Function Approximation

1: MLP approximation:

Algorithm: MLP with gradient descent, error function Least square

Activation Function: 1: Sigmoid activation function in hidden layer
2. No activation in output layer

Loss function: Least square

Parameters to tune:

- Learning rate
- Number of hidden units

Observations and Analysis:

For cross validation, I have used 90% training data as training set and remaining 10% for testing. Analysis is given with the graphs:

Optimal parameters:

mg85
hid - 5
epoch - 600
learning rate - 1e-4
Training Error - 0.22097608
Testing Error - 0.54073818

bj
hid - 4
epoch - 400
learning rate - 1e-2
Training Error - 0.020174
Testing Error - 0.03007362

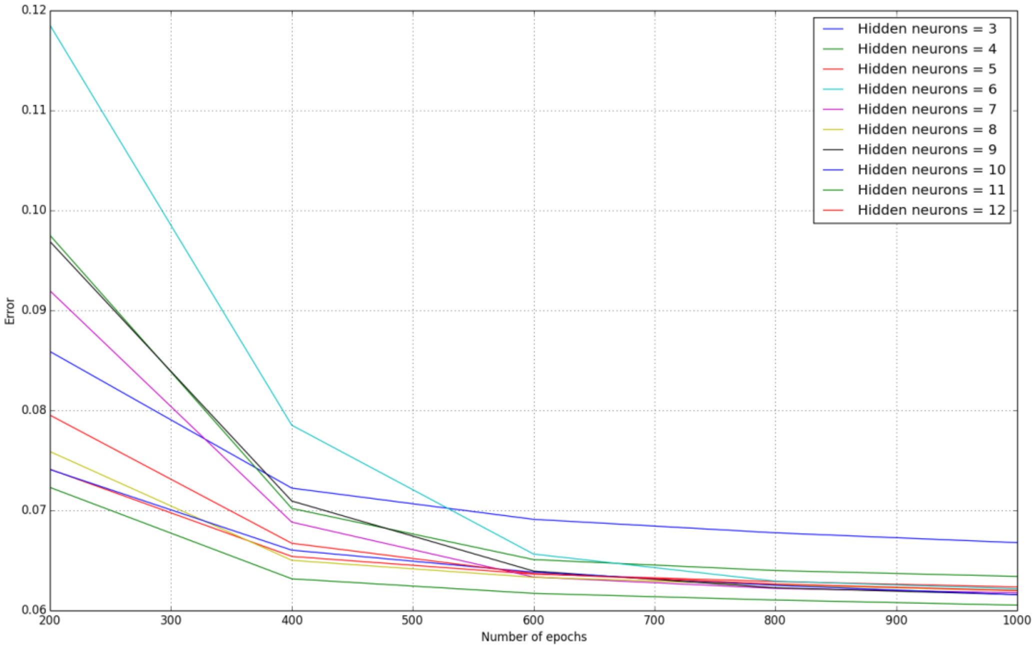
SI
hid - 5
epoch - 600
learning rate - 1e-5
Training Error - 0.06300885
Testing Error - 0.98317534

Graphs:

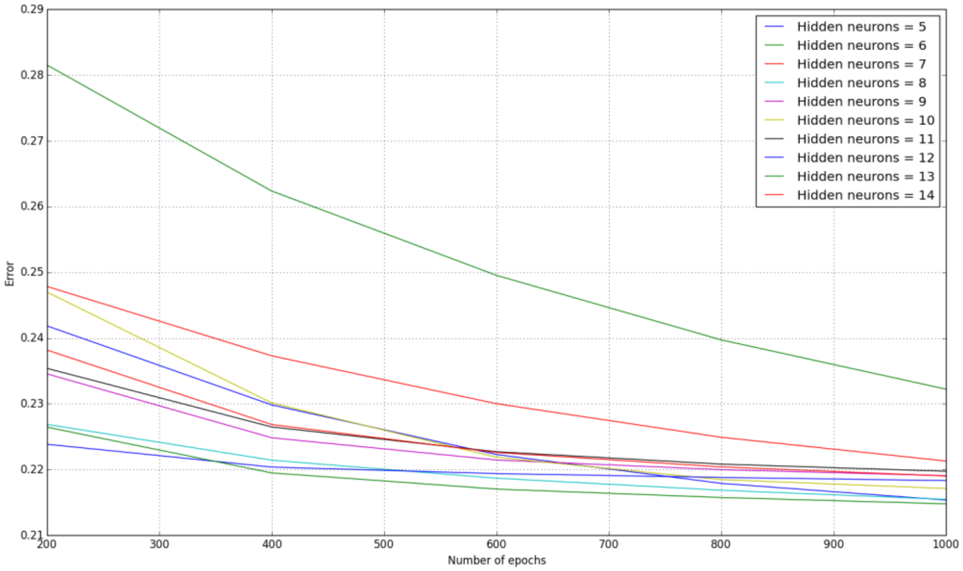
As number of hidden units increases the error is decreasing. But the decrease in error is not significant after a certain number of units (On the test data). After sometime it may be increase on test data.

Number of epochs: The main problem in gradient descent algorithm is that it is susceptible to local optima. It may get stuck on an optimum that may not be even close to the global optima. Here random initialization becomes savior. We have to run the code for 4-5 times and then we saved the best one.

