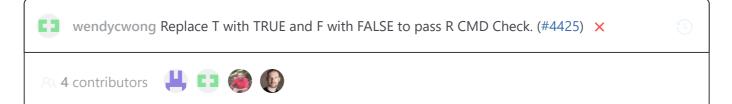


Code Pull requests 60 Actions Projects Security Insights

```
ੂੰ master ਦ
```

h2o-3 / h2o-r / tests / testdir_algos / deeplearning /

runit_deeplearning_stacked_autoencoder_large.R



```
86 lines (72 sloc) 3.39 KB
       setwd(normalizePath(dirname(R.utils::commandArgs(asValues=TRUE)$"f")))
       source("../../scripts/h2o-r-test-setup.R")
       check.deeplearning_stacked_autoencoder <- function() {</pre>
         # this function builds a vector of autoencoder models, one per layer
         get_stacked_ae_array <- function(training_data,layers,args){</pre>
           vector <- c()
           index = 0
           for(i in 1:length(layers)){
             index = index + 1
             ae model <- do.call(h2o.deeplearning,</pre>
                                  modifyList(list(x=names(training_data),
                                                   training_frame=training_data,
                                                   autoencoder=TRUE,
                                                   hidden=layers[i]),
             training_data = h2o.deepfeatures(ae_model,training_data,layer=1)
             names(training data) <- gsub("DF", paste0("L",index,sep=""), names(training data))</pre>
             vector <- c(vector, ae_model)</pre>
           }
           vector
         # this function returns final encoded contents
         apply_stacked_ae_array <- function(data,ae){</pre>
           index = 0
           for(i in 1:length(ae)){
             index = index + 1
```

```
data = h2o.deepfeatures(ae[[i]],data,layer=1)
    names(data) <- gsub("DF", paste0("L",index,sep=""), names(data))</pre>
  }
 data
}
TRAIN <- "bigdata/laptop/mnist/train.csv.gz"</pre>
TEST <- "bigdata/laptop/mnist/test.csv.gz"</pre>
response <- 785
# set to T for RUnit
# set to F for stand-alone demo
if (TRUE) {
 train_hex <- h2o.importFile(locate(TRAIN))</pre>
 test_hex <- h2o.importFile(locate(TEST ))</pre>
} else {
 library(h2o)
 h2o.init(nthreads=-1)
 homedir <- paste0(path.expand("~"),"/h2o-dev/") #modify if needed
 train_hex <- h2o.importFile(path = paste0(homedir,TRAIN), header = FALSE, sep = ',')</pre>
 test_hex <- h2o.importFile(path = paste0(homedir,TEST), header = FALSE, sep = ',')</pre>
train <- train_hex[,-response]</pre>
test <- test hex [,-response]
train_hex[,response] <- as.factor(train_hex[,response])</pre>
test_hex [,response] <- as.factor(test_hex [,response])</pre>
## Build reference model on full dataset and evaluate it on the test set
model_ref <- h2o.deeplearning(training_frame=train_hex, x=1:(ncol(train_hex)-1), y=response,</pre>
p ref <- h2o.performance(model ref, test hex)</pre>
h2o.logloss(p_ref)
## Now build a stacked autoencoder model with three stacked layer AE models
## First AE model will compress the 717 non-const predictors into 200
## Second AE model will compress 200 into 100
## Third AE model will compress 100 into 50
layers <- c(200,100,50)
args <- list(activation="Tanh", epochs=1, l1=1e-5)</pre>
ae <- get_stacked_ae_array(train, layers, args)</pre>
## Now compress the training/testing data with this 3-stage set of AE models
train_compressed <- apply_stacked_ae_array(train, ae)</pre>
test compressed <- apply stacked ae array(test, ae)
## Build a simple model using these new features (compressed training data) and evaluate it
train w resp <- h2o.cbind(train compressed, train hex[,response])</pre>
test w resp <- h2o.cbind(test compressed, test hex[,response])</pre>
model_on_compressed_data <- h2o.deeplearning(training_frame=train_w_resp, x=1:(ncol(train_w_</pre>
p <- h2o.performance(model on compressed data, test w resp)</pre>
h2o.logloss(p)
```

```
84 }
85 86 doTest("Deep Learning Stacked Autoencoder", check.deeplearning_stacked_autoencoder)
```