Radial Basis Function (RBF) Networks

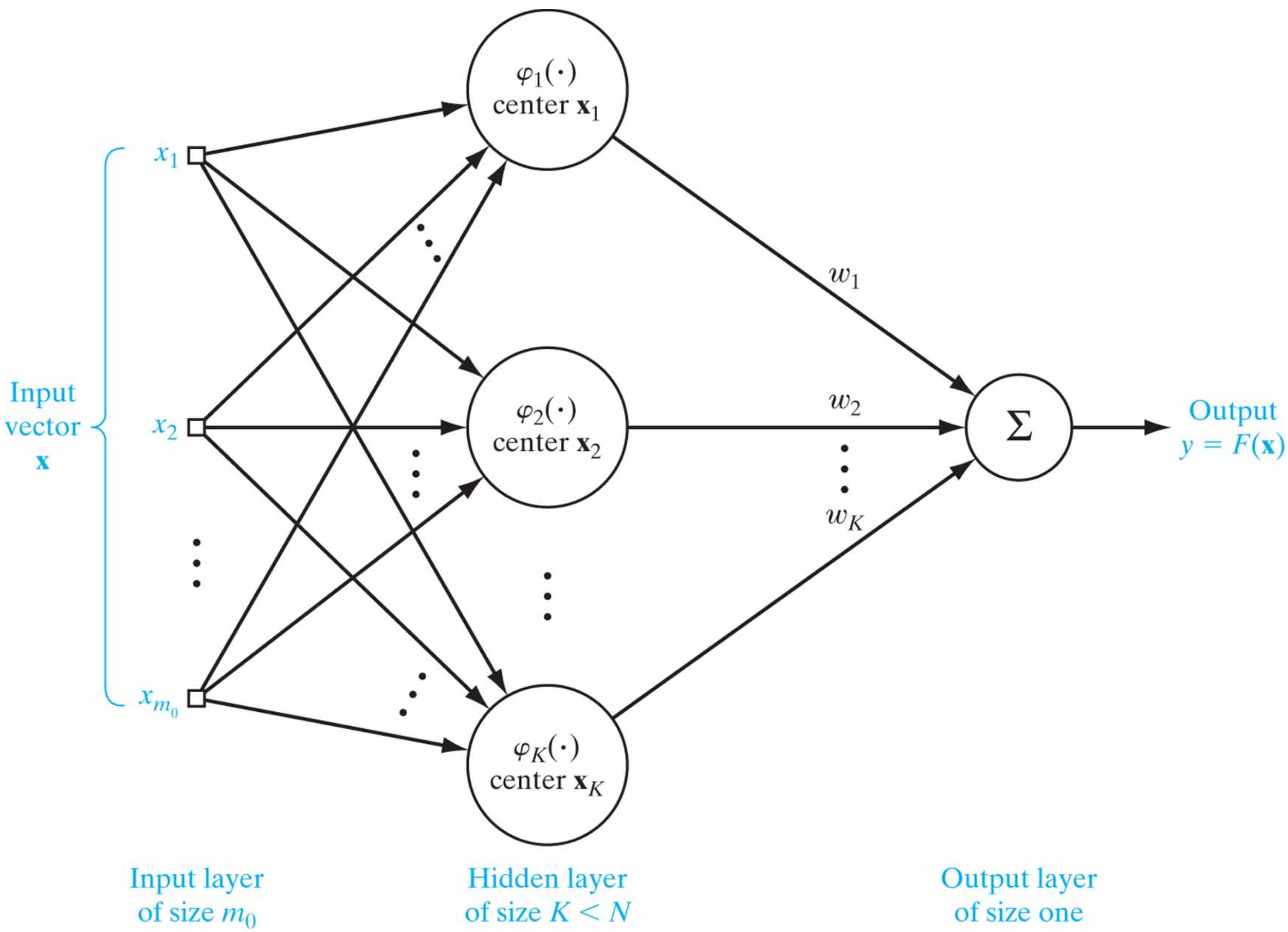
Function approximation

- We have been using MLPs as pattern classifiers
- But in general, they are function approximators
  - Depending on output layer nonlinearity
  - And error function being minimized
- As a function approximator, MLPs are nonlinear, semiparametric, and universal

Function approximation

- Radial basis function (RBF) networks are similar function approximators
- Also nonlinear, semiparametric, universal
- Can also be visualized as layered network of nodes
- Easier to train than MLPs
  - Do not require backpropagation
  - But do not necessarily find an optimal solution

RBF net illustration



Function approximation background

- Before getting into RBF networks, let's discuss approximating scalar functions of a single variable
- Weierstrass approximation theorem: any continuous real function in an interval can be approximated arbitrarily well by a set of polynomials
- Taylor expansion approximates any differentiable function by a polynomial in the neighborhood around a point
- Fourier series gives a way of approximating any periodic function by a sum of sines and cosines

Linear projection

- Approximate function f (x) by a linear combination of simpler functions

$$F(\mathbf{x}) = \sum_j w_j \varphi_j(\mathbf{x})$$

- If w_j's can be chosen so that approximation error is arbitrarily small for any function f (x) over the domain of interest, then $\{\varphi_j\}$ has the property of universal approximation, or $\{\varphi_j\}$ is complete

Example incomplete basis: sinc

$$\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x} \qquad \varphi_j(x) = \text{sinc}(x - \mu_j)$$

- Can approximate any smooth function

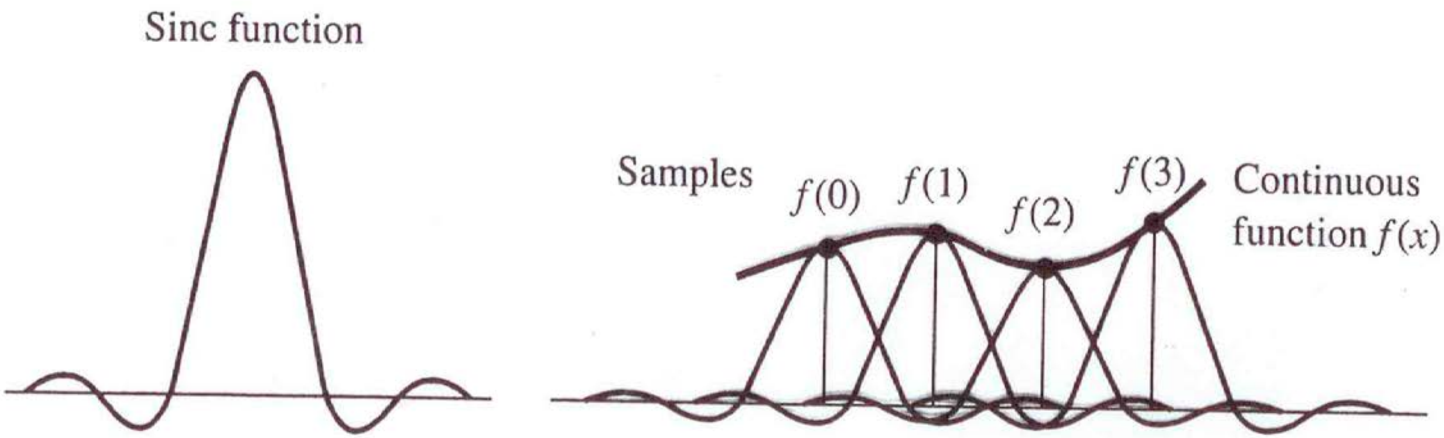


FIGURE 5-4 Decomposition by sinc functions

Example orthogonal complete basis: sinusoids

$$\varphi_{2n}(x) = \sin(2\pi n\omega x)$$
$$\varphi_{2n+1}(x) = \cos(2\pi n\omega x)$$

Complete on the interval [0,1]

