NLP Assignment 2

Matthew Dunn - mtd368

10/16/16

1 Bag of Words

1.1 Short Comings

The primary deficits of the bag-of-words (BOW) approach to language understanding is that it doesn't take into account the ordering of words. As a result, BOW models disregard contextual information that may improve their ability to predict the next word in a sentence. An example of the context information BOW models disregard, follows

I grew up in Paris and the language I speak is _____.

What we intuitively do is predict the word *French* because of the contextual information gleaned from the beginning of the sentence. Unfortunately, BOW models don't take this information into account and as a result their prediction accuracy suffers.

1.2 Modified LSTM as BOW

The simplest modification to LSTM model to recover the BOW classification model through gradient descent would be to remove all of the functions inside the LSTM cell and simply add the vectorized word embedding output coming from one LSTM cell, h_t, to the vectorized input coming into the subsequent LSTM cell, x_t. Effectively turning the memory cell of the LSTM into an addition function that adds the previous LSTM cell output, h_t to the current input, x_t as shown in Figure 1 [1]. Finally, these would be multiplied by a weight matrix W and a activation function would apply non-linearity to it. This approach would simulate the traditional BOW modeling approach.

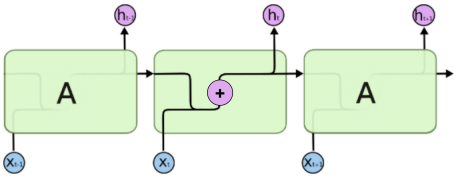
2 RNN Language Model

2.1 Train a Language Model

The process of training the language model with the starter code was fairly straightforward. Initially the LSTM was run without early stopping and as a

[1] `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`

Figure 1: LSTM to BOW



result it was manually stopped at Epoch 11 because the performance on the validation set had started to deteriorate, indicating that the model was overfitting the training data set. With an early stopping parameter set to end training after one Epoch of deteriorating validation performance, the model's perplexity score for training, validation, and test respectively was 38.41, 124.24, and 129.8.

2.2 Most Important Gate

A problem encountered when training the LSTM model was that the learning rate seemed to stay to high for too long during training. As a result, I attempted to address the issue by setting the max_epoch configuration variable to a lower number so the learning decay would start more quickly. Unfortunately, I overestimated how quickly the learning rate needed to decrease, so my three experiments examining which gates are the most important don't converge very quickly or result in reasonable perplexity scores. Nonetheless, it is clear that the most important gate was the

Table 1: LSTM Gate Test Results

LSTM Gate Removed	Test Perplexity
Output o	639.379
Input i	639.449
Forget f	143.037

2.3 Gated Recurrent Unit

To increase the speed of training after the initially training of the LSTM models, I went ahead and launched a large AWS server with Tensorflow installed to speed up training. The AWS instance had 60 GBs of ram, which allowed me to lower my training time dramatically. This fact is reflected in the Words Per Second column in Table 2 below. With this server deployed, I went ahead and tested various hidden dimension sizes and tried different learning rates. I found the large dimension sizes un-trainable in a reasonable amount of time, so I aborted two attempts at much higher dimensions. Also of note, is that I initialized the learning rate to 1.0 for all the models.

Table 2: GRU Hand Tuning Results

Max Epoch	Decay Rate	Hidden Dim	Words Per Second	Test Perplexity
2	0.25	200	Apx. 3890	536.039
2	0.5	200	Apx. 3890	124.635
2	0.5	500	Apx. 1638	Aborted
2	0.5	2000	Apx. 241	Aborted

The model that performed best on the test data set is shown in the table above. I think the reason it performed so well was the decay rate was set such that learning was fast initially, but the rate became small enough to explore small local minimum towards the end of the training cycle. The model that converged most quickly during training was the model with a 0.5 Decay Rate and only 200 Hidden Dimensions. I think it did so because the learning rate was large at the outset and the hidden dimension was not so large that the signal to noise ratio in each node was poor. The training and validation curves for this model are available in the Appendix under Best GRU Learning Rate and Train/Val Curves.

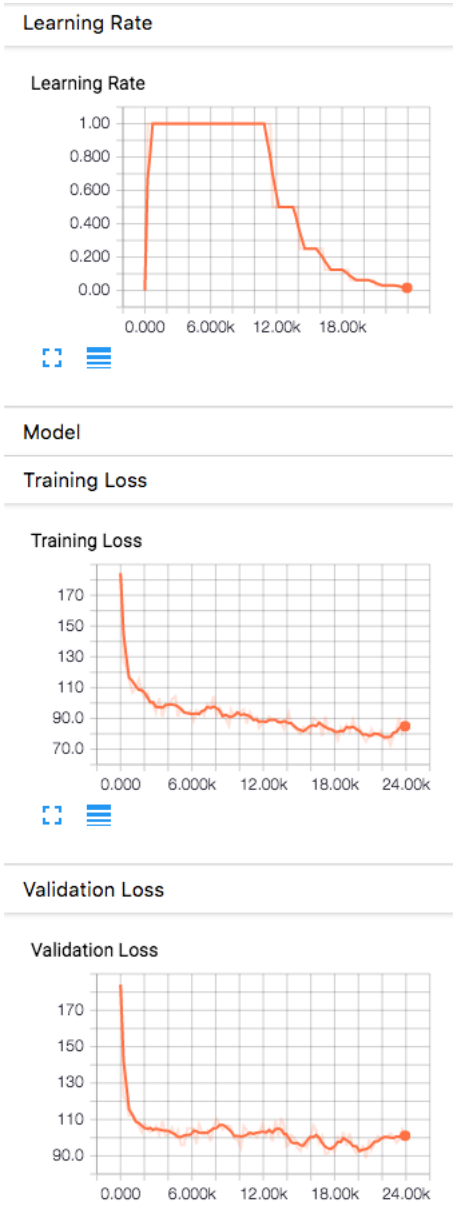
2.4 Setup Evaluation

For the best GRU model, I was able to achieve a cosine similarity of 9. Lastly, to visualize the performance on the model, I mapped my TSNE dimension reduction algorithm to plot the results shown in the gru.png file in the zip submission.

3 Appendix

3.1 LSTM Learning Rate and Train/Val Curves

Figure 2: LSTM Learning Rate and Train/Val Curves



3.2 LSTM Gate Importance

Figure 3: LSTM Gate Tests: Learning Rate and Train/Val Curves

3.3 Best GRU Learning Rate and Train/Val Curves

Figure 4: Best GRU Learning Rate and Train/Val Curves