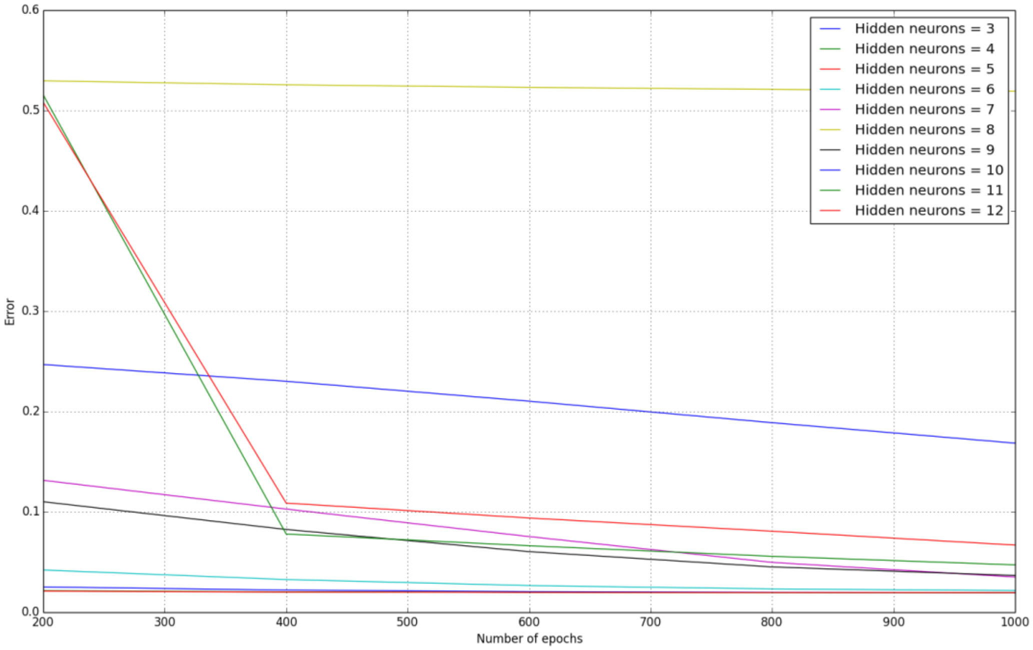
SI:



MG85:


Error
0.29
0.28
0.27
0.26
0.25
0.24
0.23
0.22
0.21
200
300
400
500
600
700
800
900
1000
Number of epochs
Hidden neurons = 5
Hidden neurons = 6
Hidden neurons = 7
Hidden neurons = 8
Hidden neurons = 9
Hidden neurons = 10
Hidden neurons = 11
Hidden neurons = 12
Hidden neurons = 13
Hidden neurons = 14

BJ:


Error
0.6
0.5
0.4
0.3
0.2
0.1
0.0
200
300
400
500
600
700
800
900
1000
Number of epochs
Hidden neurons = 3
Hidden neurons = 4
Hidden neurons = 5
Hidden neurons = 6
Hidden neurons = 7
Hidden neurons = 8
Hidden neurons = 9
Hidden neurons = 10
Hidden neurons = 11
Hidden neurons = 12

2: RBF approximation:

Algorithm: RBF with Pseudo Inverse with clustering and gradient descent different version.

Activation Function: 1: RBF kernel in the first layer

Loss function: No need

Parameters to tune:

- Number of Clusters

Algorithm:

Step:1 Normalized the features
Step:2 K-means Clustering, sigma initialized to a random number near 0.5
Step:3 Use pseudo inverse to learn weights in one-shot.

Observations and Analysis:

For cross validation I have used 90% training data as training set and remaining 10% for testing.

