LING 575K HW2

Due 11PM on Apr 15, 2021

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

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

1 Understanding Word2Vec

Q1: Parameters 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 Sigmoid is the logistic curve $\sigma(x) = \frac{1}{1+e^{-x}}$.

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

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

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

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).

  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 $\sigma(x) = \frac{e^x}{e^x+1}$.]

- Show that $\nabla_x x \cdot y = y$ (where $x \cdot y$ is the dot product of two vectors).
- Compute $\nabla_{C_{c_+}} L_{CE}$.
- Compute $\nabla_{C_{c_{-i}}} L_{CE}$.
- Compute $\nabla_{E_w} L_{CE}$.

2 Implementing Word2Vec

Q1: Data generation In `data.py`

- Implement `get_positive_samples`, which generates positive examples from a list of tokens.
- Implement `negative_samples`, which samples negative context words. [Hint: `random.choices` is your friend.]

Q2: Model computation 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 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 Run the main training loop by calling `word2vec.py` with the following command-line arguments (defined in `util.py`):

- 20 epochs
- Save vectors to a file called vectors.tsv
- Embedding dimension: 15
- Learning rate: 0.25
- Minimum frequency: 5

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.

Submission Instructions

In your submission, include the following:

- readme.(txt|pdf) that includes your answers to §1 as well as Q4 of §2.
- `hw2.tar.gz` containing:
  - –