Unit 6: Radial-Basis Function Networks

A Hybrid Approach to Pattern Classification

Kiran Bagale

August, 2025

Introduction

Traditional vs. Hybrid Approach

Traditional Approach

- Back-propagation learning
- Stochastic approximation
- Single-stage training

Hybrid Approach (RBF)

- Two-stage procedure
- Nonlinear transformation
- Linear classification

Two-Stage Hybrid Approach

Stage 1: Nonlinear Transformation

Objective

Transform nonlinearly separable patterns into a new set where patterns are likely to become linearly separable

- Mathematical foundation: Cover's theorem (1965)
- Increases likelihood of linear separability
- Unsupervised training of hidden layer

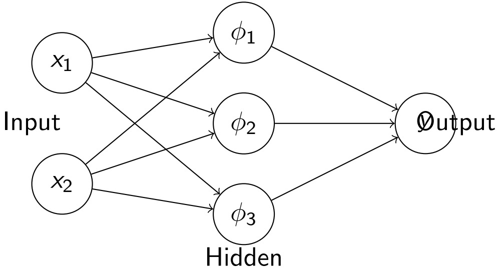
Stage 2: Linear Classification

Objective

Complete the classification using least-squares estimation

- Uses transformed feature space
- Linear output layer
- Supervised training

RBF Network Architecture: Three-Layer Structure



Layer Functions

1. **Input Layer**
   - Source nodes (sensory units)
   - Connects network to environment
2. **Hidden Layer**
   - Applies nonlinear transformation
   - High dimensionality
   - Trained unsupervised (Stage 1)
3. **Output Layer**
   - Linear transformation
   - Supplies network response
   - Trained supervised (Stage 2)

Cover's Theorem Requirements

Two Conditions for Linear Separability

1. Nonlinear transformation from input space to hidden space
2. High dimensionality of the hidden space

Key Insight

RBF networks satisfy both conditions of Cover's theorem, making linear separability in the feature space highly likely.

Gaussian Radial-Basis Function

- Most important member of RBF class
- Mathematical form: $\phi(\mathbf{x}) = \exp\left(-\frac{||\mathbf{x}-\mathbf{c}||^2}{2\sigma^2}\right)$
- Can be viewed as a **kernel**

Kernel Method

Two-stage procedure based on Gaussian function:

- Kernel transformation (Stage 1)
- Linear classification (Stage 2)

Connection to Statistics

- RBF networks relate to kernel regression in statistics
- Bridge between neural networks and statistical methods
- Theoretical foundation for understanding RBF performance

Neural Networks $\leftrightarrow$ **Statistical Methods**

Advantages of RBF Networks

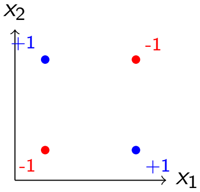
- Faster training than backpropagation
- Strong theoretical foundation
- Universal approximation capability
- Good generalization properties
- Clear separation of learning stages

6.2 Cover's Theorem on the Separability of Patterns

The Fundamental Challenge

Real-world classification problems:

- Often nonlinearly separable
- Linear classifiers fail
- Need complex decision boundaries



XOR Problem

Cover's Key Insight (1965)

Nonlinearly separable problems in low dimensions become linearly separable in high dimensions with high probability!

Problem Setup

Given

- Input patterns: $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \in \mathbb{R}^n$
- Class labels: $y_i \in \{-1, +1\}$
- Nonlinear mapping: $\phi : \mathbb{R}^n \to \mathbb{R}^m$ where $m \gg n$

Linear Separability in Feature Space

Exists hyperplane $\mathbf{w}^T \phi(\mathbf{x}) + b = 0$ such that:

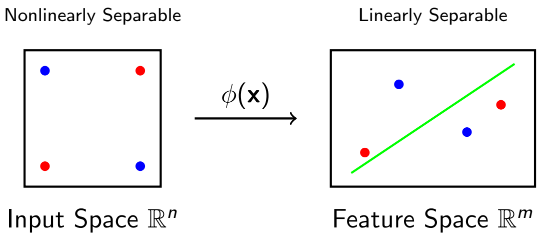
$$\mathbf{w}^T \phi(\mathbf{x}_i) + b > 0 \quad \text{for class } +1 \tag{1}$$

$$\mathbf{w}^T \phi(\mathbf{x}_i) + b < 0 \quad \text{for class -1} \tag{2}$$

Mathematical Intuition

In high dimensions, the volume grows exponentially, providing exponentially more "room" for separating hyperplanes.

