6.864 HW 1 Solutions

Faraaz Nadeem
Massachusetts Institute of Technology
faraaz@mit.edu

Abstract

In this homework, we implement a few different models for learning word representations. These models include bag of words, SVD, TFIDF, Word2Vec, and HMMs. For each implementation, we use methods such as clustering and nearest neighbors to visualize the learned representations. Finally, we train a logistic regression classifier to compare the relative performance of each representation on the task of sentiment classification. Some hyper-parameters such as model size and training data are varied to provide analysis on the importance of these features.

1 Matrix Factorization

For the LSA implementation, we used the TruncatedSVD class available in the `sklearn` package.

1.1 Nearest Neighbors in Representation Space

For LSA with and without a TF-IDF transform, with a representation size of 64, the nearest neighbors of for `good` are positive (`pretty`, `liked`) but the nearest neighbors for `bad` are more generic (`ok`, `either`). Representations seem to carry limited sentiment information.

LSA learned reasonable representations for nouns (`dog` → `pet`, `dogs`, `food`; but `jelly` → `online`, `low`, `hoping`). The nearest neighbors are some combination of synonyms and closely associated words.

TFIDF representations for `good` and `bad` were worse than representations with unnormalized counts, but noun representations looked slightly better (`cookie` → `gluten`, `flour`, `free`; `jelly` → `mixing`, `save`, ... `creamer`).

Both LSA and TFIDF learned to associate numbers with other numbers.

1.2 LSA representation Size

Reducing the LSA representation to 10 led to nearest neighbors with higher variance in semantic meaning. Some words' nearest neighbors didn't make sense at all. This behavior is expected because it is harder to encode semantic similarity with fewer features.

Nearest neighbors for LSA rep size 10:

- `good` → `better, like`
- `bad` → `taste, like, better`
- `dog` → `dogs, vet, food`
- `jelly` → `freezer, ate, unable`
- `4` → `1, 2, br`

Nearest neighbors for LSA+TFIDF rep size 10:

- `good` → `like, too, come`
- `bad` → `me, nor, lobster`
- `dog` → `dogs, we, pet`
- `jelly` → `mess, touch, needed`
- `4` → `6, 1, 11`

1.3 Singular Vectors

Given that

$$W_{td} = U\Sigma V^T$$

and U, V are orthogonal matrices, we can infer

$$W_{tt} = W_{td}W_{td}^T = U\Sigma V^T V\Sigma^T U^T = U\Sigma^2 U^T$$

U contains both the left singular vectors of W_{td} and W_{tt}. From this, we would expect that the word embedding similarity should not change between the co-occurrence matrix and term-frequency. Empirically, however, the nearest neighbors end up different across the two matrices. This happens because the SVD is not unique and the implementation used here is only approximate.

examples	features	accuracy
20	word	**0.526**
	lsa	0.524
	combo	**0.526**
100	word	0.616
	lsa	0.604
	combo	**0.626**
1000	word	**0.784**
	lsa	0.678
	combo	**0.784**
3000	word	0.784
	lsa	0.738
	combo	**0.794**

Table 1: Classification Accuracy Repr. Size 64

repr size	features	accuracy
2	lsa	0.544
	combo	**0.784**
4	lsa	0.610
	combo	**0.790**
8	lsa	0.644
	combo	**0.794**
16	lsa	0.686
	combo	**0.796**
32	lsa	0.738
	combo	**0.794**

Table 2: Classification Accuracy 3000 Examples

1.4 Review Classification

From Table 1, we see that the word features performed better than LSA, but a combination of both word features and LSA features sometimes gives small improvements (on the order of 1%). The comparatively poor performance of LSA features alone is consistent with our observation above that representations words like `bad` don't necessarily contain strong signals about negative sentiment. It is important to give models the ability to learn specific facts about individual words. Performance increased as we increased the number of training examples.

1.5 Effect of Representation Size

From table 2, we see there is a trend towards increasing accuracy with larger representation size. However, after a certain point this benefit plateaus (quickly for the combination featureizer, and somewhat more slowly for the LSA-only featurizer). We could infer that the model begins to overfit when the representation size is too large, and does not have enough meaningful features to perform well when the representation small is too small.

