Text: convolution and embeddings

Thomas Lumley

9/26/2018

We're trying to classify movie reviews as positive or negative. To do this, we need to feed a sequence of words into the classifier.

First, set up the data

```
library(keras)

max_features <- 5000
imdb <- dataset_imdb(num_words = max_features)
imdb$train$x[[22]]
```

```
##   [1]    1  466   49 2036  204 2442   40    4    2  732   15 1754    4  978
##  [15]   18    4    2    2 1890    2    2    5   41 4246   14   22  734  346
##  [29]    5 1116    2    2    2    2  186 1020   21    9 1053   17    2    6
##  [43]  109   37   34 1696   22    9 3747    8  164   53   74 4488   33    6
##  [57]    2  149  761    2    2   35    2    2    2  304  125 3355   54  761
##  [71]    2    2    2  739    4  370    2   23   41 2752 3046    2   60  231
##  [85]   41  977    8   30    2   42    2 2504    2   83  139    8    2    2
##  [99]    2    2    7   41   10   10    6    2    2    7    2    2   62   28
## [113]   77 2640   21   14   22  472  166    6    2    7  199  761    2  113
## [127]    5   15    7    6  682  251 1222 1656    5   41    2 1309    2 1432
```

```
word_index<-dataset_imdb_word_index()
word_index<-do.call(c, word_index)
head(word_index)
```

```
##    fawn tsukino nunnery   sonja     gag   woods
##   34701   52006   52007   16816    3285    1408
```

```
review<-function(numbers){
  lookup<-na.omit(match(numbers, word_index))
paste(names(word_index)[lookup],collapse=" ")
}

review(imdb$train$x[[22]])
```

```
## [1] "the throughout good scream i've angles just of and oscar for memory of meant
but of and and responsible and and to about relations as you enjoyable men to 15 and
and and and horror success not it gun movie and is being like who phone you it goldbe
rg in director up been elephant they is and doesn't feels and and so and and and beau
tiful better loser no feels and and and george of awful and are about pregnant poigna
nt and which minutes about girlfriend in at and it's and accidentally and first somet
hing in and and and and br about i i is and and br and and story one will dean not as
you \u0096 find is and br give feels and acting to for br is disappointed hard planet
ed to about and door and serial"
```

Now the model. `layer_embedding` is word-vector embedding layer that turns words into positions in a Euclidean space. In this case, we want to project the most-common 5000 words into 50-dimensional space. For this approach we need constant-length texts, so we represent the words by number labels, and pad each review to 400 words with zeroes.

The embedding layer first sets up positions in 50-d space for each word at random. Call these $x_{i1}, \ldots, x_{i50}$. It then defines weights $w_{i1}, \ldots, w_{i50}$ for each word, and outputs $\sum_k w_{ik}x_{ik}$ when given word i. The 50×5000 weights are trainable.

```
model <- keras_model_sequential()

maxlen <- 400
embedding_dims <- 50

x_train <- imdb$train$x %>%
  pad_sequences(maxlen = maxlen)
x_test <- imdb$test$x %>%
  pad_sequences(maxlen = maxlen)

model %>%
  # Start off with an efficient embedding layer which maps
  # the vocab indices into embedding_dims dimensions
  layer_embedding(max_features, embedding_dims, input_length = maxlen) %>%
  layer_dropout(0.2)
```

We now add a one-dimensional convolution. The kernel size of 3 means that we're learning consecutive three-word groups as features for the next layer. The convolutional structure means it doesn't matter where in the text a three-word group appears.

We have 250 three-word groups being learned at this layer

As with the image analysis, the convolutional outputs are fed straight into a pooling layer, and then a dropout layer. In this case the pooling is over a whole convolutional sequence: it doesn't matter where in the review the group appears.

