DEEP LEARNING

BY M.J.PASSLAR

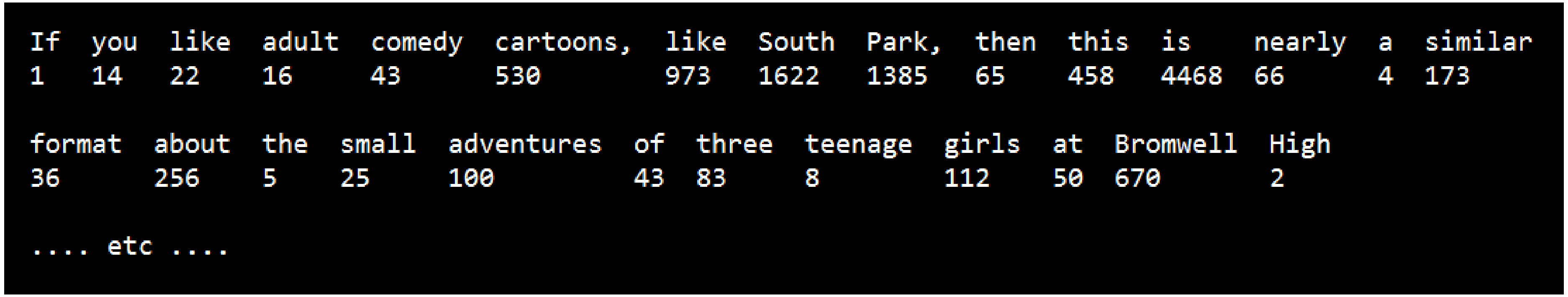
COMPUTERONIC - TEHRAN - IRAN

2023

CHAPTER 4 : IMDB

What's IMDB problem ?

- It's a set of 25,000 training text, plus 25,000 test text
- in each data set(traind & test), 50% are positive and 50% are negetive
- It's a Binary Classification problem type
- So last layer activation -> sigmoid & loss function -> binary_crossentropy



```
If  you  like  adult  comedy  cartoons,  like  South  Park,  then  this  is    nearly  a  similar
1   14   22    16     43      530        973   1622   1385   65    458   4468  66      4  173

format  about  the  small  adventures  of  three  teenage  girls  at  Bromwell  High
36      256    5    25     100         43  83     8        112    50  670       2

.... etc ....
```

Loading the IMDB dataset in Keras

```
#download dataset
from keras.datasets import imdb

(train_data,train_lable),(test_data,test_lable) = imdb.load_data(num_words=10000)
print(len(train_data))
print(len(train_lable))

print(len(test_data))
print(len(test_lable))
```

Train data
Count : 25000
Size : variable

Train data
Count : 25000
Size : variable

Let's prepare data to feed

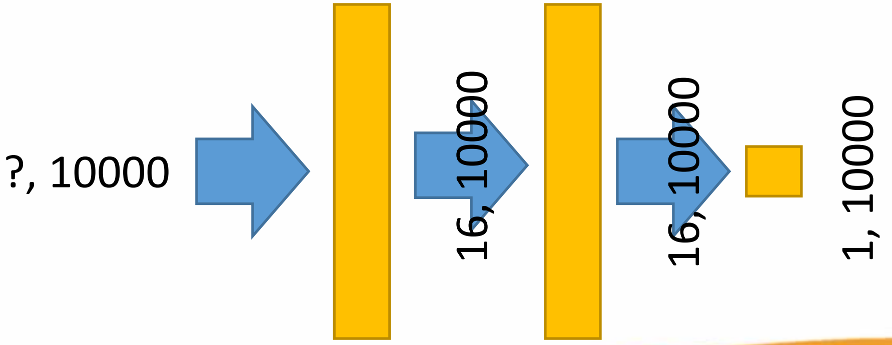
```
#preparing data
import numpy as np

def vectorize_sequences(sequences, dimension=10000):
  results = np.zeros((len(sequences), dimension))
  for i, sequence in enumerate(sequences):
    results[i, sequence] = 1
  return results
x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)

y_train = np.asarray(train_lable).astype('float32')
y_test = np.asarray(test_lable).astype('float32')
```

- Vectorize data
- Convert data to float32
- Arrange data to Tensor

What's Tensor looks like ?