2 Word2Vec

In our implementation, the model consists of an embedding layer, followed by 2 linear layers with a ReLU between them. We used a CrossEntropyLoss (equivalent to a log softmax followed by NLLLoss) to implement maximum likelihood training.

2.1 Nearest Neighbors in the Rep Space

Nearest neighbors for context size 2, embed dimension 32, train epochs 300, batch size 1024, hidden size 64:

- `good` $\rightarrow$ `great, decent, awesome, healthy, bad`
- `bad` $\rightarrow$ `good, bitter, supposed, overwhelming, overpowering`
- `cookie` $\rightarrow$ `inside, frosting, square, berry, mug`
- `jelly` $\rightarrow$ `peas, earth's, grape, were, kinds`
- `dog` $\rightarrow$ `cat, cats, baby, life`
- `the` $\rightarrow$ `their, an, amazon, your`
- `4` $\rightarrow$ `2, 6, 3, 5, 24`

Most of the nearest neighbors match what I would intuitively expect to see. An interesting observation is that the types of nearest neighbors that arise are also dependent on the source word. This indicates that different types of relationships are being encoded by word2vec.

For `good` and `bad`, we get words that are synonyms (ex. `good` $\rightarrow$ `great`), words that are antonyms (ex. `good` $\rightarrow$ `bad` and vice versa), and food related terms that also convey sentiment (ex. `good` $\rightarrow$ `healthy` and `bad` $\rightarrow$ `bitter`).

For `cookie`, we get food items that are related to `cookie` but not necessarily synonyms or antonyms (ex. `frosting`, `mug`). `jelly` is a less frequent word in the dataset so its nearest neighbors make the least amount of sense. However, we still see that `grape` is a neighbor that makes sense as describing a type of jelly. `dog` neighbors consist of other living beings. `the` is interesting because its nearest neighbors are other "structural" words such as articles and pronouns. The number `4` simply gives neighbors that are other numbers.

When we look at the clusters of words generated by the clustering algorithm, we can identify more patterns in the embedding space that align with our observations above.

1. `right, cilantro, days, date, times, sale, chemicals, review, meals, mixes` [some words related to time]
2. `once, maybe, these, absolutely, table, wine, way, nutrition, sorry, kettle, obviously, worse, greatest, bottles, seriously` [adverbs, others]
3. `pineapple, peanuts, jar, stevia, donut` [pattern unclear]
4. `appeal, buy, happen, consume, feed, work, use, serve, 'open', bring` [present tense verbs]
5. `tiny, 40, 95, half, rica, spoon, square, disposable, harmony, several, couple, reduced, lb, various, bodied, filter` [units of measurement]
6. `goes, has, i'll, into, eats, needs, wouldn't, you'd, we'll` [some verbs of desire or intent]
7. `figured, iron, decided, learned, served, thrilled, concerned` [past tense verbs]
8. `carbohydrate, body, gluten, coke, adult, eggs, fat` [nutrition related words]
9. `strange, bad, perfectly, german, healthier, quite, fresh, thick, incredible, pretty, range, weak` [adjectives]
10. `lemon, cake, salads, canned, 150, flavored` [pattern unclear]

2.2 Context Size Variation

I ran the training using context sizes of 1 and 6.

For commonly occurring words with local semantic relationships (ex. adjectives such as `good` and `bad`, and numbers such as `4`), the representations were mostly similar to the ones produced by context size 2 above. It makes sense that adjectives still have a good representation with context size 1, because they often describe the noun that directly follows.

For less commonly occurring words with medium distance semantic relationships (ex. nouns such as `cookie`, `jelly`, `dog`), the representations made less sense under both context sizes 1 and 6.

With context size 1, `cookie` → `alive, green, count`; `jelly` → `homemade, process, spots`; `dog` → `supply, cat, jack`. Nouns are often associated as direct/indirect objects relative to verbs, and many times the corresponding verb is not directly before or after the noun. So for context size 1, it makes sense that the representation of nouns is lower quality.

With context size 6, `cookie` → `cereal, liquid, http`; `jelly` → `soaked, grinder, house`; `dog` → `cat, guests, blue`. There are 12 words contributing to the representation of the noun in this case, which is often greater than the length of the sentence encompassing the word.

examples	accuracy
10	0.498
100	0.588
1000	0.716
3000	0.726

Table 3: Word2vec train examples vs accuracy.

