Assignment 1

I-Tsun Cheng, 20576079

March 2, 2019

1 LEARNING RATE

The learning rate determines the speed at which the weights and bias update upon each iteration of gradient descent. During each iteration, the inputs, weights, and bias are fed into forward propagation of the neural network, loss is calculated by comparing the prediction and true label, and gradients of weights and bias are computed through backpropagation. Weights and bias are updated depending on the learning rate, as given by: $w = w - \mu \times \frac{\partial L}{\partial w}$ and $b = b - \mu \times \frac{\partial L}{\partial b}$ where μ is the learning rate.
Finding the best learning rate is crucial to the accuracy of the neural network. Too low of the learning rate results in very slow update of the weights and bias and thus longer training time. Too high of the learning rate causes the gradients to update too large during each iteration (gradient explosion) and therefore the neural network will be difficult to converge. The best learning rate I found for uni-gram with the maximum vocab size of 10000 is: **0.001**.
With this vocab size and learning rate, the best training accuracy is **0.6911%** and the best developmental accuracy is **0.6858%**.

2 PREPROCESSING

I implemented the follow string preprocessing methods:

- Remove stopwords
- Replace punctuation with PUNC
- Replace numbers with N
- Replace html links with HTML
- Replace words out of dictionary with UNK

List of stopwords and dictionary are fetched from SCOWL and Friends.

Info	Number/Value
Words	532564
Max Sentence Length	165
Sentences	20000
Vocabulary	10000

Table 2.1: Before string preprocessing

Info	Number/Value
Words	422712
Max Sentence Length	417
Sentences	20000
Vocabulary	10000

Table 2.2: After string preprocessing

3 Normalization

Normalization squeezes all data so that the data distribution is mean 0 and standard deviation 1. It helps since it squeezes the data closer together without affecting the data values too much. Without normalization and running unigram with learning rate as 0.001, the best training accuracy is **0.5285%** and best developmental accuracy is **0.5380%**. As the results show clearly, if the data is not normalized before feature extraction, the performance is poorer since all data are scattered so widely in an open space.

4 Wrong Prediction of Test Sentences

Here is a table of ten sentences with the wrong prediction by the neural network.

ID	Sentence	Predicted label	Human label
3	cairns in 20 minutes bed here i come!!	neg	pos
20	is loving the weather	neg	pos
24	@ u should come say hi to me at work! im here by myself	neg	pos
30	@ your not baby!!	pos	neg
33	slept in and didn't mean to. oops.	pos	neg
39	eating a slice of toast with rasberry jelly yummi	neg	pos
48	@ well i don't feel bad about it that's for sure.	neg	pos
115	@ i am so glad that u are and always will be team edward, i don't really like jacob fans	neg	pos
119	farah fawcett "golden-haired sex symbol of the late 1970s" has died from cancer, aged 62 URL very sad	pos	neg
128	@ ahh i can't wait for pictures and stories! miss you boo boo	neg	pos

Table 4.1: Wrongly predicted sentences

The neural network sometimes predicts sentences wrongly since the sentiment of some individual words contributes to the overall sentiment of the sentence. For example, looking at so many training examples, "baby" might mostly come out as positive sentiment, so ID 30 is predicted as positive, and "boo" might come as negative sentiment so ID 128 is predicted as negative. Also since all punctuations are replaced as a common PUNC token, some punctuations that may express some sentiment such as exclamation mark are disregarded.

5 Bonus 1: N-gram + BoW

I implemented n-gram where the vocabulary and the test sentences are represented by tokens with n words. For bigram, the best training accuracy is **0.6962%**, while the best developmental accuracy is **0.6645%**.
Based on this result, using bigram results in higher training accuracy but lower developmental accuracy than using unigram. This indicates that representing vocabulary and sentences by two-word phrases do not necessarily result in better performance because there are not a lot of meaningful two-word tokens in the tweets, so single words might actually be better for representation. Similar conclusions can be made for n-gram representations.

6 Bonus 2: Gradient Descent with Momentum

Momentum is another more effectively method of gradient descent. The normal gradient descent without momentum for updating gradients is given by: $w = w - \mu \times \frac{\partial L}{\partial w}$ where μ is the learning rate. When the loss function reaches a saddle point or local minimum, the gradients at that point will be zero hence the gradients will not update. However, sometimes there is a lower loss elsewhere so momentum is used to give a movement in a direction even if the gradient is zero. The formula of gradient descent with momentum is:

$$\begin{aligned} v_i &= \lambda \times v_{i-1} - \mu \times \frac{\partial L}{\partial w} \\ w_i &= w_{i-1} + v_i \end{aligned} \tag{6.1}$$

I found out that for weights and bias, the coefficient of momentum of 0.9 performs well, and initial v_0 is set as 0. When weights and bias are updated using momentum with same learning rate of 0.001, the best training accuracy is **0.7483%**, while best developmental accuracy is **0.7098%**. This shows that gradient descent with momentum is more effective than purely using gradients for this sentiment analysis task.

7 Bonus 3: Some training tips or phenomenon

If learning rate is increased to more than 0.002, which is 0.001 above the optimal level, some loss would result in inf since the learning rate is too high that the gradients update too quickly causing a very big loss. If learning rate is even increased more, then some loss will become NaN, exceeding the maximum value the data type can hold.
If learning rate is set below 0.0008, the training and developmental accuracy would only be around 55%, which is only a little better than random guess. This suggests that gradient update is too slow so we need to increase the learning rate or run for longer epochs.
If string is not preprocessed before feeding into the model, the learning rate should be increased or otherwise the gradient descent will execute very slowly and it will take way more iterations to lower the loss and converge. To my surprise, the optimal learning rate if no string cleaning is performed is 0.01 which is 100 times faster than the learning rate when string cleaning is performed. When string is not preprocessed, the best training accuracy using this learning rate for unigram is 0.7989%, while the best developmental accuracy is 0.7518%. Therefore, the neural network actually performs better without string preprocessing at all.