MIT 6.806/6.864: Advanced Natural Language Processing **Fall 2021**

Homework 2

Due Date: Tues. Oct 12, 2021, 11:59 PM EDT (*Canvas submission*)

Word Representations

Introduction. In this assignment, you'll explore different ways of using unlabeled text data to learn pretrained word representations. For each of the representation learning schemes, you'll implement, train, and evaluate the scheme on a real dataset, then prove and discuss some theoretical properties. Finally, you'll submit a report describing both experiments and answers to the theoretical and computational questions in this handout. Your report should also describe the effects of different modeling decisions (representation learning objective, context size, etc.) on both qualitative properties of learned representations and their effect on a downstream prediction problem. We provide scaffolding code for each part of the lab in `6864_hw2_fa21.ipynb`, which is available here: http://colab.research.google.com/github/mit-6864/hw2/blob/main/6864_hw2_fa21.ipynb.

General report guidelines. Homework assignments should be submitted in the form of a research report on Canvas. Please upload a single PDF of your report concatenated with print outs of your code (e.g., the Jupyter Notebooks with your implementation). The report section of your submitted PDF should consist of a maximum of four single-spaced pages (6.806) or six single-spaced pages (6.864) typeset with LaTeX.[1] Reports should have one section for each part of the assignment below. Each section should describe the details of your code implementation and include whatever analysis and figures are necessary to answer the corresponding set of questions.

Part 1: Matrix Factorization

The first part of your lab report should discuss any important details and design decisions you made when implementing the LSA featurizer. Additionally, you should design and conduct experiments to answer the following questions in your report:

(a) (*Theoretical, Computational*) Recall that the we can compute the word co-occurrence matrix $W_{tt} = W_{td}W_{td}^{\top}$. What can you prove about the relationship between the left singular vectors of W_{td} and W_{tt}? Do you observe this behavior with your implementation of `learn_reps_lsa`? Why or why not?

(b) (*Experimental*) Qualitatively, what do you observe about nearest neighbors in representation space? For example, what words are most similar to *the*, *dog*, *3*, and *good*? How does the size of the LSA representation affect this behavior?

[1] If you'd like, you can use the Association for Computational Linguistics style files, available at: https://2021.aclweb.org/downloads/acl-ijcnlp2021-templates.zip

(c) (**6.864 students only**; *Experimental*) Do learned representations help with the review classification problem? What is the relationship between the number of labeled examples used for logistic regression and the effect of word embeddings?

Part 2: Language Modeling

Part 2 of your lab report should discuss any implementation details that were important to filling out the code above. Then, use the code to set up experiments that answer the following questions:

(a) (*Experimental*) Qualitatively, what do you observe about nearest neighbors in representation space? (E.g. what words are most similar to *the*, *dog*, *3*, and *good*?) How well do Word2Vec representations correspond to your intuitions about word similarity?

(b) (*Experimental*) How do results on the downstream classification problem compare to Part 1?

(c) (**6.864 students only**; *Experimental*) One important parameter in Word2Vec-style models is context size. How does changing the context size affect the kinds of representations that are learned?

Part 3: Improving the Model (6.864 students only)

In Part 3, you will extend the methods you've implemented in Parts 1 and 2 with the goal of improving final predictive performance. You should experiment with at least one idea to improve the model — feel free to focus on either the featurizer or the classifier. Some suggestions of things you could try:

- Implement a different TD matrix normalization method (see lecture slides for alternatives to TF-IDF).
- Implement a different Word2Vec formulation (in Part 2, you implemented the CBOW formulation; does the skip-gram formulation perform any better?).
- Implement a more sophisticated classifier module.
- Tune featurizer and/or classifier hyperparameters (for full marks, you should obtain at least a 1% improvement in prediction accuracy if you only tune hyperparameters).

In your report, you should discuss what you implemented (including relevant design decisions), and how your change(s) impacted performance. Note: As long as you try something with difficulty comparable to the suggested modifications and have a meaningful discussion of your results in your report, you can earn full marks (you do not necessarily need to improve performance).

