CSCE 633: Machine Learning

EXAM # 2

Fall 2018

Total Time: 75 minutes

Name: _____

UID: _____

Question	*Point*	*Grade*
1	*20*	
2	*35*	
3	*45*	
Total	*100*	

Full Name of Person Sitting to Your Left:

Full Name of Person Sitting to Your Right:

1. (20 points) Concepts.

a) What are the differences between Bagging, Random Forest, and Boosting? What are some pros and cons of each?

Bagging: (+4)

- For N times:
  - Generate independently bootstrap datasets from original data
  - Run a decision tree in each one of them
- Aggregate the results from all the trees

Random Forests: (+4)

- Same as bagging
- Also subsample the features

Boosting: (+4)

- For each new weak learner adjusts the weight based on the last results unlike other two which train independent learners

Pros/Cons:

Bagging: (any two +1)

+ variance reduction
+ handles categorical variables
- Visually not interpretable

RF: (any two +1)

+ decorrelates the trees unlike bagging
+ reduces variance
- Not interpretable visually

Boosting: (any two +1)

+ can handle qualitative data
+ learns from mistakes
- Learns slowly
- Can overfit

b) Is there a limit to the kinds of methods that can be gradient boosted?

While tree-based methods are most common, no – could guide any method that allows sample weights to be boosted (+5)

2. (35 points) Clustering

a) (5 points) Principal Component Analysis. Assume you have a matrix X that is n samples by p features. How do you prepare this data for use in PCA?

mean center data (+3)
XT*X for covariance matrix (only +1 X * XT is wrong because you want features not people) (+2)

Bonus points if they explain the rest of PCA in the eigen decomposition

b) (10 points) Assume you have a Matrix A

$$A = \begin{pmatrix} 2 & 2 \\ 5 & -1 \end{pmatrix}$$

Find the Eigenvalues λ and associated eigenvectors.

The eigenvalues are those λ for which $\det(\mathbf{A} - \lambda\mathbf{I}) = 0$. Now

$$\begin{aligned}
\det(\mathbf{A} - \lambda\mathbf{I}) &= \det\left(\begin{bmatrix} 2 & 2 \\ 5 & -1 \end{bmatrix} - \lambda\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) \\
&= \det\left(\begin{bmatrix} 2 & 2 \\ 5 & -1 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}\right) \\
&= \begin{vmatrix} 2-\lambda & 2 \\ 5 & -1-\lambda \end{vmatrix} \\
&= (2-\lambda)(-1-\lambda) - 10 \\
&= \lambda^2 - \lambda - 12.
\end{aligned}$$

The eigenvalues of $\mathbf{A}$ are the solutions of the quadratic equation $\lambda^2 - \lambda - 12 = 0$, namely $\lambda_1 = -3$ and $\lambda_2 = 4$.

+3 for determinant

+2 for correct eigenvalues

+3 for correct setup for eigenvectors

+2 for correct eigenvectors

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

and using the matrix $\mathbf{A}$ from above, we have

$$\mathbf{Ax} = \begin{bmatrix} 2 & 2 \\ 5 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2x_1 + 2x_2 \\ 5x_1 - x_2 \end{bmatrix},$$

while

$$-3\mathbf{x} = \begin{bmatrix} -3x_1 \\ -3x_2 \end{bmatrix}.$$

Setting these equal, we get

$$\begin{bmatrix} 2x_1 + 2x_2 \\ 5x_1 - x_2 \end{bmatrix} = \begin{bmatrix} -3x_1 \\ -3x_2 \end{bmatrix} \Rightarrow 2x_1 + 2x_2 = -3x_1 \quad \text{and} \quad 5x_1 - x_2 = -3x_2$$
$$\Rightarrow 5x_1 = -2x_2$$
$$\Rightarrow x_1 = -\frac{2}{5}x_2.$$

