Lecture 7: Regression, Part 2
Modeling Social Data, Spring 2017
Columbia University

Ingil Hwang

March 3, 2017

1 From the Previous Lecture

We have learned the general mathematical formulas and reasoning behind why we use linear regression for the presentation of the data. In many cases, linear regression can be a great tool in presenting the general trend of the data. Using the same data we used at the end of the last lecture, we can see that each dot on the graph is the average for a certain age group. By looking at these data points and the general trend curve, we can make a prediction(which in fact is our end goal in finding the "best" model). We can also manipulate data so that we can select specific data based on different category. After plotting predicted and actual data, we can use different colors to plot the trend curves by gender and age.

2 Evaluating the fit of model

One of the key information in the data we miss out with this trend curve is variation in the data. It might be the case that two different sets of data with almost identical trend curve have drastically different data. This can be due to the difference in the variability, which has the real world implication. So we need to take standard deviation into consideration when we evaluate the fit of model. There are two commonly used formula: root mean square error and Pearson's correlation coefficient.

- Root mean square error (RMSE)

$$\sqrt{\frac{1}{N}\sum_{i}^{N}(y_i - \hat{y})^2}$$

  The RMSD represents the sample standard deviation of the differences between predicted values and observed value.

- Pearson's correlation coefficient (PCC)

$$\frac{\sum_i (y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{[\sum_i (y_i - \bar{y})^2]^{\frac{1}{2}}[\sum_i (\hat{y}_i - \bar{\hat{y}})^2]^{\frac{1}{2}}}$$

  The PCC is a measure of the linear dependence (correlation) between two variables X and Y. In a nutshell, PCC is the covariance of the two variables divided by the product of their standard deviations.

3 Trade-off between Accuracy and Prediction

When fitting a curve to the data, we often have to deal with a question of how much accurate do we want the function to be. While making the curve fit very accurately to the data can look good and seem to be a good representation, a too accurate model can perform badly in predicting future since it is overly sensitive to noise, which is the definition of "overfitting." On the other hand, we would not want our curve to be too loose to the extent that the given data does little in predicting future, "underfitting". This complexity is represented by the order of a function – the higher it is, the more complex the model gets.

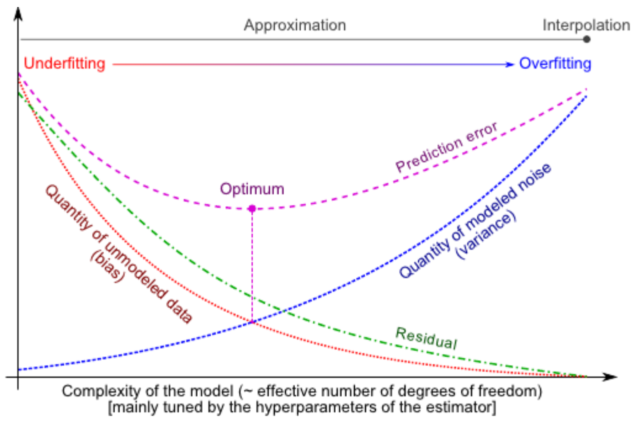

Figure 1: source: http://www.brnt.eu/phd/node14.html

As we can see from the above graph, the optimal point of accuracy happens somewhere in-between overfitting and underfitting.

4 Bias- and Variance Trade-off

$$MSE = Bias^2 + Variance + IrreducibleError$$

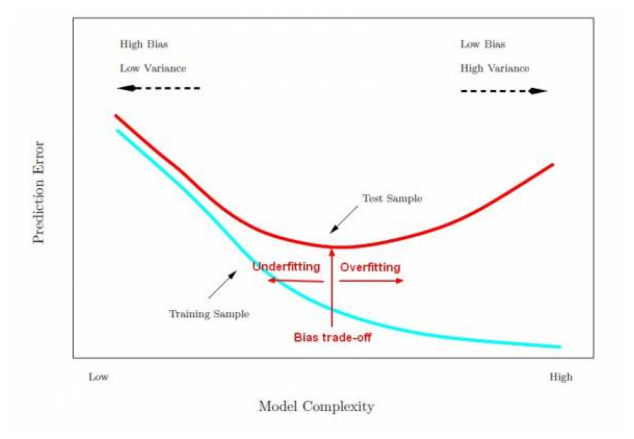

Figure 2: source: http://gerardnico.com/wiki/

Low variance: It does not change a lot as it gets difficult.
High variance: Model changes a lot with different draws of the data.
High Bias: Assumes something that might not be true, so it could always be wrong even with infinite amount of data
Low Bias: Can capture complicated patterns with enough data.

5 Setting complexity

So now we know choosing the right complexity, i.e. degree of functions is important. To evaluate this, we fit the models on the "training set", and test with "validation set". These two sets should be randomly chosen so that we can test the model.
More generally, to find a good model

- randomly split our data into 3 sets
- fit models on the training set
- Use the validation set to find the best model (different data)
- Quote final performance of this model on the test set

One technique we can make a good use of is "K-fold cross validation." We take a subset of data as a validation data out of training data, and run the test. Then we take another subset out of the training data as a new validation data. Do this until every data has been used as a training data once. Finally, average all runs. Beware that we need to make sure the data is randomized. Then we can generate many test cases based on the real examples. We choose the order of the function that is minimum. It does not have to be strictly the minimum, however, as it depends on each test. Rather, find the lowest order that is among the flat low line by looking at it.

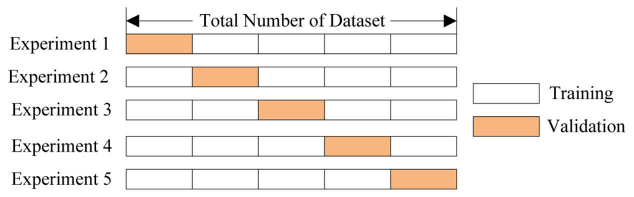

Figure 3: K-fold Cross Validation (source: http://sdsawtelle.github.io/blog/output/week6-andrew-ng-machine-learning-with-python.html)

Another technique is adding a random noise to the existing data and use that as training/validation data. The noise-generating technique is more advance and was briefly discussed during the lecture.

6 Regularization

Lastly, we discussed on ways to penalize complexity of the function.

- Ridge Regularization:
$$\mathcal{L} = \frac{1}{n}\sum_{i}^{n}(y_i - wx_i)^2 + \lambda||w||^2$$
As λ increases, the coefficients decrease and reduces the weight of the model fit.
- Lasso Regularization:
$$\mathcal{L} = \frac{1}{n}\sum_{i}^{n}(y_i - wx_i)^2 + ||w_1|| + ||w_2|| + ...$$
We can ignore near-zero coefficients.