THE UNIVERSITY OF BRITISH COLUMBIA
DEPARTMENT OF STATISTICS

STAT 447B Methods for Statistical Learning (2014/15 Term 1)
Lab 2

24/25 September, 2014

1 Review

- The optimal fit to a penalized least-squares regression with penalty being the integrated squared second derivative of the fitting function (a measure of wiggliness/adaptiveness) is a **natural** cubic spline. It is a cubic spline constrained to be linear beyond the boundary knots.

- In various regression problems we have a tuning parameter λ that controls the adaptiveness of the resulting fit. A method of choosing the optimal λ is **cross-validation** (CV). In performing CV one withholds certain data points when fitting the regression model, and then uses the resulting fit to "predict" the withheld values as if they were future realizations of the underlying process. Two important cases of CV:

  - **Leave-one-out CV**: Only one data point is withheld in each fitting (and then predicted). All data points are withheld once, resulting in a total of n fits if the original data contains n observations. For some linear smoothers it is possible to avoid this potentially time-consuming operation and the fit using all data alone is sufficient.
  - **V-fold CV**: When the data set is very large or the fitting is computationally intensive one may opt for V-fold CV instead. In this method the observations are sliced into V equal portions. At each fitting one portion is withheld and then predicted. This requires only V instead of n fittings. It can be easily seen that leave-one-out CV is equivalent to n-fold CV.

  The ultimate aim of CV is to estimate the **expected** (squared) **prediction error** for **future** observations.

- Another estimation method is **kernel smoother**. It allows local estimation by using only data close to the point being estimated. There are several important components:

  - **Kernel**: A weight function that specifies the relative importance of various data points used in smoothing.
  - **Bandwidth**: The size of the window within which data are considered in estimation. This usually takes the form of a fixed distance or fixed proportion of data (so that we always have the same number of points in each window).
  - **Degree**: The degree of local polynomial used in estimation. The kernel smoothers mentioned in Lecture 5 are of degree zero, in the sense that the fitted value is simply the weighted average of the "neighbours". Using non-zero degree means fitting a weighted polynomial regression around each point (weighted linear, quadratic regression etc). A higher degree results in a more adaptive fit as the fitted function can vary more within each local window. This comes at the price of increasing the variance associated with the fit (again bias-variance tradeoff!).

  Since kernel smoothing is essentially local weighted regression, it is also a linear model at each point of the explanatory variable to be fitted.

2 Data analysis

We consider the motorcycle data this lab, which consists of the measurements of head acceleration at various times after a simulated motorcycle accident. The training and test data sets are available in the lab webpage. Note that there are slightly more observations in the training than the test set. The focus is on applying kernel smoothers with different configurations, using the function `loess`.

2.1 Visualize the data

As always, check how the observations look like before fitting any model. We try to model head acceleration at different times, and hence head acceleration is the response and time is the covariate.

Q1 Describe the pattern of the data. Do the training and test data look very different?

2.2 Kernel smoothing

Q2 Now we perform kernel smoothing on the training data using the function `loess`. Read the documentation for this function; in particular, pay attention to the parameters `span` and `degree`.

1. Fit a kernel smoother to the training data with **local constants** and a **span of 0.1** (i.e. 10% of the data). Use a variable to store the resulting object.
2. Plot the fitted curve. To do this, first create a vector (using `seq`) spanning the range of the values of the covariate. Next, use `predict` with the aforementioned vector as new data. You should then obtain the predicted values which can be plotted against the covariate.
3. What is your observation about the fitted curve? Calculate the **predictive error sum of squares** by predicting on the test data using the fitted model.

Hint: Recall that `?predict` will not give you anything useful. What is the correct function here?

Q3 Repeat **Q2** but with a span of 0.7. How does this fitted curve compare with the one with a span of 0.1? And what is its predictive error sum of squares?

Q4 **Submit your answer and R code to this part on *Connect***

For each value of span between 0.1 and 0.7 with increment 0.02, find the predictive error sum of squares for the resulting fit, again using local constants. Hence choose the optimal model and briefly explain why you choose it.

Hint: A `for` loop will save you a lot of time here.

Q5 (Optional during the lab) Keep span at 0.7. However, instead of using local constants, try using local linear/quadratic fit. How do the fitted curves differ?

2.3 Cross-validation

In the previous section we were actually doing cross-validation since both the training and test observations come from the same data set (and I split them manually). Try to modify the code so that it truly performs a V-fold cross-validation. These are the steps involved:

1. Randomly split the data into V equal portions (or roughly equal if n is not divisible by V).
2. Perform the following for each portion i, $i = 1, \ldots, V$:

(a) Fit the model without portion i.

(b) Using the model fitted in (a), predict the observations in portion i and calculate the sum of squared errors m_i.

3. Record the total sum of squared errors $t = m_1 + \cdots + m_V$.

4. Repeat the above procedures for other candidate span values; the one that gives the smallest t wins.

This CV method is very general and is applicable to almost all regression and classification models.