Instance Based Learning

Most of the algorithms that we encounter in this course can be categorized as *eager learners*. Decision trees, regression, neural networks, SVMs, Bayes nets: all of these can be described as eager learners. In these models, we fit a function that best fits our training data; when we have new inputs, the input's features are fed into the function, which produces an output. Once a function has been computed, the data could be lost to no detriment of the model's performance (until we obtain more data and want to recompute our function). For eager learners, we take the time to learn from the data first and sacrifice local sensitivity to obtain quick, global-scale estimates on new data points.

Here, we look at an example of a *lazy learner* in the *k*-nearest neighbors algorithm. In contrast to eager learners, lazy learners do not compute a function to fit the training data before new data is received. Instead, new instances are compared to the training data itself to make a classification or regression judgment. Essentially, the data itself is the function to which new instances are fit. While the memory requirements are much larger than eager learners (storing all training data versus just a function) and judgments take longer to compute than eager learners, lazy learners have advantages in local-scale estimation and easy integration of additional training data.

Mathematically, what is the *k*-nearest neighbors algorithm? Given a parameter k and a distance or similarity function d, for a new instance q we obtain the algorithm output as a measure of centrality over the k-closest items in the training data (i.e. the items with the k-smallest $d(q,x_i)$). For classification tasks, the basic measure of centrality is the mode of the *k*-nearest neighbors; for regression tasks, the basic operation is the mean.

To work, *k*-nearest neighbors expects that the data has locality and smoothness. Data points that are close to one another in distance are expected to have similar value. The underlying function that determines a data point's output value is also expected to be smooth. *k*-nearest neighbors also suffers from the curse of dimensionality, the idea that as the number of features grow, the amount of data required for accurate generalization grows exponentially.

While the algorithm is quite simple, there is still a lot of latitude in choices of *k*, *d*, and central measure that can affect the model's performance. When *k* is small, models have high bias, fitting on a strongly local level. Larger *k* creates models with lower bias but higher variance, smoothing out the influence of individual data points on output judgements. Choice of *d* can go beyond selecting a basic distance metric (e.g. taxicab distance vs. euclidean distance); weights may be placed on factors if there is reason to believe certain factors are more or less important to classification or regression. (The distance function must still remain a valid distance metric, however.) Reducing the weight on factors can also help alleviate the effects of dimensionality on the algorithm's performance. Similarly, the measure of centrality does not need to be a simple mode or mean; averages may utilize weights on training instances as a function of their distance or similarity to the test instance. Two points in a region with the same neighbors need not have the same regression value.

For regression tasks, there is a natural extension from *k*-nearest neighbors with weighted mean to locally weighted regression (LWR). If we have test points that fall outside of the domain of the training data, *k*-nearest neighbors does not provide a way to extrapolate beyond the range observed in the training data. LWR combines the ideas of traditional regression with instance-based learning's sensitivity to training items with high similarity to the test point. In standard regression, we select coefficients for a regression function that minimize the sum of squared deviation between the observed data and the regression judgement, creating a single function that operates globally over the full range of data. With LWR, the squared deviations are weighted by a kernel function that decreases with distance, such that for a new test instance, a regression function is found for that specific point that emphasizes fitting close-by points and ignoring the pull of far-away points. In this way, new points outside the domain of the training set can take novel values outside of the training set's range.

The plots at the bottom of this page demonstrate the progression of estimates from 1-NN to LWR for some sample one-dimensional regression problems.

This article discusses the bias-variance tradeoff in model selection and uses two-dimensional *k*-NN as an example; there are a number of interactive figures that show how the algorithm changes depending on choice of *k*.