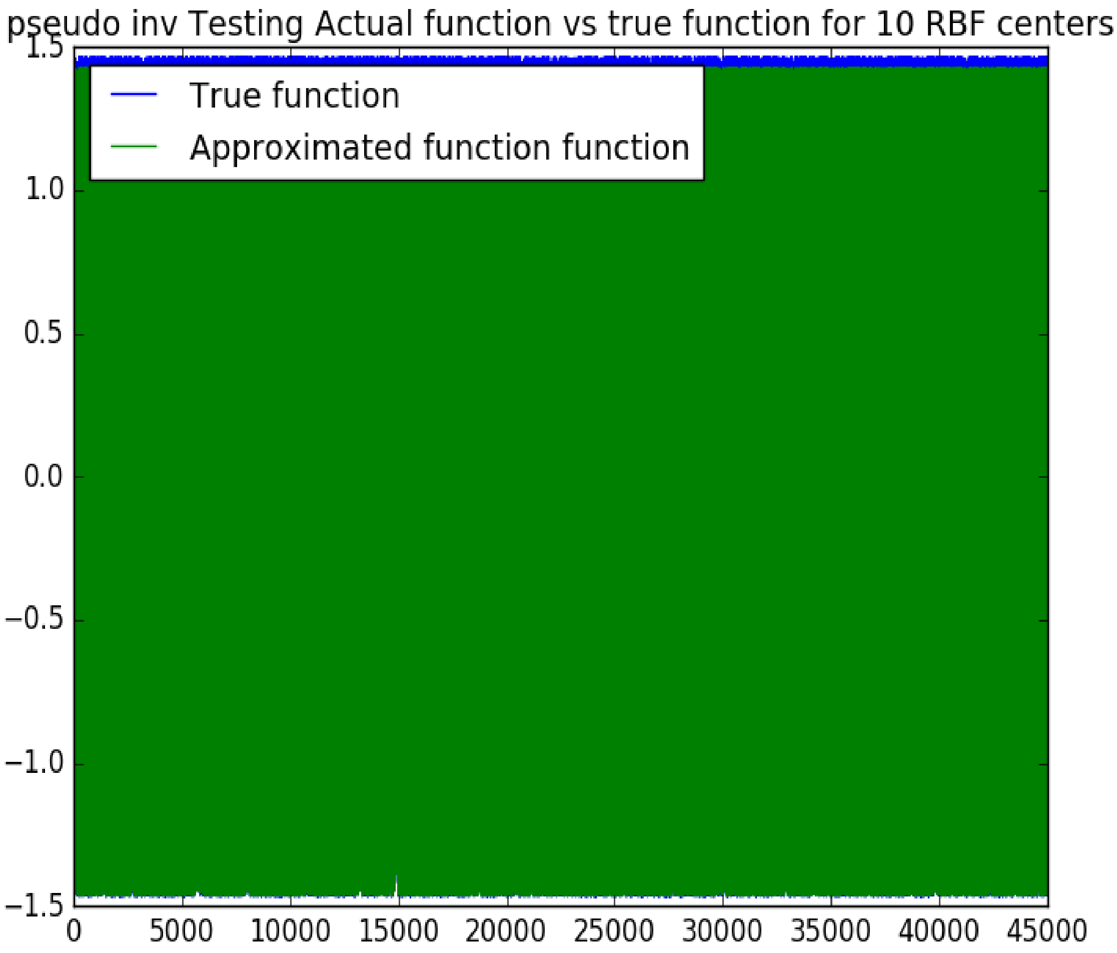
- Number of cluster centers
  - Number of cluster centers K was first initialized to (No. features)*2
  - Then I increased the K as long as I got higher accuracy on **validation data.** I have used a greedy technique in which for more accuracy I have increased K as much as I can. The stopping criteria for learning is **(Training error- validation error <= 0.25) .**
  - For better accuracy we can do more computations. Here main problem is that we are trying to invert a matrix. Taking inverse of a matrix is computationally very expensive algorithm. So I have increased K but not too much.
  - That’s why number of clusters are more than the thumb rule we have discussed in class.
- After that for visualization purpose we have plotted error vs epochs (A different version of RBF approx which is based on gradient descent)
- Other technique to plot the True function and the approximated function in same plot and see the actual fitting.

How am I choosing the parameters and Plots:

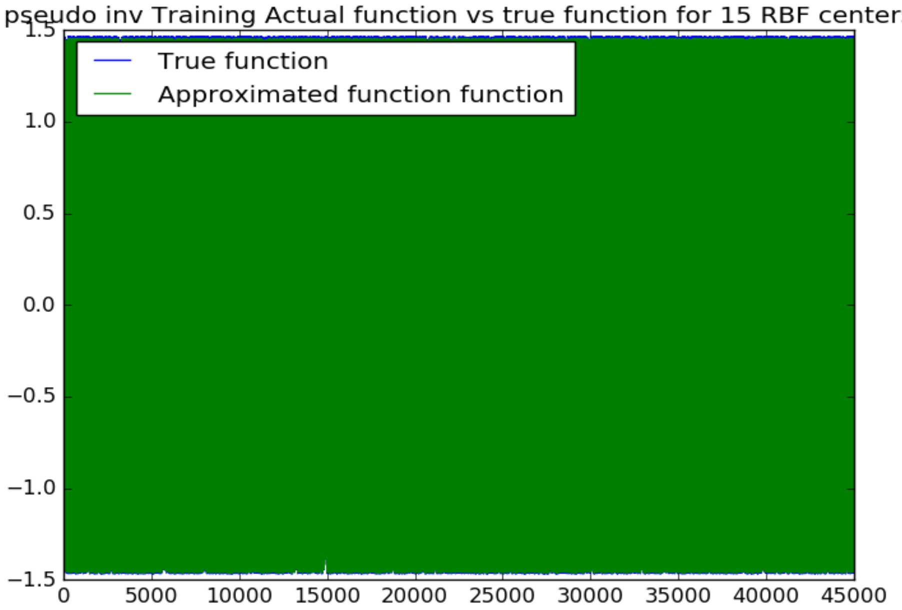
SI dataset: SI dataset is a very simple function to approximate. The data is given in its normalized form and function is not very steeply changing.

DATASET	SI	
Number of clusters	**Train**	**test**
5	-	-
10	0.022456	0.022455
20	0.015076	0.015155
40	0.01073	0.010792

