**Assignment:3**

**DEEP LEARNING**

**AO PENG**

### Load data and preprocess data

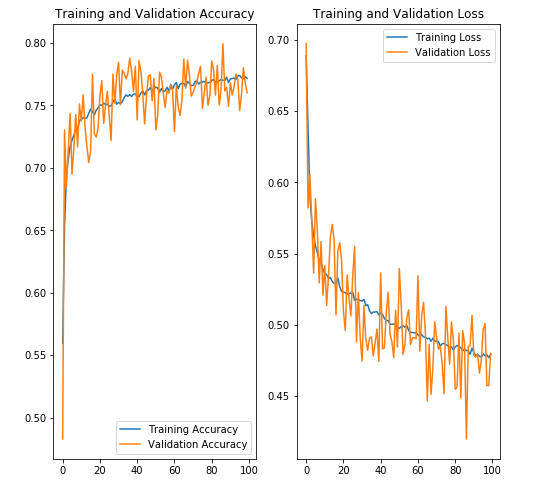
In this part, I loaded comments from files and created training and testing data sets. In addition, creating the label for data set by file names. I counted the number of words in the comments and the max num is 2470 as a standard.

For preprocessing data part, I used Tokenizer to turn each text into either a sequence of integers for training. I focus on the top 500 words with the highest frequency to reduce cost time to train the model. It is hard to train the model with high length of data. I use 1/3 part of data set as the validation data.

### Vanilla RNN

Vanilla RNN is the basic rnn. The definition of it is,

I implemented it by keras with different dimension and tune the learning rates. The big issue in training model is gradient explosion as in training processing, there is a fluctuation in loss, although it keeps decreasing(*Figure. 1*).



**Figure. 1 gradient explosion**

Here is the table of 5 different dimensions of model.

**Tabel.1 Comparison of 5 dimensions**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 20(1e-3) | 50(1e-4) | 100(1e-4) | 200(1e-4) | 500 |
| accuracy | 0.8146 | 0.8225 | 0.7882 | 0.7908 | 0.6308 |
| loss | 0.4213 | 0.4038 | 0.4583 | 0.4468 | 0.6541 |

From the training processing, I found that the low learning rate is good for preventing the gradient explosion while it is time consuming of training with low learning rate. So there should be a trade off between training speed and learning rate. When we choose the proper learning rate, the performance of the model will be increased and the training speed is relatively good(*Table.2*).

**Tabel.2 Comparison of 3 learning rate in 20-dimension**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1e-4 | 1e-2 | 1e-3 |
| accuracy | 0.8047 | 0.6807 | 0.8146 |
| loss | 0.4365 | 0.5665 | 0.4213 |

In addition, the state dimension also plays an important role in vanilla rnn. The higher dimension would result higher loss and lower accuracy as it is hard to fit the high dimension model. As we can see the results(*Tabel.1*), when the dimension higher than 50, the accuracy would be decreased even if it is trained with best learning rate.

### LSTM

LSTM is based on vanilla RNN which has an increase that it reduces the problem of gradient explosion to a certain extent as it introduces the ‘gate’. The definition of LSTM is,

The belongs to forget gate. is an input gate's activation vector. means 1 output gate vector.

LSTM allows selective memorization of the past information and selective loading of new information which depends on forget gate and input gate. The output gate controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit. The main reason why LSTM can prevent gradient explosion is because of forget gate. For backpropagation through time in LSTM, it is the activation of the forget gate that prevents the gradient vanish. Gradient vanish is caused by long series of multiplications which results in the learning process is unstable.

I implemented LSTM by Keras which is based on taking the final state of the chain as the extracted text feature.

For dimension of state, the result is the same as that of vanilla RNN. High dimension model is hard to train and the model is not good enough with low dimension. For learning rate, 0.001(1e-3) is good for almost all dimensions in LSTM. Only the model with proper learning rate can get best result(*Table.3*).

**Tabel.3 Comparison of 5 dimensions**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 20(le-2) | 50(1e-3) | 100(1e-3) | 200(1e-3) | 500(1e-3) |
| accuracy | 0.8488 | 0.8029 | 0.8501 | 0.8154 | 0.8017 |
| loss | 0.3430 | 0.4365 | 0.3384 | 0.4217 | 0.4528 |



**Figure.2 LSTM with 100-dimension**

I also build a LSTM model which is based on mean pooling for comparison. This model named model2 is compared to the model in 100-dimension LSTM model named model1 based on final state as the output. Both of them have the same learning rate which is 0.001. Here is the results.

**Tabel.4 Comparison of 2 LSTM models**

|  |  |  |
| --- | --- | --- |
|  | model1 | model2 |
| accuracy | 0.8501 | 0.8544 |
| loss | 0.3384 | 0.3362 |

### Conclusion

1. LSTM can efficiently prevent the gradient explosion and vanishing because of the ‘gates’
2. Learning rate plays an important role in the training process. The proper learning rate will increase the performance of model tremendously.
3. Vanilla RNN can also get the model with good enough performance if we set a proper learning rate, while its structure is not good enough as it cannot deal well with gradient vanishing and gradient explosion. We only can do is to reduce the learning rate which will lead to a decrease in time performance.
4. The 2 different models of LSTM(*Table.4*) have different performance. The model1 based on mean pooling takes long time in training process with higher accuracy and the model2 based on final gate also get good results but it is worse than the results of the former one.