Part 4: Skip-Gram Variant (6.864 students only)

Recall from lecture *skip-gram with negative sampling (SGNS)*, another technique for learning word embeddings. While the classic skip-gram learns to predict words from average context vectors, the SGNS model learns to classify whether a given context-word pair is likely to occur or not.

Let's break down the SGNS classification problem. When trained on a corpus C, the SGNS model sees two types of examples:

1. *Positive samples* refer to the word-context pairs that do occur in C. Let D be the collection of all word-context pairs that appear in C. Suppose $E_{w,c}$ is the indicator random variable for whether a word-context pair (w, c) appears in C. For a positive sample (w, c), the SGNS model attempts to maximize

$$P[E_{w,c} = 1|(w, c)] = P[(w, c) \in D|(w, c)] = \sigma(\overrightarrow{w} \cdot \overrightarrow{c}) = \frac{1}{1 + e^{-\overrightarrow{w} \cdot \overrightarrow{c}}}$$

2. *Negative samples* are any examples that do not occur in C. For a negative sample word-context pair (w, c), the SGNS model aims to maximize

$$P[E_{w,c} = 0|(w, c)] = P[(w, c) \notin D|(w, c)] = 1 - P[(w, c) \in D|(w, c)]$$

   One question you may ask is: how do we generate negative samples? For each (w, c) in the space given by $|V_w|\times|V_c|$, we generate k negative samples (w, c_N), where the context c_N is sampled from a unigram distribution. Note that k is a hyperparameter that we choose.

We now have what we need to write the SGNS objective function. Suppose that the word embedding vector and context embedding vector learned by the SGNS model for word w and context c are $\overrightarrow{w}$ and $\overrightarrow{c}$, respectively. The SGNS objective across all word-context pairs (w, c) is given by

$$\ell = \sum_{w \in V_w} \sum_{c \in V_c} \#(w, c) \cdot (\sum \log \sigma(\overrightarrow{w} \cdot \overrightarrow{c}) + k \cdot \mathbb{E}_{c_N \sim P_D}[\log \sigma(-\overrightarrow{w} \cdot \overrightarrow{c_N})])$$

To better understand the SGNS model, let's focus in on the objective function specific to a single word-context pair (w, c). The argument of the double summation in the objective function ℓ given above, we can get the SGNS model's local objective function, or the objective function specific to a single word-context pair (w, c).

$$\ell(w, c) = \#(w, c) \cdot \log \sigma(\overrightarrow{w} \cdot \overrightarrow{c}) + k \cdot \#(w) \cdot \frac{\#(c)}{|D|} \cdot \log \sigma(-\overrightarrow{w} \cdot \overrightarrow{c})$$

We will work toward finding an expression for $\overrightarrow{w} \cdot \overrightarrow{c}$ that maximizes the above local objective.

(a) (**6.864 students only**; *Theoretical*) Find an expression for $\frac{\partial \log \sigma(x)}{\partial x}$.

(b) (**6.864 students only**; *Theoretical*) Find the critical point of $\overrightarrow{w} \cdot \overrightarrow{c}$ for the local objective function $\ell(w, c)$.

Hint: Take the derivative of ℓ and set it equal to 0. After some simplification, you should find yourself left with a quadratic equation of the form

$$e^{2\cdot\overrightarrow{w}\cdot\overrightarrow{c}} - \left(\frac{\#(w,c)}{\lambda} - 1\right)e^{\overrightarrow{w}\cdot\overrightarrow{c}} - \frac{\#(w,c)}{\lambda} = 0$$

where $\lambda = k \cdot \#(w) \cdot \frac{\#(c)}{|D|}$.

(c) (**6.864 students only**; *Theoretical*) How is your answer from part (c) related to the word-context point-wise information (PMI) matrix? How do the embedding vectors $\overrightarrow{w}$ and $\overrightarrow{c}$ that the SGNS model learns relate to the word-context PMI matrix?

Recall that the PMI for a word-context pair (w, c) is given by

$$PMI(w,c) = \log\left(\frac{\#(w,c)}{\#(w)\cdot\#(c)} \cdot |D|\right)$$

The word-context PMI matrix is a $V_w \times V_c$ matrix whose value at entry (w, c) is $PMI(w, c)$.