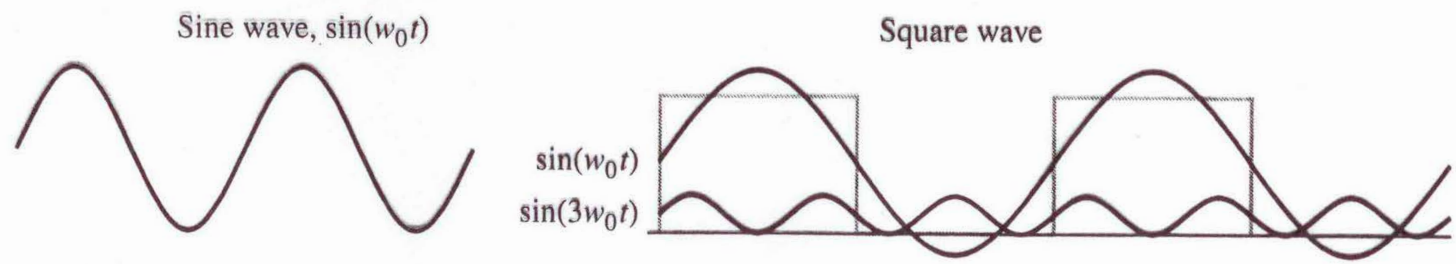


FIGURE 5-5 Decomposition by sine waves

Example orthogonal complete basis: Chebyshev polynomials

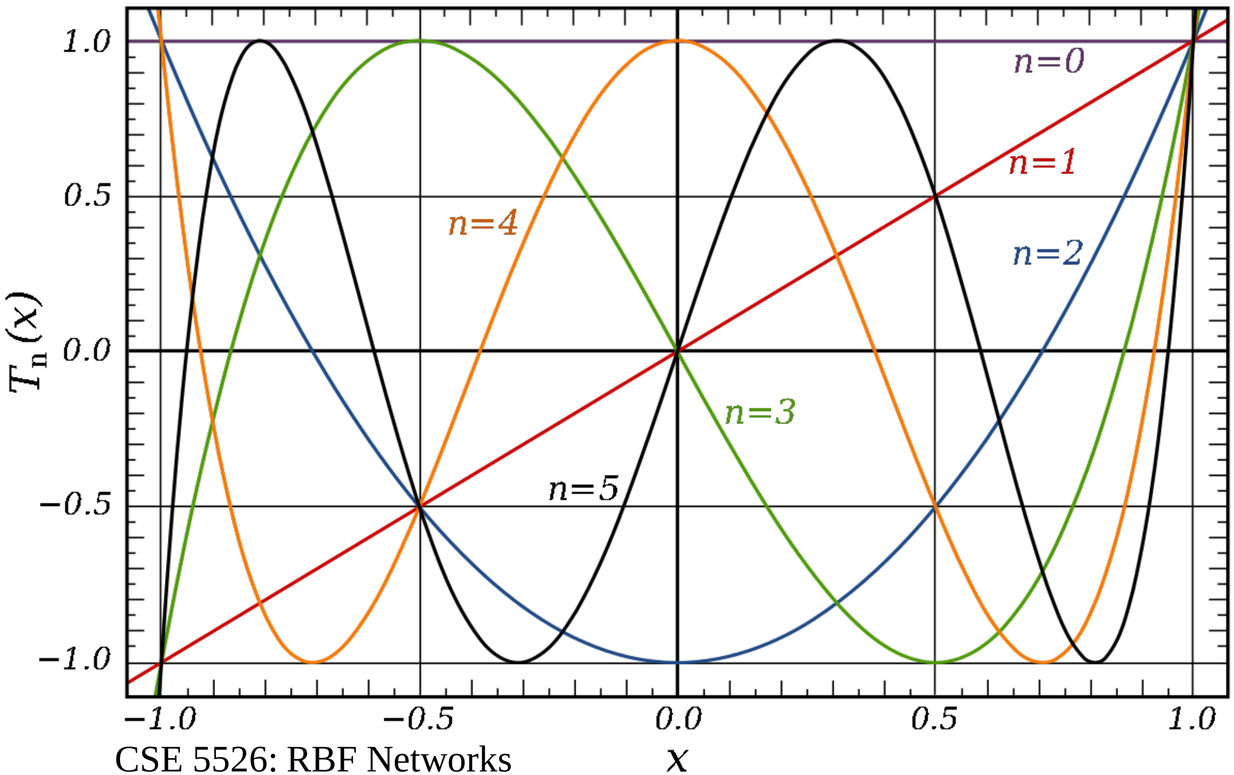
$$T_0(x) = 1 \quad T_1(x) = x \quad T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

Complete on the interval [0,1]

$$T_2(x) = 2x^2 - 1$$

$$T_3(x) = 4x^3 - 3x$$

etc.



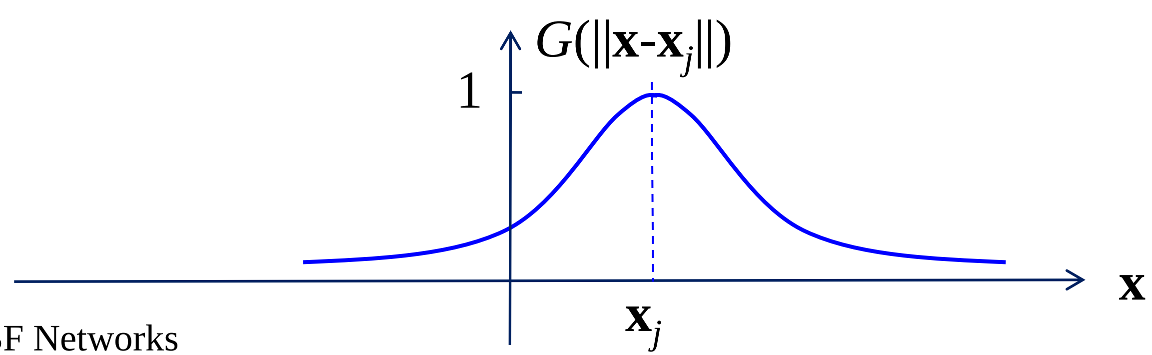
"Chebyshev Polynomials of the 1st Kind (n=0-5, x=(-1,1))" by Inductiveload - Own work. Licensed under Public domain via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:Chebyshev_Polynomials_of_the_1st_Kind_(n%3D0-5,_x%3D(-1,1)).svg

Radial basis functions

- A radial basis function (RBF) is a basis function of the form $\varphi_j(\boldsymbol{x}) = \varphi\left(\left\|\boldsymbol{x} - \boldsymbol{\mu}_j\right\|\right)$
  - Where $\varphi(r)$ is positive w/monotonic derivative for $r > 0$
- Consider a Gaussian RBF

$$\varphi_j(\mathbf{x}) = \exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{x} - \ \mathbf{x}_j\right\|^2\right) = G\left(\left\|\boldsymbol{x} - \boldsymbol{x}_j\right\|\right)$$

- A *local* basis function, falling off from the center



Radial basis functions (cont.)

- Thus approximation by Gaussian RBF becomes

$$F(\mathbf{x}) = \sum_j w_j G(||\mathbf{x} - \mathbf{x}_j||)$$

- Gaussians are universal approximators
  - I.e., they form a complete basis

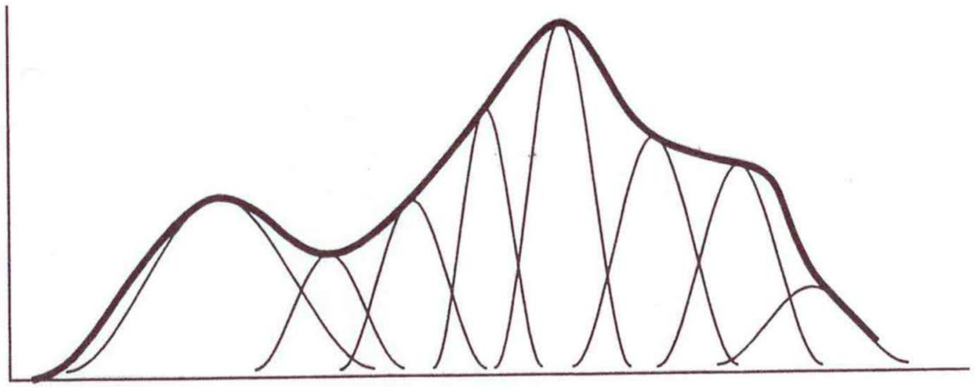
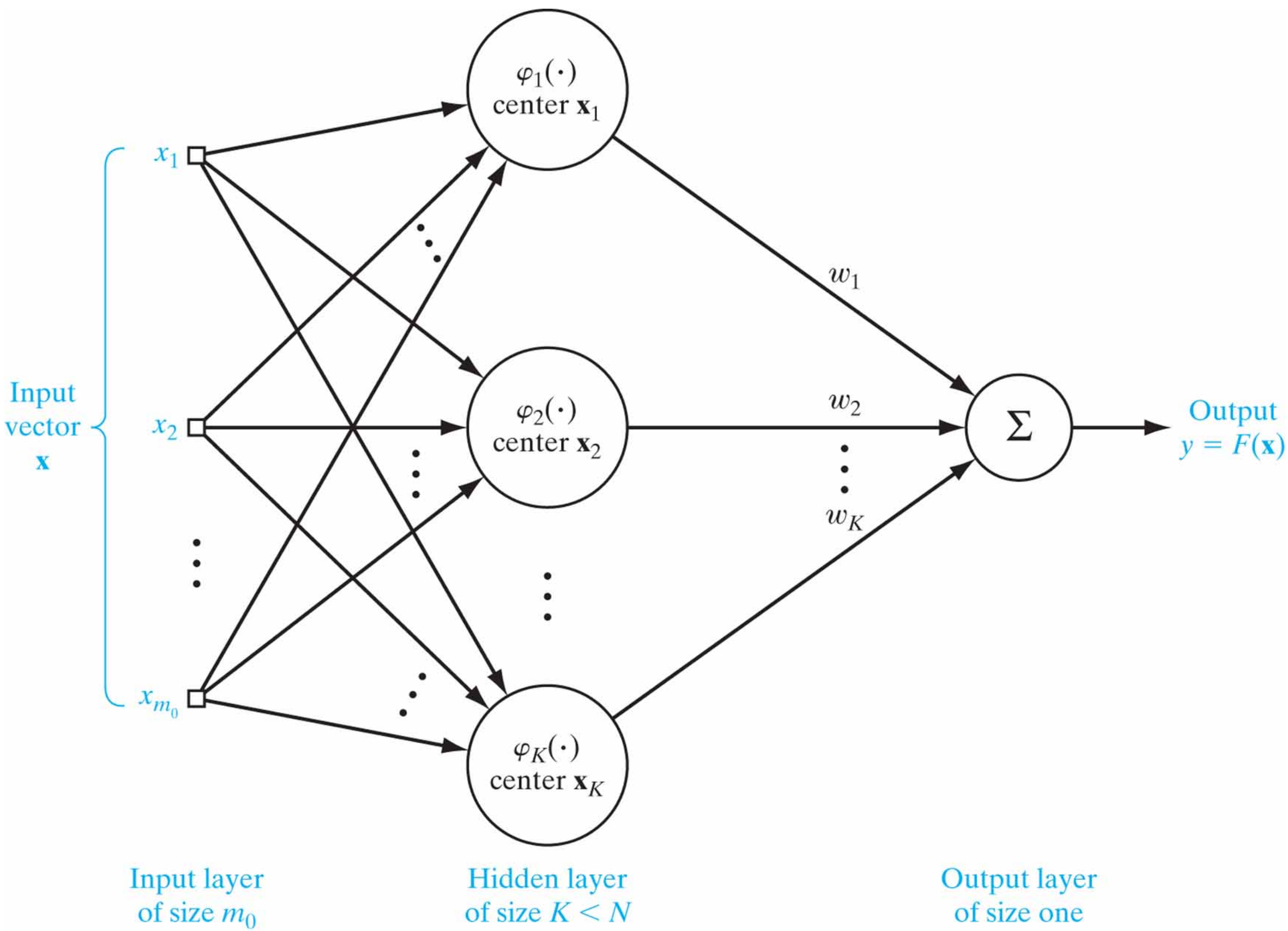


FIGURE 5-8 Approximation by RBFs in one dimension

RBF net illustration



Remarks (cont.)

