Fundamentals of Data Structures and Algorithms for AI

Radial Basis Function Network Report

Prepared By

Sintayew Zekarias and Fikir Awoke

December 14, 2021

Course Instructor - Dr. Beakal Gizachew

Addis ababa Institute of Technology

School of Information Technology and Engineering

Table of contents

1. Introduction

Radial Basis Function Network (RBFN) is an artificial neural network that uses radial basis functions as activation functions. It is one of the extremely fast and effective Machine Learning algorithms. Radial basis function networks can be used to solve function approximation and classification problems. These networks have three layers which are an input layer, a hidden layer with a non-linear RBF activation function and a linear output layer. An input vector x is used as input to all radial basis functions, each with different parameters and the output of the network is a linear combination of the outputs from radial basis functions.

The following figure illustrates the three layers of RBF which are the input, hidden and output layer. The first layer corresponds to the inputs of the network, the second is a hidden layer consisting of a number of RBF non-linear activation units, and the last one corresponds to the final output of the network.

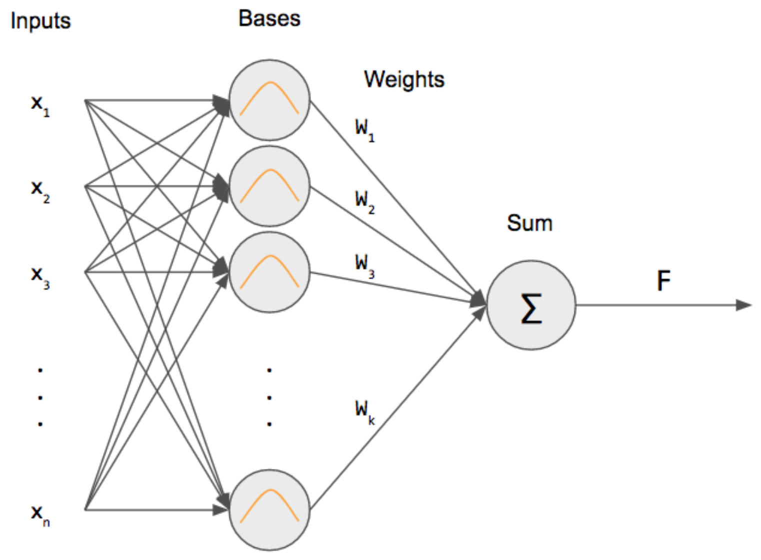


Figure 1: RBF for approximation

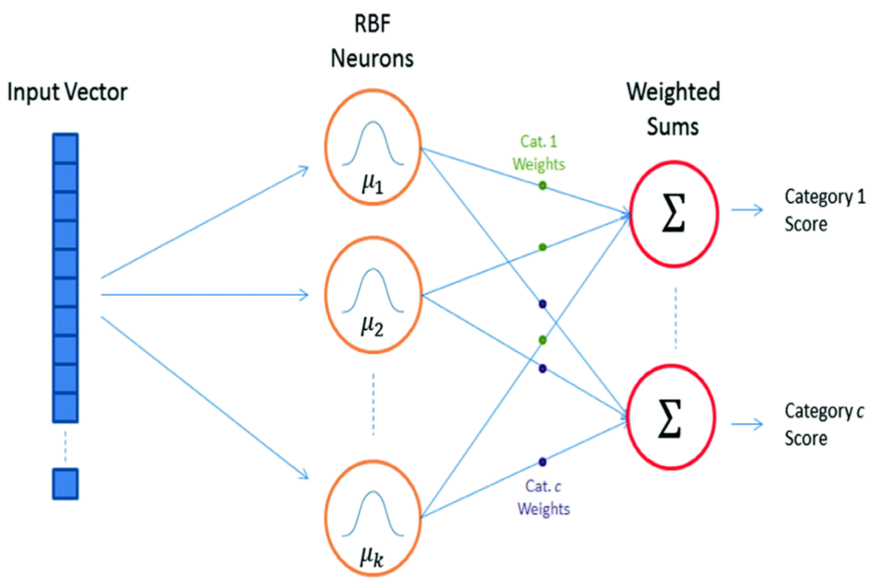


Figure 2: RBF for classification

Among different basis functions, Gaussian functions are the most commonly used ones. It is the most important of all statistical distributions and referred to as a bell curve. While implementing our RBF we used the gaussian function with the standard formula:

$$\begin{aligned}\phi(x) &= \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \\ \phi(x;\mu,\sigma) &= \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)\end{aligned}$$

The aim of this project is to implement the RBFN using different Approaches and datasets provided. In this paper we tried to show all the implementation approaches we used for RBF and the corresponding results. We have also shown the comparison between our result with the python's built in function.

