AML Assignment 3 Cronin

Hannah Cronin

2023-04-16

I made each of these changes individually on a LSTR model (see my email) for extra analysis

```
library(keras)
max_features <- 10000  # Number of words to consider as features
maxlen <- 500  # Cuts off texts after this many words (among the max_features most common words)
batch_size <- 32
cat("Loading data...\n")
```

```
## Loading data...
```

```
imdb <- dataset_imdb(num_words = max_features)
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")
```

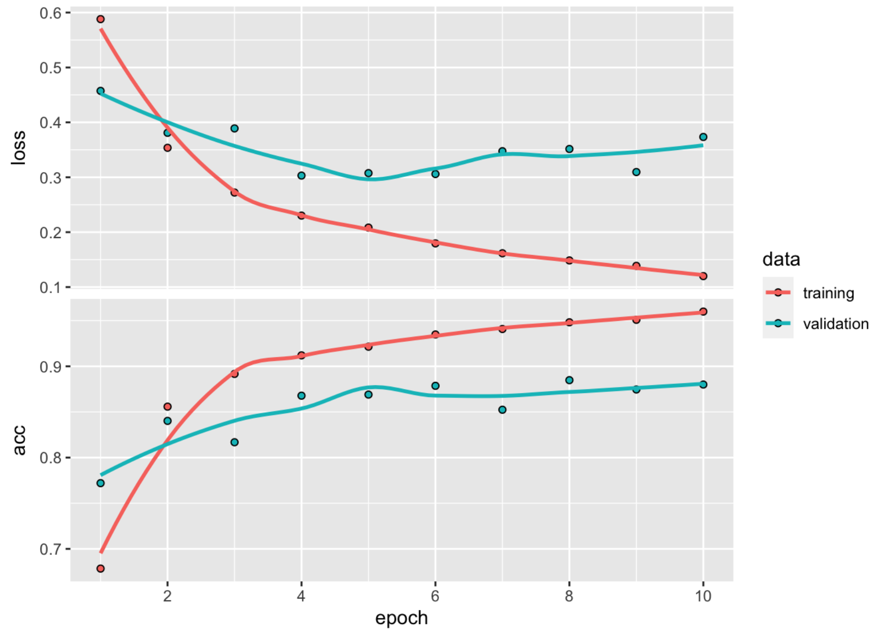
```
## input_train shape: 25000 500
```

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

```
## input_test shape: 25000 500
```

```
set.seed(123)
model <- keras_model_sequential() %>%
  layer_embedding(input_dim = max_features, output_dim = 32) %>%
  layer_lstm(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
)
```

```
plot(history)
```



```
history
```

```
##
## Final epoch (plot to see history):
##     loss: 0.12
##      acc: 0.9599
## val_loss: 0.3734
##  val_acc: 0.88
```

1. Cut off Reviews after 150 words

2. Restrict training samples to 100

```
r max_features <- 10000  # Number of words to consider as features maxlen <- 500  # Cuts off texts after this many words (among the m
```

```
## Loading data...
```

```
r imdb <- dataset_imdb(num_words = max_features) c(c(input_train, y_train), c(input_test, y_test)) %<-% imdb cat(length(input_train),
```

```
## 25000 train sequences
```

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

```
## 25000 test sequences
```

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

```
## Pad sequences (samples x time)
```

```
r input_train <- pad_sequences(input_train, maxlen = maxlen) input_test <- pad_sequences(input_test, maxlen = maxlen) cat("input_trai
```

```
## input_train shape: 25000 500
```

