Ling 282/482 hw2

Due 11PM on September 23, 2024

In this assignment, you will answer some written questions about and then implement word2vec; in particular, the method *skip-gram with negative sampling (SGNS)*. By doing so you will:

- Count parameters
- Take derivatives of a loss
- Translate mathematics into implemented code
- Train your own set of word vectors and briefly analyze them

I **strongly recommend** doing this assignment in order. Your answers in the written portion will make your implementation much easier, especially for the gradient computations.

Submission Instructions

This assignment contains both written and programming portions. The answers to written questions must be submitted in a *.txt or *.pdf file to Blackboard. You will receive an invitation link to complete the programming portion via Github Classroom. This will open a Github repository with starter code and missing portions for you to complete. When you are finished with implementation, simply commit and push your changes to the online repository that was created for you. Unless you request otherwise, I will grade your work **as of the most recent commit** in your repository, subject to any applicable late penalties.

Github Classroom & Codespaces

Github Classroom and Codespaces were introduced during class, but feel free to reach out to me with any questions! Briefly, when you click "accept assignment", it will create the repository (including starter code) in which you will complete the assignment. You can open this repository in Github Codespaces — a virtual IDE in which you can edit and run your code. You are also free to clone the repository to your local machine and work on the assignment there, but your code **must run** and **will be graded** in the Codespace environment. An Anaconda Python environment is set up for you within Codespaces, and should be activated with `conda activate ling482` before running code.

1 Understanding Word2Vec [30 pts]

Q1: Parameters [2 pts] How many parameters are there in the SGNS model? Write your answer in terms of V (the vocabulary) and d_e, the embedding dimension. (Hint: one parameter is *a single real number.*)

Q2: Sigmoid [8 pts] Sigmoid is the logistic curve $\sigma(x) = \frac{1}{1+e^{-x}}$.

- What is the range of $\sigma(x)$? [2 pts]
- How is it used in the SGNS model? [2 pts]
- Compute $\frac{d\sigma}{dx}$; show your work. (Hint: write your final answer in terms of $\sigma(x)$.) [4pts]

Q3: Loss function's gradients [20 pts] In the slides for lecture 3, we saw that the total loss for one positive example and k negative examples is given by:

$$L_{CE} = -\log P(1|w, c_+) - \sum_{i=1}^{k} \log P(0|w, c_{-i})$$

In what follows, where x is a vector and f a function of x and possibly more variables, we will define $\nabla_x f := \langle \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \rangle$.

- Rewrite this loss in terms of the parameter matrices E and C (i.e. replace the $P(\cdot)$s with the definition of the model). [2 pts]

  Use w as the integer index of the target word, c_+ as the integer index of the positive context word, and c_{-i} as the integer index of the ith negative sampled context word.
- Using the chain rule, compute $\frac{d}{dx} - \log \sigma(x)$. (Hint: first, show that $\sigma(x) = \frac{e^x}{e^x+1}$. Note: log here is the natural logarithm, i.e. logarithm with base e.) [4 pts]
- Show that $\nabla_x x \cdot y = y$ (where $x \cdot y$ is the dot product of two vectors). [2 pts]
- Compute (and show your work) $\nabla_{C_{c_+}} L_{CE}$. [4 pts]
- Compute (and show your work) $\nabla_{C_{c_{-i}}} L_{CE}$. [4 pts]
- Compute (and show your work) $\nabla_{E_w} L_{CE}$. [4 pts]

2 Implementing Word2Vec [36 pts]

Before getting started, a few notes on the implementation:

- Always start with small data! To test various components of the pipeline, you can use the toy files in the `data` folder.
- All files referenced here are available in your starter code.
- The main training loop is at the bottom of `word2vec.py`. You do not have to touch this, but can read it to see how the various components you implement are being used.
- `word2vec.py` takes a variety of arguments, which are specified in `util.py`. If you do not specify these arguments, the default values will be used.
- Remember to run `conda activate ling482` before running any Python code!

Q1: Data generation [12 pts] In `data.py`

- Implement `get_positive_samples`, which generates positive examples from a list of tokens. [8 pts]

- Implement `negative_samples`, which samples negative context words. (Hint: `random.choices` is your friend, and pay attention to how `negatives_from_positive` works.) [4 pts]

Q2: Model computation [8 pts] In `word2vec.py`

- Implement `SGNS.forward`. This represents one "forward pass" of the skip-gram with negative sampling model, i.e. this computes $P(1|w, c)$. Note: use `self.embeddings` and `self.context_embeddings`, which are defined in `__init__`.

Q3: Gradient computation [8 pts] In `word2vec.py`, implement the following methods

- `get_positive_context_gradient`: this computes $\nabla_{C_{c_+}} L_{CE}$.
- `get_negative_context_gradients`: this computes the list of $\nabla_{C_{c_{-i}}} L_{CE}$ for each negative context word c_{-i}.
- `get_target_word_gradient`: this computes $\nabla_{E_w} L_{CE}$.

Q4: Train word vectors [8 pts] Run the main training loop by calling `word2vec.py` with the following command-line arguments (defined in `util.py`):

- 4 epochs
- Embedding dimension: 16
- Learning rate: 0.2
- Minimum frequency (for inclusion in vocabulary): 4
- Number of negative samples: 4
- Save vectors to a file called vectors.tsv

After that, run `python analysis.py --save_vectors vectors.tsv --save_plot vectors.png`. This will take your saved vectors and produce a plot with the vectors (after using PCA to reduce dimensionality to 2) of a select choice of words. In your readme file, please include:

- The total run-time of your training loop. This will be printed by the main script.
- The generated plot.
- Describe in 2-3 sentences any trends that you see in these embeddings.

3 Understanding Computation Graphs [24 pts]

Q1: Worked example Consider the function $f(x) = x^2 \times cx$.

- Draw a computaton graph for this expression. [4 pts]
- How many nodes are there (including input and output)? [2 pts]
- For $x = 2$ and $c = 3$: [12 pts]

  - Compute the value of each node in a forward pass.
  - Compute $\frac{df}{dn}$ for each node n, using backpropagation.

- Consider the node corresponding to x^2 in the graph. For each of the following, write a symbolic expression and the numerical value (at $x = 2$, $c = 3$) for: [6 pts]
  - The upstream derivative.
  - The local derivative.
  - The downstream derivative(s).