CSCC11 Fall 2015
Assignment 1

Student Name: Maoting Zhang
Student Number: 999590633
UTORid(mathlab username): zhangmao

- # Matlab Scripts & functions

  - ## polynomialRegression.m

```
function w = polynomialRegression(K,x,y)
% Fill in this function.  It should return a (K+1)x1 weight
vector w where the
% estimated model function is f(x) = \sum_{i=0}^{K} w(i) x^i.
for i =1:size(x)
    for j = 0:K
      B(i, j+1) = x(i)^j;
      end
end
w = B \ y;
end
```

i.e.: we define B as a NxK matrix s.t. :

$$B_{ij} \text{ (} i^{th} \text{ row and } j^{th} \text{ column in B) } = x_i^j$$

  - ## evalPolynomial.m

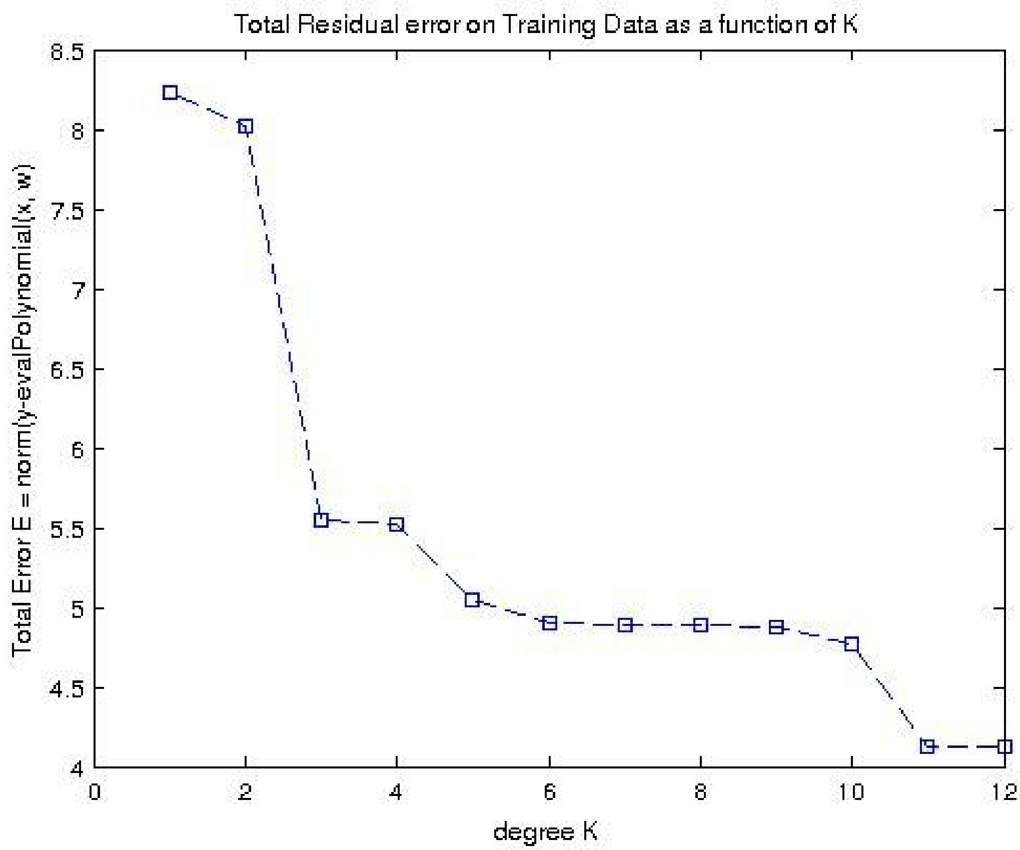
```
function y = evalPolynomial(x,w)
% Evaluate the polynomial defined by the given weight vector at
the
% given values of x.  This function should work even if x is a
vector.
y = 0;
for i = 1:size(w)
    % follow the definition f(x) = \sum_{i=0}^{K} w(i) x^i
    y = y + w(i)*(x.^(i-1));
end
```

  - ## fit.m (script)

