Deep Learning Assignment 4

Sam Royston
New York University
sfr265@nyu.edu

1 Introduction

In this assignment, we were asked to answer a few conceptual questions about rnns and the lstm model and to take an existing neural network designed to be trained on sequences of words and modify it so that it.

1. can generate sequences of words in an interactive mode
2. can be trained on sequences of characters and can generate sequences in this context as well

The data used was the canonical Penn tree-bank dataset. This dataset consists of roughly 1M words for training and, 77K and 82K for validation and testing, respectively. The data is composed of various annotated text from the Wall Street Journal, The Brown Corpus, Switchboard, and ATIS. The presence of data from the wall street journal is especially noticeable when the networks are asked to generate text; it often talks business.

The generative outputs from the word model were cleaned and re-sampled such that neither the input nor output are affected by the vocabulary limitations of the learner.

2 Architecture

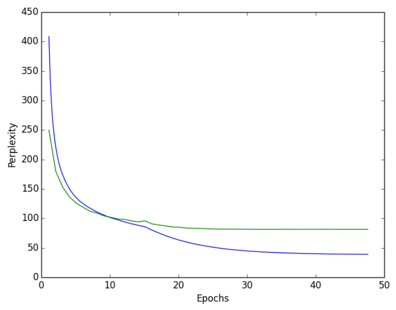
The model was a two layer recursive neural network which utilized the LSTM unit to make better sense of sequential data. The LSTM unit incorporates a novel storage dynamic into RNN architectures by offering the option to write to it, read from it, or leave it untouched. Graves [2] contends that it is a good idea to give RNNs another avenue for remembering data because while it is "theoretically possible for RNNs to learn arbitrarily long sequences", they rarely remember input beyond a few time-steps into the past. If dropout was used, it was only exercised across connections within the same time step, as per [1], a practice which helps to ameliorate the "amnesia" experienced by RNNs. It is easy to see how dropout, (or other regularization techniques for that matter) when applied across temporal connections will contribute to a loss of temporal information, and so it's restriction provides a useful way to increase the recognition of temporal patterns in a net.

3 Learning Techniques

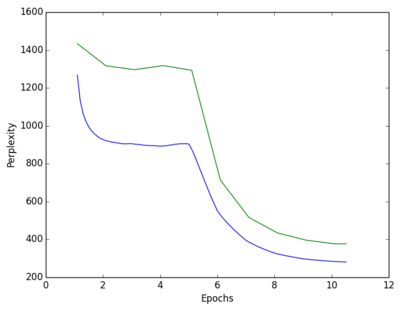
The most successful model was very similar to the baseline "hour" model for words, but reformatted for characters and run for approximately 9 hours. The distinguishing features of this model, relative to the supplied parameters, was that it used a 10% dropout rate and also began to apply the decay much earlier. The latter modification was due to an observation that the character model plateaus until the learning rate begins to decay, so that the decay should start a bit earlier.

Figure 1: In both (a) and (b) we can see a notch where the learning rate begins to decay. This is an example of how a smaller learning rate can help the optimization find it's way through "confusing" surfaces. Training error is blue and validation error is green.

(a) Word based model



(b) Character based model



Based on my experiences watching the perplexity get hung up in different situations, I would have liked to implement an adaptive learning rate that takes into account the behavior of the training error during the previous epoch.

4 Performance

Our character model achieved a validation perplexity of 360 and the word-based model reached a perplexity of 81. Had the character model been able to run longer, it surely would have continued to drop.

5 Discussion

It is interesting to think about whether the difficult-to-navigate portions of the optimization surface are representative of overcoming some actual linguistic challenge, i.e. learning about sentence structure. I had hoped that looking at the evolution of phrases through learning might shine some light on this, but ultimately, one needs much more; probably a detailed statistical analysis of the generated text at each step. One interesting observation is that the network learns a significant amount of structure even within the first epoch. Below are some entertaining and illustrative examples, computed using the character model, cut off at the first sentence.

First epoch, perplexity: 1600 *the meaning of life is*

- $ N bid to merchance marketer 's stages spiblics player as a u.s. big boosta 's month bank ¡unk¿ an affem the stons by efficient was liqity.
- medilican 's ivelation opting rise trouble much a portfonersoll poor to gain
- allowed to predecede volume the opportuses

Perplexity: 455 - 470 *the meaning of life is*

- just the end of N cents selling van according to the blapt weather tests and known on the compaq
- basing after the leader
- n't countered to financing to receive the japanese
- large scrambling on his own destroy with the second quarter an edition that 's no needed about the deadly $ N billion losing projects which come on the chicago bag groups not yesterday

6 Assigned questions

6.1 Q1

See attached file `nngraph_handin.lua`

6.2 Q2

In the function `lstm(i, prev_c, prev_h)`, the arguments represent the following:

***i*:** The output of the previous layer at the current time step, denoted as h_t^{l-1} in the paper.

***prev_c*:** Holds the previous value of the memory cell, written as c_{t-1}^l in the paper.

***prev_h*:** The output of the current layer at the previous time step, denoted as h_{t-1}^l in the paper.

6.3 Q3

The `create_network()` function returns an RNN according to the settings specified in the `params` variable, adding the specified number of layers in a loop. The network created is not unrolled, which is done with the `g_cloneManyTimes()` call in `setup()`.

6.4 Q4

`model.start_s` refers to the output which the net starts BPTT, and can also be thought of as the *output at the current time-step.* `model.start_s` is set to 0 at the start of each new input sequence because there is no input yet, and therefore should be no output either. The table `model.s` is built incrementally as the sequence length increases or shifts one unit to the right. `model.s` is a table of successive outputs (through time) from the hidden layers of the net, while `model.ds` are the gradients from performing back propagation through the different time-steps.

6.5 Q5

We define a maximum norm in the parameters variable, and this is used to limit (or clip) the size of the gradient. This is used to prevent undesirable behavior of the backprop algorithm (jumping over minima) which may occur in certain conditions, such as when the Hessian has disparate eigenvalues.

6.6 Q6

Stochastic gradient descent.

6.7 Q7

Since the extra input is not actually needed by the backprop algorithm, I just passed in a dummy tensor of the correct size so that the graph was consistent.

References

[1] Wojciech Zaremba, Ilya Sutskever, Oriol Vinyals. (2014).arXiv 1409.2329
Recurrent Neural Network Regularization. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).

[2] Alex Graves *Generating Sequences With Recurrent Neural Networks.* arXiv:1308.0850.