# CS230 Midterm

Shen Gao

TOTAL POINTS

**101.5 / 124**

QUESTION 1

## Multiple Choice 14 pts

1.1 a **2 / 2**

✓ **- 0 pts Correct answer (true)**

**- 2 pts** Incorrect answer (false) or left unanswered

1.2 b **2 / 2**

✓ **- 0 pts Correct (iv)**

**- 2 pts** Incorrect (i, ii, iii, or v)

1.3 c **2 / 2**

✓ **- 0 pts Correct answer (ii)**

**- 2 pts** Incorrect answer (i, iii, iv, or v) or Unanswered

1.4 d **1 / 2**

✓ **- 0 pts Correctly selected i)**

**- 0.5 pts** Did not select i)

✓ **- 0 pts Correctly selected ii)**

**- 0.5 pts** Did not select ii)

**- 0 pts** Correctly selected iii)

✓ **- 0.5 pts Did not select iii)**

**- 0 pts** Correctly did not select iv)

✓ **- 0.5 pts Incorrectly selected iv)**

1.5 e **2 / 2**

**- 0.5 pts** Incorrectly selected i)

**- 0.5 pts** Incorrectly selected ii)

**- 0.5 pts** Did not select iii)

**- 0.5 pts** Did not select iv)

✓ **- 0 pts Entirely correct (selected just iii and iv)**

1.6 f **2 / 2**

✓ **- 0 pts Correct (true)**

**- 2 pts** Incorrect (false or unanswered)

1.7 g **0 / 2**

**- 0 pts** Correct (true)

✓ **- 2 pts Incorrect (false)**

QUESTION 2

## Short Answer 38 pts

2.1 a **1 / 2**

**- 0 pts** Correct

✓ **- 1 pts Inaccurate reasoning about iid -> minibatch gradient is empirical, unbiased estimate of full gradient**

**- 1 pts** No explanation

**- 2 pts** Incorrect

2.2 b **2 / 2**

✓ **- 0 pts Correct**

**- 1 pts** Incorrect - input data distribution is not the fundamental issue

**- 2 pts** Incorrect

**- 1 pts** The order of minibatches matters - just alternating might still work well

2.3 c **2 / 2**

✓ **- 0 pts Correct**

**- 1 pts** Incomplete explanation

**- 2 pts** Incorrect

2.4 d **3 / 3**

✓ **- 0 pts Correct**

**- 1 pts** 2/3 correct

**- 3 pts** Incorrect

2.5 e **3 / 4**

**- 0 pts** Correct

✓ **- 1 pts Reasons not distinct enough/second reason**

not entirely correct

- **2 pts** Only one valid reason provided
- **4 pts** Incorrect

### 2.6 f 1 / 2

- **0 pts** Correct
- ✓ **- 1 pts** **Incomplete explanation**
- **2 pts** Incorrect

### 2.7 g 1 / 2

- **0 pts** Correct
- **0.5 pts** Incorrect answer but good reasoning
- ✓ **- 1 pts** **Insufficient reasoning**
- **2 pts** Incorrect

### 2.8 h(i) 2 / 2

- ✓ **- 0 pts** **Correct**
- **2 pts** Incorrect

### 2.9 h(ii) 2 / 2

- ✓ **- 0 pts** **Correct**
- **2 pts** Incorrect

### 2.10 h(iii) 0 / 2

- **0 pts** Correct
- **1 pts** Partially correct
- ✓ **- 2 pts** **Incorrect**

Random search on log scale

### 2.11 i(i) 0 / 2

- **0 pts** Correct
- ✓ **- 2 pts** **Incorrect**
- **1 pts** Partially correct
- **0 pts** Click here to replace this description.

### 2.12 i(ii) 2 / 2

- ✓ **- 0 pts** **Correct**
- **2 pts** Incorrect
- **1 pts** Correct, but no/incorrect/insufficient explanation

### 2.13 j(i) 2 / 2

- ✓ **- 0 pts** **Correct**
- **0.5 pts** Minor issue in the argument
- **1 pts** Insufficient argument
- **2 pts** The argument is wrong

### 2.14 j(ii) 2 / 2

- ✓ **- 0 pts** **Correct**
- **0.5 pts** Minor issue in the argument
- **1 pts** Insufficient argument
- **2 pts** The argument is wrong

### 2.15 j(iii) 2 / 2

- ✓ **- 0 pts** **Correct**
- **0.5 pts** Minor issue in the argument
- **1 pts** Did not explain why the Adam speeds up the convergence
- **2 pts** The argument is wrong

### 2.16 k(i) 2 / 2

- ✓ **- 0 pts** **Correct**
- **2 pts** Incorrect
- **1 pts** Partially correct

### 2.17 k(ii) 0 / 3

- **0 pts** Correct
- ✓ **- 3 pts** **Incorrect**
- **2 pts** Partially correct
- **1 pts** Partially correct

QUESTION 3

## 3 Convolutional Neural Networks 12 / 12

- ✓ **- 0 pts** **Correct**
- **6 pts** Number of parameters for all layers incorrect
- **4.5 pts** Number parameters in 4/5 layers incorrect
- **3 pts** Number of parameters in non-pool layers incorrect
- **2 pts** Number of parameters in 2/5 layers incorrect
- **1 pts** Number parameters in 1/5 layers incorrect
- **6 pts** All activation volume dimensions incorrect
- **4 pts** Incorrect HxW for all Conv/Pool layers
- **1 pts** Off by one error on multiple HxW dimensions
- **1 pts** Incorrect HxW for two layers

- **0.5 pts** Incorrect HxW for one layer
- **2 pts** Activation channel dimensions incorrect
- **1.5 pts** 4/5 activation channel dimensions incorrect
- **0.75 pts** 2/5 activation channel dimensions incorrect
- **0.5 pts** FC10 dimension incorrect
- **1 pts** Small error in # parameters in multiple layers
- **0.5 pts** Small error in # parameters in one layer

