Cmpe462 HW1

Şahin Batmaz
2012400117

Part 1 & 2

```
N = 20; # Size of a sample.
M = 100; # Number of samples.
Order = 5; # Samples will be fit to polynomial models of this number of orders.


# Dataset X creation - Nx1 vector
# Uniform distribution is used to generate N values.
sample_x = unifrnd(0,5,1,N);

# f function that constructs dataset Y_i from dataset X
# Y_i is a sample consists of N values.
function fx_y = func_x_to_y(vec_x,index)
  fx_y = 2*sin(1.5*vec_x(index));
end

# Dataset Y creation - MxN matrix
# There are M samples.
# Each sample can be represented as Y_i.
# Y_i is a sample consists of N values generated from dataset X.
# Same dataset X is used to generate each Y_i.
sample_y= zeros(M,N);
for i=1:M
  for j=1:N
    sample_y(i,j)=func_x_to_y(sample_x,j)+normrnd(0,1);
  end
end

# -------------------------------------
# -------------------------------------

# Function that finds the parameters of a polynomial function.
# It is assumed that there is a function g(x),
# that fits to dataset such that g(x_t) = y_t.
# The order of polynomial function is given as input.
# The function takes x vector and corresponding y vector as input.
# For example, if order is 2, the function is like a.x^2 + b.x +c,
# therefore, this function returns values of a,b and c.
# Note that, this function has the same functionality of 'polyfit' function.
function parameters = mypolyfit(vec_x,vec_y,order_num)

  row_number = length(vec_x);
  col_number = order_num+1;

  design_matrix = zeros(row_number,col_number);
```

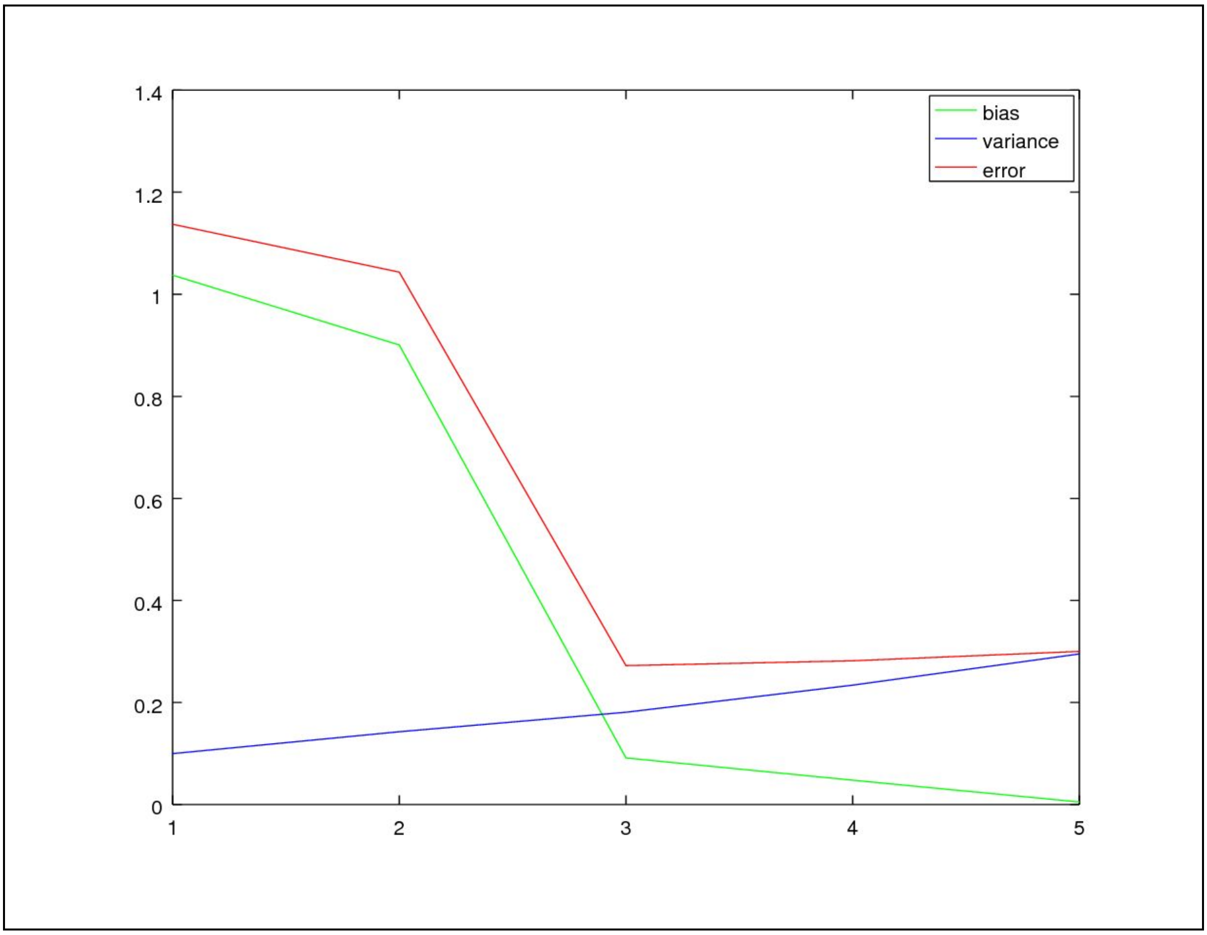
```
  for i=1:row_number
    for j=1:col_number
      design_matrix(i,j)=vec_x(i)^(col_number-j);
    end
  end
  parameters = pinv((design_matrix')*design_matrix)*(design_matrix')*(vec_y');

end

# --------------------------------------
# --------------------------------------
# PART 1 of HW1

# For each order value from 1 to given value of 'order',
# there will be values for bias and variance.
# Following vectors will store these values.
bias_list = zeros(Order,1);
variance_list = zeros(Order,1);

# For each order of polynomial fit
for order_val=1:Order

  # Model Creation
  # For each sample Y_i in dataset Y, a model will be generated.
  # Then average model will be generated
  # by taking the mean of parameters of models.
  models = zeros(M,order_val+1);
  for i=1:M
    models(i,:) = mypolyfit(sample_x,sample_y(i,:),order_val);
  end
  model_avg = mean(models);



  # Calculating bias, variance, error of model
  # Bias
  # Applying formula of bias
  # Average model, f function and dataset X is used
  bias = 0;
  for i=1:N
    diff_i = polyval(model_avg,sample_x(i))-func_x_to_y(sample_x,i);
    bias = bias + diff_i^2;
  end
  bias = bias/N;

  # Variance
  # Applying formula of variance
```

