CS178/273a Homework #2 Solution

Machine Learning & Data Mining: Winter 2015

Problem 1: Linear Regression

```
% (a) Load the data
  rand('state',0); randn('state',0);  % for repeatability
  curve = load('data/curve80.txt'); X=curve(:,1); Y=curve(:,2);
  [Xt Xe Yt Ye]=splitData(X,Y,.75);

% (b) Linear Regression
  lr = linearRegress(Xt,Yt);  % train linear regression model
  xs = [0:.05:10]';           % (note transpose!)
  ys = predict(lr,xs);
  plot(xs,ys,'k-',Xt,Yt,'r.',Xe,Ye,'g.','markersize',18,'linewidth',3);
  ax = axis();

  mse(lr,Xt,Yt),
  %  = 1.1277     % training MSE
  mse(lr,Xe,Ye),
  % = 2.2423      % test/validation MSE
```

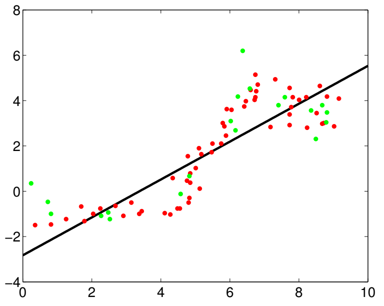


Figure 1: Linear regression on the curve dataset.

```
% (c) Polynomial Features
  degrees = 1:18; errs=[];
  for degree=degrees,

    XtP = fpoly(Xt, degree, false); % create polynomial features up to given degree
    [XtP, M,S] = rescale(XtP);      % scale the features
    Phi = @(x) rescale(fpoly(x,degree,false),M,S);  % implicit function of feature transform
    lr = linearRegress( XtP, Yt );  % create and train model

    plot(xs,predict(lr,Phi(xs)),'k-',Xt,Yt,'r.',Xe,Ye,'g.','markersize',18,'linewidth',3);
    axis(ax);
    errs(:,end+1) = [mse(lr,Phi(Xt),Yt) , mse(lr,Phi(Xe),Ye)]';

  end;
  semilogy(degrees,errs(1,:),'r-',degrees,errs(2,:),'g-','linewidth',3);
  ax=axis();
```

As expected, training error only improves with increasing degree, while the validation accuracy improves for a while, and then worsens sharply. You can see from the plots that much of this is due to poor estimation around the edges of the data, where there are fewer data points to constrain the function values.

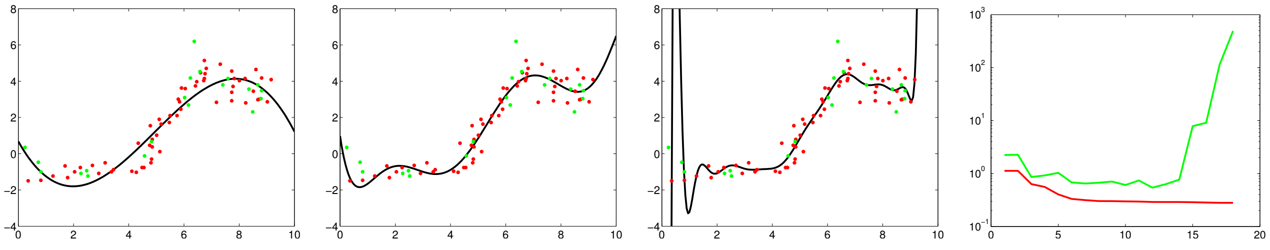


Figure 2: The effect of polynomial degree on both the regression function and its training and test performance.

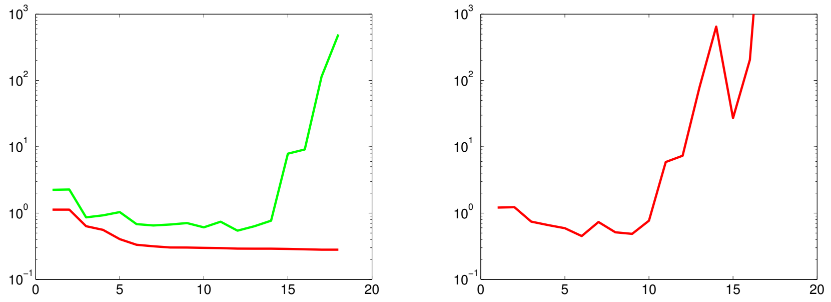
Problem 2: Cross-Validation

We can use cross-validation to assess the correct level of model complexity (here, polynomial degree of the regression). We will imagine that we do not have access to our "test" data X_e, Y_e, that we used for assessment in the previous section, and see if we can assess quality using only the "training" data X_t, Y_t. To this end, we split (again) the data multiple times, one for each of the K partitions, and repeating our entire training and evaluation procedure on each split:

```
% Use 5-fold cross-validation to score each possible degree:

nFolds = 5; degrees = 1:18; errs=zeros(nFolds,max(degrees));
for degree=degrees,             % for each degree
  for iFold = 1:nFolds,         %  and each partition of data:

    % Extract the ith cross-validation partition of the data:
    [Xti,Xvi,Yti,Yvi] = crossValidate(Xt,Yt,nFolds,iFold);

    % Build the feature transform on these training data:
    XtiP = fpoly(Xti, degree, false); % create polynomial features up to given degree
    [XtiP, M,S] = rescale(XtiP);      % scale the features
    Phi = @(x) rescale(fpoly(x,degree,false),M,S);  % implicit function of feature transform

    % Create and train the model, and save the error of this degree & split
    lr = linearRegress( XtiP, Yti );
    errs(iFold,degree) = mse(lr,Phi(Xvi),Yvi);
  end;
end;

errs = mean(errs,1);                  % compute average error over the splits
semilogy(degrees,errs,'r-','linewidth',3);  % plot (again on semilog scale
axis(ax);                             % use the same axis as before for comparision
```

The resulting plot shows a pretty similar characterization to the validation data in the previous problem (which, remember, we didn't access in this problem). A noticable difference is that moderate to high degrees begin to degrade the performance (and begin overfitting) earlier than in the previous problem; mainly, this is due to the 20% decrease in training data within our validation process, which causes slightly lower degrees to begin overfitting. (If you like, you can plot a learning curve to see how performance is changing for these degrees as a function of the number of training data.)

This illustrates how cross-validation can be used for model selection, as a pretty good surrogate for having additional test data.



(a) Train/Test from P1 (b) Cross-validation from P2

Figure 3: Using cross-validation to assess the appropriate model complexity.