- Other RBFs exist, but we won't be using them
- Multiquadrics

$$\varphi(x) = \sqrt{x^2 + c^2}$$

- Inverse multiquadrics

$$\varphi(x) = \frac{1}{\sqrt{x^2 + c^2}}$$

- Micchelli's theorem (1986)
  - Let $\{\boldsymbol{x}_i\}$ be a set of N distinct points, $\varphi(\cdot)$ be an RBF
  - Then the matrix $\phi_{ij} = \varphi\left(\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|\right)$ is non-singular

Four questions to answer for RBF nets

- If we want to use Gaussian RBFs to approximate a function specified by training data
  1. How do we choose the Gaussian centers?
  2. How do we determine the Gaussian widths?
  3. How do we determine the weights w_j?
  4. How do we select the number of bases?

1. How do we choose the Gaussian centers?

- Easy way: select K data points at random
- Potentially better way: unsupervised clustering, e.g. using the K-means algorithm

K-means algorithm

- Goal: Divide N input patterns into K clusters with minimum total variance
- In other words, partition patterns into K clusters C_j to minimize the following cost function

$$J = \sum_{j=1}^{K} \sum_{i \in C_j} ||\mathbf{x}_i - \mathbf{u}_j||^2$$

where $\mathbf{u}_j = \frac{1}{||C_j||} \sum_{i \in C_j} \mathbf{x}_i$ is the mean (center) of cluster j

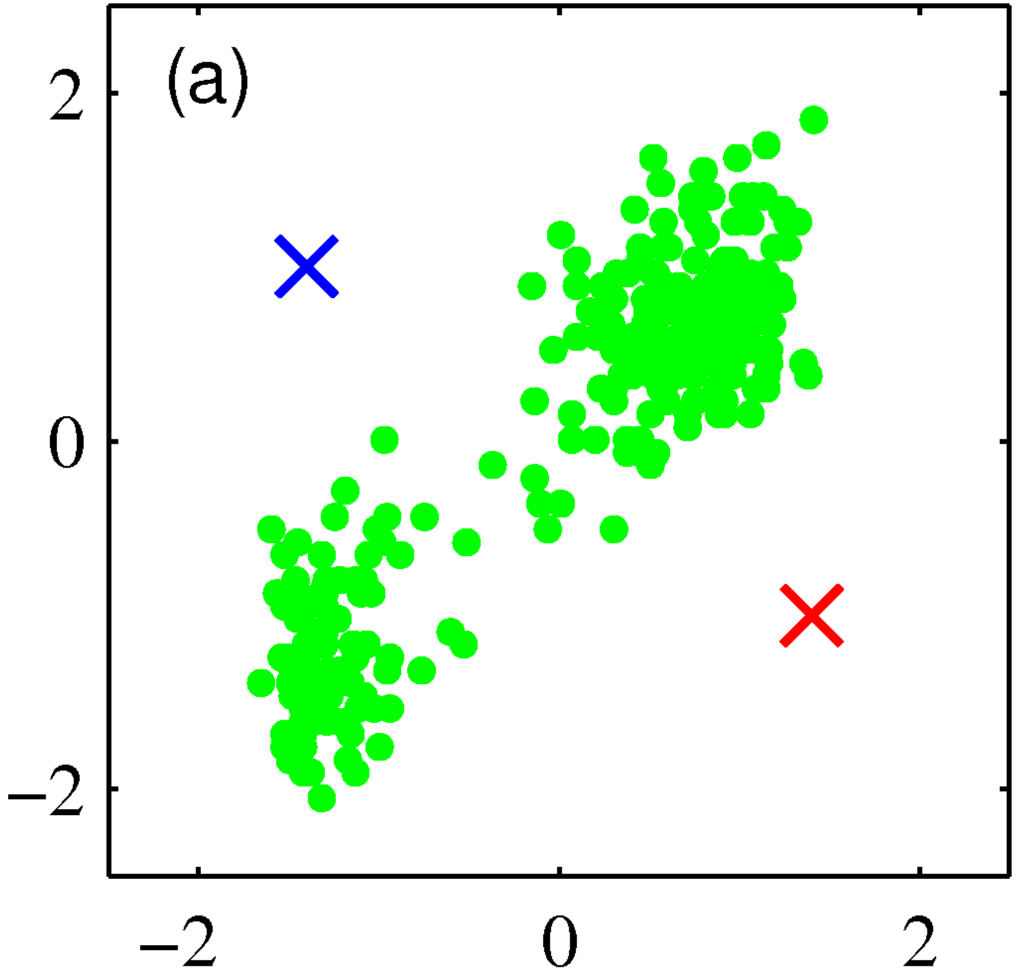
K-means algorithm

1. Choose a set of K cluster centers randomly from the input patterns
2. Assign the N input patterns to the K clusters using the squared Euclidean distance rule:
   $\mathbf{x}$ is assigned to C_j if $||\mathbf{x}-\mathbf{u}_j||^2 \leq ||\mathbf{x}-\mathbf{u}_i||^2$ for all $i \neq j$
3. Update cluster centers

$$\mathbf{u}_j = \frac{1}{|C_j|}\sum_{i \in C_j} \mathbf{x}_i$$

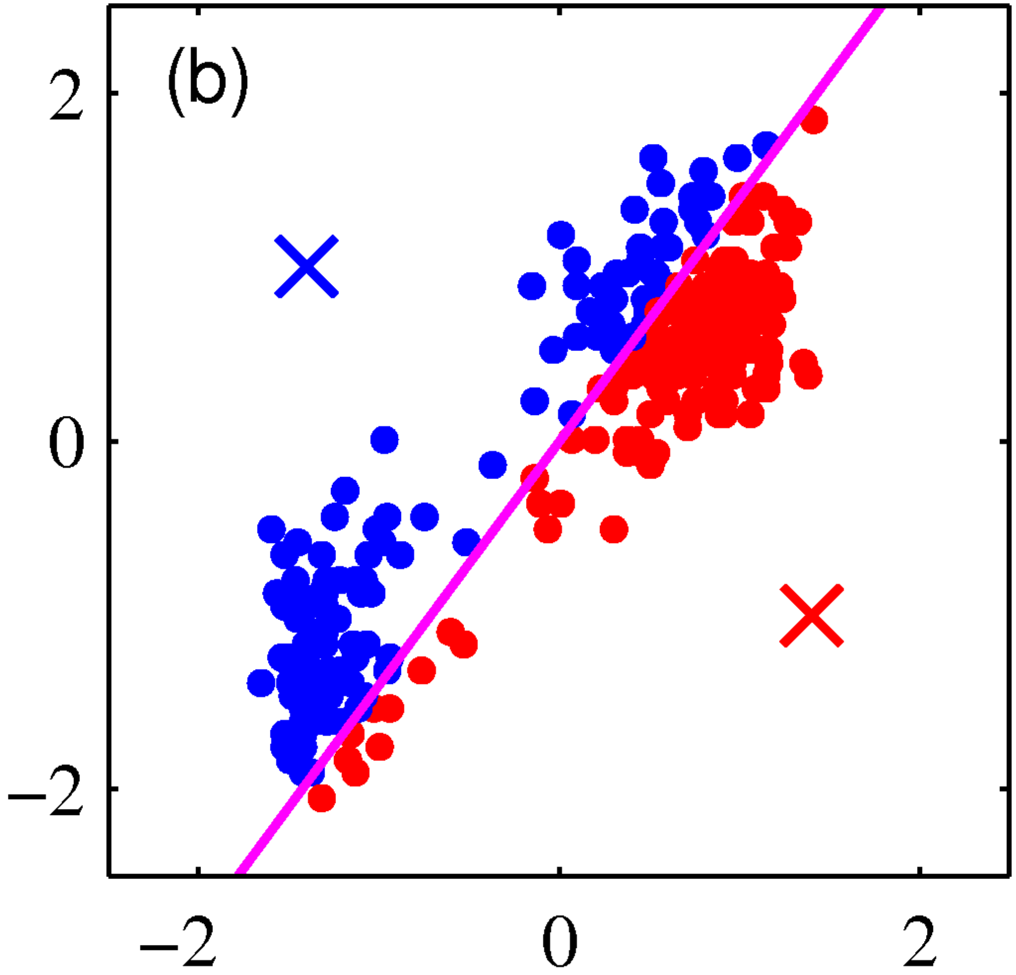
4. If any cluster center changes, go to step 2; else stop

