STAT 542 Fall 2015
Homework Assignment 3
Report

Ka Ki Lai (kakilai2)

November 04,2015

Problem 1

1. Linear Discriminant Analysis (LDA)

LDA is a linear method for classification assuming observations are drawn from Guassian distribution:

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left(-\left(\frac{1}{2}\right)(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)\right)$$

It performs classification by looking at the log of ratio of posterior probabilities of each class and LDA assumes the classes have different means but equal covariance. Hence, the linear log-odd function will give a linear decision boundary for the class:

$$\log\frac{\Pr(G=k|x)}{\Pr(G=l|x)} = \log\frac{\pi_k}{\pi_l} - \left(\frac{1}{2}\right)(\mu_k+\mu_l)^T\Sigma^{-1}(\mu_k+\mu_l) + x^T\Sigma^{-1}(\mu_k+\mu_l)$$

where $\Pr(G=k|X=x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l}$ which is based on the Bayes Theorem.

Hence, observations were classified to the class *k* that maximizes the discriminant function:

$$\delta_k(x) = x^T\hat{\Sigma}^{-1}\mu_k - \frac{1}{2}\mu_k^T\hat{\Sigma}^{-1}\mu_k + \log\pi_k$$

where the mean of k is estimated by centroid of class k and covariance is estimated by pooled in class covariance matrix.

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 based on the Bayes rule : $\Pr(G=k|X=x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l}$ with 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* was used for parametric method. Since *e1071* can only perform parametric method, NaïveBayes functions in

package *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 are no tuning parameters in this method. To avoid the zero standard error of some of the variables, I have set the zero standard error of those variables to be a standard error of very small value, 1e-6.

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) were repeated using just the top PC directions which explain 95% 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	0.0958	0.045	0.011667	0.0375	0.045	0.015
Testing Error	0.06024	0.1657	0.13855	0.1145	0.04518	0.06928	0.06928	0.0873

ble 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)

Problem 2

1. Logistic Regression with full model (LR)

   Logistic regression model:

   $\Pr(G = K \mid X = x) = \frac{1}{1+\sum_{l=1}^{K-1}\exp(\beta_{l0}+\beta_l^T x)}$,

   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. The model parameters are estimated by MLE.

   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, to perform model selection, logistic regression was performed again using AIC and BIC criteria to select models with a subset of features. The selected model tries to minimize objective function [-2logL = kp], where L stands for the likelihood function, p is the number of parameters and k = 2 for AIC and log(n) for BIC, where n stand for the sample size.

   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 estimate: $\hat{\beta}^{\text{lasso}} = \underset{\beta}{\text{argmin}} \{\frac{1}{2}\sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda \sum_{j=1}^{p}|\beta_j|\}$

   The lasso regression used the L1 norm $\sum_1^p |\beta_j|$ of the coefficient vector as the penalty function for model complexity. Because of the L1 penalty, some coefficient will become 0. 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. Besides, there are some noisy features selected, 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 for Problem 2

Problem 3

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. Specifically, the coefficients would be estimated by minimizing certain objective functions. These objective function minimizes the trade-off between margin and error.

   Giving a training data ($x_1 y_1 \ldots x_n y_n$), SVM classifier predict new point $x^*$by:

   sign(f(x^*))=sign($\Sigma_i \alpha_i (y_i) K(x_i, x^*) + \hat{\beta}_0$

   where K is the kernel function, and $0 < \alpha_i < C$ for all x_i

   Here, the following kernel functions were used:

   K(x,x') = <h(x),h(x')>

   For Quadratic kernel: $K(x,x') = (1+<x,x'>)^2$

   For Gaussian kernel: $K(x,x') = \exp(-\gamma \|x-x'\|^2)$

   By different kernels (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, gamma is chosen as 0.01 and cost is chosen as 10.
   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	0.03012	0.03012

Table 3: Summary of Error for Problem 3

Problem 4 Tree Models

1. 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.

   Answer to (b):

   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.

