T-61.5130 Machine Learning and Neural Networks

Classification using Radial Basis Function Network

Project Report

Vytautas Gudaitis,
vytautas.gudaitis@aalto.fi,
student ID: 464617

January 31, 2016

Introduction

We have been provided with The Breast Cancer Dataset from UC Irvine, that contains 699 records for potential breast cancer patients, there are 9 different attributes providing different information about the patient. Our task is to implement our version of radial basis function network with fixed centers to classify the breast cancer data set, and to evaluate the performance of the network.

Data preparation

In order to perform the evaluation of constructed classifier, we randomly mix the indexes of samples in given dataset and split it in two parts. The first part includes 80% of the data, and will act as our training set. The rest 20% will act as our testing set.

The classifier

Kernel function

The Kernel function of radial basis function network used in our project is Gaussian function[1], displayed in (1):

$$\phi(x) = e^{\frac{-x^2}{\sigma^2}} \tag{1}$$

Width of kernel function and selection of centers

Since we are using Gaussian function as our kernel function, the most common[1] way to set the spread parameter σ is as (2) shows:

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

Here N is the number of centers, and d_{max} is the maximum distance between them.

Since we are supposed to select centers randomly from the data, the σ parameter may always be different while the number of centers N is the same. Thus, in order to find the best parameters for our data, we perform an exhaustive search in order to find these optimal parameters:

1. Randomly choose N centers
2. Calculate the maximum distance (d_{max}) between them
3. Calculate σ

4. Use the randomly selected centers and calculated σ to construct a classifier
5. Train the classifier with the training set
6. Evaluate the classifier with regards to testing set
7. Repeat the process for a selected number of times
8. Select σ from the classifier that performed the best

Computation of weights

In the case of fixed centers, the problem of training a RBF network has a closed-form solution[1], as shown in (3):

$$\boldsymbol{w} = \boldsymbol{\Phi}^{+}\boldsymbol{y}_{d} \tag{3}$$

Here **w** is weight vector, $\mathbf{y_d}$ – target vector. $\boldsymbol{\Phi}$ is explained in (4):

$$\boldsymbol{\Phi} = \begin{bmatrix} \phi(\boldsymbol{x}(1), \boldsymbol{c}_1) & \cdots & \phi(\boldsymbol{x}(1), \boldsymbol{c}_N) \\ \vdots & \ddots & \vdots \\ \phi(\boldsymbol{x}(Q), \boldsymbol{c}_1) & \cdots & \phi(\boldsymbol{x}(Q), \boldsymbol{c}_N) \end{bmatrix} \tag{4}$$

Here $\boldsymbol{\varphi}$ - kernel function of RBF, $\boldsymbol{x(Q)}$ – Q-th training vector, $\boldsymbol{c}_N$ – N-th center vector.

Evaluation of classifier

After the process of center selection and kernel width computation, the classifier is re-trained with the best found hyper parameters and tested with both training and testing sets of the dataset. In order to measure the performance of the classifier, we use the following measures:

1. Mean-square error for training set
2. Mean-square error for testing set
3. Accuracy for training set
4. Accuracy for testing set
5. Confusion matrix for training set
6. Confusion matrix for testing set

The mean-square error[2] (MSE) is calculated as shown in (5):

$$MSE = E[\{y(\boldsymbol{x}) - t\}^2] \tag{5}$$

Here E – mathematical expectation, $y(\boldsymbol{x})$ – a specific estimate of target value t.

The accuracy measure evaluates whether the classifier has classified the sample correctly. Since it only evaluates the binary yes/no meaning, we approximate the values in outcome set towards 1 or 0. The approximated set then is validated against the original target set,

the correct values are counted and divided by the size of target set. The result is given in either a 0-1 range, or a percentage value.

Confusion matrix shows more details into how much the classifier has miss-classified or correctly classified the inputs.

The experiment

In order to evaluate the constructed RBF classifier, we have selected the following parameters:

1. The number of centers is varied, consisting of the following range of values: 2, 5, 10, 15, 20, 50, 100, 200, 300, 400, 559. We want to evaluate how the network behaves in different structures in hidden layer. The last number 559 is selected since this would be the maximum number of vectors in the training set.
2. We select the number of iterations in the exhaustive search for the optimal hyper parameters. In our case, due to the computational cost, our selected number of times to perform the exhaustive search was set to 20.

