Models With Transformed Data

Harpeth Lee

2023-05-02

```
train <- read.csv("/Users/johnlee/Stat254/Project/Data/poker-hand-training-transformed.csv")
test <- read.csv("/Users/johnlee/Stat254/Project/Data/poker-hand-test-transformed.csv")

train$hand <- as.factor(train$hand)
test$hand <- as.factor(test$hand)
```

Random Forest

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

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

```
mtry <- sqrt(ncol(train) - 1)

rf <- randomForest(hand ~., data = train, mtry = mtry)
```

```
set.seed(123)

test_samp <- test[sample(1:nrow(test), 100000), ]

rf_preds <- predict(rf, test_samp)

sum(rf_preds == test_samp$hand)/length(rf_preds)
```

```
## [1] 0.99643
```

```
table(rf_preds, test_samp$hand)
```

```
##
## rf_preds     0     1     2     3     4     5     6     7     8     9
##        0 50178     0     0     0    64   133     0     0     0     0
##        1     0 42264     0     0     0     0     0     0     0     0
##        2     0     0  4761     0     0     0   130     0     0     0
##        3     0     0     0  2081     0     0     7    22     0     0
##        4     0     0     0     0   279     0     0     0     0     0
##        5     0     0     0     0     0    73     0     0     1     0
##        6     0     0     0     0     0     0     6     0     0     0
##        7     0     0     0     0     0     0     0     1     0     0
##        8     0     0     0     0     0     0     0     0     0     0
##        9     0     0     0     0     0     0     0     0     0     0
```

Weighted Random Forest

```
rf_weighted <- randomForest(hand ~., data = train,
                            mtry = mtry, classwt = table(train$hand))

set.seed(112233)

test_samp <- test[sample(1:nrow(test), 100000), ]

rfw_preds <- predict(rf_weighted, test_samp)

sum(rfw_preds == test_samp$hand)/length(rfw_preds)
```

```
## [1] 0.99644
```

```
table(rfw_preds, test_samp$hand)
```

```
##
## rfw_preds     0     1     2     3     4     5     6     7     8     9
##         0 50330     0     0     0    55   143     0     0     0     0
##         1     0 42024     0     0     0     0     0     0     0     0
##         2     0     0  4779     0     0     0   129     0     0     0
##         3     0     0     0  2123     0     0     8    21     0     0
##         4     0     0     0     0   308     0     0     0     0     0
##         5     0     0     0     0     0    73     0     0     0     0
##         6     0     0     0     0     0     0     7     0     0     0
##         7     0     0     0     0     0     0     0     0     0     0
##         8     0     0     0     0     0     0     0     0     0     0
##         9     0     0     0     0     0     0     0     0     0     0
```

Adding weights doesn't improve the performance of the random forest.

Inverse Weighted Random Forest

```
rf_inweighted <- randomForest(hand ~., data = train,
                            mtry = mtry, classwt = 1/table(train$hand))

set.seed(11)

test_samp <- test[sample(1:nrow(test), 100000), ]

rfiw_preds <- predict(rf_inweighted, test_samp)

sum(rfiw_preds == test_samp$hand)/length(rfiw_preds)
```

```
## [1] 0.94443
```

```
table(rfiw_preds, test_samp$hand)
```

```
##
## rfiw_preds     0     1     2     3     4     5     6     7     8     9
##          0 50199    26     0     0    86   143     0     0     2     0
##          1     0 42273  3882  1251     0     0    26     3     0     0
##          2     0     0   855     5     0     0    83     0     0     0
##          3     0     0     0   767     0     0    23    25     0     0
##          4     0     0     0     0   262     0     0     0     0     0
```

```
##          5     0     0     0     0     0    79     0     0     0     0
##          6     0     0     0     0     0     0     7     0     0     0
##          7     0     0     0     0     0     0     0     1     0     0
##          8     0     0     0     0     0     0     0     0     0     0
##          9     0     0     0     0     2     0     0     0     0     0
```

Inverse weighting has caused the quality of the results to slightly decline.

