T-61.5130 Machine Learning and Neural Networks (5 cr)

Lecture 7: Radial Basis Function Networks

Prof. Juha Karhunen

https://mycourses.aalto.fi/

Introduction

- Radial Basis Function (RBF) networks are discussed in Chapter 10 in Du's and Swamy's book "Neural Networks and Statistical Learning", Springer 2014.
- It is useful as background material, while our notations come from Ham's and Kostanic's book used earlier in our course.
- Radial Basis Function (RBF) networks are a popular alternative to multilayer perceptron (MLP) networks.
- Both these network structures use supervised training which requires a training set of known input-output vector pairs.
- RBF networks have in general simpler structure and learning methods than MLP networks.
- In radial basis function networks, supervised learning of neural

networks is viewed as a surface-fitting process in a high-dimensional space.

- In the training phase, a surface is fitted to the training data so that the approximation error becomes as small as possible.
- Different optimization criteria could in principle be used to that end.
- But in practice the standard mean-square error is used for mathematical convenience.
- For a finite number of training pairs, this equals to the least-squares fitting error.
- Generalization then means interpolation in the found surface.
- That is, new unseen data vectors are mapped onto the approximating surface determined in training the RBF network.

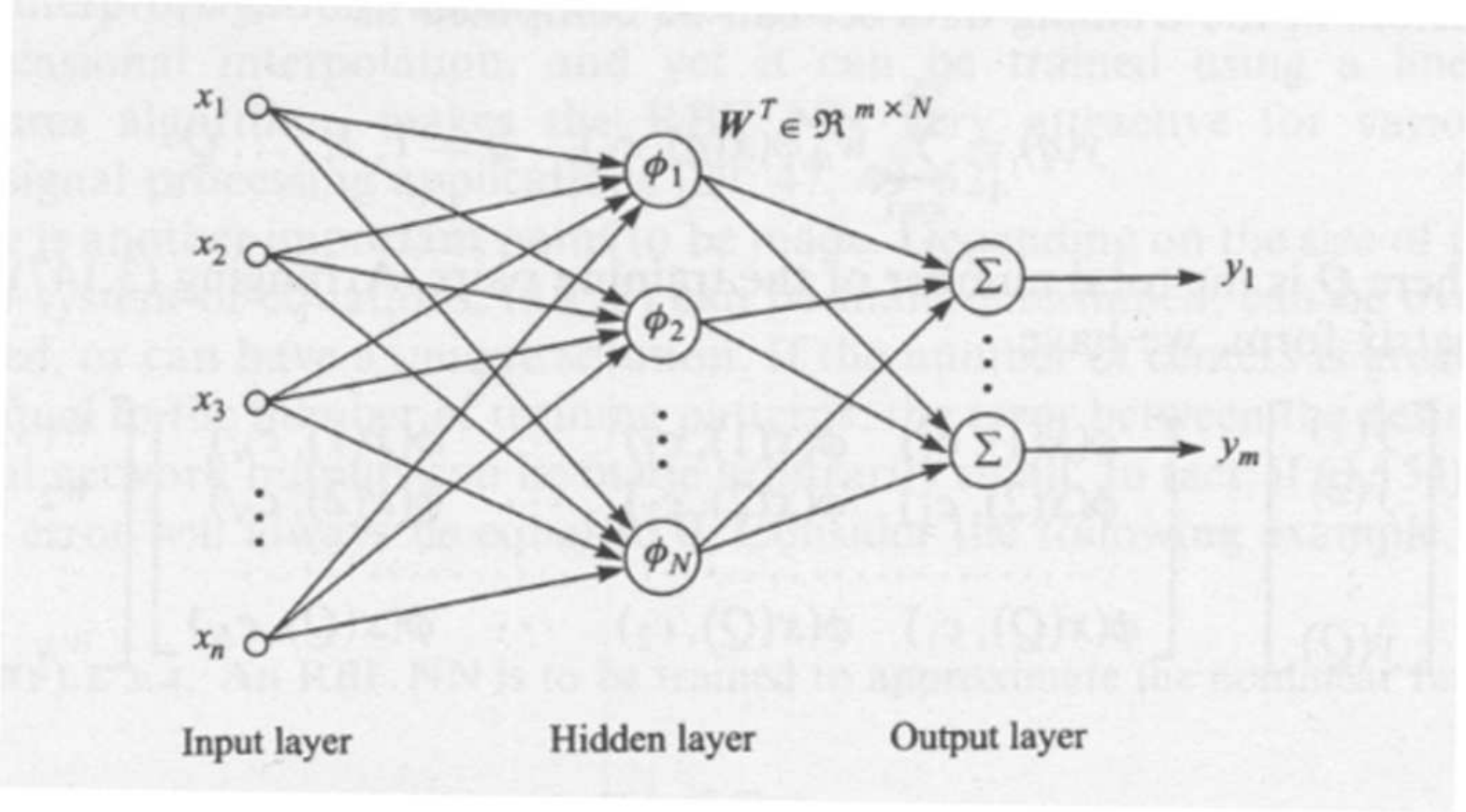

Figure 1: The architecture of radial basis function network.

