Predicting Exercise Form

Rohil

March 31, 2019

In the data set there are several variables that are blank or have NA values for most observations. We remove these are well as the variables associated with the ID of the subject and the time it was done. This cuts the number of variables down from 160 variables to 53.

```
ex_tr<-read.csv("pml-training.csv")
temp_transp<-t(ex_tr[1,])
ind<-1:160
#The below index gives the column index of variables that are not NA or empty for mos
t observations
i<-ind[(!is.na(temp_transp))&(temp_transp!="")]
tr_set<-ex_tr[,i]
tr_set<-tr_set[,-(1:7)]
```

We will use three machine learning methods to attempt to classify the type of exercise being done based on the observation. The three methods we will use are Linear discriminant analysis, Adaboost and Random forest.

```
rf3<-train(classe~.,data=tr_set,method="rf",trControl=trainControl(method="cv",number=
3))
boost3<-train(classe~.,data=tr_set,method="AdaBoost.M1",trControl=trainControl(method
="cv",number=3))
lda3<-train(classe~.,data=tr_set,method="lda",trControl=trainControl(method="cv",numbe
r=3))
```

For each method we did three-fold cross validation (as this was the best compromise of optimizing hyperparameters and keeping computation time low). Cross validation is used to find the optimal hyperparameters for each method (for boosting and random forest).

We see the optimal hyperparameter for the random forest method is to randomly select 27 variables at each node.

```
rf3$results
```

```
##   mtry  Accuracy     Kappa  AccuracySD     KappaSD
## 1    2 0.9920500 0.9899425 0.001374218 0.001739480
## 2   27 0.9923558 0.9903301 0.001858473 0.002351570
## 3   52 0.9858329 0.9820792 0.004212702 0.005328949
```

We also see the optimal parameters such as the the numbers of trees and maximum depth of the trees for adaboost (150 trees, with a maximum depth of 3)

```
boost3$results
```

```
##    coeflearn maxdepth mfinal  Accuracy     Kappa   AccuracySD     KappaSD
## 1    Breiman        1     50 0.3662217 0.1249731 0.0011902279 0.001176496
## 10    Freund        1     50 0.3662217 0.1249731 0.0011902279 0.001176496
## 19       Zhu        1     50 0.6443293 0.5477689 0.0147207267 0.017259552
## 4    Breiman        2     50 0.3680562 0.1272093 0.0006895068 0.001082784
## 13    Freund        2     50 0.3681071 0.1272858 0.0007772771 0.001214414
## 22       Zhu        2     50 0.7733172 0.7139631 0.0129765949 0.016420409
## 7    Breiman        3     50 0.4176418 0.2295708 0.0212468510 0.009652229
## 16    Freund        3     50 0.4169285 0.2282174 0.0204770896 0.009706349
## 25       Zhu        3     50 0.8590848 0.8224119 0.0167865419 0.020839129
## 2    Breiman        1    100 0.3662217 0.1249731 0.0011902279 0.001176496
## 11    Freund        1    100 0.3662217 0.1249731 0.0011902279 0.001176496
## 20       Zhu        1    100 0.7156780 0.6399251 0.0168561818 0.020644754
## 5    Breiman        2    100 0.3680562 0.1272093 0.0006895068 0.001082784
## 14    Freund        2    100 0.3680562 0.1272093 0.0006895068 0.001082784
## 23       Zhu        2    100 0.8312614 0.7873431 0.0207303536 0.025830599
## 8    Breiman        3    100 0.4176928 0.2296257 0.0212754179 0.009524608
## 17    Freund        3    100 0.4189160 0.2307361 0.0189411964 0.011911081
## 26       Zhu        3    100 0.9088253 0.8849556 0.0134443089 0.016868688
## 3    Breiman        1    150 0.3662217 0.1249731 0.0011902279 0.001176496
## 12    Freund        1    150 0.3662217 0.1249731 0.0011902279 0.001176496
## 21       Zhu        1    150 0.7467128 0.6796331 0.0059402803 0.008077986
## 6    Breiman        2    150 0.3680562 0.1272093 0.0006895068 0.001082784
## 15    Freund        2    150 0.3680562 0.1272093 0.0006895068 0.001082784
## 24       Zhu        2    150 0.8544504 0.8163967 0.0123185317 0.015488386
## 9    Breiman        3    150 0.4177947 0.2296912 0.0213916471 0.009546015
## 18    Freund        3    150 0.4175909 0.2292723 0.0211887865 0.009751594
## 27       Zhu        3    150 0.9270703 0.9079230 0.0141380497 0.017778824
```

We can also compare the cross validated accuracy across all three methods (for the optimal parameters)

```
rf3$resample
```

```
##    Accuracy     Kappa Resample
## 1 0.9944946 0.9930363    Fold1
## 2 0.9914386 0.9891696    Fold3
## 3 0.9911342 0.9887843    Fold2
```

This is the cross validated accuracy for random forest

```
boost3$resample
```

```
##    Accuracy     Kappa Resample
## 1 0.9403852 0.9246887    Fold2
## 2 0.9122324 0.8892800    Fold3
## 3 0.9285933 0.9098003    Fold1
```

This is the cross validated accuracy for Adaboost

```
lda3$resample
```

```
##    Accuracy     Kappa Resample
## 1 0.7057025 0.6275150    Fold1
## 2 0.7030581 0.6242579    Fold2
## 3 0.6971411 0.6166337    Fold3
```

This is the cross validated accuracy for linear discriminant analysis.

From the above tables we see that random forest has a significantly better cross validated accuracy than the other methods(lda is far worse than the other methods). Thus we will use the random Forest model we built to classify the test data.

Not only does the cross validation allows to find the optimal hyperparameters, but it also gives an estimate of the out of sample error (as cross validation leaves out part of the training set for each fold, so we can use it as test data to get a better accuracy). In this case is the average accuracy over the three cross validated samples which is 0.99235 or an out of sample error rate of about 0.76%. For random forest there is another way to estimate the out of sample error and that is the out of bag error rate (thus if you are not optimizing hyperparameters you do not need to do cross validation). The out of bag error rate basically runs the samples that were not included in the bootstrap samples to create each tree in the forest, down that tree, and we can calculate an average error rate. The result of the out of bag error estimate are seen below:

```
rf3$finalModel$err.rate[500,]
```

```
##          OOB            A            B            C            D 
## 0.0041280196 0.0008960573 0.0063207796 0.0055523086 0.0077736318 
##            E 
## 0.0022179096
```

```
rf3$finalModel$confusion
```

```
##      A    B    C    D    E  class.error
## A 5575    3    1    0    1 0.0008960573
## B   20 3773    4    0    0 0.0063207796
## C    0    8 3403   11    0 0.0055523086
## D    0    0   22 3191    3 0.0077736318
## E    0    1    2    5 3599 0.0022179096
```

Thus we see the OOB error rate is about 0.41%, this is slightly lower than the cross validated error. The OOB should be considered more accurate as we only use 3-fold cross validation, and the OOB is calculated from 500 bootstrapped samples. Thus the difference is due the bias in the CV error, due to the low number of folds. This is verified as when we do 10 fold CV the CV error is about 0.42%.

We can also see which variables were most important to correctly classify how the exercise is being done, by looking at the importance of each variable:

```
rf3$finalModel$importance[1:30,]
```

```
##            roll_belt           pitch_belt             yaw_belt
##           2076.80627            944.68045           1149.20393
##      total_accel_belt         gyros_belt_x         gyros_belt_y
##             62.61478             58.46328             62.59133
##         gyros_belt_z         accel_belt_x         accel_belt_y
##            244.67325             44.78724             48.27887
##         accel_belt_z        magnet_belt_x        magnet_belt_y
##            277.07431            225.41107            278.21848
##        magnet_belt_z             roll_arm            pitch_arm
##            321.09675            208.69417            121.77115
##              yaw_arm      total_accel_arm          gyros_arm_x
##            257.06856             65.70906             83.90808
##          gyros_arm_y          gyros_arm_z          accel_arm_x
##            107.49275             28.84334            152.39009
##          accel_arm_y          accel_arm_z         magnet_arm_x
##             92.44231             73.52997            152.34884
##         magnet_arm_y         magnet_arm_z        roll_dumbbell
##            144.88355            127.99558            364.19453
##       pitch_dumbbell         yaw_dumbbell total_accel_dumbbell
##             95.13789            184.10581            311.80137
```

The values in the above table represent the decrease in correct classification when that variable's effect of the data is removed (the larger the value the more incorrect classifications). Thus we see that the motions associated with the belt are very important in correctly classifying what exercise is being done.

Thus to predict whether the exercise was being done properly, we used the random Forest algorithm. We chose this algorithm as compared to LDA and Adaboost it provided the lowest CV error. We expect the out of sample error to be close to the OOB error at 0.41%.