Assignment 3

Jacob Fabian

2023-04-13

Setting up our model and loading IMBD Dataset from http://ai.stanford.edu/~amaas/data/sentiment

Setting our

Cutoff reviews at 150

Restricting training samples to 100

Validating on 10,000 samples

Considering the top 10,000 words

```
library(keras)
maxlen <- 150
training_samples <- 100
validation_samples <- 10000
max_features <- 10000
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)
  }
}
```

Using the parameters we set earlier, we will use them for validation and training size

```
tokenizer <- text_tokenizer(num_words = max_features) %>%
  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 = maxlen)
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_indices <- indices[1:training_samples]
validation_indices <- indices[(training_samples + 1):
                              (training_samples + validation_samples)]
x_train <- data[training_indices,]
y_train <- labels[training_indices]
x_val <- data[validation_indices,]
y_val <- labels[validation_indices]
```

Download the GloVe word embeddings from https://nlp.stanford.edu/projects/glove/

```
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 our matrix

```
embedding_dim <- 100
embedding_matrix <- array(0, c(max_features, embedding_dim))
for(word in names(word_index)) {
```

```
  index <- word_index[[word]]
  if (index < max_features) {
    embedding_vector <- embeddings_index[[word]]
    if (!is.null(embedding_vector))
      embedding_matrix[index+1,] <- embedding_vector
  }
}
```

Defining our pre-trained model

```
model1 <- keras_model_sequential() %>%
  layer_embedding(input_dim = max_features, output_dim = embedding_dim,
                  input_length = maxlen) %>%
  layer_flatten() %>%
  layer_dense(units = 32, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")
```

Putting GloVe in our model

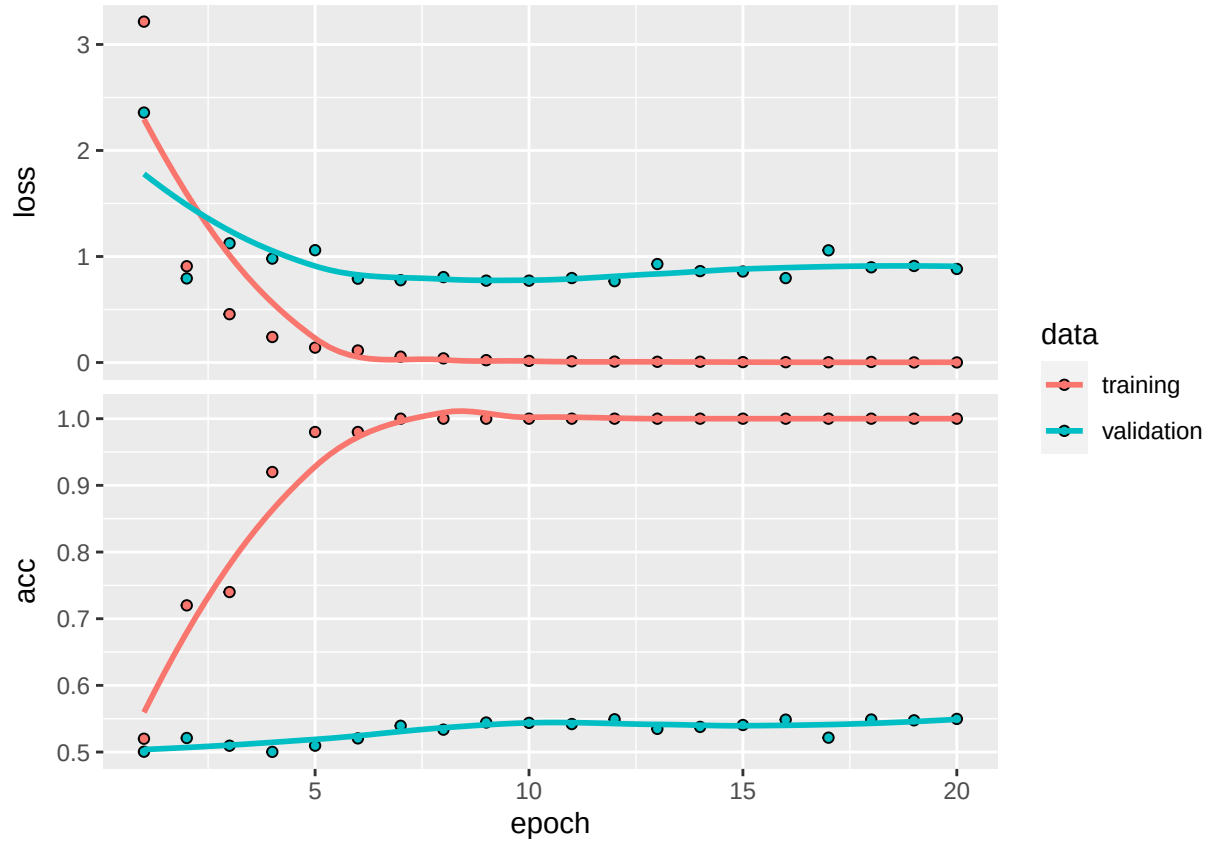
```
get_layer(model1, index = 1) %>%
  set_weights(list(embedding_matrix)) %>%
  freeze_weights()
```

Training and Evaluation