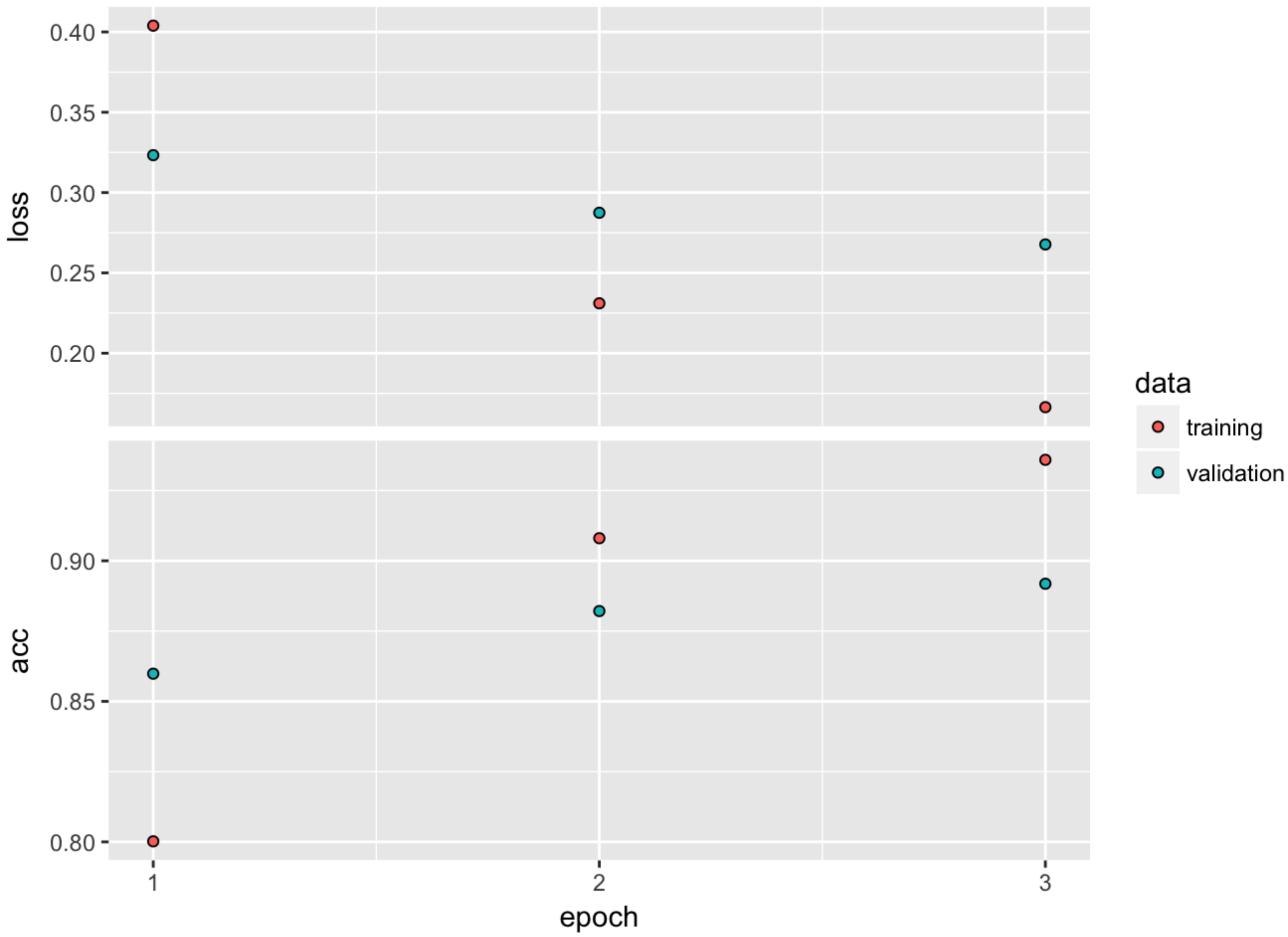
Using 3-word kernels means you can distinguish "not bad, worth seeing" from "Bad. Not worth seeing", and distinguish a movie with Alec Guiness from one where Alex Baldwin drinks a Guiness.

Finally, we've got a dense layer to do the actual fitting.

```r
filters <- 250
kernel_size <- 3

  model %>%

  layer_conv_1d(
    filters, kernel_size,
    padding = "valid", activation = "relu", strides = 1
  ) %>%
  layer_global_max_pooling_1d() %>%
  layer_dense(250) %>%
  layer_dropout(0.2) %>%
  layer_activation("relu") %>%

  # Project onto a single unit output layer, and squash it with a sigmoid

  layer_dense(1) %>%
  layer_activation("sigmoid")

# Compile model
model %>% compile(
  loss = "binary_crossentropy",
  optimizer = "adam",
  metrics = "accuracy"
)
```

```r
batch_size <- 32
epochs <- 3

history <- model %>%
  fit(
    x_train, imdb$train$y,
    batch_size = batch_size,
    epochs = epochs,
    validation_data = list(x_test, imdb$test$y)
  )
```

```r
plot(history)
```



0.40
0.35
0.30
0.25
0.20
loss
0.90
0.85
0.80
acc
1
2
3
epoch
data
training
validation