2. Methodology

The following methodologies are used to achieve the objective of the project.

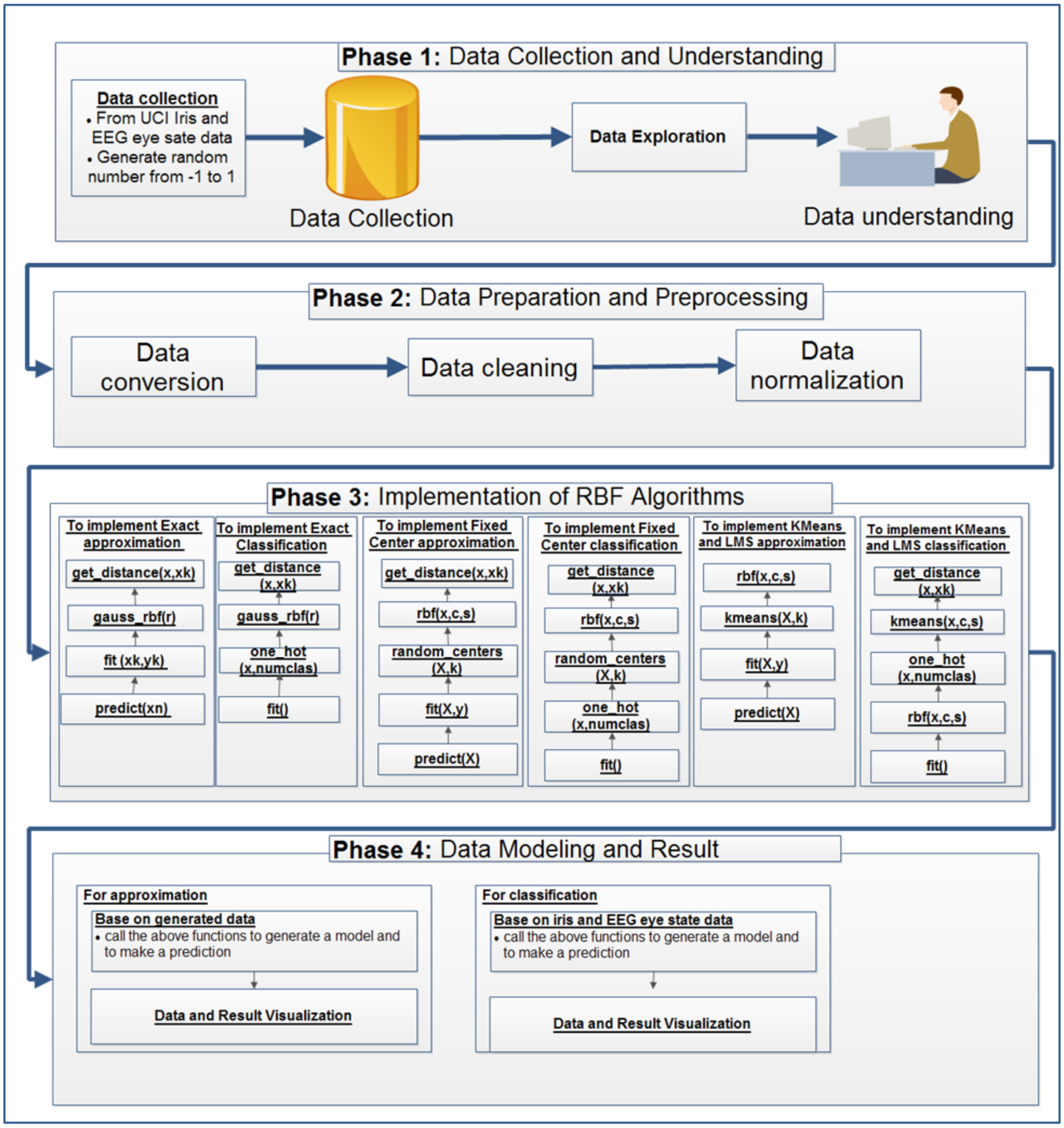

Phase 1: Data Collection and Understanding
Data collection
• From UCI Iris and EEG eye sate data
• Generate random number from -1 to 1
Data Collection
Data Exploration
Data understanding
Phase 2: Data Preparation and Preprocessing
Data conversion
Data cleaning
Data normalization
Phase 3: Implementation of RBF Algorithms
To implement Exact approximation
get_distance(x,xk)
gauss_rbf(r)
fit (xk,yk)
predict(xn)
To implement Exact Classification
get_distance (x,xk)
gauss_rbf(r)
one_hot (x,numclas)
fit()
To implement Fixed Center approximation
get_distance(x,xk)
rbf(x,c,s)
random_centers (X,k)
fit(X,y)
predict(X)
To implement Fixed Center classification
get_distance (x,xk)
rbf(x,c,s)
random_centers (X,k)
one_hot (x,numclas)
fit()
To implement KMeans and LMS approximation
rbf(x,c,s)
kmeans(X,k)
fit(X,y)
predict(X)
To implement KMeans and LMS classification
get_distance (x,xk)
kmeans(x,c,s)
one_hot (x,numclas)
rbf(x,c,s)
fit()
Phase 4: Data Modeling and Result
For approximation
Base on generated data
• call the above functions to generate a model and to make a prediction
Data and Result Visualization
For classification
Base on iris and EEG eye state data
• call the above functions to generate a model and to make a prediction
Data and Result Visualization

Figure 3: Methodology

2.1. Data Collection and Exploration

For classification problems we collected the data for Iris Dataset and EEG Eye State Dataset from the University of California Irvine (UCI), Machine Learning Repository. For the sake of readability we converted the data into a csv file and began the exploration. While exploring the iris dataset and EEG Eye state dataset, we tried to check out how the target variables are categorized, the correlation of the data and the data distribution before the RBF algorithms are applied. For the approximation problem we generated 100 random sampling points in the range of -1 to 1 for the training set.

2.2. Data Preparation and Preprocessing

