LING 575K HW3

Due 11PM on Apr 23, 2021

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

Q1: Worked example Consider the function $f(x) = x^2 \times 3x$.

- Draw a computaton graph for this expression.

- How many nodes are there? [Including input and output.]

- For $x = 2$:

  - 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 $x = 2$) for:

  - The upstream derivative.
  - The local derivative.
  - The downstream derivative(s).

2 Implementing Word2Vec in Edugrad

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/20-21/575k/data/`.

- All files referenced here are in `/dropbox/20-21/575k/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:

- `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 eleemnt 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`

- Implement `dot_product_rows`. Read the docstring for specification and hints.

- Implement `SGNS.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`):

- 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, we will include 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
  - word2vec.py
  - ops.py