A4 homework submission
David Halpern
Deep Learning 2015, Spring

David Halpern
Department of Psychology
New York University
david.halpern@nyu.edu

1 Questions

1.1 Q2

i = i
prev_c = c^l_{t-1}
pre_h = h^l_{t-1}

1.2 Q3

Returns a rolled version of the core RNN module with parameters initialized uniformly within an interval.

1.3 Q4

model.s is the trained of the sequence. model.ds is the gradients of the model for the backward pass. model.start_s is the final hidden states of the current minibatch and gets reset when the sequence ends

1.4 Q5

Clipped at params.max_grad_norm (shrink factor?)

1.5 Q6

Batch SGD

1.6 Q7

Since the prediction was essentailly outside the model and the nLL criterion was computed before the prediction, the prediction should have no impact on the gradient so the gradient is a large table of 0s.

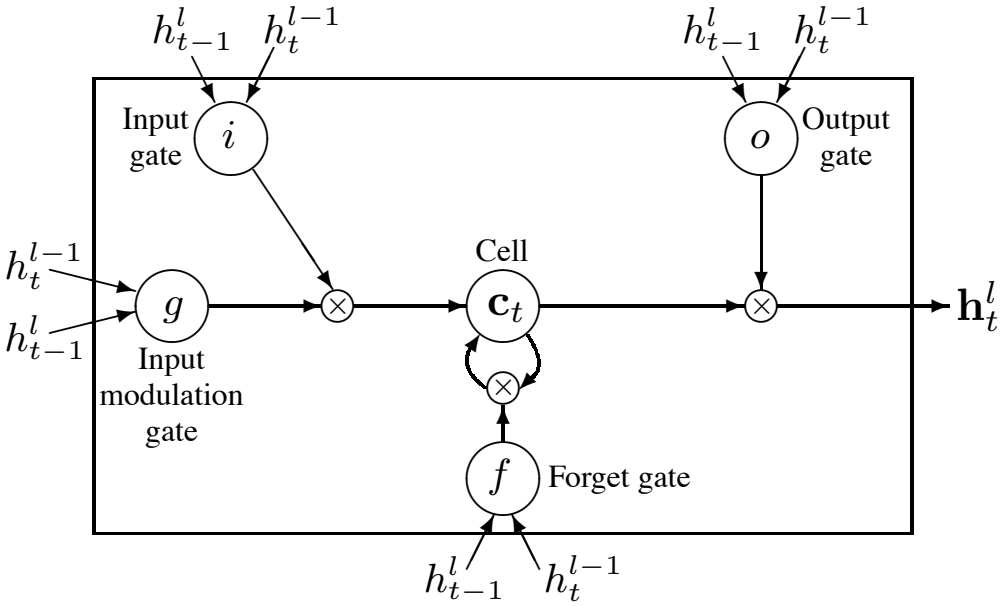

Figure 1: Example LSTM cell

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} T_{2n,4n} \begin{pmatrix} \mathbf{D}(h_t^{l-1}) \\ h_{t-1}^l \end{pmatrix}$$

$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

$$h_t^l = o \odot \tanh(c_t^l)$$

Figure 2: Example LSTM cell

2 Model

2.1 Architecture

The basic model used was very similar to the one used by Zaremba (2014). The model uses a Long Short-Term Memory (LSTM) cell as the basic neural unit (see Figure 1). The LSTM proceeds by computing the equations in Figure 2. The cell in a particular layer takes as input h_t^{l-1}, the state of the previous layer at time t, and h_{t-1}^l, the state of the current layer at the previous time step. These cells are combined into a standard recurrent neural network with two layers (Figure 3). In order to

