Assignment #2: word2vec (20 Points)

Due on Saturday March 14, 2026 by **11:59 pm**

You need to implement the word2vec model and train your own word vectors with stochastic gradient descent (SGD). Before you begin, first run the following commands within the assignment directory in order to create the appropriate conda virtual environment. This guarantees that you have all the necessary packages to complete the assignment. **Windows users** may wish to install the Linux Windows Subsystem[1]. Also note that you probably want to finish the previous math section before writing the code since you will be asked to implement the math functions in Python. You'll probably want to implement and test each part of this section in order, since the questions are cumulative.

```
conda env create -f env.yml
conda activate a2
```

Once you are done with the assignment you can deactivate this environment by running:

```
conda deactivate
```

For each of the methods you need to implement, we included approximately how many lines of code our solution has in the code comments. These numbers are included to guide you. You don't have to stick to them, you can write shorter or longer code as you wish. If you think your implementation is significantly longer than ours, it is a signal that there are some `numpy` methods you could utilize to make your code both shorter and faster. `for` loops in Python take a long time to complete when used over large arrays, so we expect you to utilize `numpy` methods.
Note: If you are using Windows and have trouble running the .sh scripts used in this part, we recommend trying Gow or manually running commands in the scripts.

(a) (12 points) We will start by implementing methods in `word2vec.py`. You can test a particular method by running `python word2vec.py m` where `m` is the method you would like to test. For example, you can test the sigmoid method by running `python word2vec.py sigmoid`.

  (i) Implement the `sigmoid` method, which takes in a vector and applies the sigmoid function to it.

  (ii) Implement the softmax loss and gradient in the `naiveSoftmaxLossAndGradient` method.

  (iii) Implement the negative sampling loss and gradient in the `negSamplingLossAndGradient` method.

  (iv) Implement the skip-gram model in the `skipgram` method.

  When you are done, test your entire implementation by running `python word2vec.py`.

(b) (5 points) Complete the implementation for your SGD optimizer in the `sgd` method of `sgd.py`. Test your implementation by running `python sgd.py`.

(c) (3 points) Show time! Now we are going to load some real data and train word vectors with everything you just implemented! We are going to use the Stanford Sentiment Treebank (SST) dataset to train word vectors, and later apply them to a simple sentiment analysis task. You will need to fetch the datasets first. To do this, run `sh get_datasets.sh`. There is no additional code to write for this part; just run `python run.py`.

  Note: The training process may take a long time depending on the efficiency of your implementation and the compute power of your machine ***(an efficient implementation takes one to two hours)****. Plan accordingly!*

[1]https://techcommunity.microsoft.com/t5/windows-11/how-to-install-the-linux-windows-subsystem-in-windows-11/m-p/2701207

After 40,000 iterations, the script will finish and a visualization for your word vectors will appear. It will also be saved as `word_vectors.png` in your project directory. **Include the plot in your homework write up.** In at most three sentences, briefly explain what you see in the plot. This may include, but is not limited to, observations on clusters and words that you expect to cluster but do not.