h2oai / **h2o-3**

Code Pull requests 60 Actions Projects Security Insights

master

...

h2o-3 / h2o-r / tests / testdir_algos / deeplearning / runit_deeplearning_stacked_autoencoder_large.R

wendycwong Replace T with TRUE and F with FALSE to pass R CMD Check. (#4425) ✕

4 contributors

86 lines (72 sloc) | 3.39 KB

```r
setwd(normalizePath(dirname(R.utils::commandArgs(asValues=TRUE)$"f")))
source("../../../scripts/h2o-r-test-setup.R")



check.deeplearning_stacked_autoencoder <- function() {
  # this function builds a vector of autoencoder models, one per layer
  get_stacked_ae_array <- function(training_data,layers,args){
    vector <- c()
    index = 0
    for(i in 1:length(layers)){
      index = index + 1
      ae_model <- do.call(h2o.deeplearning,
                          modifyList(list(x=names(training_data),
                                          training_frame=training_data,
                                          autoencoder=TRUE,
                                          hidden=layers[i]),
                                     args))
      training_data = h2o.deepfeatures(ae_model,training_data,layer=1)

      names(training_data) <- gsub("DF", paste0("L",index,sep=""), names(training_data))
      vector <- c(vector, ae_model)
    }
    vector
  }

  # this function returns final encoded contents
  apply_stacked_ae_array <- function(data,ae){
    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))
    }
    data
  }

  TRAIN <- "bigdata/laptop/mnist/train.csv.gz"
  TEST <- "bigdata/laptop/mnist/test.csv.gz"
  response <- 785

  # set to T for RUnit
  # set to F for stand-alone demo
  if (TRUE) {
    train_hex <- h2o.importFile(locate(TRAIN))
    test_hex  <- h2o.importFile(locate(TEST ))
  } 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 = ',')
    test_hex  <- h2o.importFile(path = paste0(homedir,TEST), header = FALSE, sep = ',')
  }
  train <- train_hex[,-response]
  test  <- test_hex [,-response]
  train_hex[,response] <- as.factor(train_hex[,response])
  test_hex [,response] <- as.factor(test_hex [,response])

  ## 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,
  p_ref <- h2o.performance(model_ref, test_hex)
  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)
  ae <- get_stacked_ae_array(train, layers, args)

  ## Now compress the training/testing data with this 3-stage set of AE models
  train_compressed <- apply_stacked_ae_array(train, ae)
  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])
  test_w_resp <- h2o.cbind(test_compressed, test_hex[,response])
  model_on_compressed_data <- h2o.deeplearning(training_frame=train_w_resp, x=1:(ncol(train_w_
  p <- h2o.performance(model_on_compressed_data, test_w_resp)
  h2o.logloss(p)


}

doTest("Deep Learning Stacked Autoencoder", check.deeplearning_stacked_autoencoder)
```