Machine Learning - Human Activity Recognition

Joost van Brakel

November 20, 2015

Executive summary

Human Activity Recognition - HAR - has emerged as a key research area in the last years and is gaining increasing attention by the pervasive computing research community (see picture below, that illustrates the increasing number of publications in HAR with wearable accelerometers), especially for the development of context-aware systems. There are many potential applications for HAR, like: elderly monitoring, life log systems for monitoring energy expenditure and for supporting weight-loss programs, and digital assistants for weight lifting exercises.

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har). If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz3s7IvMhWN (http://groupware.les.inf.puc-rio.br/har#ixzz3s7IvMhWN)

We utilized the knowledge generated from the paper above and focused on evluating four methods and random forest method provided most accurate report, which accurancy in the 99%

Process

For this analysis we used R and a number of key libraries, namely : xtable, AppliedPredictiveModeling, caret, randomForest, rpart, knitr,htmlTable and doMC

Data

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

Participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throw- ing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E).

Processing

The columns were reviewed to determine if there was usuable data present, if less than 90% of the rows has useable data the column was removed, if no valide model can be build we can revisit this approach. For this we used a function called 'getFractionMissing' which is available within the forums. we also reviewed the test set and see which column had data in it, this allowed us to focus on the relevant columns and reduce computing time.

The training dataset was split in to 70% training, 30% validation

```
URL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
download.file(URL, destfile = "pml-training.csv", method="curl")

URL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(URL, destfile = "pml-testing.csv", method="curl")

training <- read.csv("pml-training.csv",na.strings = "NA")
testing <- read.csv("pml-testing.csv",na.strings = "NA")

training[,7:159] <- sapply(training[,7:159],as.numeric)
testing[,7:159] <- sapply(testing[,7:159], as.numeric)

# remove columns that will not be used in prediction
training <- training[8:160]
testing <- testing[8:160]

# remove na columns in testing set from both testing and training.
nas <- is.na(apply(testing,2,sum))
testing <- testing[,!nas]
training <- training[,!nas]


colDesc <- getFractionMissing(training)
filteredCol <- colDesc[colDesc$FractionMissing < .9,1]
training2 <- training[,filteredCol]
testing2 <- testing[,filteredCol]

set.seed(1313)
# divide training set in training and validation
inTrain <- createDataPartition(y=training2$classe, p=0.7, list=FALSE)
trainSet <- training2[inTrain,]
valSet <- training2[-inTrain,]
```

Modeling performance, selection and

validation

Based on the previous analysis performed on this dataset, which was documented with the attached link, this dataset best responds to tree based predictions.

To determine the best modeling approach we examined 4 approaches
1. Recursive Partitioning and Regression Trees results (rpart)
2. Random Forest (rf)
3. Generalized Boosted Models (gbm)
4. Treebag (treebag)

We used A 3-fold cross-validation to estimate the out of sample error, we tried a number of folds, we provided the comparison below between 2,3 and 4 folD, no material difference existed therefore we selected the midde option (3 fold)

```
# model - Recursive Partitioning and Regression Trees
modRPATFit <- train(classe ~.,data=trainSet, method="rpart")
pred0 <- predict(modRPATFit, newdata = valSet)
cm0 <- confusionMatrix(pred0, valSet$classe)

#model -  Random Forest
modelRFFit <- train(classe ~., data=trainSet, method="rf",na.action = na.omit, allowP
arallel=TRUE, trControl = trainControl(method = "cv", number = 3, allowParallel = TRU
E))
pred1 <- predict(modelRFFit, newdata = valSet)
cm1 <- confusionMatrix(pred1, valSet$classe)

#model -  Generalized Boosted Models
modelGBMFit <- train(classe ~., data=trainSet, method="gbm",na.action = na.omit, trCo
ntrol = trainControl(method = "cv", number = 3),verbose = FALSE)
pred2 <- predict(modelGBMFit, newdata = valSet)
cm2 <- confusionMatrix(pred2, valSet$classe)

#model -  treebag
modelBAGFit <- train(classe ~., data=trainSet, method="treebag",na.action = na.omit,
trControl = trainControl(method = "cv", number = 3),verbose = FALSE)
pred3 <- predict(modelBAGFit, newdata = valSet)
cm3 <- confusionMatrix(pred3, valSet$classe)

results_test <-rbind(
      list("rpart", cm0$overall[1]),
      list("rf", cm1$overall[1]),
      list("gbm", cm2$overall[1]),
      list ("treebag",cm3$overall[1])
)

# added readable column headers
colnames(results_test) <- c("Method", "Accuracy")

# export results to table
tab_Q1 <-xtable(results_test, caption = "Results by method")
print(tab_Q1, floating=TRUE, caption.placement="top", type="html")
```

