CS230: Deep Learning

Winter Quarter 2018

Stanford University

Midterm Examination

180 minutes

	Problem	Full Points	Your Score
1	Multiple Choice	7	
2	Short Answers	22	
3	Coding	7	
4	Backpropagation	12	
5	Universal Approximation	19	
6	Optimization	9	
7	Case Study	25	
8	AlphaTicTacToe Zero	11	
9	Practical industry-level questions	8	
Total		120	

The exam contains 33 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: _____

SUNETID: _____@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, 7 points)

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

(a) (1 point) You want to map every possible image of size 64×64 to a binary category (cat or non-cat). Each image has 3 channels and each pixel in each channel can take an integer value between (and including) 0 and 255. How many bits do you need to represent this mapping?

(i) $256^{3^{64\times64}}$

(ii) $256^{3\times64\times64}$

(iii) $(64 \times 64)^{256\times3}$

(iv) $(256 \times 3)^{64\times64}$

(b) (1 point) The mapping from question (a) clearly can not be stored in memory. Instead, you will build a classifier to do this mapping. Recall the simple single hidden layer classifier you used in the assignment on classifying images as cat vs non-cat. You use a single hidden layer of size 100 for this task. Each weight in the $W^{[1]}$ and $W^{[2]}$ matrices can be represented in memory using a float of size 64 bits. How many bits do you need to store your two layer neural network (you may ignore the biases $b^{[1]}$ and $b^{[2]}$)?

(i) $64 \times ((256 \times 3 \times 100) + (64 \times 64 \times 1))$

(ii) $64 \times ((64 \times 64 \times 3 \times 100) + (100 \times 1))$

(iii) $64 \times ((256^3 \times 64 \times 64 \times 100) + (100 \times 64))$

(iv) $64 \times (256 \times 3 \times 64 \times 64 \times 100)$

(c) (1 point) Suppose you have a 3-dimensional input $x = (x_1, x_2, x_3) = (2, 2, 1)$ fully connected to 1 neuron with activation function g_i. The forward propagation can be written:

$$z = (\sum_{k=1}^{3} w_k x_k) + b$$
$$a_i = g_i(z)$$

After training this network, the values of the weights and bias are: $w = (w_1, w_2, w_3) = (0.5, -0.2, 0)$ and $b = 0.1$. You try 4 different activation functions (g_1, g_2, g_3, g_4) which respectively output the values $(a_1, a_2, a_3, a_4) = (0.67, 0.70, 1.0, 0.70)$. What is a valid guess for the activation functions g_1, g_2, g_3, g_4?

(i) sigmoid, tanh, indicator function, linear

(ii) linear, indicator function, sigmoid, ReLU

(iii) sigmoid, linear, indicator function, leaky ReLU

(iv) ReLU, linear, indicator function, sigmoid

(v) sigmoid, tanh, linear, ReLU

Recall that the indicator function is:

$$I(x)_{x \geq 0} = \begin{cases} 0 & if \quad x < 0 \\ 1 & if \quad x \geq 0 \end{cases}$$

(d) (2 points) A common method to accelerate the training of Generative Adversarial Networks (GANs) is to update the Generator k (≥ 1) times for every 1 time you update the Discriminator.

(i) True

(ii) False

(iii) It depends on the architecture of the GAN.

(e) (2 points) BatchNorm layers are really important when it comes to training GANs. However, the internal parameters of the BatchNorm (γ and β) are highly correlated to the input mini-batch of examples, which leads to the generated images being very similar to each other in a mini-batch.

(i) True

(ii) False

Question 2 (Short Answers, 22 points)

Please write concise answers.

(a) (2 points) What's the risk with tuning hyperparameters using a test dataset?

(b) (2 points) Explain why dropout in a neural network acts as a regularizer.

(c) (3 points) Why do we often refer to L2-regularization as "weight decay"? Derive a mathematical expression to explain your point.

(d) (2 points) Explain what effect will the following operations generally have on the bias and variance of your model. Fill in one of 'increases', 'decreases' or 'no change' in each of the cells:

	Bias	Variance
Regularizing the weights		
Increasing the size of the layers (more hidden units per layer)		
Using dropout to train a deep neural network		
Getting more training data (from the same distribution as before)		

(e) (3 points) Suppose you are initializing the weights $W^{[l]}$ of a layer with **uniform** random distribution $U(-\alpha, \alpha)$. The number of input and output neurons of the layer l are $n^{[l-1]}$ and $n^{[l]}$ respectively.

Assume the input activation and weights are independent and identically distributed, and have mean zero. You would like to satisfy the following equations:

$$E[z^{[l]}] = 0$$

$$Var[z^{[l]}] = Var[a^{[l-1]}]$$

What should be the value of α?
Hints: If X is a random variable distributed uniformly $U(-\alpha, \alpha)$, then $E(X) = 0$ and $Var(X) = \frac{\alpha^2}{3}$. Use the following relation seen in-class:

$$Var(z^{[l]}) = n^{[l-1]} Var(W^{[l]}) Var(a^{[l-1]})$$

(f) Consider the graph in Figure 1 representing the training procedure of a GAN:

(i) (2 points) Early in the training, is the value of $D(G(z))$ closer to 0 or closer to 1? Explain.

(ii) (2 points) Two cost functions are presented in figure (1), which one would you choose to train your GAN? Justify your answer.

(iii) (2 points) You know that your GAN is trained when D(G(z)) is close to 1. True / False ? Explain.

