Homework 2

Zachary DeStefano, 15247592

CS 273A: Winter 2015

Due: January 20, 2015

Problem 1

Part a

Here is the code to complete part a

```
iris=load('data/curve80.txt');
y=iris(:,2);
X=iris(:,1);

%Part A
[Xtr Xte Ytr Yte] = splitData(X,y, .75); % split data into 75/25 train/test
```

Part b

Here is the code to complete part b. It does rely on the code from part a:

```
%%
%Part B
lr = linearRegress( Xtr, Ytr ); % create and train model
xs = (0:.05:10)'; % densely sample possible x-values
ys = predict( lr, xs ); % make predictions at xs

plot(xs,ys)
hold on
plot(Xtr,Ytr,'g.','MarkerSize',10)
%plot(Xte,Yte,'rx','MarkerSize',10)
legend('Prediction','Training Data','Location','SouthEast');

%calculate MSE
YhatTr = predict(lr,Xtr); %gets predicted y for training data
YhatTe = predict(lr,Xte); %gets predicted y for test data
mseTr = mean(abs(YhatTr-Ytr).^2);
mseTe = mean(abs(YhatTe-Yte).^2);
```

The MSE for the training data was 1.1277
The MSE for the test data was 2.2423

Here is the plot for part b:

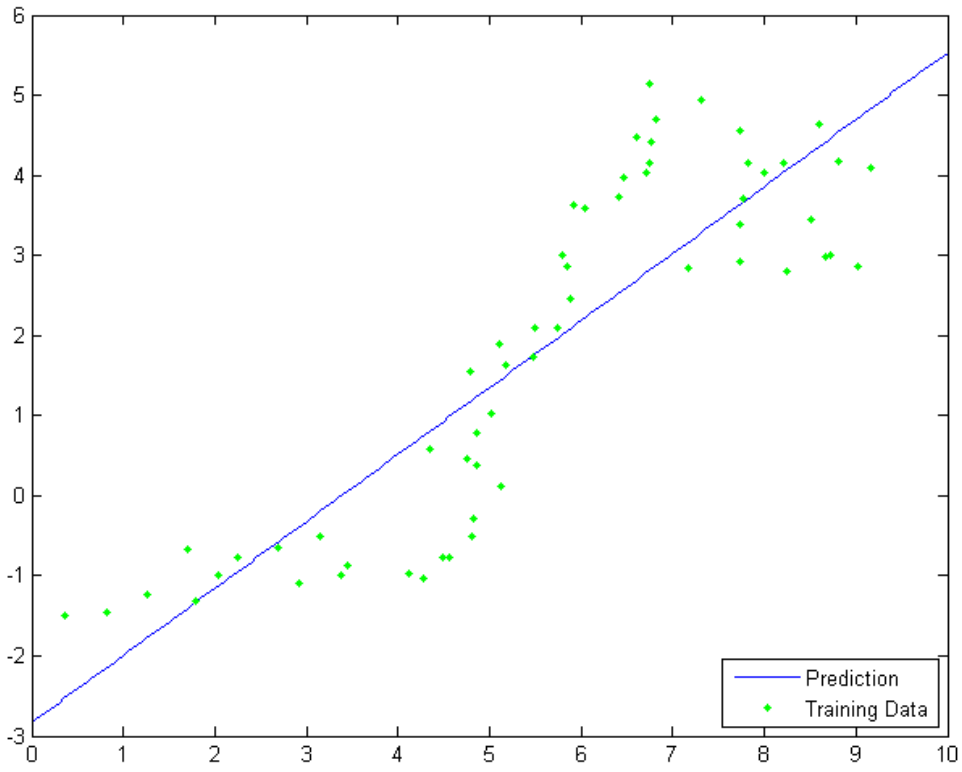
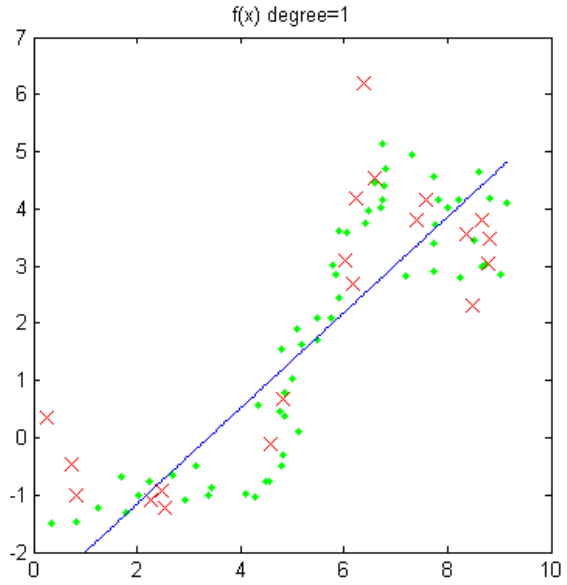
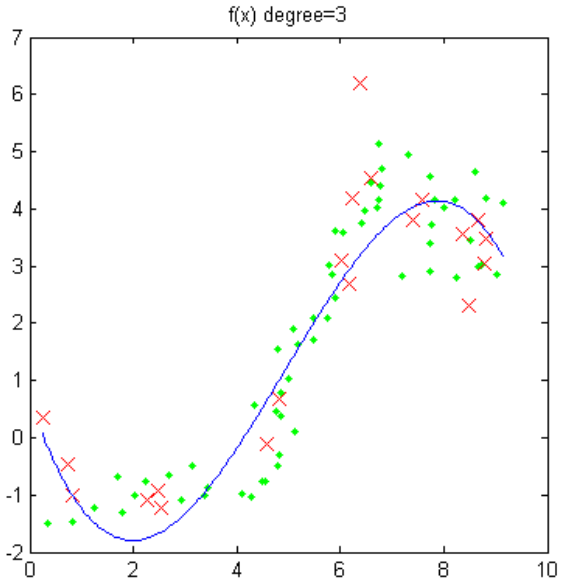