This means that, while there are infinitely many nonzero solutions (solution vectors) of the equation $\mathbf{Ax} = -3\mathbf{x}$, they all satisfy the condition that the first entry x_1 is $-2/5$ times the second entry x_2. Thus all solutions of this equation can be characterized by

$$\begin{bmatrix} 2t \\ -5t \end{bmatrix} = t \begin{bmatrix} 2 \\ -5 \end{bmatrix},$$

where t is any real number. The nonzero vectors $\mathbf{x}$ that satisfy $\mathbf{Ax} = -3\mathbf{x}$ are called *eigenvectors* associated with the eigenvalue $\lambda = -3$. One such eigenvector is

$$\mathbf{u_1} = \begin{bmatrix} 2 \\ -5 \end{bmatrix}$$

and all other eigenvectors corresponding to the eigenvalue (-3) are simply scalar multiples of $\mathbf{u_1}$ — that is, $\mathbf{u_1}$ spans this set of eigenvectors.

Similarly, we can find eigenvectors associated with the eigenvalue $\lambda = 4$ by solving $\mathbf{Ax} = 4\mathbf{x}$:

$$\begin{bmatrix} 2x_1 + 2x_2 \\ 5x_1 - x_2 \end{bmatrix} = \begin{bmatrix} 4x_1 \\ 4x_2 \end{bmatrix} \Rightarrow 2x_1 + 2x_2 = 4x_1 \quad \text{and} \quad 5x_1 - x_2 = 4x_2$$
$$\Rightarrow x_1 = x_2.$$

Hence the set of eigenvectors associated with $\lambda = 4$ is spanned by

$$\mathbf{u_2} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

This page intentionally left blank

c) (5 points) Assume you have 3 clusters in your data. Please explain how K-Means Clustering finds these clusters. How do you assign a new member to one of the existing clusters?

+3 points – Randomized cluster centers then move them through distance metrics
+2 points – Euclidean distance – strict clustering.

d) (5 points) Now assume you wanted to repeat the process using Gaussian Mixture Models, explain how Expectation Maximization works. do you assign a new member to one of the existing clusters and how does this differ from K-Means?

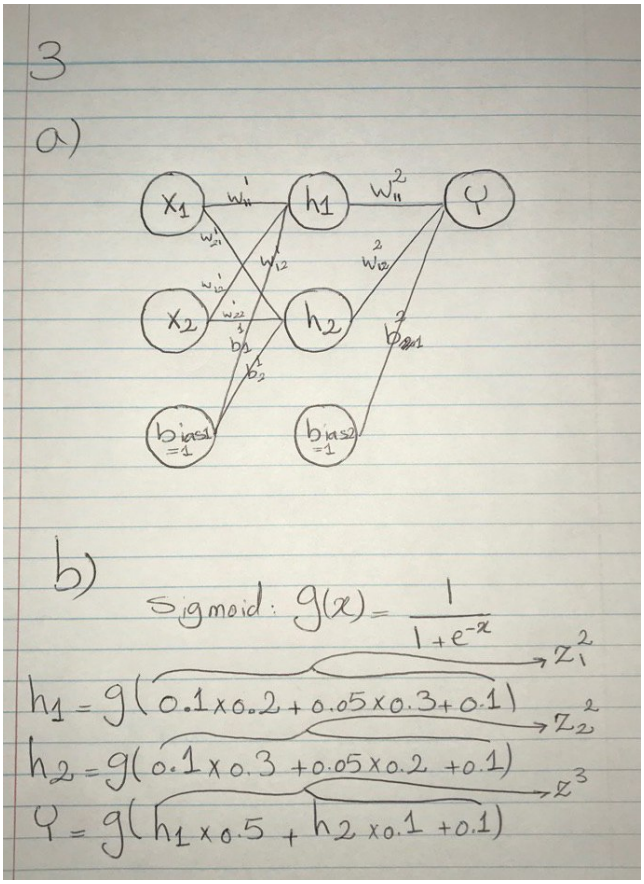
+3 points EM explanation
+2 points soft membership – probabilities

3. (25 Points) Neural Networks

a) (5 points) Assume you have two inputs X_1 and X_2 and want to calculate an Output Y, using one hidden layer of two nodes. Please draw this network fully connected network.