After the selection of parameters, we perform the algorithm (code attached to *Appendix 1*).

Results

After conducting the experiment with the chosen parameters, we have found out that the best performance was executed by the classifier with 5 centers. The following is the log of the experiment:

```
No. of centers: 2. Training sample accuracy = 96.422182%. MSE = 0.030755.
No. of centers: 2. Test sample accuracy = 97.857143%. MSE = 0.024909.
No. of centers: 5. Training sample accuracy = 96.422182%. MSE = 0.030248.
No. of centers: 5. Test sample accuracy = 97.142857%. MSE = 0.020443.
No. of centers: 10. Training sample accuracy = 95.885510%. MSE = 0.029523.
No. of centers: 10. Test sample accuracy = 97.142857%. MSE = 0.021994.
No. of centers: 15. Training sample accuracy = 96.064401%. MSE = 0.031412.
No. of centers: 15. Test sample accuracy = 97.142857%. MSE = 0.023509.
No. of centers: 20. Training sample accuracy = 95.706619%. MSE = 0.032043.
No. of centers: 20. Test sample accuracy = 97.857143%. MSE = 0.023228.
No. of centers: 50. Training sample accuracy = 95.885510%. MSE = 0.031475.
No. of centers: 50. Test sample accuracy = 95.714286%. MSE = 0.030963.
No. of centers: 100. Training sample accuracy = 96.243292%. MSE = 0.031916.
No. of centers: 100. Test sample accuracy = 95.714286%. MSE = 0.040811.
No. of centers: 200. Training sample accuracy = 95.706619%. MSE = 0.037999.
No. of centers: 200. Test sample accuracy = 92.857143%. MSE = 0.057727.
No. of centers: 300. Training sample accuracy = 96.422182%. MSE = 0.031748.
No. of centers: 300. Test sample accuracy = 89.285714%. MSE = 0.068900.
No. of centers: 400. Training sample accuracy = 97.316637%. MSE = 0.022315.
No. of centers: 400. Test sample accuracy = 90.000000%. MSE = 0.080281.
```

```
No. of centers: 559. Training sample accuracy = 100.000000%. MSE = 0.000000.
No. of centers: 559. Test sample accuracy = 88.571429%. MSE = 0.090638.
```

The following figures display the results of the experiment:

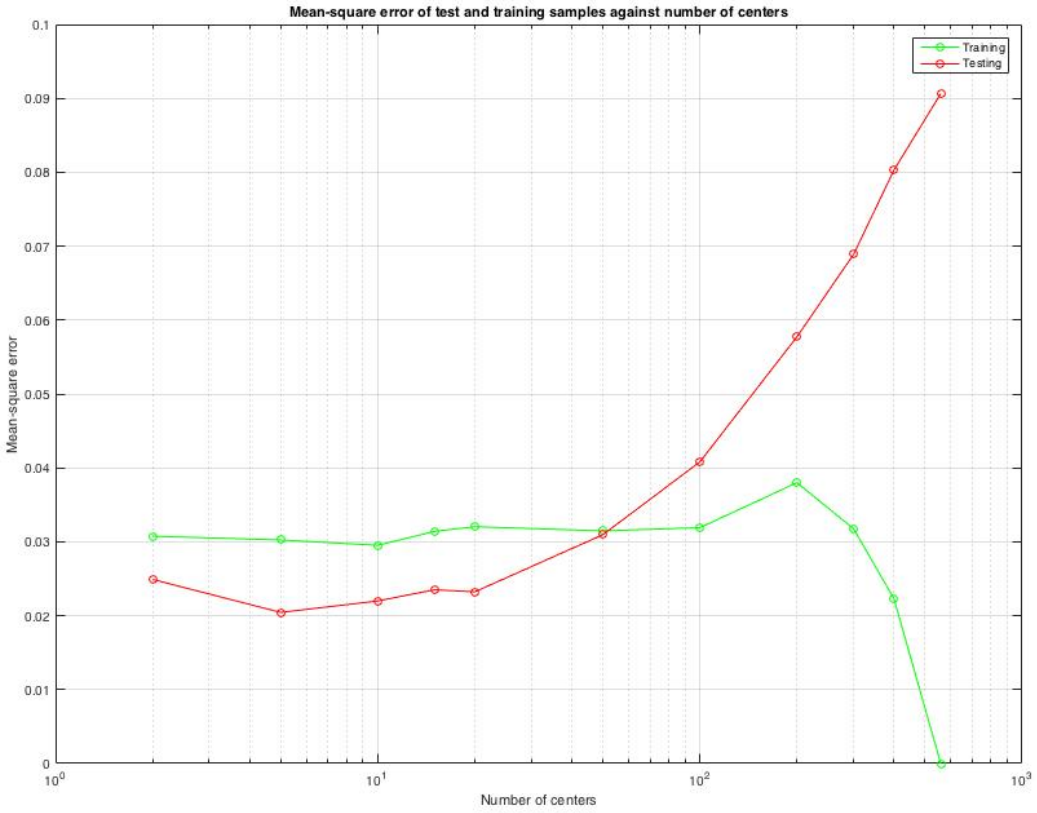

Figure 1: Mean-square error of test and training samples against number of centers