K-means illustration



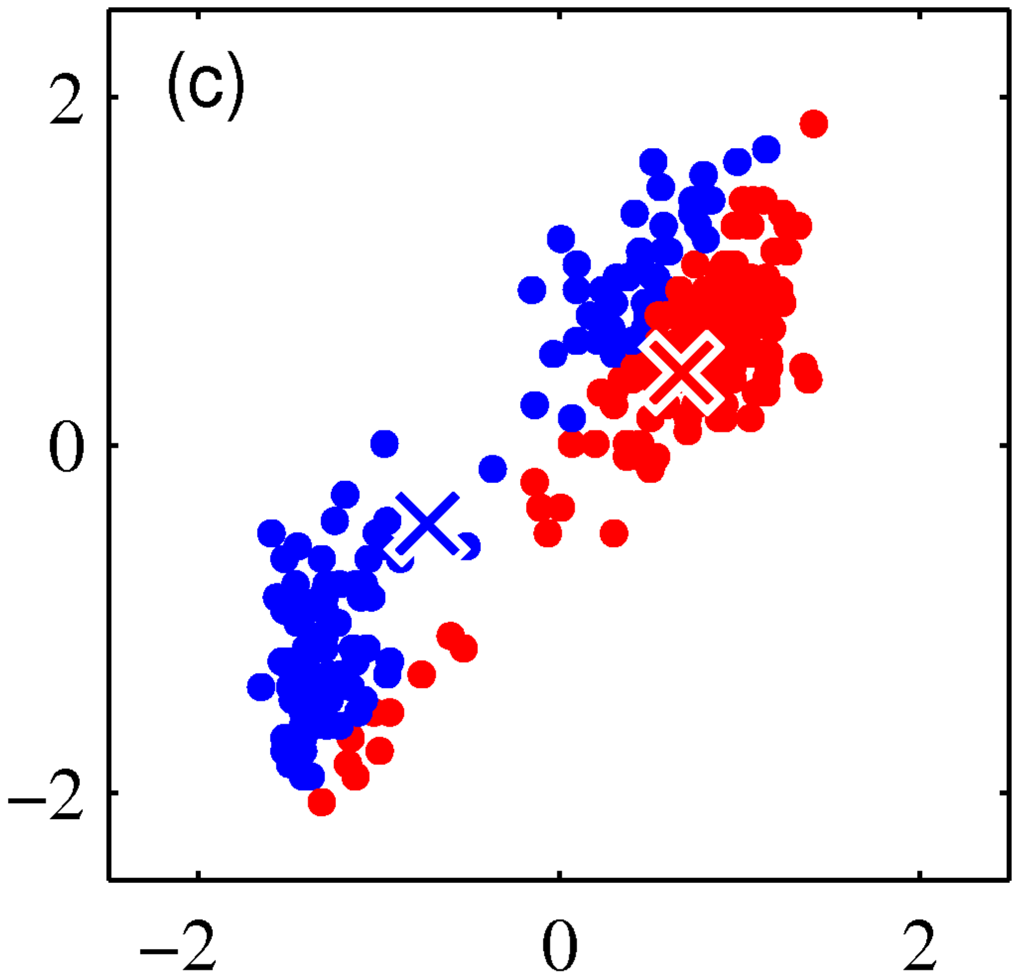
From Bishop (2006)

K-means illustration



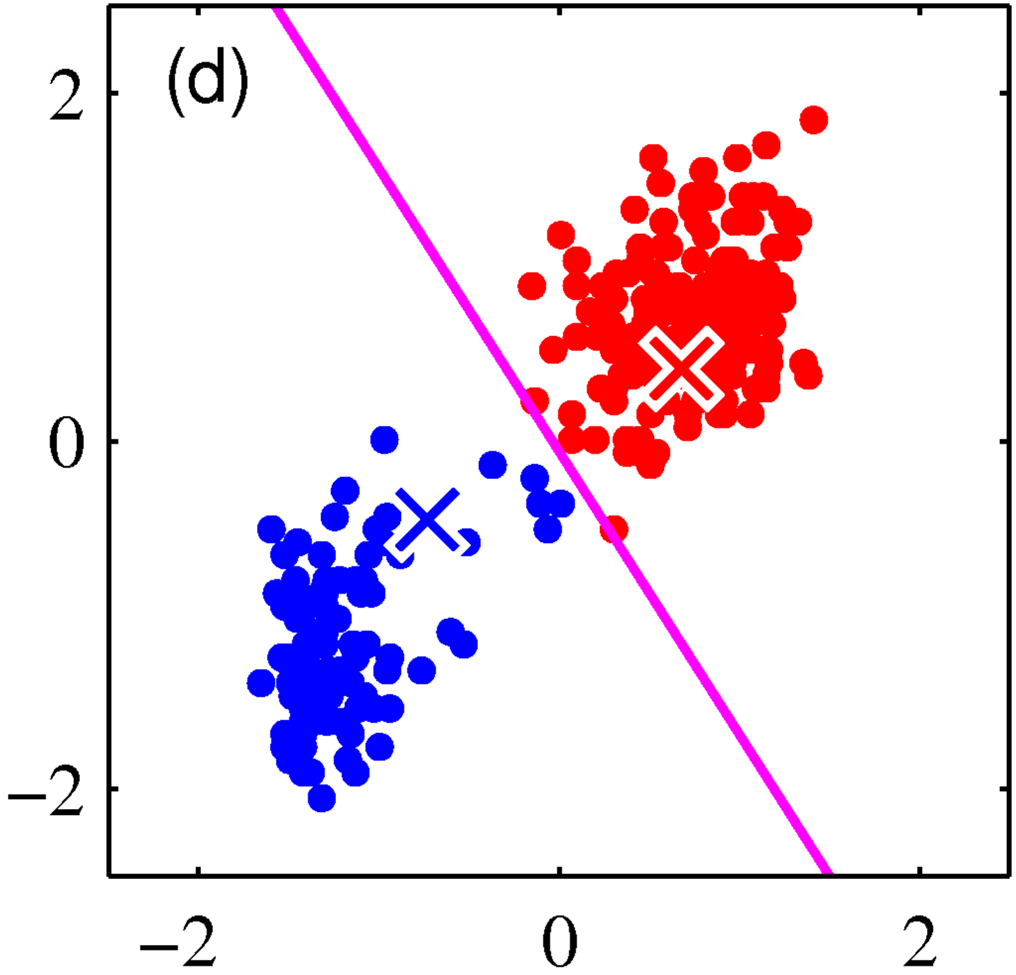
From Bishop (2006)

K-means illustration



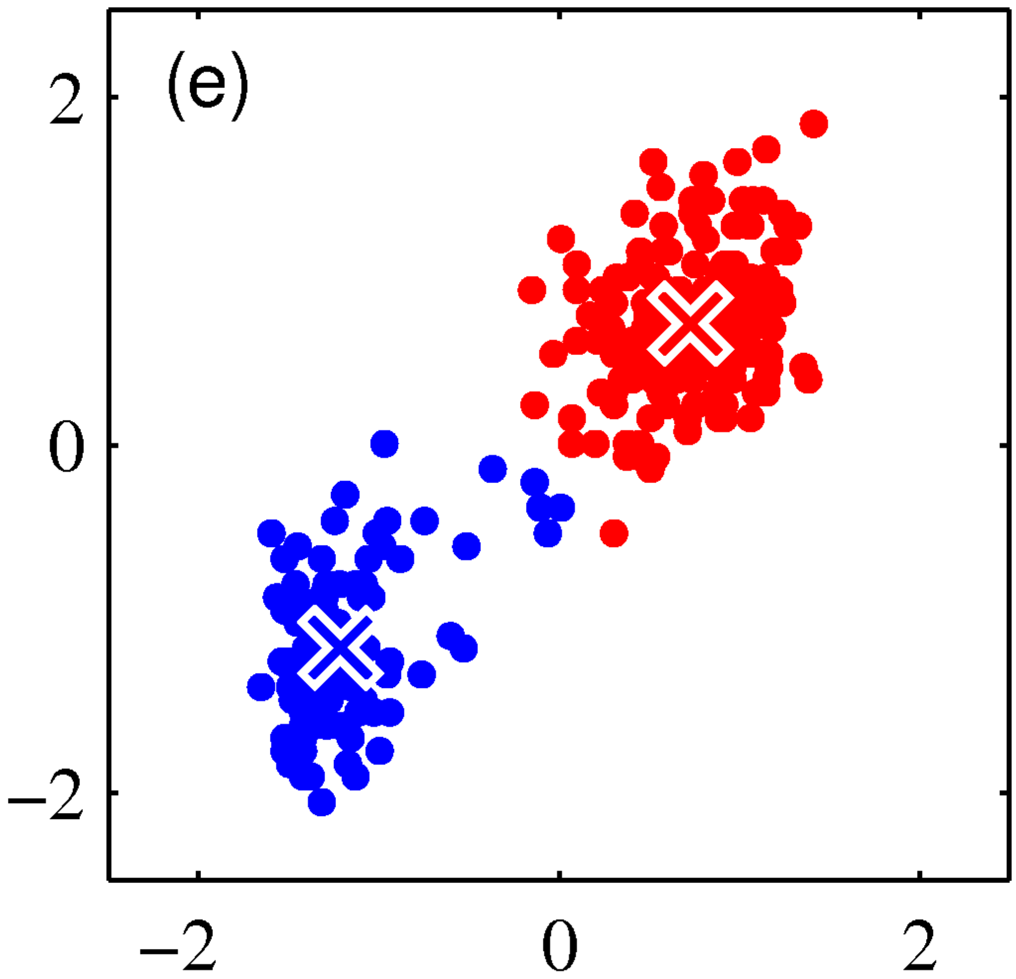
From Bishop (2006)

