HW2: Language Modeling

Jonah Philion
jonahphilion@college.harvard.edu
github

February 17, 2018

1 Introduction

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

$$x_{i+1} \sim \sigma(W\phi(x_1, ..., x_i) + b)$$

where $x_1, ..., x_{i+1}$ are word vectors for a sentence and σ is the softmax function. The models I try are

- trigram with linear interpolation
- NN language model
- LSTM language model

And just for fun

- Embedding $\rightarrow$ Embedding
- 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 model outputs a categorical distribution over the vocabulary $\mathcal{V}$. Embeddings $\mathcal{E}_d$ map a one hot vector x_j to a dense vector of size d.

3 Model and Algorithms

All models are trained on the Penn Treebank. For models requiring gradient descent, training loss and validation loss are recorded in real time with visdom. Final accuracy, roc score, and binary cross entropy which are displayed in the tables are calculated on the test set. These metrics are calculated using sklearn.metric.

Figure 1: 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.

4 Experiments

4.1 Naive Bayes

Word counts for positive and negative labels are all initialized to a global initial count of α. Figure ?? 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 2.

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 1 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 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 pooled. The order of the words is therefore lost during the transformation $\phi(\boldsymbol{x}_i)$ 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 Wikipedia 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.

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 with 0.5 dropout. As an experiment, we train first using the wikipedia 300d embedding and second using the GloVe 6B 300d embedding. The optimizer is an Adam optimizer of learning rate .00006 in both cases. We treat the embeddings as fixed. The model is trained for 250 epochs with batch size 50 on wikipedia word vectors, and for 350 epochs with batch size 50 on GloVe word vectors. Training and validation metrics tracked during training are shown in Figure ??. Results for the two different embeddings are shown in Table 6.

Model	Acc.	Bce.	Roc.
Wiki	0.800	0.533	0.877
GloVe	0.833	0.577	0.907

Table 6: Higher quality word vectors improve the model.

4.5 ResNet

ResNet18is fine-tuned to classify images of the text of movie reviews. This problem is strictly harder than any of the above problems in that the network has to learn how to "read" in addition to learning how to interpret.
To create the images, I use DejaVuSansMono size 14 font and make images of size (224,224), which is the smallest recommended size to feed to ResNet18. The text is white on a black background and begins at a random location in the row line of the image. Example images are shown in Figure ??.
The only modification made to the ResNet18 architecture is I change the fully convolution output layer from 512→ 1000 to 512→ 1.
The model is trained for 225 epochs with batch size 4, and Adam optimizer of learning rate 0.0001. The network memorizes the training data fairly quickly as seen in Figure ??. However, in order to memorize, the network still manages to learn features which allow it to generalize to near naive bayes performance on the validation and test sets.
The final results are shown in Table 7. Without doing more evaluation, it is not clear to me if the performance of this model is surprising or not.

Model	Acc.	Bce.	Roc.
ResNet18	0.774	0.685	0.855

Table 7: ResNet learns features straight from the image of the text which generalize to near-naive bayes accuracy on the test set.

5 Conclusion

Four standard NLP models and a fifth model which has no predefined understanding of a vocabulary are trained and evaluated on SST1. The best results from all models are shown in Figure 8.

Model	Acc.	Bce.	Roc.
NB Counts	0.793	0.686	0.867
Binarized LR	0.789	0.496	0.876
Mean CBOW	0.824	0.416	0.898
GloVe CNN	0.833	0.577	0.907
ResNet18	0.774	0.685	0.855

Table 8