CSE 253
Winter 2018
Midterm 1
Section A - Version 1

DO NOT TURN THIS PAGE UNTIL YOU ARE TOLD TO START!!!!

February 7th, 2018

- Name:
- PID:

- No Notes Allowed
- Once the exam has started, SORRY, NO TALKING!!!
- There are 5 problems: Make sure you have all of them - AFTER YOU ARE TOLD TO START!
- Read each question carefully.
- Remain calm at all times!

Problem	Type	Points
1	True or False	10
2	Short Answer	24
3	Back Propagation	15
4	CNN	6
	Total	**55**

Problem 1: True/False (10 pts) PID:

(10 pts: +1 for correct, -0.5 for incorrect, 0 for no answer) If you would like to justify an answer, feel free. **Please CLEARLY write ONLY T or F.**

______ Because it is easier to train a smaller network, it is easier to train an MNIST classifier by reducing the ten outputs to one, with a single, linear activation output unit, which will produce the integer value of the number seen in the input. False; 1) if you do this, you've turned it into a regression problem, but this is inherently a classification problem. 2) it would not produce an integer, it would produce a real number, and getting the network to activate a linear output in steps like this makes little sense.

______ In a problem with a very gradually sloping error surface, momentum will not help the network converge faster. False: In fact, this is just the kind of case momentum works well for. Momentum will cause the network to speed up down this gentle slope.

______ In a diverse dataset like Imagenet, images share general pixel value statistics, despite being from 1000 different classes. True: This is the "stationary distribution" principle, which is reflected in convnets by...convolutional filters.

______ When computing weight updates for a minibatch, you must first compute and apply the Δw_{jk} update before you can compute Δw_{ij} [Assume here w_{ij} is the weight from unit i to unit j]. False: You always compute the deltas first, *then* change the weights.

______ Convolutional networks work better on image recognition tasks, but always end up requiring more parameters per layer. False: Fully connected networks generally require more parameters per layer. Look at the slides about cutting out different layers in Alexnet.

______ Given δ_k from output units, the computation of δ_js for the hidden layer does not depend directly on the network's error function. True: The computation of the deltas always follows the same formula for the hidden nodes.

______ When building a convolutional neural network, one of the most important decisions is choosing your first layer's features. False - The features are learned, you don't choose them.

______ Pixels in one region of an image don't typically inform what pixels in another region might be, and this property is represented in convnets by small receptive fields. True: This property of convnets reflects the *locality property* of images.

______ Suppose UCSD administration asks you to build a neural network which can take attendance in a lecture hall, given a clear image of all the seats during lecture. You know you need n output units, one for each of n students enrolled in the class. **You should choose sigmoid over softmax activation for these output units.** True: To have the network turn on outputs for multiple students, you don't want the outputs to compete with each other. This is similar to the image tagging example from Clarifai that we talked about in class.

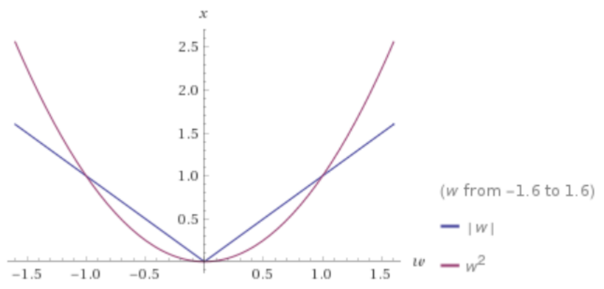
_______ No one knows where the hidden units are hidden, or who put them there. Some suspect Geoff Hinton did it. True or False (free point)

Problem 2: Short answer questions. (24 pts)

a) (2 pts) Imagine you have a dataset of images of animals collected during a sunny day, so the training images are bright. You train a neural network to classify the animals. The network does not generalize well on some images collected in dimmer conditions. Which input pre-processing step did you miss while training the network?

Normalizing the input set. Subtracting by mean should also work.

b) (3 pts) Which regularizer, L1 or L2, is more aggressive at eliminating weights? Why? Support your answer using graphs of the two regularizer objective functions.



L1 is more aggressive at eliminating weights.
For L1, we penalize the loss by $L_1(w) = |w|$ whereas for L2, we penalize the loss by $L_2(w) = w^2$. The gradients for L1 and L2 are
$\frac{dL_1(w)}{dw} = sign(w)$
$\frac{dL_2(w)}{dw} = 2w$
We updates the weights in the opposite direction of the gradient. Steeper the gradient, larger the step we make. You can see from the figure that L1 has steeper gradient than L2 in [-1,1].

For L1, the gradient is either +1 or -1. That means in case of L1 regularization, weights move towards zero irrespective of their values. While in case of L2, gradient depends on w. In case of L2, as w approaches 0, the gradient decreases and hence we take smaller and smaller steps. Therefore, L1 is more aggressive in pulling weights towards zero than L2.

c) (2pts) Explain in what scenario we want to use Sum-of-Squared error (SSE) as the loss function. Be sure to give the rationale for SSE.
In a regression problem with the probability of error gaussian, maximizing likelihood is same as minimizing SSE

d) (2 pts) Why are adaptive learning rates a good idea when training a neural network?