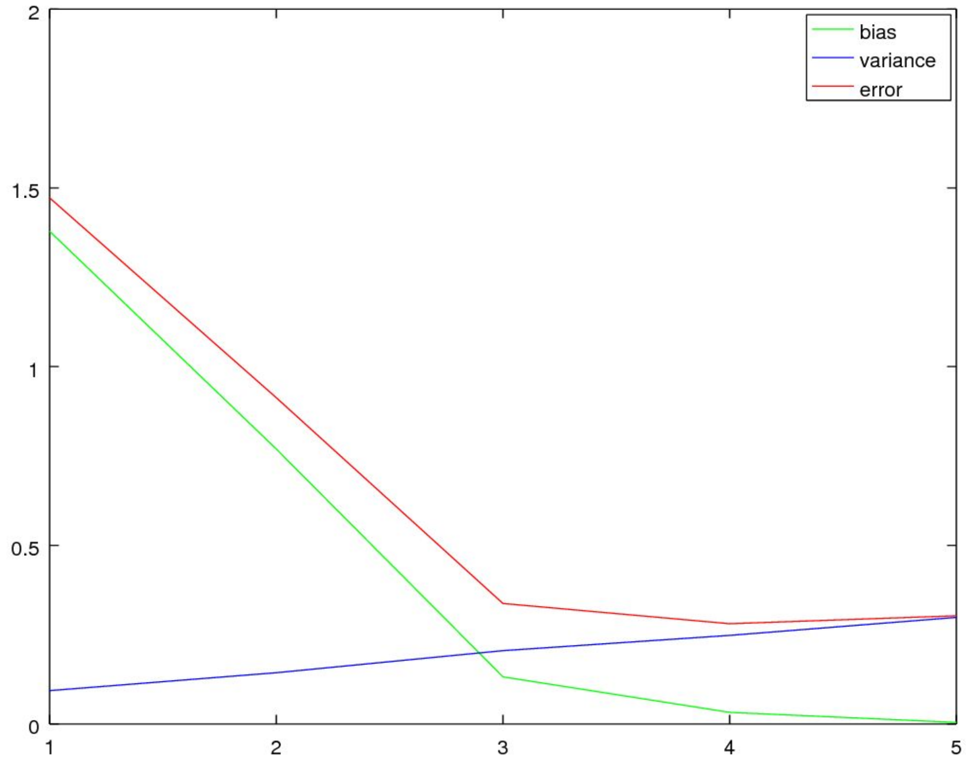
```
  # Average model, all models, and dataset X is used
  variance = 0;
  for i=1:N
    for j=1:M
      diff_i = polyval(models(j,:),sample_x(i)) - polyval(model_avg,sample_x(i));
      variance = variance + diff_i^2;
    end
  end
  variance = variance/(M*N);

  # Save bias and variance in the vectors to be able to plot them together
  bias_list(order_val) = bias;
  variance_list(order_val) = variance;
end

# -------------------------------------
# -------------------------------------

# Result of PART 1
# Plot bias, variance and error
figure;
hold;
plot(bias_list,"g")
plot(variance_list,"b")
plot(bias_list+variance_list,"r")

legend("bias","variance","error")

# -------------------------------------
# -------------------------------------
# PART 2 of HW1

# Each sample will be fit to polynomial models of this number of orders.
Order=5;

# For each order value from 1 to given value of 'order',
# there will be values for bias and variance.
# For training and validation datasets,
# there will be different bias and variance values.
# Following vectors will store these values.
bias_list_training = zeros(Order,1);
variance_list_training = zeros(Order,1);

bias_list_validation = zeros(Order,1);
variance_list_validation = zeros(Order,1);

# For each order of polynomial fit
for order_val=1:Order
```

