Supervised and Unsupervised Learning for Sentiment Analysis

Hao Sheng

March 21^{st} 2017

Part 1

Looking through the data set of reviews, they are generally a few paragraphs long in length. When reading them, it would seem difficult to classify a review as positive or negative by purely looking for one or two key words. In other words, it does not seem to be the case that there is a single word that can differentiate a positive review from a negative one. However it does seem like there are a "collection" of words that would favour a review to being positive or negative. Thus if there were a combination of a few words such as "bad", "worst", and "horrible", it could suggest that the review is a negative one. Below is the frequency of occurrence of three words.

Word	Occurrence in Positive Reviews	Occurrence Negative Reviews
bad	25.9%	51.4%
worst	4.4%	19.4%
hilarious	12.5%	5.1%

It can be seen that the word "bad" occurs in both positive reviews. However with a higher occurrence in negative reviews, it can act as a partial indicator. The presence of more such "negative" words would likely indicate that the review is a negative one.

Part 2

To implement the Naive Bayes algorithm, we need the probability of the words appearing in positive or negative comments. The probability can be acquired by counting the number of appearance of each word and divided by the total number of positive/negative comments.

$$\begin{aligned} class &= argmax_{class}(Pr(class|a_1, ..., a_n)) \\ &= argmax_{class}(Pr(a_1, ..., a_n|class)Pr(class)) \\ &= argmax_{class}(Pr(class)\prod_i Pr(a_i|class)) \end{aligned}$$

In this case, Pr(Positive)=Pr(Negative). Taking log on the left hand side, we can convert multiplication to summation. The class that argmax the multiplication still argmax the summation.

$$class = argmax_{class}(log(\prod_i Pr(a_i|class))$$
$$= argmax_{class}(\sum_i \log(Pr(a_i|class))$$
$$= argmax_{class}(\sum_i log(\frac{count(a_i = 1, class)}{count(class)}))$$

The word that is not in the training will lead $log(Pr(a_i|class))$ to have a value of negative infinity. To avoid the this, instead of summing all $log(Pr(a_i|class))$ in a given comment, we sum up $log(\frac{count(a_i=1,class)+m*k}{count(class)+k})$.
To tune the parameter m and k, we used the following code and pick the combination with the highest performance on the validation set.

```
for k in [0.005,0.01,0.015,0.02,0.03,0.05]:
    for m in [1,2,3,5,10,15,30]:
        print('k',k,' m',m,' Accuracy',get_accuracy(posValSet,negValSet,m,k))
```

After averaging 5 runs, the performance peak at m=1 and k=0.015. Using Naive Bayes, the performance on the training set is 0.994 and performance on the test set is 0.805.

Part 3

The top ten words that suggest a negative review with the Naive Bayesian algorithm are: 'waste', 'poorly', 'lame', 'bland', 'wasted', 'stupid', 'worst', 'awful', 'unfunny', and 'ridiculous'.

The top ten words that suggest a positive review with the Naive Bayesian algorithm are: 'terrific', 'ben', 'contrast', 'fantastic', 'wonderfully', 'portrayal', 'outstanding', 'believes', 'era', and 'allows'.

For each word, the occurrence of the word in the positive review data set is normalized by the total occurrence of the word in the total review data set. The same is done with respect to the negative review set. Thus words with the highest percentage of occurrence in the positive data set were selected as indicators of positive reviews and words with the highest percentage of occurrence in the negative data set were selected as indicators of negative reviews.

Part 4

Logistic regression was performed using the review data set. Moreover for regularization, the magnitude of the weights was added to the cost function such that:

$$Cost_{WD} = Cost + \frac{\lambda}{2}\sum_{i,j,k}(w^{(i,j,k)})^2$$

and,

$$\frac{\partial Cost_{WD}}{\partial w^{(i,j,k)}} = \frac{\partial Cost}{\partial w^{(i,j,k)}} + \lambda w^{(i,j,k)}$$

This was done to prevent any single weight in the regression to "overpower" the rest of the weights. Thus limiting the ability of a single word to become the deciding factor on whether the review is positive or negative. Below is the plot of test set performance versus iteration for multiple λ constants ranging from 0 to 0.4. It can be seen that a λ does not have a huge impact on the test set performance. This is likely be because the weights were already quite well distributed among all the words and this potential problem of a single weight "overpowering" the others did not occur. Overall, the performance of the regression plateaued around 83% for the test set.

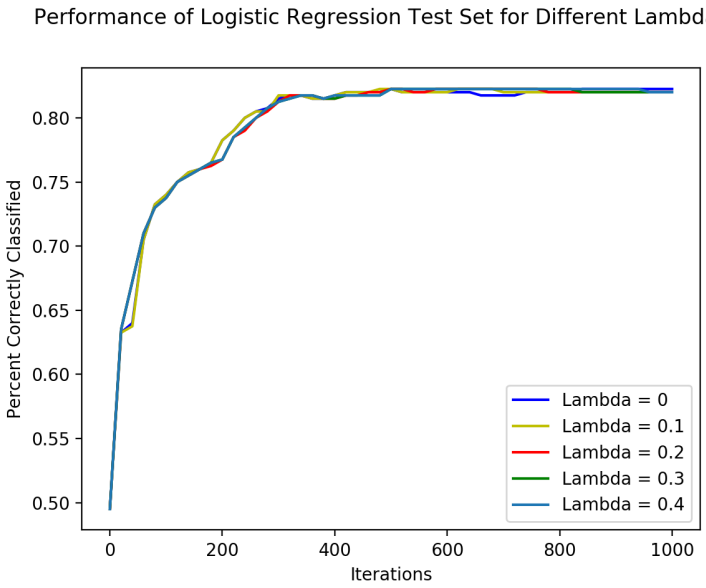

Figure 1: Logistic Regression for different λ values

Part 5

For Logistic Regression, the θ_i values represent the weights of the regression for word i^{th} and the I_i value represents the occurrence of the word i^{th} in the particular test review (either 0 or 1). More specifically, a weight of large magnitude for a particular word would mean that the presence of word i^{th} in a review either increases the possibility of the regression classifying the review as positive (if θ_i is positive) or negative (if θ_i is negative).

For Naive Bayes,

$$class = argmax_{class}(\sum_i \log(Pr(a_i|class))$$

For the class being positive, we have:

$$\sum_i \log(Pr(a_i|Positive) > \sum_i \log(Pr(a_i|Negative)$$

$$\sum_i \log(Pr(a_i|Positive) - \sum_i \log(Pr(a_i|Negative) > 0$$

the I_i value also represents the occurrence of the word i^{th} in the particular test review (either 0 or 1). On the other hand, the θ_i values represent $log(Pr(a_i|Positive)) - log(Pr(a_i|Negative))$, the log probability that the i^{th} word is a positive or negative review given the training data. Note that a value greater than zero is a positive review, while less than zero is a negative review.

Part 6

A list of the top 100 θs (in magnitude) obtained from logistic regression using a lambda of zero and 2000 iterations and Naive Bayes:
Logistic Regression Negative Weights: ['bad', 'only', 'worst', 'unfortunately', 'should', 'awful', 'nothing', 'script', 'waste', 'have', 'boring', 'poor', 'supposed', 'given', 'plot', 'lame', 'wasted', 'fails', 'tries', 'mess', 'guess', 'poorly', 'worse', 'anyway', 'stupid', 'neither', 'editing', 'interesting', 'got', 'laughable', 'tv', 'reason', 'maybe', 'else', 'potential', 'naked', 'lifeless', 'used', 'd', 'bland', 'william', 'stuck', 'looked', 'point', 'save']
Logistic Regression Positive Weights: ['memorable', 'others', 'well', 'fun', 'performances', 'excellent', 'hilarious', 'quite', 'brilliant', 'flaws', 'best', 'very', 'amazing', 'great', 'different', 'yet', 'deserves', 'overall', 'throughout', 'pace', 'change', 'sometimes', 'definitely', 'people', 've', 'witty', 'surprised', 'seen', 'known', 'fight', 'follows', 'fantastic', 'become', 'using', 'hits', 'wonderfully', 'true', 'many', 'see', 'performance', 'solid', 'breathtaking', 'most', 'brings', 'normal', 'especially', 'laughs', 'sweet', 'right', 'details', 'job', 'other', 'makes', 'while', 'first']
Naive Bayes Negative Weights: ['unimaginative', 'hewitt', 'mining', 'incoherent', 'twilight', 'ludicrous', 'vehicles', 'ridiculously', 'miserably', 'fares', 'insulting', 'stream', 'yawn', 'seagal', 'unexciting', 'sucks', 'unlikable', 'fisherman', 'preston', 'idiots', 'stinks', 'climbing', '13th', 'formed', 'monotone', 'unbearable', 'passable', 'mortal', 'eszterhas', 'gunton', 'stupidity', 'justin', 'henstridge', 'bore', 'idiotic', 'massacre', 'suvari', 'excruciatingly', 'maniac', 'ritual', 'joanna', 'kidding', 'ditto', 'surveillance', 'inexplicable', 'garbage', 'spooky', 'predator', 'shane', 'turkey']
Naive Bayes Positive Weights: ['admits', 'avoids', 'endlessly', 'owes', 'macy', 'mpaa', 'concentration', 'outstanding', 'flees', 'doubts', 'affecting', 'addresses', '1970', 'manipulation', 'absorbing', 'recognizable', 'poker', 'grip', 'settled', 'marvelous', 'galore', 'lovable', 'groundbreaking', 'elliot', 'gas', 'resonance', 'keen', 'peaceful', 'skillfully', 'vulnerable', 'brilliance', 'regard', 'odyssey', 'thematic', 'cruelty', 'complaints', 'bastard', 'finest', 'hatred', 'seats', 'tad', 'communication', 'nazis', 'moody', 'darker', 'uplifting', 'guilt', 'magnificent', 'studies', 'sweetness']