QUESTION 4

## Numpy Coding 10 pts

### 4.1 1 1 / 2

- **0 pts** Correct implementation
- **0.5 pts** Minor syntax error
- ✓ **- 1 pts** **Correct implementation for the thresholding**
- **2 pts** Wrong implementation

### 4.2 2 2 / 2

- ✓ **- 0 pts** **Correct implementation**
- **0.5 pts** Minor syntax error
- **2 pts** Wrong implementation

### 4.3 3 2 / 2

- ✓ **- 0 pts** **Correct implementation**
- **0.5 pts** Minor syntax error
- **2 pts** Wrong implementation

### 4.4 4 2 / 2

- ✓ **- 0 pts** **Correct implementation**
- **0.5 pts** Minor syntax error
- **2 pts** Wrong implementation

### 4.5 5 0 / 2

- **0 pts** Correct Implementation
- **0.5 pts** Minor syntax error
- ✓ **- 1 pts** **Hard samples are not selected**
- ✓ **- 1 pts** **The probability is not normalized (did not sum up to one)**
- **2 pts** Wrong implementation

QUESTION 5

## Backpropagation 32 pts

### 5.1 a 2 / 2

- ✓ **- 0 pts** **Correct**
- **1 pts** partially correct
- **2 pts** incorrect

### 5.2 b 2 / 2

- ✓ **- 0 pts** **Correct**
- **1 pts** partially correct, one dimension wrong
- **2 pts** Incorrect

### 5.3 c(i) 2 / 2

- ✓ **- 0 pts** **Correct**
- **1 pts** Partially Incorrect
- **2 pts** Incorrect

### 5.4 c(ii) 4 / 4

- ✓ **- 0 pts** **Correct**
- **1 pts** beta > alpha correct but ratio not very close to dataset class imbalance (10:1) or values are excessively large/small (order 10^2 or 10^-2)
- **2 pts** alpha >= beta>0
- **3 pts** alpha,beta <= 0 OR attempted but specific values not given
- **4 pts** No response

### 5.5 d(i) 2.5 / 3

- **0 pts** Correct
- **0.5 pts** Generally correct, with minor errors like sign mistakes
- ✓ **- 0.5 pts** **Missing constant -1/m OR gives gradients of individual examples without including them as entries in m-vector or elements in a sum**
- **1 pts** Incorrect alpha, beta usage
- **1 pts** Incorrect y or y_hat terms
- **3 pts** no differentiation or insignificant progress

### 5.6 d(ii) 1 / 2

- **0 pts** Correct
- ✓ **- 1 pts** **Attempted but some mistake**
- **2 pts** No attempt

y_hat within the sigma functions instead of z2 :(

### 5.7 d(iii) 2 / 2

✓ **- 0 pts Correct**

**- 1 pts** minor mistake like includes extra terms or incorrect usage of W_2

**- 2 pts** doesn't include either W_2 or W_2 transposed

### 5.8 d(iv) 3 / 3

✓ **- 0 pts Correct**

**- 1.5 pts** one case incorrect (must cover both negative and positive inputs)

**- 3 pts** both cases incorrect or provides a number without indicating which case

### 5.9 d(v) 2 / 2

✓ **- 0 pts Correct**

**- 1 pts** minor mistake like extra terms or incorrect usage of x

**- 2 pts** no x term at all

### 5.10 d(vi) 3 / 3

✓ **- 0 pts Correct**

**- 1 pts** minor mistakes like missing some terms or sign/shape issues (check that it must be D_a1 by D_x) or not recognizing that gradients computed in various parts are example-dependent (see common mistake below and solution)

**- 2 pts** Doesn't attempt to combine gradients from previous parts OR many minor mistakes

**- 3 pts** incorrect

**- 0 pts** common mistake: if J w.r.t y_hat was calculated as a scalar (i.e. using summation) in the first part, can't directly include this term in chain rule. must include summation to average over individual examples (see solution). iff it was calculated as an m-vector (meaning each entry is gradient for one example), okay to use this delta -- we'll be kind and assume sigmoid gradient is a diagonal matrix, summation can happen in another step, like stacking m x^i row vectors in the last step, etc.. demonstrated understanding of chain rule is the most important factor for earning majority credit in this question

### 5.11 d(vii) 2 / 2

✓ **- 0 pts Correct**

**- 0.5 pts** minor mistake in reg terms (missing 1 term, extra term like bias regularization, notational issues like simply writing W^2 doesn't square, sign issues etc.)

**- 1 pts** Doesn't correctly include original loss function

**- 1 pts** major reg term mistakes or many minor mistakes or regularization terms completely incorrect

**- 2 pts** incorrect

### 5.12 d(viii) 3 / 3

✓ **- 0 pts Correct**

**- 1 pts** minor mistake (like incorrect gradient of norm, sign issues, not including gradient of loss, missing eta in some places)

**- 2 pts** no update rule of the form new = old +- eta * (gradient) OR multiple mistakes

**- 3 pts** incorrect

### 5.13 d(ix) 2 / 2

✓ **- 0 pts Correct**

**- 1 pts** reasonable attempt but only partially correct; should mention sparsity in L1 or L2 resulting in smaller (non-zero) magnitude weights since it penalizes much more for larger weights

**- 2 pts** incorrect

QUESTION 6

## Fun with Activation Functions 11 pts

### 6.1 a(i) 3 / 3

✓ **- 0 pts Correct with correct explanation**

**- 1 pts** Correct with incomplete explanation

**- 2 pts** Correct answer with incorrect/absent explanation

**- 3 pts** Blank/Incorrect

6.2 a(ii) **3 / 4**

- **0 pts** Correct

✓ **- 1 pts** **Correct, with right idea but a couple incorrect weights/biases**

- **2 pts** Correct, with several incorrect weights/biases
- **3 pts** Correct, with missing weights/biases
- **3 pts** Incorrect, but gave a valid reason it's not learnable
- **4 pts** Blank/Incorrect