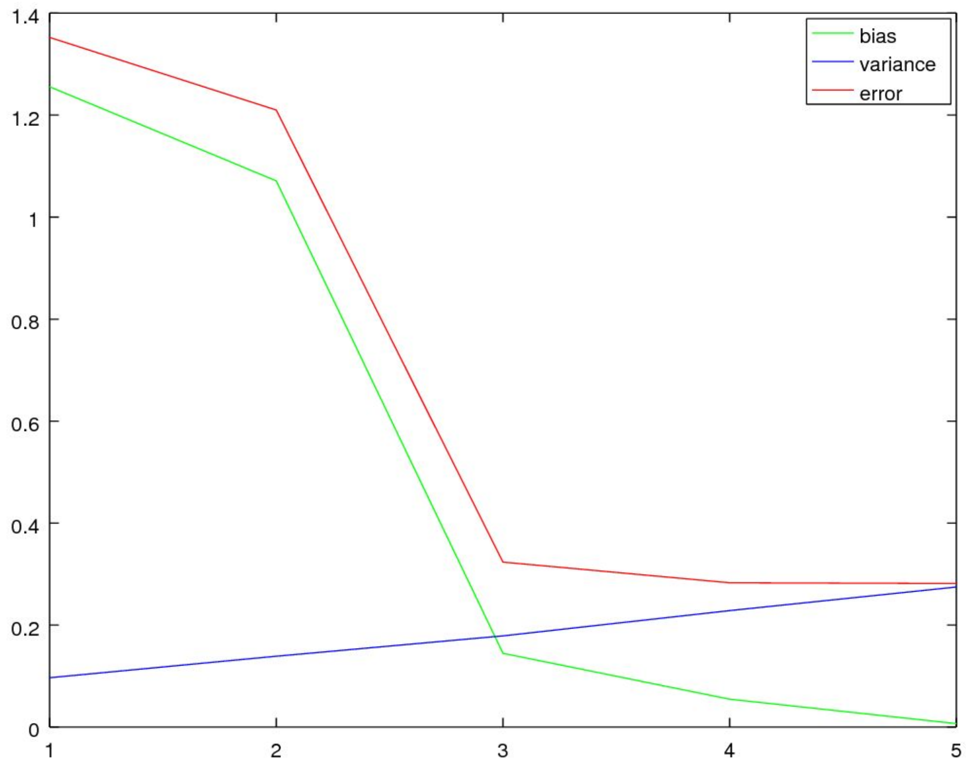
```
# Model Creation
# For each sample Y_i in dataset Y, a model will be generated.
# In each sample, only odd indexed values are used to construct model.
# Then average model will be generated
# by taking the mean of parameters of models.
models = zeros(M,order_val+1);
for i=1:M
  models(i,:) = mypolyfit(sample_x(1:2:20),sample_y(i,1:2:20),order_val);
end
model_avg = mean(models);

# Bias, variance, error of model
# Bias
# Applying formula of bias
# Average model, f function and dataset X is used

# Calculate bias for training dataset
bias = 0;
for i=1:2:20
  diff_i = polyval(model_avg,sample_x(i))-func_x_to_y(sample_x,i);
  bias = bias + diff_i^2;
end
bias = bias/10;

bias_list_training(order_val) = bias;


# Calculate bias for validation dataset
bias = 0;
for i=2:2:20
  diff_i = polyval(model_avg,sample_x(i))-func_x_to_y(sample_x,i);
  bias = bias + diff_i^2;
end
bias = bias/10;

bias_list_validation(order_val) = bias;

# Variance
# Applying formula of variance
# Average model, all models, and dataset X is used

# Calculate variance for training dataset
variance = 0;
for i=1:2:20
  for j=1:M
    diff_i = polyval(models(j,:),sample_x(i)) - polyval(model_avg,sample_x(i));
    variance = variance + diff_i^2;
```

