10 Cross Validation

Suppose we must choose between two possible ways to fit some data. How do we choose between them? Simply measuring how well they fit they data would mean that we always try to fit the data as closely as possible — the best method for fitting the data is simply to memorize it in big look-up table. However, fitting the data is no guarantee that we will be able to **generalize** to new measurements. As another example, consider the use of polynomial regression to model a function given a set of data points. Higher-order polynomials will always fit the data as well or better than a low-order polynomial; indeed, an $N-1$ degree polynomial will fit N data points exactly (to within numerical error). So just fitting the data as well as we can usually produces models with many parameters, and they are not going to generalize to new inputs in almost all cases of interest.

The general solution is to evaluate models by testing them on a new data set (the "test set"), distinct from the training set. This measures how **predictive** the model is: Is it useful in new situations? More generally, we often wish to obtain empirical estimates of performance. This can be useful for finding errors in implementation, comparing competing models and learning algorithms, and detecting over or under fitting in a learned model.

10.1 Cross-Validation

The idea of empirical performance evaluation can also be used to determine model parameters that might otherwise to hard to determine. Examples of such model parameters include the constant K in the K-Nearest Neighbors approach or the σ parameter in the Radial Basis Function approach.

Hold-out Validation. In the simplest method, we first partition our data randomly into a "training set" and a "validation set." Let K be the unknown model parameter. We pick a set of range of possible values for K (e.g., $K = 1, ..., 5$). For each possible value of K, we learn a model with that K on the training set, and compute that model's error on the validation set. For example, the error on validation set might be just the squared-error, $\sum_i ||y_i - f(x_i)||^2$. We then pick the K which has the smallest validation set error. The same idea can be applied if we have more model parameters (e.g., the σ in KNN), however, we must try many possible combinations of K and σ to find the best.

There is a significant problem with this approach: we use less training data when fitting the other model parameters, and so we will only get good results if our initial training set is rather large. If large amounts of data are expensive or impossible to obtain this can be a serious problem.

N**-Fold Cross Validation.** We can use the data much more efficiently by N-fold cross-validation. In this approach, we randomly partition the training data into N sets of equal size and run the learning algorithm N times. Each time, a different one of the N sets is deemed the test set, and the model is trained on the remaining $N-1$ sets. The value of K is scored by averaging the error across the N test errors. We can then pick the value of K that has the lowest score, and then learn model parameters for this K.

A good choice for N is $N = M - 1$, where M is the number of data points. This is called **Leave-one-out cross-validation**.

Issues with Cross Validation. Cross validation is a very simple and empirical way of comparing models. However, there are a number of issues to keep in mind:

- The method can be very time-consuming, since many training runs may be needed. For models with more than a few parameters, cross validation may be too inefficient to be useful.
- Because a reduced dataset is used for training, there must be sufficient training data so that all relevant phenomena of the problem exist in both the training data and the test data.
- It is safest to use a random partition, to avoid the possibility that there are unmodeled correlations in the data. For example, if the data was collected over a period of a week, it is possible that data from the beginning of the week has a different structure than the data later in the week.
- Because cross-validation finds a minimum of an objective function, over- and under-fitting may still occur, although it is much less likely. For example, if the test set is very small, it may prefer a model that fits the random pattern in the test data.

Aside:
Testing machine learning algorithms is very much like testing scientific theories: scientific theories must be predictive, or, that is, falsifiable. Scientific theories must also describe plausible models of reality, whereas machine learning methods need only be useful for making decisions. However, statistical inference and learning first arose as theories of scientific hypothesis testing, and remain closely related today.

One of the most famous examples is the case of planetary motion. Prior to Newton, astronomers described the motion of the planets through onerous tabulation of measurements — essentially, big lookup tables. These tables were not especially predictive, and needed to updated constantly. Newton's equations of motion — which could describe the motion of the planets with only a few simple equations — were vastly simpler and yet also more effective at predicting motion, and became the accepted theories of motion.

However, there remained some anomolies. Two astronomers, John Couch Adams and Urbain Le Verier, thought that these discrepancies might be due to a new, as-yet-undiscovered planet. Using techniques similar to modern regression, but with laborious hand-calculation, they independently deduced the position, mass, and orbit of the new planet. By observing in the predicted directions, astronomers were indeed able to observe a new planet, which was later named Neptune. This provided powerful validation for their models.

Incidentally, Adams was an undergraduate working alone when he began his investigations.

Reference: http://en.wikipedia.org/wiki/Discovery_of_Neptune