Oak Ridge National Lab PIC Math Project: Feature selection when the sample size is small

Industrial Liason: Dr. Robert A. Bridges, lab scientist

Suppose we want to do a regression problem. A publicly available dataset for such a problem can be found here: http://ama.liglab.fr/

The data describes TomsHardware forum social interaction. Basically, the features are things like “how many other discussions were spawned from a given one” and “how many users interacted in this discussion”. There are 96 such features, and the target to be predicted is a metric (real number) describing how impactful that discussion was. Such machine learning + social networking research is a big area for social understanding, predictive analysis, marketing, etc. This project focuses on the machine learning, not on the social impacts or pushing the state of the art in understanding online forums.

TomsHardware has 28k such data points, each with 96 (input) features, and 1 target (output). The goal is to predict the target form the features, which is a regression problem. We are seeing (possibly noisy) samples from a function from R^96 to R and want to learn the function. Are there ways to combine features or reduce the number of features to do more accurate predictions? Suppose we only had 100 such data points (say randomly sample 100 of them). Now most models will have too many parameters for the number of data points — they’ll be overfit. How do we figure out what features or say combination of features should be used to reduce the number of inputs to the model? One common technique is correlation analysis. For each feature (coordinate direction in the input domain), compute its correlation with the target (the output of the function we’re trying to learn) across all the data. Keep only those features with the highest absolute correlation. L1 regression is common, but may require a large data set. Auto-encoding and PCA are unsupervised methods that don’t consult the output (target) changes, so they’re less ideal for such a problem.

Main question: When faced with a small sample of high-dimensional data, what method should scientists use to determine the most important features or combinations of old features? Compare and modify novel methods against methods like correlation analysis and PCA to decide when the new method does and doesn’t work.

As suggested above, use a subset of 100 data points from the TomsHardware data. To test various methods, for each fixed regression model run 10-fold cross validation and measure the mean absolute error across the folds. What features give the best model? We also suggest looking at your trained model’s accuracy on the training set and the test set for each fold — if it’s really good on the training set, but poor on the testing set then you’ve overtrained.

I suggest using python with sci-kit learn (`sklearn`) http://scikit- and sci-kit neuralnetwork (`sknn`) http://scikit-. These are well documented machine learning packages, with sk-learn set up for beginners. Further, easy and quick data sets for testing are included (esp. the Boston housing data set for regression), and 10-fold cross validation testing are one-liners to load. Learning what model parameters to use is often done by grid searches (grid searches refer to simply trying all possible combinations), and this can be done in sk-learn as well. Both sknn and sci-kit learn have the `.get_params` method, which gives a python dictionary of the parameters used for that model. This will allow programmatically saving meta-data (what is the model configuration). Lastly, python has the `time` module. Calling `time.time()` just before and after training will allow users to see training time.