K-means illustration



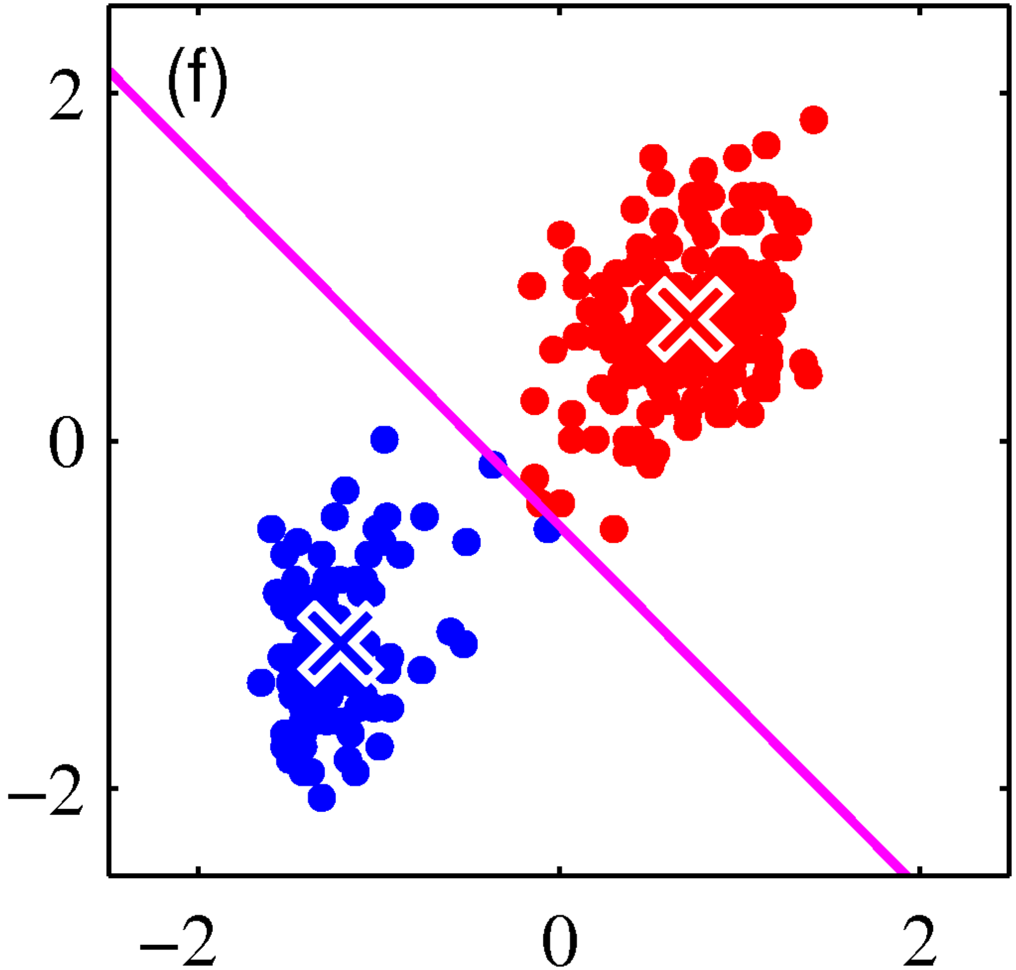
From Bishop (2006)

K-means illustration



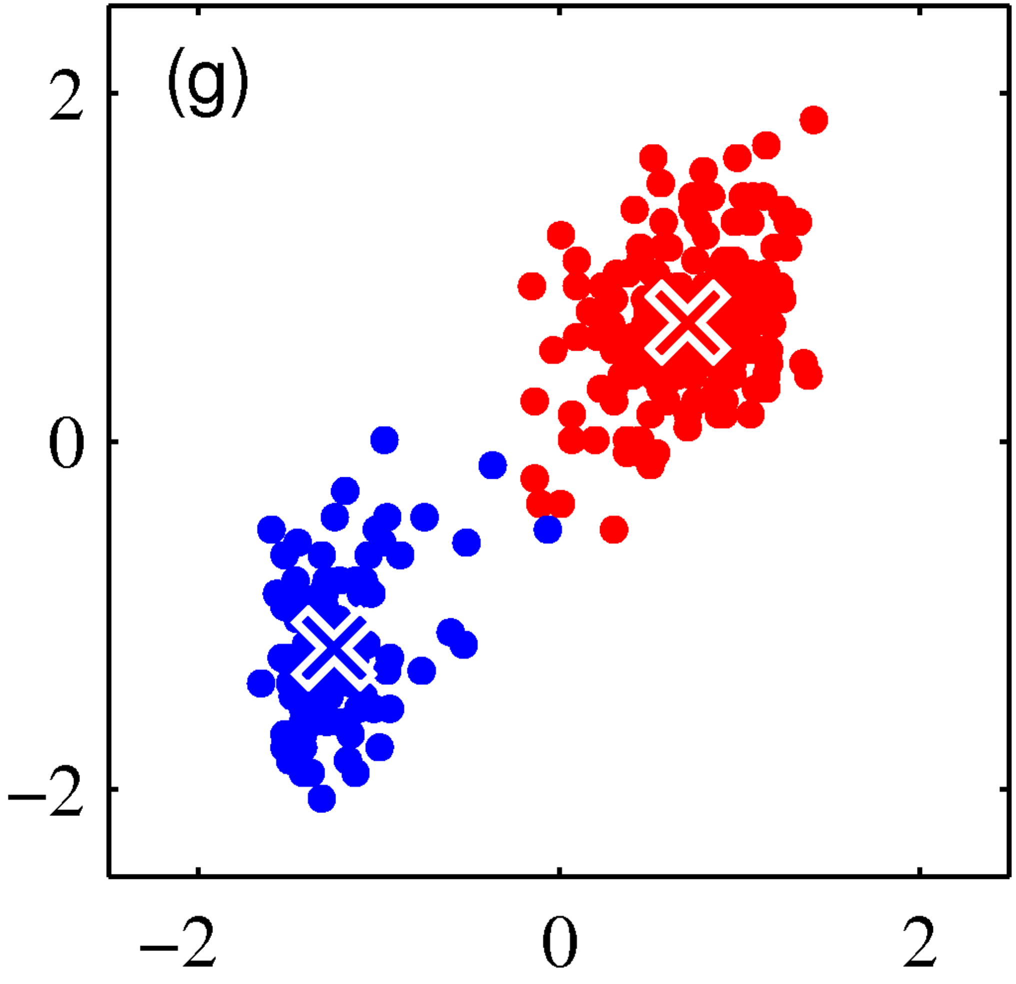
From Bishop (2006)

K-means illustration



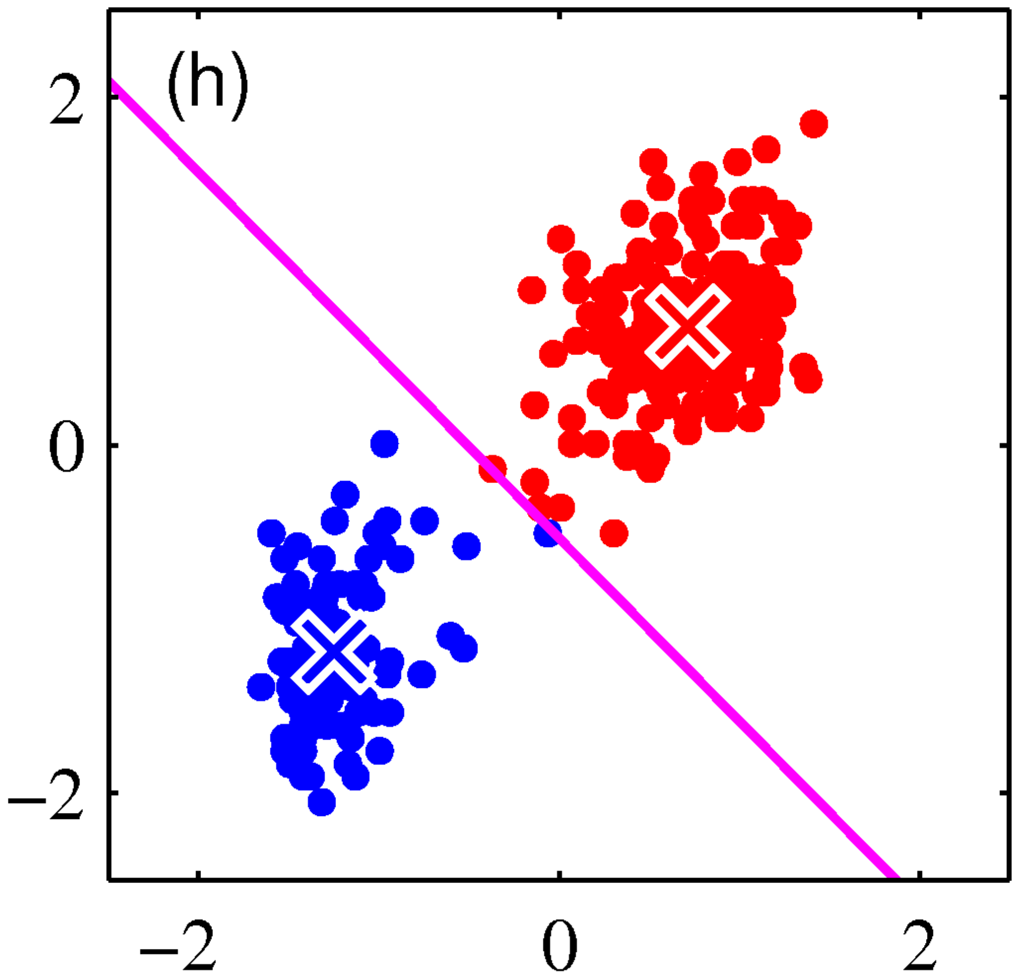
From Bishop (2006)