The Transformation Process



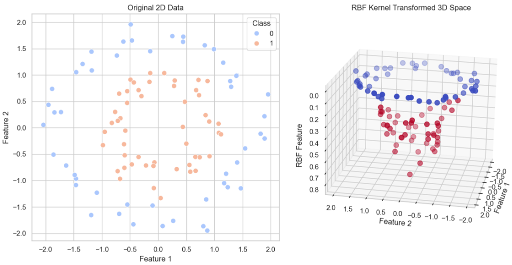


Figure: Separability of Patterns

Theorem Statement

Theorem (Cover's Theorem)

Given N patterns $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ *in* $\mathbb{R}^n$*, if mapped nonlinearly into feature space of dimension m, the probability of linear separability is:*

$$P(\mathit{linear\ separability}) = \frac{1}{2^N} \sum_{k=0}^{m-1} \binom{N-1}{k}$$

Key Conditions

1. General nonlinear mapping (not data-specific)
2. Sufficiently large feature dimension m
3. Patterns in "general position"

Critical Cases

Case 1: $m \geq N$

$$P = 1$$

Perfect separability!
High dimensions provide enough "room" for any separating hyperplane.

Case 2: $m < N$

$$P = \frac{1}{2^{N-1}} \sum_{k=0}^{m-1} \binom{N-1}{k}$$

Probability increases exponentially as $m \to N$.

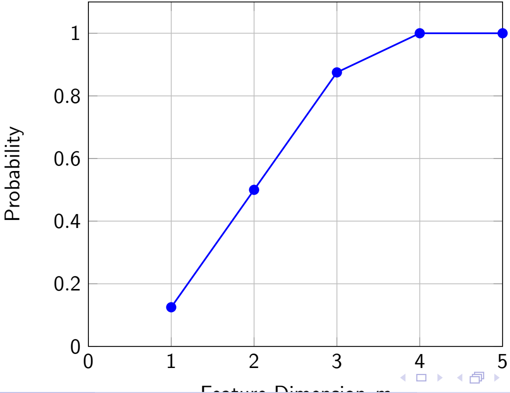
Key Insight

As feature dimension increases toward the number of patterns, linear separability becomes almost certain!

Example: N=4 Patterns

For $N = 4$ patterns: $P = \frac{1}{8} \sum_{k=0}^{m-1} \binom{3}{k}$

m	Binomial Sum	P	Percentage
1	$\binom{3}{0} = 1$	$\frac{1}{8}$	12.5%
2	$1 + \binom{3}{1} = 4$	$\frac{4}{8}$	50%
3	$1 + 3 + \binom{3}{2} = 7$	$\frac{7}{8}$	87.5%
≥ 4	$1 + 3 + 3 + 1 = 8$	$\frac{8}{8}$	100%



XOR Problem Transformation

Input Space ($\mathbb{R}^2$): Not linearly separable

Input $\mathbf{x}$	Class	2D Coordinates
$(0, 0)$	-1	$(0, 0)$
$(0, 1)$	$+1$	$(0, 1)$
$(1, 0)$	$+1$	$(1, 0)$
$(1, 1)$	-1	$(1, 1)$

Feature Space ($\mathbb{R}^3$): $\phi(\mathbf{x}) = [x_1, x_2, x_1 x_2]^T$

Input	Feature Vector	Class
$(0, 0)$	$[0, 0, 0]^T$	-1
$(0, 1)$	$[0, 1, 0]^T$	$+1$
$(1, 0)$	$[1, 0, 0]^T$	$+1$
$(1, 1)$	$[1, 1, 1]^T$	-1

Separating hyperplane: $w_1 x_1 + w_2 x_2 - 2 w_3 x_1 x_2 + b = 0$

Curse vs. Blessing of Dimensionality

Low Dimensions

- Points are "crowded"
- Limited hyperplane orientations
- Complex boundaries needed

