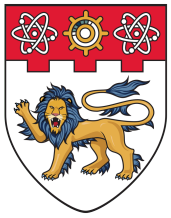

NANYANG TECHNOLOGICAL UNIVERSITY

SINGAPORE

NANYANG TECHNOLOGICAL UNIVERSITY

EE7207 Neural Networks & Deep Learning

Assignment 1 Report

MSc in Electrical and Electronic Engineering
Yan Ziming
G2507084J

February 8, 2026

Contents

1 Introduction

The report introduces the process of classification with RBF neural network. The RBF contains three layers: input layer, hidden layer, and output layer.

The key obstacle during the development is the selection of parameters, like Gaussian width (σ) and hidden layer neuron number. I used different methods to deal with these difficulties, such as SOM, Kmeans, and nonlinear optimization apart from basic process.

Briefly, I divided my design into two parts: (1) basic method to develop basic RBF (2) improve parameters with backpropogation. The final result reflect positive improvement.

2 RBF Neural Network Frame

2.1 Input Layer

The input layer of RBF is different from normal neural network, it just transmit the training data vector $\mathbf{x} = [x_1, x_2, ..., x_n]^T$ to the hidden layer. In this stage, no mathematical transformations or learning processes occur.

2.1.1 Data Analysis

The datasets are provided as static files, namely *data_train.mat*, *data_test.mat* and *label_train.mat*; therefore, manual partitioning of training and testing sets is not required.

Data Information:

Table 1: Sample Sets

Parameter	*Training Data*	*Testing Data*
Number of Samples (n)	301	50
Number of Features (m)	33	33
Categories	2	2
Missing Value	0	0
Outlier Ratio (IQR)	49.83%	54.00%

Table 2: Label Set

Parameter	*Training Label*
Number of Samples (n)	301
Categories	{-1, 1}
Distribution	[106, 195]
Missing Value	0

I tried to seek for outliers with IQR metric and considered the sample which has outlier feature element as outlier sample to compute ratio. According to the data analysis, the outlier of sample set too large to be normal. Therefore, I plot boxplot to visualize the real distributions of training data in each feature.

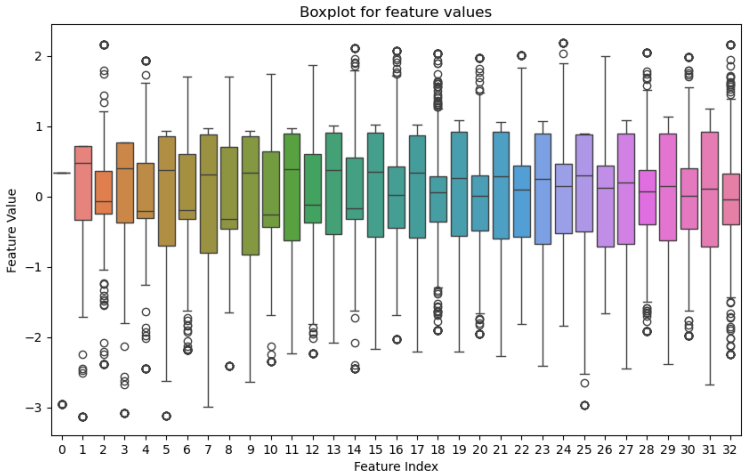

Figure 1: Boxplot of Data_train

According to figure, the main outliers come from several features. The distribution of data, like mean value and variation standardized and the range of value have normalized as well. There is no need to normalizztion and standarlization.

Generally, the high dimension leads to extreme complexity in feature space. It is the **Curse of Dimensionality**. Since the input layer need to transmit sample to hidden layer, the input layer should have 33 neurons.

2.2 Hidden Layer

The hidden layer is the computational core of the RBF network, responsible for mapping the input data into a new feature space based on the neurons in hidden layer. The RBF hidden layer utilizes radial basis functions to capture local responses. The output of the j-th hidden neuron, $\phi_j(\mathbf{x})$ is determined by the distance between the input vector $\mathbf{x}$ and a predefined center $\mathbf{c}_j$:

$$\phi_j(\mathbf{x}) = \exp\left(-\frac{|\mathbf{x}-\mathbf{c}_j|^2}{2\sigma_j^2}\right) \tag{1}$$

where $|\cdot|$ denotes the Euclidean norm, and σ_j represents the Gaussian width (spread) of the j-th neuron. This local mapping ensures that only inputs near the center $\mathbf{c}_j$ produce a significant activation.

2.2.1 Centers Selection

The selection of centers $\mathbf{c}_j$ is the most critical step in designing the hidden layer, as they serve as the "prototypes" that define the localized receptive fields within the input space. Given the 33-dimensional complexity, a robust and systematic selection strategy is employed.