K-means illustration



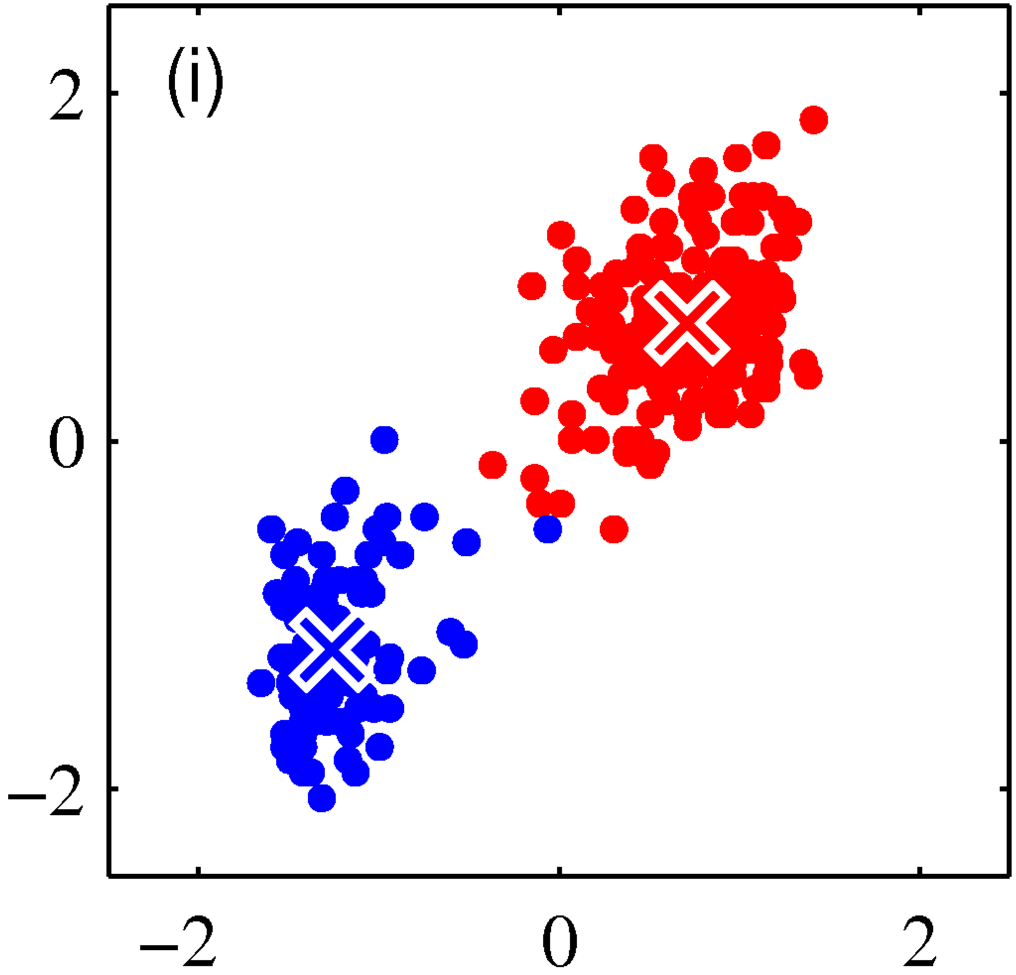
From Bishop (2006)

K-means illustration



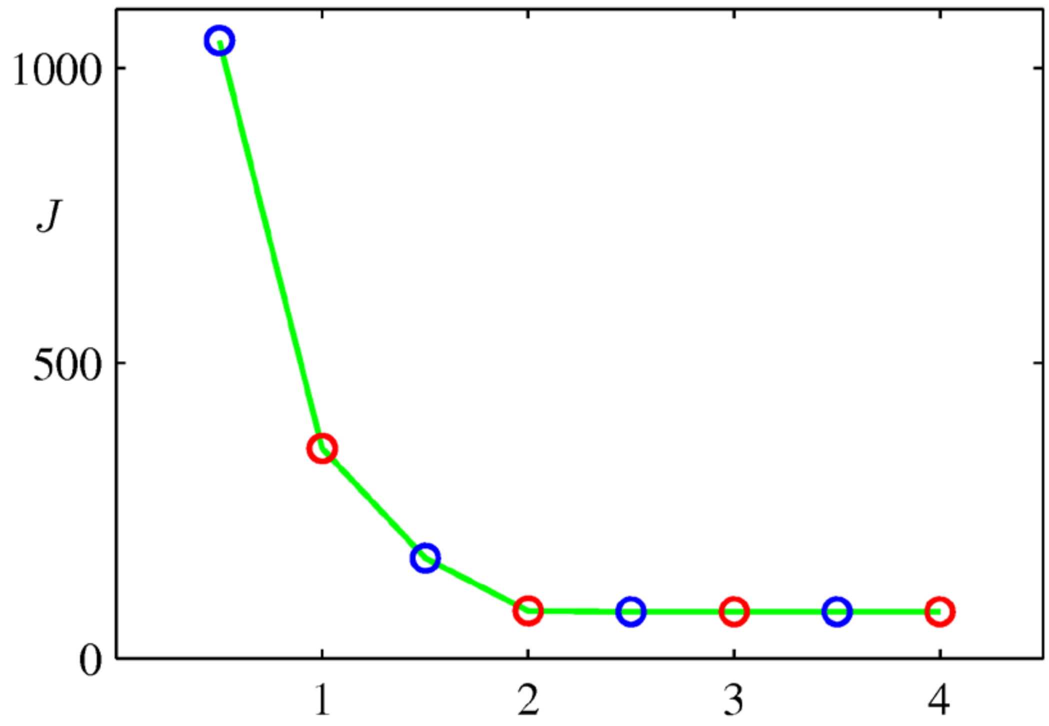
From Bishop (2006)

K-means illustration



From Bishop (2006)

K-means cost function



From Bishop (2006)

K-means algorithm remarks

- The *K*-means algorithm always converges, but only to a local minimum

2. How to determine the Gaussian widths?

- Once cluster centers are determined, the variance within each cluster can be set to

$$\sigma_j^2 = \frac{1}{|C_j|} \sum_{i \in C_j} ||\mathbf{u}_j - \mathbf{x}_i||^2$$

  - **Remark**: to simplify the RBF net design, all clusters can assume the same Gaussian width:

$$\sigma = \frac{d_{\max}}{\sqrt{2K}}$$

where $d_{\max}$ is the maximum distance between the K cluster centers

3. How do we determine the weights w_j?

- With the hidden layer decided, weight training can be treated as a linear regression problem
$$\boldsymbol{\Phi}\boldsymbol{w} = \boldsymbol{d}$$
- Can solve using the LMS algorithm
- The textbook discusses recursive least squares (RLS) solutions
- Can also solve in one shot using the pseudo-inverse
$$\boldsymbol{w} = \boldsymbol{\Phi}^{+}\boldsymbol{d} = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{T}\boldsymbol{d}$$
- Note that a bias term needs to be included in $\boldsymbol{\Phi}$

4. How do we select the number of bases?

- The same problem as that of selecting the size of an MLP for classification
- The short answer: (cross-)validation
- The long answer: by balancing bias and variance

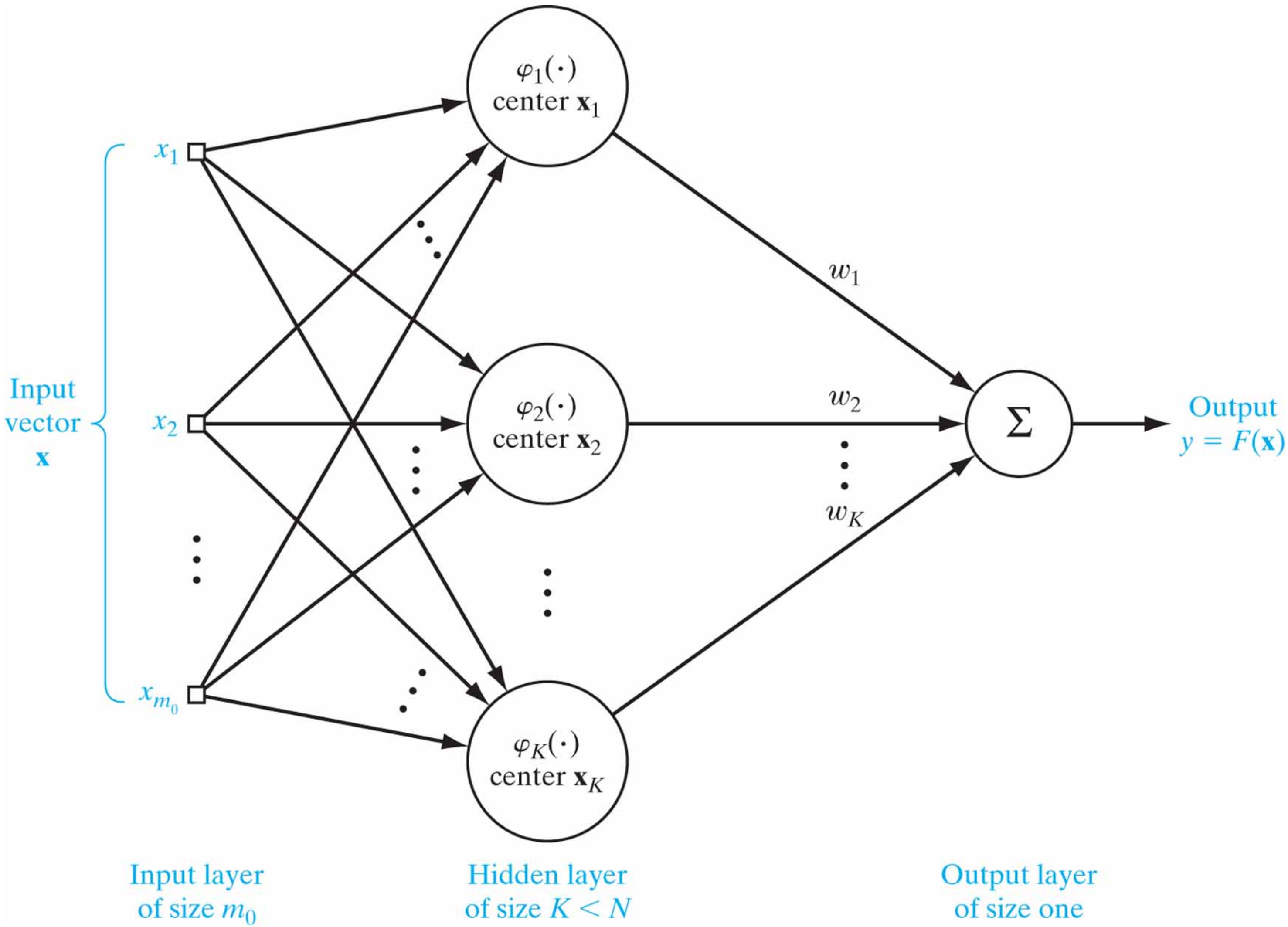
Bias and variance

- Bias: training error
  - Difference between desired output and actual output for a particular training sample
- Variance: generalization error
  - difference between the learned function from a particular training sample and the function derived from all training samples
- Two extreme cases: zero bias and zero variance
- A good-sized model is one where both bias and variance are low