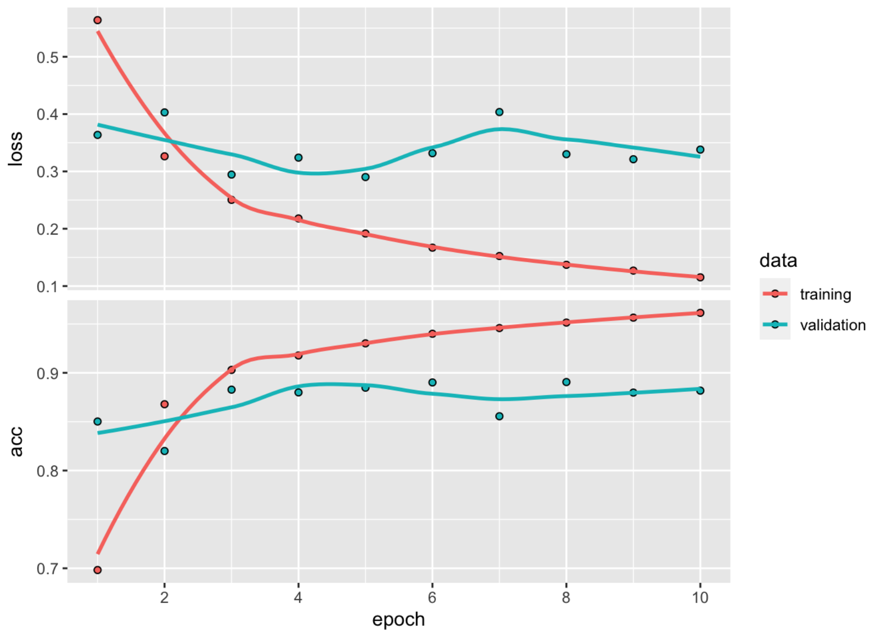
```
r cat("input_test shape:", dim(input_test), "\n")
```

```
## input_test shape: 25000 500
```

```
r set.seed(123) model2 <- keras_model_sequential() %>% layer_embedding(input_dim = max_features, output_dim = 32) %>% layer_lstm(unit
```

```
r plot(history2)
```

1. Cut off Reviews after 150 words



```
r history2
```

`## ## Final epoch (plot to see history): ##      loss: 0.1152 ##       acc: 0.9614 ## val_loss: 0.3381 ##  val_acc: 0.8818` With training s

3. Validate on 10,000 samples

```
max_features <- 10000  # Number of words to consider as features
maxlen <- 500  # Cuts off texts after this many words (among the max_features most common words)
batch_size <- 32
cat("Loading data...\n")
```

```
## Loading data...
```

```
imdb <- dataset_imdb(num_words = max_features)
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")
```

