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

Name: Aksel Kretsinger-Walters

Uni: adk2164@columbia.edu

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.

   You'd expect bias to decrease but variance to increase

   (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? You increase variance (and potentially bias too) by applying dropout at test time. Instead, we should normalize the predicted values by multiplying them proportionally to the dropout (1-p)

   (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? $\sigma(\vec{x})_j = \frac{e^{x_j}}{\sum_{i=1}^{K} e^{x_i}}$ It assumes network only desires one output value

   (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).

These 3 tasks are ordered by descending applicability / similarity to the original task, hence we will "bisect" the neural network at different levels.

a) In the lions vs wolves case, the neural network would almost perfectly work on the new task. This is because lion is a type of cat, and wolf is a type of dog. I would take the neural network as-is and retrain the network on the new images, anticipating some small weight updates (as the network takes advantage of the narrower scope of the problem - the generalized dog process can be reduced to just a wolf identification process, and same with a cat -> lion)

b) I would bisect the network quite a bit lower. The first few convolution layers would naturally identify edges & shapes, which are transferable to car and airplane classification. A caveat / rule of thumb with such a transfer is to set a slower learning rate for the transferred nodes to avoid "unlearning" the helpful patterns transferred as training begins

c) I would probably start fresh here. There probably isn't much overlap between cat/dog recognition and lung classification, so there's little value to transferring knowledge

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 ?

$$\text{out} = \left\lfloor \frac{n - k + 2p}{s} \right\rfloor + 1$$

Stride = 1 (assumption)

$$\text{out}_{K1} = \left\lfloor \frac{512 - 3}{1} \right\rfloor + 1 = 510$$

$$\text{out}_1 = 510/2 = 255$$

$$\text{dim}_2 = 255$$

$$\text{out}_{K2} = 255 - 3 + 1 = 253 \rightarrow 252$$

$$\text{out}_2 = 252/2 = 126$$

$$\text{dim}_3 = 126$$

X We can see that the dimension halves at each step

$2^9 \geq 512$

After 9 layers the dim = 1

$512 \rightarrow 126 \rightarrow 62 \rightarrow 30 \rightarrow 14 \rightarrow 6 \rightarrow 2 \rightarrow 1$

1 2 3 4 5 6 7 8

$n_{max} = 8$

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:

$3 \cdot s_2 + 2 \cdot s_1$

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

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.

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

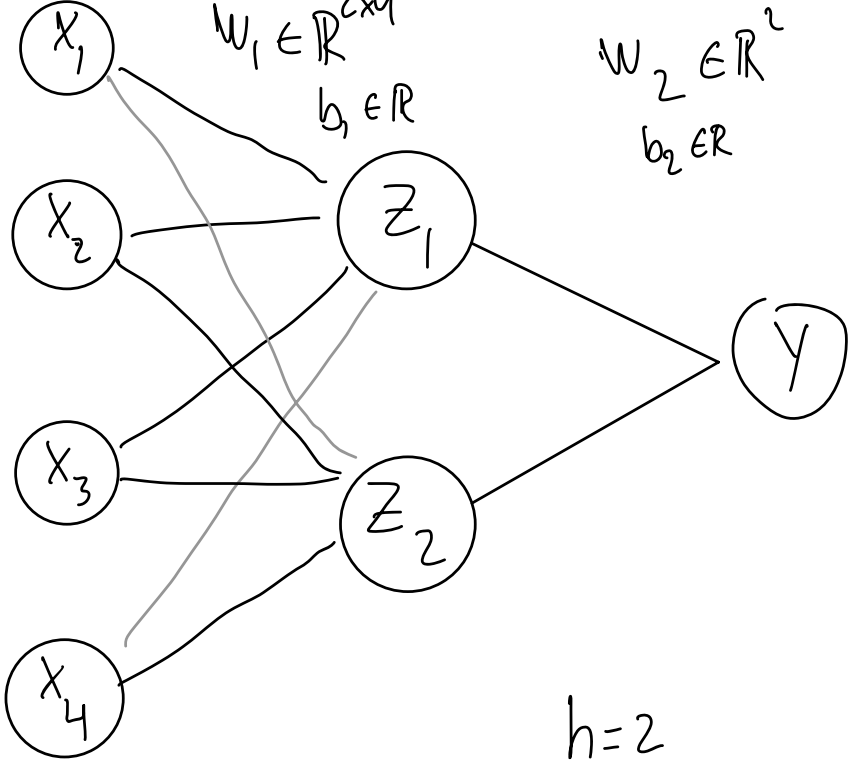
The shapes of the weights and biases are $W_1 \in \mathbb{R}^{h \times 4}, \mathbf{b}_1 \in \mathbb{R}^4, w_2 \in \mathbb{R}^h, b_2 \in \mathbb{R}$. **You must determine the appropriate number of hidden units for your network.** Justify your choice of hidden unit number and provide all four sets of weights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

3a)