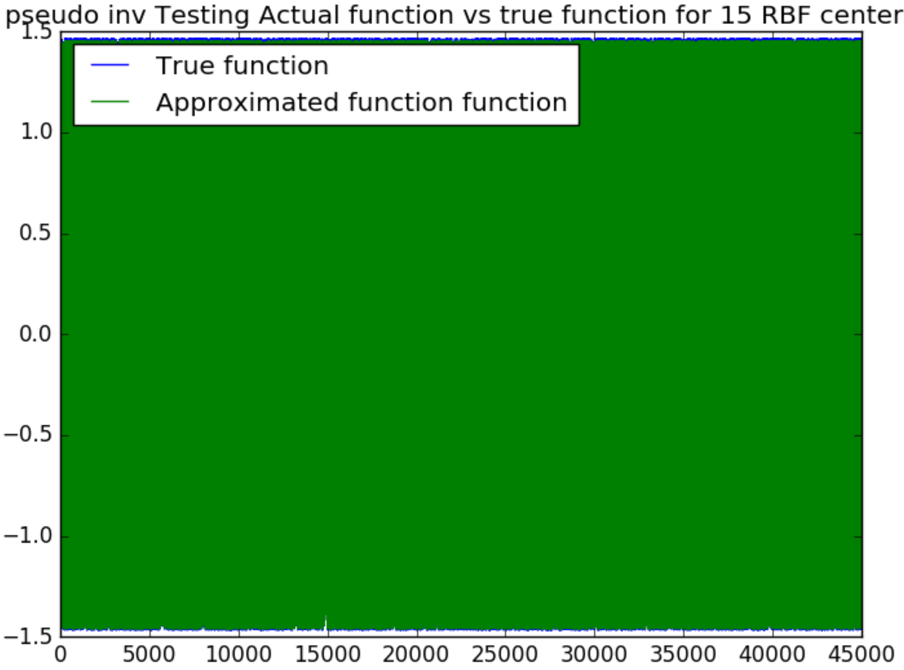
Using 10 clusters: Here you can see that this model so accurately approximates the function. The green (approximated) function is totally overlaps the True function. (On Test data set.)



After Increasing number of RBF clusters: The error decreased by a very small amount. But it is accurately approximates the function.



pseudo inv Training Actual function vs true function for 15 RBF centers

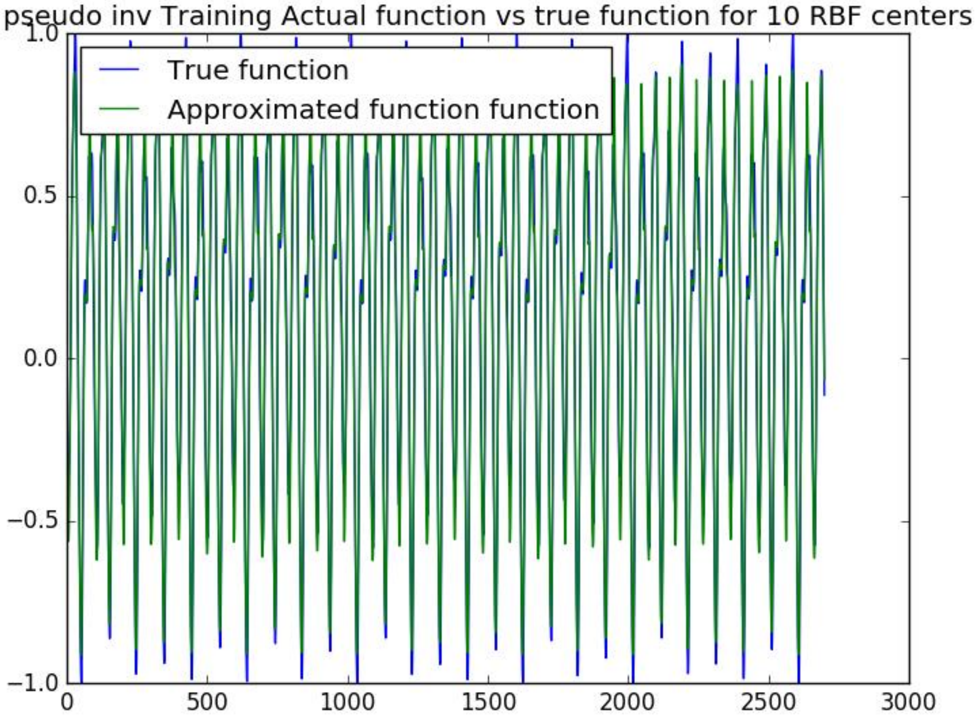


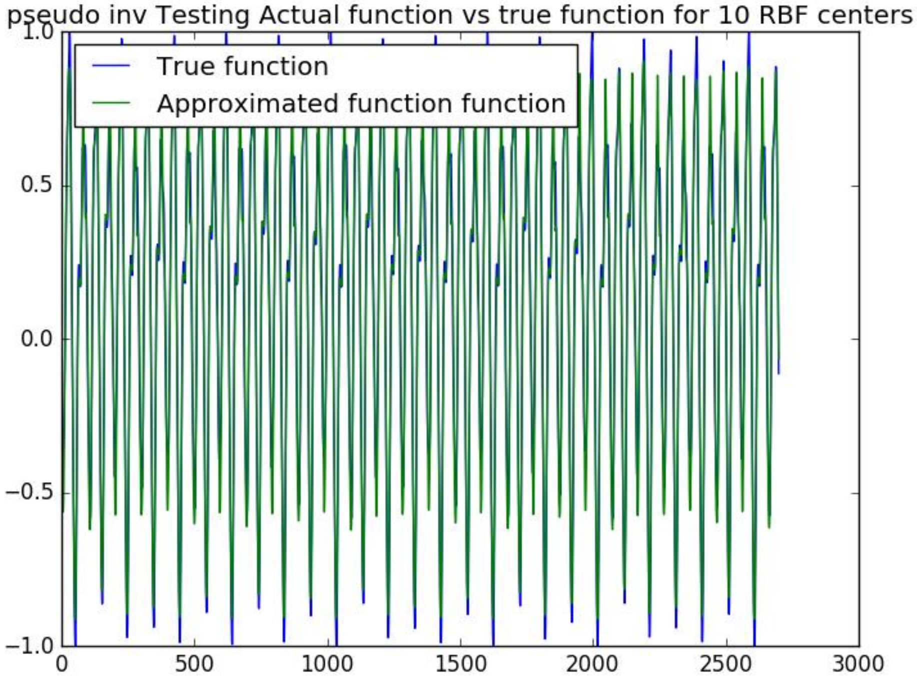
pseudo inv Testing Actual function vs true function for 15 RBF centers

