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 $\to$ Embedding
- Fine-tuning ResNet classifier on images of text

2 Problem Description

Given 10 consecutive words, we would like to predict the next word. 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 Mean Average Precision (MAP) is calculated on a validation set.

3.1 Evaluation

1. All models assume a vocabulary of 10000 and have a predict function which given a batch of sentence fragments outputs an array containing the probabilities for the next word for all batches.

2. All models are tested with the same evaluation function. The function evaluates the model on batches of size 67 of bptt_len 11 of "valid.txt". The first 10 words of the batch are fed into the predict function and the 11th word is treated as ground truth. The top 20 words are then measured against the correct word with MAP.

The validation is assumed to be fairly similar to the Kaggle set as justified by the fact that the MAP-20 for the trigram model on the validation set is 0.296 and on Kaggle the model receives an MAP of 0.292. Unless explicitly mentioned, all MAP reported below are calculated with this evaluation code, not on Kaggle.

4 Experiments

4.1 Trigram

The interpolated trigram model states

$$p(y_{i+1}) = \alpha \cdot [p^{y_{i+1}}_{y_{i-1},y_i}, p^{y_{i+1}}_{y_i}, p^{y_{i+1}}]$$

Where α is a vector of length 3 with $|\alpha|_1 = 1$, and p^a_b is the probability of word a given sequence b as determined by count data in the training set.
The unary probabilties are implemented as a vector of size $\mathcal{V}$, the binary probabilties are $\mathcal{V} \times \mathcal{V}$, and the ternary probabilties are a dictionary with keys (w_1, w_2) and values vectors of size $\mathcal{V}$.
During training, all sequences of size 1,2 and 3 are counted. The counts are then normalized over $\mathcal{V}$. At test time, we for the most part return $\alpha \cdot p$. However, for sequences of length 1 not seen in training, we replace p_{binary} with p_{unary} and for sequences of length 2 not seen in training, we check if p_{binary} is in training and if so use p_{binary}, if not use p_{unary}.
To determine the best alpha, we sample randomly from vectors of length 3 that sum to 1. We do so 5000 times and use the best one. The alpha chosen is

$$\alpha = [0.4306712668382596, 0.4897915705677378, 0.07953716259400256]$$

Scores for the above and other alpha of interest are reported in Table 1.

4.2 NN Language Model

In the neural language model, we map the sparse one-hot vectors used to represent words to dense vectors of dimension d. We then concatenate the *lookback* most recent words and operate an MLP on the resultant vector of length $lookback \cdot d \to \mathcal{V}$. The model is motivated by Bengio.
We train the NN with an Adam optimizer of learning rate .0003 for five epochs on batches of size 140 and bptt_len 6. The network architecture is

Embedding $\to$Linear(D $\cdot$*lookback*, D $\cdot$*lookback*) $\to$Dropout(0.3) $\to$Linear(D $\cdot$*lookback*, $\mathcal{V}$)

α	MAP
[1,0,0]	0.271
[0,1,0]	0.260
[0,0,1]	0.124
[1/3,1/3,1/3]	0.294
[0.43,0.49,0.08]	0.296

Table 1: The first entry of α corresponds to ternary counts, and the third entry cooresponds to unary counts. The binary and ternary counts both contain considerably more information than unary.

lookback	MAP
3	0.257
4	0.255
5	0.250
Trigram + NN	0.300

Table 2: The NN does about the same on its own independent of *lookback*. The ensemble with Trigram waited by 0.643 and NN weighted 1-0.643 achieves higher accuracy than the Trigram model on its own.

where D is the dimension of the embeddings, and *lookback* is the number of words that are concatenated. The MAP is recorded in Table 2 for a lookback of 3, 4, and 5 all with $D = 300$. Finally, the output of the best Trigram model is linearly interpolated with the best NN model.

The word vectors learned by this procedure are not particularly good. Closest vectors as measured by cosine distance are below

good: ['good', 'shea', 'opted', 'marine', 'warren', 'pcs', ”'m”, 'affiliate', 'owner', 'appearances']

bad: ['bad', 'mexico', 'accurate', 'banco', 'rolled', 'acknowledge', 'market', 'kemp', 'federated', 'greed']

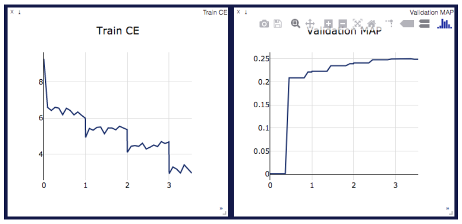

Figure 1: Training the NN with *lookback* = 3

input			
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 3

4.3 LSTM language model

4.4 Embedding $\rightarrow$ Embedding

4.5 ResNet

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 4.

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 4