Deep Learning with R - Regularization

Chenshu Liu

April 2022

Data & Preprocessing

```r
# Applying regularization to deal with overfitting
library(keras)
library(readr)
library(tidyr)
library(tibble)

# specify the number of feature variables for the dataset to be downloaded
num_words <- 5000
imdb <- dataset_imdb(num_words = num_words)

# train test split
c(train_data, train_labels) %<-% imdb$train
c(test_data, test_labels) %<-% imdb$test

# multi-hot encoding
multi_hot_sequences <- function(sequences, dimension){
  multi_hot <- matrix(0,
                      # the number of samples in the sequences
                      # sequences are stored as lists
                      nrow = length(sequences),
                      ncol = dimension)
  for(i in 1 : length(sequences)){
    # sequences[[i]] extracts the label of the words in the text sample i
    # which ever word is included in that sequence will be assigned 1 at row i
    multi_hot[i, sequences[[i]]] <- 1
  }
  multi_hot
}

train_data <- multi_hot_sequences(train_data, num_words)
test_data <- multi_hot_sequences(test_data, num_words)
```

L2 Regularization Model

```r
l2_model <-
  keras_model_sequential() %>%
  layer_dense(units = 16, activation = "relu", input_shape = num_words,
              # apply regularization in the layer_dense function's argument
              kernel_regularizer = regularizer_l2(l = 0.001)) %>%
  layer_dense(units = 16, activation = "relu",
              kernel_regularizer = regularizer_l2(l = 0.001)) %>%
  layer_dense(units = 1, activation = "sigmoid")

l2_model %>% compile(
  optimizer = "adam",
  loss = "binary_crossentropy",
  metrics = list("accuracy")
)

l2_history <- l2_model %>% fit(
  train_data,
  train_labels,
  epoch = 20,
  batch_size = 512,
  validation_data = list(test_data, test_labels),
  verbose = 2
)
```

Dropout Regularization Model

```r
drop_model <- keras_model_sequential() %>%
  layer_dense(units = 16, activation = "relu", input_shape = num_words) %>%
  # a new layer to specify the dropout rate
  layer_dropout(0.6) %>%
  layer_dense(units = 16, activation = "relu") %>%
  layer_dropout(0.6) %>%
  layer_dense(units = 1, activation = "sigmoid")

drop_model %>% compile(
  optimizer = "adam",
  loss = "binary_crossentropy",
  metrics = list("accuracy")
)

drop_history <- drop_model %>% fit(
  train_data,
  train_labels,
  epoch = 20,
  batch_size = 512,
  validation_data = list(test_data, test_labels),
  verbose = 2
)
```