2.3 Downstream Classification

The word2vec results are slightly worse, which was initially surprising because I found the embedding representations to be semantically more accurate than in part 1. One explanation for this result is that the word2vec embeddings model a lot of relationships (ex. verb tenses, units of measurement, parts of speech, etc.) that are not important to the sentiment classification task.

2.4 Learned Embedding Tradeoffs

Advantages: the learned embeddings capture more complicated trends in the embedding space, as shown by the result of clustering above. The nearest neighbors produced by word2vec have much better semantic similarity. Representations of infrequent words are also reinforced by the surrounding context words during training.

Disadvantages: it is hard to get interpretability in the word2vec model. Why is 'their' closer to 'the' than 'an'? What do the individual components of the embedding space represent? The embedding space is also created in a non deterministic fashion, so relationships established between words in one session of training may be different from another. These inconsistencies make it difficult to debug downstream tasks, especially because the high dimensionality of the embedding space is hard to visualize.

2.5 Embedding Averaging Problems

There are a few potential problems.

1. We make the assumption that each word should be given an equal weight when contributing to the overall sentence embedding. This could lead to commonly occurring words dominating sentence representations, despite having low importance. 2. We lose the grammatical structure / ordering of words in the sentence. This could be bad if sentence structure is an important feature

for predicting sentiment (ex. if poor structure indicates negative sentiment).

3 Hidden Markov Models

We used log scaling to ensure numerical stability, and a combination of numpy vector operations / for loops to compute matrices for the algorithm.

3.1 Hidden State Encoding Analysis

3.1.1 2 states

State 0 seems to have a lot more grammatical terms (stopwords, punctuation, etc.) Meanwhile, state 1 has more nouns and adjectives. When generating sentences, the model does a good job of alternating between the "structural" and "noun/adj" states, but the sentences that were generated are still incoherent.

Generated sentence: `salty br good drink but of much stars also one`

State 0 most probable words:

`<unk>, '.', the, ',', and, a, i, to, it, of, is, in, this, that, but, not, '!', are, my`

State 1 most probable words:

`'.', br, i, for, ',', this, <unk>, taste, them, good, product, like, of, so, these, don't, love, now, mix, price`

3.1.2 5 states

State 0 unique probable words include `coffee, love, product`. State 1 unique probable words include `no, food, taste`. State 2 consists of general stopwords and grammatical terms. State 3 consists of more conjunctions and grammatical terms. State 4 unique probable words include `chips, potato, drink, sugar, price, better, best`.

Generated sentence: `expect me to leave a you excellent about the my`

3.1.3 10 states

Stopwords and punctuation occur frequently in all of the states. It was difficult to determine if some states had different higher level semantic meaning than others. Examples of terms that occurred in the top 20 terms of one hidden state but not others: `coffee, orange, amazon, product, chips`.

states	examples	accuracy
2	10	0.448
	100	0.476
	1000	0.516
	3000	**0.548**
5	10	0.476
	100	0.476
	1000	0.556
	3000	**0.548**
10	10	0.548
	100	0.476
	1000	0.508
	3000	**0.570**

Table 4: Classification accuracy while varying the number of states and training examples

Generated sentence: `advertised fructose replacement cups easy has as br was amazon`

3.2 Train Size vs Classification Accuracy

From table 4, we see there is generally an upward trend when adding more labeled examples. Although the hidden state distributions seem more coherent with 2 and 5 hidden states, using 10 hidden states achieves similar and slightly better performance.

Overall, the learned HMM representations are not as good as representations learned in parts 1 and 2. This could be for a number of reasons. First, we used a smaller representation size (10 in HMMs vs 32 in word2vec). Second, the model structure of HMMs lends itself to encoding elements of sentence structure in the hidden states (stopwords, nouns/adj, etc). This is less useful for the task of sentiment prediction, where the sentiment of individual words is strongly correlated with the sentiment of the review as a whole.

Ideally the HMM model would learn hidden states with word sentiment. Indeed, the improved performance with a larger number of hidden states seems to indicate more of this focus on words. But we did not get any nice sentiment clusters with the limited data and computation time available to us. Performance improvements are also non-monotonic, possibly due to bad random initializations for particular #state/#example combinations.