Assignment 3

CS5480: Deep Learning
IIT-Hyderabad
Jan-Apr 2018

Max Marks: 35
Due: 25th Apr 2018 11:59 pm

This homework is intended to cover the following topics:

- Recurrent Neural Networks

Instructions

- Please use Google Classroom to upload your submission by the deadline mentioned above. Your submission should comprise of a single file (ZIP), named `<Your_Roll_No>_Assign3`, with all your solutions.
- For late submissions, 10% is deducted for each day (including weekend) late after an assignment is due. Note that each student begins the course with 7 grace days for late submission of assignments (with a max of 4 per submission). Late submissions will automatically use your grace days balance, if you have any left. You can see your balance on the CS5480 Marks and Grace Days document under the course Google drive.
- You should use PYTHON for the programming assignments.
- Please read the department plagiarism policy. Do not engage in any form of cheating or plagiarism - if we find such behavior in your submission, both receiver and giver will be imposed with severe penalties. Please talk to instructor or TAs if you have concerns.

1 Programming Questions

In this assignment, you will design and implement a character-level Recurrent Neural Network (RNN).

1. **Find and Load Training Text (3 marks):**
   (a) Choose the text you want your neural network to learn, but keep in mind that your data set must be quite large in order to learn the structure! RNNs have been trained on highly diverse texts (novels, song lyrics, Linux Kernel, etc.) with success, so you can get creative. As one easy option, Gutenberg Books is a source of free books where you may download full novels in a .txt format.

(b) We will use a *character-level* representation for this model. To do this, you may use extended ASCII with 256 characters. As you read your chosen training set, you will read in the characters one at a time into a one-hot-encoding, that is, each character will map to a vector of ones and zeros, where the one indicates which of the characters is present:

$$char \rightarrow [0, 0, \cdots, 1, \cdots, 0, 0]$$

Your RNN will read in these length-256 binary vectors as input.

2. **Implement an RNN:**

(a) **Backpropagation Through Time (2 + 2 marks):** In an RNN with a single hidden layer, you should have three set of weights: W_{xh} (from the input layer to the hidden layer), W_{hh} (the recurrent connection in the hidden layer), and W_{ho} (from the hidden layer to the output layer). Suppose you use *Softmax* units for the output layer and *Tanh* units for the hidden layer, show:

i. Write the equation for the activation at time step t in the hidden layer and the equation for the output layer in the *forward propagation* step.

ii. Write the equation for the *weight update rule* at time step t for W_{xh}, W_{hh}, and W_{ho} in a vectorized notation. Suppose the backpropagation goes back to time step $k(0 < k < t)$.

(b) **Network Training (3 + 3 + 3 marks):** Train your recurrent neural network using the dataset you created in 1(b). You are free to choose learning parameters (sequence length, learning rate, stopping criteria, etc.). Complete the following task:

i. Report your training procedure. Plot the training loss vs. the number of training epochs.

ii. During training, choose 5 breaking points (for example, you train the network for 100 epochs and you choose the end of epoch 20,40,60,80,100) and show how well your network learns through time. You can do it by feeding in the network a chunk of your training text and show what is the output of the network. Report your result.

iii. To add randomness into the outputted sequence, we can either change the beginning character, we can randomly initialize the hidden state or we can introduce randomness via the Softmax function. In this case, if we have already seen sequence $x = (x_1, \cdots, x_t)$ of characters, then the probability that the output will be character j at time-step $t + 1$, i.e. $y_{t+1} = j$, is given by the Softmax function,

$$p(y_{t+1} = j|x) = \frac{e^{x^T w_j}}{\sum_k e^{x^T w_k}} \rightarrow \frac{e^{(x^T w_j)/\tau}}{(\sum_k e^{x^T w_k})/\tau}$$

where $\tau \in (0, \infty)$ is the temperature. Intuitively, at higher temperatures, $\tau \rightarrow \infty$, all characters will be chosen equally likely and at low temperatures $\tau \rightarrow 0$, only the most probable character will be chosen with nearly probability 1.0. So in order to generate the next character of the sequence, simply sample from this probability. Please report your result of 5-different generated texts of length-100 once your model is fully trained. Try this for three different temperatures t. What dynamics do you expect and what do you see?

(c) **Experiment with Network Structure (2 + 2 marks):** As before, we want to explore how the network learns when we change parameters:

i. **Number of hidden units.** Try doubling and halving your number of hidden units. Like in 2(b), plot the training loss vs. the number of training epochs and show your text sampling results. Discuss your findings.

ii. **Sequence length.** Try doubling and halving your length of sequence that feeds into the network. Like in 2(b), plot the training loss vs. the number of training epochs and show your text sampling results. Discuss your findings.

3. **Extension to LSTM (6 marks):** Generalize your RNN to utilize LSTM units. This is a non-trivial generalization, but you may find the equations of Chapter 4 of Alex Graves thesis to be very useful. http://www.cs.toronto.edu/ graves/preprint.pdf. As an additional resource for RNNs, the link `https://github.com/kjw0612/awesome-rnn` is extremely helpful. Show your result in the format of 2(b)(i), and discuss your observations.

In addition to the above marks, there will be 9 marks for goodness/correctness/documentation for code, thus totaling to 35 marks.

To Submit: Your code and your observations as a report.