MG85 dataset:

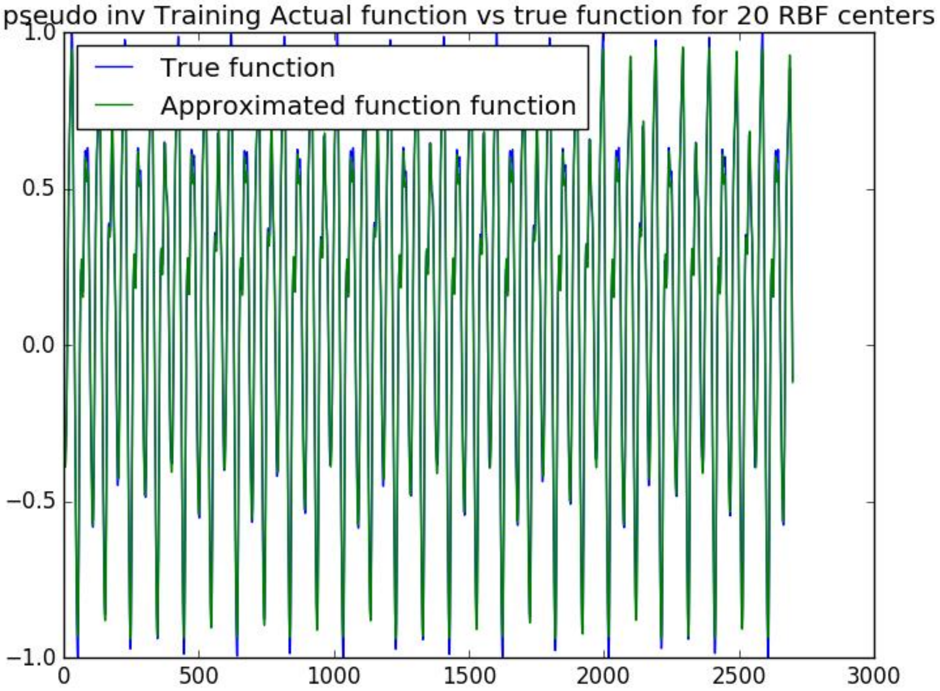
DATASET	mg85	
Number of clusters	train	test
5	-	-
10	0.092433	0.093222
20	0.042252	0.043253
40	0.027819	0.029408

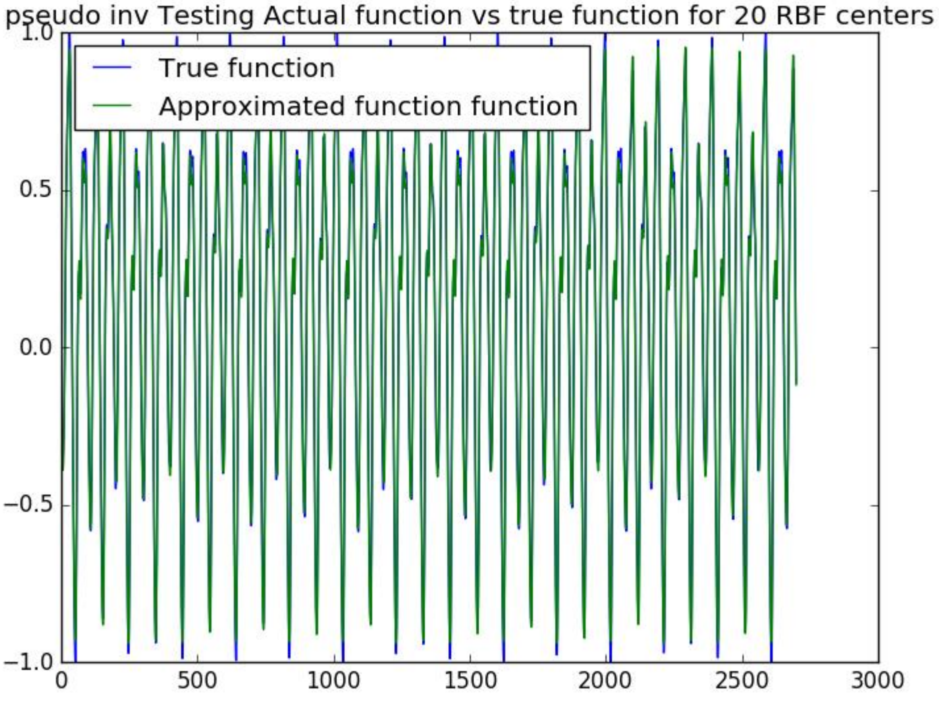
10 neurons:


pseudo inv Training Actual function vs true function for 10 RBF centers
True function
Approximated function function
1.0
0.5
0.0
−0.5
−1.0
0
500
1000
1500
2000
2500
3000