RBF net training summary

- To train
  1. Choose the Gaussian centers using K-means, etc.
  2. Determine the Gaussian widths as the variance of each cluster, or using $d_{\max}$
  3. Determine the weights w_j using linear regression
- Select the number of bases using (cross-)validation

RBF net illustration

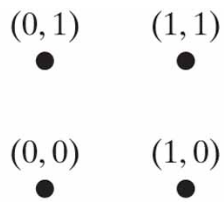


Comparison between RBF net and MLP

- For RBF nets, bases are local, while for MLP, “bases” are global
- Generally, more bases are needed for an RBF net than hidden units for an MLP
- Training is more efficient for RBF nets

XOR problem, again

- RBF nets can also be applied to pattern classification problems
  - XOR problem revisited



(a)

Let

$$\varphi_1(x) = \exp(-\|x - t_1\|^2)$$
$$\varphi_2(x) = \exp(-\|x - t_2\|^2)$$

Where

$$t_1 = [1,1]^T$$
$$t_2 = [0,0]^T$$

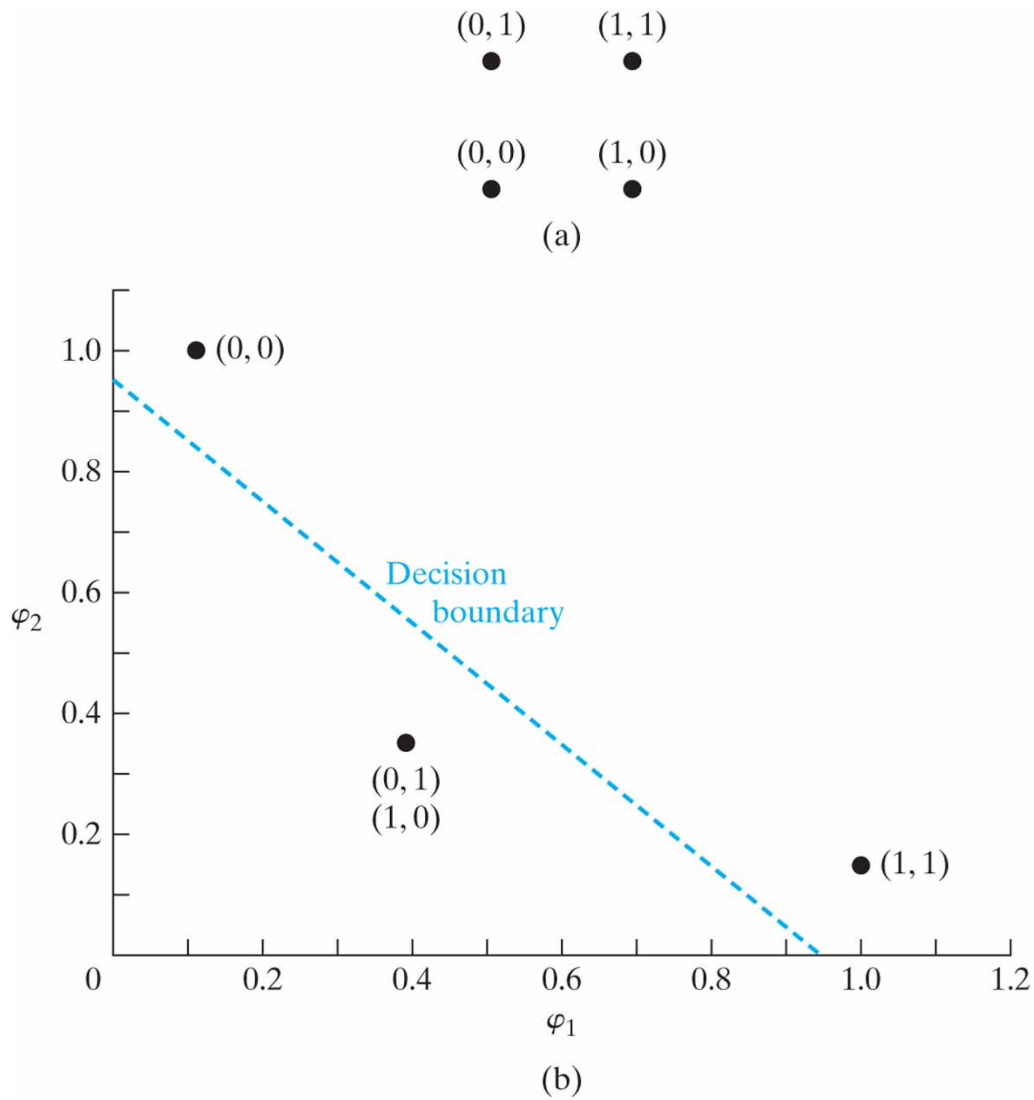
XOR problem (cont.)

TABLE 5.1 Specification of the Hidden Functions for the XOR Problem of Example 1

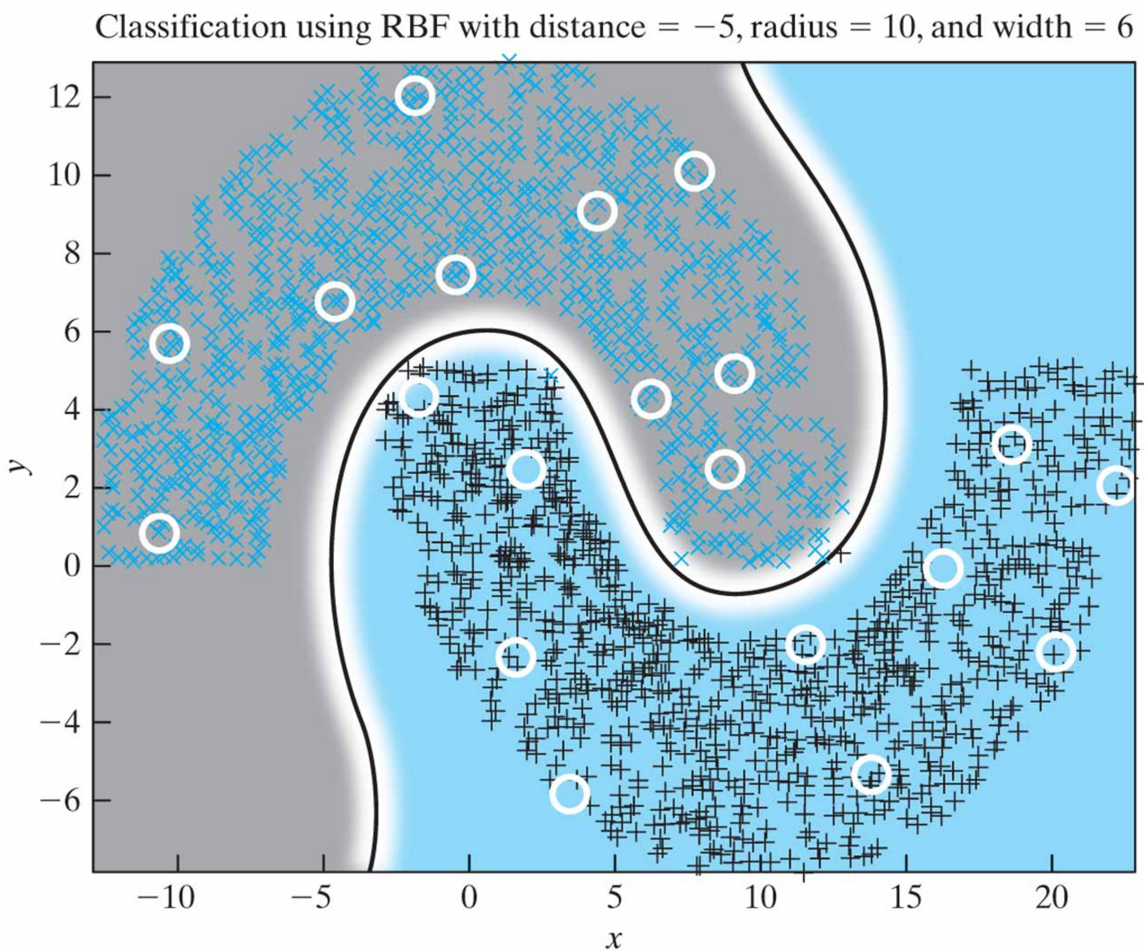
Input Pattern $\mathbf{x}$	First Hidden Function $\varphi_1(\mathbf{x})$	Second Hidden Function $\varphi_2(\mathbf{x})$
(1,1)	1	0.1353
(0,1)	0.3678	0.3678
(0,0)	0.1353	1
(1,0)	0.3678	0.3678

XOR problem, again

- RBF nets can also be applied to pattern classification problems
  - XOR problem revisited