For classification problems, to make the data suitable for a machine learning algorithm , on iris dataset we need to remove the target variable, therefore first we converted the 'target' feature which holds the flower type to a numerical data and stored the labels into a separate variable. On the EEG eye state dataset we first stored the target variable 'eyeDetection' labels into a separate variable and then removed it. Then we normalized the dataset on the pre-processing step.
For the exact classification problem, to convert the iris dataset to a one dimension, we used PCA and made a dimensionality reduction. For the approximation problem, based on the generated training set we generated the labels value using the following function.

x2-x-cos(pi*x)**

2.3. Implementation of RBF Algorithms

2.3.1. Exact approach for approximation and classification

To train and test the RBF that is implemented using the Exact Interpolation approach we calculated gaussian RBF with standard deviation.

$$\varphi(r)=\exp\left(-\frac{r^2}{2\sigma^2}\right)$$

To calculate the distance between the data points and center we used
get_distance(x,xk)

A Gaussian RBF to scale, and to approximate given functions
gauss_rbf(r)

To get the RBF of unknown data point x with respect to all centroids and to calculate the RBF and yk to get W we used
fit (xk, yk)

To get the RBF of unknown data point xn with respect to all centroids and predict the output by calculating the dot product of RBF and W we used
predict(xn)

2.3.2. Fixed center at random for approximation and classification

The fixed centers approach is simple and fast. The number of hidden units is less than the number of input patterns H < N.

To train and test the RBF implemented using the Fixed Centers Selected at Random approach the number of centers is set to 2 for approximation and 10 for classification. The Radial Basis Function (RBF) is:

$$\varphi_i(x) = \exp\left(-\frac{\|x - \mu_i\|^2}{2\sigma_i^2} \right)$$

where mu_i is the i-th fixed center and sigma_i is:

$$\sigma_i = \frac{d_{\max}}{\sqrt{2M}}$$

where d_max is the maximum distance between the chosen centers.