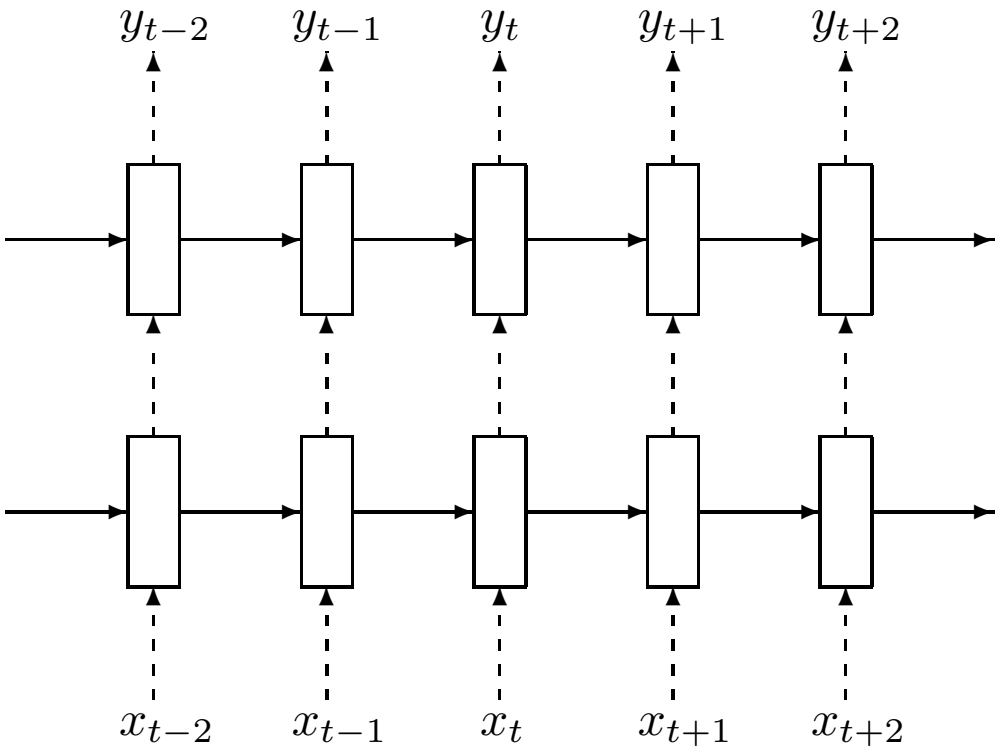

Figure 3: Example RNN. x_t is the previous character and y_t is the next character to be predicted

be read by the network, words and characters were transformed into a sparse vector representation with each column representing a word in the vocabulary of the model. The Penn Treebank dataset included 10000 words in it's vocabulary and 50 characters. The model was unrolled to a sequence length of 20.

2.2 Training

The model was trained using mini-batches of size 20, a learning weight of 1, a gradient clip of 5 and no dropout. Parameters were initialized uniformly on [-.1, 1]. After the fourth epoch, learning rates decayed by half every epoch. The model was run for 13 epochs. The Penn Treebank contains 929k words in its training set, 73k words in it's validation set and 82k words in its test set.

3 Experiments

3.1 1

In this experiment, I attempted to increase the number of nodes in the rnn from 200 to 400, under the assumption that more nodes would allow for more discriminable states of the network. Having more states also increased the effectiveness of dropout since there were still more nodes turned on even while training so I used a dropout rate of .2. Dropout and more nodes allow for relaxing the gradient clipping from 5 to 7 since there is more regularization. This (theoretically) allows for more flexibility in learning the parameters in the larger network. The model was also unrolled to a length of 50 under the assumption that memory could be stored longer in the bigger network. However, in order to speed up the training scheme, I used a batch size of 50. It is possible that this had a negative effect on parameter learning and in the end perplexity of this model bottomed out at 367.211 after 13 epochs. Network parameters:

```
{
  max_grad_norm : 7
  seq_length : 50
  batch_size : 50
  lr : 1
  max_max_epoch : 13
  rnn_size : 400
  init_weight : 0.1
  decay : 2
  dropout : 0.2
  layers : 2
  vocab_size : 50
  max_epoch : 4
}
```

3.2 2

In this model, I simply tried increasing the number of layers. I used a small amount of dropout here to regularize the gradients which could explode even more in the larger network. Again to speed up training, I used a larger batch size which likely had a negative impact and this model got to a validation perplexity of 425.576 after 13 epochs.

```
Network parameters:
{
  max_grad_norm : 5
  seq_length : 50
  batch_size : 50
  lr : 1
  max_max_epoch : 13
  rnn_size : 200)
  init_weight : 0.1
```

```
  decay : 2
  dropout : 0.2
  layers : 3
  vocab_size : 50
  max_epoch : 4
}
```

References

[1] Zaremba, W., Sutskever, I., & Vinyals, O. (2014). Recurrent Neural Network Regularization. Arxiv, http://arxiv.org/pdf/1409.2329v4.pdf