Structure of the radial basis network

- The architecture of the basic radial basis function network is presented in Figure 1.
- The RBF network consists of three layers:
  - An input layer. There the data (input) vector is only inputted to the network without performing any computations.
  - A single hidden layer of nonlinear processing neurons.
  - A linear output layer.
- The output of the RBF networks is calculated according to $(i = 1, 2, \ldots, m)$

$$y_i = f_i(\mathbf{x}) = \sum_{k=1}^{N} w_{ik}\phi_k(\mathbf{x}, \mathbf{c}_k) = \sum_{k=1}^{N} w_{ik}\phi_k(\| \mathbf{x} - \mathbf{c}_k \|_2) \qquad (1)$$

- Here $y_i = f_i(\mathbf{x})$ is the output of the i:th neuron in the output layer.

- $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ is n-dimensional input vector.
- $\phi_k(\cdot)$ is the k:th scalar-valued radial basis function.
- $\| \cdot \|_2$ denotes the Euclidean norm.
- w_{ik} are the weights of the output layer.
- More specifically, w_{ik} is the connection weight from k:th neuron in the hidden layer to the i:th neuron in the output layer.
- N is the number of neurons in the hidden layer.
- And the n-dimensional vectors $\mathbf{c}_k$ are the centers of the radial basis functions in the input space.
- For each neuron k in the hidden layer, the Euclidean distance $\| \mathbf{x} - \mathbf{c}_k \|_2$ between its associated center vector $\mathbf{c}_k$ and the input vector $\mathbf{x}$ to the RBF network is computed.
- The output of the neuron k in the hidden layer is a nonlinear

function $\phi_k(\| \mathbf{x} - \mathbf{c}_k \|_2)$ of the distance.

- Finally, the output of the RBF network is computed as a weighted linear sum of the hidden layer outputs; see Eq. (1).
- The functional form $\phi_k(\cdot)$ of the radial basis functions used is fixed beforehand.
- The centers $\mathbf{c}_k$ are defined points that are assumed to perform an adequate sampling of the input vector space.
- They are usually chosen as a subset of the input data.
- From Figure 1 one can see that in the RBF network, there is no interaction between the neurons of the output layer.
- For this reason, without loss of generality, we can consider a single scalar output RBF from this on.
- A RBF network having more outputs can be considered as a

superposition of several single-output RBF networks sharing a common hidden layer.

- This is discussed in more detail in an exercise problem.

Typical choices for the radial basis functions

1. The most widely used radial basis function in practice is the Gaussian function

$$\phi(x) = \exp(-x^2/\sigma^2) \tag{2}$$

   There the spread parameter σ controls the width of the Gaussian RBF.

2. The thin-plate-spline function

$$\phi(x) = x^2 \ln x \tag{3}$$

3. The multiquadratic function

$$\phi(x) = \sqrt{x^2 + \sigma^2} \tag{4}$$

4. The inverse multiquadratic function

$$\phi(x) = \frac{1}{\sqrt{x^2 + \sigma^2}} \tag{5}$$

- Du and Swamy mention also the logistic function given in their Eq. (10.8), but the Gaussian function is by far the most popular choice in practice.
- Note that the Gaussian function and the inverse multiquadratic function are localized in the sense that $\phi(x) \to 0$ as $x \to \infty$.
- On the contrary, the thin-plate-spline function and the multiquadratic function are nonlocal because $\phi(x) \to \infty$ as $x \to \infty$.

Training the RBF network with fixed centers

- There are two sets of parameters governing the mapping properties of the RBF network.
- Namely the weights w_{ik} in the output layer and the centers $\mathbf{c}_k$ of the radial basis functions.
- The simplest form of training a RBF network is to use fixed centers.
- They are commonly chosen randomly from the input vectors.
- The justification for this choice is that if there are sufficiently many centers selected randomly from input data, they are distributed according to the probability density of the input data.
- A problem with this approach is that it is very difficult to determine what is a sufficient number of centers to achieve adequate sampling of the input space.