Solution
1. Each weight will have different gradient value, so individual weights should have different learning rates to speedup training.
2. Learning rate should typically be large at start of training, and small, when near the minima for better convergence.

e) (2 pts) Is it true that any multi-layered neural net with linear activation functions at hidden layers can be represented as a neural net without any hidden layer? Explain.

No: An autoencoder network that reduced the dimensionality of the data would not be represented by a single layer net. However, I will accept either answer here. If they said true, then they should say that you can just multiply the weight matrices together and get the same function.

f) (2 pts). Explain how the choice of activation function affects gradient backpropagation in deep networks.

g) (3 pts) A perceptron has two inputs with weights $w_1 = 0.5$ and $w_2 = -0.2$, and a threshold $\theta = 0.3$. What are the new values of the weights and threshold after one training step with input $x = [0, 1]^T$, target=1, and learning rate $\eta = 0.5$? Show work.

Solution
Perceptron Update rule: $w = w + \eta(t - y)x$
$\eta = 0.5 \quad t = 1$
$y = 0(0.5x_1 - 0.2x_2 < 0.3)$
$(t - y) = 1 - 0 = 1$

$w_1 = 0.5 + 0.5(1)0 = 0.5$
$w_2 = -0.2 + 0.5(1)1 = 0.3$
$-\theta = -0.3 + 0.5(1)1 => \theta = 0.2$
$[w_1, w_2, \theta] = [0.5, 0.3, 0.2]$

h) (2 pts) In mini-batch learning, why is it advisable to shuffle data after each epoch?

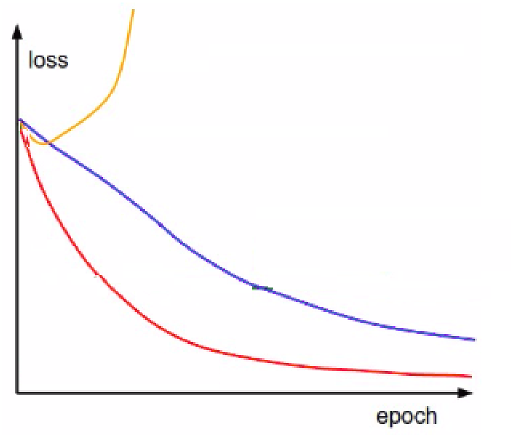
examples of all classes are captured/examples are not repeated/Get a more representative distribution of data within the batch

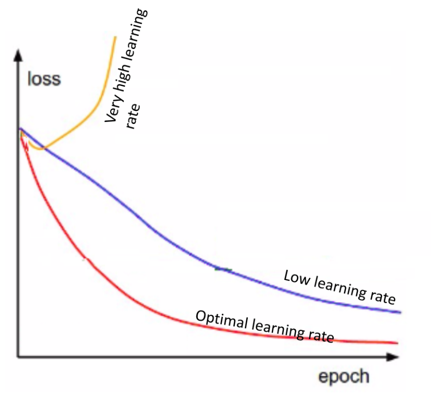
i) (2 pts) How many parameters are there in a neural network with d inputs (not including the bias), c outputs, and H hidden units? Include the bias parameters in your answer.
$(d+1)H + (H+1)c :$ if bias not included in d

$(d)H + (H+1)c :$ if bias included in d

j) (2 pts) Why does batch normalization allow scaling and shifting of the variable being normalized?
Batch normalization allows scaling and shifting using learnable parameters γ and β. These parameters can learn to scale and shift the normalized variable to a range best suited for the task and have the capacity to undo the normalization if that is the desirable thing to do for a task.

k) (2 pts) The figure to the right plots the training set loss over training time, using different learning rates. Label each line with one of these labels: (i) Low learning rate; (ii) Optimal learning rate; (iii) Very high learning rate.


loss
epoch


loss
Very high learning rate
Low learning rate
Optimal learning rate
epoch

PID:

Problem 3: Forward/Backward Propagation. (15 pts)

a. (2 pts) Write down the equation for computing δ_j (the hidden delta). Assume that the next layer up is indexed by k. No derivation required.

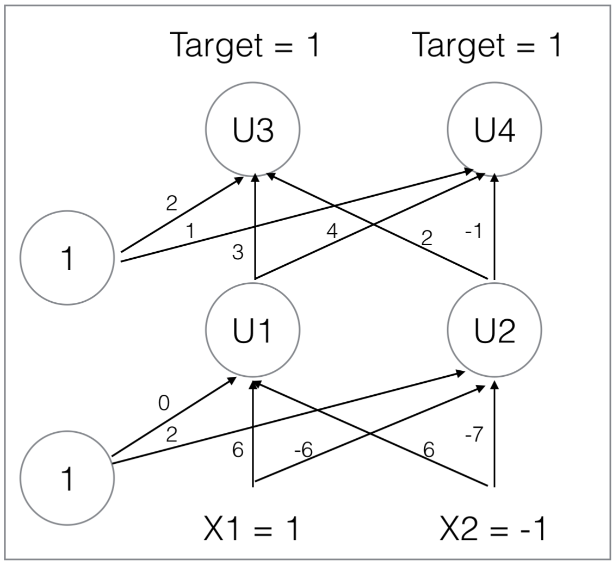
b. (10 pts) Consider the simple network below. The initial weights and biases are as shown (biases are shown as a weight from units with a "1" inside). All units use the **ReLU** activation function $g(a) = max(a, 0)$. Assume the δ at the outputs is simply $t - y$. The inputs, $X1$ and $X2$ are 1 and -1 respectively. In the Panels given below fill in the values:

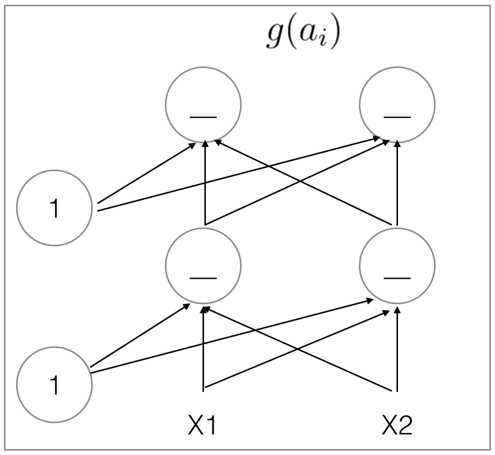
   (a) (2 pts) Fill in the activations of the neurons in Panel a (after applying ReLU).

   (b) (4 pts) Fill in the deltas of the 4 nodes computed via backprop in Panel b. If a ReLU unit's activity is 0, you can assume the slope is 0.

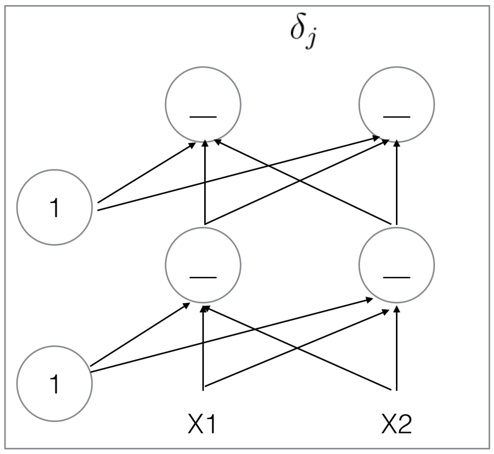
   (c) (0.5 pts each) Show the weight and bias updates requested in the table. $\eta = 1.0$.

c. (3 pts) Report the new output activation for (1,1) input (assuming you've applied *all* weight changes). Is it closer to the targets? Why or why not?





Panel (a).



Panel (b).

WHAT	VALUE
New value of weight from U1 to U3	
New value of weight from U1 to U4	
New value of weight from U2 to U3	
New value of weight from U2 to U4	
New value of weight from X1 to U1	
New value of weight from X1 to U2	
New value of bias of U3	
New value of bias of U4	

$activations$
$au1 = 0$
$au2 = 3.0$
$au3 = 8.0$
$au4 = 0$
$deltas$
$delta_1 - 0.0$
$delta_2 - 15.0$
$delta_3 - 7.0$
$delta_4 1$
$tablevalues$
$w13 = 3.0$
$w14 = 4.0$
$w23 = -19.0$
$w24 = 2.0$
$w11 = 6.0$
$w12 = -21.0$
$bu3 = -5.0$
$bu4 = 2.0$
$Q3B$
$au1 = 0$
$au2 = 0$
$au3 = 0$
$au4 = 2.0$

Problem 4. Convnets (6 pts)

a) (2 pts) Following is an input image (I) of dimention 5x5. Apply kernel (K) of size 3x3 on this image. Use “0”-padding of size 1 and stride of 2.

$$I = \begin{bmatrix} 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 2 & 1 & 1 & 0 \end{bmatrix}$$

$$K = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Write the output of the kernel as a matrix

Size of the output $= (5 - 3 + 2)/2 + 1 = 3$

$$\begin{bmatrix} 0 & 2 & 2 \\ 0 & 7 & 0 \\ 3 & 1 & 0 \end{bmatrix}$$

b) (2 pts) We stack three convolutional layers on top of one another with filter size of 3×3, stride 1, and no zero-padding. What would be the size of the *effective receptive field* (the size of the image patch that it receives input from) of a neuron in the third layer?

7x7

c) (2 pts) Discuss why we use max pooling layers in ConvNets.

The most important use of maxpooling is to help your network invariant to small translations. (Note, small not large translations). This helps your convolutional network to start growing it's knowledge about global semantics by aggregating local knowledge.
By aggregating outputs of local neurons together, you enhancing the overall receptive field when you perform convolution over the max-pooled output. Hence, it helps in increasing the overall receptive filed of the network without introducing new parameters.
Lastly we should not forget, that it helps in making the architecture less memory intensive. You can reduce the size by max pooling the input by introducing a stride parameter.