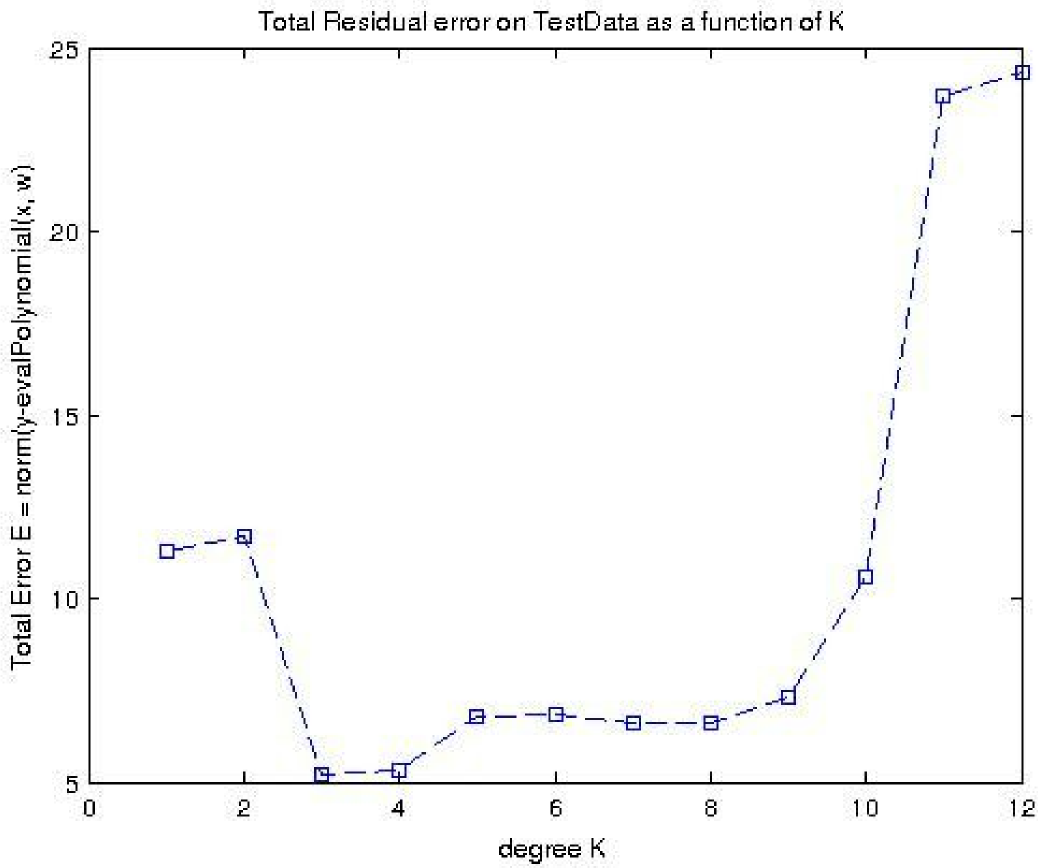
```
load('a1TrainingData.mat');
load('a1TestData.mat');

figure(1);
```

```
% random color map (12-by-3 vector)
cmap = hsv(12);
for K = 1:12
    w = polynomialRegression(K, x, y)
    E = norm(y - evalPolynomial(x, w))
    errorArray(K) = E;
    testE = norm(yTest - evalPolynomial(xTest, w))
    newErrorArray(K) = testE;

    my_x = -2.1:0.1:2.1;
    % plot data set regarding each degree K in a random color in
cmap
    plot(my_x, evalPolynomial(my_x, w), 'Color', cmap(K, :));
    % make a cell array with legend info so we can call legend
after the loop
    legendInfo{K} = ['K = ' num2str(K)];
    hold on;
end
legend(legendInfo)
title('Fitted model for x = [-2.1:0.1:2.1] in different K')

% error plot on training data
figure(2);
plot(1:12, errorArray, '--bs');
xlabel('degree K');
ylabel('Total Error E = norm(y-evalPolynomial(x, w)');
title('Total Residual error on Training Data as a function of
K');

% total error plot test data
figure(3);
plot(1:12, newErrorArray, '--bs');
xlabel('degree K');
ylabel('Total Error E = norm(y-evalPolynomial(x, w)');
title('Total Residual error on TestData as a function of K');
```