- Common practice is to select a relatively large number of input vectors as the centers for ensuring adequate sampling.
- After the network has trained, some of the centers may be removed systematically without significantly degrading the performance of the RBF network.
- Assume now that we have chosen N center vectors $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_N$, and that we have at disposal Q training vector $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_Q$.
- The (scalar) output of the RBF network to the q:th training vector $\mathbf{x}_q$ is

$$\hat{y}(q) = \sum_{k=1}^{N} w_{ik}\phi(\mathbf{x}(q), \mathbf{c}_k), \qquad q = 1, 2, \ldots, Q \tag{6}$$

- Here we have assumed that all the k radial basis functions used in (6) have the same functional form $\phi_k(\cdot) = \phi(\cdot)$.
- The equations (6) can be arranged in vector-matrix form

$$\hat{\mathbf{y}} = \mathbf{\Phi}\mathbf{w} \tag{7}$$

- There the Q-dimensional actual output vector of the network is

$$\hat{\mathbf{y}} = [\hat{y}(1), \hat{y}(2), \ldots, \hat{y}(Q)]^T \tag{8}$$

- The N-dimensional weight vector of the single output neuron is (omitting the index i of the neuron from now on)

$$\mathbf{w} = [w_1, w_2, \ldots, w_N]^T \tag{9}$$

- And $\mathbf{\Phi}$ is $Q \times N$ matrix whose element (i,j) is $\phi_k(\mathbf{x}(i), \mathbf{c}_j)$:

$$\mathbf{\Phi} = \begin{bmatrix} \phi(\mathbf{x}(1), \mathbf{c}_1) & \phi(\mathbf{x}(1), \mathbf{c}_2) & \dots & \phi(\mathbf{x}(1), \mathbf{c}_N) \\ \phi(\mathbf{x}(2), \mathbf{c}_1) & \phi(\mathbf{x}(2), \mathbf{c}_2) & \dots & \phi(\mathbf{x}(2), \mathbf{c}_N) \\ \dots & \dots & \dots & \dots \\ \phi(\mathbf{x}(Q), \mathbf{c}_1) & \phi(\mathbf{x}(Q), \mathbf{c}_2) & \dots & \phi(\mathbf{x}(Q), \mathbf{c}_N) \end{bmatrix} \tag{10}$$

- Note that after the centers have been chosen and the training data is available, the matrix $\mathbf{\Phi}$ is fixed.
- The only free adjustable parameters are the elements of the weight vector $\mathbf{w}$.
- They can be determined by optimizing the performance of the network in some sense.
- The optimal weights are usually computed by minimizing the least-squares error between the actual $\hat{y}(q)$ and desired outputs

$y_d(q)$ of the RBF network.

- This yields the criterion

$$J(\mathbf{w}) = \frac{1}{2}\sum_{q=1}^{Q}[y_d(q) - \hat{y}(q)]^2 = \frac{1}{2}(\mathbf{y}_d - \hat{\mathbf{y}})^T(\mathbf{y}_d - \hat{\mathbf{y}}) \tag{11}$$

where the Q-dimensional vector of desired outputs is

$$\mathbf{y}_d = [y_d(1), y_d(2), \ldots, y_d(Q)]^T \tag{12}$$

- The optimal weight vector $\mathbf{w}_{LS}$ can be solved from the normal equations

$$\mathbf{\Phi}^T\mathbf{\Phi}\mathbf{w}_{LS} = \mathbf{\Phi}^T\mathbf{y}_d \tag{13}$$

- See exercise 5 for the derivation of this standard result.
- Solving for $\mathbf{w}_{LS}$ yields (assuming that $Q \geq N$)

$$\mathbf{w}_{LS} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{y}_d = \mathbf{\Phi}^+\mathbf{y}_d \tag{14}$$

where the matrix $\mathbf{\Phi}^+$ is called the pseudoinverse of the matrix $\mathbf{\Phi}$.

- Hence the problem of training the RBF network has a closed-form solution in the case of fixed centers.
- This means that such a RBF network can be trained very fast.
- Compared with the iterative, slowly converging backpropagation algorithm, and many other learning algorithms.
- Yet the RBF network performs a nonlinear multidimensional interpolation.
- These properties make the RBF network very attractive for various types of signal processing applications.
- Note that if the number N of centers is larger than the number Q of training pairs, the normal equations (13) become underdetermined.
- That is, there are more unknown parameters than equations.

- Then $N - Q$ of the weights w_i can be selected arbitrarily.
- In this case, the pseudoinverse solution is not given by Eq. (14).
- Clearly, the underdetermined case is not desirable, leading to overfitting.
- If desired, the weights of the RBF network can be regularized by computing them from the modified normal equations

$$(\mathbf{\Phi}^T\mathbf{\Phi} + \lambda\mathbf{G})\mathbf{w}_{LS} = \mathbf{\Phi}^T\mathbf{y}_d \tag{15}$$

- There λ is the regularization parameter.
- And $\mathbf{G}$ is $N \times N$ matrix whose element (i,j) is $\phi(\mathbf{c}_i, \mathbf{c}_j)$.
- Derivation of this result in quite involved.
- See S. Haykin, "Neural Networks - A Comprehensive Foundation", 2nd ed., Wiley 1998, Chapter 5.

