HW1: Sentiment Classification

Jonah Philion
jonahphilion@college.harvard.edu
github

February 2, 2018

1 Introduction

In this problem set, we attempt to classify movie reviews as positive or negative. All models investigated take the form of

$$p(y_i) = \sigma(W\phi(x) + b)$$

where $p(y_i)$ is the probability a review x is negative. The models studied are

- naive bayes
- logistic regression with "bag of words" features
- multi-layer perceptron with "continuous bag of words" features
- convolutional neural net

And just for fun

- Fine-tuning ResNet classifier on images of text

2 Problem Description

For all models, a sentence $\boldsymbol{x}_i$ is encoded as a sequence $x_1, ..., x_n$ where each x_j is a one-hot vector of length the vocabulary $\mathcal{V}$. The classification y_i associated with $\boldsymbol{x}_i$ is 0 if $\boldsymbol{x}_i$ is positive and 1 if $\boldsymbol{x}_i$ is negative. Embeddings $\mathcal{E}$ map a one hot vector x_j to a dense vector of size d.

3 Model and Algorithms

All models are trained on the Stanford Sentiment Treebank (SST1). Unless otherwise specified, models requiring gradient descent are trained with an Adam optimizer of learning rate 0.001 and weight decay 0.0005. Training loss and validation loss are recorded in real time with visdom.

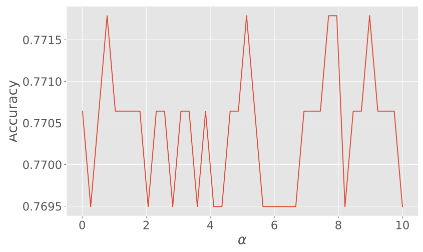

Figure 1: Affect of global smoothing parameter α on validation accuracy. Validation accuracy is calculated using sklearn's `accuracy_score`*. The performance is not sensitive to α.*

4 Experiments

4.1 Naive Bayes

All words for both labels are initialized to a count of α. Figure 1 shows that performance of the algorithm on the validation set is robust to changes in α. With $\alpha = .5$, performance on the test set using binarized and count word vectors are analyzed in Table 1.
The weight vector determined by Naive Bayes can be interpreted as the sentiment associated with particular words. The most positive and most negative words and their weights for binarized bag of words are shown in Table **??**.

Model	Acc.	Bce.	Roc.
BINARIZED	0.791	0.679	0.867
COUNTS	0.793	0.686	0.867

Table 1: Binarizing or counting words does not significantly affect the performance of the model on the test set.

stupid	unfunny	pointless	poorly	suffers	Feels	tiresome	car
5.39452	5.1085	5.03998	4.96641	4.96641	4.88701	4.88701	4.88701
powerful	solid	perfectly	inventive	refreshing	riveting	wonderfully	universal
-5.79766	-5.7109	-4.99019	-4.85754	-4.78398	-4.70458	-4.61832	-4.61832

Table 2: Words with the highest and lowest weights in the naive bayes weight vector agree with intuition for being negative and positive words respectively.

4.2 Logistic Regression

In the logistic regression model, we take the bag of words $\phi(x)$ but train w and b instead of calculating w and b. Figure 3 shows that as the model decreases cross entropy, the accuracy on the validation set also increases. However, this model does not do significantly better than naive bayes. The model is trained with batch size of 150 for 150 iterations with an adam optimizer of

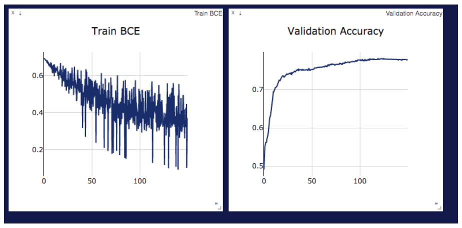

Figure 2: The left is the binary cross entropy recorded during training of binarized logistic regression and the right is the classification accuracy tracked during training. The objective function is tightly correlated with classification accuracy. The model is no better than naive bayes.

learning rate 0.0002 and weight decay 0.0005. The weight vector is initialized to zero to guard against words in the validation set which are not in the training set. The model is trained on binary and count vectors as with naive bayes. The metrics are recorded in Table 3. Significant words are recorded in Table 4.

Model	Acc.	Bce.	Roc.
BINARIZED	0.789	0.496	0.876
COUNTS	0.789	0.497	0.873

Table 3: Binarizing or counting words does not significantly affect the performance of the model on the test set.

bad	dull	mess	worst	too	flat	stupid	?
0.772932	0.687341	0.657265	0.648685	0.645756	0.611209	0.575218	0.572873
solid	fun	best	powerful	entertaining	heart	fascinating	love
-0.689454	-0.672897	-0.6613	-0.657444	-0.591135	-0.586589	-0.578906	-0.574162

Table 4: Words with the highest and lowest weights in the logistic regression weight vector agree with intuition for being negative and positive words respectively. There is some overlap with naive bayes.

4.3 Continuous bag of Words

In the Continuous bag of words (CBOW) model, one-hot vectors x_i for each word are mapped to a dense vector by ab embedding $\mathcal{E}$ in a lower dimensional feature space. These lower dimensional vectors are then summed or averaged. The order of the words is therefore lost during the transformation $\phi(x_i)$ as it is in the naive bayes model. The model is trained with an Adam optimizer of learning rate 0.00002 and batch size 100 for 350 epochs. The embedding is initialized to AWS word vectors but allowed to change during training. The embeddings improve the accuracy of the model over logistic regression and naive bayes as shown in Table 5 regardless of whether sum or max pooling is used. Figure **??** shows the objective and validation accuracy during training of the model with max pooling.

Model	Acc.	Bce.	Roc.
MEAN	0.824	0.416	0.898
SUM	0.816	0.448	0.886
MAX	0.798	0.447	0.880

Table 5: The word vectors improve the accuracy of the model.

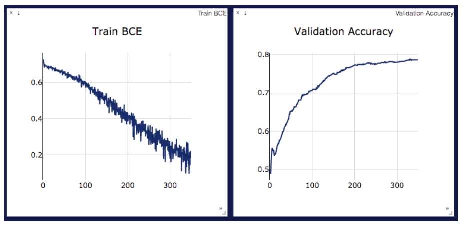

Figure 3: The left is the binary cross entropy recorded during training of CBOW and the right is the classification accuracy tracked during training. The objective function is tightly correlated with classification accuracy. The model is significantly better than naive bayes.

4.4 CNN

Following Yoon's CNN paper, we train a simple CNN with one layer of convolution. The convolution layer includes

- 300→ 50, kernel 3, stride 1, padding 1
- 300→ 25, kernel 4, stride 1, padding 1
- 300→ 25, kernel 5, stride 1, padding 1

Each convolution layer consists of the convolution, then ReLU, then single output max pool. We concatenate the resultant features and output a fully connected layer. As an experiment, we train first using the AWS 300d embedding and second using the GloVe 6B 300d embedding. The optimizer is an Adam optimizer of learning rate .00004 in both cases. We treat the embeddings as fixed. The model is trained for 150 epochs with batch size 50.

4.5 ResNet

5 Conclusion

End the write-up with a very short recap of the main experiments and the main results. Describe any challenges you may have faced, and what could have been improved in the model.