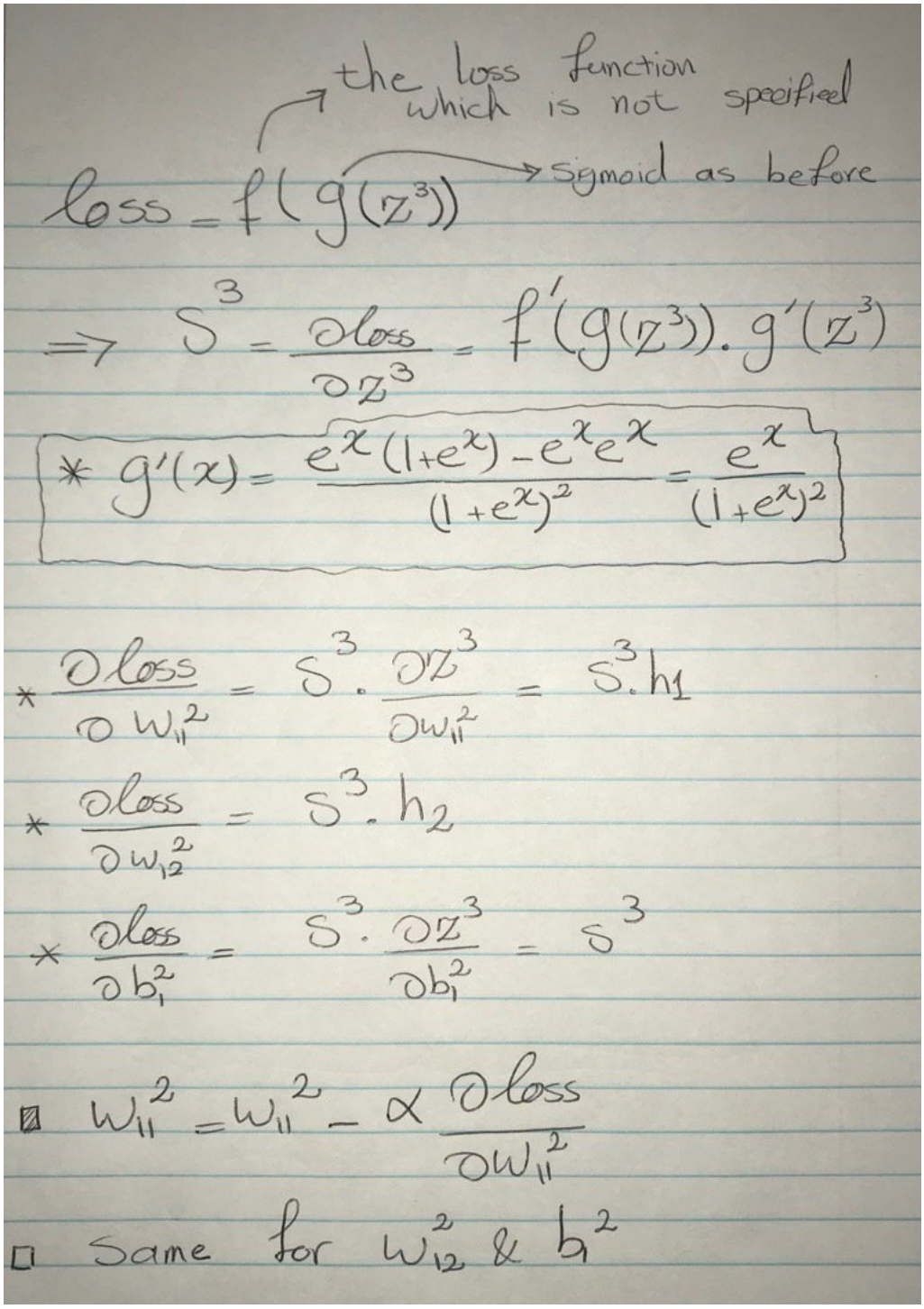
+3 points for network

+2 points for correct bias



b) (15 points) Assuming you have a sigmoid activation functions, X1 = 0.10, X2 = 0.05, initial weights between X1 and hidden node 1 = 0.2, between X1 and hidden node 2 = 0.3, X2 and hidden node 1 = 0.3, and X2 and hidden node 2 = 0.2, the weights between hidden node 1 and the output is 0.5, and the weight between hidden node 2 and the output is 0.1. Assume all biases are 0.1. Please write the formulation for the feed forward output and backward propagation for one iteration of updating weights. Note: We are not asking you to compute the values but do

expect to see the numeric values in the equations



This page intentionally left blank

$$S_1^2 = \frac{\partial loss}{\partial z_1^2} = \frac{\partial loss}{\partial z_1^3} \times \frac{\partial z_1^3}{\partial z_1^2}$$

S^3

$$z_1^3 = w_{11}^2 g(z_1^2) + \cdots \Rightarrow \frac{\partial z_1^3}{\partial z_1^2} = w_{11}^2 g'(z_1^2)$$

$$\Rightarrow S^2 = S^3 \cdot w_{11}^2 \cdot g'(z_1^2)$$

$$\frac{\partial loss}{\partial w_{11}^1} = \frac{\partial loss}{\partial z_1^2} \cdot \frac{\partial z_1^2}{\partial w_{11}^1} = S_1^2 \cdot X_1$$

$$\Rightarrow w_{11}^1 = w_{11}^1 - \alpha \frac{\partial loss}{\partial w_{11}^1} \cdots$$

Also, for weights connected to the second hidden node, we need to compute S_2^2.

c) (10 points) Autoencoder. What is an autoencoder, what is it used for, and why does it generally use a smaller number of dimensions for hidden layer compared to its input and output layers?

What is AE: (+4)

- Keywords:
  - A type of neural network
  - Encoder and decoder