2.3.3. RBF using K-Means and LMS for approximation and classification

For classification problems one of the main advantages of RBF is the utilization of the Least Squares Linear Regression equation in which obtaining a global minimum of the cost function is relatively fast and guaranteed. On the other hand, other optimization algorithms such as Batch Gradient Descent can also be applied to update weights.

In the iris and EEG eye state classification problem:

- X is the 2-dimensional matrix of RBFs
- y is a one-hot-encoded 2-dimensional matrix.

The prediction of the class of the unknown point can be obtained as follow:

1. Get RBF of an unknown data point x with respect to all centroids.
2. Calculate dot product of RBF and W and select an index of the maximum value.

For the function approximation problems we used k-means clustering on our input data to figure out where to place the Gaussians. K-means clustering is used to determine the centers c_j for each of the radial basis functions φ_j. Given an input x an RBF network produces a weighted sum output.

$$F(x) = \sum_{j=1}^{k} w_j \varphi_j(x, c_j) + b$$

where w_j are the weights, b is the bias, k is the number of bases/clusters/centers, and $\varphi_j(\cdot)$ is the Gaussian RBF:

$$\varphi_j(x, c_j) = \exp\left(\frac{-||x - c_j||^2}{2\sigma_j^2}\right)$$

There are other kinds of RBFs, but we will stick with the Gaussian RBF.

Using these definitions, we can derive the update rules for w_j and b for gradient descent. We use the quadratic cost function to minimize.

$$C = \sum_{i=1}^{N} (y^{(i)} - F(x^{(i)}))^2$$

We can derive the update rule for w_j by computing the partial derivative of the cost function with respect to all of the w_j.

$$
\begin{aligned}
\frac{\partial C}{\partial w_j} &= \frac{\partial C}{\partial F}\frac{\partial F}{\partial w_j} \\
&= \frac{\partial}{\partial F}[\sum_{i=1}^{N}(y^{(i)} - F(x^{(i)}))^2] \ \cdot \ \frac{\partial}{\partial w_j}[\sum_{j=0}^{K} w_j \varphi_j(x, c_j) + b] \\
&= -(y^{(i)} - F(x^{(i)})) \ \cdot \ \varphi_j(x, c_j) \\
w_j &\leftarrow w_j + \eta \ (y^{(i)} - F(x^{(i)})) \ \varphi_j(x, c_j)
\end{aligned}
$$

Similarly, we can derive the update rules for b by computing the partial derivative of the cost function with respect to b.

$$
\begin{aligned}
\frac{\partial C}{\partial b} &= \frac{\partial C}{\partial F}\frac{\partial F}{\partial b} \\
&= \frac{\partial}{\partial F}[\sum_{i=1}^{N}(y^{(i)} - F(x^{(i)}))^2] \ \cdot \ \frac{\partial}{\partial b}[\sum_{j=0}^{K} w_j \varphi_j(x, c_j) + b] \\
&= -(y^{(i)} - F(x^{(i)})) \cdot 1 \\
b &\leftarrow b + \eta \ (y^{(i)} - F(x^{(i)}))
\end{aligned}
$$

2.4. Model Building

Based on the generated data, iris and EEG eye state dataset we trained a model and made a prediction and got the following results.

3. Results

3.1. Exact RBF approach

On the exact function approximation problem we generated 100 random sampling points in the range of -1 to 1 for the training set. The following figure compares our function approximation result with that of the python's built in function:

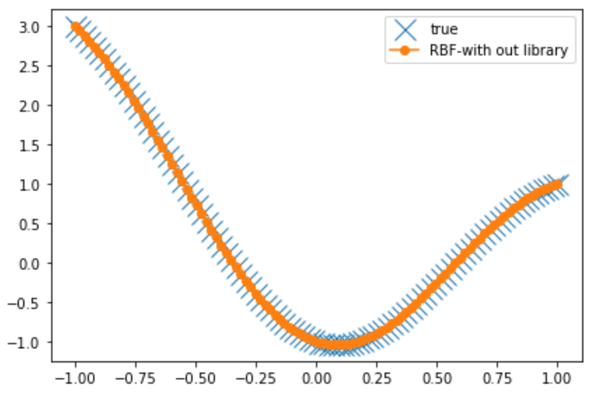


Figure 4: Exact RBF

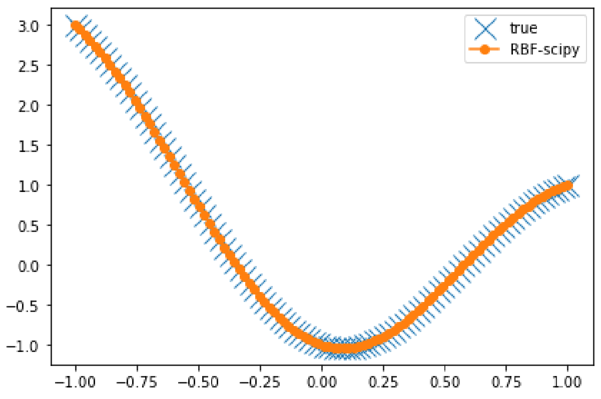


Figure 5: Exact RBF in python's built function

On the classification problem with the iris dataset. we splitted the test and training data and based on the training data we made the prediction. The following figure compares our classification result with that of the python's built in function:

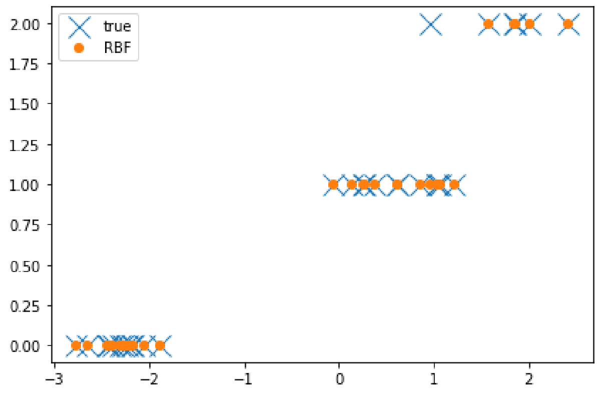


Figure 3: Exact RBF classification

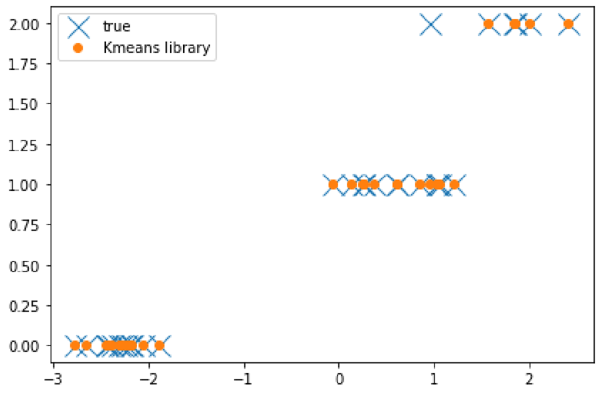


Figure 4: Exact RBF classification built function

3.2. Fixed Centers selected at random

The following figure compares our function approximation result with that of the python's built in function for the fixed centers approach:

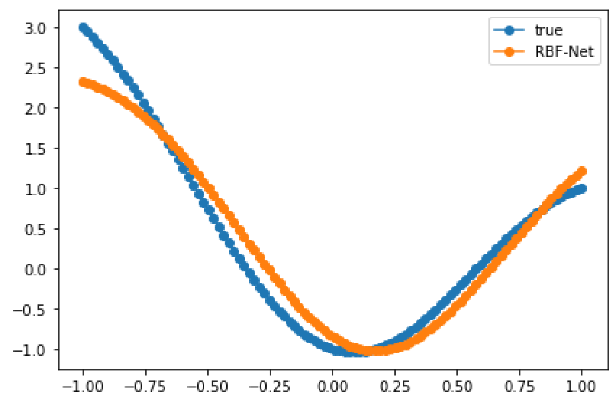


Figure 5: Fixed RBF approximation

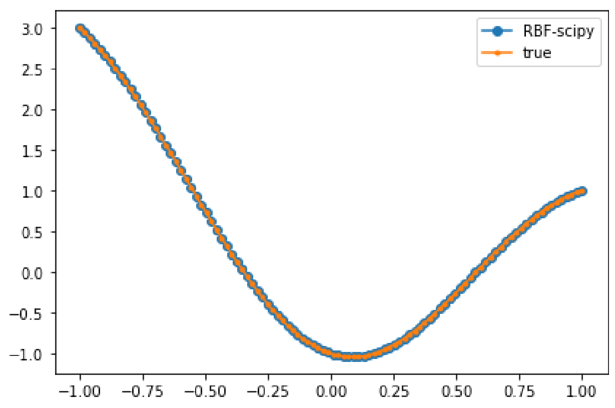


Figure 6: Fixed RBF approximation built function

The following figure compares our classification result with that of the python's built in function for the fixed centers approach:

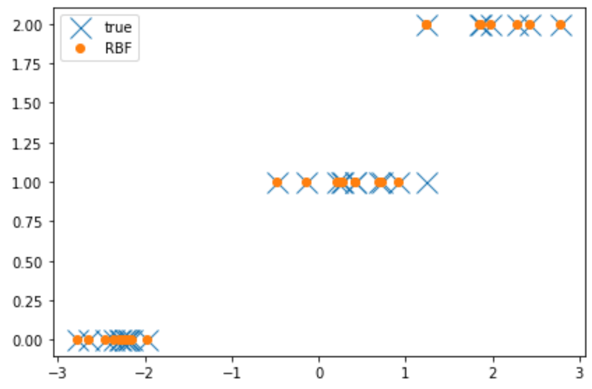


Figure 7: Fixed RBF classification

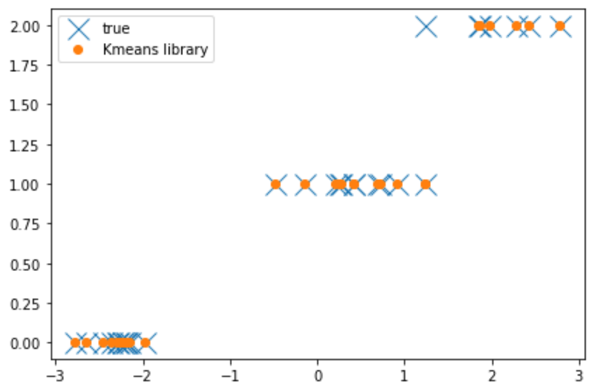


Figure 8: Fixed RBF classification built function

3.3. Unsupervised Learning (Clustering)

After implementing this approach for both function approximation and classification problems, we found the following results:

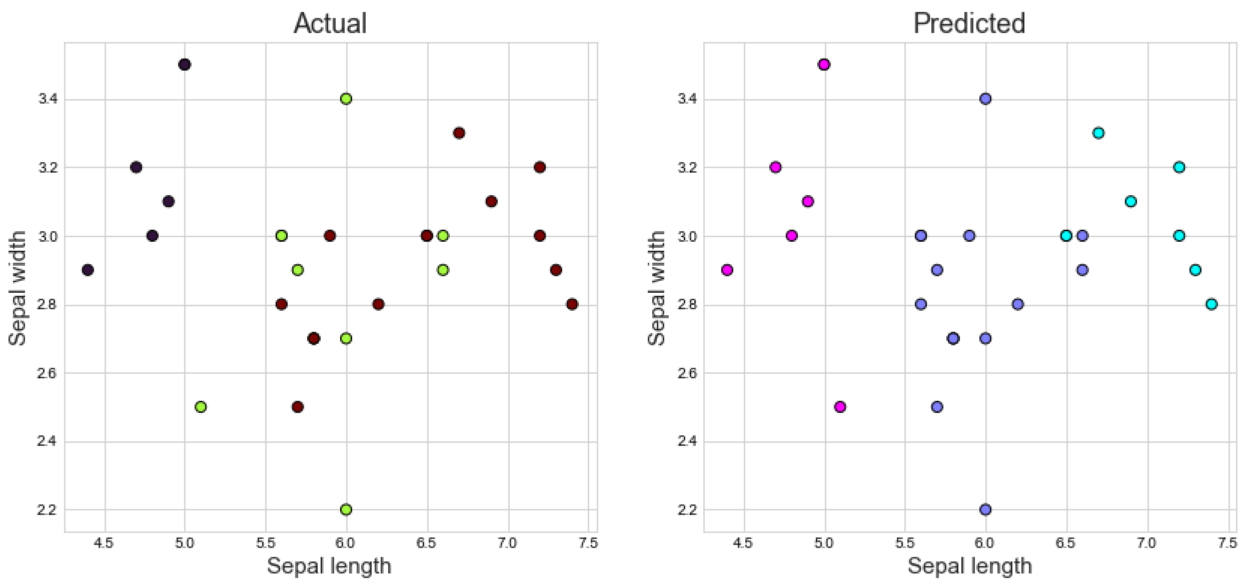


Figure 9: K-means, LMS RBF classification

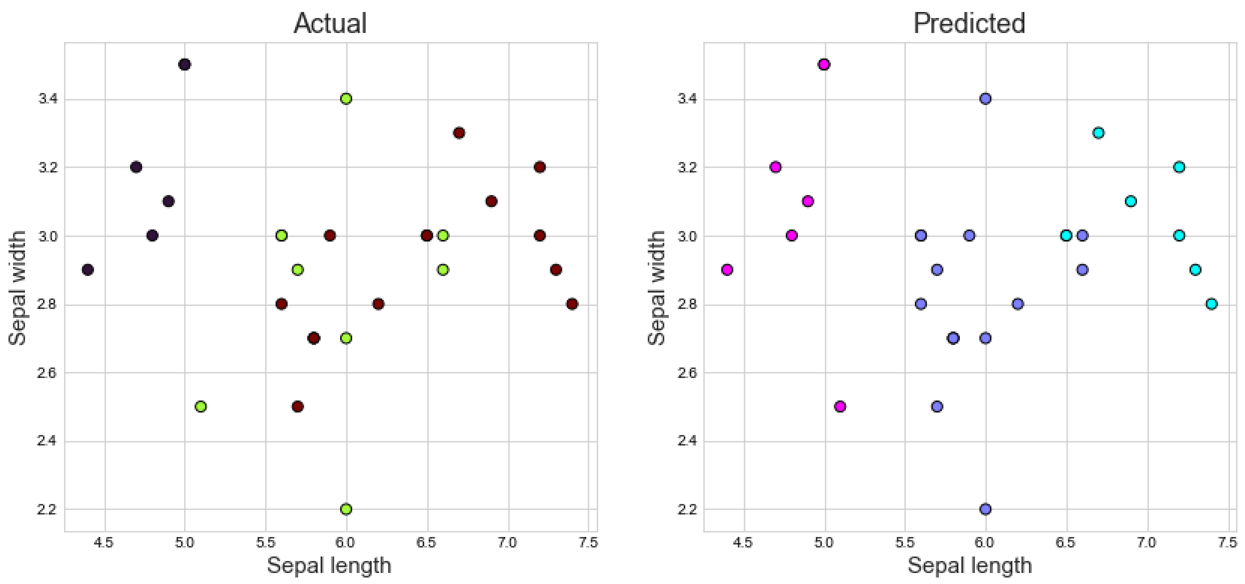
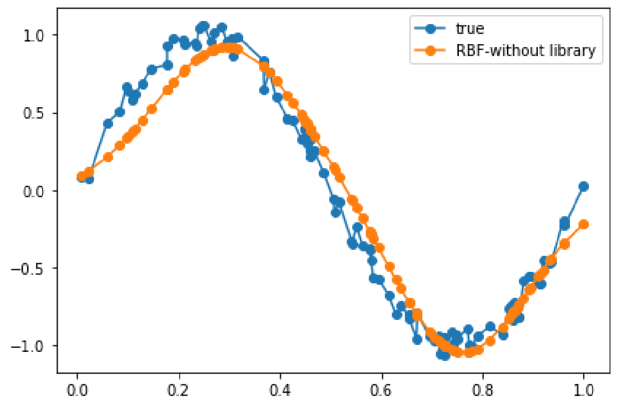