6.3 b(i) **2 / 2**

✓ **- 0 pts** **Correct**

- **1 pts** Incorrect on some parts of domain
- **2 pts** Blank/Incorrect

6.4 b(ii) **2 / 2**

✓ **- 0 pts** **Correct**

- **2 pts** Blank/Incorrect

QUESTION 7

## Bonus 7 pts

7.1 (i) **3 / 3**

✓ **- 0 pts** **Correct (in the form of either x or y is fine)**

- **2 pts** Correctly applied the chain rule but final outcome is wrong or still contains the differentiation sign
- **3 pts** No answer; wrong anser; got wrong at the chain rule step
- **1 pts** Wrong sign

7.2 (ii) **0 / 2**

- **0 pts** Correct answer

✓ **- 2 pts** **Wrong answer or blank**

- **1 pts** Wrong sign

7.3 (iii) **1 / 2**

- **0 pts** Correct
- **2 pts** Wrong answer

✓ **- 1 pts** **One of the conditions is wrong**

**END OF PAPER**

# CS230: Deep Learning

Fall Quarter 2019

Stanford University

## Midterm Examination

180 minutes

| | Problem | Full Points | Your Score |
|---|---|---|---|
| 1 | Multiple Choice | 14 | |
| 2 | Short Answers | 38 | |
| 3 | Convolutional Architectures | 12 | |
| 4 | Numpy Coding | 10 | |
| 5 | Backpropagation | 32 | |
| 6 | Fun with Activation Functions | 11 | |
| 7 | Softmax (Bonus) | 7 | |
| Total | | 124 | |

The exam contains 27 pages including this cover page.

- This exam is **closed book i.e. no laptops, notes, textbooks, etc. during the exam**. However, you may use one A4 sheet (front and back) of notes as reference.
- In all cases, and especially if you're stuck or unsure of your answers, **explain your work, including showing your calculations and derivations!** We'll give partial credit for good explanations of what you were trying to do.

Name: Shen Gao

SUNETID: shengao @stanford.edu

**The Stanford University Honor Code:**
I attest that I have not given or received aid in this examination, and that I have done my share and taken an active part in seeing to it that others as well as myself uphold the spirit and letter of the Honor Code.

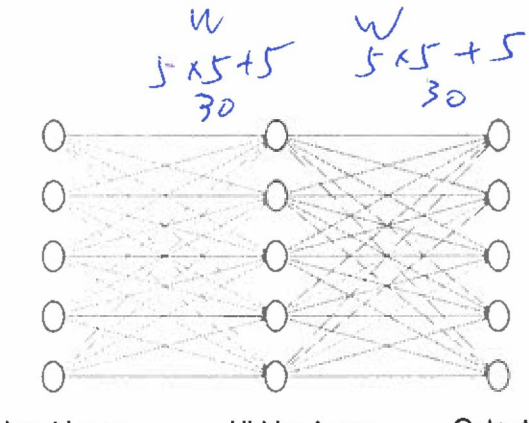
Signature: 

## Question 1 (Multiple Choice Questions, 14 points)

For each of the following questions, circle the letter of your choice. There is only ONE correct choice unless explicitly mentioned. No explanation is required.

(a) **(2 points)** A 2-layer neural network with 5 neurons in each layer has a total of 60 parameters (i.e. weights and biases)

(i) True
(ii) False




(b) **(2 points)** Given an input volume of shape (10, 10, 3), you consider using one of the two following layers:

- Fully-connected layer with 2 neurons, with biases
- Convolutional layer with three 2x2 filters (with biases) with 0 padding and a stride of 2

If you use the fully-connected layer, the input volume is "flattened" into a column vector before being fed into the layer. What is the **difference** in the number of trainable parameters between these two layers?

(i) The fully-connected layer has 566 fewer parameters
(ii) The convolutional layer has 566 fewer parameters
(iii) The convolutional layer has 564 fewer parameters
(iv) The convolutional layer has 563 fewer parameters
(v) None of the above

(c) **(2 points)** You decide to use the convolutional layer. What will be the size of the output of the convolutional layer described above?

(i) (5, 5, 4)
(ii) (5, 5, 3)
(iii) (4, 4, 4)
(iv) (4, 4, 3)
(v) None of the above

(d) **(2 points)** Assume the vectors v and u are defined as numpy arrays. The vectors are multiplied together using the * operator: v*u. Which of the following shapes are valid, meaning the operation will succeed without errors? (Circle all that apply)

(i) v = (5, 1), u = (5, 1)
(ii) v = (1, 5), u = (50, 5)
(iii) v = (1, 1), u = (50, 5)
(iv) v = (1, 5), u = (1, 50)
(v) None of the above

(e) **(2 points)** Which of the following is true about the vanishing gradient problem? (Circle all that apply)

(i) Tanh is usually preferred over sigmoid because it doesn't suffer from vanishing gradients
(ii) Vanishing gradient causes deeper layers to learn more slowly than earlier layers
(iii) Leaky ReLU is less likely to suffer from vanishing gradients than sigmoid
(iv) Xavier initialization can help prevent the vanishing gradient problem
(v) None of the above

batch norm.

(f) **(2 points)** The backpropagated gradient through a tanh non-linearity is always smaller or equal in magnitude than the upstream gradient. (Recall: if z = tanh(x) then $\frac{\partial z}{\partial x} = 1 - z^2$ )

(i) True
(ii) False

$W \leftarrow W - \alpha \frac{\partial z}{\partial W}$

$W - 0.1(1 - z^2)$

$W - \alpha + \alpha z^2$

$0.1 \quad 0.5^2$

(g) **(2 points)** Consider a trained logistic regression. Its weight vector is W and its test accuracy on a given data set is A. Assuming there is no bias, dividing W by 2 won't change the test accuracy.