Results by method

	Method	Accuracy
1	rpart	0.50
2	rf	0.99
3	gbm	0.96
4	treebag	0.91

```
#model -  Random Forest - 2-fold Cross validatio
modelRFFit1 <- train(classe ~., data=trainSet, method="rf",na.action = na.omit, allow
Parallel=TRUE, trControl = trainControl(method = "cv", number = 2, allowParallel = TR
UE))
pred1a <- predict(modelRFFit1, newdata = valSet)
cm1a <- confusionMatrix(pred1a, valSet$classe)

#model -  Random Forest - 4-fold Cross validatio
modelRFFit2 <- train(classe ~., data=trainSet, method="rf",na.action = na.omit, allow
Parallel=TRUE, trControl = trainControl(method = "cv", number = 4, allowParallel = TR
UE))
pred1b <- predict(modelRFFit2, newdata = valSet)
cm1b <- confusionMatrix(pred1b, valSet$classe)

results_test <-rbind(
      list("2-fold", cm1a$overall[1]),
      list("3-fold", cm1$overall[1]),
      list("4-fold",cm1b$overall[1])
)

# added readable column headers
colnames(results_test) <- c("Method", "Accuracy")

# export results to table
tab_Q2 <-xtable(results_test, caption = "Random forest results by fold")
print(tab_Q2, floating=TRUE, caption.placement="top", type="html")
```