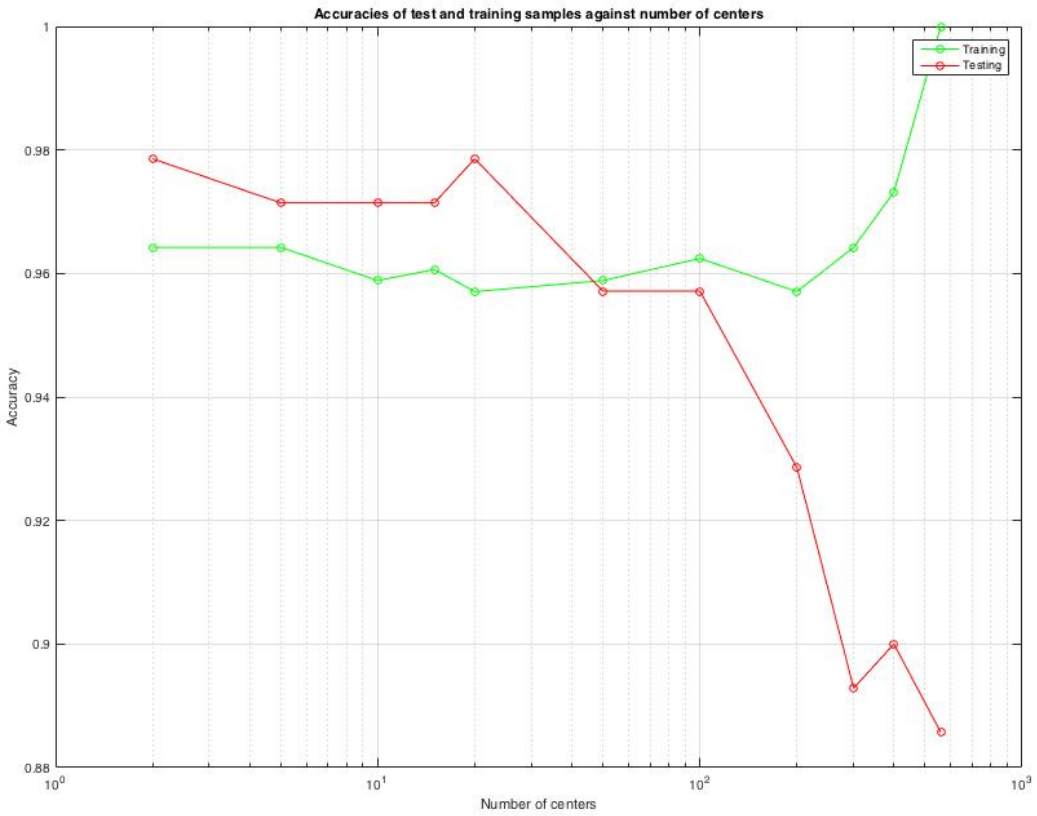

Figure 2: Accuracies of test and training samples against number of centers

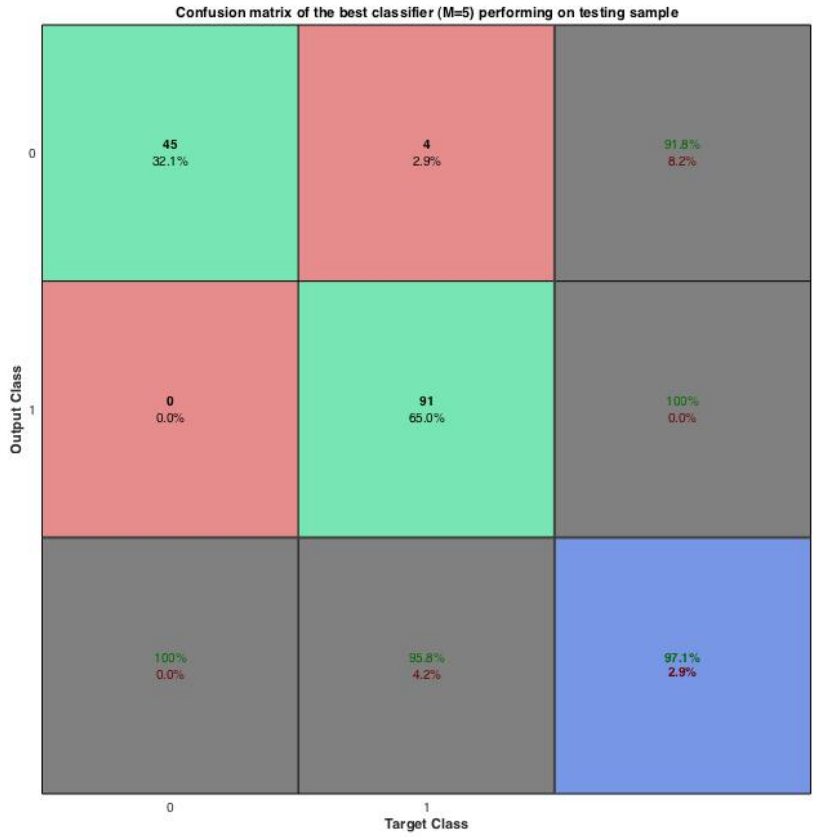

Figure 3: Confusion matrix of the best classifier (M=5) performing on testing sample

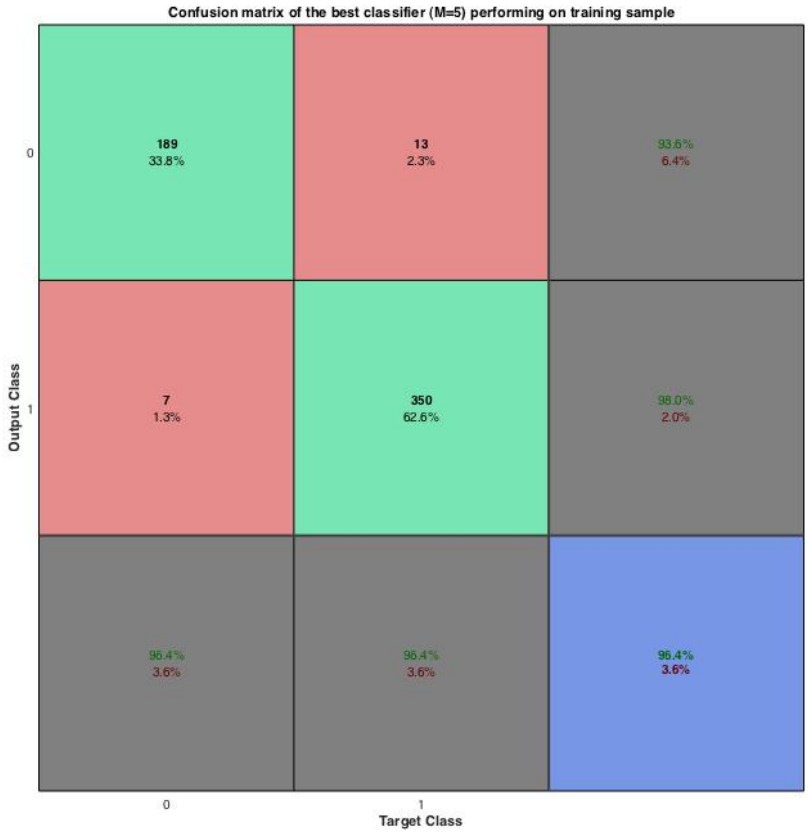

Figure 4: Confusion matrix of the best classifier (M=5) performing on training sample

Note: please find the executable file of the code of the experiment together with the images of the experiment attached to folder of this file.