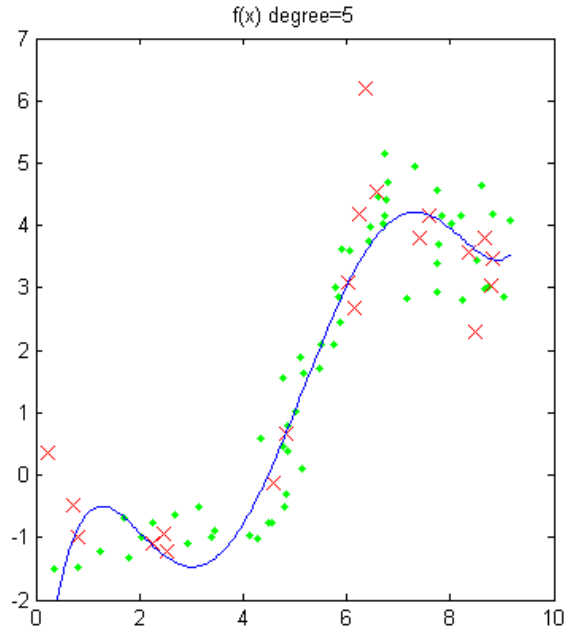
Figure 1: The training data, test data, and the predicted values

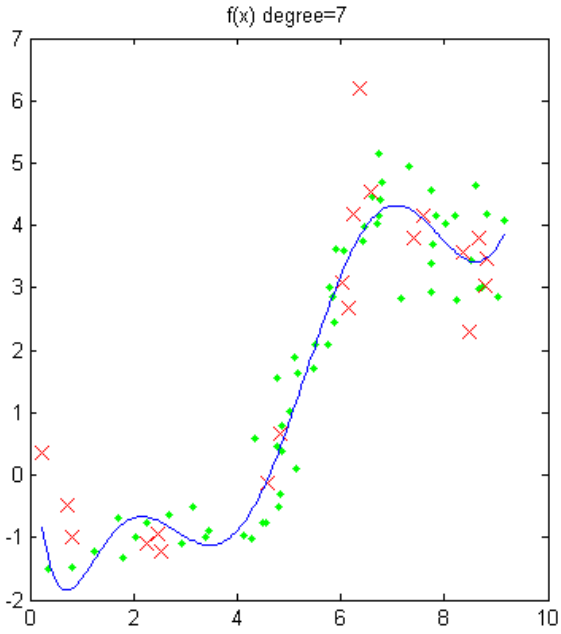
Part c

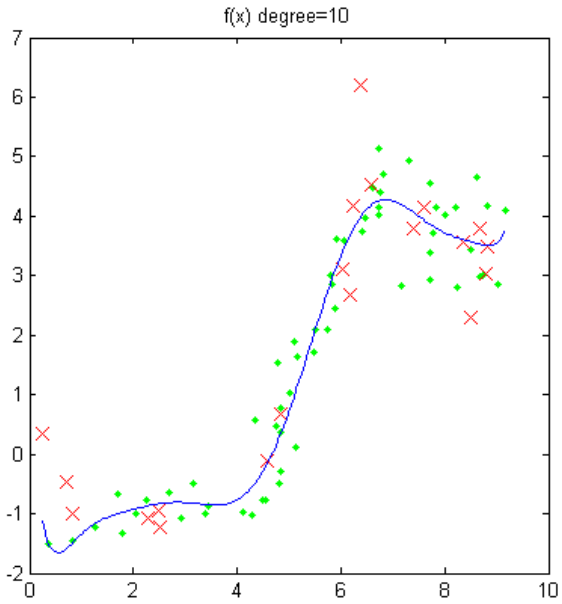
Here are the plots of the $f(x)$ functions and the training and test data. The legend is the same as the plot in part b. Each plot represents a prediction for a different polynomial degree.

