Assignment 2: Neural Networks

Instructor: Noah Smith *CSE P517 – Spring 2025*

Due at 11:59pm PT, April 28, 2025.
70 points

In this assignment, you will learn how to design, train, and evaluate feed-forward neural networks from scratch in PyTorch to solve sentiment analysis and social commonsense reasoning problems.

You will submit both your **code** and **writeup** (as PDF) via Gradescope. Remember to **specify your collaborators** (including AI tools like ChatGPT) and **how they contribute** to the completion of your assignment at the beginning of your writeup. If you work on the assignment independently, please specify so, too.

Required Deliverables

- **Code Notebook**: The assignment is associated with Jupyter notebook (CSE447_Assignment2.ipynb). Please download the notebook as Jupyter notebook files (`.ipynb`) and submit them in Gradescope. On Google Colab you can do so by `File` → `Download` → `Download .py`. **Please comment out any additional code you had written to solve the write-up exercises before submitting on gradescope to avoid timeouts.**
- **Write-up**: For written answers and open-ended reports, produce a single PDF and submit it in Gradescope. We recommend using Overleaf to typeset your answers in LaTeX, but other legible typed formats are acceptable. We do not accept hand-written solutions because grading hand-written reports is incredibly challenging.
- **NLU Exercise** For question 4, along with the write-up, also submit the Python script (.py) file with your code used for defining the data processing, network architecture, and training, tuning, and evaluating the model.

Recommended Reading

The homework is based on chapter 7 of Jurafsky and Martin. You may also be interested in reading the GloVe paper (Pennington et al. 2014), as we will be making heavy use of GloVe word vectors in this homework. We also recommend reading the SocialIQA paper (Sap et al. 2019), which introduces the social commonsense reasoning dataset that you will be working with in the last exercise of this homework. We provide all the details necessary to solve the homework in this handout and the notebooks.

Required Compute

You can run the notebook on Google Colab with CPU runtime. If your implementation is optimized, all test cases should run fairly quickly. Please do not hardcode the device on PyTorch as `"cuda"`, as Gradescope autograder won't use a GPU. Set device dynamically as: `DEVICE = "cuda" if torch.cuda.is_available() else "cpu"`

Acknowledgment

This assignment is designed by Kabir Ahuja with invaluable feedback from Khushi Khandelwal, Melissa Mitchell, Kavel Rao, and Riva Gore. Kavel Rao also helped design autograder for the homework.

1 Neural Networks for NLP Tasks (70 pt)

You will learn how to define, train, and evaluate neural network models for NLP tasks. In particular, we will work with the simplest form of neural networks, i.e., feedforward networks (FFNs), also called multi-layer perceptrons (MLPs). We will focus on two tasks, sentiment analysis from the previous homework and a task that operationalizes social commonsense reasoning.

Deliverables:

1. **Coding Exercises:** You should complete the code blocks denoted by `YOUR CODE HERE:` in the Python notebook. Do not forget to remove `raise NotImplementedError()` from the code blocks. To submit your code, download your notebook as a Jupyter notebook file (`CSE447_Assignment2.ipynb`).
2. **Write-up:** Your report for §1.2 should be **no more than four pages**. However, you will most likely be able to answer all questions within three pages.
3. **NLU Exercise** For question 4 the write-up, along with the write up, also submit the Python script (.py) file with your code used for defining the data processing, network architecture, and training, tuning, and evaluating the model.

FFNs for text classification. For text classification, we will work with FFNs with a single hidden layer and ReLU activation function. The network architecture is provided in Figure 1. Note that FFNs take a fixed-length vector as input (just like we saw in multinomial and binomial logistic regression). Each text input (here, sentence) needs to be converted into a fixed-length vector representation, regardless of how many words it contains. You will use **sentence embeddings**— either derived by summing up GloVE vectors for the words in the input or sentence transformer embeddings—in your coding exercises. We will discuss how these embeddings are constructed later in the course.

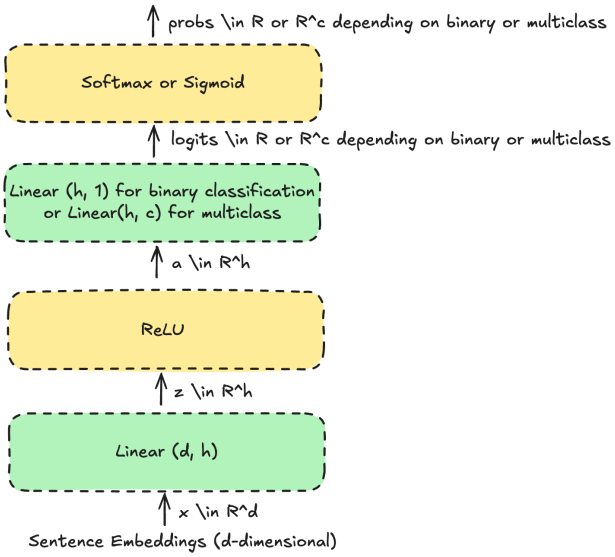

Figure 1: A feedforward neural network architecture for sentence classification.