In this implementation, the K-means clustering algorithm is utilized as the primary method to partition the feature space. To determine the optimal number of hidden neurons K, the Elbow Method is performed by plotting the Within-Cluster Sum of Squares (WCSS) against a range of cluster counts. The "elbow" point of the curve is identified as the optimal balance between model complexity and error reduction, providing a quantitative basis for the hidden layer's scale.

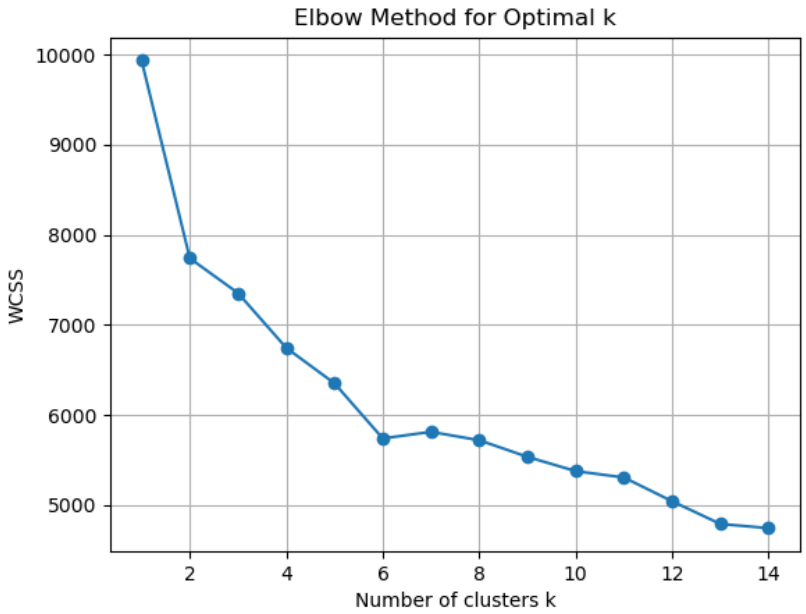

Figure 2: Cluster Number & WCSS

Since the elbow occours at $k = 6$ in the figure, I decide to use 6 hidden neuron to develop hidden layer.

To further validate the reliability of these centers in the presence of high-dimensional noise, a Self-Organizing Map (SOM) is used for auxiliary verification. While K-means efficiently minimizes local variance, the SOM provides a topological mapping that ensures the centers are not merely biased by dense clusters but also adequately cover t he "outlier-heavy" boundaries identified in our data analysis. This dual-verification approach ensures that the prototype centers $\mathbf{c}_j$ maintain a representative coverage of the manifold.

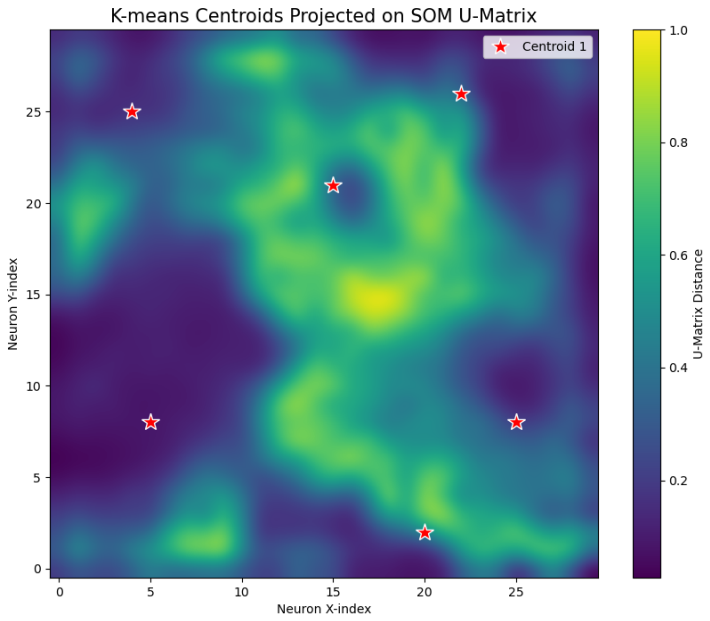

Figure 3: Centroids on U_matrix

The U_matrix map reflected the 30×30 neurons' distances, the color darker, the distances of neuron to neighbors closer, meaning more possible to belongs to same cluster. Since the centroids computed by Kmeans clustering fall into the blue "valley", the selection of 6 neurons in hidden layer roughly feasible.

