Week 2
CS 273a - Introduction to Machine Learning (Winter '15)*
Prof. Alex Ihler

Varad Meru
Student # 26648958
Due Date: 01/20/2015

Homework 2†‡

Problem 1: Linear Regression

(a) I loaded the data from `data/curve80.data` into the `curve` variable. Then the variable `X` stores the data and `Y` stores the regressed values. I then split the data into a 75-25 partition and store it in the individual variables. Shown in Listing 1.

Listing 1: Fetching the `data/curve80.data` and splitting it with a 75-25 partition.

```
% Fetching the dataset and separating it into X and Y.
curve=load('data/curve80.txt');      % load the text file
y = curve(:,end);             % target value is last column
X = curve(:,1);       % features are other columns
whos

% Part (a)
% Splitting the data into 75-25 split.
[Xtr, Xte, Ytr, Yte] = splitData(X,y, .75);

>> whos
  Name         Size              Bytes  Class     Attributes

  X           80x1                 640  double
  Xte         20x1                 160  double
  Xtr         60x1                 480  double
  Yte         20x1                 160  double
  Ytr         60x1                 480  double
  curve       80x2                1280  double
  y           80x1                 640  double
```

(b) After loading the data, we use the `linearRegress` class's `predict()` to get the prediction of the model. The code is given in Listing 2. The plot can be seen at Figure 1.

Listing 2: Getting a prediction for the model built on the Training data.

```
%% Part (b)
% Training the linear regression with Xtr as an input
lr = linearRegress( Xtr, Ytr );
xs = [0:.05:10]';
ys = predict(lr, xs);
ys1 = predict(lr, Xte);

% Plotting the training data and the predicted function in the same plot.
f=figure;
scatter(Xtr, Ytr, 'filled');
hold on;
plot(xs, ys);
hold off;
saveas(f,'lr.jpg','jpg');

% MSE for both Training and Test data
mse(lr, Xtr, Ytr)
```

*Website: http://sli.ics.uci.edu/Classes/2015W-273a
†Questions available at http://sli.ics.uci.edu/Classes/2015W-273a?action=download&upname=HW2.pdf
‡All the figures and listing numbers are auto-refered.

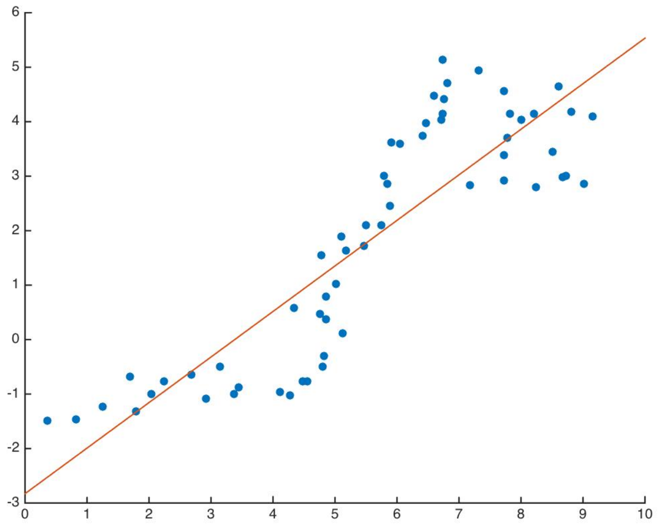


Figure 1: Training data and the learned function in Listing 2

```
% ans = 1.1277
mse(lr, Xte, Yte)
% ans = 2.2423
```

(c) The higher degree polynomials are created as per the Listing 3. The higher-degree polynomials are created by the `fploy()` function provided. Using the new dataset, we rescale it and train the linear regression model using `linearRegress` class.

