Deep Learning with R - Dense Layer Model

Chenshu Liu

April 2022

Data & Preprocessing

```
library(readr)
library(keras)

setwd("~/Documents/Programming/R/Deep Learning with R/Datasets")
data <- read_csv("SimulatedBinaryClassificationDataset.csv", col_names = TRUE)

data  <- as.matrix(data)
dimnames(data) = NULL

# data splitting
set.seed(123)
index <- sample(2, nrow(data), replace = TRUE, prob = c(0.9, 0.1))
x_train <- data[index == 1, 1:10]
x_test <- data[index == 2, 1:10]
y_test_actual <- data[index == 2, 11]

# use to_categorical function in keras package for one-hot encoding
y_train <- to_categorical(data[index == 1, 11])
y_test <- to_categorical(data[index == 2, 11])
```

Dense Layer Model

```
model <- keras_model_sequential() %>%
  # layer_dense means a densely connected layer
  layer_dense(name = "DeepLayer1",
              units = 10, # hyperparameter: the number of nodes
              activation = "relu",
              # the first layer need to have specification about the input dimension
              input_shape = c(10)) %>%
  layer_dense(name = "DeepLayer2",
              units = 10,
              activation = "relu") %>%
  layer_dense(name = "OutputLayer",
              units = 2,
              # softmax function will provide probabilities of the nodes
              activation = "softmax")

model %>% compile(
  # another way to calculate loss, besides mean-squared-error
  loss = "categorical_crossentropy",
  # a special way of gradient descent
  optimizer = "adam",
  # measurement of model performance - using accuracy to measure
  metrics = c("accuracy"))

history <- model %>%
  fit(x_train,
      y_train,
      # number of full forward & backward propagation
      # (i.e. run 10 times back and forth of all samples)
      epoch = 10,
      # instead of propagating the whole dataset at one go, use smaller batches
      batch_size = 256,
      # splitting the training set to test itself during training
      validation_split = 0.1,
      verbose = 2)

model %>%
  evaluate(x_test,
           # NOTE: here we are still using the one-hot encoded y_test
           y_test)
```

```
##      loss  accuracy
## 0.1473719 0.9604589
```

```
# form predictions
pred <- model %>%
  predict(x_test) %>%
  k_argmax()
pred <- as.array(pred)
table(Predicted = pred,
      # NOTE: for confusion matrix, we are using the original y_test_actual, not encoded
      Actual = y_test_actual)
```

```
##           Actual
## Predicted    0    1
##         0 2420  155
##         1   38 2268
```

Model Evaluation

```
summary(model)
```

```
## Model: "sequential"
## _________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## =========================================================================
##  DeepLayer1 (Dense)                 (None, 10)                      110
##
##  DeepLayer2 (Dense)                 (None, 10)                      110
##
##  OutputLayer (Dense)                (None, 2)                       22
##
## =========================================================================
## Total params: 242
## Trainable params: 242
## Non-trainable params: 0
## _________________________________________________________________________
```

```
# plot the training history
plot(history)
```

```
## `geom_smooth()` using formula 'y ~ x'
```