Sample Quiz 1 for COMS 4776
Neural Networks and Deep Learning
Spring 2025

Name: ______________________________

Uni: ____________________

This is a closed-book test. It is marked out of 100 marks. Please answer ALL of the questions. Here is some advice:

- The questions are NOT arranged in order of difficulty, so you should attempt every question.
- Questions that ask you to "briefly explain" something only require short (1-3 sentence) explanations. Don't write a full page of text. We're just looking for the main idea.
- None of the questions require long derivations. If you find yourself plugging through lots of equations, consider giving less detail or moving on to the next question.
- Many questions have more than one right answer.

.

Q1:	______	/ 16
Q2:	______	/ 20
Q3:	______	/ 20
Q4:	______	/ 18
Q5:	______	/ 20
Q6:	______	/ 20

Final mark: ______ /

1. Give short answers for each of the following (4 points each).

   (a) For a fully-connected deep network with one hidden layer, increasing the number of hidden units should have what effect on bias and variance? Explain briefly.

   Solution: Adding more hidden units should decrease bias and increase variance. In general, more complicated models will result in lower bias but larger variance, and adding more hidden units certainly makes the model more complex.

   (b) Suppose you are training a neural network with dropout probability $p = 0.3$. Why might we want to avoid applying dropout at test time? What should we do instead?

   Solution: Using dropout at test time will require testing each example multiple times, which could be costly. If we do not use dropout during test time, all the units will receive more inputs than they do during training time, so their responses will be very different. Therefore, at test time, we compensate for this by multiplying the values of the weights by $1 - p$, or in this case 0.7.

   (c) Write down the softmax activation function, for a network with K output units. What is its inductive bias, i.e., what does it assume about the target outputs for the network?

   Solution:

$$y_j = \exp(z_j) / \sum_{k=1}^{K} \exp(z_k)$$

   The outputs are all between 0 and 1 and sum to one. Assumes that only one of the K units is the target, a one-hot target vector.

   (d) Imagine you have trained a large neural network on 1,000,000 images to classify between cats and dogs. Please describe and briefly justify your strategy for transferring this pre-trained model to each of these classification problems: (a) lions vs. wolves; (b) cars vs. airplanes; (c) healthy lung vs. unhealthy lung (in an X-ray).

Solution: Possible strategies include freezing the network or fine-tuning some or all of the layers; in some cases, the best option may be training from scratch, e.g., if you would expect negative transfer.

2. **CNN:** A CNN consists of N layers. Each layer consists of a convolution with a 3x3 kernel, a sigmoid activation, and a max pooling operation with a 2×2 window. There is no padding. The input image is 512x512. Compute the dimensions of the second and third layers. After how many layers is the image of size 1 x 1, i.e., what is the maximum value for N ?

   Solution: Image width (or height) after first convolution: $512 - 3 + 1 = 510$. The activation does not change the size. After pooling: $510/2 = 255 = 2^8 - 1$. Second layer: $255 - 3 + 1 = 253$. After pooling: $253/2 = 127 = 2^7 - 1$. The image width becomes $1 = 2^1 - 1$ after $N = 8$ layers.

3. **a)** [**15pt**]We can construct hard-coded neural networks to solve algorithmic questions. In this problem, you will design a multilayer perceptron that determines if an ordered sequence of inputs follows a *Shmibonacci* sequence. Element s_i, $i \geq 3$, of a *Shmibonacci* sequence satisfies the property:

$$s_i \geq i * s_{i-1} + (i-1)s_{i-2}. \tag{1}$$

The first two elements of a *Shmibonacci* sequence can be any two numbers. Your network will receive four inputs, $\mathbf{x} := [x_1, x_2, x_3, x_4]^\top \in \mathbb{R}^4$. The network should output 1 if the input is a *Shmibonacci* sequence. Your network will use the following architecture.

$$\mathbf{z} \in \mathbb{R}^h := \phi(W_1\mathbf{x} + \mathbf{b}_1) \tag{2}$$

$$\mathbf{y} \in \{0,1\} := \phi(w_2^\top \mathbf{z} + b_2) \tag{3}$$

Where ϕ is an element-wise hard-threshold activation function.

$$\phi(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

The shapes of the weights and biases are $W_1 \in \mathbb{R}^{h\times 4}, \mathbf{b}_1 \in \mathbb{R}^h, w_2 \in \mathbb{R}^h, b_2 \in \mathbb{R}$. **You must determine the appropriate number of hidden units for your network.** Justify your choice of hidden unit number and provide all four sets of weights and biases.

b) [**5pt**]If element s_i of a *Shmibonacci* sequence was defined as:

$$s_i := i * s_{i-1} + (i-1)s_{i-2}. \tag{4}$$

How could the network be changed to compute this? *Hint: Change the hard threshold function.*

Solution: **a)** The network should only have two hidden units, as we only need to compute two conditions: whether x_3 is a *Shmibonacci* number, and whether x_4 is a *Shmibonacci* number.

The general architecture of the network should be checking if both inputs *Shmibonacci* numbers, and then taking their conjunction.