(i) True
(ii) False

## Question 2 (Short Answers, 38 points)

The questions in this section can be answered in 2-4 sentences. Please be concise in your responses.

(a) **(2 points)** The gradient estimated during a step of mini-batch gradient descent has on average a lower bias when the data is i.i.d. (independent and identically distributed). True or False? Explain why.

True. I.I.D makes each batch contain similar amount of information making each batch contribute similarly to gradient descent. Non-iid batches tend to have more variability in grad descent.

(b) **(2 points)** You have two data sets of similar size for a binary classification task. However, one contains almost entirely positive examples, and the other contains only negative examples. You would like to use both sets to train your model. Describe a scenario in which combining these two data sets could lead to a failure of the model to learn.

use stochastic gradient descent with two data sets stacked w/o randomly ~~set~~ shuffling. When data changes from positive to negative, Loss function would go up dramatically because the single-unit batch hasn't seen negative examples

(c) **(2 points)** Why do the layers in a deep architecture need to be non-linear?

if not non-linear, deep networks w/ multiple layers will only represent linear relationships, equivalent to a single layer network. not complex enough to capture all possible functions. ($\hat{y} = W_3 W_2 W_1 x$ for example).

(d) **(3 points)** Cite 3 layers commonly used in a convolutional neural network.

① Conv layer: convolution layer w/ filters
② Pooling layer: ~~max~~ e.g. max or avg pooling
③ ~~Batch normalization layers~~ 1×1 conv layers w/ filtersize 1×1 (special case of ①)
④ Fully-connected layer: image is flattened as 1 vector

(e) **(4 points)** Alice recommends the use of convolutional neural networks instead of fully-connected networks for image recognition tasks since convolutions can capture the ~~spatial~~ relationship between nearby image pixels.

Bob points out that fully-connected layers can capture spatial information since each neuron is connected to all of the neurons in the previous layer.

Both are correct, but describe two reasons we should prefer Alice's approach to Bob's.

① FC requires a lot more data to train,
② generally has more parameters than Conv layers.

(f) **(2 points)** You're solving a binary classification task. The final two layers in your network are a ReLU activation followed by a sigmoid activation. What will happen?

~~Relu(σ(z))~~
σ(Relu(z)): Relu(z) ∈ [0, +∞]
Since sigmoid is monotonically increasing, σ(0) = 0.5
therefore result is ∈ [0.5, 1)
if threshold is > 0.5 it will only return 1 as prediction.

(g) **(2 points)** You are searching the best learning rate for your model. You decide to test the following values between 0.01 and 1:

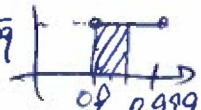
- learning rate = 0.01
- learning rate = 0.16
- learning rate = 0.21
- learning rate = 0.84
- learning rate = 0.94

Is that a good method? Explain why.

0.84/0.94 is almost always too large.
Better to plot loss function against # of iteration/epochs

if loss decrease slowly, then use larger learning rate,
if loss oscillate a lot then learning rate is too big

(h) You are randomly (with a uniform distribution) searching for the best $\beta_1$ parameter for the Adam optimizer in the range of values [0.9; 0.999].




(i) **(2 points)** What is the probability of choosing $\beta_1$ such that $\beta_1 < 0.94$?

$\beta_1 \sim U(0.9, 0.999)$ $p(\beta_1) = \frac{1}{0.999-0.9} = \frac{1}{0.099}$ $P(\beta_1 < 0.94) = (0.94-0.9) \cdot p(\beta_1)$

(ii) **(2 points)** What is the probability of choosing $\beta_1$ such that $\beta_1 > 0.99$?

$P(\beta_1 > 0.99) = 1 - P(\beta_1 \leq 0.99) = 1 - ((0.99-0.9) \cdot p(\beta_1))$
$= 1 - (0.09 \times \frac{1}{0.099}) = 1 - \frac{0.090}{0.099} = \frac{9}{99} = \frac{1}{11}$
$= \frac{0.04}{0.099} = \frac{40}{99}$

(iii) **(2 points)** Propose a better search method to find the best $\beta_1$ hyperparameter value.

Select a few values ~~0.9, 0.99, 0.999~~ and train model and see dev set performance. ~~if midpoint is better than 0.99 and run through dev set~~ find β, w/ best dev set performance

(i) You're solving a binary classification task.

(i) **(2 points)** You first try a logistic regression. You initialize all weights to 0.5. Is this a good idea? Briefly explain why or why not.

That seems rather arbitrary. ~~Could lead to issues w/ improper initialization. Use Xavier~~
this will ~~cause~~ not "break symmetry" and all neurons will change by the same amount — equivalent to a 1-neuron network. Use random weights instead. Better yet, use Xavier initialization / He initialization / Glorot for better ~~weight~~ activation variance
$\mathcal{N}(0, 1/n^{[l-1]})$ $N(0, 2/n^{[l-1]})$ $N(0, 2/(n^{[l-1]} + n^{[l]}))$

(ii) **(2 points)** Then, you try a 4-layer neural network. You initialize all weights to 0.5. Is this a good idea? Briefly explain why or why not.

Same answer as above. Except w/ a deeper network the vanishing gradient issue is much more pronounced and using above initialization methods become more essential

(j) Variations of Gradient Descent

(i) **(2 points)** Describe one advantage of using mini-batch gradient descent instead of full-batch gradient descent.

to update weight once

FB(full-batch) takes a long time if data is large mini batch updates weight much quicker while also utilizing vector operation (advantage over Stochast. gradient descent)

(ii) **(2 points)** Describe one advantage of using mini-batch gradient descent instead of stochastic gradient descent with batch size 1.

① mini-batch converges to minimum with a much smaller error band because of law of large numbers

② mini-batch calculates each batch w/ vectorized operation, much faster than iterating through every training example (in SGD).

(iii) **(2 points)** Describe one advantage of using Adam optimizer instead of vanilla gradient descent.