- In the case of Gaussian radial basis functions, the spread parameter σ is commonly set for all basis functions as

$$\sigma = \frac{d_{max}}{\sqrt{N}} \tag{16}$$

- Here N is the number of centers and d_{max} is the maximum distance between them.
- With this choice, the radial basis function of k:th neuron in the hidden layer is given by

$$\phi(\mathbf{x}, \mathbf{c}_k) = \exp(-\frac{N}{d_{max}^2} \parallel \mathbf{x} - \mathbf{c}_k \parallel^2) \tag{17}$$

- The formula (16) guarantees that the individual radial basis functions are not too flat or too peaked.
- Alternatively, one could use individually scaled centers with broader widths in areas of lower density.

- But this requires experimentation with the training data.
- The resulting batch algorithm for training RBF networks with fixed centers is summarized below.

Training algorithm for a Gaussian RBF network

1. Choose a sufficient number N of centers $\mathbf{c}_i$ for the radial basis functions from the set of Q input vectors $\mathbf{x}_j$. The selected centers should provide adequate sampling of the input space.
2. Calculate the spread parameter σ for the Gaussian RBF functions according to (16).
3. Calculate the matrix $\mathbf{\Phi}$ according to Eq. (10).
4. Solve the the weights of the RBF network from the normal equations (13), or from the pseudoinverse solution (14).

Training the RBF network using the stochastic gradient approach

- Using fixed centers in the RBF network leads to very simple training algorithm.
- However, a large number of centers must be selected from the input data set to perform adequate sampling.
- This produces a relatively large network even for relatively simple problems.
- In the stochastic gradient approach for training RBF networks, all the three sets of network parameters are learned.
- That is, the weights, positions of the RBF centers, and the widths of the RBF's.
- Consider now briefly the derivation of the stochastic gradient based

supervised learning algorithm for RBF networks.

- This matter is discussed in Section 10.6 in Du's and Swamy's book.
- The first step is to define the instantaneous error cost function according to

$$J(n) = \frac{1}{2}[e(n)]^2 = \frac{1}{2}[y_d(n) - \sum_{k=1}^{N} w_k(n)\phi(\mathbf{x}(n), \mathbf{c}_k(n))]^2 \quad (18)$$

- For Gaussian radial basis functions,

$$\phi(\mathbf{x}(n), \mathbf{c}_k(n)) = \exp(-\frac{\| \mathbf{x}(n) - \mathbf{c}_k(n) \|^2}{\sigma_k^2(n)}) \quad (19)$$

- The update equations for the network parameters are

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu_w \frac{\partial J(n)}{\partial \mathbf{w}} \quad (20)$$

$$\mathbf{c}_k(n+1) = \mathbf{c}_k(n) - \mu_c \frac{\partial J(n)}{\partial \mathbf{c}_k} \tag{21}$$

$$\sigma_k(n+1) = \sigma_k(n) - \mu_\sigma \frac{\partial J(n)}{\partial \sigma_k} \tag{22}$$

- The gradients in the update formulas (20), (21), and (22) are evaluated for the current values $\mathbf{w}(n)$, $\mathbf{c}_k(n)$, and $\sigma_k(n)$.
- Derivation of the formulas for the gradients is left as an exercise problem.
- Updating of the centers and spread parameters of the RBF network improves its performance greatly compared to the fixed centers case.
- But the training algorithm becomes more complicated, and training of the RBF takes more time.
- Furthermore, the training algorithms can get stuck in a local minimum.

- In general, the learning rate parameters μ_w, μ_c, and μ_c are set to different values.
- The learning rules are still less complex than backpropagation.

Orthogonal least squares

- A major challenge in the design of the radial basis function network is the selection of the centers.
- The orthogonal least-squares (OLS) method offers a systematic method for center selection in RBF networks.
- It significantly reduces the size of the RBF neural network.
- Both a batch and a recursive version of the OLS method are presented in Du's and Swamy's book in Section 10.6.
- And the recursive version more thoroughly in subsection 3.6.3 in the

earlier used book F. Ham and I. Kostanic, "Principles of Neurocomputing for Science and Engineering", McGraw Hill, 2001.

- The OLS method is based on orthogonalization and least-squares regression.
- We skip the derivations and formulas; you can find them in the books mentioned above.