Figure 10: K-means, LMS RBF classification with built in



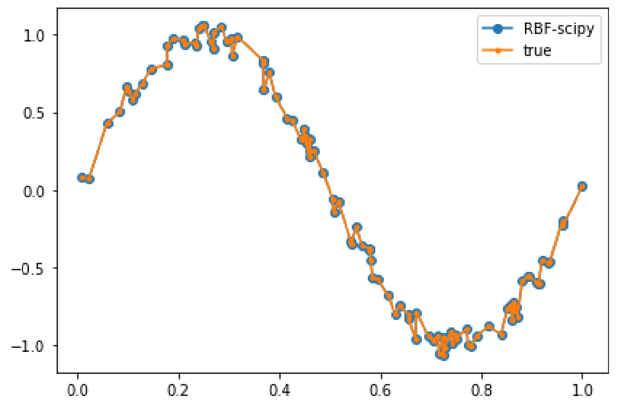


Figure 11: K-means, LMS RBF approximation *Figure 12: K-means, LMS RBF approximation builtin*

3.4. Implementing in EEG Eye State Data Set

After applying our algorithms to the EEG Eye State Data Set, we found the following results:

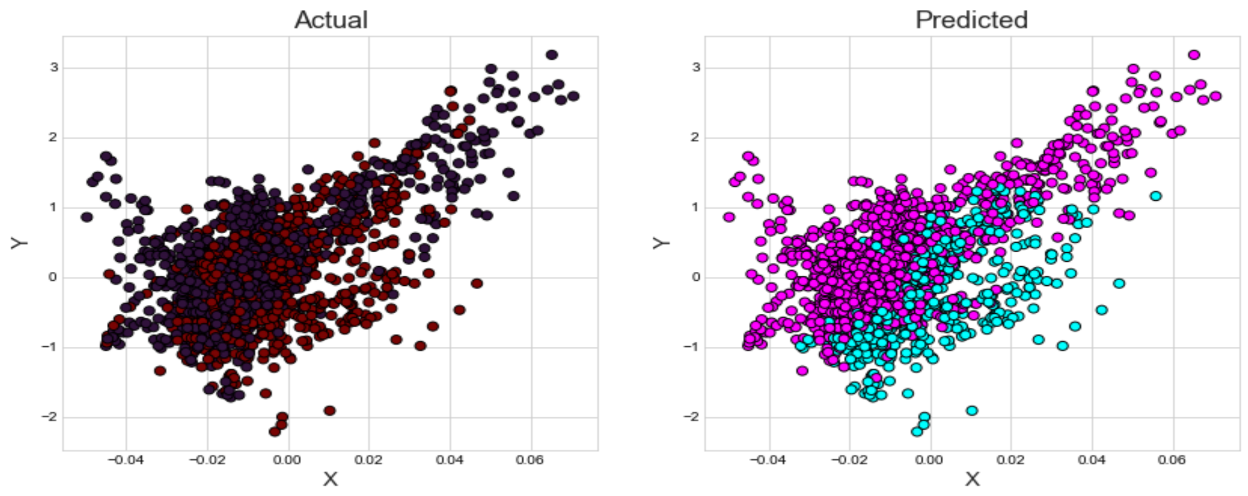


Figure 11: EEG Eye State Data Set RBF

4. Conclusion

We have implemented the RBF in Exact, Fixed center selected at random and K-means and LSM algorithms for both classification and function approximation problems from scratch. We used the Iris and EEG Eye State Datasets and we tried to compare our results with that of the pythons builtin library. We have learned a lot throughout the implementation process.

References

https://en.wikipedia.org/wiki/Radial_basis_function_network
https://en.wikipedia.org/wiki/Least_mean_squares_filter
https://www.hindawi.com/journals/isrn/2012/324194/
https://www.sciencedirect.com/topics/engineering/radial-basis-function-network
https://www.cs.bham.ac.uk/~jxb/INC/l13.pdf
https://www.cc.gatech.edu/~isbell/tutorials/rbf-intro.pdf
https://archive.ics.uci.edu/ml/datasets/EEG+Eye+State#
https://archive.ics.uci.edu/ml/datasets/iris