CAP5619-program2

Hansong Zhou, Yu Zheng

1 Introduction to Programming files

This section is to introduce the coding files.

1. Environment Requirements:
   (a) tensorflow-gpu 2.5.0rc2
   (b) numpy 1.22.1
   (c) matplotlib 3.4.1
   (d) keras 2.7.0
   (e) cuDNN v8.3.2 for CUDA 11.5
   (f) python 3.9.4
2. `text_lstm.py`
   This file contains the following sections:
   (a) **Load dataset**: Due to the computation limitation of our computer, we use 4000 of the total dataset.So we have 3200 training dataset, 800 validation dataset.
   (b) **Prepare dataset**: structure dataset for the LSTM network input and output
   (c) **Model training**: For Task 2 part 1
   (d) **Analyze long-term dependencies**: For Task 2 part 1
   (e) **3-gram language models**: For Task 3 (3)
3. `seed_protein_gen.py`
   This file is for Task 3 (1) and (2)

2 Task 1 - Recurrent Neural Network Design

The recurrent neural network is composed of two layers, a LSTM layer for input and a Fully-connected layer for output. Before input the raw string data into the network, we truncate the string to size of 100 and do one-hot encoding. If the size of string is less than target length, we padded the string with zeros.

In addition to the given 20 one letter codes, because we pad the string with 0, the total number of feature is 21. In our work, we adapt the training strategy that stacking the LSTM units and training a model on multiple time steps simultaneously. Specifically, for a data with 100 timesteps, we use the first 99 timesteps to predict the data of last 99 timesteps for training the parameters of neural network. Hence, the size of input of our RNN is $(batchsize, 99, 21)$. As for the LSTM layer, we use 512 units which has the best performance during our test. If we increase the number of layer to 1024, the model is too complex and the learning speed is low. If we decrease it to 128, the accuracy is unsatisfactory, only 61%. We use softmax activation function in dense layer to predict the right one letter codes. The structure of our network is given as follow:

layers	**neurons**	**parameters**	**activation function**
LSTM	512 units	1093632	sigmoid, tanh
Dense	21	10773	softmax

Table 1: RNN with LSTM units

3 Task II – Language Models for Protein Sequences and Evaluation

We initialize the network weights as follows:

1. kernel initializer: Glorot uniform,
2. recurrent initializer: orthogonal
3. bias initializer: zeros

The hyperparameters:

1. batch size: 32
2. epochs: 300

The optimizer we use is Root Mean Squared Propagation (RMSprop), the learning rate is 0.001, the ρ value is 0.9. The training loss, training accuracy, validation loss, and validation accuracy during training epochs are shwon in Figure **??**

Then we find out the longest dependencies the network is capable of capturing as follows. we use one sequence example as testing sequence which contains 100 acids. We change each acids character as a random character and pass it through our network, then calculate the 2-norm of the difference between the outputted probability with the true labels. We perform this operation for all first 99 acids in the testing sequence. In order to make the result general, we repeat the above operations in 100 times Monte Carlos experiments. The final result shows the longest dependencies our network could capture are $0, 1, 2, 3$ (the first 4 acids in the sequence).

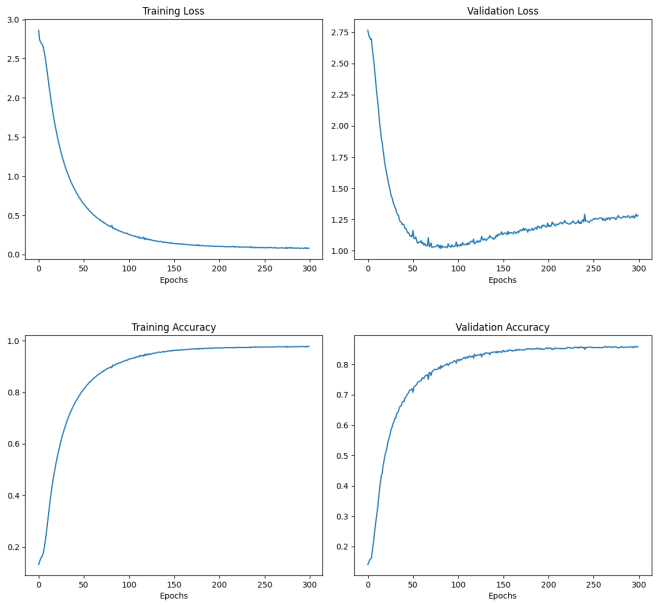

Figure 1: The training loss, training accuracy 97.9%, validation loss, and validation accuracy 86.1% during training epochs

4 Task III – Sequence Generation Techniques

(1) Inspired by question 2, we use a random seed to generate the protein sequence with our trained LSTM neural network. We set the length of random seed as 5. In this way, we get following five new protein sequences:

1. LPQRGQTMFPFTGDHGDHVISATKSADPIAALDALVGLKLDSFIVKHHHNGN VNGPTGDSHLQQQFNEAKITNSMEHYAAESLNAGWQAVKLSNRPDAEF
2. RPTLEEKIADKMRTQEIRKGAIFAVESKTQNDAEILKAVGHRLTDENQNFTG VEKMLRQLARKE
3. MKGEQMQVIKVLLLTPQDQDYLVAYHREFRVEMDHHVYNAIENTGACAAGYV GRPAKGLTKDETGQAYFLQNAKGKRWEAIDKYSPANRECALRKDELGA
4. YNECEIHDGSRVDRTSSYVSGPFTWYQDEIHYPDLTGSRLAEPVDIALCGTN PDSILVLSIAAQDKTKPADANTPLSFVAVDGKQTQLVLIRSAGTMEVE
5. GSQFGLPASQILLSISSTTARFADRRMHHLPAARYTARCSILMREAENHAGW QEKVLNSGKDWVAGTVVCLDAGSDPSFLNPAVNNSAGLPSTRLDLRAM

After each new sequence is generated, we compare the new sequence with the sequences existed in the dataset. If a same sequence is found in the dataset, we will re-generate until we find a new protein sequence.