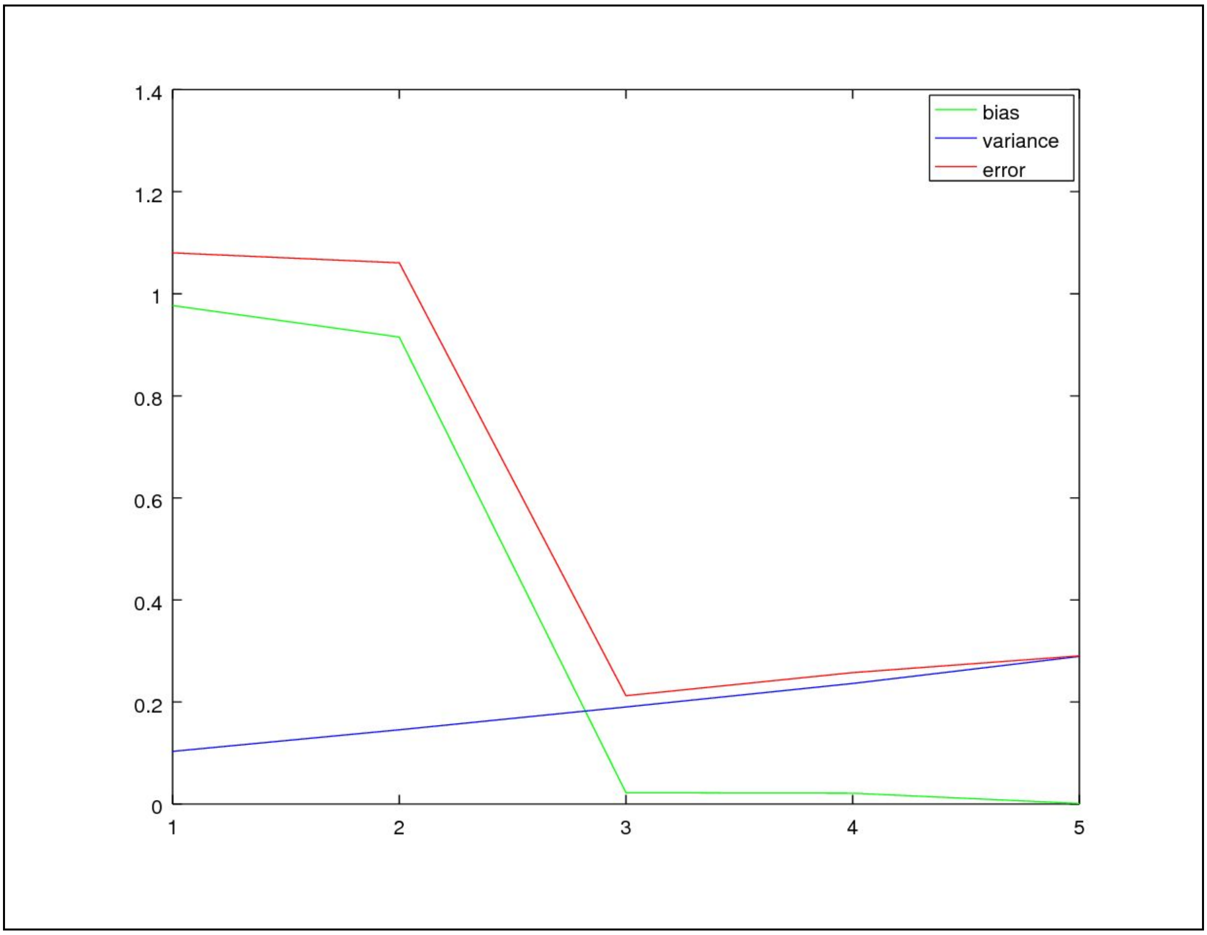
```
    end
  end
  variance = variance/(M*10);

  variance_list_training(order_val) = variance;


  # Calculate variance for validation dataset
  variance = 0;
  for i=2:2:20
    for j=1:M
      diff_i = polyval(models(j,:),sample_x(i)) - polyval(model_avg,sample_x(i));
      variance = variance + diff_i^2;
    end
  end
  variance = variance/(M*10);

  variance_list_validation(order_val) = variance;

end


# -------------------------------------
# -------------------------------------

# Result of PART 2
# Plot error values of training and validation datasets
figure;
hold;
plot(bias_list_training+variance_list_training,"r")
plot(bias_list_validation+variance_list_validation,"b")
legend("training","validation")


# -------------------------------------
# -------------------------------------
```

