CMSI 537
Problem Set 1
Due beginning of class Tuesday, 10/6.

Name:
Collaborators:

Learning Objectives
Students will be able to (1) compute gradients for backpropagation, (2) read a research paper to determine what model was used, (3) implement a feed-forward neural network in `PyTorch`, and (4) initialize word embedding layers with pre-trained embeddings.

Reading
Chapters 7, 9, and 10 in Jurafsky and Martin's textbook: https://web.stanford.edu/~jurafsky/slp3/

Theory

0. **Backpropagation**
The following network takes in three inputs $x = 1$, $y = 2$, and $z = 1$:

$$a = x^2 + y^2 + xz \tag{1}$$
$$b = max(yz, a) \tag{2}$$
$$c = a - 2b \tag{3}$$

(a) Derive the *local* gradients for each layer in the neural network, e.g., $\frac{da}{dx}$, ..., $\frac{dc}{da}$, $\frac{dc}{db}$. You should have 8 derivatives in total.

(b) Using your local derivatives and the chain rule, compute the derivatives of the network output c, with respect to all its inputs x, y, and z. Which input variable influences the output of the network the most, and why?

1. **Softmax gradient**
In class, we derived the gradient of the loss for the sigmoid output activation function in a binary logistic regression classifier, with respect to weight w_j: $\frac{dL}{d_{w_j}} = (\sigma(w \cdot x + b) - y)x_j$

Now, derive the gradient for a softmax output layer, again assuming one hidden layer, where the loss is as follows: $L = -\log P(y = k|x) = -\log \frac{e^{w_k \cdot x + b_k}}{\sum_{j=1}^{K} e^{w_j \cdot x + b_j}}$

Programming
Go to this link to create your own private repository and to import the data I have provided for you: https://classroom.github.com/a/sd-TOKrA

Hate speech detection
Create a new Python 3 file named `classifier.py` in which you will classify tweets into three classes: 0) hate speech, 1) offensive language, or 2) neither. *Note: due to the nature of the task, the data contains offensive language. If you would prefer, we can consider other datasets as inoffensive alternatives.*

0. **Research paper**
   The authors of this work recently published a research paper in which they built a logistic regression classifier: https://arxiv.org/pdf/1905.12516.pdf. Skim the paper, focusing on the abstract and section 3.2 for training classifiers. What features and data pre-processing did they use? How did they evaluate their models–which performed best? What are their takeaways? Do you trust the results? Why?

1. **Logistic regression**—`classifier.py`
   Replicate their results using the code from their Jupyter notebook: https://bit.ly/3hsRiXX in the Python 3 `classifier.py` script already provided for you. Note that they use the `Pandas` library to load the data, and the `scikit-learn` toolkit's `classification_report` method to evaluate precision, recall, and F1-score for each of the three classes. What are your results? Do they match the paper?

2. **Neural networks**—`nn.py`
   Now, implement your own classifier using a feed-forward neural network in `PyTorch`. Rather than using hand-crafted features, the neural network will learn features *automatically* during training. How many layers do you use, and what dimensions? How many epochs do you train for? What loss, optimizer, and learning rate do you use? How does your network compare to the logistic regression model? Notice that they train and test their model on the same data. Is this a good idea—why or why not?

3. **(Optional) RNNs**
   Next, try using a recurrent neural network, i.e., an LSTM. How does it compare to the feed-forward network and the logistic regression classifier? Try a GRU and bidirectional LSTM and compare.

4. **Word embeddings**
   Finally, download one (or more) of the following pre-trained word embeddings to initialize the word embedding layer of your neural networks:

   (a) word2vec: https://code.google.com/archive/p/word2vec/

   (b) GloVe: https://nlp.stanford.edu/projects/glove/

   (c) FastText: https://fasttext.cc/

   Does the performance improve? Does it train faster? **(Optional)** Which embeddings do best here?

What to turn in
Submit the files you modified to your GitHub Classroom repository—make sure the code is beautiful, with well-chosen names, perfect formatting, and appropriate comments (if called for). To submit your code, type the following into your Terminal (or Bit Bash if you're on Windows) from inside your cloned GitHub repository for this assignment (use the `cd` command to navigate into the correct folder):

```
git add classifier.py nn.py
git commit -m "adding files for ps1"
git push
```

Upload to Brightspace your answers to the written questions above, as well as the following questions:

0. Number of hours spent working on this problem set:

1. (Optional) Feel free to let me know what you liked/disliked about this homework, what you learned, etc: