Deep Learning A4
LSTM Network

Mark Liu
Courant Institute
New York University
ml4133@nyu.edu

Abstract

I use a regularized LSTM RNN to build a probabilistic character level language model for English.

1 Data set

I use the character level PTB data set processed by Tom Secru [2]. The data is a collection of natural English sentences, split into sequences of characters. There are 50 unique characters appearing in the data set.

2 LSTM RNN

2.1 Architecture

The neural network is 2 layers. Input is connected sequentially through

1. An $M \to N$ lookup table
2. A dropout module
3. An LSTM module with state size N
4. A dropout module
5. An LSTM module with state size N
6. A dropout module
7. An $N \to M$ linear module
8. A LogSoftMax module

(Actually the loss function is also built into the network as a final node but I will talk about the loss function later)

The LSTM modules are each connected to themselves through time as usual

$M = 50$, the number of characters that the neural network can recognize (this was determined by the data set.) $N = 200$, the size of the states inside the LSTM modules.

The network is supposed to take in one of M character classes and provide a distribution (log probabilities) of the next character class to appear in a natural English sentence.

2.2 Learning Procedure / Error Criterion

Since I want to interpret the output of the network as a list of log-probabilities, I use the ClassNLL-Criterion as the loss function.

At each time step, I pass the current integer/character into the network and I put the output of the network into the loss function together with the next integer/character.

The parameters of the network are optimized using back-propagation through time an unrolling of 10 time steps and a learning rate of 1 with a decay of 2 and a dropout parameter of .2. Gradients are clipped at euclidean norm 5. The network was trained for 2 epochs over 250874 batches of size 20. Validation of the network was performed over 19652 batches of size 20.

2.3 Other Experiments

I experimented with sequence length (number of times that I unrolled the network) from 5 to 50 and with the dropout parameter. Even reducing dropout to zero did not seem to affect my results much, possibly because the time spent training was too low (on the order of 1 hour). Dropout would only be beneficial to counteract over-training on long time scales. Changing the sequence length also did not affect results significantly, possibly because the character-level model is so local. My intuition is that the most significant information about English character transitions is possibly contained within a window of about five characters.

3 Results

I use perplexity as the error metric and obtain 4051.978 perplexity on the training set and 2201.205 perplexity on the validation set.

4 Questions

1. I have provided this file in the base directory of the repository
2. (a) $i = h_t^{l-1}$
   (b) $prev\_c = c_{t-1}^l$
   (c) $prev\_h = h_{t-1}^l$
3. create_network() returns one time slice of the unrolled network. setup() copies this time slice t times so that they can be fed through each other during the backprop through time. In my case $t = 10$
4. (a) model.s holds the prior memories and outputs (c and h) of the LSTM modules, needed for forward and backprop.
   (b) model.ds holds the gradients with respect to the memories and outputs of the LSTM modules at the current time during backprop.
   (c) model.start_s holds the starting memories and outputs of the LSTM modules when running forward prop. Since the network is unrolled a finite number of times ($t = 10$), model.start_s is reset every time the network is exposed to a sequence of t inputs during training/validation.
5. Gradient clipping is used when gradients get bigger than $max\_grad\_norm = 5$. This means during optimization, gradients larger than norm 5 are scaled down to be norm 5.
6. Stochastic gradient descent is used with backprop through time
7. The extra output is a node which exposes the LogSoftMax layer. This node is never connected to the loss function so it has no effect hence when I do backprop, I set the gradient with respect to this node as 0.

References

[1] Zaremba, W., Sutskever, I. (2015) *Recurrent Neural Network Regularization* ICLR 2015

[2] Secru, T. (2015) *PTB Data* https://github.com/tomsercu/lstm/tree/master/data