Stablizes gradients by normalizing gradient by roging average mean of grad and ~~gradient~~ variance of gradient. Reduces the impact on updates by large variations in ~~grad~~ mini-batch gradients

(k) Consider a model trying to learn an encoding of some input $x \in \mathbb{R}$. The goal is to encode the input $x$ using $z = w_1 x \in \mathbb{R}$, then accurately reconstruct the original $x$ from the encoded representation using $\hat{x} = w_2 z \in \mathbb{R}$. Here, $(w_1, w_2) \in \mathbb{R} \times \mathbb{R}$. The model is trained with the squared reconstruction error:

$$L(W) = \frac{1}{n}\sum_{i=1}^{n}(x^{(i)} - w_2 w_1 x^{(i)})^2$$

$w_2 w_1 = 1$

$w_1 = \frac{1}{w_2}$

(i) **(2 points)** What is the set of solutions for $w_1$ and $w_2$ which makes loss zero?

$x^{(i)} = w_2 w_1 x^{(i)} \Rightarrow w_1 = \frac{1}{w_2}$

$w_1 = w_2 = 1$

Or $w_1 = w_2 = -1$ or $(w_1, \frac{1}{w_1})$ for $\forall w_1$ iff $w_1 \neq 0$

(ii) **(3 points)** Does the loss have a saddle point? Where?

~~Yes.~~ There are many values L can reach global minimum 0, but they are all in the same direction $\langle w_1, \frac{1}{w_1} \rangle$

Not a saddle point.

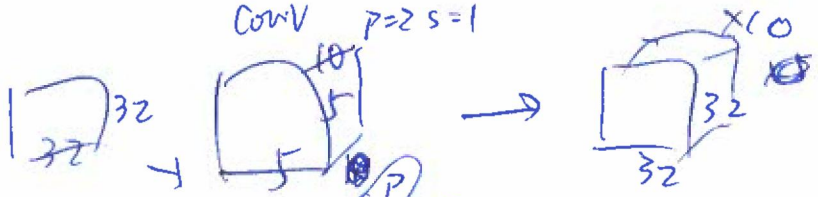
## Question 3 Convolutional Architectures (12 points)

Consider the convolutional neural network defined by the layers in the left column below. Fill in the shape of the output volume and the number of parameters at each layer. You can write the shapes in the numpy format (e.g. (128,128,3)).

Notation:

- CONV5-N denotes a convolutional layer with N filters with height and width equal to 5. Padding is 2, and stride is 1.
- POOL2 denotes a 2x2 max-pooling layer with stride of 2 and 0 padding.
- FC-N denotes a fully-connected layer with N neurons

| Layer | Activation Volume Dimensions | Number of parameters |
|---|---|---|
| Input | 32×32 × 1 | 0 |
| CONV5-10 | 32×32×10 | 260 |
| POOL2 | 16×16×10 | 0 |
| CONV5-10 | 16×16×10 | 2510 |
| POOL2 | 8×8×10 | 0 |
| FC10 | 10×1 | 6410 |




$\frac{32+2\times2-5}{1}+1$

$32+4-5+1=32$

$\frac{32-2}{2}+1$

$15+1$

16

$(5\times5+1)\times10$

26

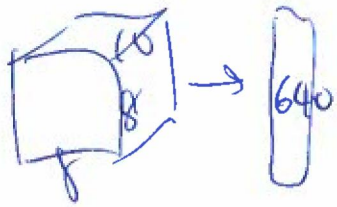
$(5\times5\times10+1)\times10$

$250+1$

$251\times10$

$\frac{16+2\times2-5}{1}+1$

$16+4-5+1$




$10\times640+10$

$641\times10$

## Question 4 (Numpy Coding, 10 points)

In this question, you will mine hard examples from a large training set. Examples for which the model's predictions are very different from the ground truth are called "hard examples." Your goal is to sample hard examples after each training epoch, so that they could be passed to your model for faster training and better performance.
Below, we use 0 and 1 to label the negative and positive classes, respectively. We use y_hat to denote predictions (such that y_hat > 0.5 indicates predictions for the positive class) and y to denote true labels.
Fill in the blanks in the code below. Your code should compile, so please use correct syntax.

```
import numpy as np

def mine_hard_examples(X, y, y_hat, min_sample_size):
    """
    y_hat -- numpy array of shape (m,); model predictions
    y -- numpy array of shape (m,); ground-truth labels
    """
    ### START CODE HERE ###
    # 1) Compute a vector of booleans indicating whether
    # or not the examples are misclassified
    misclassified = 

    # 2) Compute the absolute difference between y and y_hat
    absolute_difference = 

    # 3) Create a boolean vector indicating whether or not
    # the examples are hard given a threshold of 0.7
    # (i.e., misclassified with absolute_difference > 0.7)
    mask = 

    # 4) Compute the sample size selected by the mask array
    sample_size = 

    assert sample_size < len(mask)
    if sample_size < min_sample_size:
        # 5) Randomly flip 0's in mask to ensure we return
        # min_sample_size examples. We'll use flip_probs
        # to select supplemental indices to set in mask;
        # recall that flip_probs must sum to 1
        flip_probs = 

        ### END CODE HERE ###
        idx = np.random.choice(
            len(mask),
```

```
            min_sample_size - sample_size,
            replace=False,
            p=flip_probs)
        mask[idx] = ~mask[idx]
    return X[mask], y[mask]
```

## Question 5 (Backpropagation , 32 points)

You're trying to classify RGB images in giraffe present (1) and giraffe absent (0) using a deep neural network. Unfortunately, your data set is imbalanced. The class counts are:

| | |
|---|---|
| 2000 images with a giraffe | 1 |
| 200 examples with no giraffe | 0 |

(a) **(2 points)** Name two data augmentation techniques you could use to help address the class imbalance problem.

increase number of "no giraffe" images by:
① horizontal flip image
② slight rotation of image
③ increase saturation/contrast
④ blurr the image