Conclusions

As we have found, the number of neurons (centers) in the hidden layer of RBF network does not have to be high. Furthermore, our results show that at approximately 100 neurons (centers), the performance of classifier begins to degrade due to the over fitting of the input data. This is an optimistic result, since networks with many neurons in their hidden non-linear layer are computationally expensive to maintain.

However, we think that random selection of centers is not the best nor computationally cheapest way to detect the optimal hyper-parameters for the RFB network. We understand that the task of this project was to select the centers in such way, however, the more popular and probably computationally cheaper way is to use k-means clustering.

References

1. J. Karhunen, Lecture on Radial Basis Function, T-61.5130, Machine Learning and Neural Networks. CS Dept, Aalto University, 2015.
2. J. Karhunen, Lecture on Model Assessment and Selection, T-61.5130, Machine Learning and Neural Networks. CS Dept, Aalto University, 2015.

Appendix 1: MATLAB code of the algorithm

```
clear all;

%% Data preparation
data = load('breastcancer.mat');
x = data.cancerInputs;
y = data.cancerTargets(:,1);

% Use 80% of the set for training and 20% for testing.
indexes = randsample(1:size(x,1), round(size(x,1)*0.8));
x_train = x(indexes, :)';
y_train = y(indexes, :)';
x_test = x;
x_test(indexes, :) = [];
x_test = x_test';
y_test = y;
y_test(indexes, :) = [];
y_test = y_test';

N_train = size(x_train,2); % no. of training data. Should be 140.
N_test=size(x_test,2); % no. of test data. Should be 559.

Ms = [2, 5, 10, 15, 20, 50, 100, 200, 300, 400, 559]; % Select amounts of
centers

results_data = {};

for m = 1 : length(Ms)
    M = Ms(m);

    %% Trying to find best parameters
    attempts = 20; % no. of random attempts

    lowest_MSE = inf;
    best_centers = [];
    best_d_max = [];

    for attempt = 1 : attempts
        index = randsample(1:size(x_train,2), M); % random selection of
centers
        centers = x_train(:,index);

        % Compute d_max. d_max is the maximum distance between the selected
        % centers.
        d_max=0.0;
        for i = 1 : M
            for j = 1: M
                d_max = max(d_max, dist(centers(:,i)',centers(:,j)));
            end
        end

        % Define the radial basis function used.
        % input is an input data and i is the i-th center.
        rbf_i = @(input,i) exp( -M / d_max^2.0 * dist(input',
```

```
centers(:,i))^2.0 );
        % Construct the interpolation matrix.
        interpolation_mat = zeros(N_train, M);
        for r = 1: N_train
            for c = 1: M
                interpolation_mat(r,c) = rbf_i(x_train(:,r), c);
            end
        end
        interpolation_mat = horzcat(ones(N_train, 1), interpolation_mat);
        % Calculate the weights
        w = pinv(interpolation_mat) * y_train';

        %% RBFN Testing:
        bias = w(1,1); %retrieve bias for cleaner code later.

        y_test_outcome = zeros(1,N_test);
        for i = 1 : N_test
            for j = 1 : M
                y_test_outcome(1,i) = y_test_outcome(1,i) + w(j+1,1) * rbf_i(
x_test(:,i), j);
            end
            y_test_outcome(1,i) = y_test_outcome(1,i) + bias;
        end

        %% Performance of RBFN:

        abs_errors_test = abs(y_test_outcome - y_test);
        % Count no. of correct predictions.
        epsilon = 0.5;
        correct_test=0;
        for i = 1 : N_test
            if( abs_errors_test(1, i) < epsilon)
                correct_test = correct_test + 1;
            end
        end

        % Compute MSE
        MSE_testing = (abs_errors_test.^2 * ones(N_test, 1))/ N_test;
        if (MSE_testing < lowest_MSE)
            lowest_MSE = MSE_testing;
            best_centers = centers;
            best_d_max = d_max;
        end
    end

    %% Re-train with best parameters found