Create and compile your model

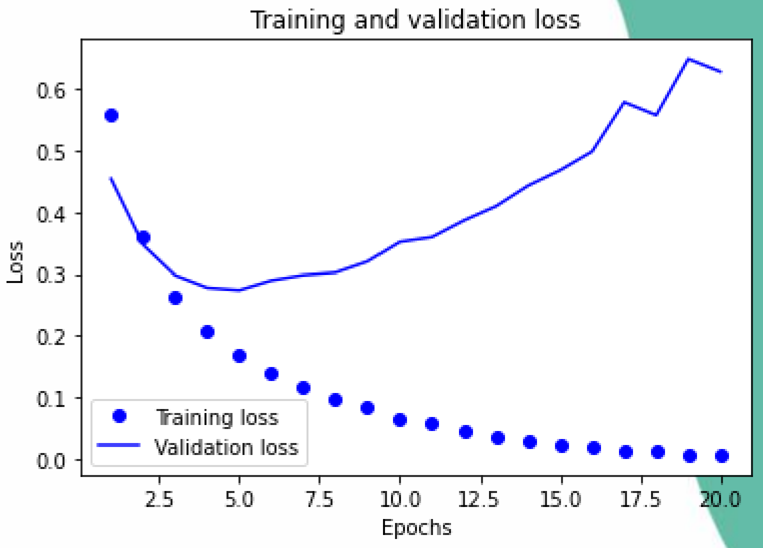
```
#creat and compile yout model
from keras import models
from keras import layers

model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

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

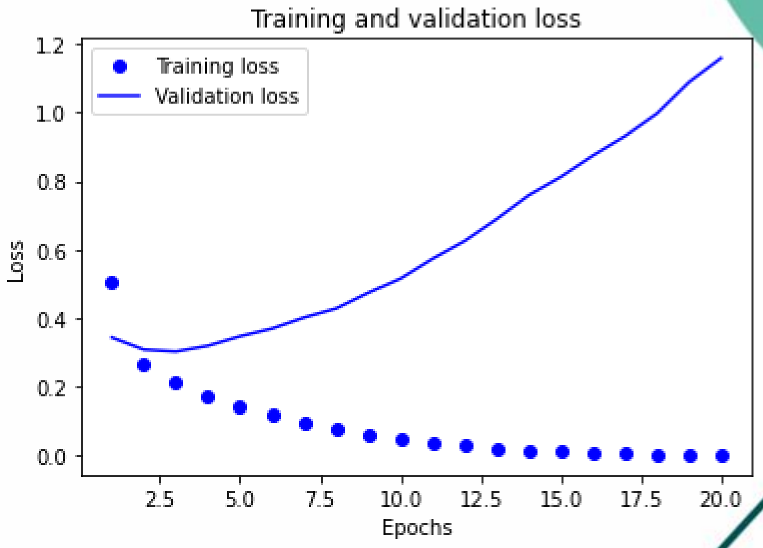


Train and Evaluate your model

```
x_val = x_train[:10000]
partial_x_train = x_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))

test_loss, test_acc = model.evaluate(x_test, y_test)
```

Why ?





Train and Evaluate your model

```
x_val = x_train[:10000]
partial_x_train = x_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=4,
                    batch_size=512,
                    validation_data=(x_val, y_val))

test_loss, test_acc = model.evaluate(x_test, y_test)
```

Plot loss and accuracy

```
#plot loss and accuracy
history_dict = history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
epochs = range(1, 21)

import matplotlib.pyplot as plt
plt.plot(epochs, loss_values, 'bo', label='Training loss')
plt.plot(epochs, val_loss_values, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

Now it’s your turn ...

Parameter	Smaller	Bigger	Result
Meddle layer size			
Epochs number			
Batch size			
Second layer activation function			
third layer activation function			
Fit model with partial data			
Fit model with test data			