Homework 2

PSTAT 131/231, Winter 2018

Due on February 18, 2017 at 11:00 pm

Spam detection with spambase dataset

Following packages are needed below:

```r
library(tidyverse)
library(tree)
library(plyr)
library(randomForest)
library(class)
library(rpart)
library(maptree)
library(ROCR)
```

Data Info: The Data Set was obtained by the UCI Machine Learning database. From the website,

> The "spam" concept is diverse: advertisements for products/web sites, make money fast schemes, chain letters, pornography...
>
> Our collection of spam e-mails came from our postmaster and individuals who had filed spam. Our collection of non-spam e-mails came from filed work and personal e-mails, and hence the word 'george' and the area code '650' are indicators of non-spam. These are useful when constructing a personalized spam filter. One would either have to blind such non-spam indicators or get a very wide collection of non-spam to generate a general purpose spam filter.

Dataset `spambase.tab` can be read with the following code. Next, standardize each numerical attribute in the dataset. Each standardized column should have zero mean and unit variance.

```r
spam <- read_table2("spambase.tab", guess_max=2000)  ## ?read_table2 to find out about guess_max
spam <- spam %>%
    mutate(y = factor(y, levels=c(0,1), labels=c("good","spam"))) %>%   # label as factors
    mutate_at(.vars=vars(-y), .funs=scale)                              # scale others
```

Attribute Information: The last column of 'spambase.tab' denotes whether the e-mail was considered spam (1) or not (0), i.e. unsolicited commercial e-mail. Most of the attributes indicate whether a particular word or character was frequently occuring in the e-mail. The run-length attributes (55-57) measure the length of sequences of consecutive capital letters. For the statistical measures of each attribute, see the end of this file. Here are the definitions of the attributes:

- 48 continuous real [0,100] attributes of type `word_freq_WORD` = percentage of words in the e-mail that match `WORD`, i.e. 100 * (number of times the `WORD` appears in the e-mail) / total number of words in e-mail. A `WORD` in this case is any string of alphanumeric characters bounded by non-alphanumeric characters or end-of-string.
- 6 continuous real [0,100] attributes of type `char_freq_CHAR` = percentage of characters in the e-mail that match `CHAR`, i.e. 100 * (number of `CHAR` occurences) / total characters in e-mail
- 1 continuous real [1,...] attribute of type `capital_run_length_average` = average length of uninterrupted sequences of capital letters

- 1 continuous integer [1,. . .] attribute of type `capital_run_length_longest` = length of longest uninterrupted sequence of capital letters
- 1 continuous integer [1,. . .] attribute of type `capital_run_length_total` = sum of length of uninterrupted sequences of capital letters = total number of capital letters in the e-mail
- 1 nominal {0,1} class attribute of type `spam` = denotes whether the e-mail was considered spam (1) or not (0), i.e. unsolicited commercial e-mail.

Classification Task: We will build models to classify emails into good vs. spam.

In this dataset, we will apply several classification methods and compare their training error rates and test error rates. We define a new function, named `calc_error_rate()`, that will calculate misclassification error rate. Any error in this homework (unless specified otherwise) imply misclassification error.

```
calc_error_rate <- function(predicted.value, true.value){
  return(mean(true.value!=predicted.value))
}
```

Throughout this homework, we will calculate the error rates to measure and compare classification performance. To keep track of error rates of all methods, we will create a matrix called `records`:

```
records = matrix(NA, nrow=3, ncol=2)
colnames(records) <- c("train.error","test.error")
rownames(records) <- c("knn","tree","logistic")
```

Training/test sets: Split randomly the data set in a train and a test set:

```
set.seed(2)
test.indices = sample(1:nrow(spam), 1000)
spam.train=spam[-test.indices,]
spam.test=spam[test.indices,]
```

9-fold cross-validation: Using `spam.train` data, 9-fold cross validation will be performed throughout this homework. In order to ensure data partitioning is consistent, define `folds` which contain fold assignment for each observation in `spam.train`.

```
nfold = 9
set.seed(2)
folds = seq.int(nrow(spam.train)) %>%       ## sequential obs ids
    cut(breaks = nfold, labels=FALSE) %>%   ## sequential fold ids
    sample                                  ## random fold ids
```