    % Define the radial basis function used.
    % input is an input data and i is the i-th best center.
    rbf_i = @(input,i) exp( -M / best_d_max^2.0 * dist(input',
best_centers(:,i))^2.0 );
    % Construct the interpolation matrix.
    interpolation_mat = zeros(N_train, M);
    for r = 1: N_train
        for c = 1: M
            interpolation_mat(r,c) = rbf_i(x_train(:,r), c);
        end
    end
```

```
    interpolation_mat = horzcat(ones(N_train, 1), interpolation_mat);
    % Calculate the weights
    w = pinv(interpolation_mat) * y_train';

    %% RBFN Testing:
    bias = w(1,1); %retrieve bias for cleaner code later.
    y_training_outcome = zeros(1,N_train);
    for i = 1 : N_train
        for j = 1 : M
            y_training_outcome(1,i) = y_training_outcome(1,i) + w(j+1,1) *
rbf_i( x_train(:,i), j);
        end
        y_training_outcome(1,i) = y_training_outcome(1,i) + bias;
    end

    y_test_outcome = zeros(1,N_test);
    for i = 1 : N_test
        for j = 1 : M
            y_test_outcome(1,i) = y_test_outcome(1,i) + w(j+1,1) * rbf_i(
x_test(:,i), j);
        end
        y_test_outcome(1,i) = y_test_outcome(1,i) + bias;
    end

    %% Performance of RBFN:
    abs_errors_training = abs(y_training_outcome - y_train);
    % Count no. of correct predictions.
    epsilon = 0.5;
    correct_training=0;
    for i = 1 : N_train
        if( abs_errors_training(1, i) < epsilon)
            correct_training = correct_training + 1;
        end
    end

    abs_errors_test = abs(y_test_outcome - y_test);
    % Count no. of correct predictions.
    epsilon = 0.5;
    correct_test=0;
    for i = 1 : N_test
        if( abs_errors_test(1, i) < epsilon)
            correct_test = correct_test + 1;
        end
    end

    % Compute MSE & accuracy
    MSE_training = (abs_errors_training.^2 * ones(N_train, 1))/ N_train;
    accuracy_training = correct_training / N_train;
    fprintf('No. of centers: %i. Training sample accuracy = %f%%. MSE = %f.
\n', M, accuracy_training*100, MSE_training);
    MSE_testing = (abs_errors_test.^2 * ones(N_test, 1))/ N_test;
    accuracy_test = correct_test / N_test;
    fprintf('No. of centers: %i. Test sample accuracy = %f%%. MSE = %f. \n',
M, accuracy_test*100, MSE_testing);

    results_data{m, 1} = M;
    results_data{m, 2} = MSE_training;
```

```
    results_data{m, 3} = MSE_testing;
    results_data{m, 4} = accuracy_training;
    results_data{m, 5} = accuracy_test;
    results_data{m, 6} = y_training_outcome;
    results_data{m, 7} = y_test_outcome;
end

%% Create visuals
% find the smallest MSE_testing and it's index
MSEs_test = cell2mat(results_data(:,cat(1,3)));
[M,I] = min(MSEs_test);
% get the row of data
best_classifier_data = results_data(cat(1,I),:);

% plot train sample confusion matrix
figure(1)
plotconfusion(y_train, best_classifier_data{1,6})
title(sprintf('Confusion matrix of the best classifier (M=%i) performing on
training sample', best_classifier_data{1}))

% plot test sample confusion matrix
figure(2)
plotconfusion(y_test, best_classifier_data{1,7})
title(sprintf('Confusion matrix of the best classifier (M=%i) performing on
testing sample', best_classifier_data{1}))

% plot train_MSEs and test_MSEs against Ms
MSEs_train = cell2mat(results_data(:,cat(1,2)));
Ms = cell2mat(results_data(:,cat(1,1)));
figure(3)
semilogx(Ms,MSEs_train,'-go',Ms,MSEs_test,'-ro')
title('Mean-square error of test and training samples against number of
centers')
xlabel('Number of centers')
ylabel('Mean-square error')
legend('Training','Testing')
grid on

% plot accuracy_trainings and accuracy_tests against Ms
accuracy_trainings = cell2mat(results_data(:,cat(1,4)));
accuracy_tests = cell2mat(results_data(:,cat(1,5)));
figure(4)
semilogx(Ms,accuracy_trainings,'-go',Ms,accuracy_tests,'-ro')
title('Accuracies of test and training samples against number of centers')
xlabel('Number of centers')
ylabel('Accuracy')
legend('Training','Testing')
grid on
```