exercise81

Lev Mazaev

Exercise 8.1 (Applied Predictive Modeling, p. 218)

Loading the packages

```
library(mlbench)
library(caret)
library(doMC)
registerDoMC(8)
library(randomForest)
library(partykit)
library(Cubist)
```

Creating the data

```
set.seed(200)
simulated <- mlbench.friedman1(200, sd = 1)
simulated <- cbind(simulated$x, simulated$y)
simulated <- as.data.frame(simulated)
colnames(simulated)[ncol(simulated)] <- 'y'
```

A: Fit a random forest model to all of the predictors, then estimate the variable importance scores.

```
rf1 <- randomForest(y ~ ., data = simulated, importance = TRUE, ntree = 1000)
rfImp1 <- varImp(rf1, scale = FALSE)
rfImp1
```

```
##          Overall
## V1   8.732235404
## V2   6.415369387
## V3   0.763591825
## V4   7.615118809
## V5   2.023524577
## V6   0.165111172
## V7  -0.005961659
## V8  -0.166362581
## V9  -0.095292651
## V10 -0.074944788
```

The random forest model did not significantly use the uninformative predictors (V6-V10).

B: Now add an additional predictor that is highly correlated with one of the informative predictors.

```r
simulated$duplicate1 <- simulated$V1 + rnorm(200) * .1
cor(simulated$duplicate1, simulated$V1)
```

```
## [1] 0.9460206
```

Now fitting another random forest model:

```r
rf2 <- randomForest(y ~ ., data = simulated, importance = TRUE, ntree = 1000)
rfImp2 <- varImp(rf2, scale = FALSE)
rfImp2
```

```
##                 Overall
## V1           5.69119973
## V2           6.06896061
## V3           0.62970218
## V4           7.04752238
## V5           1.87238438
## V6           0.13569065
## V7          -0.01345645
## V8          -0.04370565
## V9           0.00840438
## V10          0.02894814
## duplicate1   4.28331581
```

The importance score for V1 has changed. Now duplicate1 also has a significant score.

Adding another highly correlated predictor:

```r
simulated$duplicate2 <- simulated$V1 + rnorm(200, 5) * .1
cor(simulated$duplicate2, simulated$V1)
```

```
## [1] 0.9408631
```

```r
rf3 <- randomForest(y ~ ., data = simulated, importance = TRUE, ntree = 1000)
rfImp3 <- varImp(rf3, scale = FALSE)
rfImp3
```

```
##                 Overall
## V1           4.91687329
## V2           6.52816504
## V3           0.58711552
## V4           7.04870917
## V5           2.03115561
## V6           0.14213148
## V7           0.10991985
## V8          -0.08405687
## V9          -0.01075028
## V10          0.09230576
## duplicate1   3.80068234
## duplicate2   1.87721959
```

The importance score for V1 decreases further while both duplicate1 and duplicate2 predictors have significant importance score.

C: Use the cforest function in the party package to fit a random forest model using conditional inference trees.

```
modelC <- cforest(y ~ ., data = simulated[1:11], ntree = 1000)
CImpc <- varimp(object = modelC, conditional = TRUE)
CImpuc <- varimp(object = modelC, conditional = FALSE)
as.data.frame(CImpc)
```

```
##           CImpc
## V1   6.09202646
## V2   5.03981436
## V3   0.05447701
## V4   5.83953896
## V5   1.33653409
## V6  -0.13229331
## V7  -0.24300091
## V8  -0.37798087
## V9  -0.33878692
## V10 -0.21076946
```

```
as.data.frame(CImpuc)
```

```
##          CImpuc
## V1   7.97905534
## V2   6.06917117
## V3   0.17207240
## V4   7.04020522
## V5   2.12514170
## V6   0.02341251
## V7  -0.04618777
## V8  -0.20974399
## V9  -0.06769006
## V10 -0.08035645
```

The importance score of cforest model in case of absence of additional predictors (duplicate1 and duplicate2) shows other pattern: V3 also has low importance (on one level with uninformative, which wasn't the case previously). Also conditional varimp function estimates uninformative predictors' importance (V6-V10) higher than unconditional, but still much lower than informative ones. Now let's try with duplicates:

```
modelCd <- cforest(y ~ ., data = simulated, ntree = 1000)
CdImpc <- varimp(object = modelCd, conditional = TRUE)
CdImpuc <- varimp(object = modelCd, conditional = FALSE)
as.data.frame(CdImpc)
```

```
##                  CdImpc
## V1         2.6155145760
## V2         5.0508731751
## V3        -0.0002894988
## V4         5.5274159198
## V5         1.3419562624
## V6          -0.1135858166
## V7          -0.1198127197
## V8          -0.3421302534
## V9          -0.0642303814
## V10         -0.1365404819
## duplicate1   2.1594566357
## duplicate2   0.7762612679
```

```r
as.data.frame(CdImpuc)
```

```
##                 CdImpuc
## V1           5.71857126
## V2           5.69713667
## V3           0.12664552
## V4           6.35262907
## V5           1.78168014
## V6           0.10986986
## V7           0.12537108
## V8          -0.21606502
## V9           0.10508967
## V10         -0.09061111
## duplicate1   5.37562008
## duplicate2   2.98872128
```

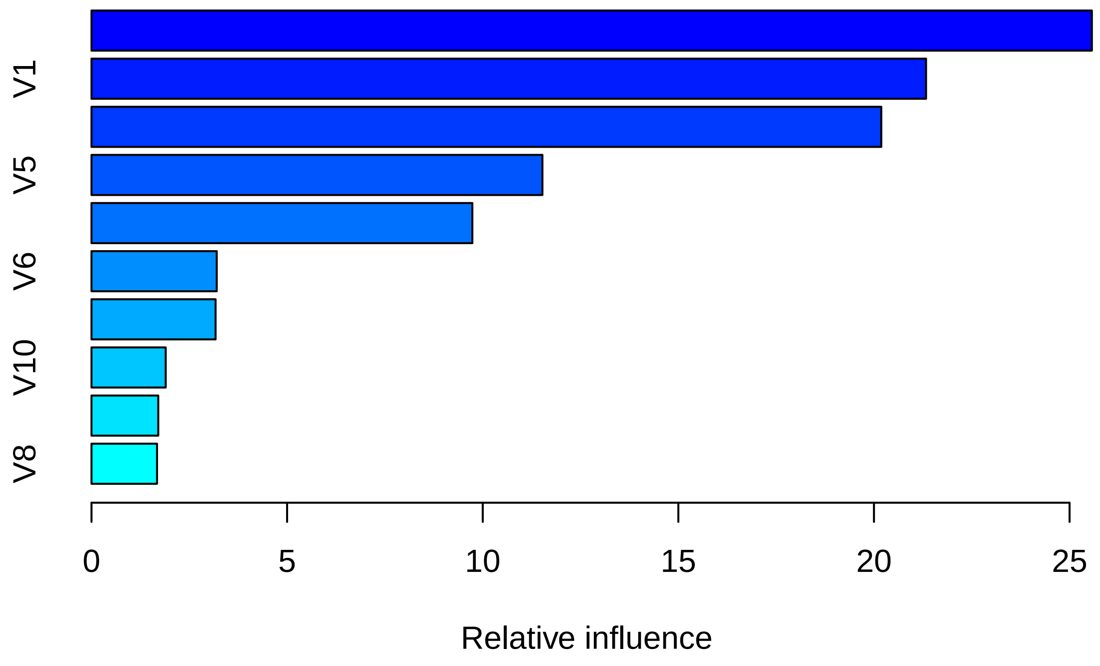
V3 has very low importance in both cases, uninformative predictors also do. But conditional varimp function correctly recognizes (if it could be said so) highly correlated predictors (V1, duplicate1 and duplicate2) and lowers their importance score, while unconditional varimp does not.

D: Repeat the process with different tree model, such as boosted trees and cubist.

```r
library(gbm)
```

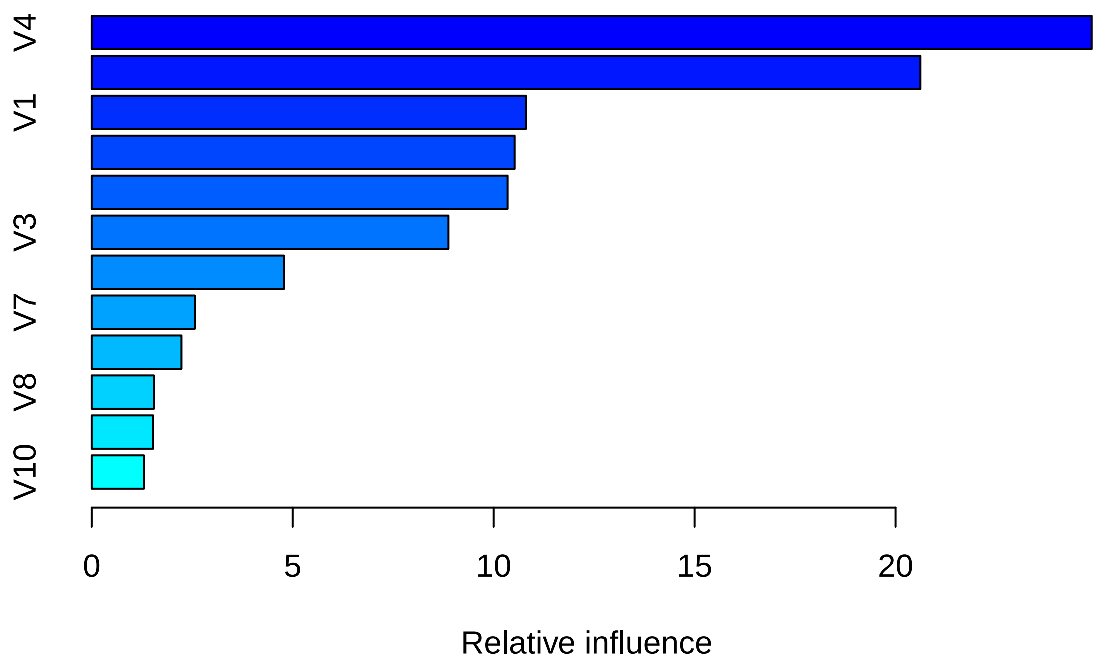
```
## Loaded gbm 2.1.4
```

```r
gbm1 <- gbm(y ~ ., data = simulated[1:11], n.trees = 1000, distribution = 'gaussian')
summary(gbm1)
```



```
##      var   rel.inf
## V4   V4 25.569778
## V1   V1 21.330829
## V2   V2 20.186527
## V5   V5 11.525698
## V3   V3  9.735027
## V6   V6  3.201344
## V7   V7  3.171857
## V10 V10  1.897044
## V9   V9  1.707005
## V8   V8  1.674890
```

```r
gbm2 <- gbm(y ~ ., data = simulated, n.trees = 1000, distribution = 'gaussian')
summary(gbm2)
```



```
##                   var   rel.inf
## V4                 V4 24.878586
## V2                 V2 20.617640
## V1                 V1 10.801053
## duplicate1 duplicate1 10.522368
## V5                 V5 10.346836
## V3                 V3  8.875361
## duplicate2 duplicate2  4.784226
## V7                 V7  2.564380
## V6                 V6  2.233020
## V8                 V8  1.547613
## V9                 V9  1.529735
## V10               V10  1.299182
```

In case of boosted trees, uninformative predictors are insignificant. Duplicates are taken into account, but on lower level than V1.

```
cubist1 <- train(y ~ ., data = simulated[1:11], method = 'cubist')
varImp(cubist1)
```

```
## cubist variable importance
##
##     Overall
## V1   100.00
## V2    75.69
## V4    68.06
## V3    58.33
## V5    55.56
## V6    15.28
## V8     0.00
## V9     0.00
## V7     0.00
## V10    0.00
```

```r
cubist2 <- train(y ~ ., data = simulated, method = 'cubist')
varImp(cubist2)
```

```
## cubist variable importance
##
##            Overall
## V2         100.000
## V1          77.698
## V4          71.942
## V5          54.676
## V3          46.043
## duplicate1  35.971
## duplicate2  35.971
## V6          14.388
## V8           4.317
## V7           0.000
## V10          0.000
## V9           0.000
```

The Cubist neglects uninformative predictors and takes into consideration duplicates, but with smaller score than V1.

Summary.

All models take into account informative predictors (differently), but almost neglect uninformative. In case of added higly-correlated predictors each model reacts differently, but some pattern may be traced: 1. Duplicates drain the importance from the V1. 2. V1 still has higher importance score than duplicates in all models. 3. Duplicate1 is usually more important than duplicate2.