Week 5 Exercise: Neural Network with Keras

Melissa Wickers

11/5/2019

## Data Setup

We must first extract the critical\_temp column in order to use it for the response column. We do not want to include it with the training set. Then, the dataset is split into the training and test set. Once this is done, the data must then be scaled. The data must be split before this is done, otherwise, the information from the test set would be included in the training set.

data <- read.csv('Data/super.csv', stringsAsFactors = F)  
super.class <- data$critical\_temp # a vector  
super <- data[,names(data) != "critical\_temp"] # Remove response column  
#  
# Test/train split  
#  
set.seed (6881)  
train.ind <- sample (nrow (super), 18000, replace=FALSE)  
super.train <- as.matrix (super[train.ind,]) # no categoricals  
super.train.y <- matrix (super.class[train.ind], ncol = 1) # as matrix  
# If there had been factors we would have had to turn them into one-hot sets.  
  
super.test <- as.matrix (super[-train.ind,])   
super.test.y <- matrix (super.class[-train.ind], ncol = 1)  
  
#Now scale using only the training data.  
#  
super.train <- scale(super.train)  
super.test <- scale(super.test,  
 center = attributes(super.train)$'scaled:center',  
 scale = attributes(super.train)$'scaled:scale')

## Set Up Neural Network

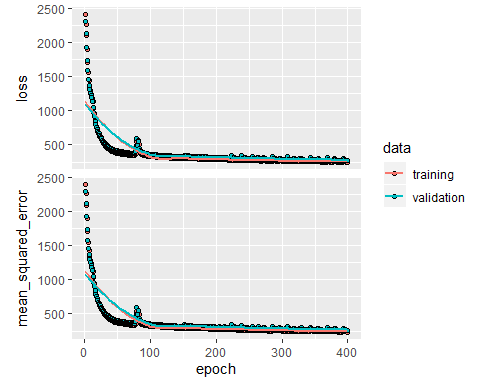
The package keras will be used to build the Neural Network. This particular network has an input layer, one hidden layer and an output layer. The activation for the input and hidden layer is sigmoid. We are predicting a number, therefore we must use linear as our output layer activation. We must compile the model in order to use it on our data. The loss and metric parameter is Mean Squared Error.

library(keras)  
reg <- regularizer\_l1\_l2 (l1 = 0.01, l2 = 0.01)  
mod <- keras\_model\_sequential ()%>% # I give you this for free!  
 layer\_dense(units = 81, activation = 'sigmoid', kernel\_regularizer = reg, input\_shape = 81)%>%  
 layer\_dense(units = 5, activation = 'sigmoid', kernel\_regularizer = reg)%>%  
 layer\_dense(units = 1)  
  
  
  
compile(mod, loss = 'mse', optimizer = optimizer\_sgd(lr = 0.01, momentum = 0.5, decay = 0.01), metrics = 'mse')

## Run the Model

Before running the model, we should split the data into a validation and training set. This helps us validate the model as it is running. As the model runs, it shows the MSE for the training and validation data. Ideally, we want these to be similar.

val.ind <- sample (18000, 2000, replace=F)  
  
history <- fit(mod,x = super.train[-val.ind,], y = super.train.y[-val.ind,],  
 validation\_data = list (super.train[val.ind,], super.train.y[val.ind,]),  
 epochs=400, batch\_size=18000)  
  
plot(history)



## RMSE and R-squared value

Now, we will use the test data along with our model to predict what the crtical\_temp. In order to determine how well the model performed, we can look at the RMSE and the R-squared value. We want an R-squared value greater than .74. This model has a RMSE of 15.7, and an R-Square of .71. The model performs well.

ModelPred <- predict(mod,super.test)  
  
##R-squared  
cor(super.test.y, ModelPred)^2

## [,1]  
## [1,] 0.8035336

##RMSE  
sqrt(mean((super.test.y - ModelPred)^2))

## [1] 15.70742