[COMP6247] Lab Two Report

Feng Xie — 30502322 — fx1n18@soton.ac.uk

1 Introduction

The aim of this lab is to get familiar with function approximation using Radial Basis Functions(RBF) which consists of nonlinear basis functions and learnable weights.

2 Part one

In this part, we solved a regression problem with RBF using the dataset *housing.data* and compare the result with a linear model.

2.1 Methodology

We first applied a linear model to the dataset *housing.data* by calculating the pseudo inverse to get the optimal weights and plotted a figure about the true values and predicted values using the optimal weights. To build a RBF model, KMeans clustering was used to find where to place the basis function. The number of basis functions was equal to the number of clusters. The Gaussian was used as the basis function. The standard deviation was derived using the entire input data. After the clustering, the input data were mapped to the RBF model and the optimal weights were computed using the pseudo inverse

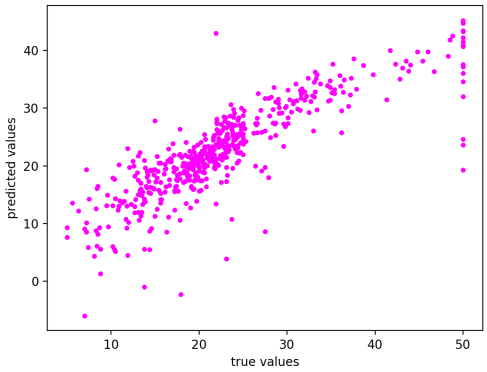
2.2 Results

Pseudo inverse was used to compute the optimal solution for the linear and RBF model and the result is shown in the figure 1. We can see from the figure that when we use 50 basis functions to build the RBF model, the performance is close to the linear model.

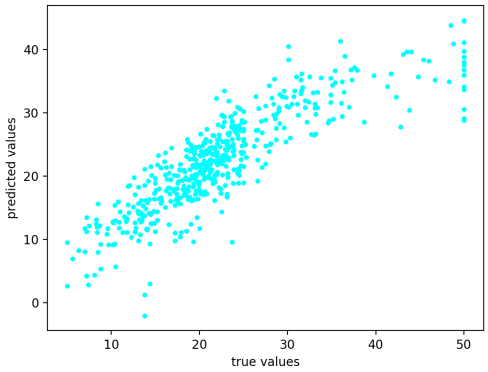
In fact, when the number of basis functions increases, the RBF model can usually work out a better solution. But overfitting is likely to occur if we use too many basis functions. In the figure 2, the orange line represents the error for the best solution of the linear model. We can find that when more than 50 basis functions were used, the MSE for the RBF model was lower than the linear model and it continued to decrease, which indicates that overfitting occurred.

3 Part Two

In this part, we solved the problem above using stochastic gradient descent.