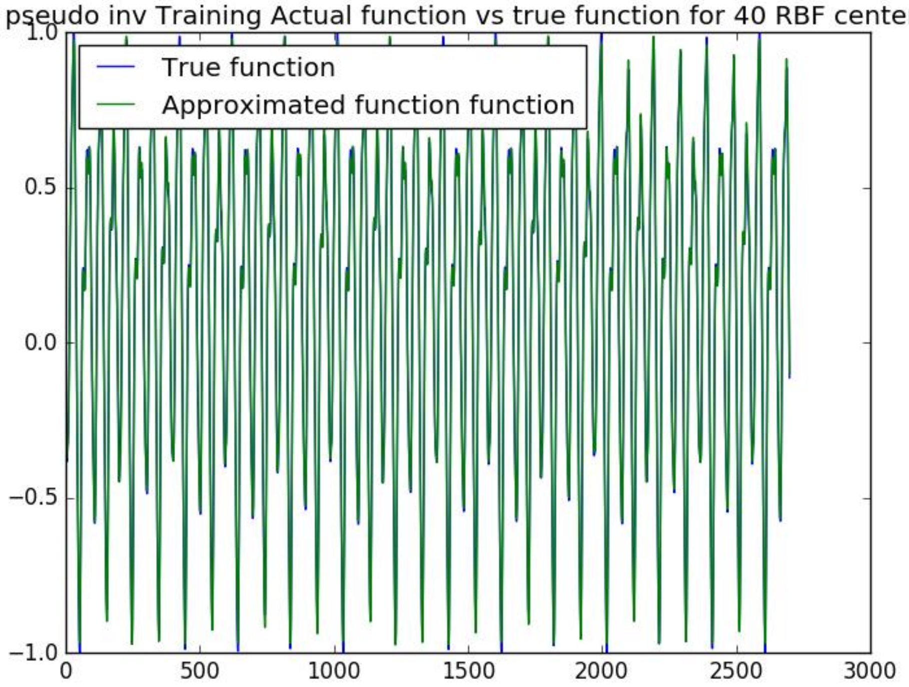

pseudo inv Testing Actual function vs true function for 10 RBF centers
True function
Approximated function function
1.0
0.5
0.0
−0.5
−1.0
0
500
1000
1500
2000
2500
3000

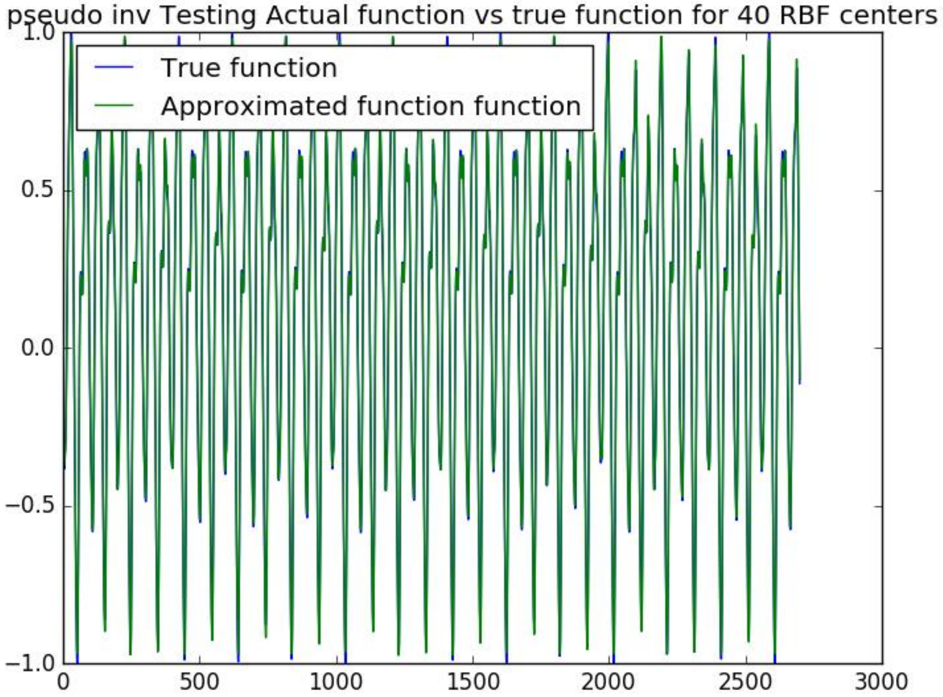
20 neurons:


pseudo inv Training Actual function vs true function for 20 RBF centers
True function
Approximated function function
1.0
0.5
0.0
−0.5
−1.0
0
500
1000
1500
2000
2500
3000


pseudo inv Testing Actual function vs true function for 20 RBF centers
True function
Approximated function function
1.0
0.5
0.0
−0.5
−1.0
0
500
1000
1500
2000
2500
3000

40 neurons:

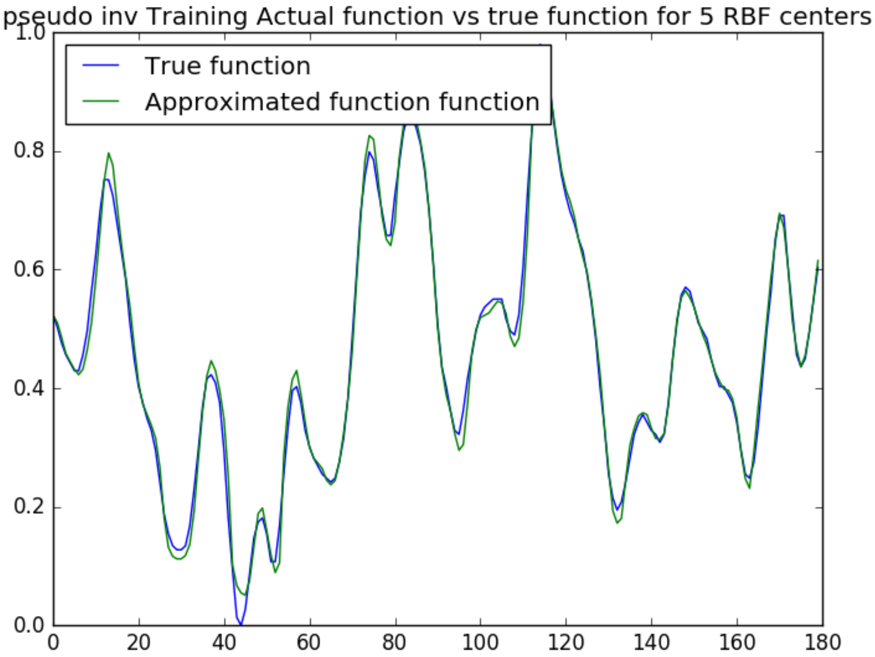

pseudo inv Training Actual function vs true function for 40 RBF centers
True function
Approximated function function
1.0
0.5
0.0
−0.5
−1.0
0
500
1000
1500
2000
2500
3000

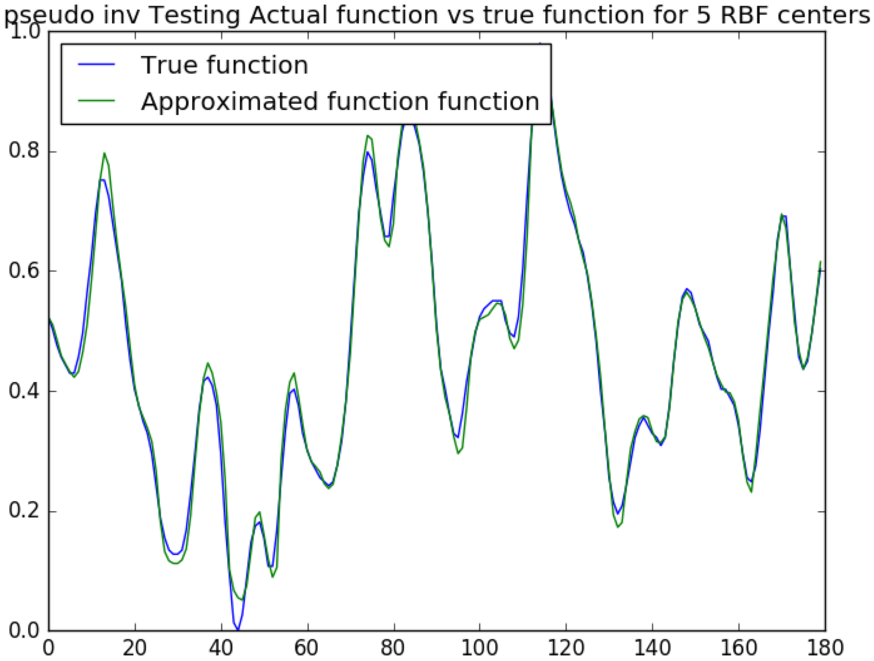

pseudo inv Testing Actual function vs true function for 40 RBF centers
True function
Approximated function function
1.0
0.5
0.0
−0.5
−1.0
0
500
1000
1500
2000
2500
3000

BJ dataset:

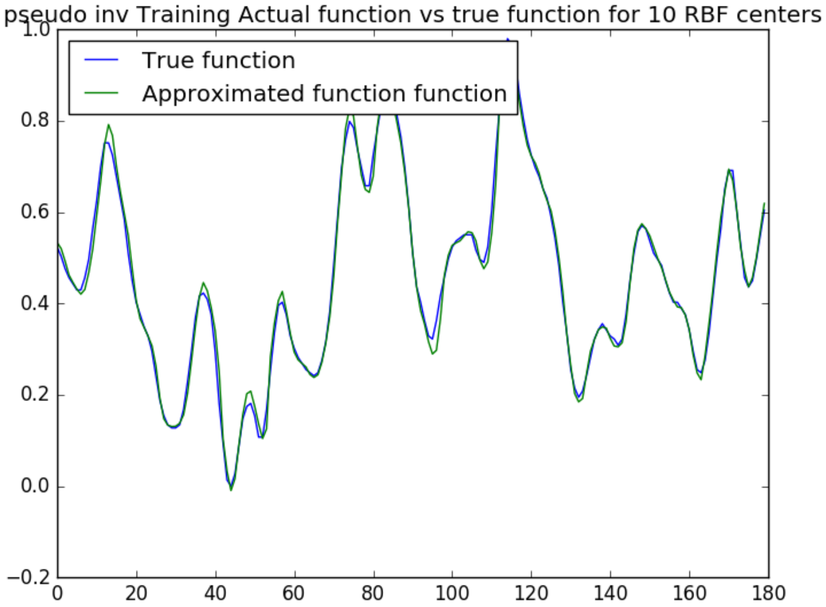
DATASET		bj
Number of clusters	train	test
5	0.021595	0.203708
10	0.01893	0.206543
20	0.016416	0.198388
40	0.015671	0.213052

