Lecture 21: Cross Validation

March 7, 2005

Last time we talked about automated methods for finding the "best" model. We just scratched the surface of this subject. The approaches were all concerned with finding the best "subset" model.

I want to caution you again about believing too much in what is found. Consider that the Bigmac dataset had 10 predictors, and therefore about 1024 sub-models, and our stepwise procedure examined only 1-+9+8+7 = 34 of them and you'll see that we could very easily have overlooked some.

Last time we did backwards stepwise regression, and this time I want to do forward. I oculdn't get the step function in R to work, but there's another way that uses the "add1" function.

The handout shows that for the bigmac data, both methods produce the same model.

However, suppose we used BIC instead. In R, we just change the input to the same function. And, in fact, in this case the result is the same.

this would suggest that, assuming all of the assumptions of the linear model hold for all submodels, the model with the best predictive power is also the most "true".

Another method for comparing models is cross-validation. This method basically says that if you're most interested in making predictions, you should compare models by actually testing them on new data. For example, our movie data attempts to predict grosses based on the first Friday a film is released. We fit several different models for 1999. It would be nice to test it on 2000, or 2005 data and see which model does best?

This brings us back to measuring what "best" means. All of our measures so far are based on the RSS, residual sum of squares. And dividing by n-p gives us the MSE which estimates the amount of "noise" left unfit. A simple approach is just to estimate this MSE based on the "new" data.

The idea in cross validation is to imagine we have two data sets. The training data set is the data we use to fit the model. We do transforms and everything we can to get the model to fit, and then use whatever model-building strategy we like.

Then it's time for the rubber to hit the road, as they say, and see how well the model predicts the

"test" set.

Of course in real life we rarely get two separate data sets. So our solution is to create one from what we have. We take our data, randomly divide it into two groups. One group we call the training set, the other the test set.

This is fine, but one problem is that our estimate of the MSE is based on a small data set (smaller than the original data.)

So an approach is k-fold cross-validation. It looks like this:

1. randomly divide the data into k subsets
2. make the first subset the test data. Combine all of the other pieces and make them the training data.
3. fit the model on the training data
4. use the model to predict the responses with teh test data
5. compute the residuals for the test data to see how you did. Calculate the RSS
6. now repeat the above steps $k-1$ more times Each time a different subset will be the test data.
7. when you're done, you'll have k different estimates of the RSS. You can combine these to form one very good estiamte of the MSE.
8. you also can see how the slopes vary each time.

The DAAGs package of R has a routine that does this. You need to type library(DAAG) before running it. You might have to download the DAAG package (the URL is in the front of your textbook.)

We can see what happens with a dataset comparing the area of a home with the sale price for 15 randomly selected homes.

And also with the bigmac data.