High Dimensions

- Points spread out
- Many hyperplane orientations
- Linear separation likely

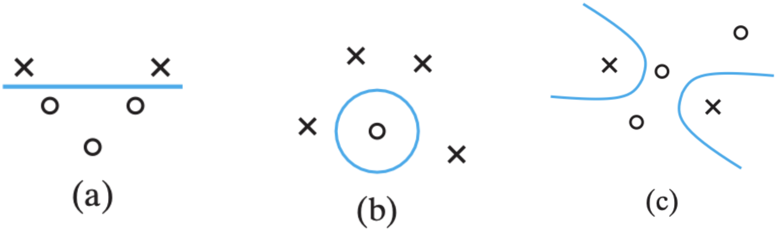
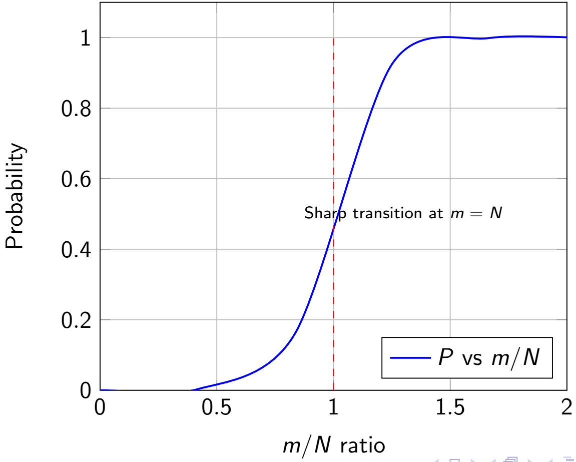


Figure: Three examples of φ-separable dichotomies of different sets of five points in two dimensions: (a) linearly separable dichotomy; (b) spherically separable dichotomy; (c) quadrically separable dichotomy.

Asymptotic Behavior

For large N and $m \approx N$:

$$P \approx \frac{1}{\sqrt{\pi N/2}} \sum_{k=0}^{m-1} \exp\left(-\frac{(k - N/2)^2}{N/2}\right)$$



Kernel Interpretation

The nonlinear mapping $\phi(\mathbf{x})$ defines a kernel function:

$$K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$$

Popular Kernels

- **Gaussian RBF**: $K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x}-\mathbf{x}'||^2}{2\sigma^2}\right)$
- **Polynomial**: $K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + 1)^d$
- **Sigmoid**: $K(\mathbf{x}, \mathbf{x}') = \tanh(\alpha\mathbf{x}^T\mathbf{x}' + \beta)$

Key Advantage

We can work with kernels without explicitly computing $\phi(\mathbf{x})$ - the "kernel trick"!

RKHS Connection

Reproducing Kernel Hilbert Spaces (RKHS)

Cover's theorem explains why kernel methods work:

1. Kernel implicitly maps to high-dimensional space
2. Linear separability highly probable in this space
3. Optimization becomes convex (linear in feature space)
4. Strong theoretical guarantees

$$\boxed{\text{Input Space}} \xrightarrow{K(\cdot,\cdot)} \boxed{\text{RKHS}} \xrightarrow{\text{Linear Sep.}} \boxed{\text{Linear Classifier}}$$

RBF Network Design Principles

Principle 1: High-Dimensional Hidden Layer

Use $m \gg n$ hidden units to ensure high probability of linear separability

Principle 2: Appropriate Basis Functions

Choose RBF centers $\mathbf{c}_i$ and widths σ_i:

$$\phi_i(\mathbf{x}) = \exp\left(-\frac{||\mathbf{x} - \mathbf{c}_i||^2}{2\sigma_i^2}\right)$$

Principle 3: Two-Stage Learning

1. **Stage 1**: Learn $\mathbf{c}_i$, σ_i (unsupervised)
2. **Stage 2**: Learn output weights (supervised, linear)

Trade-offs

Benefits

- High probability of separability
- Simple linear learning stage
- Strong theoretical foundation
- Universal approximation

Costs

- Computational complexity
- Risk of overfitting
- Need more training data
- Parameter selection

Design Guideline

Choose m large enough for separability but not so large as to cause overfitting: typically $m = O(\sqrt{N})$ to $O(N)$.

Assumptions and Limitations

Key Assumptions

- **General Position**: Patterns not in special geometric arrangements
- **Random Mapping**: Transformation not specifically designed for data
- **Infinite Precision**: Mathematical idealization

Practical Considerations

- **Finite Sample Effects**: Real datasets have limited samples
- **Noise Sensitivity**: High dimensions can amplify noise
- **Computational Reality**: Memory and time constraints

Bottom Line

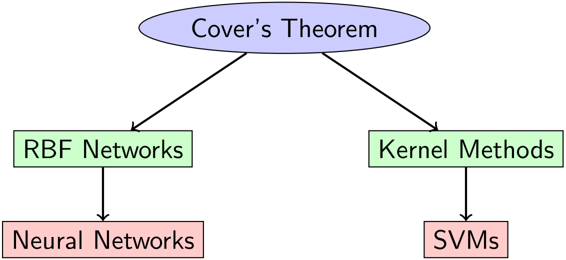
Cover's theorem provides theoretical justification, but practical implementation requires careful parameter tuning and regularization.

