An Introduction to Deep Learning With Python

[6.6] Sequence processing with convnets

Prof. Yuzo Iano

pgs: 226 - 228

Preparing the IMDB data

In [1]:

```
from keras.datasets import imdb
from keras.preprocessing import sequence

max_features = 10000
max_len = 500

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')

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

```
Using TensorFlow backend.

Loading data...
25000 train sequences
25000 test sequences
Pad sequences (samples x time)
x_train shape: (25000, 500)
x_test shape: (25000, 500)
```

Training and evaluating a simple 1D convnet on the IMDB data

In [2]:

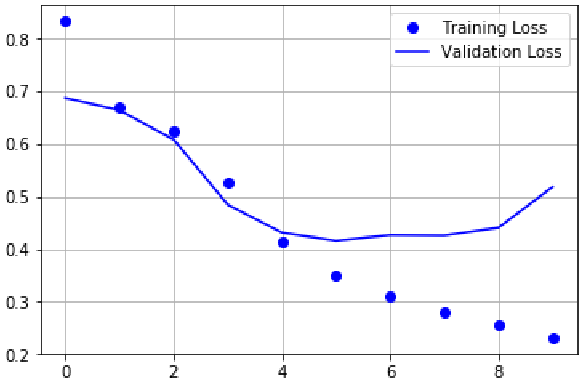
```
from keras.models import Sequential
from keras.layers import Embedding, Conv1D, MaxPooling1D, GlobalMaxPooling1D, Dense
from keras.optimizers import RMSprop

model = Sequential()
model.add(Embedding(max_features, 128, input_length=max_len))
model.add(Conv1D(32, 7, activation='relu'))
model.add(MaxPooling1D(5))
model.add(Conv1D(32, 7, activation='relu'))
model.add(GlobalMaxPooling1D())
model.add(Dense(1))
model.summary()

model.compile(optimizer=RMSprop(lr=1e-4),
              loss='binary_crossentropy',
              metrics=['acc'])

history = model.fit(x_train, y_train,
                    epochs=10,
                    batch_size=128,
                    validation_split=0.2)
```

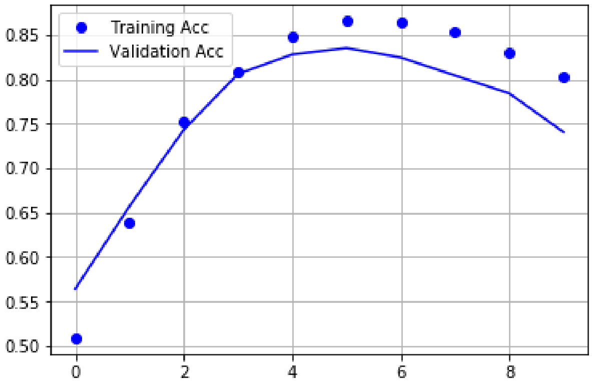
```
WARNING:tensorflow:From C:\Users\pablo\AppData\Roaming\Python\Python36\site-packages\tensorflow\py
thon\framework\op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is depr
ecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_1 (Embedding)      (None, 500, 128)          1280000   
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 494, 32)           28704     
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 98, 32)            0         
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 92, 32)            7200      
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 32)                0         
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 33        
=================================================================
Total params: 1,315,937
Trainable params: 1,315,937
Non-trainable params: 0
_________________________________________________________________
WARNING:tensorflow:From C:\Users\pablo\AppData\Roaming\Python\Python36\site-packages\tensorflow\py
thon\ops\math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will b
e removed in a future version.
Instructions for updating:
Use tf.cast instead.
WARNING:tensorflow:From C:\Users\pablo\AppData\Roaming\Python\Python36\site-packages\tensorflow\py
thon\ops\math_grad.py:102: div (from tensorflow.python.ops.math_ops) is deprecated and will be rem
oved in a future version.
Instructions for updating:
Deprecated in favor of operator or tf.math.divide.
Train on 20000 samples, validate on 5000 samples
Epoch 1/10
20000/20000 [==============================] - 92s 5ms/step - loss: 0.8337 - acc: 0.5089 - val_los
s: 0.6875 - val_acc: 0.5642
Epoch 2/10
20000/20000 [==============================] - 90s 4ms/step - loss: 0.6700 - acc: 0.6389 - val_los
s: 0.6643 - val_acc: 0.6574
Epoch 3/10
20000/20000 [==============================] - 86s 4ms/step - loss: 0.6237 - acc: 0.7533 - val_los
s: 0.6084 - val_acc: 0.7434
Epoch 4/10
20000/20000 [==============================] - 118s 6ms/step - loss: 0.5261 - acc: 0.8077 - val_lo
ss: 0.4845 - val_acc: 0.8064
Epoch 5/10
20000/20000 [==============================] - 119s 6ms/step - loss: 0.4127 - acc: 0.8480 - val_lo
ss: 0.4316 - val_acc: 0.8282
Epoch 6/10
20000/20000 [==============================] - 117s 6ms/step - loss: 0.3488 - acc: 0.8659 - val_lo
ss: 0.4160 - val_acc: 0.8354
Epoch 7/10
20000/20000 [==============================] - 118s 6ms/step - loss: 0.3103 - acc: 0.8637 - val_lo
ss: 0.4269 - val_acc: 0.8248
Epoch 8/10
20000/20000 [==============================] - 118s 6ms/step - loss: 0.2805 - acc: 0.8538 - val_lo
ss: 0.4262 - val_acc: 0.8046
Epoch 9/10
20000/20000 [==============================] - 118s 6ms/step - loss: 0.2551 - acc: 0.8296 - val_lo
ss: 0.4410 - val_acc: 0.7844
Epoch 10/10
20000/20000 [==============================] - 119s 6ms/step - loss: 0.2315 - acc: 0.8036 - val_lo
ss: 0.5183 - val_acc: 0.7408
```

Plotting the results

In [4]:

```
import matplotlib.pyplot as plt

loss = history.history['loss']
val_loss = history.history['val_loss']

acc = history.history['acc']
val_acc = history.history['val_acc']

epochs = range(len(loss))

plt.plot(epochs, loss, 'bo', label='Training Loss')
plt.plot(epochs, val_loss, 'b', label='Validation Loss')
plt.legend()
plt.grid()

plt.figure()
plt.plot(epochs, acc, 'bo', label='Training Acc')
plt.plot(epochs, val_acc, 'b', label='Validation Acc')
plt.legend()
plt.grid()
plt.show()
```





Pablo Minango

- pablodavid218@gmail.com