K-Nearest Neighbor Method

1. **(Selecting number of neighbors)** Use 9-fold cross validation to select the best number of neighbors `best.kfold` out of eleven values of k in `kvec = c(1, seq(10, 50, length.out=9))`. Use the folds defined above and use the following `do.chunk` definition in your code. Again put `set.seed(2)` before your code. What value of k leads to the smallest estimated test error?

```r
do.chunk <- function(chunkid, folddef, Xdat, Ydat, k){

  train = (folddef!=chunkid)

  Xtr = Xdat[train,]
  Ytr = Ydat[train]

  Xvl = Xdat[!train,]
  Yvl = Ydat[!train]

  ## get classifications for current training chunks
  predYtr = knn(train = Xtr, test = Xtr, cl = Ytr, k = k)

  ## get classifications for current test chunk
  predYvl = knn(train = Xtr, test = Xvl, cl = Ytr, k = k)

  data.frame(train.error = calc_error_rate(predYtr, Ytr),
             val.error = calc_error_rate(predYvl, Yvl))
}
```

2. **(Training and Test Errors)** Now that the best number of neighbors has been determined, compute the training error using `spam.train` and test error using `spam.test` for the $k =$ `best.kfold`. Use the function `calc_error_rate()` to get the errors from the predicted class labels. Fill in the first row of `records` with the train and test error from the `knn` fit.

Decision Tree Method

3. **(Controlling Decision Tree Construction)** Function `tree.control` specifies options for tree construction: set `minsize` equal to 6 (the minimum number of observations in each leaf) and `mindev` equal to 1e-6. See the help for `tree.control` for more information. The output of `tree.control` should be passed into `tree` function in the `control` argument. Construct a decision tree using training set `spam.train`, call the resulting tree `spamtree`. `summary(spamtree)` gives some basic information about the tree. How many leaf nodes are there? How many of the training observations are misclassified?

4. **(Decision Tree Pruning)** We can prune a tree using the `prune.tree` function. Pruning iteratively removes the leaves that have the least effect on the overall misclassification. Prune the tree until there are only 10 leaf nodes so that we can easily visualize the tree. Use `draw.tree` function from the `maptree` package to visualize the pruned tree. Set `nodeinfo=TRUE`.

5. In this problem we will use cross validation to prune the tree. Fortunately, the `tree` package provides and easy to use function to do the cross validation for us with the `cv.tree` function. Use the same fold partitioning you used in the KNN problem (refer to `cv.tree` help page for detail about `rand` argument). Also be sure to set `method=misclass`. Plot the misclassification as function of tree size. Determine the optimal tree size that minimizes misclassification. **Important**: if there are multiple tree sizes that have the same minimum estimated misclassification, you should choose the smallest tree. This reflects the idea that we want to choose the simplest model that explains the data well. Show the optimal tree size `best.size.cv` in the plot.

6. **(Training and Test Errors)**

   We previous pruned the tree to a small tree so that it could be easily visualized. Now, prune the original tree to size `best.size.cv` and call the new tree `spamtree.pruned`. Calculate the training error and test error when `spamtree.pruned` is used for prediction. Use function `calc_error_rate()` to compute misclassification error. Also, fill in the second row of the matrix `records` with the training error rate and test error rate.

Logistic regression

7. In binary classification context, let p represent probability of class label 1, which imply that $1-p$ represents probability of class label 0. *Logistic function* is the cumulative distribution function of logistic distribution, which maps a real number z to the open interval $(0, 1)$:

$$p(z) = \frac{e^z}{1+e^z}. \tag{1}$$

It is easy to see that when $z \to -\infty$, function $p(z) \to 0$, and as $z \to \infty$, function $p(z) \to 1$.

Show that the inverse of logistic function is the *logit function*:

$$z(p) = \ln\left(\frac{p}{1-p}\right). \tag{2}$$

Logit link function is commonly used as a link function in binary classification. To see the function of the link function, suppose we use linear model to represent the unobserved quantity z. That is $z = \beta_0 + \beta_1 x_1$, for example. In such setup, z represents a log odds ratio that is being modeled with a linear model. Therefore, $z \to -\infty$ implies $p \to 0$ (predict class label 0) and, similarly, $z \to \infty$ implies $p \to 1$ (predict class label 1).

8. Use logistic regression to perform classification. Logistic regression specifically estimates the probability that an observation as a particular class label. We can define a probability threshold for assigning class labels based on the probabilties returned by the `glm` fit.