Example on the orthogonal least squares method

- Instead, we present example 3.5 from Ham's and Kostanic's book.
- The task is to design an RBF network that approximates the mapping

$$z = f(x, y) = \cos(3x)\sin(2y) \tag{23}$$

in the square $-1 \leq x \leq 1$ and $-1 \leq y \leq 1$.

- To accomplish this, an RBF network having 121 centers is first used.

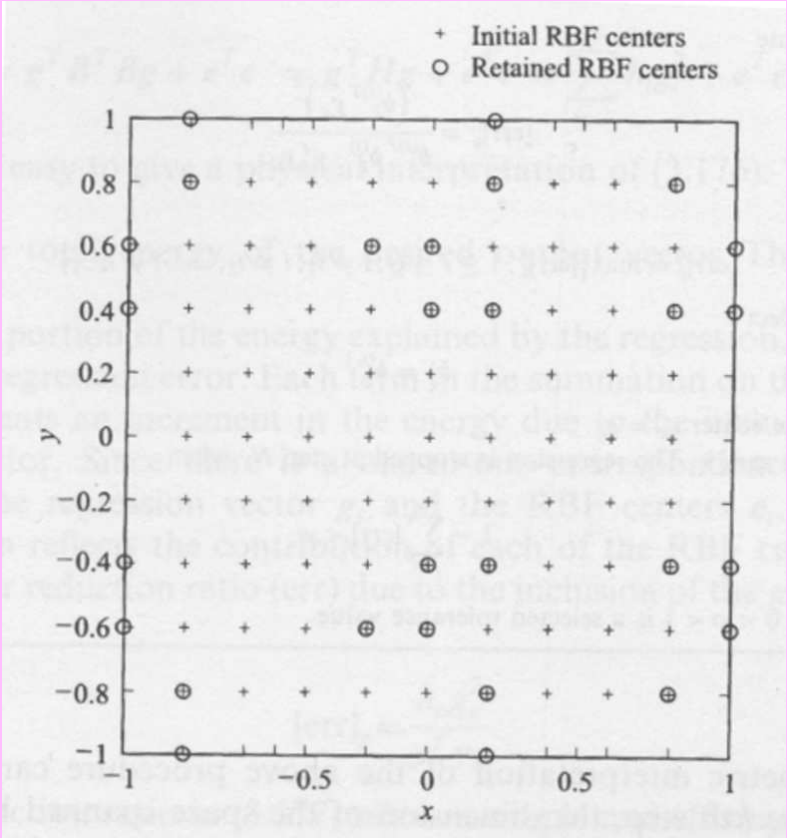

Figure 2: Initial RBF centers (+) and centers retained by the orthogonal least squares method (∘) in Example 3.5.

- These centers are placed regularly with equal spacings 0.2 in both x and y directions over the square.
- They are denoted by the symbol $+$ in Figure 2.
- The network was trained to perform exact mapping of all 121 centers using a Gaussian RBF function.
- There the spread parameter was set to $\sigma = 0.3$.
- The orthogonal least squares algorithm was used to reduce the size of the RBF network, and the error was set to 1%.
- The OLS algorithm retained 28 centers, denoted by $\circ$ in Figure 2.
- Thus the size of the hidden layer of the RBF network was reduced by 77%, w ith a minimal reduction in performance.

Universal approximation property of RBF networks

- Radial basis function networks have a similar universal approximation property as multilayer perceptron networks.
- Let $\phi(\mathbf{x})$ be a continuous, integrable bounded scalar function of the vector $\mathbf{x}$.
- Consider the mapping realized by RBF networks:

$$F(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi(\frac{\mathbf{x} - \mathbf{c}_i}{\sigma}) \tag{24}$$

- There the common width $\sigma > 0$.
- The center vectors $\mathbf{c}_i$ have the same dimension as the data (input) vector $\mathbf{x}$.

- The universal approximation theorem states that for any continuous input-output mapping $f(\mathbf{x})$:
  - One can find a RBF network with a set of centers $\mathbf{c}_i$, $i = 1, 2, \ldots, N$ and a common width $\sigma > 0$
  - Such that the input-output mapping $F(\mathbf{x})$ is close to $f(\mathbf{x})$.
- Note that the functions $\phi(\mathbf{x})$ are not required to be radially symmetric.
- The theorem provides the theoretical foundation for the design of RBF neural networks for practical applications.

A comparison with extreme learning machine

- RBF network resembles in several ways extreme learning machine (ELM):
  - Both are feedforward neural networks with a single hidden layer.

  - The single hidden layer is usually large with many neurons for providing adequate performance.
  - The output layer is linear.
  - The weights of the output layer are computed from the linear least-squares (pseudoinverse) solution.