FFNs for Multiple Choice Question Answering. Apart from the sentiment analysis task, in this homwework you will also be working with a social common sense reasoning dataset called SocialIQA. The

SocialIQA task has as input a context c about some social situation, e.g., *Jordan wanted to tell Tracy a secret, so Jordan leaned towards Tracy*, a question q about the context, e.g. *Why did Jordan do this?*, and three options about the correct answer, a_1, a_2, a_3, e.g., *a) to make sure no one hears it, b) so that everyone hears it, and c) get flirty with Tracy.* The task, given input c, q, a_1, a_2, a_3, is to select the correct answer among a_1, a_2, and a_3. Strictly speaking, this is not a typical classification problem with fixed labels (like positive and negative clases in the sentiment analysis task). Therefore, we might be able to get better performance with a neural network architecture that's different from the one we used for text classification. You will be implementing the architecture in Figure 2 below for this task.

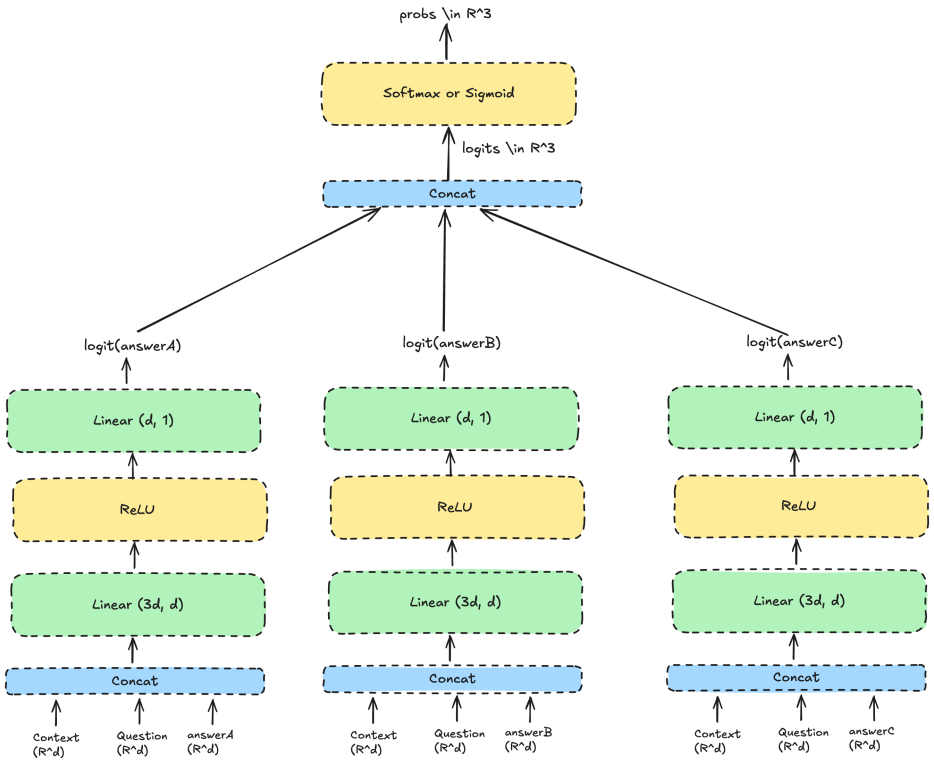

Figure 2: A feedforward neural network architecture for multiple choice question answering.

This architecture performs three forward passes, one for each candidate answer with the context and question. Each pass scores the validity of an answer given the context and question. The scores for the three answers are then concatenated and passed through a softmax layer to get a vector of probabilities, one per answer. Note that the network parameters across the three forward passes are shared, i.e., we use the exact same weights (and biases) to compute the representations for each combination. In PyTorch's terminology the we use the same linear layers for each forward pass instead of creating separate linear layers for each combination.

1.1 Coding Questions

- class `FFNN` (3 pts)
- functions `evaluate` and `train` (12 pts)
- function `predict` (4 pts)

- class `MCQFFNN` (6 pts)
- functions `evaluate_siqa` and `train_siqa` (15 pts)
- function `predict_siqa` (4 pts)

1.2 Write-up Questions

Check section 1 of CSEP517_Assignment2_Writeup.pdf for detail.