data.
- Consider now the overfitting problem and how to cure it.
- Recall the basic philosophy behind the RBF networks:
- *Learning is viewed as a problem of hypersurface reconstruction, given a set of data points that may be sparse.*
- There often exist two related problems called *direct problem* and *inverse problem*.

- An example: find a mapping $y = f(x)$ and the inverse mapping $x = f^{-1}(y)$.
- A problem of interest may be well-posed or ill-posed.
- Consider specifically the problem of reconstructing a fixed but unknown mapping $f$ between two metric spaces.



- This problem is *well-posed* if the following three conditions are satisfied:

  1. **Existence.** For every input vector $\mathbf{x}$, there does exist an output $y = f(\mathbf{x})$.
  2. **Uniqueness.** $f(\mathbf{x}) = f(\mathbf{t})$ if and only if $\mathbf{x} = \mathbf{t}$.
  3. **Continuity (stability).** The mapping is continuous: a small change in $\mathbf{x}$ leads to a finite change in $y$.

- If any of these conditions is not satisfied, the problem is said to be *ill-posed*.

- In our case, the physical phenomenon responsible for generating the training data is a well-posed direct problem.

- However, the hypersurface reconstruction problem for the training data is an ill-posed inverse problem.

- It can happen that all the three conditions required for a well-posed problem are violated.

- In particular, if the learning problem lacks the property of continuity, the computed input-output mapping may have nothing to do with the true solution to the learning problem.

- This may happen if there is too much noise in the input vectors.

- The only solution to this is to have more information about the input-output mapping.

- The lack of information cannot be remedied by any mathematical trickery (Lanczos, 1964).

- An ill-posed problem can be made into a well-posed one by using a suitable regularization technique.