2.2.2 Gaussian Width

The Gaussian width σ determines the receptive field of each hidden neuron. Setting an appropriate σ is a balancing act between sensitivity and generalization:

- **Addressing Dimensionality:** In a 33-dimensional space, the Euclidean distance between points tends to be large and sparse. If σ is too small, the receptive fields will not overlap, causing the network to lose its interpolation capability (the "dead neuron" problem).
- **Robustness to Outliers:** Conversely, an excessively large σ would over-smooth the decision boundary, potentially blurring the distinction between the categories, especially given the imbalanced distribution ([106, 195]).

A common heuristic is used to set a global width:

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

where $d_{max}$is the maximum distance between chosen centers and K is the number of hidden units. This ensures that the hidden layer provides a continuous and smooth mapping across the entire input manifold.

In this experiment, the σ I computed is 2.4604.

2.2.3 Improvement

2.3 Output Layer

3 Results Analysis

3.1 Baseline

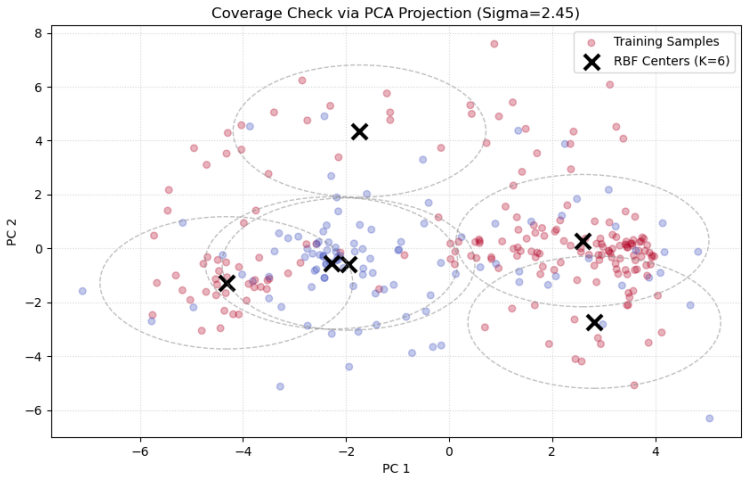

Figure 4: Coverage of Centroids

3.2 After Backpropogation

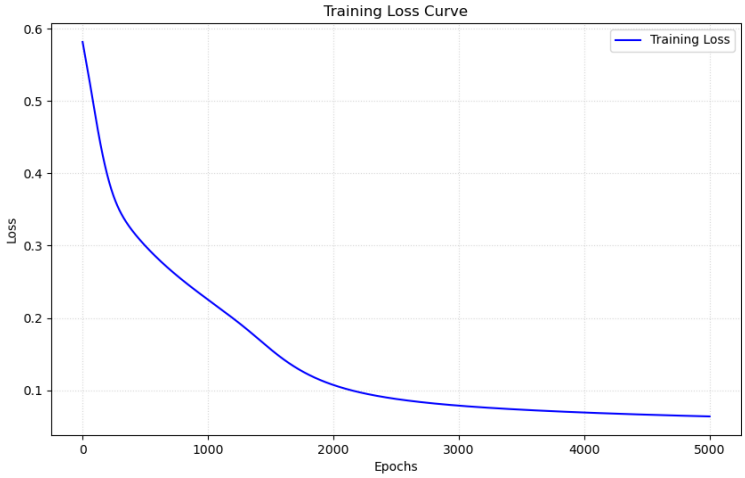

Figure 5: Training Loss

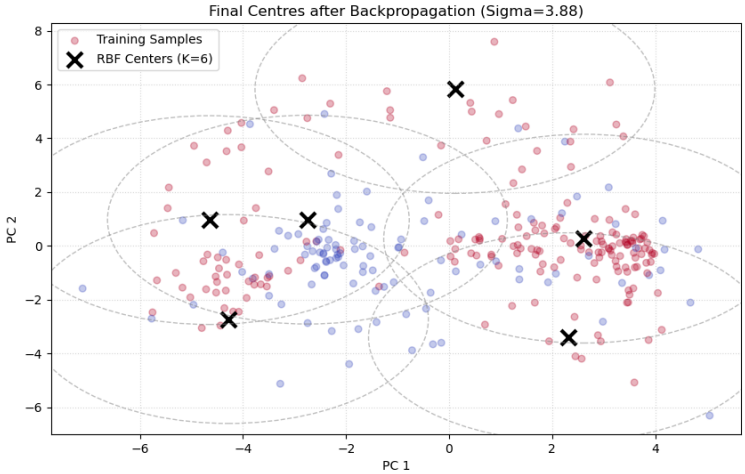

Figure 6: Center Coverage After Optimization

4 Conclusion