SVM

```
library(e1071)

svm_mod <- svm(hand ~., data = train, kernel = "linear")
```

```
set.seed(321)

svm_test <- test[sample(1:nrow(test), 100000), ]

svm_preds <- predict(svm_mod, svm_test)

sum(svm_preds == svm_test$hand)/length(svm_preds)
```

```
## [1] 0.50641
```

```
table(svm_preds, svm_test$hand)
```

```
##
## svm_preds     0     1     2     3     4     5     6     7     8     9
##         0 50306 41850  4573  1836   312   193   119     9     0     0
##         1     0   334   162   234    31     0    37     2     0     0
##         2     0     0     0     0     0     0     0     0     0     0
##         3     0     0     0     0     0     0     0     0     0     0
##         4     0     0     0     0     0     0     0     0     0     0
##         5     0     0     0     0     0     0     0     0     0     0
##         6     0     0     0     0     0     0     0     0     0     0
##         7     0     0     0     0     0     0     0     0     0     0
##         8     0     0     0     0     0     0     0     0     0     0
##         9     0     1     0     0     0     0     0     0     0     1
```

SM Radial Kernel

```
svm_rad_mod <- svm(hand ~., data = train, kernel = "radial")

set.seed(4321)

svm_test <- test[sample(1:nrow(test), 100000), ]

svm_rad_preds <- predict(svm_rad_mod, svm_test)

sum(svm_rad_preds == svm_test$hand)/length(svm_rad_preds)
```

```
## [1] 0.99516
```

```
table(svm_rad_preds, svm_test$hand)
```

```
##
```

```
## svm_rad_preds     0     1     2     3     4     5     6     7     8     9
##             0 50209     0     0     0   274     1     0     0     0     0
##             1     0 42142     0     0     0    39     0     0     0     0
##             2     0     0  4811     0     0     0     0     0     0     0
##             3     0     0     0  2086     0     0   142    27     0     0
##             4     0     0     0     0   105     0     0     0     0     0
##             5     0     0     0     0     0   163     0     0     0     1
##             6     0     0     0     0     0     0     0     0     0     0
##             7     0     0     0     0     0     0     0     0     0     0
##             8     0     0     0     0     0     0     0     0     0     0
##             9     0     0     0     0     0     0     0     0     0     0
```

Adding the radial kernel greatly improves the performance of the

SVM Weighted Radial Kernel

```
svmwr_mod <- svm(hand ~., data = train,
                 kernel = "radial", class.weights = table(train$hand))

set.seed(3321)

svm_test <- test[sample(1:nrow(test), 100000), ]

svmwr_preds <- predict(svmwr_mod, svm_test)

sum(svmwr_preds == svm_test$hand)/length(svmwr_preds)
```

```
## [1] 0.99848
```

```
table(svmwr_preds, svm_test$hand)
```

```
##
## svmwr_preds     0     1     2     3     4     5     6     7     8     9
##           0 50461     0     0     0     0     2     0     0     0     0
##           1     0 42043     0     0     0     0     0     0     0     0
##           2     0     0  4633     0     0     0     0     0     0     0
##           3     0     0     0  2117     0     0   110    18     0     0
##           4     0     0     0     0   349     0     0     0     0     0
##           5     0     0     0     0     0   204     0     0     0     0
##           6     0     0     0    13     0     0    32     3     0     0
##           7     0     0     0     0     0     0     0     8     0     0
##           8     0     0     0     0     1     3     0     0     0     0
##           9     0     0     0     0     0     2     0     0     0     1
```

SVM Inverse Weighted Radial Kernel

```
svmiwr_mod <- svm(hand ~., data = train,
                 kernel = "radial", class.weights = 1/table(train$hand))

set.seed(33)

svm_test <- test[sample(1:nrow(test), 100000), ]

svmiwr_preds <- predict(svmiwr_mod, svm_test)
```

```
sum(svmiwr_preds == svm_test$hand)/length(svmiwr_preds)
```

```
## [1] 1e-05
```

```
table(svmiwr_preds, svm_test$hand)
```

```
##
## svmiwr_preds     0     1     2     3     4     5     6     7     8     9
##            0     0     0     0     0     0     0     0     0     0     0
##            1     0     0     0     0     0     0     0     0     0     0
##            2     0     0     0     0     0     0     0     0     0     0
##            3     0     0     0     0     0     0     0     0     0     0
##            4     0     0     0     0     0     0     0     0     0     0
##            5     0     0     0     0     0     0     0     0     0     0
##            6     0     0     0     0     0     0     0     0     0     0
##            7     0     0     0     0     0     0     0     0     0     0
##            8 50253 42277  4679  2078   364   200   124    23     1     1
##            9     0     0     0     0     0     0     0     0     0     0
```

Inverse weighting does not look to work well in this case where there are drastically different sizes of classes.