Key Takeaways

1. **Core Insight**: Nonlinear transformation to high dimensions makes linear separability highly probable
2. **Mathematical Foundation**: Probability approaches 1 as $m \to N$
3. **Practical Impact**: Justifies two-stage learning in RBF networks
4. **Kernel Connection**: Explains success of kernel methods and SVMs
5. **Design Guidance**: Choose sufficient but not excessive dimensionality

Cover's theorem bridges theory and practice in pattern recognition

The Big Picture



Unifying theoretical framework for nonlinear classification

6.3 The Interpolation Problem: From Theory to Practice

Cover's Theorem Application

Cover's theorem shows that non-linear mapping to high dimensions enables linear separability. But how do we implement this practically?

The Solution:

- Transform nonlinear classification into interpolation
- Use radial-basis functions
- Two-stage hybrid approach

Two-Phase Learning Process

Training Phase

Optimization of a fitting procedure for surface Γ, based on known input-output examples (patterns).

Generalization Phase

Synonymous with **interpolation** between data points, with interpolation performed along the constrained surface generated by the fitting procedure.

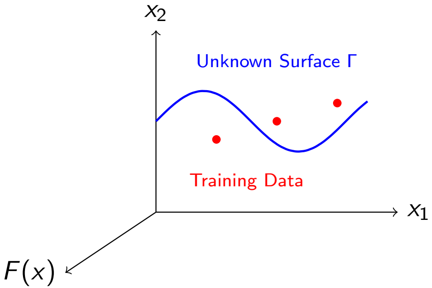
The Network as a Hypersurface

Consider a feedforward network: Input $\rightarrow$ Hidden $\rightarrow$ Output

Mathematical Representation

The network represents a mapping: $s : \mathbb{R}^{m_0} \rightarrow \mathbb{R}^1$

We can think of this map s as a **hypersurface** $\Gamma \subset \mathbb{R}^{m_0+1}$



The surface Γ is unknown and training data are contaminated with noise.
This leads us to the theory of multivariable interpolation in high dimensional space, with a long history (Davis, 1963).

The Strict Interpolation Problem

Problem Statement

Given a set of N different points $\{\mathbf{c}_i \in \mathbb{R}^{m_0} | i = 1, 2, \ldots, N\}$ and corresponding real numbers $\{d_i \in \mathbb{R}^1 | i = 1, 2, \ldots, N\}$, find a function $F : \mathbb{R}^{m_0} \to \mathbb{R}^1$ that satisfies the interpolation condition:

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

Strict Interpolation

The interpolating surface (function F) is constrained to pass through **all** the training data points.

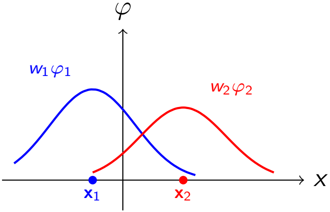
Radial-Basis Functions Solution

The RBF technique chooses a function F of the form:

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

where:

- $\{\varphi(||\mathbf{x} - \mathbf{c}_i||) | i = 1, 2, \ldots, N\}$ are **radial-basis functions**
- $|| \cdot ||$ denotes a norm (usually Euclidean)
- $\mathbf{c}_i \in \mathbb{R}^{m_0}$ are the **centers** of the RBFs
- w_i are the unknown weights to be determined



Linear System of Equations

Inserting the interpolation conditions into the RBF expansion:

$$\sum_{j=1}^{N} w_j \varphi(||\mathbf{x}_i - \mathbf{c}_j||) = d_i, \quad i = 1, 2, \ldots, N$$

This gives us the matrix equation:

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

where $\varphi_{ij} = \varphi(||\mathbf{x}_i - \mathbf{c}_j||)$

Compact Matrix Form

Let:

$$\mathbf{d} = [d_1, d_2, \ldots, d_N]^T \quad \text{(desired response vector)} \tag{3}$$

$$\mathbf{w} = [w_1, w_2, \ldots, w_N]^T \quad \text{(linear weight vector)} \tag{4}$$

$$\mathbf{\Phi} = \{\varphi_{ij}\}_{i,j=1}^{N} \quad \text{(interpolation matrix)} \tag{5}$$

The system becomes:

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

Solution

Assuming $\mathbf{\Phi}$ is nonsingular (invertible):

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

Critical Question

How can we ensure the interpolation matrix $\mathbf{\Phi}$ is nonsingular?