Hidden layer: $\mathbf{W}^{(1)} = \begin{pmatrix} -2 & -3 & 1 & 0 \\ 0 & -3 & -4 & 1 \end{pmatrix}$, $\mathbf{b}^{(1)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$,

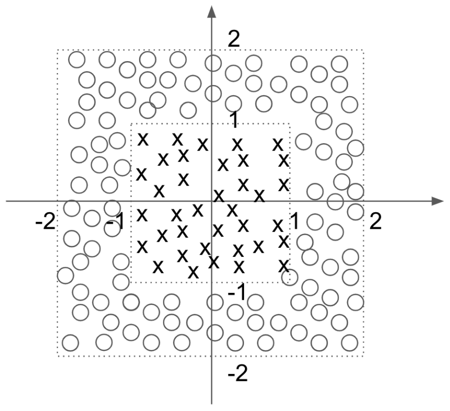
Output layer:

$\mathbf{w}^{(2)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, $b^{(2)} = \begin{pmatrix} -1.5 \end{pmatrix}$ or any parameterization that computes a conjunction when used with a hard threshold.

b) One could redefine the hard-threshold function as the following:

$$\phi(x) = \begin{cases} 1 & x = 0, \\ 0 & x \neq 0 \end{cases}$$

4. Consider the following 2D classification problem. Assume the training examples are drawn i.i.d. from a uniform square in the input space $x_1, x_2 \in [-2, 2]$. Data points are assigned to the positive class if they fall in the center square between -1 to 1, and to the negative class otherwise.



(a) **[6pt]** When there are *more* than three training examples, will a linear classifier alwyas obtain zero training error on this dataset? Provide proof to justify your answer. (Hint: convex set)

Solution: Linear classifier will not always obtain zero training error. The mean of the training examples from either the positive or the negative class is $(0, 0)$. If the dataset is linearly separable then $(0, 0)$ can only be assigned to either of the classes. Thus, contradiction.

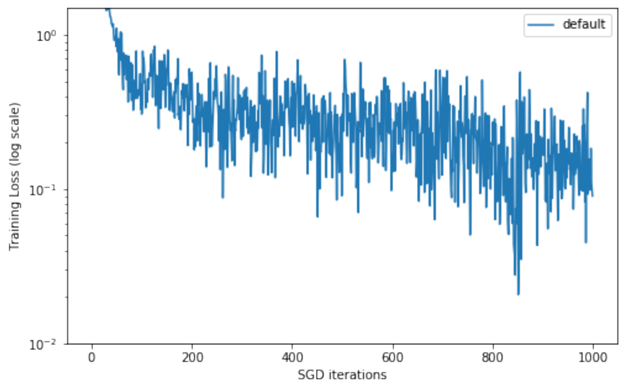
(b) **[6pt]** When there are *less* than or equal to three training examples, can a linear classifier on top of feature maps obtain zero training error on this dataset? Provide proof to justify your answer.

Solution: Yes, it can simply memorize the class assignment with appropriate feature maps. However, when all three training examples lie on a line (o, x, o), then a linear classifier will not able to classify this perfectly.

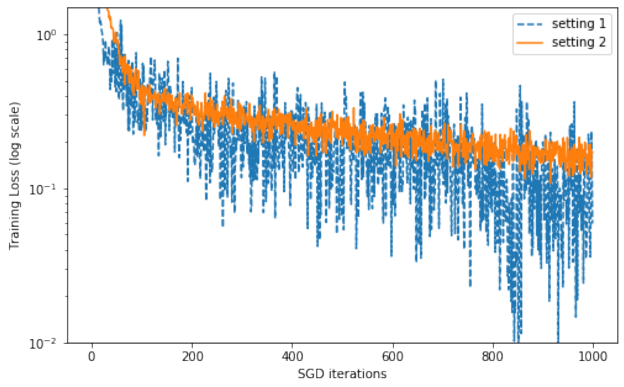
(c) **[6pt]** When there are *more* than three training examples, can a one-hidden-layer MLP containing a single sigmoid unit (i.e. the hidden size is 1) achieve zero training error? You do not need to justify your answer.

Solution: No. A single sigmoid neuron MLP is no more powerful than a linear classifier.

5. **[20pt]** Consider training a neural network language model from Assignment 1 using stochastic gradient descent (SGD). Given the starter code learning rate and batch size, we obtain the following training curve, that is the training loss vs. the number of iteration plot.



To improve convergence, we tweaked learning rates and batch sizes in two different settings (setting 1 and setting 2):



Describe what you think may have been changed in each setting, in terms of learning rate and/or batch size. Briefly justify your answer.

setting 1: Solution: increased learning rate and/or decreased batch size, which increases the update variance.

setting 2: Solution: increased batch size and/or decreased learning rate, which reduces the update variance.

6. **[20pt]** When training a deep neural network, we encounter an out-of-memory error during back-propagation. Assume there is no bug in the code. We also want to keep the same model architecture and the dataset. Describe two potential ways to avoid the out-of-memory error, without adding more memory.

   Solution: 1) Reduce the minibatch size. 2) To keep the minibatch size the same, we can perform backprop on a partition of the training examples and accumulate the gradients across the different partitions. 3) Recompute hidden activation during backprop, so we only need to store the error signal and the most immediate hidden activations.

(Scratch work or continued answers)