$W_1 \in \mathbb{R}^{2\times 4}$ $\quad b_1 \in \mathbb{R}$

$W_2 \in \mathbb{R}^2$ $\quad b_2 \in \mathbb{R}$

$$W_1 = \begin{bmatrix} -3 & -2 & 1 & 0 \\ 0 & -4 & -3 & 1 \end{bmatrix}$$

$b_1 = 0$

$$W_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$b_2 = -2$

$h = 2$

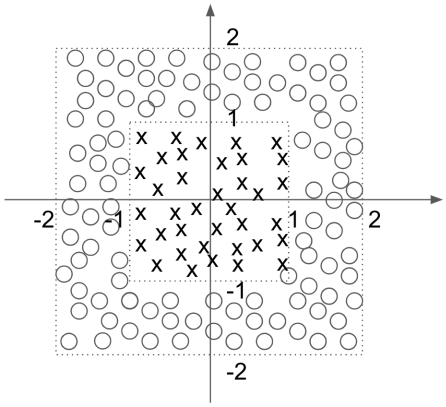
$h = 2$ because there are $n-2$ comparison checks to make for a shmibanocci sequence

3b) To be clear, we could keep the (final) threshold funtion as is, or modify it to match the proposed function. Either would work.

$$\phi(x) = \begin{cases} 1 & |x| < \varepsilon \\ 0 & |x| \geq \varepsilon \end{cases} \qquad \varepsilon = 1e-6$$

$$:= \vec{1}[|x| < \varepsilon]$$

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.

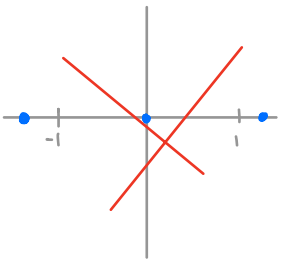


(a) **[6pt]** When there are *more* than three training examples, will a linear classifier alwyas obtain zero training error on this dataset? Provide proof to justify your answer. (Hint: convex set)

(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.

(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.

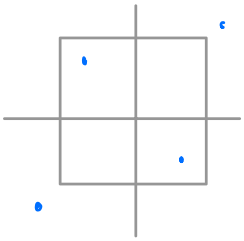
a) No, we can create a linearly non-separable convex set w/ three points

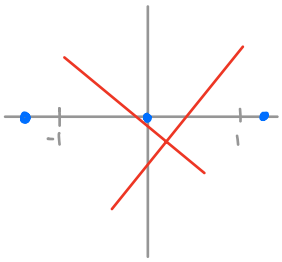


$$X = \begin{matrix} (x) & (y) \\ \end{matrix}$$
$$\begin{bmatrix} 1.5 & 0 \\ 0 & 0 \\ -1.5 & 0 \end{bmatrix} \begin{matrix} x_1^T \\ x_2^T \\ x_3^T \end{matrix}$$

Adding more points only makes separation more difficult

Not sure why we need 4 units?



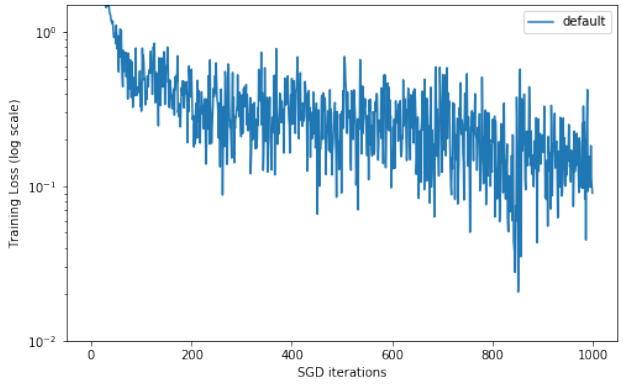
b) 

No, due to this ex

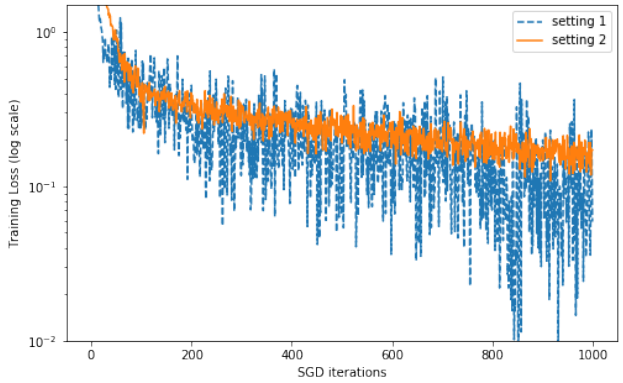
This is a "degenerate" example of collinearity – can't draw conclusions from it

c) Yes

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: ↓ batch size ↑ learning rate

- doing both changes would result in a "noisier" loss function.

setting 2: ↑ batch size or ↓ learning rate or both

This training example is much smoother. This could result from increasing batch size or decreasing the learning rate, or both

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.

1) We could reduce the "connectedness" of the net, akin to what a CNN does. This would reduce the # of stored weights & biases

2) Reduce depth / width of net. This would also shrink the size of the net & hence the weights

3) Decrease batch size. Load in less samples per step & reduce computation burden

(Scratch work or continued answers)