Ling 575j hw3

Due 11pm on April 20, 2023

In this assignment, you will

- Develop your understanding of computation graphs by doing a worked example by hand and then
- Implementing skip-gram with negative sampling in the edugrad library.
- Using some Operations that you will implement using the forward/backward API.

1 Understanding Computation Graphs [20 pts]

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

- Draw a computaton graph for this expression. [5 pts]
- How many nodes are there (including input and output)? [2 pts]
- For $x = 2$ and $c = 3$: [10 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: [3 pts]
  - The upstream derivative.
  - The local derivative.
  - The downstream derivative(s).

2 Implementing Word2Vec in Edugrad [55 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 `/dropbox/22-23/575j/data/`.
- All files referenced here are in `/dropbox/22-23/575j/hw3` 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 homework relies on `data.py` from HW2. 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 it in the same directory as your copy of the files for this assignment or by using a symbolic link.

Q1: Basic Operations In `ops.py`, implement the forward and backward methods of the following operations: [24 pts]

- `log`
- `sigmoid`
- `multiply`. This is *element-wise* multiplication of two matrices.

Recall: (i) you can use the list `ctx` in forward to store any values that you need when computing gradients in `backward`; (ii) backward needs to return a *list* of the same size as the number of inputs to forward; each element of the list contains the gradient for the respective input. We have provided shell lists for this purpose.

Q2: Model and BCE Loss In `word2vec.py` [24 pts]

- Implement `dot_product_rows`. Read the docstring for specification and hints.
- Implement `Word2Vec.forward`. This represents one "forward pass" of the skip-gram with negative sampling model, i.e. this computes $P(1|w, c)$ for a batch of inputs. You should use dot_product_rows here.
- Implement `bce_loss`.

Q3: Train word vectors Run the main training loop by calling `word2vec.py` with the following command-line arguments (defined in `util.py`): [7 pts]

- 6 epochs
- Embedding dimension: 15
- Learning rate: 0.2
- Minimum frequency: 5
- Number of negative samples: 15

In your readme file, please include:

- The total run-time of your training loop.
- The per-epoch training loss.

These will be printed by the main script.

Testing your code In the dropbox folder for this assignment, there is 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.

Submission Instructions

In your submission, include the following:

- readme.(txt|pdf) that includes your answers to §1 as well as Q3 of §2.
- `hw3.tar.gz` containing:
  - run_hw3.sh. This should contain the code for activating the conda environment and your command for Q3 above. You can use run_hw2.sh from the previous assignment as a template.
  - word2vec.py
  - ops.py