Instead of data augmentation, you want to experiment with other techniques. Here's the architecture of your network:

$$z_1 = W_1 x^{(i)} + b_1 \quad (D_{a_1} \times m)$$
$$a_1 = \text{ReLU}(z_1) \quad (D_{a_1} \times m)$$
$$z_2 = W_2 a_1 + b_2 \quad (1 \times m)$$
$$\hat{y}^{(i)} = \sigma(z_2) \quad (1 \times m)$$
$$L^{(i)} = \alpha * y^{(i)} * \log(\hat{y}^{(i)}) + \beta * (1 - y^{(i)}) * \log(1 - \hat{y}^{(i)})$$
$$J = -\frac{1}{m} \sum_{i=1}^{m} L^{(i)}$$

The dimensions are as follows: $\hat{y}^{(i)} \in \mathbb{R}$, $y^{(i)} \in \mathbb{R}$, $x^{(i)} \in \mathbb{R}^{D_x \times 1}$, $W_1 \in \mathbb{R}^{D_{a_1} \times D_x}$, $W_2 \in \mathbb{R}^{1 \times D_{a_1}}$. Note $m$ is the size of the dataset and that the RGB images are flattened into vectors of length $D_x$ before being fed into the network.

(b) **(2 points)** What are the dimensions of $b_1$ and $b_2$?

$b_1 : D_{a_1} \times 1$

$b_2 : 1 \times 1$

(c) Let's focus on the hyperparameters $\alpha$ and $\beta$ of the loss function.

(i) **(2 points)** Why are $\alpha$ and $\beta$ useful?

~~$f_1 = \log \hat{y}^{(i)}$ (positive class) $f_2 = \log(1-\hat{y}^{(i)})$ (negative class)~~

These help w/ class imbalance because you could assign higher penalty to negative class (under-represented) s.t. $L = \sum$(overrepresented class, underrepresented) have similar influence to loss function.

(ii) **(4 points)** What is a reasonable pair of values for $(\alpha, \beta)$? Provide specific values for these weightings.

positive class is 10 times more than negative class therefore $\beta = 10\alpha$, could use $(1, 10)$ for $(\alpha, \beta)$

(d) Backpropagation,

Hint: For these derivations, it will save you time to refer to earlier answers using symbols. For example, gradients computed in an earlier part can be denoted $\delta_1$, $\delta_2$, etc.

(i) **(3 points)** What is $\frac{\partial J}{\partial \hat{y}}$?

$J = -\frac{1}{m}\sum_{i=1}^{m} L^{(i)} = -\frac{1}{m} \vec{L} \cdot \vec{1}$

$\vec{L} = (L^{(1)}, L^{(2)} \dots L^{(m)})_{1\times m}$

$\vec{L} = \alpha \vec{y} \odot \log \vec{\hat{y}} + \beta(1-\vec{y}) \odot \log(1-\vec{\hat{y}})$

$\vec{y} = (y^{(1)}, y^{(2)} \dots y^{(m)})_{1\times m}$

$\vec{\hat{y}} = (\hat{y}^{(1)}, \dots \hat{y}^{(m)})_{1\times m}$

$\frac{\partial J}{\partial \hat{y}} = \frac{-\frac{1}{m}(\alpha \vec{y} \odot \log \vec{\hat{y}} + \beta(1-\vec{y}) \odot \log(1-\vec{\hat{y}}))\vec{1}}{\partial \hat{y}}$

$= -\frac{1}{m}\left(\alpha \vec{y} \frac{1}{\hat{y}} + \beta(1-y)\frac{1}{1-\hat{y}}(-1)\right)$

$= -\frac{\alpha \vec{y}}{m \hat{y}} + \frac{\beta}{m}\frac{1-\vec{y}}{1-\hat{y}}$

(ii) **(2 points)** What is $\frac{\partial \hat{y}^{(i)}}{\partial z_2}$?

$\hat{y} = \sigma(z_2)$ ~~$= \sigma(W_2 a_1 + b_2)$~~

$\frac{\partial \hat{y}}{\partial z_2} = \sigma'(z_2) = \sigma(\hat{y})[1-\sigma(\hat{y})]$

$\frac{\partial \hat{y}^{(i)}}{\partial z_2} = \sigma(\hat{y}^{(i)})[1-\sigma(\hat{y}^{(i)})]$

(iii) **(2 points)** What is $\frac{\partial z_2}{\partial a_1}$? $z_2 = W_2 a_1 + b_2$

$$\frac{\partial z_2}{\partial a_1} = W_2$$

(iv) **(3 points)** What is $\frac{\partial a_1}{\partial z_1}$? $a_1 = ReLu(z_1)$

$$\frac{\partial a_1}{\partial z_1} = \begin{cases} 1 & z_1 > 0 \\ 0 & z_1 < 0 \end{cases}$$

$$= 1\{z_1 > 0\}$$

(iv) **(2 points)** What is $\frac{\partial z_1}{\partial W_1}$? $z_1 = W_1 X + b_1$

$$\frac{\partial z_1}{\partial W_1} = X$$

$m \times D_x$

$n_{a_1} \times 1$

$W_2$ $1 \times m$ $*$ $1 \times m$ $X^T$

(v) **(3 points)** What is $\frac{\partial J}{\partial W_1}$? $\to n_{a_1} \times D_x$

Hint: reuse work from previous parts.

$$\frac{\partial J}{\partial W_1} = \frac{\partial J}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_1} \cdot \frac{\partial a_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial W_1}$$

$$= \underbrace{\left(-\frac{\alpha}{m}\frac{y}{\hat{y}} + \frac{\beta}{m}\frac{1-y}{1-\hat{y}}\right)}_{1 \times m} \underbrace{\odot \sigma(\hat{y})(1-\sigma(\hat{y}))}_{1 \times m} \cdot \underbrace{W_2}_{1 \times n_{a_1}} \underbrace{\odot 1\{z_1 > 0\}}_{1 \times m} \cdot \underbrace{X}_{D_x \times m}$$

$\Rightarrow$ reorganize by dimension

$$= W_2 \cdot \left(\frac{\beta}{m}\frac{1-y}{1-\hat{y}} - \frac{\alpha}{m}\frac{y}{\hat{y}}\right) \odot \sigma(\hat{y}) \odot (1-\sigma(\hat{y})) \odot 1\{z_1 > 0\} \cdot X^T$$

$\lambda \vec{1}^T$ $\gamma \vec{1}^T$

(vi) **(2 points)** You decide to to add L2 regularization to this model. Write your new cost function. Assume that the value of any regularization constant(s) is 1.