2. Answer to (c)
   The pruned tree has use 10 variables to build the tree and the variables used somewhat similar to the results selected by forward AIC and BIC in problem 2. 5 over 10 variables chosen were chosen by AIC and BIC too.For example, both algorithms select variables V167, V169, V85, V329 and V182. 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

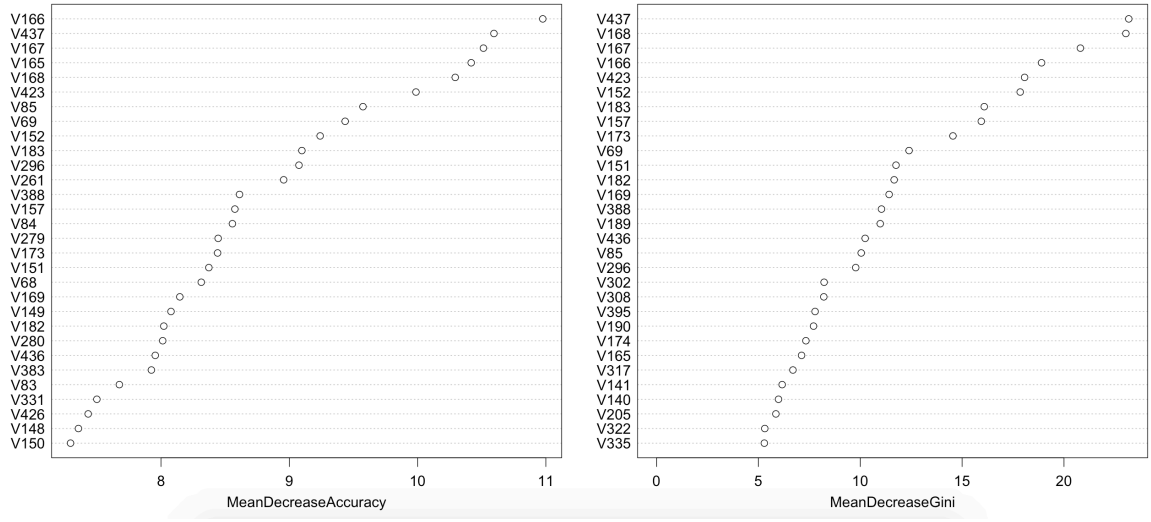
Table 4: Summary of Error for Problem 4

Problem 5 randomForest

1. 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.

2. Answer to (a)
   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.

3. Answer to (b)
   The Variable importance plot is generated by the varImpPlot function which meansure importance of variable based on mean accuracy and Gini Index , and is plotted as below:



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

The results are summarized in the table below:

Training Error	0
Testing Error	0.03012

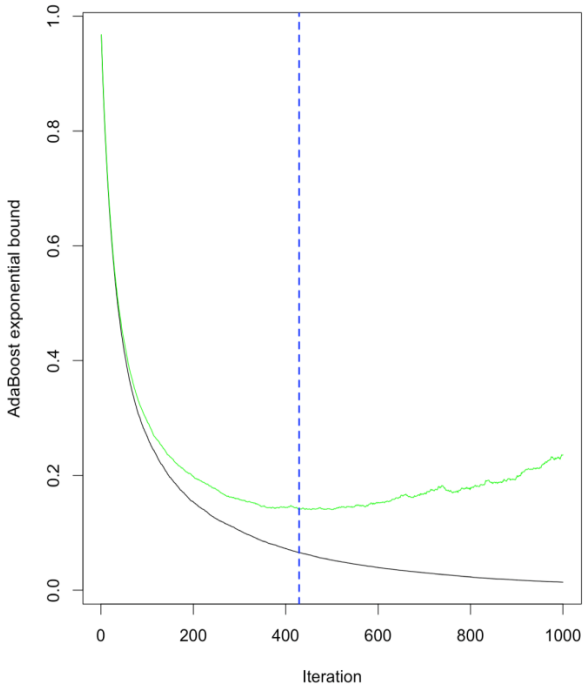
Table 5: Summary of Error for Problem 5

Problem 6 gbm package

1. 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.

2. Answer to (a)
   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.

3. Answer to (b)
   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.

4. Answer to (c)

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 for Problem 6