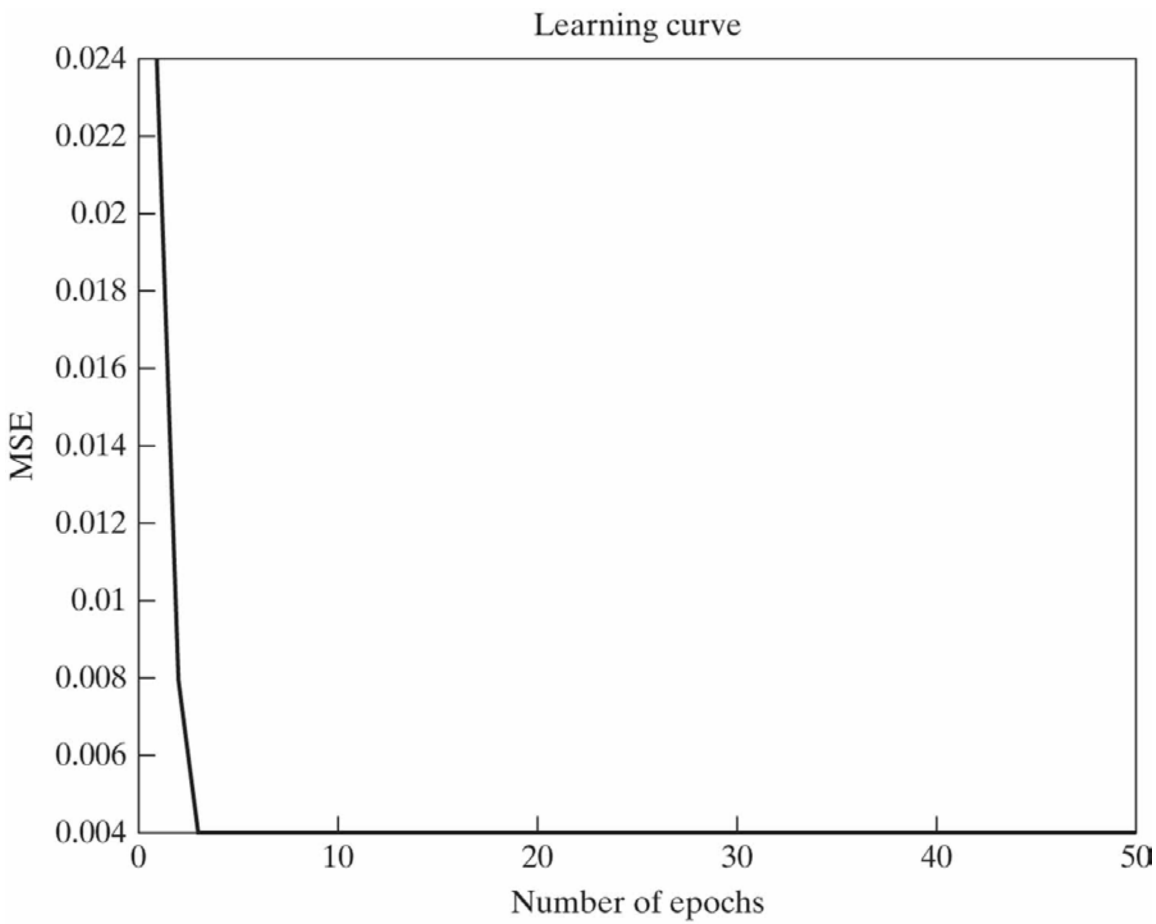


RBF net on double moon data, d = -5



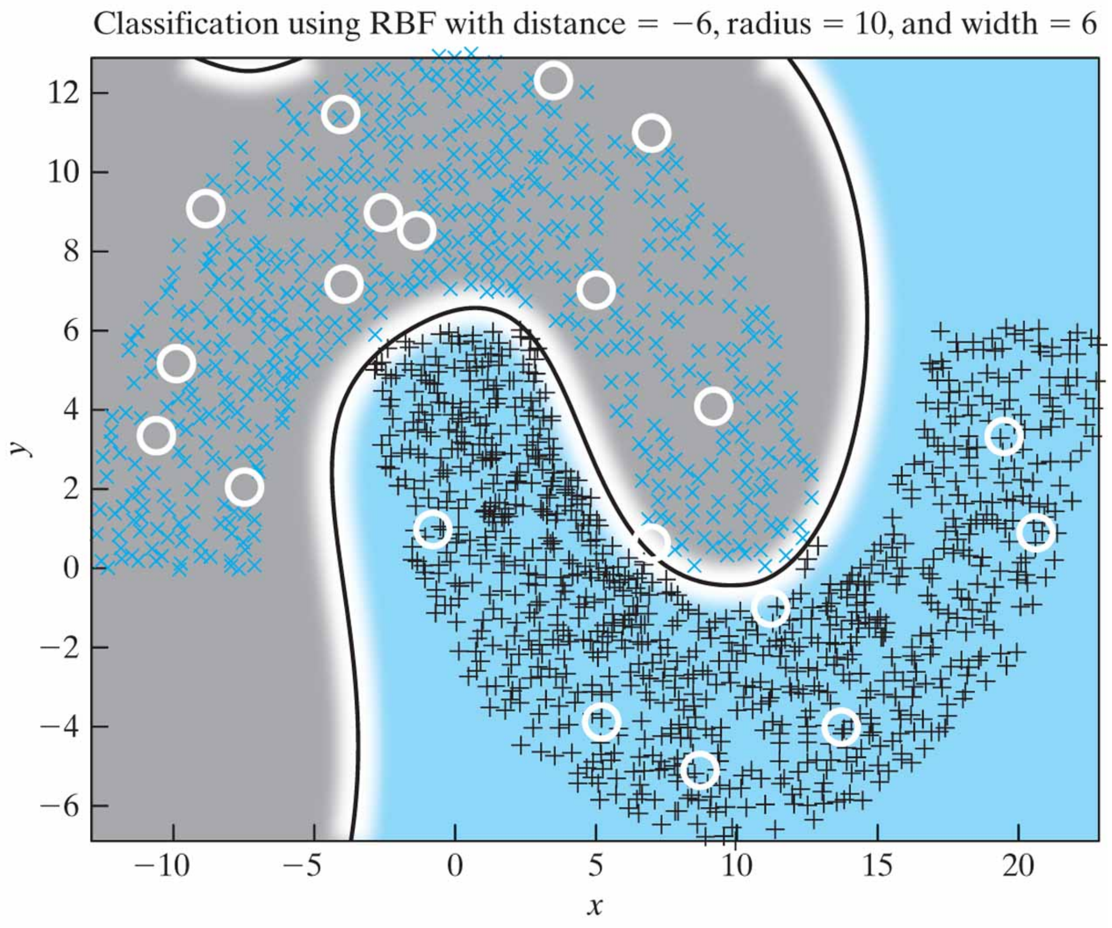
(b) Testing result

RBF net on double moon data, d = -5



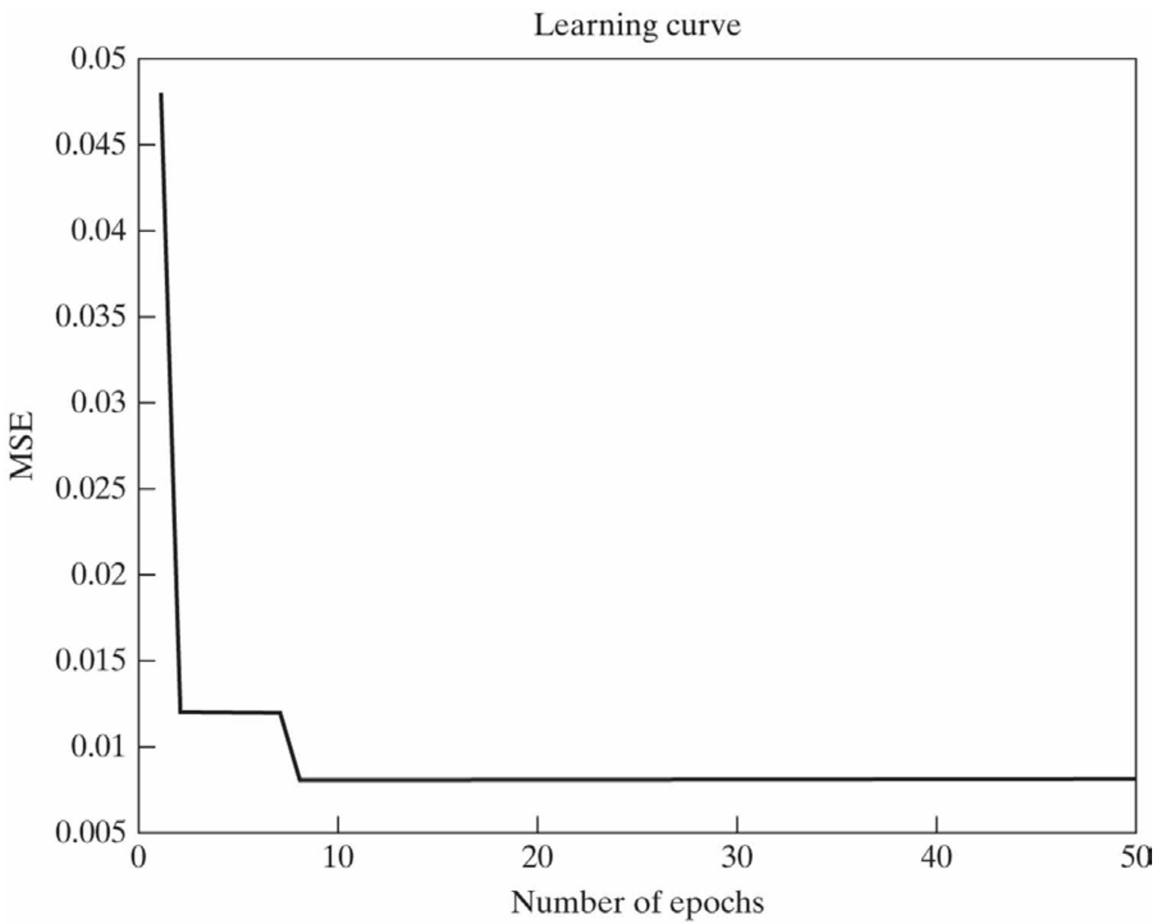
(a) Learning curve

RBF net on double moon data, d = -6



(b) Testing result

RBF net on double moon data, d = -6



(a) Learning curve