STAT 542 Fall 2015
Report on Comparative Analysis of Machine Learning Algorithms

Ka Ki Lai (kakilai2)

November 04,2015

I. Linear Discriminant Analysis (LDA) and Naïve Bayes (NB)

1. Linear Discriminant Analysis (LDA)
   LDA is a linear method for classification assuming observations are drawn from Guassian distribution. It performs classification by looking at the log of ratio of posterior probabilities of each class. Since LDA assumes the classes have different means but equal covariance, the linear log-odd function will give a linear decision boundary for the class.

   For this method, the lda function from Package MASS was used. By estimating the means and covariance by MLE, no turning parameters has been used.

2. Parametric and Non Parametric Naïve Bayes (Par-NB, NonPar-NB)
   Naïve Bayes conduct classification by the Bayes rule by the independent assumption of all features in each class. The Parametric Naïve Bayes assumes Gaussian Distribution of the feature to find the means and variances of each class, while in the non parametric Naïve Bayes , there is no assumption on distribution density. Instead, it directly estimates the probability of an observation belong to a class by some density estimation, such as Kernel density.

   For this method, NaiveBayes function in package *e1071* is used for parametric method. Since *e1071* can only perfrom parametric method, NaïveBayes functions is packed *klaR* was therefore used for non-parametric method. (Both parametric and non-parametric method are available from the package *klaR*.). On the other hand, the parametric Naïve Bayes has been self-coded and the result are compared(Par-NB(Self)). There is no tuning parameters in this method.
3. Repeating method 1 and 2 by applying PCA
   Principal component analysis (PCA) computes the principal components which are the linear combination of the p features, where p is the space dimension of the observations. The PCAs are then used to understand the data. Here, PCA was conducted on the data and LDA, Naïve Bayes (both parametric and nonparametric) are repeated using just the top PC directions which explain 99% of the total variation in the data. For this method, the function princomp is used to conduct PCA, and 114 PCs has been selected for repeating the analysis.

The result of the above methodology has been summarized below:

Method	Without PCA				With PCA			
	LDA	Par-NB	Par-NB(self)	NonPar-NB	LDA	Par-NB	Par-NB(self)	NonPar-NB
Training Error	0.005	0.1375	0.10	0.045	0.011667	0.0375	0.045	0.015
Testing Error	0.06024	0.1657	0.13855	0.1145	0.04518	0.06927	0.06928	0.0873

Table 1: Summary of Error for Problem 1

Analysis: In general, LDA performs better than Naïve Bayes in terms of testing error. With PCA, the testing error for both LDA and Naïve Bayes have been improved. The test error of Naïve Bayes with my own code is close to the one conducted by package, while the self-code Naïve Bayes performs slightly better (0.1657 for package and 0.13855 for self-coded)

II. Regression

1. Logistic Regression with full model (LR)

   Logistic Regression conducts classification by modelling the posterior probabilities of the classes by logit transformation of the linear functions of independent variables x. This ensures the total sum of probabilities of different classes equal to 1.

   In this method, the glm function in the package glmnet is used, with family = 'binomial'. There is no tuning parameter.

2. Logistic Regression with forward selection via AIC and BIC

   Next, logistic regression is performed again using AIC and BIC criteria to select models with a subset of features. The step functions in R are used to do so by setting direction = 'forward', and the 'k' set for AIC and BIC are 2 and log(n) respectively, where n is number of observations. There is no tuning parameter. The variables selected for AIC and BIC are the same: (which includes: V167, V,261, V169, V85, V189, V281, V83, V188, V137, V335, V268, V163, V237, V14, V38, V203, V182, V339, V329, V74, V107). Among this set of variables, there is no noisy feature.
3. Logistic Regression with Lasso
   The lasso regression used the L1 norm of the coefficient vector as the penalty function for model complexity. Hence, simpler models are selected with Lasso. The function cv.glmnet is used in order to perform cross-validation to select the turning parameter. The penalty parameter is chosen by lambda.1se (0.003672) which is the largest value of lambda so that the cv error is within 1 standard error of the minimum cv error. 95 variables are selected for this model which is much more than the AIC and BIC models and the variables selected are quite different. There are some noisy features selected too, for example, V430, V443, V452 etc.

The results are summarized in the table below:

Logistic Regression				
Method	Full	AIC	BIC	Lasso
Training Error	0	0	0	0.0025
Testing Error	0.08434	0.0542	0.0542	0.04819
Variables selected	457	21	21	95

Table 2: Summary of Error

III. SVM based on PCA

1. Support Vector Machines by linear, quadratic and Guassian kernels (SVM)
   SVM conducts classification by enlarging the feature space using kernels to find the decision boundaries so that the margin between the classes could be maximized. By different kernel (linear, quadratic and Guassian), SVM of different features of decision boundary would be generated to perform classifications. In this method, the svm function in R package e1071 is used. Since SVM does not perform feature selection, I use the 114 PCs selected from Problem 1 instead of all the original features to conduct svm. To choose the optimal tuning parameter gamma and cost, the tune.svm command is used.

The results are summarized in the table below:

SVM based on PCA			
Kernels	Linear	Quadratic	Gaussian
Training Error	0	0	0
Testing Error	0.0512	0.0301	0.0301

Table 3: Summary of Error

IV. Tree Models

Classification Tree
Classification Tree predicts each of the observations to belong to the most commonly occurring class of training observations in the region to which it belongs to. Recursive binary splitting is used to grow the classification tree. The criterion of binary splits is the classification error rate which is calculated as the ratio of the training observations that do no belong to the most common class of that region. Usually, a large tree is first grown and then pruned to the size such that the tree has minimal too cross validation error. In this method, the *tree* function in the *tree* package is used.

By 10-fold cross validation , and setting the minimum deviance as 0.0005, the optimal size of the tree is chosen as 9. The impurity measure I use to grow the tree is the default one, deviance , and the impurity measure to prune the tree is misclassification rate,the tree has been pruned into size 9.

The pruned tree has use 21 variables to build the tree and the variables used are very similar to the results selected by forward AIC and BIC in problem 2. For example, both algorithms select variables V167, V169, V85, V329 and etc. Similar to forward AIC and BIC, the tree does not use any noise features.

The results are summarized in the table below:

Training Error	0.02417
Testing Error	0.05722

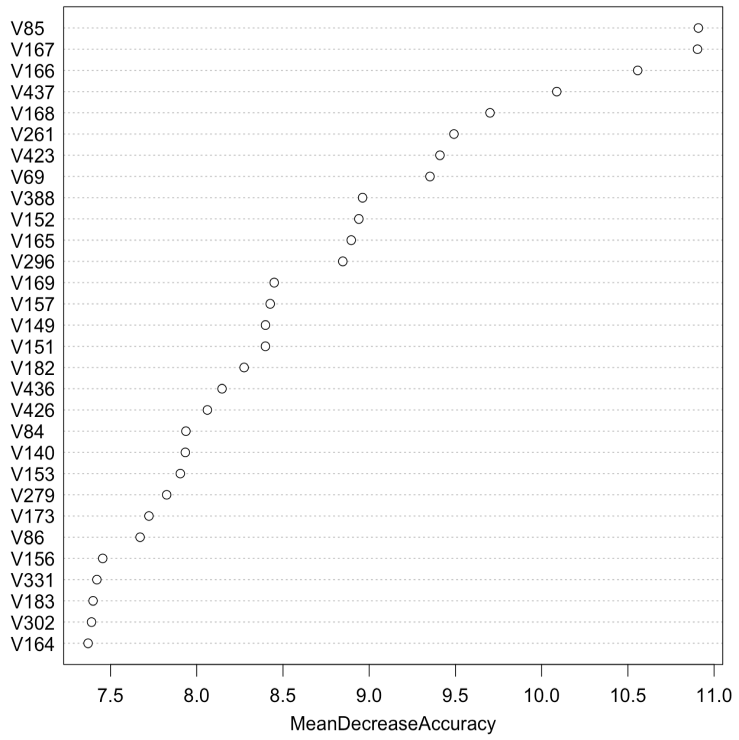
Table 4: Summary of Error for Problem 4

V. Random Forest

Random forests build a large number of decision trees by bootstrapping training samples. For every split of the tree, a random sample of m predictors is used as the candidate for split instead of the full set of all p predictors. In other words, in a specific split, there is a fresh sample of m predictors and only predictors in the sample could be used for the split. This way of classification decorrelates the trees built to decrease the variability and increase the reliability of the result. In this method, the randomForest in the R package randomForest is used.

In this algorithm, 500 trees were built. The tuning parameter m is set to be square root of total number of predictor which is 21. This is the typical way of choosing m for classification by Random Forest.

The Variable importance plot is generated by the varImpPlot function , and is plotted as below:



The variables selected by random Forest are similar to those selected by AIC and BIC, while there are a few noise features.

The results are summarized in the table below:

Training Error	0
Testing Error	0.03012

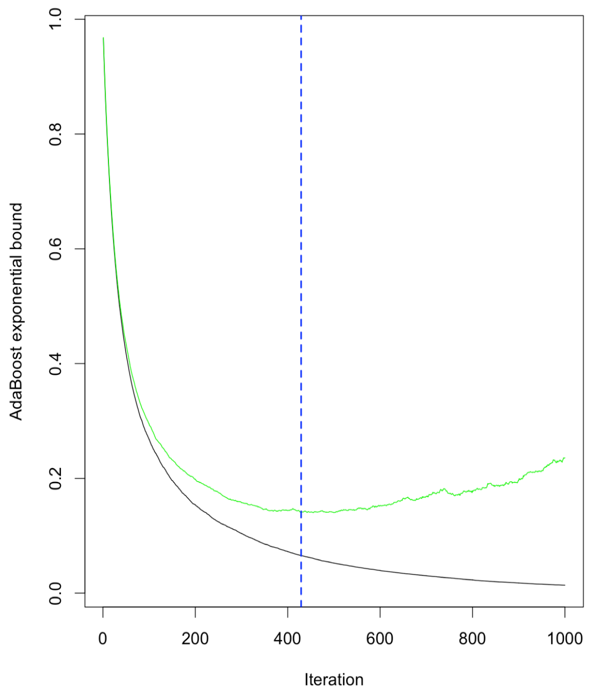
Table 5: Summary of Error for

VI. gbm package

The algorithm Boosting tries to improve the prediction resulting from a decision tree. It builds the tree sequentially by grown each new tree according to the information from previous grown trees. The gbm function in the gbm package is used to perform the Boosting algorithm.

Initially, 1000 boosting iterations were conducted. The value for shrinkage is 0.05 which is the rate which the boosting learns. To resample the data, the bag fraction is set as 0.5, which means 0.5 of the training observations are randomly selected to propose the tree in next iteration.

To choose the best number of observations, 5-fold CV was conducted. Based on the result of the following performance plot



the optimal numbers would be chosen for the one with lowest bound of error. According to the R output, the optimal iterations number would be 429 in this case.

The algorithm is then rerun with iterations and the results are summarized in the table below:

Training Error	0.0025
Testing Error	0.04819

Table 6: Summary of Error