**Project 2. Sequence Classification in Brain functional networks**

Xinyue Hu tul07341@temple.edu

## Stage 1: basic LSTM

In this first stage, I applied traditional lstm to the dataset. As we know, for each person, there are 94 tracks of brain activities at the same time. Then for each time point, the input will have 94 attributes. And there are 240 time points, which means the lstm need 240 steps to finish training process for one sample.

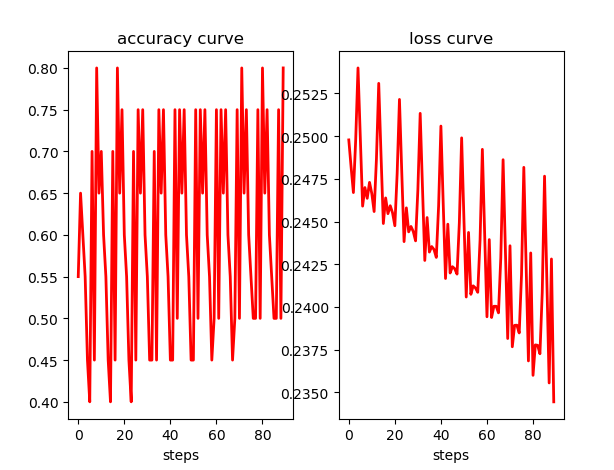
The hidden dimension number I set to the lstm is 64. The learning rate is 0.0001. number of epochs is 10. Batch size is 20.

And I have changed the target into one-hot encoding. Because the target dimension is 2, I connect a fully connected layer whose output layer dimension is 2, to the last hidden vector of the LSTM. Then add a relu function and a softmax function. I used MSE function to calculate the loss.

I used 5 fold cross validation in my whole project. As you can see from the table below. The final accuracy is 0.515.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Fold | 1 | 2 | 3 | 4 | 5 | average |
| accuracy | 0.575 | 0.600 | 0.450 | 0.450 | 0.500 | 0.515 |

The figure below is one of the training curve of accuracy and loss.



We can see that the accuracy is increasing with strong vibration. And it doesn’t increase much. The loss is keep decreasing.

## Stage 2: Attention

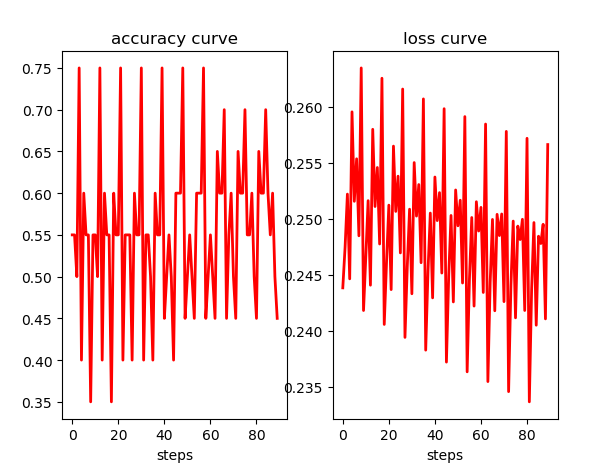
In this, stage I add attention to the original lstm. At first, I initial 3 weight matrixes to calculate query, key and value. They’re obtained by multiplying the input embedding and the corresponding weight matrix. Then multiply query and key’s transpose, and divided by the square root of the hidden dimension. Then followed by a softmax function and multiplying with value. Finally, get the output of the attention layer. The difference between query and others is that, the query each time is obtained from one single person, and then multiplying the queries from others.

So, the weight matrixes I initial is three 64 by 32, where 64 is the hidden layer dimension, 32 is the query dimension which is obtained after a lot of times of tune.

The final accuracy after adding attention is: 0.55

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Fold | 1 | 2 | 3 | 4 | 5 | average |
| accuracy | 0.500 | 0.500 | 0.550 | 0.550 | 0.650 | 0.550 |

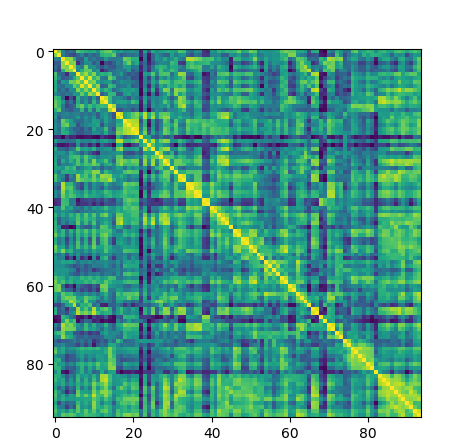
The figure below is one of the training curve of accuracy and loss.



We can see that the accuracy is improved a little bit compared with the stage 1 LSTM. But not too much. Then I entered into stage3.

## Stage3: correlation matrix

In this stage, I add a sliding window into the input sequences. And then calculate the Pearson correlation matrix of this window. Then I get a 94 by 94 correlation matrix. With this matrix, I can get the information of the correlation between each track of the brain activity and other tracks. The graph below is one of the Pearson correlation matrix



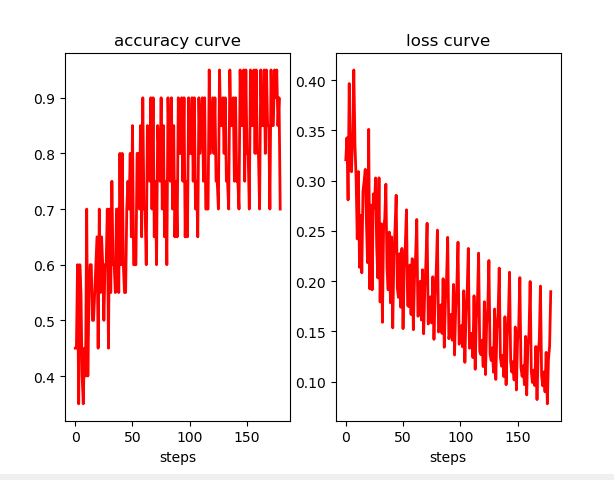
Then I flatten it into a one-dimensional vector and input it into my network.

The window size I set finally is 64. I tried lots of different size and I found 64 is the best size. In the whole sequence, Each window is overlapped with others a little bit, which means the moving distance each time is not equal to the window size. The moving distance I set is 60, which means there are 4 points are overlapped with each other.

The final accuracy after adding correlation is: 0.62

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Fold | 1 | 2 | 3 | 4 | 5 | average |
| accuracy | 0.675 | 0.675 | 0.525 | 0.600 | 0.625 | 0.62 |

The figure below is one of the training curve of accuracy and loss.



.we can see that both accuracy and loss are improved a lot.

## Predicting results:

My prediction results for the test set is:

([1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1])