Programming Project II – Sequence Modeling Using Recurrent Neural Networks
CAP 5619, Deep & Reinforcement Learning (Spring 2020), Department of Computer Science, Florida State University

---

Points: 100
Maximum Team Size: 2
Due: Beginning of the class (11:00am) on Thursday, April 16th, 2020

Submission: You need to submit electronically via Canvas by uploading a) a pdf file (named "**lab2-Lastnames.pdf**") as your report (including analysis and experimental results), and b) the program(s) you have created (named as "**lab2-prog-Lastnames.???**"); if there are multiple program files, please zip them as a single archive. Here replace "Lastnames" using your last names of the group members in the file names. Only one submission is required for each group.

The main purpose of this assignment is to apply recurrent neural networks to model protein sequences. Note that the focus of this programming project is to gain a deeper understanding of different recurrent neural networks and having good recognition performance itself is not sufficient.

For this assignment, we use a protein sequence dataset, consisting of 440,115 protein sequences (with a total of 109,364,666 amino acids), which can be found at http://www.cs.fsu.edu/~liux/courses/deepRL/assignments/pdb_seqres.txt. A brief description about the dataset is available at http://www.cs.fsu.edu/~liux/courses/deepRL/assignments/pdb_inform.txt. You need to use the 5th, 10th, … and 440,115th sequences as the validation set, and the rest as the training set Each protein sequence is a variable length array of 20 characters (representing the 20 different types of amino acids). If you do not have the computational resources to process the entire dataset, you can use 10% of the sequences with the same percentage for validation. In addition, to simplify the models, you can set a maximum length of 100 (or a different number) and truncate the ones that are longer and pad the ones that are shorter (using zeros).

Task I – Recurrent Neural Network Design
In the deep learning framework you have established, design one recurrent neural network. Here each character (input) will be encoded as a 21-dimensional vector using one-hot encoding (20 different amino acids and one special end of sequence symbol). You can use the LSTM given by Equations (10.40) to (10.44) (in the textbook) or any other valid variant of LSTM; in the latter case, you need to describe your network clearly in the report. You can also use the GRU (gated recurrent unit) given by Equations (10.45) to (10.47) (in the textbook) or any other valid variant of GRU; similarly, in the latter case, you need to describe your network clearly in the report.
In addition, in your report, you need to provide explanations of your design choices and describe your recurrent neural network clearly. In particular, you need to describe all the hyperparameters your design has and the loss function you use. When you use a network available already in your framework, make sure that you will be able to describe it correctly.

Task II – Language Models for Protein Sequences and Evaluation
First you need to explain how you initialize the weights and choose the hyperparameters and the optimizer for your network. Then you need to train a recurrent neural network using the given dataset. You need to plot the loss, accuracy, and perplexity on both the training set and validation set with respect to the epochs.
One of the advantages of recurrent neural networks using LSTM or other gated cells is that they can capture the long-term dependencies in the data well. Here you need to find out the longest dependencies your network is capable of capturing experimentally. You can design/use any reasonable method. One way is to change one or more letters in sequences and see when the change(s) will have no effect. You need to describe your method clearly and document your results.

Task III – Sequence Generation Techniques

For this task, first you need to use your trained RNN to complete protein sequences and generate new protein sequences. You need to demonstrate this by giving five examples. Then you need to perform the following experiments.

(1) For each of the sequences in the validation set, provide the first k (varies from 1, 2, … , 10) letters as the seed input and predict a sequence, and then find out the maximal letters that are predicted correctly. Use the information to complete a table similar to the following:

K	Number of sequences with the given maximal matches																			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19+
1																				
2																				
3																				
4																				
5																				
6																				
7																				
8																				
9																				
10																				

As an example, for k=5, you fix the first five amino acids in each of the sequences in the validation set, generate a sequence as long as the original sequence, and then compare the two sequences, and find the longest matches right after the initial ones (i.e., you need to exclude the first five in this case). The counts for the cases are the row for case 5. The last column includes all the matches at least 19 (19, 20, and so on).

(2) Use your network to generate a 4-gram language model, i.e., estimating the probability for all the possible protein sequences of length 4. Then you also estimate a 4-gram model from the training set by counting the 4-amino acids; if a combination does not occur in the training set, you can use 0 or any back-off model. Compute the distance between the two (as long vectors) in L_2-norm; identify the 20 most closely matched entries and 20 most different entries. By analyzing your network and the dataset, explain why these particular entries are so similar or so different.

Extra Credit Options

(1) **Amino acid embedding**. Instead of using the one-hot encoding of amino acids, you need to train a dense vector representation for the amino acids jointly with language modeling training. You need to illustrate your resulting embeddings and demonstrate the advantages with improved performance.

(2) **Sequence DNA**. Using an encoder-decoder sequence-to-sequence architecture, you need to be able to summarize a protein sequence using a fixed context vector (as the signature (or DNA) of the protein) and be able to generate a sequence from the context vector that is close to the original sequence. You need to provide working examples and measure the effectiveness of your sequence DNAs.

(3) **Architecture optimization**. As demonstrated in "An Empirical Exploration of Recurrent Network Architectures" (available from http://proceedings.mlr.press/v37/jozefowicz15.pdf), a customized recurrent neural network can perform (significantly) better than LSTM or GRU. Design a search and optimization strategy over different recurrent neural network architectures (see the paper for details about the search space). You need to show improved performance in order to receive full credit for this option.

(4) **Integration of LSTM (or GRU) and n-gram models**. LSTM models generalize better as the values vary smoothly with changes in the input while n-gram models just memorize the (smoothed) statistics in the data and can be matched very efficiently via hashing. Design an n-gram model that can take the advantage of the performance of an LSTM (or GRU) based recurrent neural network while retaining its retrieval efficiency (as the speed is crucial for many practical applications (including speed recognition)). You need to explain your method clearly and demonstrate improved performance.

Grading

- **Report and analysis** – 30 points
  - Note that the focus is on understanding and you have to provide meaningful/insightful analysis for what you expect from your experiments before doing them and explain what you have observed in your experiments as explained in the tasks.
- **Correct implementation and experimental results** – 70 points
  - **Task I – 10 points**
    - **Correct recurrent neural network architecture** – 10 points
  - **Task II – 30 points**
    - **Hyperparameter selection, weight initialization, and effectiveness of training –** 15 points
    - **Long-term dependency estimation** – 15 points
  - **Task III – 30 points**
    - **Sequence generation –** 10 points
    - **Sequence generation evaluation (the table) –** 10 points
    - **4-gram model estimation, comparison, and analysis –** 10 points
- **Amino acid embedding –** 10 points
- **Sequence DNA –** 10 points
- **Architecture optimization** – 10 points
- **LSTM and N-gram integration** – 10 points

Additional information:

By viewing vector representations for amino acids as parameters, you can optimize the vector representations while training a recurrent neural network. For example, see the paper " A Neural Probabilistic Language Model" available from http://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf.

You can think the sequence DNA as a machine translation problem except that both the input and output are in the same language. In this view, you may find papers on neural machine translation techniques helpful. Here is an example of using the encoder-decoder architecture for translation: https://github.com/keras-team/keras/blob/master/examples/lstm_seq2seq.py.

You can find a language model using LSTM at https://github.com/shivam5992/language-modelling (in Keras).