  - Latent representation
  - Replicate the input

What is AE used for: (+3)

- Keywords:
  - Feature extraction
  - Dimensionality reduction
  - Pretraining

What does the hidden layer have a smaller dimensionality: (+3)

- Keywords:
  - Trivial solution possible if not smaller
  - Compact representation
  - Dimensionality reduction

d) (15 points) Convolutional Neural Network. Consider one **convolution layer** with one 2 x 2 kernel and stride 1, succeeded by a **ReLU activation** and then a 2 x 2 **max pooling layer** with stride 2. The input and the kernel are shown as below. Please calculate the final output. Be sure to show your work.

Note: The convolution takes the formula

$$y_{i,j} = \sum_{k=0}^{K-1} \sum_{l=0}^{L-1} x_{i+k,j+l} \cdot f_{k+1,l+1}$$

Where K, L are the width and height of the kernel f. i, j iterate over the height and width of the input. So when the convolution starts the first element of the kernel will overlap with the first element of the input. Please use zero-padding to make sure the output of the convolution will have the same shape as the input.

Input:

0.3	**0.2**	**-0.1**	**0.7**
-0.7	**0.6**	**0.8**	**0.3**
0.9	**-0.3**	**-0.1**	**0.4**
-0.4	**0.2**	**1.0**	**-0.7**

Kernel:

0.1	**1.0**
0.8	**-0.2**

0. attempt: 1pt

1. examine the calculation of first convolution: 2pts

- in question I use the convolution without flipping/rotating the kernel,

  which is used in deep learning frameworks like TensorFlow, PyTorch, etc.

  Actually this should be called ``correlation'' instead of ``convolution''.

  Some students used flipping before convolution. I will give credits for

  both cases.

