machine-learning

Asem Berkalieva

12/1/2018

Machine Learning Goal: Classify based on Endo vs. No endo

Importing data

```
library(caret)  # contains most of the prediction functions we'll use
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(rpart)  # needed for the plotting
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(e1071)

# # read in the core gene data
core_data <- read.csv("../../../data/genbank/06-twofold-genes.csv")
core_class <- core_data[,4] # obtain endo classifications
core_exp <- core_data[,6:387] # expression data
core_genes <- cbind(core_class, core_exp)
colnames(core_genes)[1] <- "endo"

# read in training set
train <- read.csv("../../../data/machine-learning/mann-whitney/model-8020.csv")
train_class <- train[,4]
train_exp <- train[,6:387]
endo_train <- cbind(train_class, train_exp)
colnames(endo_train)[1] <- "severity"

# read in test set
test <- read.csv("../../../data/machine-learning/mann-whitney/validation-8020.csv")
test_class <- test[,4]
test_exp <- test[,6:387]
endo_test <- cbind(test_class, test_exp)
colnames(endo_test)[1] <- "severity"
```

kNN

```
# define tuning parameters
fitControl <- trainControl(method = "cv", number = 5)
values.k <- data.frame(k = 1:94)

# run kNN model with 5-fold CV and various k's
set.seed(4747)
trainKNN <- train(severity ~ ., data = endo_train,
                  method = "knn", trControl = fitControl,
                  tuneGrid = values.k)

# output results to find optimal k
trainKNN$finalModel
```

```
## 13-nearest neighbor model
## Training set outcome distribution:
##
##     Endometriosis Non-Endometriosis
##                64                54
```

Optimal model is k = 13. The associated model-building accuracy with this is 0.43321032.

We apply this model to the test set of 15 observations. Our test set accuracy is 0.6333333.

```
# obtain accuracy on the OUTER test set with k = 51 model
set.seed(47)
predict_knn <- confusionMatrix(data = predict(trainKNN, newdata = endo_test),
                reference = endo_test$severity)

# print test accuracy
knn_accuracy <- predict_knn$overall[1]
knn_accuracy
```

```
##  Accuracy
## 0.6333333
```