- But these methods differ somewhat in that:
  - ELM uses inner products $\mathbf{w}_i^T \mathbf{x}$ of the input vector $\mathbf{x}$ and weight vectors $\mathbf{w}_i$ of the hidden layer neurons;
  - While RBF network uses Euclidean distances $\| \mathbf{x} - \mathbf{c}_i \|$ between the input vector and center vectors $\mathbf{c}_i$ of hidden layer neurons.
  - In ELM, the weight vectors $\mathbf{w}_i$ and bias terms in the hidden layer neurons are selected completely randomly;
  - While in RBF network the center vectors $\mathbf{c}_i$ are selected at most randomly among input vectors, or learned from the data.

Computer experiment: two overlapping Gaussians

- Recall the classification problem discussed at the end of lecture 4.
- The two-dimensional data vectors belong to two overlapping classes $\mathcal{C}_1$ and $\mathcal{C}_2$.
- Their Gaussian probability densities are reproduced in Figure 3.
- Figure 4 shows samples of the data vectors; note different vertical and horizontal scales.
- Due to the overlap, the theoretically optimal Bayes classifier achieves the classification rate 81.51%.
- The optimum decision boundary is a circle of center $\mathbf{x}_c = [-2/3, 0]^T$ and radius $r = 2.34$.

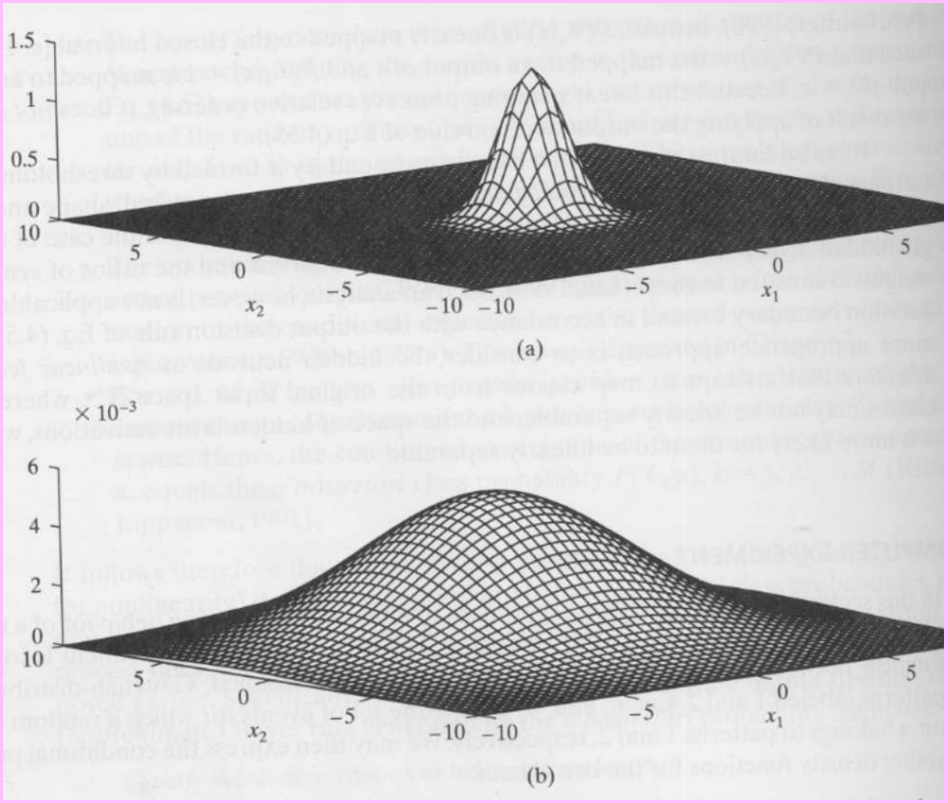

Figure 3: The Gaussian probability density functions of class $\mathcal{C}_1$ (upper figure a) and $\mathcal{C}_2$ (lower figure b).

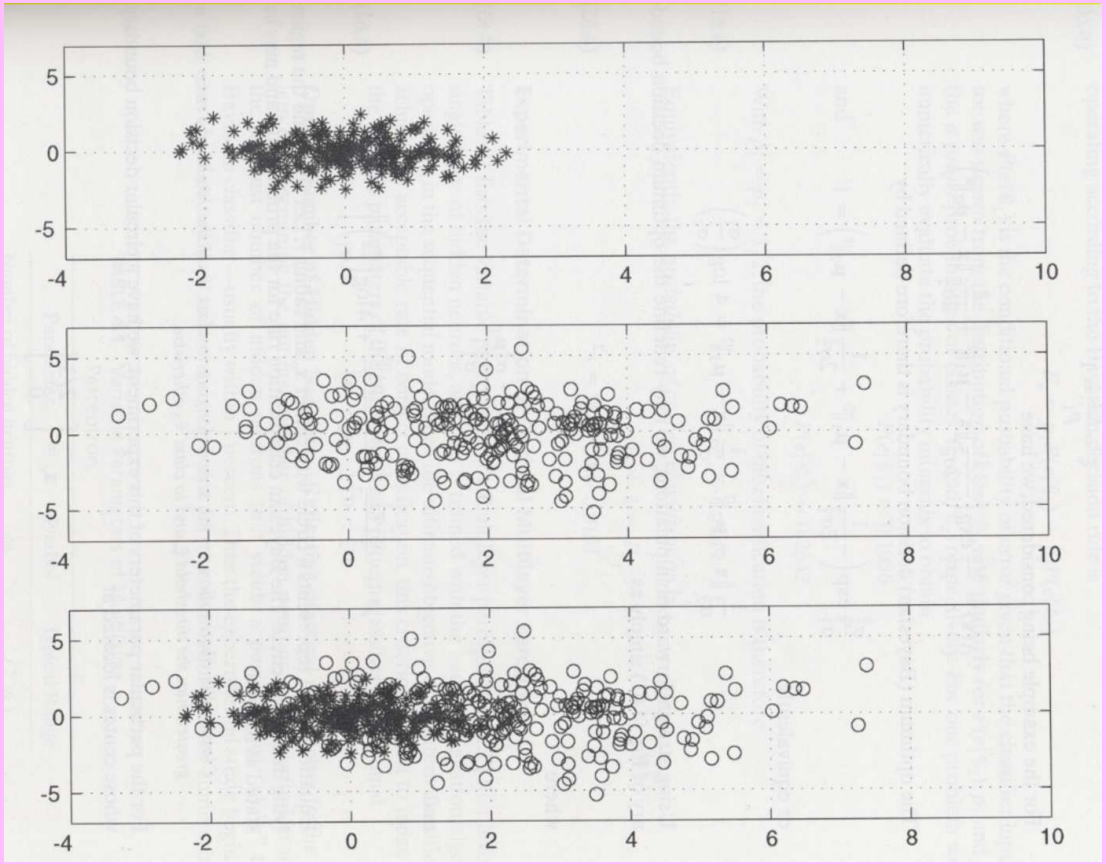

Figure 4: Data vectors of class $\mathcal{C}_1$ (top), class $\mathcal{C}_2$ (middle), and both the classes (bottom).

- In these experiments, a regularized RBF network using interpolation was used.
- This version is discussed in part 4 of Section 5.13 in S. Haykin's book "Neural Networks - A Comprehensive Foundation, 2nd ed.", Prentice-Hall, 1998.
- This variant is not discussed in our course.
- But we present experimental results for giving an idea on the performance of RBF networks.
- The number of centers (neurons in the hidden layer) was 100; with 20 centers the results were clearly poorer.
- For 50 trials, the average classification accuracy was 77.87% with the best choice of regularization parameter.
- The best result was 79.10% and worst one 75.10%.

- Three best classification borders are shown in Figure 5 and three worst ones in Figure 6.
- The average classification accuracy is somewhat poorer than for a MLP network trained using backpropagation algorithm which required only 2 neurons in the hidden layer.
- But for the RBF networks the results vary less.

Final remarks

- In general, RBF networks are fairly easy to implement, and in their basic form learning is very fast.
- But their performances in classification and regression problems are not as good as with the best methods.

