lab3_bashtovyi

December 1, 2021

[36]:
```
from __future__ import print_function
import numpy as np
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Embedding, Dropout, InputLayer
from keras.layers import LSTM
from keras.datasets import imdb
```

Lets define constants

[37]:
```
max_features = 10000
maxlen = 80
batch_size = 32
```

Lets import data and split it into the train and test part

[38]:
```
(training_data, training_targets), (testing_data, testing_targets) = imdb.
 ↪load_data(num_words=10000)
data = np.concatenate((training_data, testing_data), axis=0)
targets = np.concatenate((training_targets, testing_targets), axis=0)
x_test = data[:10000]
y_test = targets[:10000]
x_train = data[10000:]
y_train = targets[10000:]

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)
```

```
x_train shape: (40000, 80)
x_test shape: (10000, 80)
```

[63]:
```
x_train[0].shape
```

[63]:
```
(80,)
```

Lets build, train and test dense model.

[51]:
```
def dense():
    model = Sequential()
    model.add(InputLayer(input_shape=(80,)))
    model.add(Dense(100, activation = "relu"))
    model.add(Dropout(0.2, noise_shape=None, seed=None))
    model.add(Dense(100, activation = "relu"))
    model.add(Dense(1, activation='sigmoid'))

    model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])

    model.summary()
    return model
```

[52]:
```
model = dense()
model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=2,
          validation_data=(x_test, y_test))
score, acc = model.evaluate(x_test, y_test,
                            batch_size=batch_size)
print('Test score:', score)
print('Test accuracy:', acc)
```

```
Model: "sequential_15"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_28 (Dense)             (None, 100)               8100
_________________________________________________________________
dropout_19 (Dropout)         (None, 100)               0
_________________________________________________________________
dense_29 (Dense)             (None, 100)               10100
_________________________________________________________________
dense_30 (Dense)             (None, 1)                 101
=================================================================
Total params: 18,301
Trainable params: 18,301
Non-trainable params: 0
_________________________________________________________________
Train on 40000 samples, validate on 10000 samples
Epoch 1/2
40000/40000 [==============================] - 6s 161us/step - loss: 7.9898 -
accuracy: 0.4988 - val_loss: 7.8867 - val_accuracy: 0.5053
Epoch 2/2
40000/40000 [==============================] - 5s 136us/step - loss: 7.9923 -
```

```
accuracy: 0.4987 - val_loss: 7.8867 - val_accuracy: 0.5053
10000/10000 [==============================] - 1s 52us/step
Test score: 7.8866977249145505
Test accuracy: 0.505299985408783
```

Dense model completley failed to classify reviews on positive and negative.

Lets build, train and test LSTM model

[49]:
```
def lstm(max_features):
    model = Sequential()
    model.add(Embedding(max_features, 128))
    model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
    model.add(Dense(50, activation = "relu"))
    model.add(Dropout(0.2, noise_shape=None, seed=None))
    model.add(Dense(50, activation = "relu"))
    model.add(Dense(1, activation='sigmoid'))

    model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])

    model.summary()
    return model
```

[50]:
```
model_l = lstm(max_features)
model_l.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=2,
          validation_data=(x_test, y_test))
score, acc = model_l.evaluate(x_test, y_test,
                            batch_size=batch_size)
print('Test score:', score)
print('Test accuracy:', acc)
```

```
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_4 (Embedding)      (None, None, 128)         1280000
_________________________________________________________________
lstm_2 (LSTM)                (None, 128)               131584
_________________________________________________________________
dense_25 (Dense)             (None, 50)                6450
_________________________________________________________________
dropout_17 (Dropout)         (None, 50)                0
_________________________________________________________________
dropout_18 (Dropout)         (None, 50)                0
_________________________________________________________________
```

```
dense_26 (Dense)             (None, 50)               2550
_________________________________________________________________
dense_27 (Dense)             (None, 1)                51
=================================================================
Total params: 1,420,635
Trainable params: 1,420,635
Non-trainable params: 0
_________________________________________________________________
Train on 40000 samples, validate on 10000 samples
Epoch 1/2
40000/40000 [==============================] - 314s 8ms/step - loss: 0.4485 -
accuracy: 0.7932 - val_loss: 0.3487 - val_accuracy: 0.8451
Epoch 2/2
40000/40000 [==============================] - 313s 8ms/step - loss: 0.3146 -
accuracy: 0.8698 - val_loss: 0.3500 - val_accuracy: 0.8482
10000/10000 [==============================] - 24s 2ms/step
Test score: 0.3499836674928665
Test accuracy: 0.8482000231742859
```

Conclusion:

Even on the 2 epochs LSTM model showed 85% accuracy on the binary text classification task. With more complex architecture and longer training, accuracy could be improved.