Result of Part 1



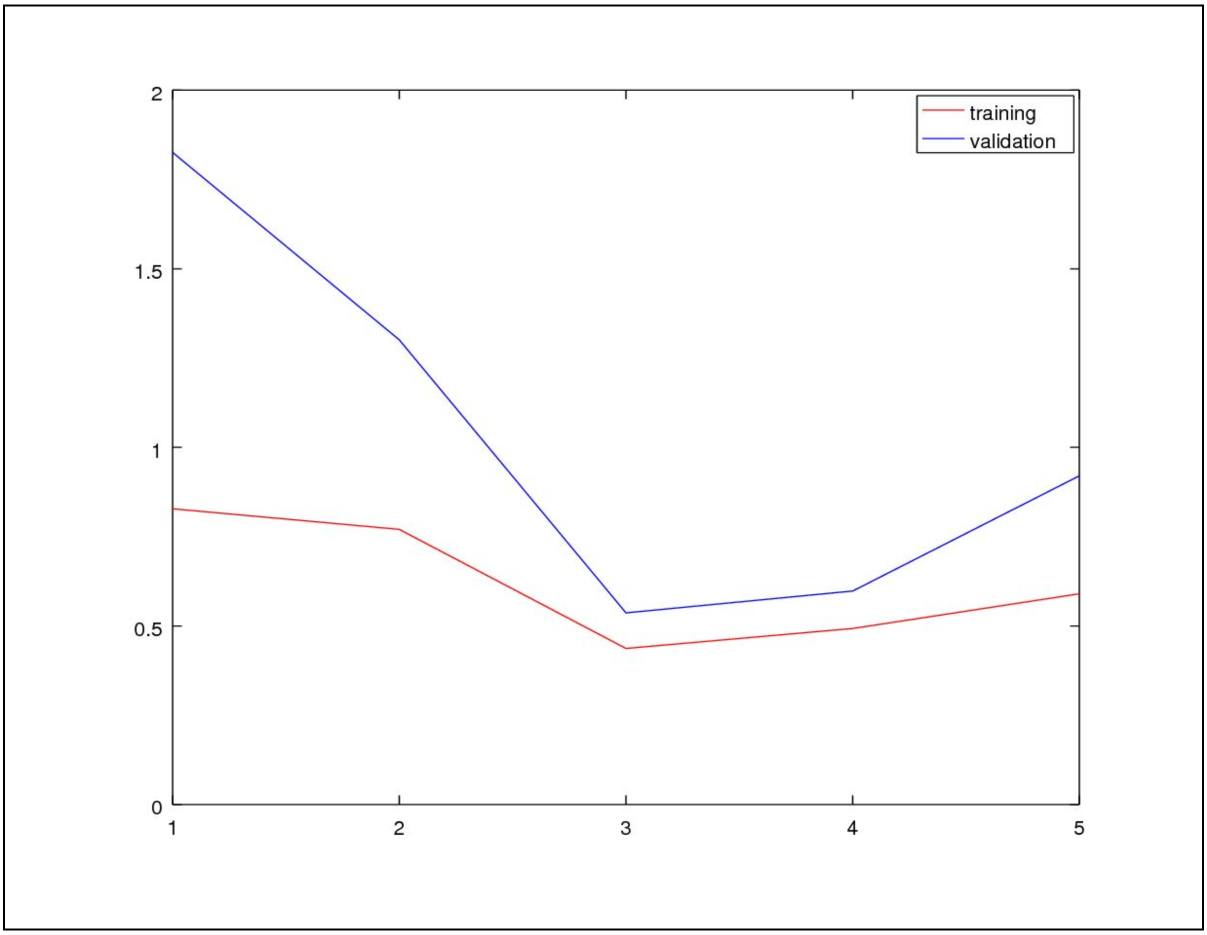

bias
variance
error
2
1.5
1
0.5
0
1
2
3
4
5

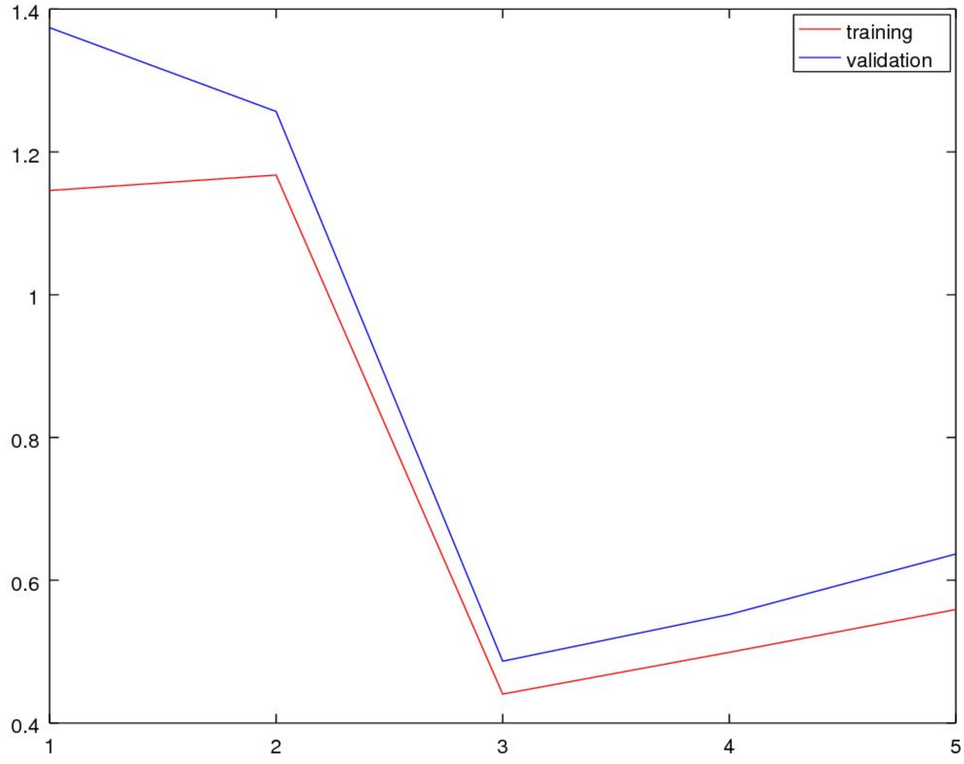

bias
variance
error
