DEEP LEARNING

BY M.J.PASSLAR

COMPUTERONIC – TEHRAN – IRAN

2023

CHAPTER 5 : REUTERS

What's Reuters problem ?

- It's a set of 8,982 training news, plus 2,246 test news
- Set of short news and their topics (46 subject) build in 1986
- It's a muilticlass ,singlelabel Classification problem type
- So last layer activation -> softmax & loss function -> categorical_crossentropy

Loading the Reuters dataset in Keras

```
#load data set
from keras.datasets import reuters
(train_data,train_labels),(test_data,test_labels) =
                                        reuters.load_data(num_words = 10000)


print(len(train_data))
print(len(train_labels))

print(len(test_data))
print(len(test_labels))
```

Train data
Count : 8982
Size : variable

Train data
Count : 2246
Size : variable

Let's prepare data to feed

```
#preparing the data
import numpy as np
def vectorze_sequences(sequences,dimention=10000):
    result = np.zeros((len(sequences),dimention))
    for i,sequence in enumerate(sequences):
        result[i,sequence] = 1
    return result
x_train = vectorze_sequences(train_data)
x_test = vectorze_sequences(test_data)
```

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

What's Tensor looks like ?

Let's prepare labels

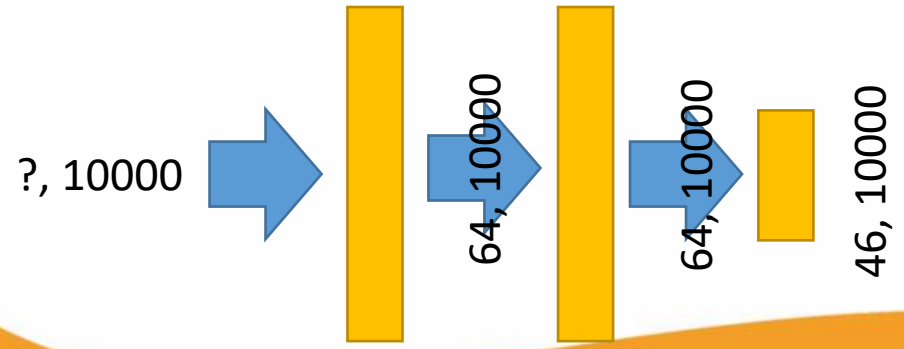
```
#preparing the labels
def to_one_hot(labels,dimention=46):
    results = np.zeros((len(labels),dimention))
    for i, labels in enumerate(labels):
      results[i,labels]=1
    return results
one_hot_train_labels = to_one_hot(train_labels)
one_hot_test_labels = to_one_hot(test_labels)
```

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

What's Tensor looks like ?

Create and compile your model

```
#model definitation
from keras import models
from keras import layers
model = models.Sequential()
model.add(layers.Dense(64,activation = 'relu' , input_shape =(10000,)))
model.add(layers.Dense(64,activation = 'relu'))
model.add(layers.Dense(46,activation = 'softmax'))

#compiling the model
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

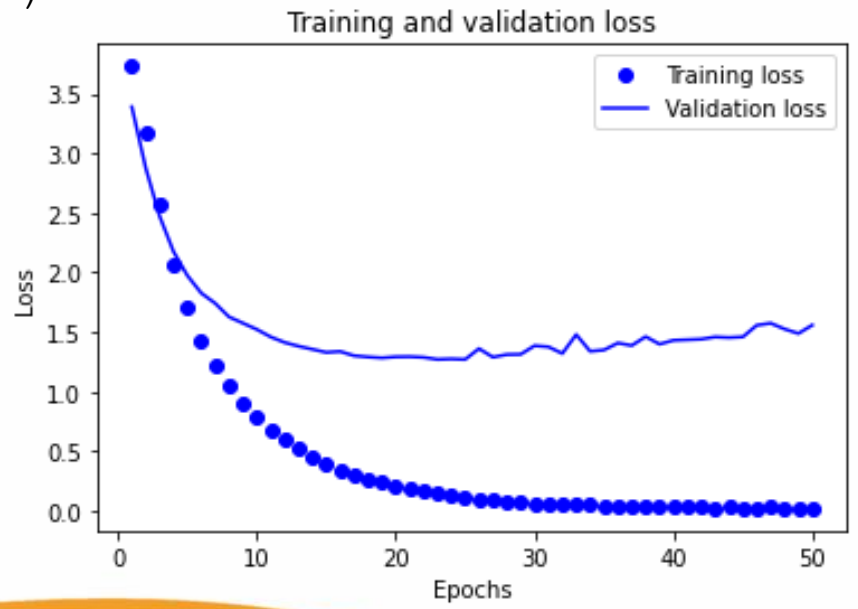


Train and Evaluate your model

```
#setting aside a validation set
x_val = x_train[:1000]
partial_x_train = x_train[1000:]

y_val = one_hot_train_labels[:1000]
partial_y_train = one_hot_train_labels[1000:]

#Training the model
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, one_hot_test_labels)
```

Plot loss and accuracy

```
#plot loss and accuracy
history_dict = history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

import matplotlib.pyplot as plt
plt.plot( loss_values, 'bo', label='Training loss')
plt.plot(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			