Listing 3: Getting a prediction for the model built on the Training data.

```
f = figure;
i = 1;
errorsTr = zeros(1, 6);
errorsTe = zeros(1, 6);
d=[1,3,5,7,10,18];
for degree=[1,3,5,7,10,18];

    XtrP = fpoly(Xtr, degree, false); % create poly features up to given ...
        degree; no "1" feature.
    [XtrP, M,S] = rescale(XtrP); % it's often a good idea to scale the ...
        features
    lr = linearRegress( XtrP, Ytr ); % create and train model

    % XS vs YS plot.
    xs = [0:.05:10]';
    xsP = rescale( fpoly(xs,degree,false), M,S);
    ysP = predict( lr, xsP );

    % Scatter plot and the learned function on the data.
    subplot(2,3,i);
    title(strcat({'degree= '},num2str(degree)))
    hold on;
    scatter(Xtr, Ytr, 'filled');
    ax = axis;
    hold on;
    plot(xs, ysP);
    axis(ax);

```

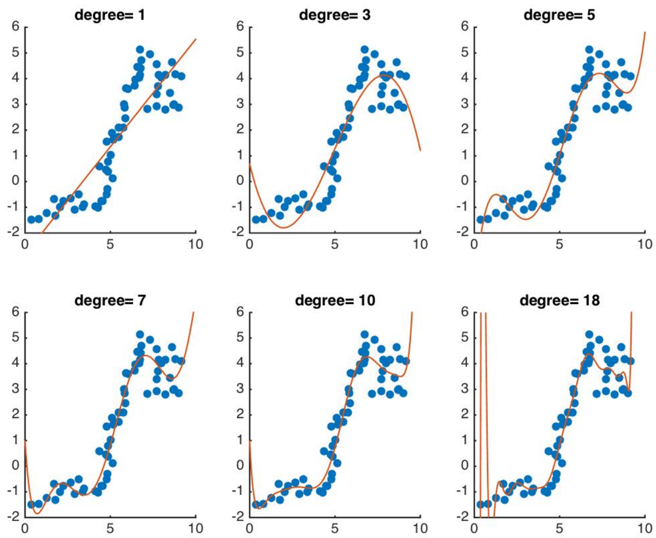


Figure 2: Plots for different degree polynomials and the learned functions

```
    % For getting the right degree to get the best learning function.
    XteP = rescale(fpoly(Xte,degree,false), M,S);
    YteP = predict(lr,XteP);

    % MSE for both Training and Test data
    errorsTr(1, i) = mse(lr, XtrP, Ytr)
    errorsTe(1, i) = mse(lr, XteP, Yte)
    i= i+1;
    % now, apply the same polynomial expansion & scaling transformation ...
        to Xtest:
    %
end;
hold off;
```

The trained functions and the scatter for various degree values can be seen in the visual representation seen in Figure 2. To get to a better approximation of the right degree, we learned the performance of the learner with the test data and plotted it, as can be seen in Figure 3. From this observation, I can see that for **degree = 10**, the test error was the least.

Problem 2: Cross Validation

For each degree, I use cross validation within the loop iterating on all the degrees. Before I apply the cross-validation on this data and change its degree as well as rescale it, as seen in Listing 4. After that, I save the errors into an array and use that to plot, which can be seen in 4(a). From this plot, it can be seen that for **degree = 5**, the error is minimum. When we compare it with the error on the actual test data, the difference shows up. The actual test data fits better for degree = 10, whereas cross validated data fits better with degree = 5. This plot shows that error for cross-validated model follows similar pattern to a test data, thus we can build models on data which would generalize better, without the knowledge of the actual test data.

Listing 4: Training linear regression with cross-validation.

```
i = 1;
nFolds = 5;
```