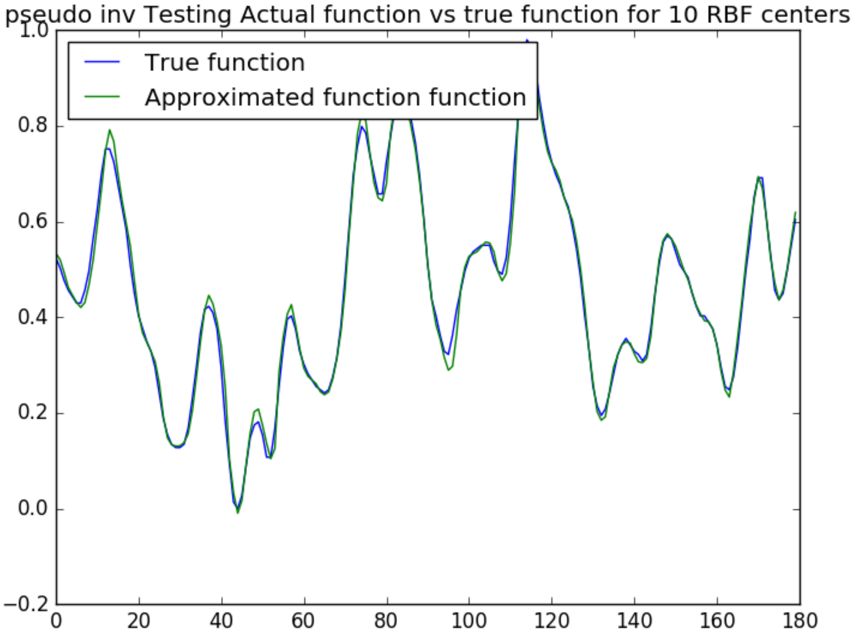
5 Neurons:


pseudo inv Training Actual function vs true function for 5 RBF centers
True function
Approximated function function
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100
120
140
160
180

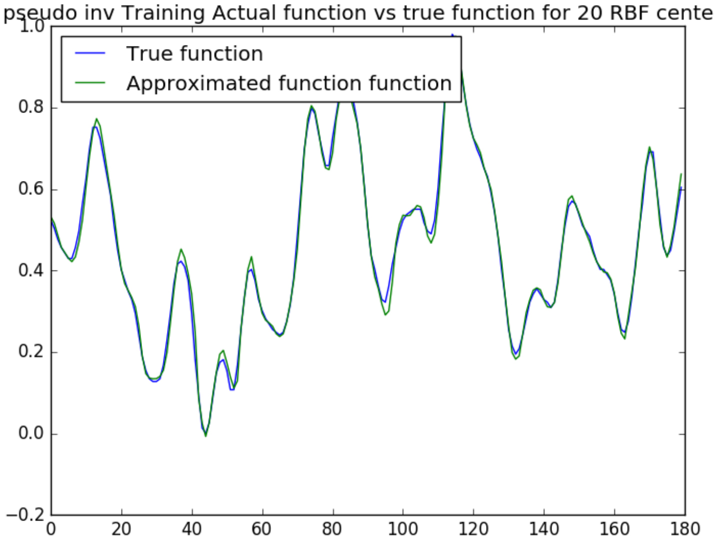

pseudo inv Testing Actual function vs true function for 5 RBF centers
True function
Approximated function function
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100
120
140
160
180

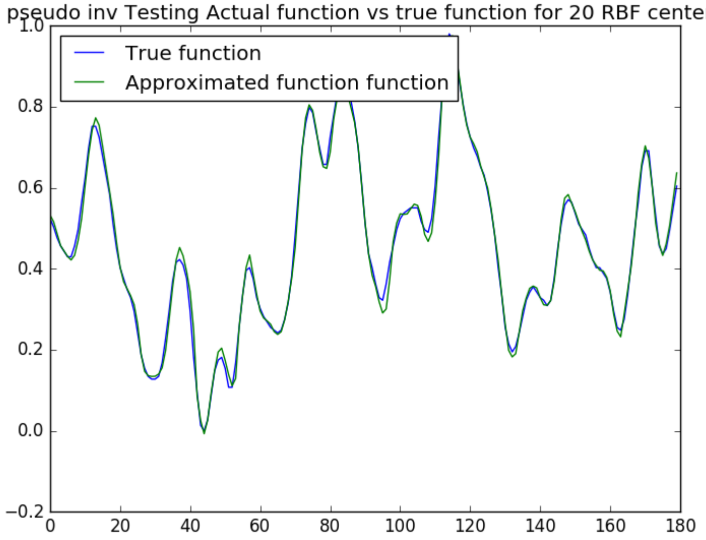
10 Neurons:


pseudo inv Training Actual function vs true function for 10 RBF centers
True function
Approximated function function
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
20
40
60
80
100
120
140
160
180


pseudo inv Testing Actual function vs true function for 10 RBF centers
True function
Approximated function function
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
20
40
60
80
100
120
140
160
180

20 Neurons:


pseudo inv Training Actual function vs true function for 20 RBF centers
True function
Approximated function function
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
20
40
60
80
100
120
140
160
180


pseudo inv Testing Actual function vs true function for 20 RBF centers
True function
Approximated function function
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
20
40
60
80
100
120
140
160
180