(2)

k	Number of sequences with the maximal matches																		
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19+**
1	660	59	13	1	5	0	2	1	1	0	0	1	3	0	0	0	0	0	0	54
2	453	45	4	4	1	3	2	3	1	2	2	3	1	1	0	0	0	1	2	272
3	210	16	6	5	12	7	2	4	5	3	6	1	4	0	1	2	4	2	2	508
4	124	19	6	12	8	3	6	5	3	6	1	5	1	1	3	4	2	2	0	589
5	113	14	12	8	3	6	5	3	6	1	5	2	1	3	4	2	2	0	2	608
6	117	17	8	3	6	5	5	6	1	5	2	1	3	4	2	2	0	2	2	609
7	123	20	3	6	5	5	6	1	5	2	2	3	4	2	2	0	2	2	4	603
8	123	14	9	6	5	6	1	6	2	2	3	4	2	2	1	2	2	4	1	605
9	120	16	6	6	6	1	5	3	2	3	4	2	2	0	2	2	4	1	1	614
10	124	13	7	7	1	5	4	2	3	5	2	2	0	2	2	4	1	1	6	609

Overall, due to the great performance of our LSTM network, nearly 80% predicted sequence can have more than 19+ amino acids matched. With more head letters fixed, more followed letters can be predicted correctly. Such trend is significant from $k = 1$ to $k = 4$. After the fixed number increased to 5, the improvement is slight according to the table.

(3) We constructed 2 3-gram sequence search models based on network and the training dataset respectively. There are 8000 possible types of 3-gram acids sequences, and we generate 313600 3-gram acids sequences from network and training dataset. The 2-norm distance between the probability vector is shown in Figure 2.

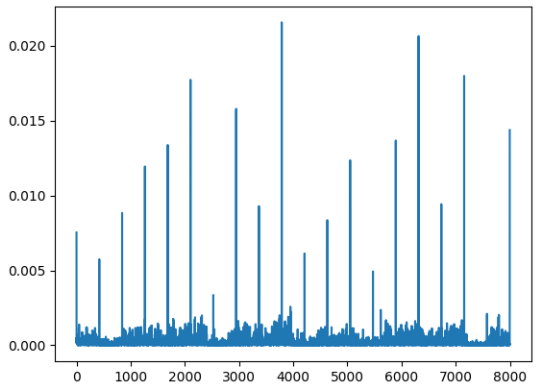

Figure 2: The 2-norm difference between the probability vector from the 2 3-gram models based the network and training dataset

The most different 3-gram sequences include VLL, HHH, QQQ, CCC, MMM, NNN, AAA, DDD, TTT, KKK, EEE, PPP, RRR, YYY, FFF, III, GGG, VVV, SSS, LLL.

The most closed 3-gram sequences include FHA,MWA, ADA, AEA, AFA, AGA, AHA, AIA, AKA, ALA, AMA, ANA, APA, AQA, ARA, ASA, ATA, AVA, AWA, AYA.

The following figure 3 shows the probabilities for dataset (left) and network (right) and the corresponding probabilities for maximal 20 differences (red) and minimal 20 differences (blue).

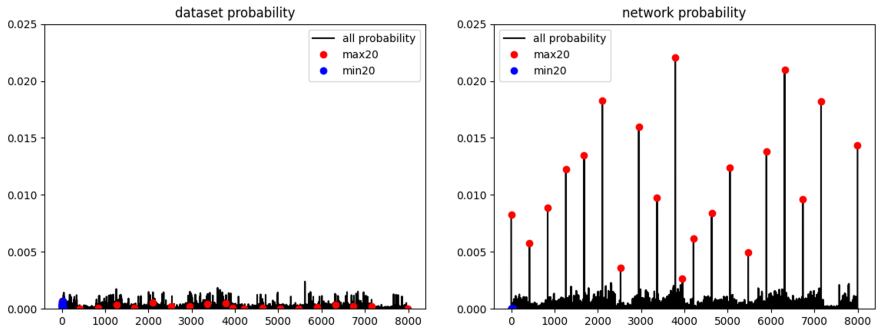

Figure 3: The probability vector from the 2 3-gram models based the network and training dataset (The left is the probability from dataset, the right is the probability from network, red corresponds to the maximal 20 differences of probability, blue corresponds to the minimal 20 differences of probability.

It could be seen that the probabilities of different 3-gram sequence are averaged, the maximal difference happens when network-based 3-gram model estimated the sequences too frequently. Those 3-gram sequences network estimates too frequently are those repeated 3-gram sequence, such as AAA. This means the network's generalization capability is not good enough and could be improved further. The most closed probabilities happen more at sequence "A...A", this might because those 3-gram sequences are easier to be leaned due to their special structure.