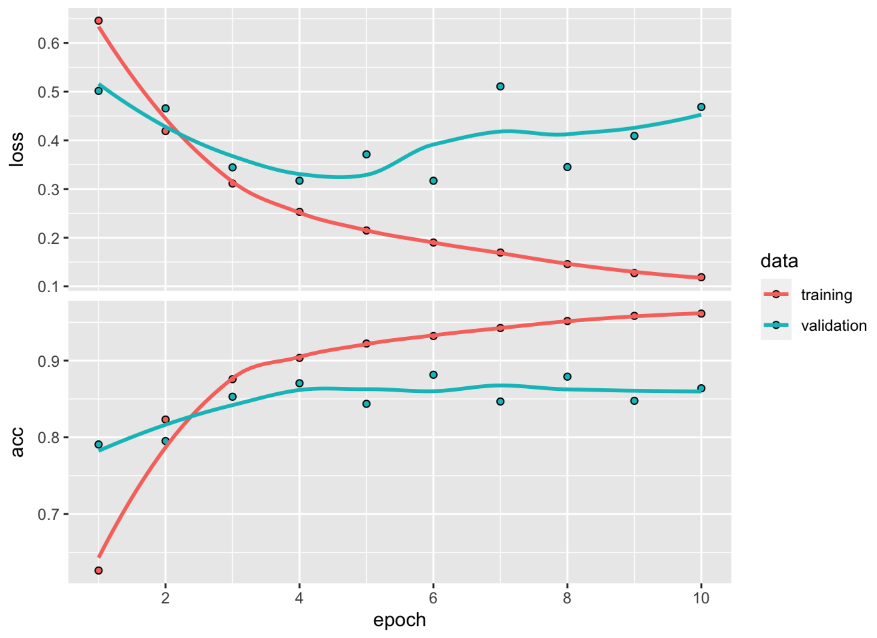
```
## input_train shape: 25000 500
```

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

```
## input_test shape: 25000 500
```

```
set.seed(123)
model3 <- keras_model_sequential() %>%
  layer_embedding(input_dim = max_features, output_dim = 32) %>%
  layer_lstm(units = 32) %>%
  layer_dense(units = 1, activation = "sigmoid")
model3 %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("acc")
)
history3 <- model3 %>% fit(
  input_train, y_train,
  epochs = 10,
  batch_size = 128,
  validation_split = 0.4
)
```

```
plot(history3)
```



```
history3
```

```
##
## Final epoch (plot to see history):
##     loss: 0.1189
##      acc: 0.9613
## val_loss: 0.4685
##  val_acc: 0.8639
```

The training accuracy and loss were very similar to the original - maybe a little better and validation accuracy and loss performed a little worse than the original.

4. Keep top 10,000 words

Combined model (for fun)

```
r top_words <- 10000  # Top 10,000 words maxlen <- 150  # Cuts off texts after this many words (among the max_features most common wo
```

```
## Loading data...
```

```
r imdb <- dataset_imdb(num_words = top_words) c(c(input_train, y_train), c(input_test, y_test)) %<-% imdb cat(length(input_train), "t
```

```
## 25000 train sequences
```

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

```
## 25000 test sequences
```

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

```
## Pad sequences (samples x time)
```

```
r input_train <- pad_sequences(input_train, maxlen = maxlen) input_test <- pad_sequences(input_test, maxlen = maxlen) cat("input_trai
```

4. Keep top 10,000 words

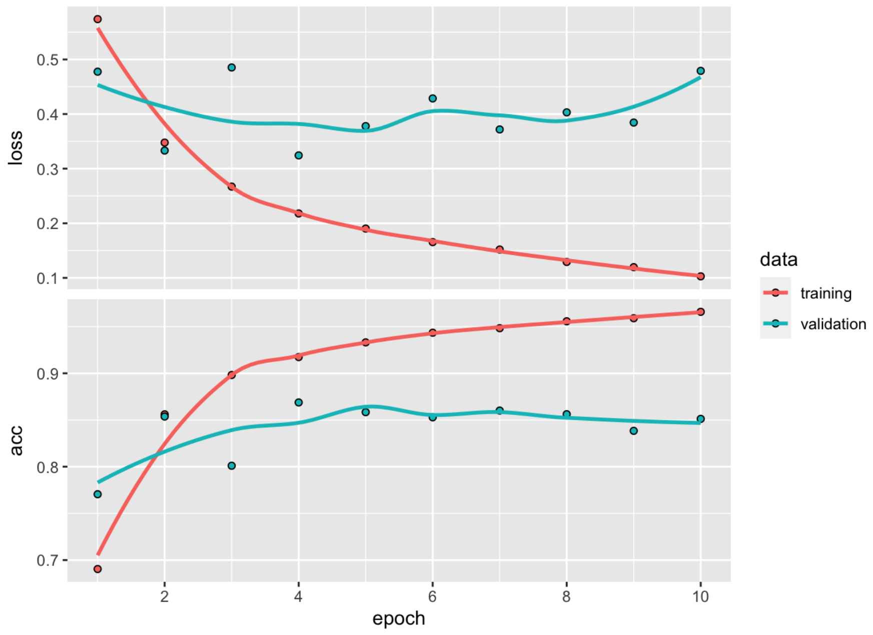
```
## input_train shape: 25000 150
```

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

```
## input_test shape: 25000 150
```

```
r set.seed(123) model5 <- keras_model_sequential() %>% layer_embedding(input_dim = max_features, output_dim = 32) %>% layer_lstm(unit
```

```
r plot(history5)
```



```
r history5
```

```
## ## Final epoch (plot to see history): ##     loss: 0.1029 ##      acc: 0.9659 ## val_loss: 0.4791 ##  val_acc: 0.8512
```