(a) Linear Model



(b) RBF model with 50 basis functions

Figure 1: Results for different models

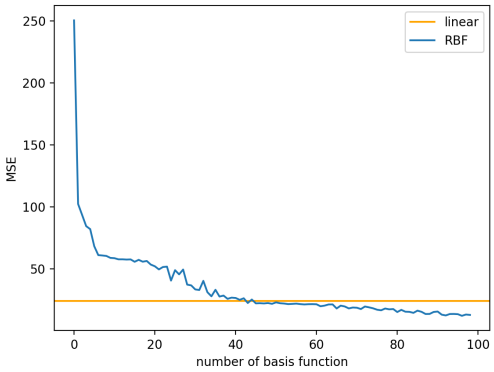


Figure 2: Mean Square Error for different numbers of basis functions

3.1 Methodology

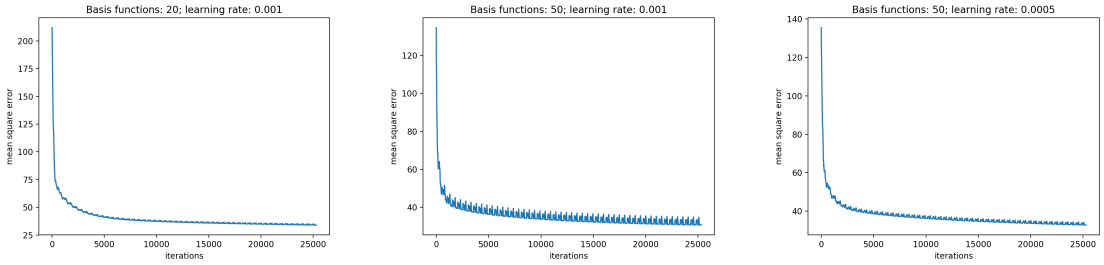
Stochastic gradient descent(SGD) was applied for learning the weights of the RBF model. The weights were updated on each data sample and the mean square error was computed and stored after the update to check if the error converges over iterations.

3.2 Results

It can be seen from the figure 3 that the mean square error decreased fast in the initial iterations and converged. Besides, when the number of basis functions or the learning late was reduced, the fluctuation range of mean square error became smaller.

4 Part Three

In this part, function approximation was applied in the Mountain Car problem using the RBF model.



(a) (b) (c)

Figure 3: Mean Square Error over iterations

4.1 Methodology

4.1.1 Approximate the learned action values

RBF models were built with different numbers of basis functions to approximate the action values that were learned in *Lab One*. And the mean square error was computed after the SGD(stochastic gradient descent) training to measure the accuracy of approximation.

When testing the performance of approximation, the environment of the Mountain Car problem was reset and for each observation, we did the discretization as what we did in *Lab One* and chose the action according the new policy which was derived from the approximation.

4.1.2 Online approximation using Q-Learning

When doing the online RBF approximation, the discretization for the states is not necessary. However, in the Mountain Car problem, since the ranges of positions and velocities are not the same, they were scaled before use. In order to apply KMeans clustering to find the representative vectors of input data, 10,000 observations were sampled from the environment. Before the clustering, the samples were scaled by removing the mean and scaling to unit variance. In the later testing, each observation was scaled in the same way.

The Q-Learning approximation was run for 100 episodes and each episode ended when the car was able to reach the goal state. The greedy policy was applied to choose the action. Because when the car commits an action, it will receive only one state(i.e., the next state), stochastic gradient descent was applied to learn the weights.

To compare the results of approximation using tabular and online methods, an analysis will be made in the following part.

4.2 Results

4.2.1 Approximate the learned action values

From the figure 4, we can see that as the number of basis functions increased, the mean square error dropped dramatically and started to converge, which means the accuracy of approximation increased and got steady as well.

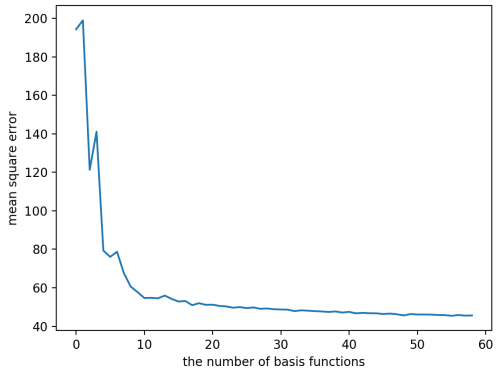


Figure 4: Approximation accuracy

4.2.2 Online approximation using Q-Learning

To compare the tabular method and the online method, the total steps for reaching the goal position after training were used to measure the difference. To do this, a random seed was used to ensure the initial position was the same when the environment was reset. Twenty basis functions were used for both methods. It turned out that it took 109 steps for the online method to reach the goal while it only took 84 steps for the tabular method under the same conditions. This indicates that in the Mountain Car problem, using the tabular method is likely to get a more efficient solution.

However, in terms of the space used, apparently, the tabular method needs to store all the values of a whole table and the size of the table will become much larger as the state space increases. For the online method, when we apply a RBF approximator, it only needs to save the learned weights for later testings, which can save a lot of space.

5 Conclusion

Function approximation is very helpful in reinforcement learning to significantly reduce the space that is needed for working out a solution.

References

[1] https://openai.com/

[2] https://github.com/openai/gym/wiki/MountainCar-v0

[3] https://pythonmachinelearning.pro/using-neural-networks-for-regression-radial-basis-function-networks/

[4] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[5] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html