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1
2
3
4
5



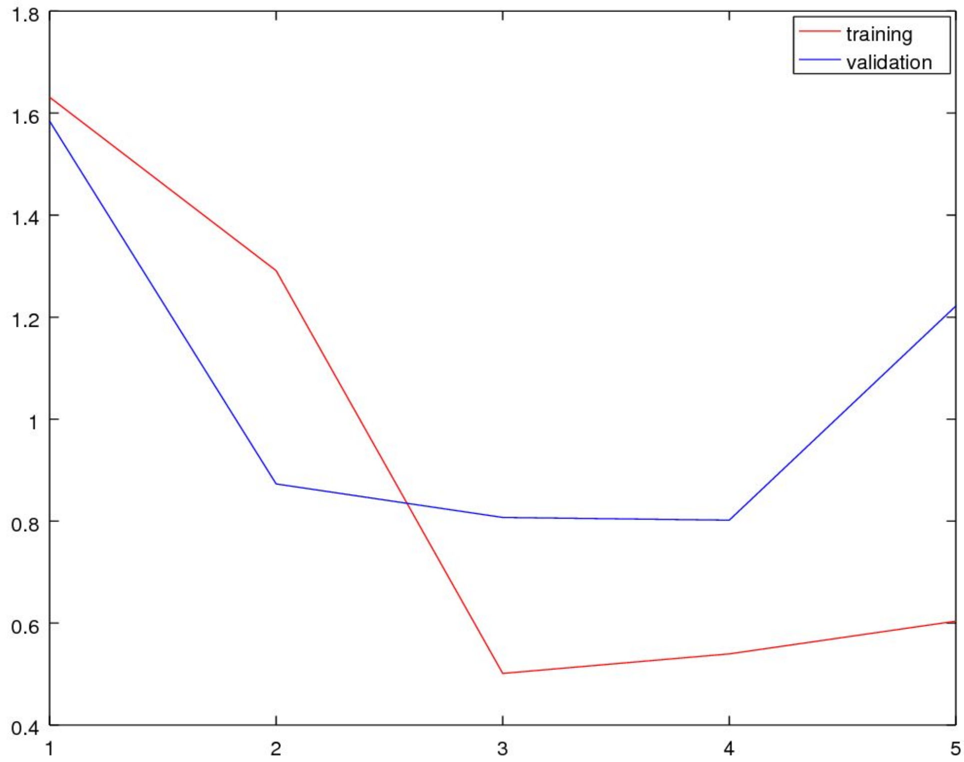
bias
variance
error
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1
2
3
4
5