$J_{L^2} = \frac{1}{m} \vec{L} \vec{1} + \lambda \|W_1\|^2 \vec{1} + \gamma \|W_2\|^2 \vec{1}$

where $\vec{L} = (L^{(1)}, L^{(2)}, \ldots, L^{(m)})$ $L$'s definition same as before

$\|W_1\|^2$ is element wise square

$\vec{1} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}_{m \times 1}$, $\lambda$ and $\gamma$ are penalty scalars for $W_1$ and $W_2$ in this case $\lambda = \gamma = 1$ (by assumption of question)

(vii) **(3 points)** Using this new cost function, write down the update rule for $W_1$. Hint: You should reuse some of your work from previous parts. Assume you are using gradient descent without optimizers. Use $\eta$ as your learning rate.

~~$W_1 = W_1 - \eta \frac{\partial J}{\partial W_1}$~~

$W_1 = W_1 - \eta \frac{\partial J}{\partial W_1} - 2\eta \cdot W_1$

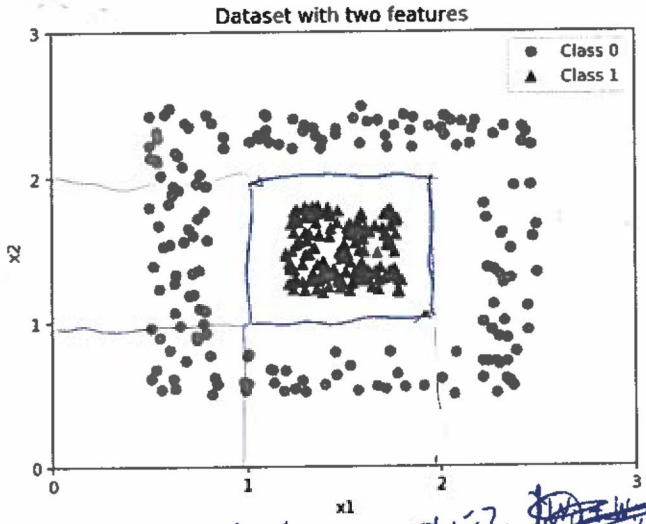
Where $\frac{\partial J}{\partial W_1}$ is defined in part (v)

(vii) **(2 points)** Suppose you used L1 regularization instead. How would you expect the weights learned using L1 regularization to differ from those learned using L2 regularization?

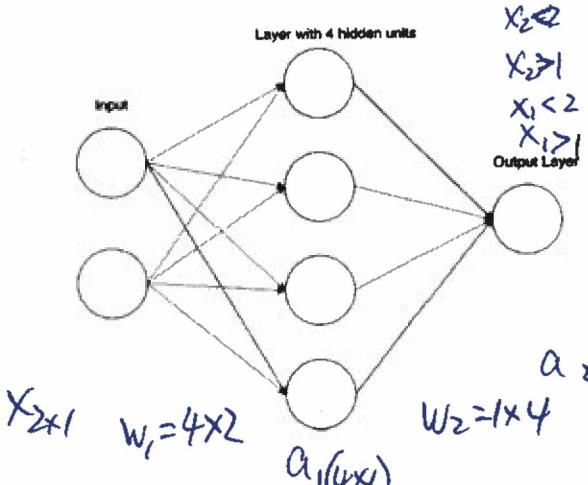
L1 encourages sparsity, therefore many weights will be forced to 0 whereas ~~L1~~ L2 have small weights for ~~the~~ corresponding weights.

## Question 6 (Fun with Activation Functions, 11 points)

(a) You have a dataset where each example contains two features, x1 and x2, and a binary label. Here's a plot of the dataset:




You want to develop a model to perform binary classification. Suppose you're using a small neural network with the architecture shown below.




Below is a precise definition of the network. Note $h$ is the activation function, $w_{i,j}$ denotes the $j$th weight in the $i$th hidden unit in the network, and $w_{output,j}$ denotes the $j$th weight in the output layer. The biases follow similar rules.

$$a_i = h(w_{i,1} * x_1 + w_{i,2} * x_2 + b_i)$$

$$a_i = h(w_{i,1} x_1 + w_{i,2} x_2 + b_i)$$

$$\text{output} = f\Big(\sum_{j=1}^{4} (a_j * w_{\text{output},j}) + b_{\text{output}}\Big)$$

Note that $h$ is the activation function for the hidden units and $f$ is the activation function for the output layer. For all of these questions, $f$ is defined as:

$$f(x) = \begin{cases} 0 & x < 0 \\ 1 & x >= 0 \end{cases}$$


(i) **(3 points)** If you used the function $h(x) = c * x$ for some $c \in \mathbb{R}$, is it possible for this model to achieve perfect accuracy on this dataset? If so, provide a set of weights that achieves perfect accuracy. If not, briefly explain why.

not possible since $h(x) = cx$ is linear

model is output $= \begin{cases} 1 & w_2(w_1 x + b_1) + b_2 \geq 0 \\ 0 & w_2(w_1 x + b_1) + b_2 < 0 \end{cases}$ → decision boundary is a plane

As seen the data is best modeled by a rectangle, need non-linearity to model this.

