Classifying Hand Written Digits Using Neural Networks

Monika Baloda

Contents

Setup

- Going to utilize the mnist data for the challenge.
- Data is classifying hand written digits (0 - 10) for Y = 0, 1, 2, . . . 9 classification
- Each observation is 28 x 28 matrix of 0/1's representing a 28 x 28 pixel grid whether or not a pixel is filled in.
- Total training data size is a three dimensional array (60000 x 28 x 28) representing 60,000 data points
- We are going to use Neural Networks to build a model taking the pixel representation of X's and predict the 0-10 as a classifier

Loading the data and library

```
library(tensorflow)
library(keras)

# Load the MNIST dataset
mnist <- dataset_mnist()

# Split the dataset into training and testing sets
x_train <- mnist$train$x
y_train <- mnist$train$y
x_test <- mnist$test$x
y_test <- mnist$test$y


# Standardization of X points
x_train <- x_train / 255
x_test <-  x_test / 255
```

- Building a Neural Network requires 4 steps:
- Initialization of the Neural Network
- Neural Network Layer Architype
- Neural Network Optimization Algorithm setup
- Neural Network Fitting step

- Refer to Week 6 (Thursday) lecture slides on how to build a neural network

Step-1: - Initialization of the Neural Network

```
#I've setup the neural network initlization step, the data in a 28 x 28 matrix and is specified here
model <- keras_model_sequential(input_shape = c(28,28))  # Create a model object we will configure
```

Step-2 :- Neural Network Layer Architype

```
#I've included the first step to the Neural Network and that's to flatten the 28 x 28 matrix down
#Activation functions include: Relu, sigmoid, softplus, softsign, tanh, selu, elu, exponential

# Input layer
model <- model %>%
  layer_flatten(input_shape = c(28,28))

# Hidden layers
model <- model %>%
  layer_dense(units = 128, activation = "relu")
model <- model %>%
  layer_dense(units = 64, activation = "sigmoid")
model <- model %>%
  layer_dense(units = 32, activation = "softplus")
model <- model %>%
  layer_dense(units = 16, activation = "tanh")

# Output layer
model <- model %>%
  layer_dense(units = 10, activation = "softmax")

#Use this to see the number of parameters and design of your neural network
model
```

```
## Model: "sequential"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## ================================================================================
##  flatten (Flatten)                  (None, 784)                     0
##  dense (Dense)                      (None, 128)                     100480
##  dense_1 (Dense)                    (None, 64)                      8256
##  dense_2 (Dense)                    (None, 32)                      2080
##  dense_3 (Dense)                    (None, 16)                      528
##  dense_4 (Dense)                    (None, 10)                      170
## ================================================================================
## Total params: 111,514
## Trainable params: 111,514
## Non-trainable params: 0
## ________________________________________________________________________________
```

Step-3: Neural Network Optimization Algorithm setup

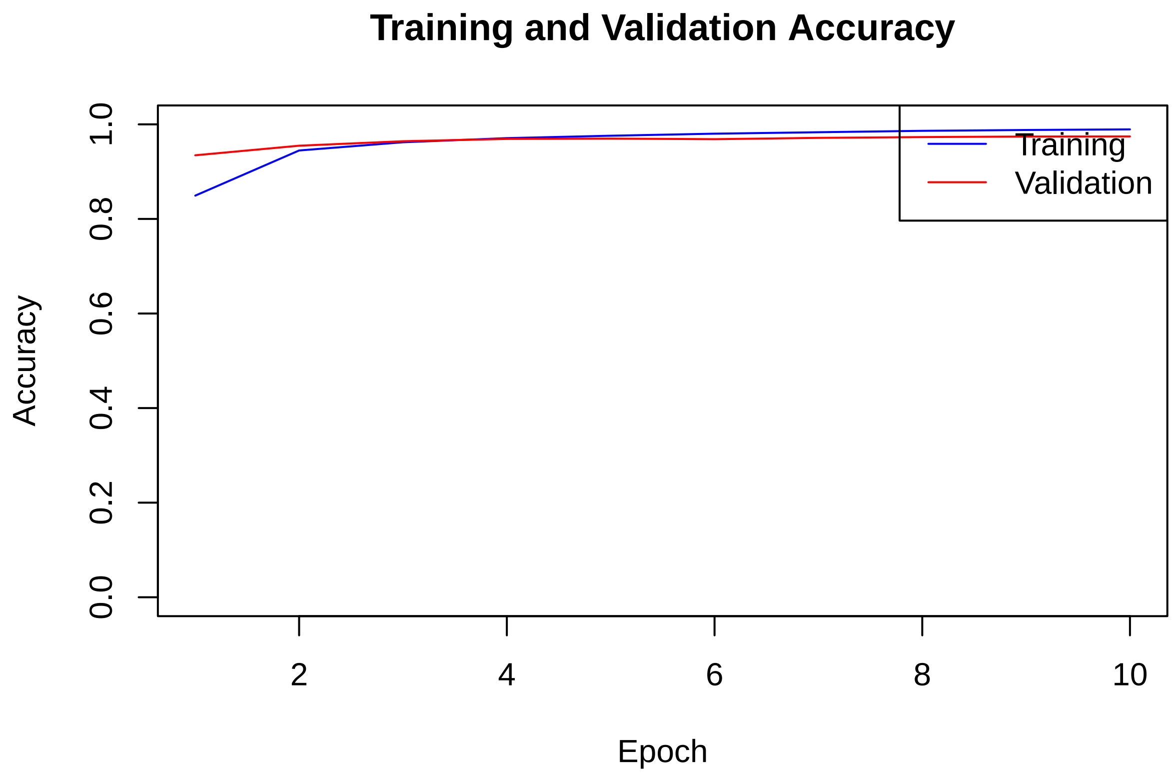
```r
model <- model %>%
  compile(loss = loss_sparse_categorical_crossentropy(from_logits = TRUE),
          optimizer = optimizer_rmsprop(), # Gradient Optimization Algorithm
          metrics = "accuracy")
```

Step-4 Neural Network Fitting step

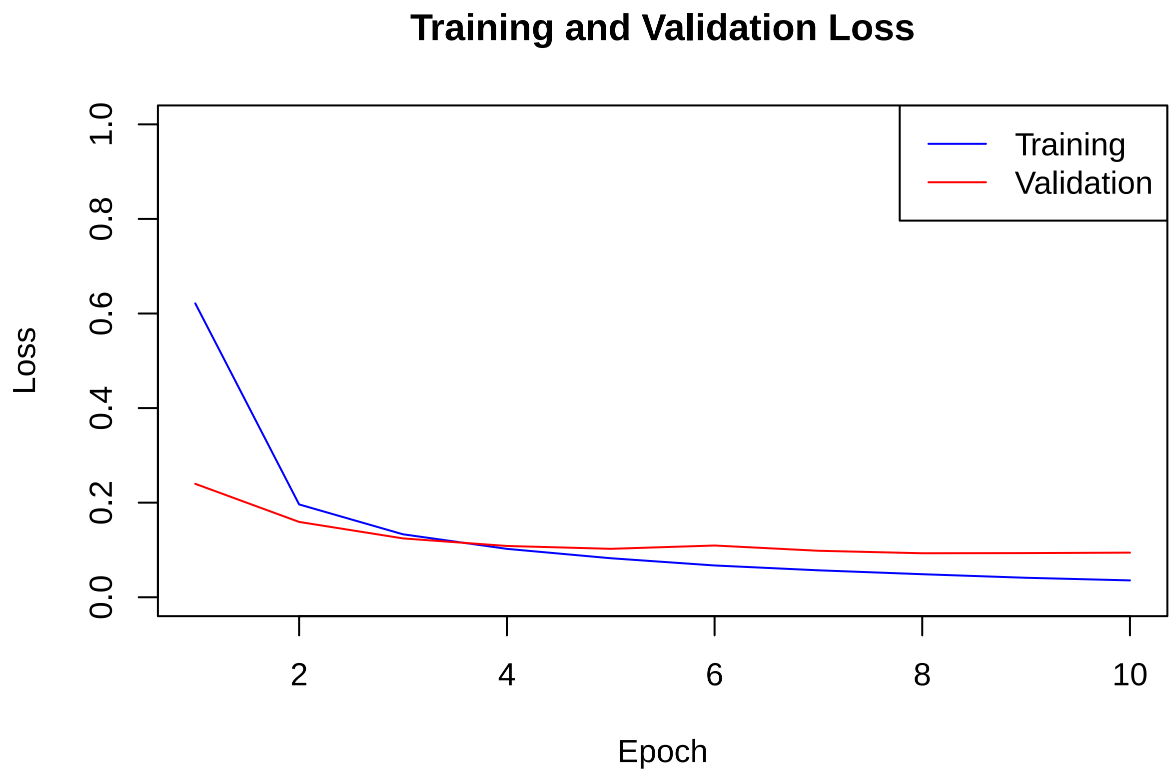
```r
history <-
  model %>%
  fit(x_train,y_train, # Plug in formatted Data
      batch_size=100, #
      epochs =10 ,
      validation_split = 0.2)
#print out the fitted summary
history
```

```
##
## Final epoch (plot to see history):
##         loss: 0.03564
##     accuracy: 0.9894
##     val_loss: 0.0943
## val_accuracy: 0.9745
```

```r
# Plot the training and validation accuracy
plot(history$metrics$accuracy, type = "l", col = "blue", xlab = "Epoch", ylab = "Accuracy", main = "Trai
      ,ylim=c(0,1))
lines(history$metrics$val_accuracy, type = "l", col = "red")
legend("topright", legend = c("Training", "Validation"), col = c("blue", "red"), lty = 1)
```




```
# Plot the training and validation loss
plot(history$metrics$loss, type = "l", col = "blue", xlab = "Epoch", ylab = "Loss", main = "Training and
lines(history$metrics$val_loss, type = "l", col = "red")
legend("topright", legend = c("Training", "Validation"), col = c("blue", "red"), lty = 1)
```


Training and Validation Loss
Training
Validation
Loss
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10
Epoch