Deep Learning
Assignment 3
10 points

In this assignment you will be answering some questions about recurrent network architectures and word embeddings, and running a simple LSTM implemented in TensorFlow.

1. Reber Grammar and Extended Reber Grammar

a. [1 mark]

Which of the following strings are in the language specified by the Reber Grammar (i.e. they can be generated by the automaton)? Which are not?

(i) BPVPSE

(ii) BTSXXTVPXXVPSE

(iii) BTSSXXTVPXTTVPXVVE

(i) and (iii) are in the language specified by the Reber Grammer.

(ii) is not.

b. [1 mark]

For each of these strings of the Extended Reber Grammar, what would be the correct prediction for the next letter?

(i) BPBPVVE

(ii) BTBTSSXXTTVPSE

(i) P

(ii) T

2. [3 marks] Briefly describe the problem of long range dependencies, and comment on the ability of each of the following architectures to learn long range dependencies:
NETtalk, Simple Recurrent Networks (SRN), Long Short Term Memory (LSTM), Gated Recurrent Units (GRU).

When the gap between the relevant information and the place that it's needed is small, RNNs can learn to use the past information. But when the gap between the relevant information and the point where it is needed is very large, in practice, RNNs don't seem to be able to handle such "long-range dependencies.

NETtalk: This kind of approach can only learn short term dependencies, not the medium or long term dependencies that are required for some tasks.

SRN: Simple Recurrent Networks can learn medium-range dependencies but have difficulty learning long range dependencies. Context layer is combined directly with the input to produce the next hidden layer.

LSTM: LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior. The repeating module has four neural network layer interacting in a very special way instead of having a single layer. It has three gates. It can learn long range dependencies better than SRN.

GRU: Gated Recurrent Units can learn long range dependencies better than SRN. GRU is similar to LSTM but has only two gates instead of three.

3. [3 marks] Briefly explain why full softmax may not be computationally feasible for language processing tasks. Name, and briefly describe, two alternatives to full softmax which are computationally feasible.

For language processing tasks, the model needs to predict the next word in a sequence when a sequence of words are given as input. So the last computational step in the model involves choosing word to select from a huge list of words in the vocabulary. This step is very computation intensive and full softmax may not be computationally feasible for language processing tasks.

So there are some alternative to full softmax which can reduce the computation.
Hierarchical softmax: target words are organized in a Huffman-coded binary tree. Only those nodes that are visited along the path to the target word are evaluated.
Negative sampling: To make the computation cheap due to the volumn of the vocabulary, we only work on a few negative words that are wrong. This is a simplified version of noise constrastive estimation.

4. [2 marks] Understanding LSTM in TensorFlow using MNIST
python tf_lstm.py
Copy the final Training Accuracy, Loss and Testing Accuracy into your answers.

```
For iter  790
Accuracy  0.9765625
Loss  0.10105695
__________________
Testing Accuracy: 0.984375
```

Increase the number of iterations from 800 to 2000, by changing this line
and copy the final Training Accuracy, Loss and Testing Accuracy into your answers.

```
For iter  1990
Accuracy  0.9921875
Loss  0.06635397
__________________
Testing Accuracy: 0.984375
```