(ii) **(4 points)** Now assume $h$ is a modified version of the sign function:

$$h(x) = \begin{cases} 0 & x < 0 \\ 1 & x >= 0 \end{cases}$$

$2 < x_1 + x_2 < 4$

$w_{i,1} + w_{i,2} + b_i < a_i < 2w_{i,1} + 2w_{i,2} + b_i$

$1 < x < 2$

$w_1 + b_1 < a_i < 2w_1 + b_1$

Is it possible for this model to achieve perfect accuracy? If so, provide a set of weights that achieves perfect accuracy. If not, briefly explain why.

Yes. $w_2 \, 1\{w_1 x + b_1 \geq 0\} + b_2 \geq 0$ $\quad x = \begin{bmatrix} x_1 \in (1,2) \\ x_2 \in (1,2) \end{bmatrix}$

output 1 when $\begin{cases} w_2 \cdot 1 + b_2 \geq 0 \rightarrow w_2 \geq -b_2 \\ w_2 \cdot 0 + b_2 \geq 0 \rightarrow b_2 \geq 0 \end{cases}$ & $\begin{cases} a_i \geq 0 \\ w_1 x + b_1 < 0 \end{cases}$

output 0 when $\begin{cases} w_2 \cdot 1 + b_2 < 0 \\ w_2 \cdot 0 + b_2 < 0 \end{cases}$

set 4 boundaries:

output $= \begin{cases} 1 & x_1 \in (1,2) \;\&\; x_2 \in (1,2) \\ 0 & \text{elsewhere} \end{cases}$

$z = w_1 x + b$ $\quad a = h(z)$

$W_1 = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}$ $\quad b_1 = \begin{bmatrix} -2 \\ -1 \\ -2 \\ -1 \end{bmatrix}$ for $x_1 \in (1,2)$, $x_2 \in (1,2)$ $\quad \begin{bmatrix} <0 \\ >0 \\ <0 \\ >0 \end{bmatrix}$ $\quad \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$

$f(w_2 a + b_2) = 1$

$w_2 \cdot 0 + b_2 \geq 0 \Rightarrow b_2 \geq 0$

$w_2 + b_2 \geq 0 \Rightarrow w_2 \geq -b_2$

$w_2 \cdot 0 + b_2 \geq 0 \Rightarrow b_2 \geq 0$

$w_2 + b_2 \geq 0 \Rightarrow w_2 \geq -b_2$

conclusion: $W_1 = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}$ $\quad b_1 = \begin{bmatrix} -2 \\ -1 \\ -2 \\ -1 \end{bmatrix}$ $\quad b_2 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ $\quad W_2 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$

would give perfect accuracy.

(b) Recall the activation functions ReLU:

$$\text{ReLU}(x) = \max(0, x)$$

(i) **(2 points)** Consider an alternative to ReLU called Exponential Linear Unit (ELU).

$$\text{ELU}(x) = \begin{cases} x & x >= 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

What is the gradient for ELU?

$\frac{\partial ELU(x)}{\partial x} = \begin{cases} x \geq 0 & 1 \\ x < 0 & \alpha e^x \end{cases}$

(ii) **(2 points)** Name one advantage of using ELU over ReLU.

$\frac{\partial ELU(x)}{\partial x}$ is always positive, no longer suffer vanishing gradient problem (for $|\alpha| \geq 1$) and will not squash neurons to 0 (causing dead neurons).

## Question 7 (Bonus, Softmax, 7 points)

**This is a bonus question. Please spend your time wisely.**

Softmax takes in an $n$-dimensional vector $x$ and outputs another $n$-dimensional vector $y$:

$$y_i = \frac{e^{x_i}}{\sum_k e^{x_k}}$$

In this question we're going to compute the gradient of y with respect to x. Let $\delta_{ij} = \frac{\partial y_i}{\partial x_j}$.

(i) **(3 points)** Derive an expression for $\delta_{ii}$.
Hint: Recall the Quotient Rule of Calculus: Let $h(x) = \frac{f(x)}{g(x)}$. Then

$$\frac{\partial h}{\partial x} = \frac{\frac{\partial f(x)}{\partial x} * g(x) - \frac{\partial g(x)}{\partial x} * f(x)}{(g(x))^2}$$

$\delta_{ii} = \frac{\partial y_i}{\partial x_i} = \frac{\partial}{\partial x_i} \frac{e^{x_i}}{\sum_k e^{x_k}}$

$= \frac{e^{x_i}(\sum_k e^{x_k}) - e^{2x_i}}{(\sum_k e^{x_k})^2}$

(ii) **(2 points)** Now derive an expression for $\delta_{ij}$, where $i \neq j$.

$\delta_{ij}\ (i \neq j) = \frac{\partial}{\partial x_j} \left( \frac{e^{x_i}}{\sum_k e^{x_k}} \right) = 0$

(iii) **(2 points)** Write a general expression for $\delta_{ij}$. Hint: Feel free to use curly brace notation to distinguish between the two cases where $i = j$ and $i \neq j$. For a concrete example, see how ELU was defined in the previous question.

$$\delta_{ij} = \frac{e^{x_i}(\sum_k e^{x_k}) - e^{2x_i}}{(\sum_k e^{x_k})^2} \cdot \mathbb{1}\{i=j\}$$

$\mathbb{1}\{\cdot\}$ is indicator function where value is 1 when $\{\cdot\}$ condition is met and 0 elsewhere

**Extra Page 1/6**

**Extra Page 2/6**

**Extra Page 3/6**

**Extra Page 4/6**

**Extra Page 5/6**

**Extra Page 6/6**

**END OF PAPER**

output = 1 : $w_2 a_1 + b_2 \overset{\nearrow 1\{z_1>0\}}{}$

$w_{1,1} + w_{i,2} + b_i < a_1 < 2w_{i,1} + 2w_{i,2} + b_i$

$w_1 + b_1 \leq 0$