IE 345 - K “Introduction to Deep Learning: Fundamentals Concepts”

Prof. Yuzo

Build a Recurrent Neural Network.

pg. 105 - 110

In [1]:

```
import keras
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Embedding
from keras.layers import LSTM
from keras.datasets import imdb
```

```
Using TensorFlow backend.
```

In [2]:

```
#Set Hyperparameters

max_features = 20000
maxlen = 80
batch_size = 32
```

In [3]:

```
print('Loading data...')
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
print(len(x_train), 'Train Sequences')
print(len(x_test), 'Test Sequences')
```

```
Loading data...
Downloading data from https://s3.amazonaws.com/text-datasets/imdb.npz
17465344/17464789 [==============================] - 3s 0us/step
25000 Train Sequences
25000 Test Sequences
```

In [4]:

```
print('Pad sequences (samples % time)')
x_train = sequence.pad_sequences(x_train, maxlen=maxlen)
x_test = sequence.pad_sequences(x_test, maxlen=maxlen)
print('x_train shape: ', x_train.shape)
print('x_test shape: ', x_test.shape)
```

```
Pad sequences (samples % time)
x_train shape:  (25000, 80)
x_test shape:  (25000, 80)
```

In [6]:

```
# Model definition

model = Sequential()
model.add(Embedding(max_features, 128))
model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_2 (Embedding)      (None, None, 128)         2560000   
_________________________________________________________________
lstm_1 (LSTM)                (None, 128)               131584    
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129       
=================================================================
Total params: 2,691,713
Trainable params: 2,691,713
Non-trainable params: 0
_________________________________________________________________
```

In [7]:

```
# Compile model by setting learning procedure
model.compile(loss='binary_crossentropy',
             optimizer='adam',
             metrics=['accuracy'])
```

In [8]:

```
# Fit the model
model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=15,
          validation_data=(x_test, y_test))
```

```
Train on 25000 samples, validate on 25000 samples
Epoch 1/15
25000/25000 [==============================] - 87s 3ms/step - loss: 0.4632
- acc: 0.7813 - val_loss: 0.3824 - val_acc: 0.8349
Epoch 2/15
25000/25000 [==============================] - 84s 3ms/step - loss: 0.2964
- acc: 0.8799 - val_loss: 0.3818 - val_acc: 0.8359
Epoch 3/15
25000/25000 [==============================] - 87s 3ms/step - loss: 0.2142
- acc: 0.9192 - val_loss: 0.4185 - val_acc: 0.8158
Epoch 4/15
25000/25000 [==============================] - 86s 3ms/step - loss: 0.1532
- acc: 0.9430 - val_loss: 0.4410 - val_acc: 0.8290
Epoch 5/15
25000/25000 [==============================] - 88s 4ms/step - loss: 0.1147
- acc: 0.9584 - val_loss: 0.6245 - val_acc: 0.8288
Epoch 6/15
25000/25000 [==============================] - 87s 3ms/step - loss: 0.0811
- acc: 0.9728 - val_loss: 0.6212 - val_acc: 0.8222
Epoch 7/15
25000/25000 [==============================] - 84s 3ms/step - loss: 0.0558
- acc: 0.9824 - val_loss: 0.7047 - val_acc: 0.8230
Epoch 8/15
25000/25000 [==============================] - 82s 3ms/step - loss: 0.0423
- acc: 0.9858 - val_loss: 0.8372 - val_acc: 0.8135
Epoch 9/15
25000/25000 [==============================] - 86s 3ms/step - loss: 0.0433
- acc: 0.9855 - val_loss: 0.7952 - val_acc: 0.8178
Epoch 10/15
25000/25000 [==============================] - 85s 3ms/step - loss: 0.0252
- acc: 0.9925 - val_loss: 0.9534 - val_acc: 0.8205
Epoch 11/15
25000/25000 [==============================] - 81s 3ms/step - loss: 0.0210
- acc: 0.9934 - val_loss: 0.9493 - val_acc: 0.8172
Epoch 12/15
25000/25000 [==============================] - 77s 3ms/step - loss: 0.0156
- acc: 0.9949 - val_loss: 1.0864 - val_acc: 0.8138
Epoch 13/15
25000/25000 [==============================] - 76s 3ms/step - loss: 0.0174
- acc: 0.9954 - val_loss: 1.0464 - val_acc: 0.8138
Epoch 14/15
25000/25000 [==============================] - 76s 3ms/step - loss: 0.0149
- acc: 0.9954 - val_loss: 1.0173 - val_acc: 0.8120
Epoch 15/15
25000/25000 [==============================] - 77s 3ms/step - loss: 0.0112
- acc: 0.9966 - val_loss: 1.1175 - val_acc: 0.8122
```

Out[8]:

```
<keras.callbacks.History at 0x1632a6dfe48>
```

In [9]:

```
score, acc = model.evaluate(x_test, y_test,
                            batch_size=batch_size)
print('Test score: ', score)
print('Test accuracy: ', acc)
```

```
25000/25000 [==============================] - 9s 379us/step
Test score:  1.1174586215376854
Test accuracy:  0.81224
```

Pablo David Minango Negrete

$pablodavid218@gmail.com$