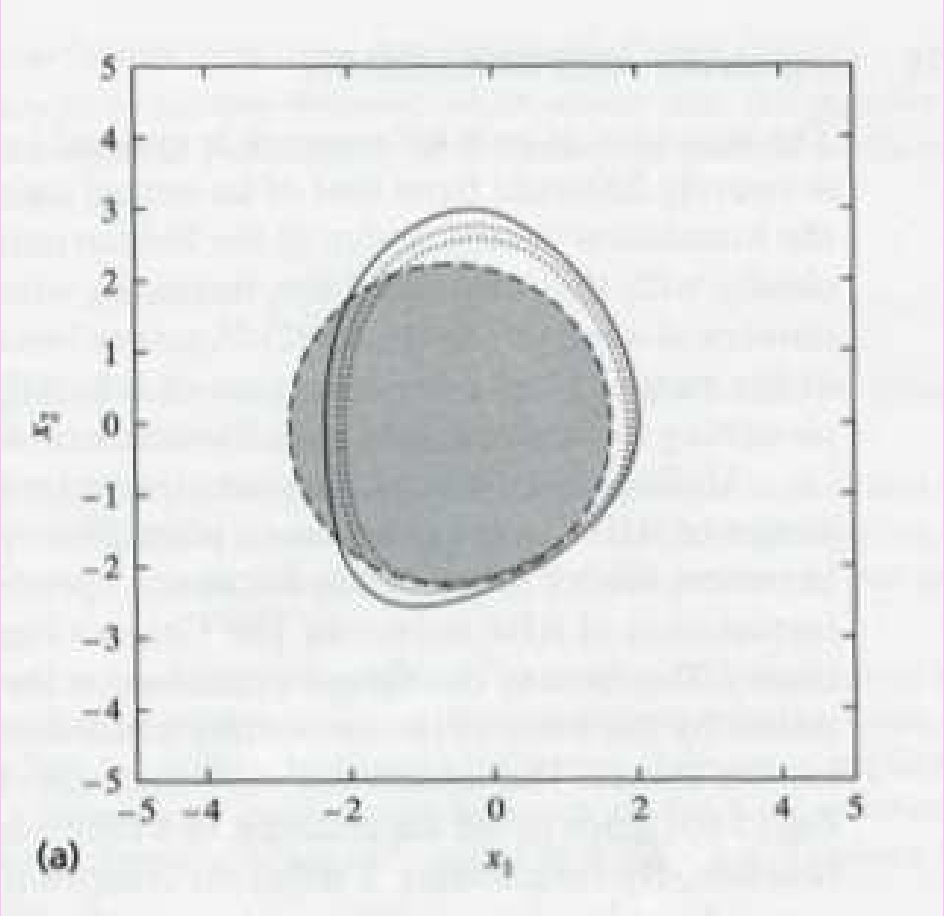

Figure 5: Three best classification borders.

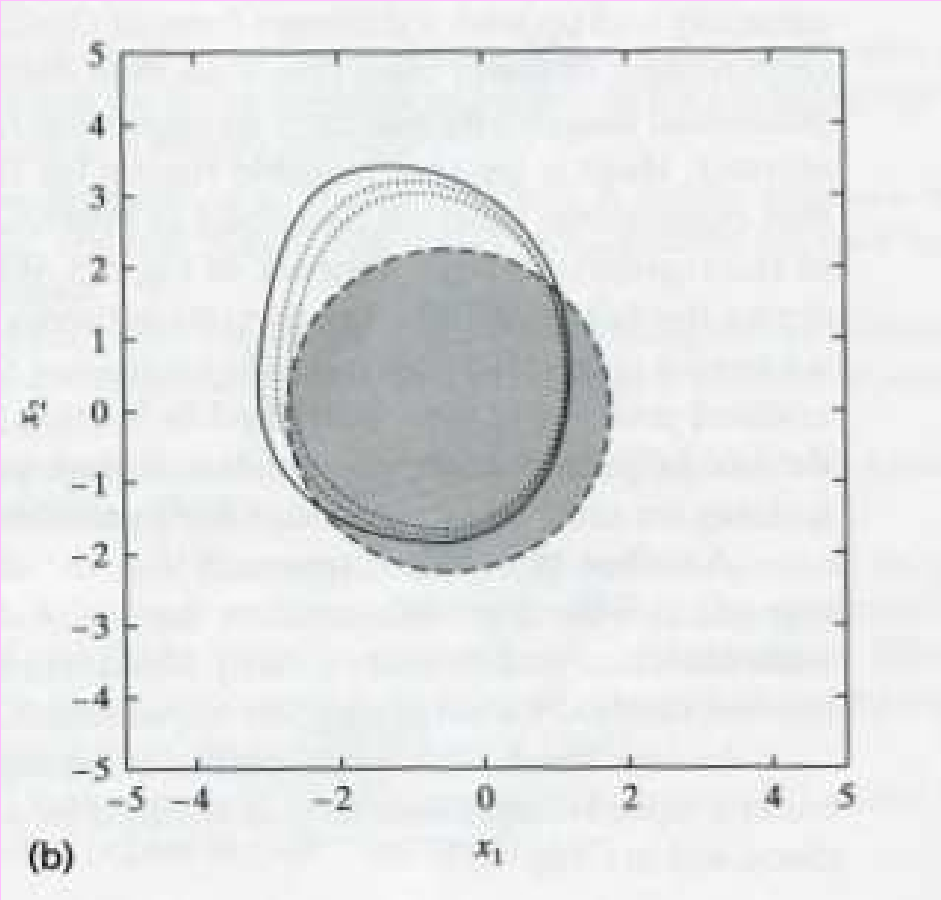

Figure 6: Three worst classification borders.