Random Forests

```
# set tuning method and parameter
control <- trainControl(method = "oob")
mtry.tune <- data.frame(mtry = 1:100)

# run and optimize random forest with ntree = 250
# optimal mtry = 6
# OOB error = 28.57%
set.seed(4747)
rf.250 <- train(severity ~ ., data = endo_train, method = "rf",
                trControl = control, na.action = na.roughfix,
                tuneGrid = mtry.tune, ntree = 250, importance = TRUE)
rf.250$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 250, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 250
## No. of variables tried at each split: 65
##
##         OOB estimate of  error rate: 27.97%
## Confusion matrix:
##                   Endometriosis Non-Endometriosis class.error
## Endometriosis                49                15   0.2343750
## Non-Endometriosis            18                36   0.3333333
```

```
# run with ntree = 300
# optimal mtry = 10
# OOB error = 27.82 %
set.seed(4747)
rf.300 <- train(severity ~ ., data = endo_train, method = "rf",
                trControl = control, na.action = na.roughfix,
                tuneGrid = mtry.tune, ntree = 300, importance = TRUE)
rf.300$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 300, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 300
## No. of variables tried at each split: 70
##
##         OOB estimate of  error rate: 26.27%
## Confusion matrix:
##                   Endometriosis Non-Endometriosis class.error
## Endometriosis                50                14   0.2187500
## Non-Endometriosis            17                37   0.3148148
```

```
# run with ntree = 400
# optimal mtry = 17
# OOB error = 27.07
set.seed(4747)
rf.400 <- train(severity ~ ., data = endo_train, method = "rf",
```

```
                trControl = control, na.action = na.roughfix,
                tuneGrid = mtry.tune, ntree = 400, importance = TRUE)
rf.400$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 400, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 400
## No. of variables tried at each split: 46
##
##         OOB estimate of  error rate: 26.27%
## Confusion matrix:
##                   Endometriosis Non-Endometriosis class.error
## Endometriosis                50                14   0.2187500
## Non-Endometriosis            17                37   0.3148148
```

```
# run with ntree = 500
# optimal mtry = 17
# OOB error = 27.82 %
set.seed(4747)
rf.500 <- train(severity ~ ., data = endo_train, method = "rf",
                trControl = control, na.action = na.roughfix,
                tuneGrid = mtry.tune, ntree = 500, importance = TRUE)
rf.500$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 500, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 68
##
##         OOB estimate of  error rate: 26.27%
## Confusion matrix:
##                   Endometriosis Non-Endometriosis class.error
## Endometriosis                50                14   0.2187500
## Non-Endometriosis            17                37   0.3148148
```

```
# run with ntree = 550
# optimal mtry = 17
# OOB error = 29.32 %
set.seed(4747)
rf.550 <- train(severity ~ ., data = endo_train, method = "rf",
                trControl = control, na.action = na.roughfix,
                tuneGrid = mtry.tune, ntree = 550, importance = TRUE)
rf.550$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 550, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 550
## No. of variables tried at each split: 34
##
##         OOB estimate of  error rate: 27.97%
```

```
## Confusion matrix:
##                   Endometriosis Non-Endometriosis class.error
## Endometriosis                49                15   0.2343750
## Non-Endometriosis            18                36   0.3333333
```

```
# run with ntree = 600
# optimal mtry = 8
# OOB error = 30.08 %
set.seed(4747)
rf.600 <- train(severity ~ ., data = endo_train, method = "rf",
                trControl = control, na.action = na.roughfix,
                tuneGrid = mtry.tune, ntree = 600, importance = TRUE)
rf.600$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 600, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 600
## No. of variables tried at each split: 68
##
##         OOB estimate of  error rate: 26.27%
## Confusion matrix:
##                   Endometriosis Non-Endometriosis class.error
## Endometriosis                50                14   0.2187500
## Non-Endometriosis            17                37   0.3148148
```

```
# run with ntree = 700
# optimal mtry = 17
# OOB error = 29.32 %
set.seed(4747)
rf.700 <- train(severity ~ ., data = endo_train, method = "rf",
                trControl = control, na.action = na.roughfix,
                tuneGrid = mtry.tune, ntree = 700, importance = TRUE)
rf.700$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 700, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 700
## No. of variables tried at each split: 22
##
##         OOB estimate of  error rate: 25.42%
## Confusion matrix:
##                   Endometriosis Non-Endometriosis class.error
## Endometriosis                50                14   0.2187500
## Non-Endometriosis            16                38   0.2962963
```

```
# run with ntree = 800
# optimal mtry = 17
# OOB error = 27.82 %
set.seed(4747)
rf.800 <- train(severity ~ ., data = endo_train, method = "rf",
                trControl = control, na.action = na.roughfix,
                tuneGrid = mtry.tune, ntree = 800, importance = TRUE)
```

```
rf.800$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 800, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 800
## No. of variables tried at each split: 37
##
##         OOB estimate of  error rate: 26.27%
## Confusion matrix:
##                   Endometriosis Non-Endometriosis class.error
## Endometriosis                49                15   0.2343750
## Non-Endometriosis            16                38   0.2962963
```

```
# run with ntree = 900
# optimal mtry = 17
# OOB error = 27.82 %
set.seed(4747)
rf.900 <- train(severity ~ ., data = endo_train, method = "rf",
                trControl = control, na.action = na.roughfix,
                tuneGrid = mtry.tune, ntree = 900, importance = TRUE)
rf.900$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 900, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 900
## No. of variables tried at each split: 40
##
##         OOB estimate of  error rate: 27.12%
## Confusion matrix:
##                   Endometriosis Non-Endometriosis class.error
## Endometriosis                48                16   0.2500000
## Non-Endometriosis            16                38   0.2962963
```

```
# run with ntree = 1000
# optimal mtry = 17
# OOB error = 28.57 %
set.seed(4747)
rf.1000 <- train(severity ~ ., data = endo_train, method = "rf",
                trControl = control, na.action = na.roughfix,
                tuneGrid = mtry.tune, ntree = 1000, importance=TRUE)
rf.1000$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 1000, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 61
##
##         OOB estimate of  error rate: 27.97%
## Confusion matrix:
##                   Endometriosis Non-Endometriosis class.error
```

```
## Endometriosis                48                16   0.2500000
## Non-Endometriosis            17                37   0.3148148
```

We found that a Random Forests model with ntree = 400 and mtry = 17 works optimally. The associated model building error with those parameters is 0.7444.

```
# build the final model with mtry = 17 and ntree = 400
rf.final <- rf.250
```

To confirm that this optimization worked, I will apply the same parameters using the RandomForests() function rather than the **caret** package. We obtain a similar OOB error rate/accuracy.

```
# try using Random Forest function
set.seed(47)
rf.function <- randomForest(severity ~ ., data = endo_train, mtry = 17, ntree = 400, importance = TRUE)
rf.function
```

```
##
## Call:
##  randomForest(formula = severity ~ ., data = endo_train, mtry = 17,      ntree = 400, importance = TR
##                Type of random forest: classification
##                      Number of trees: 400
## No. of variables tried at each split: 17
##
##         OOB estimate of  error rate: 27.12%
## Confusion matrix:
##                   Endometriosis Non-Endometriosis class.error
## Endometriosis                50                14   0.2187500
## Non-Endometriosis            18                36   0.3333333
```