Random forest results by fold

	Method	Accuracy
1	2-fold	0.99
2	3-fold	0.99
3	4-fold	0.99

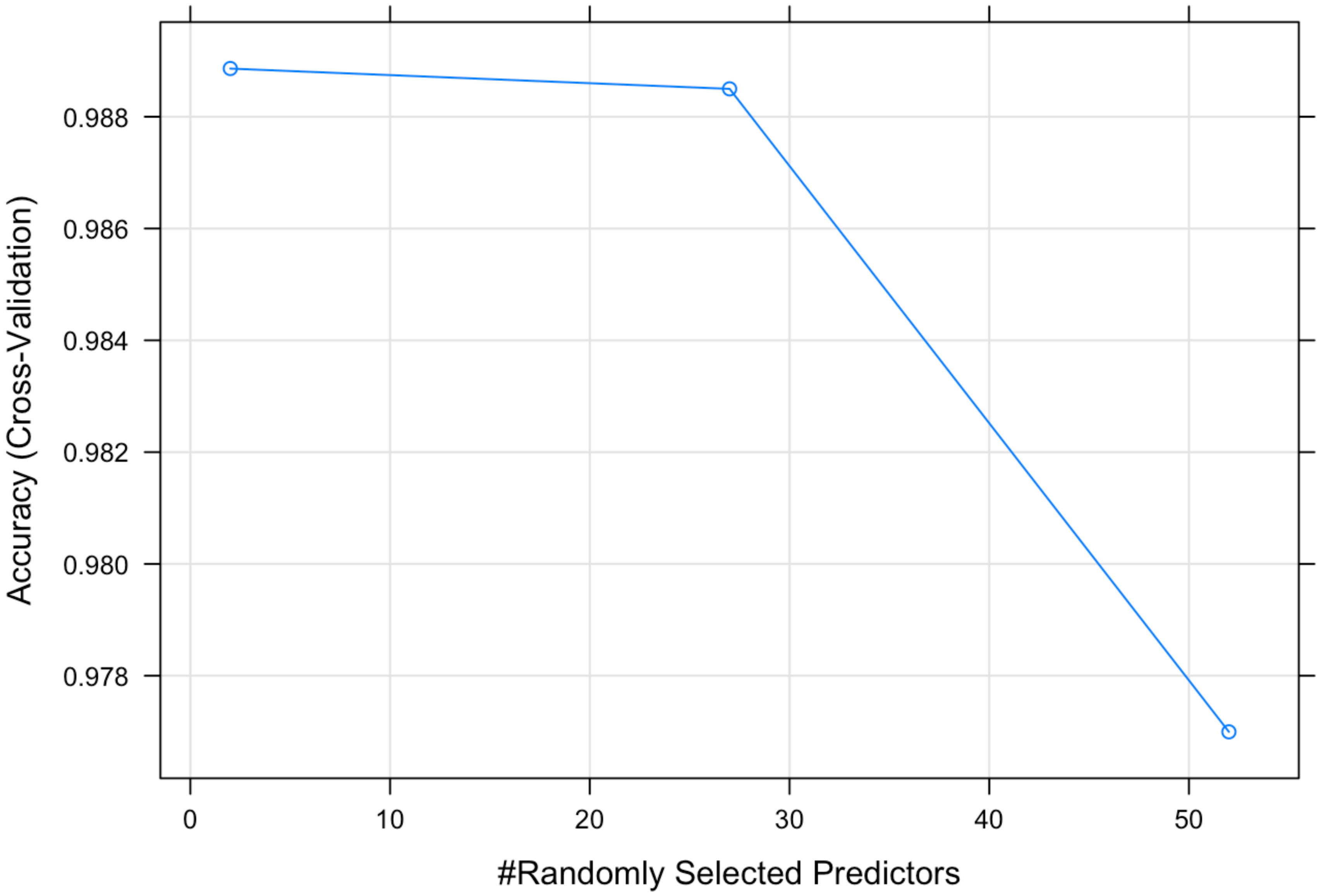
Predictions

The RF methods provides the strongest test on an accuracy basis, if we look at the figures below, you can see in the confusion Matrix, that the larges out of sample error is within D, which means that D will be the most challenging to forecast

