Homework 6

Aaron Politsky

November 8, 2015

```
set.seed(99)
library(h2o)
load("data.Rda")
source("~/HelpR/EvaluationMetrics.R")
```

```
source("ParseData.R")
data <- parse_human_activity_recog_data()
```

1. Neural Network Models

```
# start or connect to h2o server
h2oServer <- h2o.init(max_mem_size="4g", nthreads=-1)

# we need to load data into h2o format
train_hex = as.h2o(data.frame(x=data$X_train, y=data$y_train))
test_hex = as.h2o(data.frame(x=data$X_test, y=data$y_test))

predictors <- 1:(ncol(train_hex)-1)
response <- ncol(train_hex)
```

Let's see how different models perform when we try different parameters.

```
hyper.params <-
  list(
    epochs=c(2,5,10),
    hidden=list(c(64), c(128), c(256), c(512), c(1024),
                c(256,256), c(1024,1024), c(128,128,128))
  )
```

```
dl.grid <- h2o.grid(
  algorithm = "deeplearning",
  x=predictors, y=response,
  training_frame=train_hex,
  activation="Tanh",
  classification_stop=-1,  # Turn off early stopping
  l1=1e-5,
  hyper_params = hyper.params
)
summary(dl.grid)
dl.grid.models <- lapply(dl.grid@model_ids, function(id) h2o.getModel(id))
model.paths <- lapply(dl.grid.models, function(m) h2o.saveModel(m, path="models"))
```

```
# performance on test set
ptest.list <- lapply(dl.grid.models, function(m) h2o.performance(m, test_hex))
cm.test.list <- lapply(ptest.list, function(ptest) h2o.confusionMatrix(ptest))
```

Which performed the best?

```
library(plyr)
ptest.df <- ldply(cm.test.list,
                  function(cm)
                    c(tot.test.error.rate = cm$Error[7]))
ptest.df <- cbind(ptest.df, expand.grid((hyper.params)))

best.model.index <- which.min(ptest.df$tot.test.error.rate)
best.dl.model <- dl.grid.models[[best.model.index]]
cm.test.list[[best.model.index]]
```

```
## Confusion Matrix - (vertical: actual; across: predicted): vertical: actual; across: predicted
##                    Laying Sitting Standing Walking WalkingDownstairs
## Laying                513       0       24       0                 0
## Sitting                 0     425       63       0                 0
## Standing                0      10      521       1                 0
## Walking                 0       0        0     491                 4
## WalkingDownstairs       0       0        2       4               403
## WalkingUpstairs         0       0        1      25                16
## Totals                513     435      611     521               423
##                    WalkingUpstairs  Error          Rate
## Laying                           0 0.0447 =    24 / 537
## Sitting                          3 0.1344 =    66 / 491
## Standing                         0 0.0207 =    11 / 532
## Walking                          1 0.0101 =     5 / 496
## WalkingDownstairs               11 0.0405 =    17 / 420
## WalkingUpstairs                429 0.0892 =    42 / 471
## Totals                         444 0.0560 = 165 / 2,947
```

Model 2 did the best, with a **5.60%** test error rate.

How did our choice of epochs and number and levels of neurons parameters affect performance?

```
ptest.df[order(ptest.df$tot.test.error.rate),]
```

```
##    tot.test.error.rate epochs        hidden
## 2           0.05598914      5            64
## 22          0.05632847      2 128, 128, 128
## 4           0.06073974      2           128
## 8           0.06073974      5           256
## 18          0.06175772     10      256, 256
## 6           0.06311503     10           128
## 19          0.06345436      2    1024, 1024
## 17          0.06413302      5      256, 256
## 1           0.06515100      2            64
## 10          0.06582966      2           512
## 7           0.06650831      2           256
```

```
## 12          0.06684764     10           512
## 20          0.06718697      5    1024, 1024
## 11          0.06820495      5           512
## 15          0.06820495     10          1024
## 14          0.06922294      5          1024
## 16          0.06990159      2      256, 256
## 5           0.07024092      5           128
## 21          0.07024092     10    1024, 1024
## 24          0.07227689     10 128, 128, 128
## 3           0.07261622     10            64
## 9           0.07431286     10           256
## 13          0.07567017      2          1024
## 23          0.07940278      5 128, 128, 128
```

I can't make much of a pattern out of that. I guess we can just try different things and see how it performs.

2. Tree Models

Let's see how different models perform when we try different parameters.

Random Forests

Let's try a few random forest parameters:

```
rf.hyper.params <-
  list(
    ntrees=c(250,500,1000),
    min_rows = c(50,100)
  )
```

```
rf.grid <- h2o.grid(
  algorithm = "randomForest",
  x=predictors, y=response,
  training_frame=train_hex,
  hyper_params = rf.hyper.params
)
summary(rf.grid)
rf.grid.models <- lapply(rf.grid@model_ids, function(id) h2o.getModel(id))
rf.model.paths <- lapply(rf.grid.models, function(m) h2o.saveModel(m, path="models"))
```

Boosting

Let's try a few boost parameters.

```
boost.hyper.params <-
  list(
    ntrees = c(500,2000),
    learn_rate = c(.2, .01),
    max_depth=c(4,10)
  )
```

```
boost.grid <- h2o.grid(
  algorithm = "gbm",
  x=predictors, y=response,
  training_frame=train_hex,
  hyper_params = boost.hyper.params
)
summary(boost.grid)
boost.grid.models <- lapply(boost.grid@model_ids, function(id) h2o.getModel(id))
boost.model.paths <- lapply(boost.grid.models, function(m) h2o.saveModel(m, path="models"))
```

Let's assess performance:

```
# performance on test set
rf.ptest.list <- lapply(rf.grid.models, function(m) h2o.performance(m, test_hex))
rf.cm.test.list <- lapply(rf.ptest.list, function(ptest) h2o.confusionMatrix(ptest))

boost.ptest.list <- lapply(boost.grid.models, function(m) h2o.performance(m, test_hex))
boost.cm.test.list <- lapply(boost.ptest.list, function(ptest) h2o.confusionMatrix(ptest))
```

Which did the best?

```
rf.ptest.df <- ldply(rf.cm.test.list,
                  function(cm)
                    c(tot.test.error.rate = cm$Error[7]))
rf.ptest.df <- cbind(rf.ptest.df, expand.grid((rf.hyper.params)))


rf.best.model.index <- which.min(rf.ptest.df$tot.test.error.rate)
rf.best.model <- rf.grid.models[[rf.best.model.index]]


boost.ptest.df <- ldply(boost.cm.test.list,
                  function(cm)
                    c(tot.test.error.rate = cm$Error[7]))
boost.ptest.df <- cbind(boost.ptest.df, expand.grid((boost.hyper.params)))

boost.best.model.index <- which.min(boost.ptest.df$tot.test.error.rate)
boost.best.model <- boost.grid.models[[boost.best.model.index]]

rf.ptest.df[rf.best.model.index,]
```

```
##   tot.test.error.rate ntrees min_rows
## 3          0.08618935   1000       50
```

```
boost.ptest.df[boost.best.model.index,]
```

```
##   tot.test.error.rate ntrees learn_rate max_depth
## 2          0.06447234   2000        0.2         4
```

```
ptest.df[best.model.index,]
```

```
##   tot.test.error.rate epochs hidden
## 2          0.05598914      5     64
```