In this problem, we will simply use the "majority rule". If the probability is larger than 50% class as spam. Fit a logistic regression to predict spam given all other features in the dataset using the `glm` function. Estimate the class labels using the majority rule and calculate the training and test errors. Add the training and test errors to the third row of `records`. Print the full `records` matrix. Which method had the lowest misclassification error on the test set?

Receiver Operating Characteristic curve

9. (ROC curve) We will construct ROC curves based on the predictions of the *test* data from the model defined in `spamtree.pruned` and the logistic regression model above. Plot the ROC for the test data for both the decision tree and the logistic regression on the same plot. Compute the area under the curve for both models (AUC). Which classification method seems to perform the best by this metric?

Hints: In order to construct the ROC curves one needs to use the vector of predicted probabilities for the test data. The usage of the function `predict()` may be different from model to model.

- For trees the matrix of predicted probabilities (for Good and Spam) will be provided by using

```
predict(tree.model, test.data, type="vector")
```

- For logistic regression one needs to predict type `response`

```
predict(glm.obj, test.data, type="response")
```

10. In the SPAM example, "positive" means "spam". In this context, what kind of an email would be considered a false positive email? Also, interpret the ROC curve in plain words. If you are the designer of a spam filter, how do you balance the trade-offs? Argue your case.

Problems below for 231 students only

11. A multivariate normal distribution has density

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

In quadratic discriminant analysis with two groups we use Bayes rule to calculate the probability that Y has class label "1": i.e.

$$Pr(Y = 1 \mid X = x) = \frac{f_1(x)\pi_1}{\pi_1 f_1(x) + \pi_2 f_2(x)},$$

where $\pi_2 = 1 - \pi_1$ is the prior probability of being in group 2. Suppose we classify $\hat{Y} = k$ whenever $Pr(Y = k \mid X = x) > \tau$ for some probabilty threshold τ and that f_k is a multivariate normal density with covariance Σ_k and mean μ_k. Note that for a vector x of length p and a $p \times p$ symmetric matrix A, $x^T Ax$ is the *vector quadratic form* (the multivariate analog of x^2). Show that the decision boundary is indeed quadratic by showing that $\hat{Y} = 1$ if

$$\delta_1(x) - \delta_2(x) > M(\tau)$$

where

$$\hat{\delta}_k(x) = -\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) - \frac{1}{2}\log|\Sigma_k| + \log\pi_k$$

and $M(\tau)$ is some function of the probability threshold τ. What is the decision threshold, M(1/2), corresponding to a probability threshold of 1/2?

Questions 12-13 relate to `algae` dataset. Get the dataset `algae.txt` from the homework archive file, and read it with the following code:

```r
algae <- read.table("algae.txt", header=T, na.strings = "NA")
```

In homework 1 and homework 2, we investigated basic exploratory data analysis for the `algae` dataset. One of the explaining variables is `a1`, which is a numerical attribute. In homework 2, we conducted linear regression for variable `a1` using other 8 chemical variables and 3 categorical variables. Here, after standardization, we will transform `a1` into a categorical variable with 2 levels: high and low, and conduct classification predictions using those 11 variables.

12. **(Variable Standardization and Discretization)** Improve the normality of the the numerical attributes by taking the square root of all chemical variables. *After* square root transformation, impute missing values using the median method from homework 1. Transform the variable `a1` into a categorical variable with two levels: high if a1 is greater than 14, and low if a1 is smaller than or equal to 14.

13. **Linear and Quadratic Discriminant Analysis**

    a. In LDA we assume that $\Sigma_1 = \Sigma_2$. Use LDA to predict whether `a1` is high or low using the `MASS::lda()` function. The `CV` argument in the `MASS::lda` function uses Leave-one-out cross validation LOOCV) when estimating the fitted valuess to avoid overfitting. Set the `CV` argument to true. Plot an ROC curve for the fitted values.

    b. Quadratic discriminant analysis is strictly more flexible than LDA because it is not required that $\Sigma_1 = \Sigma_2$. In this sense, LDA can be considered a special case of QDA with the covariances constrained to be the same. Use a quadratic discriminant model to predict the `a1` using the function `MASS::qda`. Again setting `CV=TRUE` and plot the ROC on the same plot as the LDA ROC. Compute the area under the ROC (AUC) for each model. To get the predicted class probabilities look at the value of `posterior` in the `lda` and `qda` objects. Which model has better performance? Briefly explain, in terms of the bias-variance tradeoff, why you believe the better model outperforms the worse model?