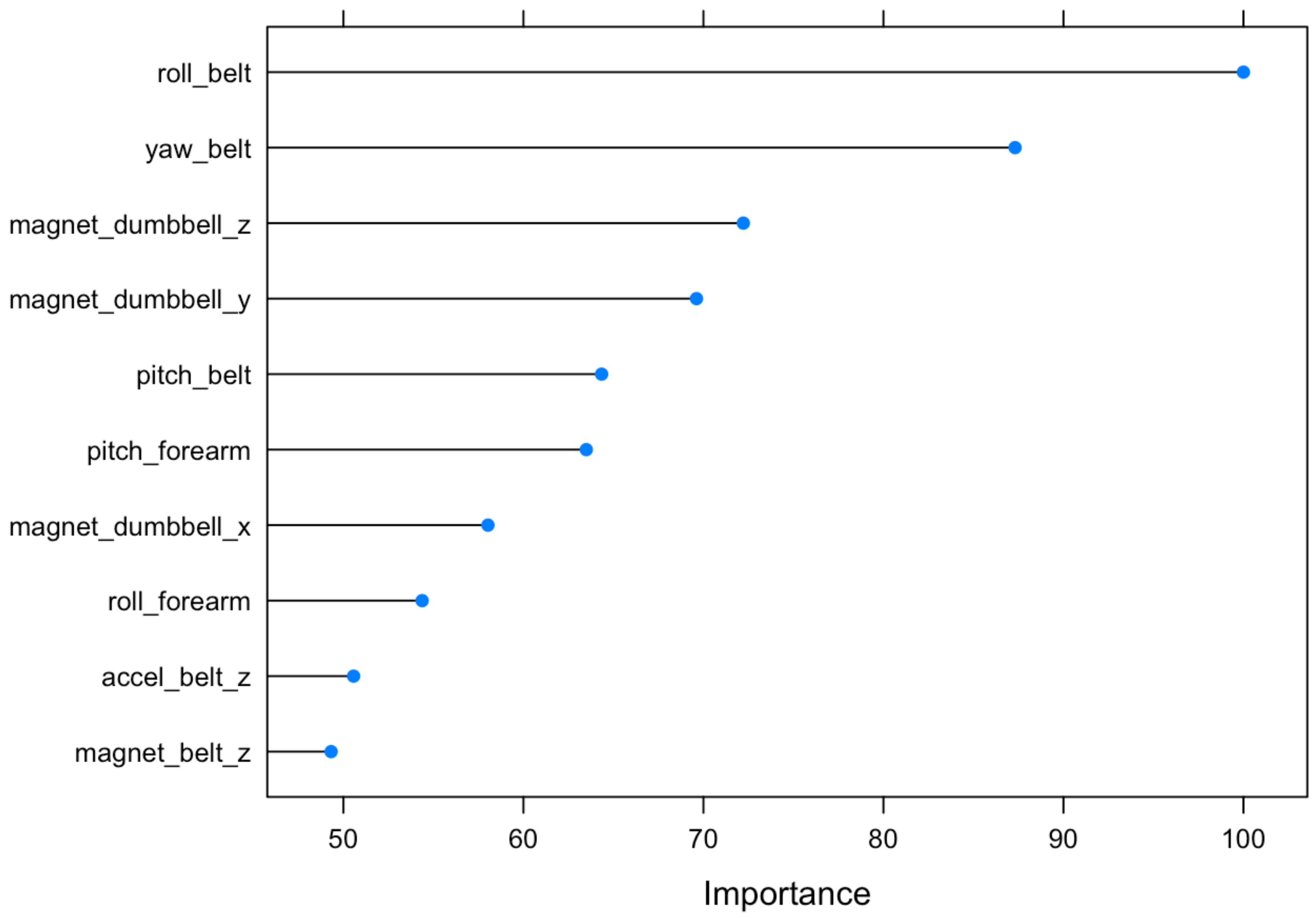
```
varImp(modelRFFit)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                    Overall
## roll_belt           100.00
## yaw_belt             87.31
## magnet_dumbbell_z    72.22
## magnet_dumbbell_y    69.61
## pitch_belt           64.34
## pitch_forearm        63.49
## magnet_dumbbell_x    58.03
## roll_forearm         54.37
## accel_belt_z         50.57
## magnet_belt_z        49.32
## accel_dumbbell_y     47.88
## roll_dumbbell        43.44
## magnet_belt_y        43.32
## accel_dumbbell_z     42.61
## roll_arm             34.75
## accel_forearm_x      34.22
## gyros_belt_z         34.14
## yaw_dumbbell         31.20
## accel_dumbbell_x     30.36
## magnet_arm_x         30.32
```

```
plot(modelRFFit)
```



```
plot(varImp(modelRFFit), top = 10)
```



```
tab_Q3 <-xtable(cm1$table, caption = "final results - confusion matrix")
print(tab_Q3, floating=TRUE, caption.placement="top", type="html")
```

final results - confusion matrix

	A	**B**	**C**	**D**	**E**
A	1674	4	0	0	0
B	0	1133	5	0	0
C	0	2	1020	17	0
D	0	0	1	946	1
E	0	0	0	1	1081

```
cm1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    4    0    0    0
##          B    0 1133    5    0    0
##          C    0    2 1020   17    0
##          D    0    0    1  946    1
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.9947
##                  95% CI : (0.9925, 0.9964)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9933
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9942   0.9813   0.9991
## Specificity            0.9991   0.9989   0.9961   0.9996   0.9998
## Pos Pred Value         0.9976   0.9956   0.9817   0.9979   0.9991
## Neg Pred Value         1.0000   0.9987   0.9988   0.9964   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1925   0.1733   0.1607   0.1837
## Detection Prevalence   0.2851   0.1934   0.1766   0.1611   0.1839
## Balanced Accuracy      0.9995   0.9968   0.9951   0.9905   0.9994
```

Forecasting

Utilizing the result set to forecast based on the ‘testing’ dataset. This resulted in a 100% accurate result.