Result of Part 2

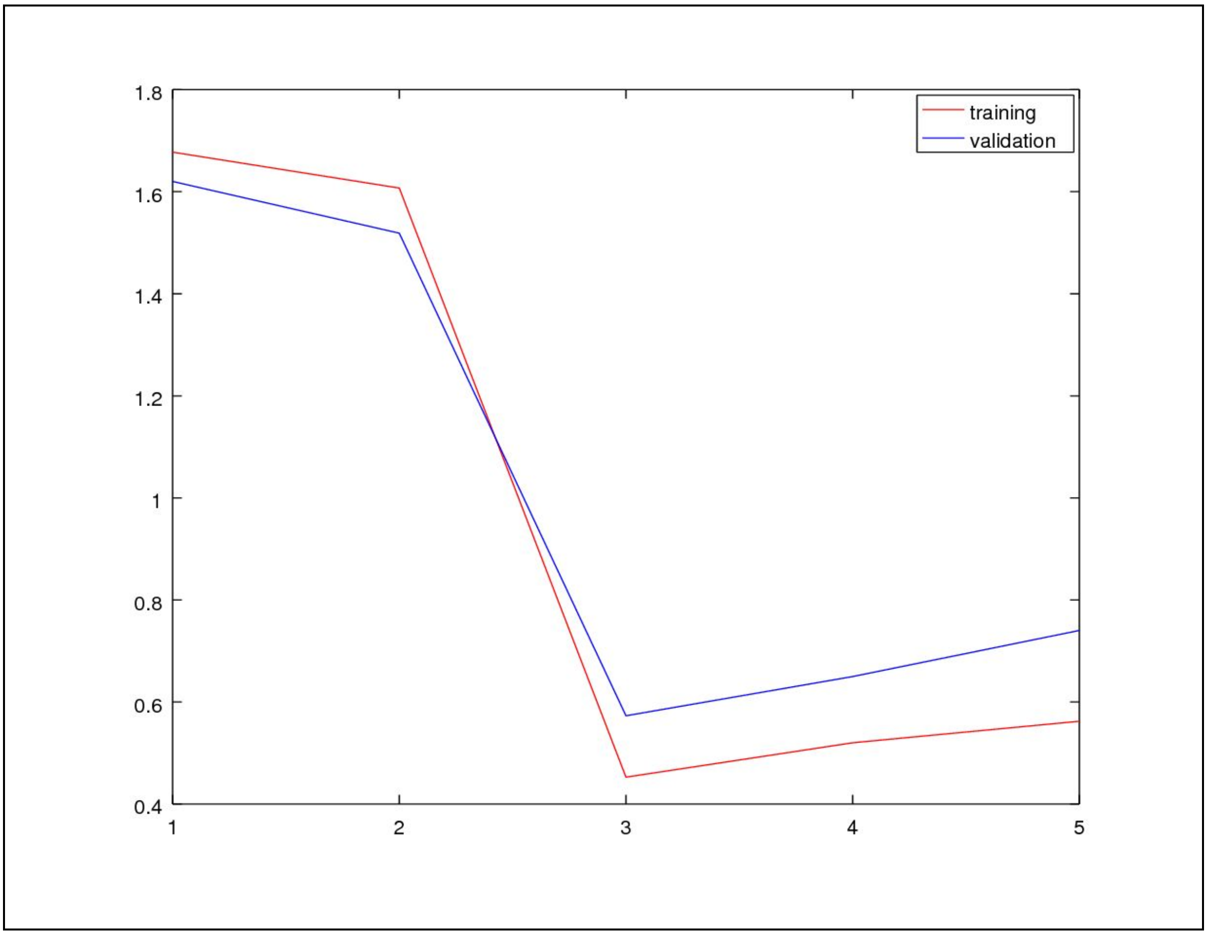




1.4
1.2
1
0.8
0.6
0.4
1
2
3
4
5
training
validation



1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
1
2
3
4
5
training
validation


training
validation
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
1
2
3
4
5