Micchelli's Theorem

Theorem (Micchelli, 1986)

Let $\{\mathbf{x}_i\}_{i=1}^N$ be a set of distinct points in $\mathbb{R}^{m_0}$. Then the N-by-N interpolation matrix $\mathbf{\Phi}$, whose (i,j)-th element is $\varphi_{ij} = \varphi(||\mathbf{x}_i - \mathbf{c}_j||)$, is nonsingular.

Key Requirement

The points $\{\mathbf{c}_i\}_{i=1}^N$ must all be **distinct** (different from each other).

This theorem covers a large class of radial-basis functions of particular interest in RBF networks.

Important RBF Functions

Micchelli's theorem applies to:

1. Multiquadrics (Hardy, 1971)

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

2. Inverse Multiquadrics (Hardy, 1971)

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

3. Gaussian Functions

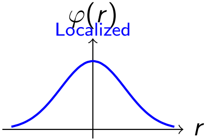
$$\varphi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right) \quad \text{for some } \sigma > 0, r \in \mathbb{R}$$

All require only that data points be **distinct** - no constraints on data size N or dimensionality m_0.

Properties of RBF Functions

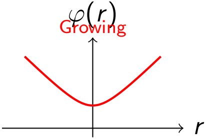
Localized Functions:

- Inverse multiquadrics
- Gaussian functions
- Property: $\varphi(r) \to 0$ as $r \to \infty$



Growing Functions:

- Multiquadrics
- Property: $\varphi(r) \to \infty$ as $r \to \infty$
- Still yield nonsingular matrices!



Remarkable Result

Functions that grow to infinity can still be used for smooth input-output mapping with great accuracy, yielding positive-definite interpolation matrices.

Matrix Properties and Implications

For Localized Functions (Inverse Multiquadrics, Gaussian)

- Interpolation matrix $\mathbf{\Phi}$ is positive definite
- All eigenvalues are positive
- Matrix is well-conditioned for numerical computation

For Growing Functions (Multiquadrics)

- Matrix has $N - 1$ negative eigenvalues
- Only one positive eigenvalue
- Matrix is NOT positive definite
- But still nonsingular and suitable for RBF networks!

Practical Implication

Both localized and growing RBF functions can achieve smooth input-output mapping with great accuracy (Powell, 1988).

Summary: Interpolation Problem

1. **Problem Transformation**: Neural network learning becomes multivariable interpolation
2. **RBF Solution**: Use radial basis functions with centers at data points
3. **Linear System**: Results in solvable linear equation $\mathbf{\Phi w} = \mathbf{d}$
4. **Theoretical Guarantee**: Micchelli's theorem ensures matrix nonsingularity
5. **Function Classes**: Multiquadrics, inverse multiquadrics, and Gaussian functions all work
6. **Practical Success**: Both localized and growing functions achieve high accuracy

The interpolation approach provides a solid mathematical foundation for RBF network design

In light of interpolation theory, we can envision a **radial-basis-function (RBF)** network as a layered structure with three layers:

1. **Input layer**: consists of m_0 source nodes, where m_0 is the dimensionality of the input vector $\mathbf{x}$
2. **Hidden layer**: consists of the same number of computation units as the size of the training sample, namely N
3. **Output layer**: consists of a single computational unit in the RBF structure

RBF Network Structure Each unit in the hidden layer is mathematically described by a radial-basis function:

$$\varphi_j(\mathbf{x}) = \varphi(||\mathbf{x} - \mathbf{c}_j||), \quad j = 1, 2, ..., N$$

Key characteristics:

- The jth input data point $\mathbf{c}_j$ defines the center of the radial-basis function
- The vector $\mathbf{x}$ is the signal (pattern) applied to the input layer
- Links connecting source nodes to hidden units are **direct connections with no weights**
- Output layer size is typically much smaller than the hidden layer

We focus on using a **Gaussian function** as the radial-basis function:

$$\varphi_j(\mathbf{x}) = \varphi(\mathbf{x} - \mathbf{c}_j)$$

$$= \exp\left(-\frac{1}{2\sigma_j^2}||\mathbf{x} - \mathbf{c}_j||^2\right), \quad j = 1, 2, ..., N \quad (5.20)$$

where:

- σ_j is a measure of the **width** of the jth Gaussian function with center $\mathbf{c}_j$
- Typically, all Gaussian hidden units are assigned a common width σ
- The parameter that distinguishes one hidden unit from another is the center $\mathbf{c}_j$

Practical Modifications to RBF Networks: Addressing Real-World Issues

Challenge: The theoretical RBF network formulation has practical limitations:

- Training samples $\{\mathbf{x}_i, d_i\}_{i=1}^N$ are typically **noisy**
- Interpolation based on noisy data can lead to misleading results
- Having a hidden layer of the same size as the input layer is computationally wasteful for large training samples

Theoretical vs Practical RBF Networks

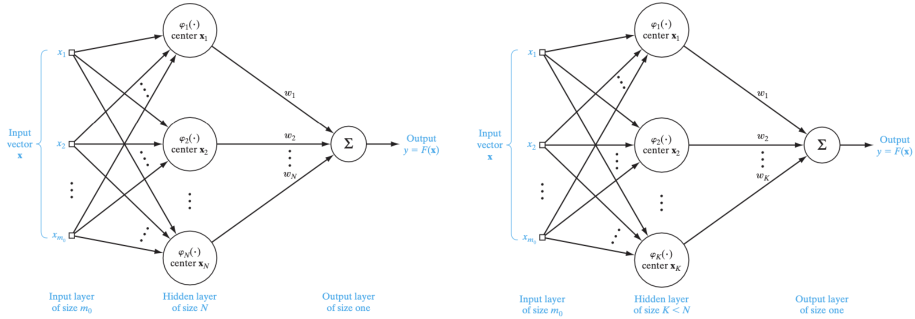


Figure: Structure of a practical RBF networks

Note: $x_j = c_j$

Improved RBF Network Structure

Reducing Computational Complexity

Key insight: Correlation between adjacent data points in the training sample creates redundancy in the hidden layer.
Practical design principle: Make the size of the hidden layer a fraction of the size of the training sample.

Advantages

- Reduces computational resources
- Networks in Fig. (a) and (b) share a common mathematical structure
- Unlike multilayer perceptron, RBF training does **not involve back propagation** of error signals

The approximating function realized by practical RBF structures:

$$F(\mathbf{x}) = \sum_{j=1}^{K} w_j \varphi(\mathbf{x}, \mathbf{c}_j)$$

where:

- Input vector dimensionality: m_0
- Each hidden unit characterized by: $\varphi(\mathbf{x}, \mathbf{c}_j)$, $j = 1, 2, ..., K$
- $K < N$ (smaller than training set size)
- Output layer characterized by weight vector $\mathbf{w}$ with dimensionality K

Comparison: Theoretical vs. Practical RBF Networks

Key Differences

Theoretical RBF (Fig. (a))

- Hidden layer dimensionality: N
- $N =$ size of training set
- Centers defined by input vectors
- Noiseless training assumption

Practical RBF (Fig. (b))

- Hidden layer dimensionality: $K < N$
- Reduced computational complexity
- Requires new procedure for center selection
- Handles noisy training data

Next Steps

The next section addresses center selection procedures for practical RBF networks using Gaussian functions.

RBF Network Training Algorithm

Input

- Training data: $\{(\mathbf{x}_i, y_i)\}$ for $i = 1$ to N
- Number of RBF centers: K (randomly selected)

Algorithm Steps

1. **Choose centers ($\mathbf{c}_j$):**
   Use K-Means clustering or randomly select K samples from the training data.
2. **Choose spread (σ):**
   A common choice: $\sigma = \frac{d_{\text{max}}}{\sqrt{2K}}$
   where d_{max} is the maximum distance between any two centers.

RBF Network Algorithm (continued)

Algorithm Steps (continued)

3. **Compute RBF activations for each input:**
   For each input $\mathbf{x}_i$ and center $\mathbf{c}_j$, compute:

$$\Phi_{ij} = \varphi_j(\mathbf{x}_i) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{c}_j\|^2}{2\sigma^2}\right) \tag{6}$$

4. **Train output weights using linear regression:**

$$\mathbf{w} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{y} \tag{7}$$

5. **Predict for new input x:**

$$\hat{y} = \sum_{j=1}^{K} w_j \cdot \varphi_j(\mathbf{x}) + b \tag{8}$$

Thank You!!!