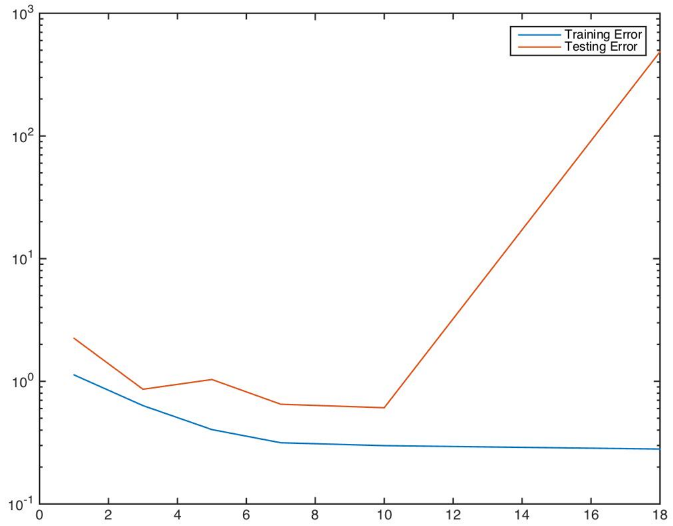
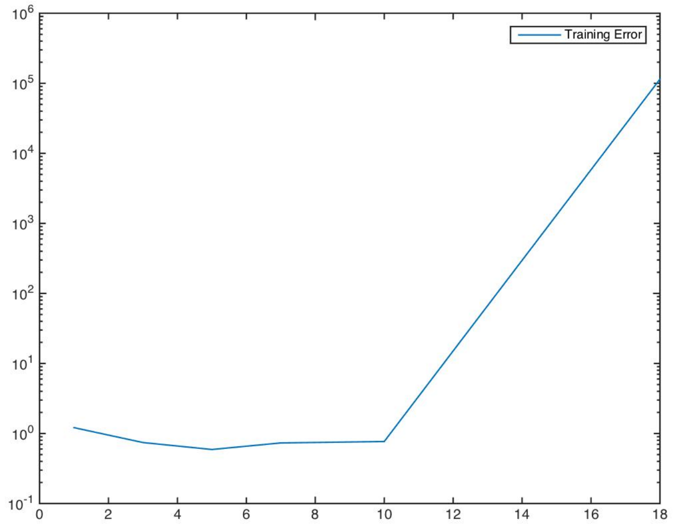


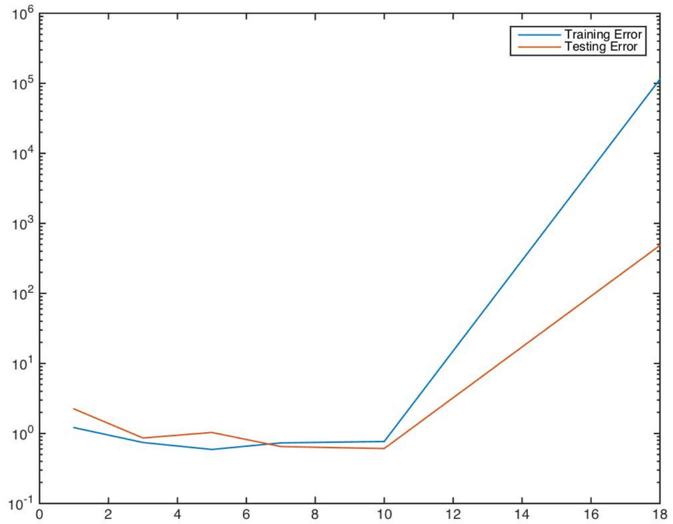
Figure 3: Train and Test Errors.

```
d=[1,3,5,7,10,18];
for degree=[1,3,5,7,10,18];
    % Degrees and scaling of the data
    XtrP = fpoly(Xtr, degree, false); % create poly features up to given ...
        degree; no "1" feature.
    [XtrP, M,S] = rescale(XtrP); % it's often a good idea to scale the features
    for iFold = 1:nFolds;
        [Xti,Xvi,Yti,Yvi] = crossValidate(XtrP,Ytr,nFolds,iFold);
        % take ith data block as v learner = linearRegress(...
        lr = linearRegress( Xti, Yti ); % create and train model
        % TODO: train on Xti, Yti , the data for this fold J(iFold) = ...

        % TODO: now compute the MSE on Xvi, Yvi and save it
        J(iFold) = mse(lr, Xvi, Yvi);
    end;
    % the overall estimated validation performance is the average of the ...
        performance on ea
    errors(i) = mean(J);
    i = i + 1;
end;
```



(a) Training Errors for various degrees when the data is cross-validated



(b) Train Errors on Cross-validated error as well as the Error on the Actual Test data

Figure 4: Running Various Experiments. The three bars are for Brute Force, I-Consistency and I-Consistency with MRV search. All the times given are in Seconds