Apply this to our test data to obtain the test accuracy. We obtain an accuracy of 86.67%.

```
set.seed(47)
predict_rf <- confusionMatrix(data = predict(rf.final, newdata = endo_test),
                reference = endo_test$severity)
predict_rf$table
```

```
##                    Reference
## Prediction          Endometriosis Non-Endometriosis
##   Endometriosis                 8                 9
##   Non-Endometriosis             5                 8
```

```
# print test accuracy
rf_accuracy <- predict_rf$overall[1]
rf_accuracy
```

```
##  Accuracy
## 0.5333333
```

SVM

We use linear SVM because the relationship between the Endo vs no Endo is that one is two times the other (a linear separation). We have to tune the cost: 'C' parameter, which tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.

Single SVM on entire data

```
set.seed(47)

# create vector for costs
cost <- c(0.001, 0.01, 0.1, 1, 5, 10, 100, 1000)

# fit linear support vector classifier
svmL <- train(severity ~ ., data = endo_train, method="svmLinear",
                 trControl = trainControl(method = "cv", number = 5),
                 tuneGrid = expand.grid(C = cost),
                 preProcess = c("center", "scale"))

# output the results
svmL
```

```
## Support Vector Machines with Linear Kernel
##
## 118 samples
## 382 predictors
##   2 classes: 'Endometriosis', 'Non-Endometriosis'
##
## Pre-processing: centered (382), scaled (382)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 94, 94, 94, 94, 96
## Resampling results across tuning parameters:
##
##   C      Accuracy   Kappa
##   1e-03  0.6946970  0.3785725
##   1e-02  0.8643939  0.7259785
##   1e-01  0.8462121  0.6892289
##   1e+00  0.8462121  0.6892289
##   5e+00  0.8462121  0.6892289
##   1e+01  0.8462121  0.6892289
##   1e+02  0.8462121  0.6892289
##   1e+03  0.8462121  0.6892289
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.01.
```

```
svmL$finalModel
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
```

```
##  parameter : cost C = 0.01
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 74
##
## Objective Function Value : -0.3384
## Training error : 0.016949
```

Optimal cost parameter of 0.01. The associated model-building error is 0.7442.

```
predict_svm <- confusionMatrix(data = predict(svmL, newdata = endo_test),
                reference = endo_test$severity)
predict_svm$table
```

```
##                     Reference
## Prediction          Endometriosis Non-Endometriosis
##   Endometriosis                 9                 3
##   Non-Endometriosis             4                14
```

```
# print test accuracy
svm_accuracy <- predict_svm$overall[1]
svm_accuracy
```

```
##  Accuracy
## 0.7666667
```

```
# ANOTHER WAY
# predict the test data
predict_SVM <- predict(svmL, endo_test)
confusion_matrix <- table(endo_test$severity, predict_SVM)

# calculate test error rate
linearError <-  (confusion_matrix[1,2] + confusion_matrix[2,1]) / sum(confusion_matrix)
linearError
```

```
## [1] 0.2333333
```

We obtain an optimal SVM model with cost, C = 0.01. The training accuracy associated with this cost parameter is 0.8667. When applied to the test set, we obtain a test error rate of 0.20.

SVM on 10-fold test validation

Now, we use Edie's code to 10-fold CV on test data.

```
cost <- c(0.001, 0.01, 0.1, 1, 5, 10, 100, 1000)

ten_fold <- sapply(1:100, function(t) {
  random_ix <- sample(x=c(1:148), size=round(148*0.8), replace=FALSE)
  random_ix_2 <- setdiff(c(1:148), random_ix)

  train_core <- core_genes[random_ix,]
  test_core <- core_genes[-random_ix,]

  trainSVM <- train(endo ~ ., data = train_core, method = "svmLinear",
                trControl = trainControl(method = "none"),
                tuneGrid = expand.grid(C = 0.10),
```

```
                preProcess = c("center", "scale"))

  test.predict <- predict(trainSVM, test_core)

  confusion_matrix <- table(test_core$endo, test.predict)

  linearError <-  (confusion_matrix[1,2] + confusion_matrix[2,1]) / sum(confusion_matrix)
  linearError
})

mean(ten_fold)
```

```
## [1] 0.2223333
```