- Error Plots



Total Error on **training data** as a function of K



Total Error on **test data** as a function of K

- ## Analysis on Error Plots

Error Plot on Trainning Data:

From the prespective on total error, those models with less Total Error E should be "good" models, so I would choose K = 12. (as it minimizes E = norm(y - f(x)))

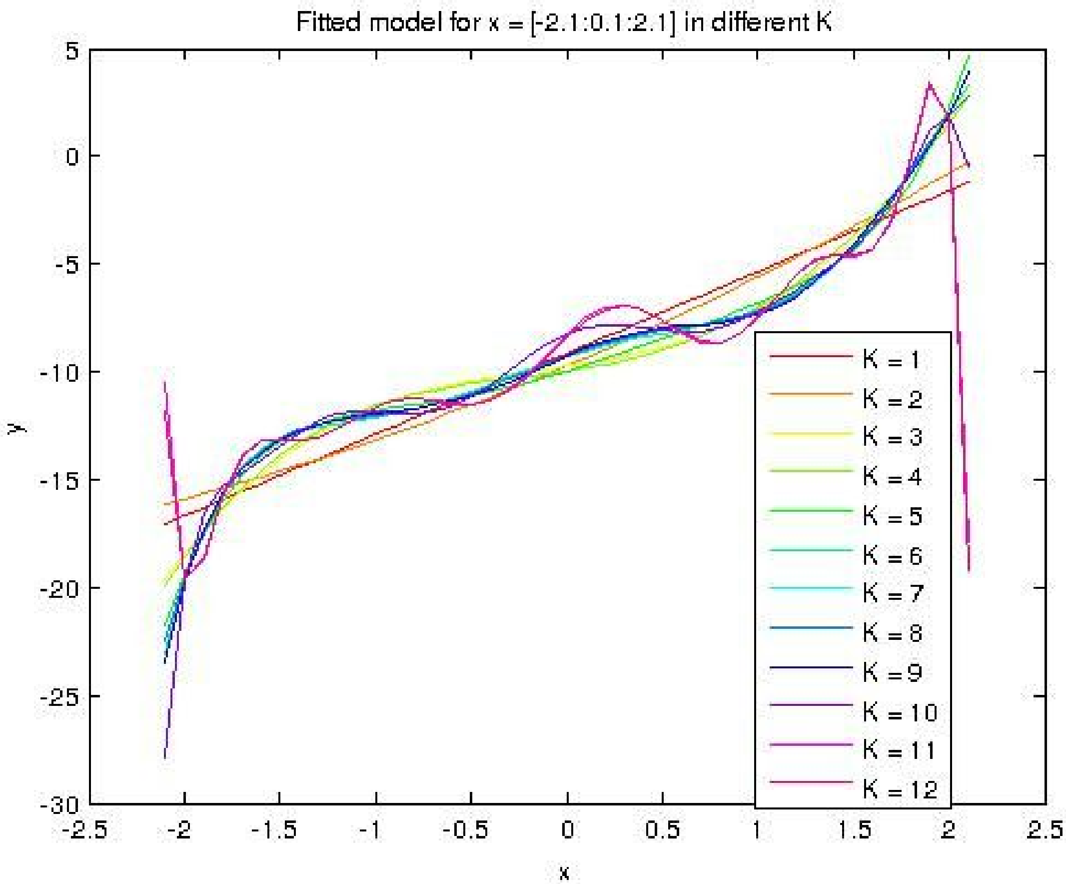
Error Plot on Test Data:

The slope of the two plots are in opposite order: total error on Training data are going down as degree K goes up, while on Test Data E goes down first and stay stable for a few K's then go up rapidly in the next step.

When K = 11 and/or K = 12, total error E tends to be small on training data and considerably high on test Data, so I could say it "overfit" the training data.

when K = 4, E is 5.5159 on training data and 5.3305 on Test Data and have the smallest difference among E for all other degrees. Therefore, I would guess a polynomial of degree 4 was used to generate the training and test data.

- Fit Models



Fitted model for x = [-2.1:0.1:2.1] in different K
5
0
-5
-10
-15
-20
-25
-30
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
x
y
K = 1
K = 2
K = 3
K = 4
K = 5
K = 6
K = 7
K = 8
K = 9
K = 10
K = 11
K = 12