Embedding Layers:

```
imdb_dir <- "/Users/hannahcronin/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)
  }
}
```

```
embedding_dim <- 100
maxlen <- 150
training_samples <- 100
validation_samples <- 10000
max_words <- 10000
tokenizer <- text_tokenizer(num_words = max_words) %>%
  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]
```

```
set.seed(123)
modela <- keras_model_sequential() %>%
  layer_embedding(input_dim = max_words, output_dim = embedding_dim,
                  input_length = maxlen) %>%
  layer_flatten() %>%
  layer_dense(units = 32, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")
modela %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("acc")
)
historya <- modela %>% fit(
  x_train, y_train,
  epochs = 20,
  batch_size = 32,
  validation_data = list(x_val, y_val)
)
```

This model did not perform well and is very overfitted.

```
embedding_dim <- 100
maxlen <- 150
training_samples <- 10000
validation_samples <- 10000
max_words <- 10000
tokenizer <- text_tokenizer(num_words = max_words) %>%
  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
```

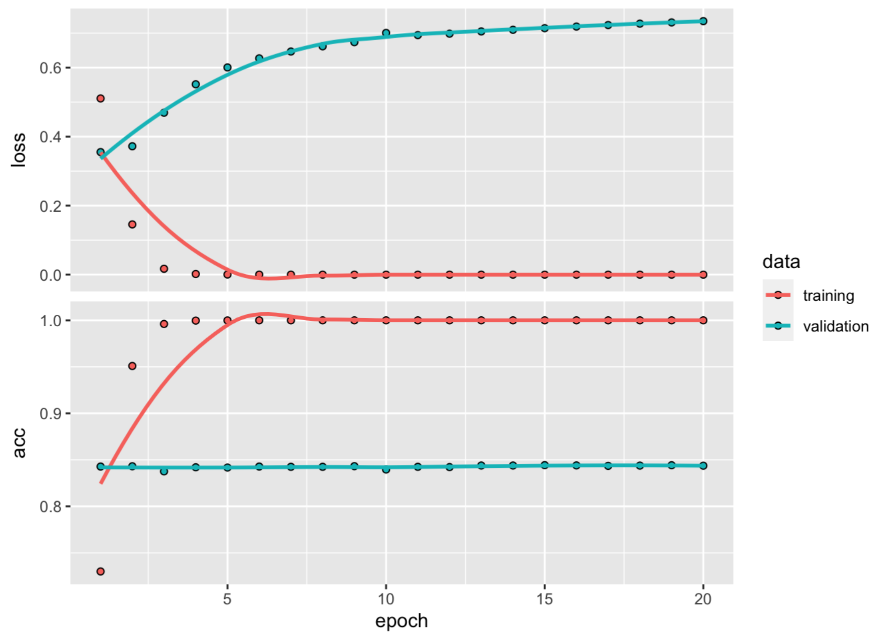
```
cat('Shape of label tensor:', dim(labels), "\n")
```

```
## 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]
```

```
set.seed(123)
modelb <- keras_model_sequential() %>%
  layer_embedding(input_dim = max_words, output_dim = embedding_dim,
                  input_length = maxlen) %>%
  layer_flatten() %>%
  layer_dense(units = 32, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")
modelb %>% compile(
  optimizer = "rmsprop",
  loss = "binary_crossentropy",
  metrics = c("acc")
)
historyb <- modelb %>% fit(
  x_train, y_train,
  epochs = 20,
  batch_size = 32,
  validation_data = list(x_val, y_val)
)
```

```
plot(historyb)
```



```
historyb
```

```
##
## Final epoch (plot to see history):
##     loss: 0.000008103
##      acc: 1
## val_loss: 0.7346
##  val_acc: 0.8437
```

The model is still overfitted, but performances substantially better against the validation data when the the training sample sizes were 10,000.