Advanced Machine Learning - Assignment 3

Julia Thacker

4/11/2022

```
library(keras)

## Warning: package 'keras' was built under R version 4.1.2

max_features <- 10000
maxlen <- 500
batch_size <- 32

cat("Loading data...\n")

## Loading data...

imdb <- dataset_imdb(num_words = max_features)

## Loaded Tensorflow version 2.8.0

c(c(input_train, y_train), c(input_test, y_test)) %<-% imdb
cat(length(input_train), "train sequences\n")

## 25000 train sequences

cat(length(input_test), "test sequences")

## 25000 test sequences

cat("Pad sequences (samples x time)\n")

## Pad sequences (samples x time)

input_train <- pad_sequences(input_train, maxlen = maxlen)
input_test <- pad_sequences(input_test, maxlen = maxlen)
cat("input_train shape:", dim(input_train), "\n")

## input_train shape: 25000 500

cat("input_test shape:", dim(input_test), "\n")

## input_test shape: 25000 500

model <- keras_model_sequential() %>%
  layer_embedding(input_dim = max_features, output_dim = 32) %>%
  layer_simple_rnn(units = 32) %>%
  layer_dense(units = 1, activation = "sigmoid")

model %>% compile(
```

```
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("acc")
)

history <- model %>% fit(
  input_train, y_train,
  epochs = 10,
  batch_size = 128,
  validation_split = 0.2
)
```

Ran the IMDB example model from chapter 6.2 above, then created a new model using the criteria from the assignment instructions.

```
max_features2 <- 10000 #Considered only the 10,000 words
maxlen2 <- 150 #Cutoff reviews at 150 words
batch_size2 <- 32

cat("Loading data...\n")

## Loading data...

imdb <- dataset_imdb(num_words = max_features2)
c(c(input_train2, y_train2), c(input_test2, y_test2)) %<-% imdb
cat(length(input_train2), "train sequences\n")

## 25000 train sequences

cat(length(input_test2), "test sequences")

## 25000 test sequences

cat("Pad sequences (samples x time)\n")

## Pad sequences (samples x time)

input_train2 <- pad_sequences(input_train2[1:100], maxlen = maxlen2) #reduced
the training data to 100 samples
input_test2 <- pad_sequences(input_test2, maxlen = maxlen2)
cat("input_train shape:", dim(input_train2), "\n")

## input_train shape: 100 150

cat("input_test shape:", dim(input_test2), "\n")

## input_test shape: 25000 150

model2 <- keras_model_sequential() %>%
  layer_embedding(input_dim = max_features2, output_dim = 32) %>%
  layer_simple_rnn(units = 32) %>%
  layer_dense(units = 1, activation = "sigmoid")
```

```
model2 %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("acc")
)

history2 <- model2 %>% fit(
  input_train2, y_train2,
  epochs = 10,
  batch_size = 128,
  validation_split=0.4 #Set the validation data to 40%/10,000 samples
)
```

Next, I created a third model using the same variables but with an embedding layer.

```
max_features3 <- 10000 #Considered only the 10,000 words
maxlen3 <- 150 #Cutoff reviews at 150 words

imdb <- dataset_imdb(num_words = max_features3)
c(c(x_train3, y_train3), c(x_test3, y_test3)) %<-% imdb

x_train3 <- pad_sequences(x_train3[1:100], maxlen = maxlen3) #reduced the
training data to 100 samples
x_test3 <- pad_sequences(x_test3, maxlen = maxlen3)

model3 <- keras_model_sequential() %>%
  layer_embedding(input_dim = 10000, output_dim = 8,
                  input_length = maxlen3) %>%
  layer_flatten() %>%
  layer_dense(units = 1, activation = "sigmoid")

model3 %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("acc")
)

history3 <- model3 %>% fit(
  x_train3, y_train3,
  epochs = 10,
  batch_size = 32,
  validation_split = 0.4 #Set the validation data to 40%/10,000 samples
)
```

My fourth model will use the same variables, with a pretrained word embedding.

```
imdb_dir <- "~/Downloads/aclimdb"
train_dir <- file.path(imdb_dir, "train")

labels <- c()
```

```
texts <- c()

for (label_type in c("neg", "pos")) {
  label <- switch(label_type, neg = 0, pos = 1)
  dir_name <- file.path(train_dir, label_type)
  for (fname in list.files(dir_name, pattern = glob2rx("*.txt"),
                           full.names = TRUE)) {
    texts <- c(texts, readChar(fname, file.info(fname)$size))
    labels <- c(labels, label)
  }
}

library(keras)

maxlen4 <- 150
training_samples4 <- 100
validation_samples4 <- 10000
max_words4 <- 10000

tokenizer <- text_tokenizer(num_words = max_words4) %>%
  fit_text_tokenizer(texts)

sequences <- texts_to_sequences(tokenizer, texts)

word_index = tokenizer$word_index
cat("Found", length(word_index), "unique tokens.\n")

## Found 88582 unique tokens.

data <- pad_sequences(sequences, maxlen = maxlen4)

labels <- as.array(labels)
cat("Shape of data tensor:", dim(data), "\n")

## Shape of data tensor: 25000 150

cat('Shape of label tensor:', dim(labels), "\n")

## Shape of label tensor: 25000

indices <- sample(1:nrow(data))
training_indices4 <- indices[1:training_samples4]
validation_indices4 <- indices[(training_samples4 + 1):
                               (training_samples4 + validation_samples4)]

x_train <- data[training_indices4,]
y_train <- labels[training_indices4]

x_val <- data[validation_indices4,]
y_val <- labels[validation_indices4]
```

```
glove_dir = '~/Downloads/glove.6B'
lines <- readLines(file.path(glove_dir, "glove.6B.100d.txt"))

embeddings_index <- new.env(hash = TRUE, parent = emptyenv())
for (i in 1:length(lines)) {
  line <- lines[[i]]
  values <- strsplit(line, " ")[[1]]
  word <- values[[1]]
  embeddings_index[[word]] <- as.double(values[-1])
}
cat("Found", length(embeddings_index), "word vectors.\n")

## Found 400000 word vectors.

embedding_dim <- 100

embedding_matrix <- array(0, c(max_words4, embedding_dim))

for (word in names(word_index)) {
  index <- word_index[[word]]
  if (index < max_words4) {
    embedding_vector <- embeddings_index[[word]]
    if (!is.null(embedding_vector))
      embedding_matrix[index+1,] <- embedding_vector
  }
}

model4 <- keras_model_sequential() %>%
  layer_embedding(input_dim = max_words4, output_dim = embedding_dim,
                  input_length = maxlen4) %>%
  layer_flatten() %>%
  layer_dense(units = 32, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")

get_layer(model4, index = 1) %>%
  set_weights(list(embedding_matrix)) %>%
  freeze_weights()

model4 %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("acc")
)

history4 <- model4 %>% fit(
  x_train, y_train,
  epochs = 20,
  batch_size = 32,
  validation_data = list(x_val, y_val)
)
```

This model had slightly more validation loss and slightly lower validation accuracy than the prior model with the embedding layer. Using an embedding layer seems to work better than using a pretrained word embedding.

Next, I will replicate my third model, but adjust the number of training samples to see if I can improve performance.

```
max_features5 <- 10000
maxlen5 <- 150

imdb <- dataset_imdb(num_words = max_features5)
c(c(x_train5, y_train5), c(x_test5, y_test5)) %<-% imdb

x_train5 <- pad_sequences(x_train5[1:15000], maxlen = maxlen5) #Increased the
training data to 15000 samples
x_test5 <- pad_sequences(x_test5, maxlen = maxlen5)

model5 <- keras_model_sequential() %>%
  layer_embedding(input_dim = 10000, output_dim = 8,
                  input_length = maxlen5) %>%
  layer_flatten() %>%
  layer_dense(units = 1, activation = "sigmoid")

model5 %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("acc")
)

history5 <- model5 %>% fit(
  x_train5, y_train5,
  epochs = 10,
  batch_size = 32,
  validation_split = 0.4
)
```

A training data set of 15,000 samples resulted in 86% validation accuracy and only 33% validation loss. This is the best performance of the 5 models I have tested above.