Parallel Random KNN Classification and Regression with Feature Selection

Shengqiao Li[1,*], Donald A. Adjeroh[2] and E. James Harner[3]

1. West Virginia University/PNC Bank
2. Lane Department of Computer Science and Electrical Engineering, West Virginia University
3. Department of Statistics, West Virginia University
*Contact author: lishengqiao@yahoo.com

Keywords: Statistical Learning, K Nearest Neighbor, High Dimensional Data, Parallel Computing

Random KNN (RKNN) generalizes the classical nearest-neighbor statistical learner. Random KNN consists of an ensemble of base k-nearest neighbor models, each constructed from a random subset of the input variables. A collection of r such base classifiers is combined to build the final Random KNN classifier. Since the base classifiers can be computed independently of one another, the overall computation is *pleasantly parallel*.

Random KNN can be used to select key features using the RKNN-FS algorithm. RKNN-FS is an innovative feature selection procedure for "small n, large p problems." Empirical results on microarray data sets with thousands of variables and relatively few samples show that RKNN-FS is an effective feature selection approach for high-dimensional data. RKNN is similar to Random Forests (RF) in terms of classification accuracy without feature selection. However, RKNN provides much better classification accuracy than RF when each method incorporates a feature-selection step. RKNN is significantly more stable and robust than Random Forests for feature selection when the input data are noisy and/or unbalanced. Further, RKNN-FS is much faster than the Random Forests feature selection method (RF-FS), especially for large scale problems involving thousands of variables and/or multiple classes.

Random KNN and feature selection algorithms are currently implemented in an R package rknn on CRAN, which supports both classification and regression. The time complexity of the algorithm, including feature selection, is $O(rkpn \log n)$, assuming the number of variables randomly selected in a base classifier is $m = \log p$. This choice of m, in contrast to $\sqrt{p}$, reduces the time complexity from exponential time to linear time. However, it is important to choose r sufficiently large to ensure adequate variable coverage. By paralleling the code in rknn, the time can be reduced linearly depending on the number of cores or compute nodes.