```
model1 %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("acc")
)
history1 <- model1 %>% fit(
  x_train, y_train,
  epochs = 20,
  batch_size = 32,
  validation_data = list(x_val, y_val)
)
save_model_weights_hdf5(model1, "pre_trained_glove-model1.h5")
```

Plotting and Comparing

```
plot(history1)
```



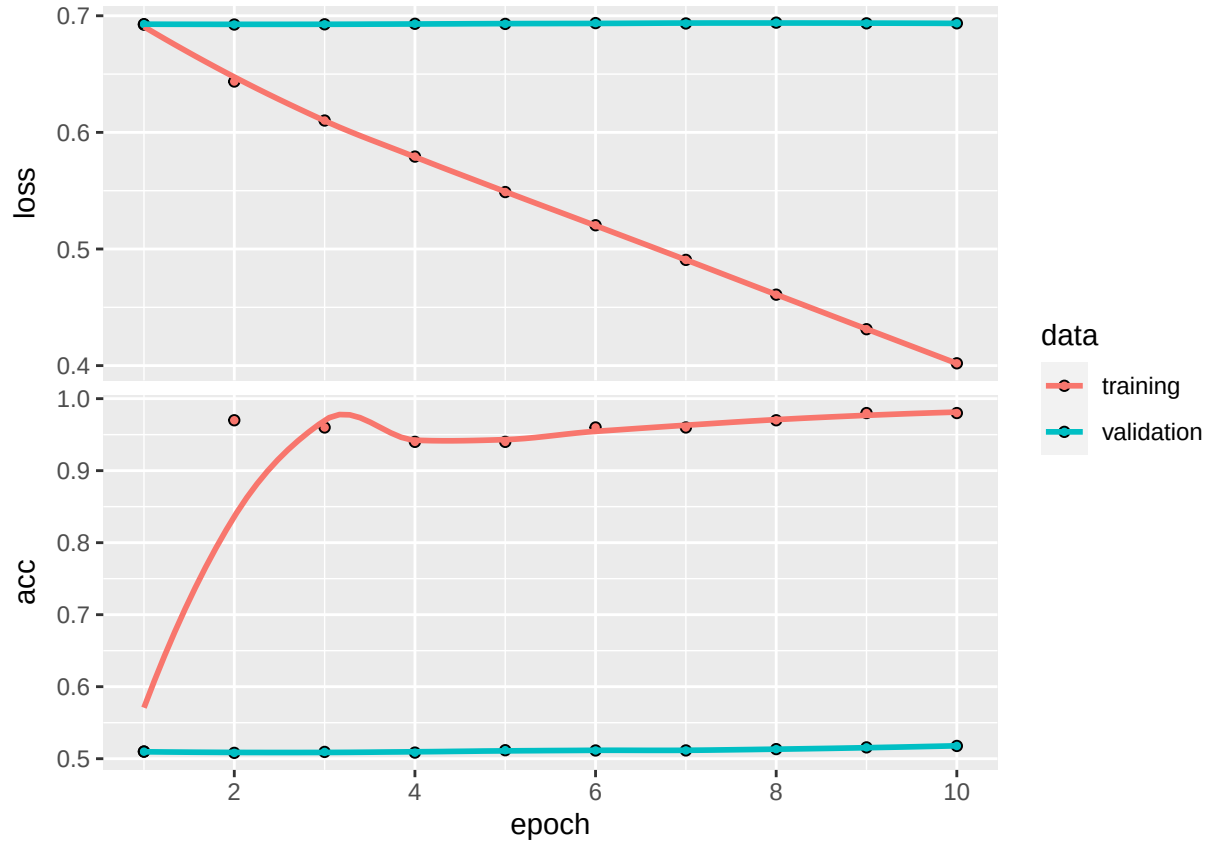
Since our training sample size is small, our model overfits. This tends to happen with sample sizes being this small. At 56% accuracy our model starts to bend.

Building an embedding layer instead of relying on the pre-trained set.