Comparing the thetas of logistic and Naive Bayes, the words for logistic seem to be more "negatively" or "positively" oriented whereas there are some words in the Naive Bayes that do not seem like they would suggest if a review is positive or negative. These words include"13th", "1970", "poker", and "vehicles" to name a few. Although a few neutral words like this exist in the logistic thetas aswell, it is clear that there are many more of these in the Naive Bayes.

Part 7

The logistic regression I trained is intended to tell if the given word t and w can appear together or not. The positive data set is built using the review data where each word is formed into a word pair with each adjacent word that it occurs with in the text. The target output for this positive data set is (1,0). Then, we pick separate the word pairs into training, validation and testing sets.

The negative data set of word pairs, which have a target output of (0,1), is generated randomly from the "word2vec" dictionary and are added to the negative data set only if they do not occur

in the word pairs in the positive data set. Afterward, the word pairs are again seperated into training, validation and testing sets.

Then, the program uses word2vec function to make each word pair into a 1 by 256 vector(where each word is converted into a 128 dimensional vector). These training set vectors are then vertically stacked as our input. The architecture of the model takes 256 inputs which are fully connected to the output layer with 2 output units and a softmax layer which uses the logistic function to give the probability output. We also used the negative log loss as our loss function. All the weights are initialize to some small random value around zero. Finally, we trained the model with gradient decent for a maximum of 10 000 iterations.

The accuracy on the test set was 0.765 while the accuracy on the training set was 0.801.Below is a graph of the training and test set accuracy.

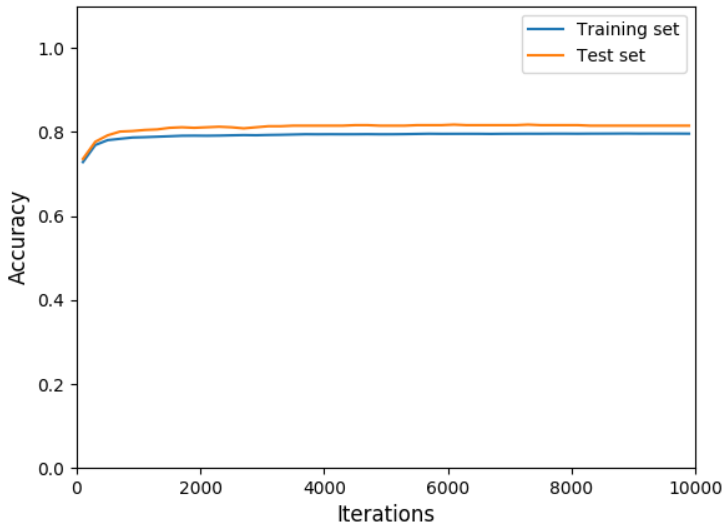

Figure 2: Accuracy of Word Pair Classification

Part 8

The cosine distance between embeddings was used to determine similar words. Below is a list of top ten closest words in terms of cosine distance.

Good: bad, great, wonderful, reinforcing, decent, funny, manipulate, underused, admiral, and perplexing.
Story: plot, film, benito, simmer, sitter, lift, domineering, ricci, interviews, and acclaim.

We found two more interesting examples of words that demonstrates that word2vec works to find similar words:

I: are we, they, you, he, she, it, shrinks, nuff, ive, and infectious.
Man: woman, journal, guy, boy, showtime, men, cancer, arising, olympia, and preacher.