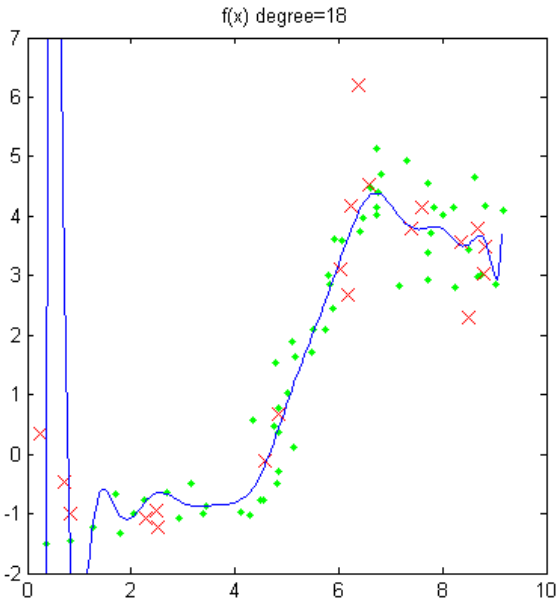




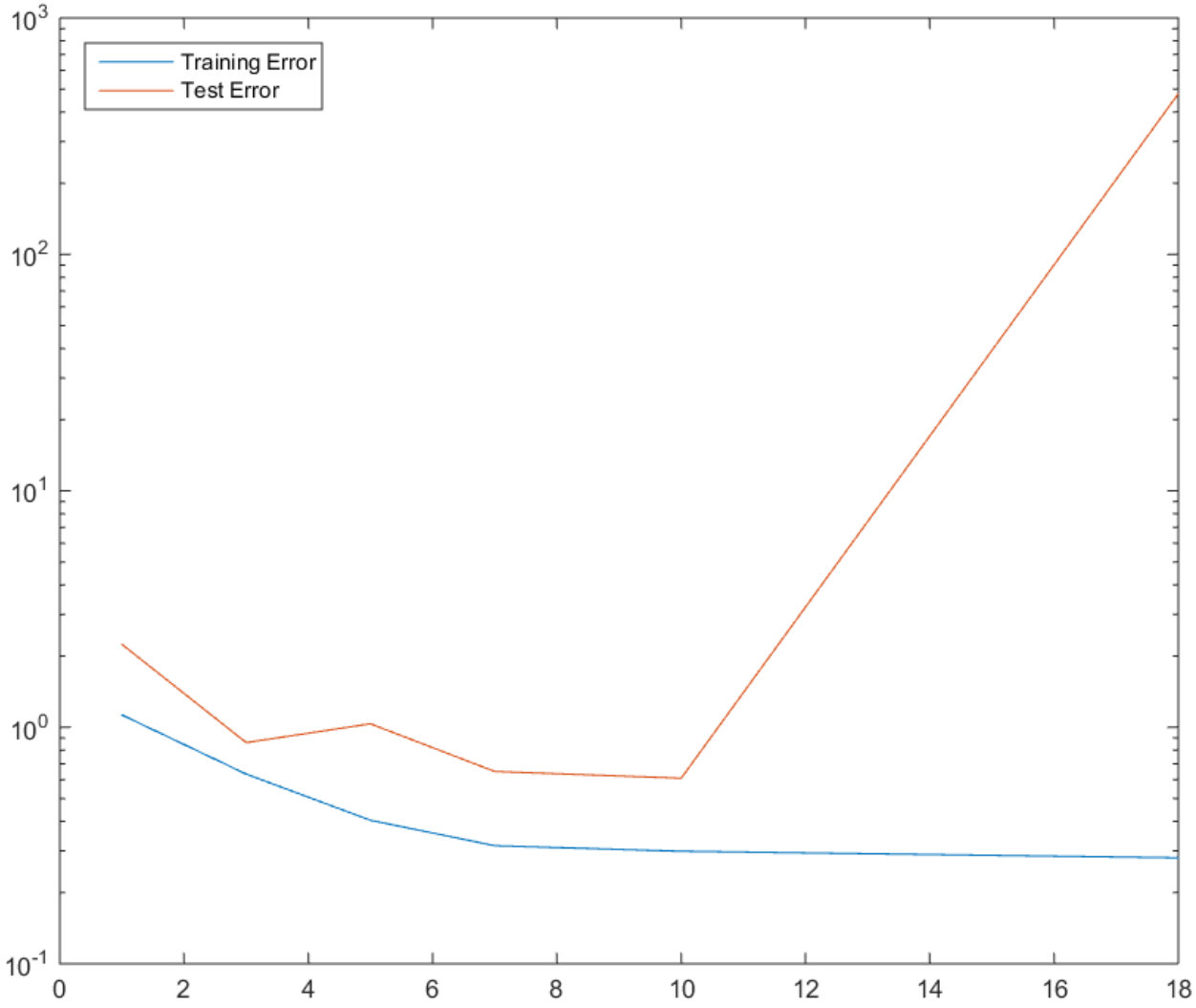





f(x) degree=10
7
6
5
4
3
2
1
0
-1
-2
0
2
4
6
8
10


f(x) degree=18
7
6
5
4
3
2
1
0
-1
-2
0
2
4
6
8
10

Here is the plot of the training and test error. Based on this plot, I would choose a polynomial of degree 10.



This is the code used to accomplish these plots. It is a continuation of the code from part a as it uses the arrays created there.

```
degs = [1 3 5 7 10 18];
YtrError = zeros(1,length(degs));
YteError = zeros(1,length(degs));
xs = (min(X):.05:max(X))'; % densely sample possible x-values

figure

for i = 1:length(degs)

    degree = degs(i);

   % create poly features up to given degree; no "1" feature
    XtrP = fpoly(Xtr, degree, false);

    [XtrP, M,S] = rescale(XtrP); % it's often a good idea to scale the features
    lr = linearRegress( XtrP, Ytr ); % create and train model

    % defines an "implicit function" Phi(x)
    Phi = @(x) rescale( fpoly(x,degree,false), M,S);

    % parameters "degree", "M", and "S" are memorized at the function definition
    % Now, Phi will do the required feature expansion and rescaling:
    YhatTrain = predict( lr, Phi(Xtr) ); % predict on training data
    YhatTest = predict(lr, Phi(Xte) );

    ys = predict( lr, Phi(xs) ); % make predictions at xs

    subplot(3,2,i)
    plot(Xtr,Ytr,'g.','MarkerSize',10);
    hold on
    plot(Xte,Yte,'rx','MarkerSize',10);
    plot(xs,ys)
    axis([0 10 -2 7]);
    title(strcat('f(x) degree=',num2str(degree)));
    hold off

    %now get the training and test error
    YtrError(i) = sum((YhatTrain-Ytr).^2)/length(Ytr);
    YteError(i) = sum((YhatTest-Yte).^2)/length(Yte);

end

%creates the training and test error plots
figure
semilogy(degs,YtrError);
hold on
semilogy(degs,YteError);
legend('Training Error','Test Error','Location','Northwest');
```

Problem 2

This is the plot. As can be observed, the minimum average MSE occurs where $degree = 5$.