Part 3

```
# Note that, iris.data.txt file is updated.
# Last 50 lines are deleted since we only need first two classes.
# Last column of csv file is for the class name.
# Class name Iris-setosa is replaced with 0.
# Class name Iris-versicolor is replaced with 1.

# Iris dataset that consists of 100 lines.
iris_data = csvread('iris.data.txt');
# Last column of dataset. It is the list of classes of the data.
iris_class = iris_data(:,end);

N=100; # Number of data entries in the dataset.
N_class_0=50; # Number of data for class 0 in the dataset.
N_class_1=50; # Number of data for class 1 in the dataset.
N_class_0_training=35; # Number of training data for class 0 in the dataset.
N_class_1_training=35; # Number of training data for class 1 in the dataset.
N_class_0_test=15; # Number of test data for class 0 in the dataset.
N_class_1_test=15; # Number of test data for class 1 in the dataset.
N_test=30; # Number of all test data in the dataset.

# There are 4 features in the dataset. (4 columns of data)
# For each feature following code will be executed.
# It will calculate mean and variance values for class 0 and
# class 1 from training datasets.
# Then, to find the success of the classification, test dataset is used.
# Parametric classification is used.
# By applying normpdf function, likelihood values are
# obtained for class 0 and class 1.
# By comparing likelihood values that come from class 0 and class 1,
# decision of classifying is made.
# Then the decision is compared with actual class of the data.
# Number of false decisions is counted and considered as error.
# Error value and size of the test dataset is printed on the screen.
for i=1:4
  # Current column of feature.
  feature = iris_data(:,i);

  # Indeces for training datasets are randomly selected
  training_indeces_0 = randperm(N_class_0,N_class_0_training);
  training_indeces_1 = randperm(N_class_1,N_class_1_training)+N_class_0;

  # Training data vectors are obtained by using indeces of training datasets.
  feature_from_class0 = feature(training_indeces_0);
```

```
  feature_from_class1 = feature(training_indeces_1);

  # Mean and variance values are calculated for class 0 and class 1.
  f0_mean = mean(feature_from_class0);
  f0_var = var(feature_from_class0);

  f1_mean = mean(feature_from_class1);
  f1_var = var(feature_from_class1);

  # Then test dataset is obtained by subtracting training datasets
  # from the all Iris dataset.
  test_indeces_0 = setdiff([1:N_class_0],training_indeces_0);
  test_indeces_1 = setdiff([N_class_0+1:N],training_indeces_1);
  test_indeces = [test_indeces_0,test_indeces_1];

  # pdf values are calculated for class 0 and class 1
  pdf0 = normpdf(feature(test_indeces),f0_mean,f0_var);
  pdf1 = normpdf(feature(test_indeces),f1_mean,f1_var);

  # pdf values are compared and classification is done.
  # then the classification is compared with actual classes.
  # number of wrong decisions is counted and considered as error number.
  # and error number for that feature is printed with the size of test dataset.
  results = zeros(N_test,1);
  results(pdf1>pdf0)=1;
  error = sum(abs(results-iris_class(test_indeces)));

  fprintf("In feature %d, out of %d elements in test dataset,
  there are %d wrong decisions.\n\n",i,N_test,error);

end
```

Result of Part 3

Best feature to classify setosa and versicolor is the feature 4, petal width in cm.

In feature 1, out of 30 elements in test dataset, there are 6 wrong decisions. In feature 2, out of 30 elements in test dataset, there are 7 wrong decisions. In feature 3, out of 30 elements in test dataset, there are 6 wrong decisions. In feature 4, out of 30 elements in test dataset, there are 0 wrong decisions.
In feature 1, out of 30 elements in test dataset, there are 6 wrong decisions. In feature 2, out of 30 elements in test dataset, there are 7 wrong decisions. In feature 3, out of 30 elements in test dataset, there are 2 wrong decisions. In feature 4, out of 30 elements in test dataset, there are 0 wrong decisions.
In feature 1, out of 30 elements in test dataset, there are 5 wrong decisions. In feature 2, out of 30 elements in test dataset, there are 4 wrong decisions. In feature 3, out of 30 elements in test dataset, there are 1 wrong decisions. In feature 4, out of 30 elements in test dataset, there are 1 wrong decisions.
In feature 1, out of 30 elements in test dataset, there are 5 wrong decisions. In feature 2, out of 30 elements in test dataset, there are 4 wrong decisions. In feature 3, out of 30 elements in test dataset, there are 1 wrong decisions. In feature 4, out of 30 elements in test dataset, there are 0 wrong decisions.
In feature 1, out of 30 elements in test dataset, there are 8 wrong decisions. In feature 2, out of 30 elements in test dataset, there are 3 wrong decisions. In feature 3, out of 30 elements in test dataset, there are 2 wrong decisions. In feature 4, out of 30 elements in test dataset, there are 0 wrong decisions.
In feature 1, out of 30 elements in test dataset, there are 3 wrong decisions. In feature 2, out of 30 elements in test dataset, there are 1 wrong decisions. In feature 3, out of 30 elements in test dataset, there are 0 wrong decisions. In feature 4, out of 30 elements in test dataset, there are 0 wrong decisions.