# LING 574 HW2

Due 11:59PM on Apr 17, 2025

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 [30 pts]

**Q1: Parameters [3 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 [7 pts]** Sigmoid is the logistic curve $\sigma(x) = \frac{1}{1+e^{-x}}$.

- What is the range of $\sigma(x)$? [1 pt]
- 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 $\nabla_x f := \langle \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \rangle$. You can assume here and throughout, unless otherwise specified, that log refers to the natural logarithm (i.e. base $e$).

- Rewrite this loss in terms of the parameter matrices $W$ and $C$, i.e. expand 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 $i$th negative sampled context word.

- Using the chain rule, compute $\frac{d}{dx}(-\log\sigma(x))$. [Hint: you can also write this in terms of $\sigma(x)$, using your answer from Q2.] [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) $\nabla_{W_w} L_{CE}$. [4 pts]

# 2 Implementing Word2Vec [45 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 `/mnt/dropbox/24-25/574/data/`.
- All files referenced here are in `/mnt/dropbox/24-25/574/hw2` on patas.
- 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.
- This implementation uses a Vocabulary class, as implemented in HW1. We will make a reference implementation available for use on Monday morning (after the late submission deadline); until then, you can use your own, by placing `vocabulary.py` in the same directory as your copy of the files for this assignment.

**Q1: Data generation [10 pts]** In `data.py`

- Implement `get_positive_samples`, which generates positive examples from a list of tokens. [7 pts]
- Implement `negative_samples`, which samples negative context words. [Hint: `random.choices` is your friend.] [3 pts]

**Q2: Model computation [10 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 [15 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 $\nabla_{W_w} L_{CE}$.

**Q4: Train word vectors [10 pts]** Run the main training loop by calling `word2vec.py` with the following command-line arguments (defined in `util.py`):

- 15 epochs
- Save vectors to a file called vectors.tsv
- Embedding dimension: 15
- Learning rate: 0.2
- Minimum frequency: 5
- Number of negative samples: 15

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.

**Testing your code** In the dropbox folder for this assignment, we have included a file `test_all.py` with a few very simple unit tests for the methods that you need to implement. You can verify that your code passes the tests by running `pytest` from your code's directory, with the course's conda environment activated. **N.B.:** these are very small and minimal tests; passing them is ncessary for your code to be correct, but not necessarily sufficient. So you will still want to reason about your own code and/or do more testing to convince yourself that it is doing the right thing.

## 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:
  - run_hw2.sh
  - word2vec.py
  - data.py