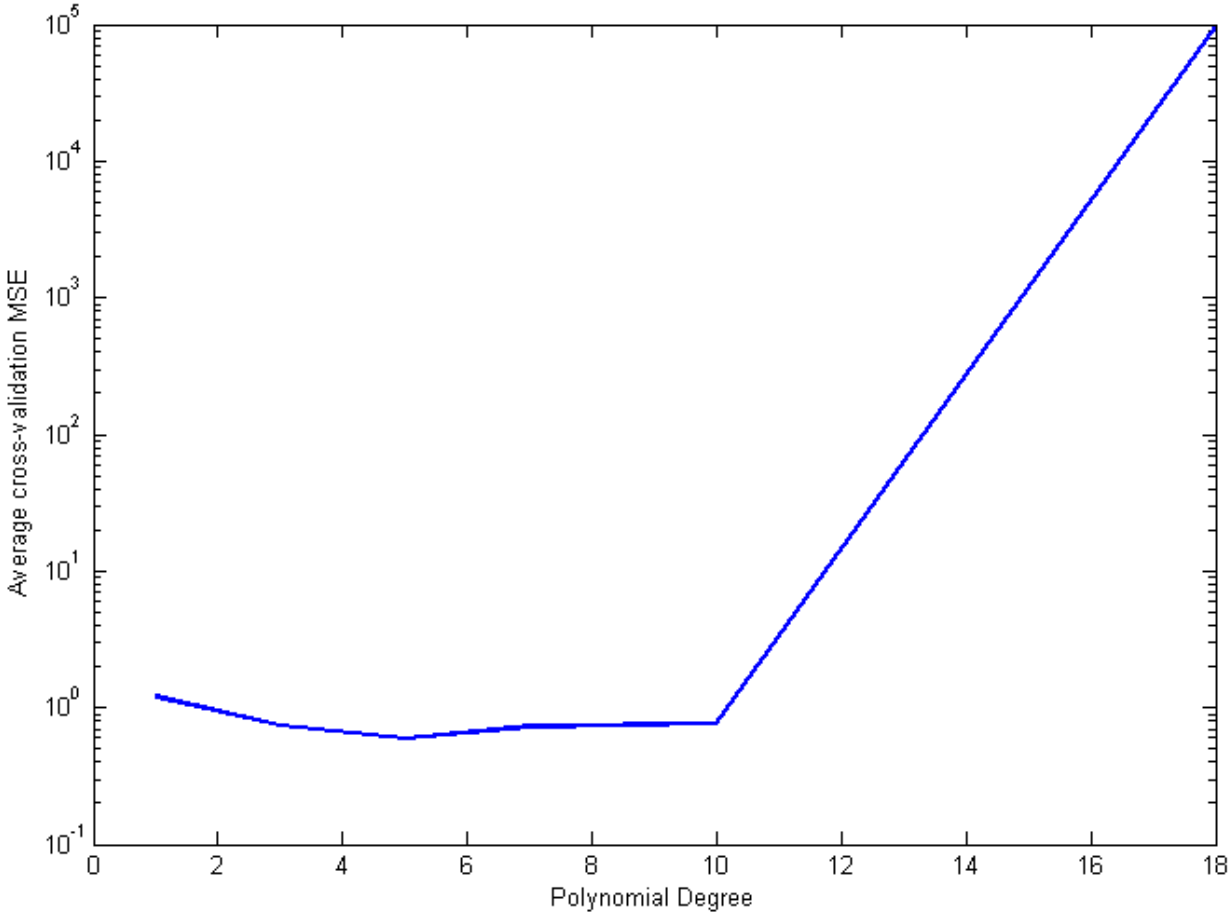


Figure 2: The average cross-validation MSE as function of polynomial degree

The cross-validation error with degree 5 was 0.5911.
The error for the degree 5 polynomial on actual test data was 1.0344 in the previous problem.
Therefore using cross-validation reduced the error that we saw.

It also changed which degree polynomial had the lowest error. On the actual test data, the degree 10 polynomial had the lowest error with an average MSE of 0.6091.

Here is the code that I used to get the numbers and the plot for problem 3

```
%%
iris=load('data/curve80.txt');
y=iris(:,2);
X=iris(:,1);

[Xtr Xte Ytr Yte] = splitData(X,y, .75); % split data into 75/25 train/test

%%
degs = [1 3 5 7 10 18];
crossValidError = zeros(1,length(degs));

nFolds = 5;
J = zeros(1,nFolds);

for k=1:length(degs)

    degree = degs(k);

    for iFold = 1:nFolds,

        % take ith data block as validation
        [Xti,Xvi,Yti,Yvi] = crossValidate(Xtr,Ytr,nFolds,iFold);

        XtiP = fpoly(Xti, degree, false);

        [XtiP, M,S] = rescale(XtiP); % it's often a good idea to scale the features
        lr = linearRegress( XtiP, Yti ); % create and train model
        XviP = rescale( fpoly(Xvi,degree,false), M,S);
        YhatTest = predict(lr, XviP );

        J(iFold) = mean((YhatTest-Yvi).^2);
    end;

    % the overall estimated validation performance is the average of the performance on each fold
    crossValidError(k) = mean(J);
end

semilogy(degs,crossValidError,'b-','LineWidth',2);
xlabel('Polynomial Degree');
ylabel('Average cross-validation MSE');
```