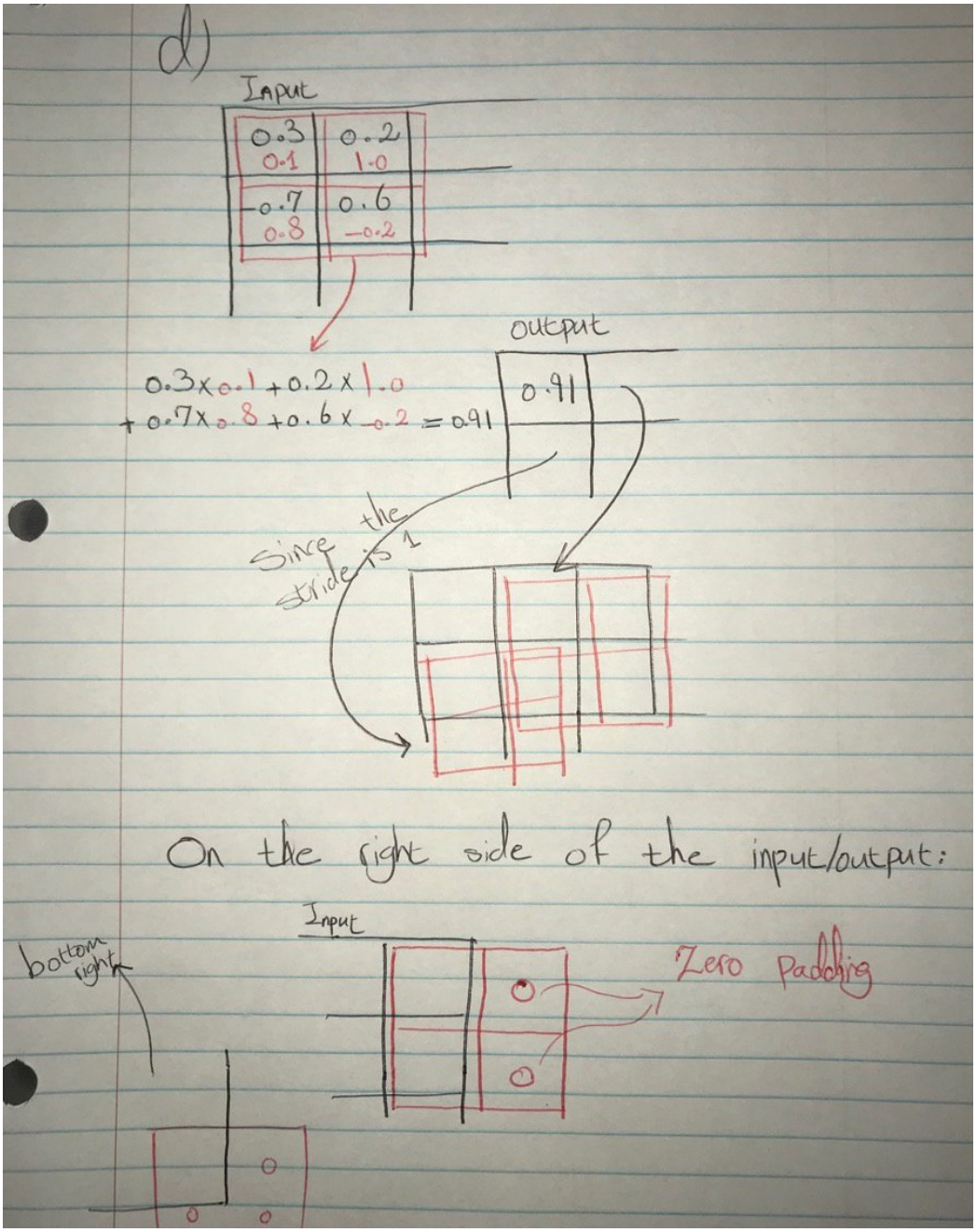
2. examine one calculation of ReLU: 1pt

3. examine one calculation of maxpooling: 1pt

4. right padding (expect them to pad zeros to the right and the bottom.): 1pt

5. right striding of maxpooling: 1pt

6. the left 3 credits will be assigned based on their attempt



Assuming we have computed the output of the CNN layer to be:

0.8	-0.5
0.1	0

Passing it through Relu, we have:

0.8	0
0.1	0

Then, max pooling: