LING572 Hw9: Neural Networks
Due: 11pm on March 13, 2019

A few notes about this assignment:

- The total raw score is 130 points. Your final grade for this assignment will be the minimum of 100 and the raw score. In other words, you can get 100 points, even if some of your answers are wrong.
- The answers to the questions should be pretty short. I've left some space for you to fill in the answers. I've also made the latex file available in case you want to add the answers to the latex file directly. In that case, you need to run pdf2latex (or something like that) to generate a pdf from the latex file.
- If you prefer to write formulas on paper (instead of typing them with latex or Word), it's ok. You just need to fill out the rest of the assignment, print out the file, insert formulas by hand, scan the paper, and then submit via Canvas.
- Since no programming is required, you only need to submit a single file. Please call it **readme.pdf**.
- The assignment has two parts:
  - Q1-Q3 on derivatives: Recall that college-level calculus is a prerequisite of Ling572. If you've forgetten how (partial) derivatives work, feel free to check any calculus textbook or review the Wikipedia pages on those topics. (Just search for derivatives, gradient, partial derivatives, etc.)
  - Q4-Q7: Q4-Q5 are covered in class, and for Q6-Q7 (and also Q5), please read Chapter 1 of Nielsen's NN online book at at http://neuralnetworksanddeeplearning.com/chap1.html

Q1 (10 points): Let $f'(x)$ denote the derivative of a function $f(x)$ w.r.t. the variable x.

(a) 2 pts: What does f'(x) intend to measure?

(b) 2 pts: Let $h(x) = f(g(x))$. What is $h'(x)$?

(c) 2 pts: Let $h(x) = f(x)g(x)$. What is $h'(x)$?

(d) 2 pts: Let $f(x) = a^x$, where $a > 0$. What is $f'(x)$?

(e) 2 pts: Let $f(x) = x^{10} - 2x^8 + \frac{4}{x^2} + 10$. What is $f'(x)$?

Q2 (15 points): The logistic function is $f(x) = \frac{1}{1+e^{-x}}$. The tanh function is $g(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.

(a) 5 pts: Prove that $f'(x) = f(x)(1 - f(x))$.

(b) 5 pts: Prove that $g'(x) = 1 - g^2(x)$.

(c) 5 pts: Prove that $g(x) = 2f(2x) - 1$

Q3 (15 points): Let us denote the partial derivative of a multi-variate function f w.r.t. one of its variables x by f'_x or $\frac{df}{dx}$.

(a) 2 pts: What is f'_x trying to measure?

(b) 2 pts: Let $f(x, y) = x^3 + 3x^2y + y^3 + 2x$. What is f'_x? What is f'_y?

(c) 2 pts: Let $z = \sum_{i=1}^{n} w_i x_i$. What is $\frac{dz}{dw_i}$?

(d) 4 pts: Let $f(z) = \frac{1}{1+e^{-z}}$ and $z = \sum_{i=1}^{n} w_i x_i$.
What is $\frac{df}{dz}$?

What is $\frac{df}{dw_i}$?

Hint: Use the answers that contain $f(z)$.

(e) 5 pts: Let $E(z) = \frac{1}{2}(t - f(z))^2$, $f(z) = \frac{1}{1+e^{-z}}$ and $z = \sum_{i=1}^{n} w_i x_i$. What is $\frac{dE}{dw_i}$? Hint: the answer should contain $f(z)$.

Q4 (10 points): The softmax function:

(a) 5 pts: In general where in NNs is the softmax function used and why?

(b) 5 pts: If a vector x is [1, 2, 3, -1, -4, 0], what is the value of softmax(x)?

Q5 (15 points): Suppose a feedforward neural network has m layers: the input layer is the 1st layer, the output layer is the last layer, and there are $m-2$ hidden layers in between. The number of neurons in the i^{th} layer is n_i. Each neuron in one layer is connected to every neuron in the next layer and there is no other connection.

(a) 5 pts: How many connections (i.e., weights) are there in this network?

(b) 10 pts: Let x be a column vector that denotes the values of the input layer. Let M_k denote the weight matrix between layer k and $k+1$; that is, the cell $a_{i,j}$ in M_k stores the weight on the arc from the j^{th} neuron in layer k to the i^{th} neuron in layer $k+1$. Let g be the activation function used in each layer.

- Given the input x, what is the formula for calculating the output of the first hidden layer?

- Given the input x, what is the formula for calculating the output of the output layer?

- Hint: In class, we show the formula for calculating the z and y value for a neuron, where $z = b + \sum_j w_j x_j$ and $y = g(z)$. Now there are n_2 neurons in the 2nd layer. The output of this layer, y, is going to be a column vector, not a real number. The weights between the two layers are no longer a vector, but a $n_2 \times n_1$ matrix denoted by M_1. So the answer to the 1st question should be a simple formula that uses matrix operations. For the sake of simplicity, let's assume the bias b is always zero.
- Terminology: A row vector is a $1 \times n$ matrix (e.g., $[a_1, a_2, ..., a_n]$); a column vector is a $n \times 1$ matrix. If you transpose a row vector, you get a column vector.

Q6 (40 points): Read Chapter 1 of the NN book, and answer the following based on that chapter:

(a) 5 pts: What's the loss function used in the digit recognition task? Why do they choose to minimize this function instead of maximizing classification accuracy?

(b) 10 pts: In gradient descent, what's the formula for updating the weight matrix (or vector)? And why is that a good formula?

(c) 15 pts: What are the main idea and benefit of stochastic gradient descent?

What is a training epoch?

Let T be the size of the training data, m be the size of mini-batch, and your training process contains E training epoches. How many times is each weight in the NN updated?

(d) 10 pts: How can one choose the learning rate? What's the risk if the rate is too big? What's the risk if the rate is too small?

Q7 (25 points): Go over the source code in Nielson's package stored under /dropbox/18-19/572/hw9/nielsen-nn/ on patas and understand the part explained in Chapter 1 of the NN book.

- Run the code (following the instructions in chapter 1) and fill out Table 1. For this exercise, use only one hidden layer.
- It seems that the code works with python 2.*, not with python 3.*. If you run the default python version on patas, which is 2.7.5, the code should work.
- Note that as the package uses random functions a few times, your results will not be the same when running it multiple times.

Table 1: Results on digit recognition

Expt id	# of hidden neurons	epoch #	mini batch size	learning rate	accuracy
1	30	30	10	3.0	
2	10	30	10	3.0	
3	30	30	10	0.5	
4	30	30	10	10	
5	30	30	100	3.0	

Submission: Submit the following to Canvas:

- Since hw9 has no coding part, you only need to submit your **readme.pdf** which includes answers to all the questions, plus anything you want TA to know. No need to submit anything else.