```
model2 <- keras_model_sequential() %>%
  layer_embedding(input_dim = max_features, output_dim = 32,
                  input_length = maxlen) %>%
  layer_flatten() %>%
  layer_dense(units = 1, activation = "sigmoid")
model2 %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("acc")
)
history2 <- model2 %>% fit(
  x_train, y_train,
  epochs = 10,
  batch_size = 32,
  validation_data = list(x_val, y_val)
)
save_model_weights_hdf5(model2, "assignment_3_model2.h5")
```

Plotting the results

```
plot(history2)
```



This model has the best accuracy at 51% and does not preform better than the previous model.

Optimizing our model, adjusting our training sample size to preform better.

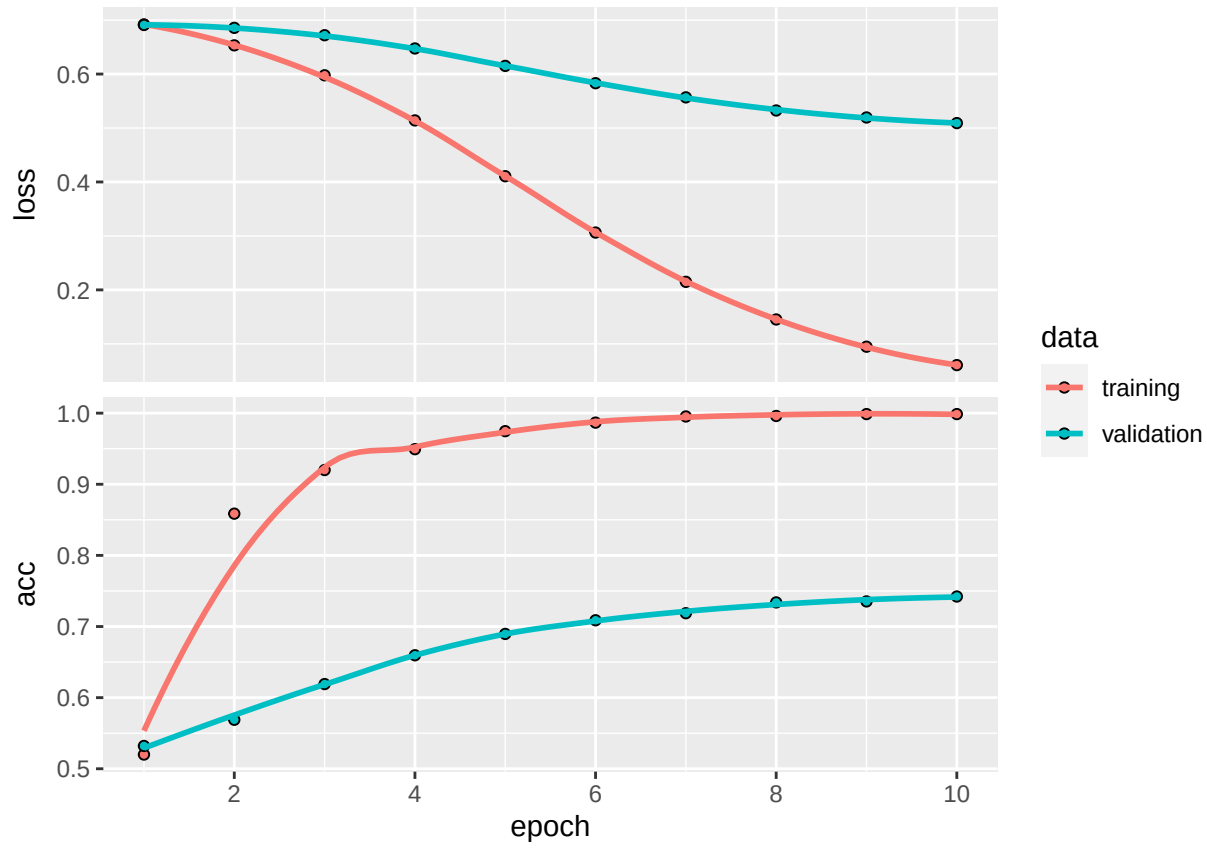
```
training_samples2 <- 1500
training_indices2 <- indices[1:training_samples2]
validation_indices2 <- indices[(training_samples2 + 1):
                               (training_samples2 + validation_samples)]
x_train2 <- data[training_indices2,]
y_train2 <- labels[training_indices2]
x_val2 <- data[validation_indices2,]
y_val2 <- labels[validation_indices2]
```

Adjusting our sample size of traingng data from 100 to 1500 units.

```
model3 <- keras_model_sequential() %>%
  layer_embedding(input_dim = max_features, output_dim = 32,
                  input_length = maxlen) %>%
  layer_flatten() %>%
  layer_dense(units = 1, activation = "sigmoid")
model3 %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("acc")
)
history3 <- model3 %>% fit(
  x_train2, y_train2,
  epochs = 10,
  batch_size = 32,
  validation_data = list(x_val2, y_val2)
)
save_model_weights_hdf5(model2, "assignment_3_model3.h5")
```

Plotting our new model.

```
plot(history3)
```



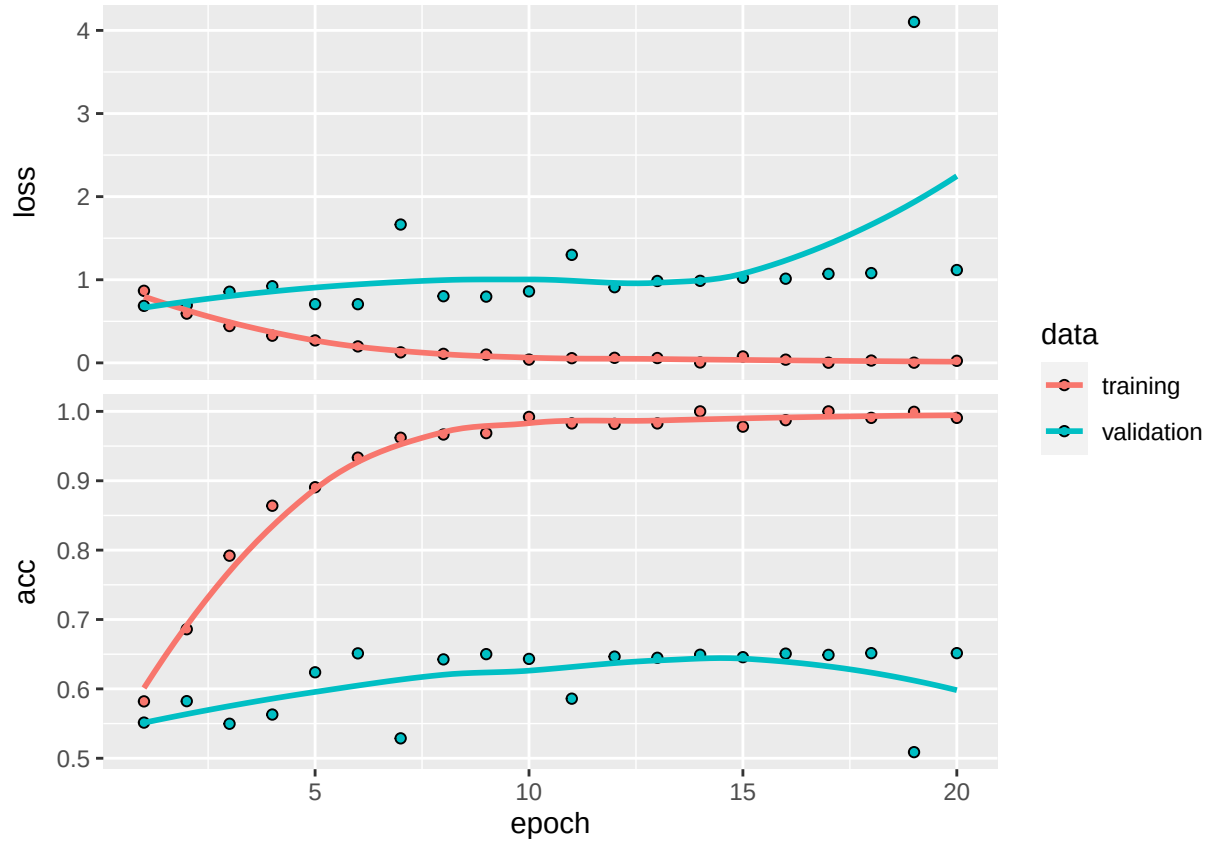
The best preformance with this model occurs around epoch 7. Val_acc at 71% and Val_loss at 56% which is better than previous models. We will continue to see improvement as we increase the training data.

New Sample Size

```
history4 <- model1 %>% fit(
  x_train2, y_train2,
  epochs = 20,
  batch_size = 32,
  validation_data = list(x_val2, y_val2)
)
```

Plotting our new model.

```
plot(history4)
```



New sample size with the pretrained model shows it preforms better at this sample size.