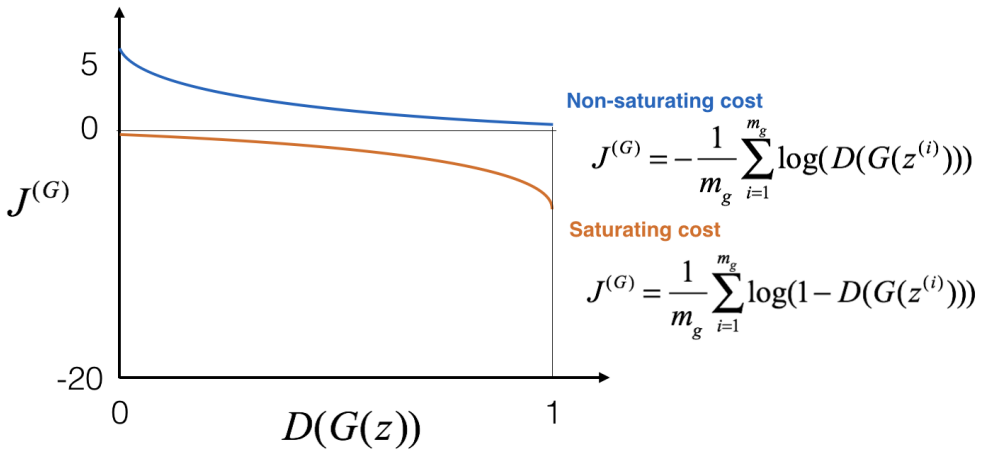


Figure 1: Cost function of the generator plotted against the output of the discriminator when given a generated image $G(z)$. Concerning the discriminator's output, we consider that 0 (resp. 1) means that the discriminator thinks the input "has been generated by G" (resp. "comes from the real data").

(g) (2 points) A neural network has been encrypted on a device, you can access neither its architecture, nor the values of its parameters. Is it possible to create an adversarial example to attack this network? Explain why.

(h) In a neural network, consider a layer that has $n^{[l-1]}$ inputs, $n^{[l]}$ outputs and uses a linear activation function. The input $a^{[l-1]}$ is independently and identically distributed, with zero mean and unit variance.

(i.) (1 point) How can you initialize the weights of this layer to ensure the output $z^{[l]}$ has the same variance as $a^{[l-1]}$ during the forward propagation?

(ii.) (1 point) How can you initialize the weights of this layer to ensure the gradient of input have unit variance as the gradient of the output during back-propagation? Explain your answer.

Question 3 (Coding, 7 points)

In this question you are asked to implement a training loop for a classifier. The input data is X, of shape (n_x, m) where m is the number of training examples. You are using a 2-layer neural network with:

- one hidden layer with n_h neurons
- an output layer with n_y neurons.

The code below is meant to implement the training loop, but some parts are missing. Between the tags (START CODE HERE) and (END CODE HERE), implement:

(i) the parameters initialization: initialize all your parameters, the weights should be initialized with Xavier initialization and the biases with zeros.

(ii) the parameters update: update your parameters with Batch Gradient Descent with momentum.

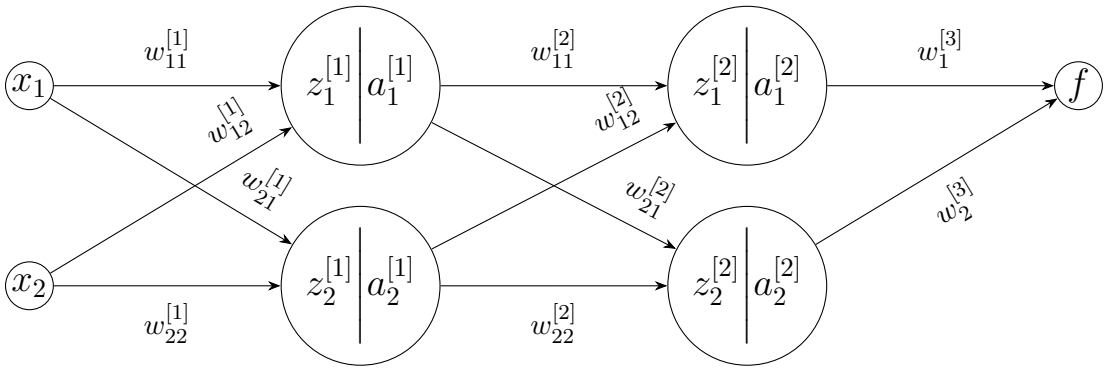
We won't penalize syntax errors.

```
import numpy as np

def train(X_train, Y_train, n_h, n_y, num_iterations, learning_rate, beta):
    """
    Implement train loop of a two layer classifier
    Arguments:
    X_train -- training data
    Y_train -- labels
    n_h -- size of hidden layer
    n_y -- size of output layer
    num_iteration -- number of iterations
    learning_rate -- learning rate, scalar
    beta -- the momentum hyperparameter, scalar

    Returns:
    W1, W2, b1, b2 -- trained weights and biases
    """

    m = X_train.shape[1] # number of training examples
    n_x = X_train.shape[0] # size of each training example
```

```
    # initialize parameters
    ### START CODE HERE ###









    ### END CODE HERE ###

    # training loop
    for i in range(num_iterations):

        # Forward propagation
        a1, cache1 = activation_forward(X, W1, b1, activation = "relu")
        a2, cache2 = activation_forward(a1, W2, b2, activation = "sigmoid")

        # Compute cost
        cost = compute_cost(a2, Y_train)

        # Backward propagation
        da2 = - 1./m * (np.divide(Y_train, a2) - np.divide(1 - Y_train, 1 - a2))
        da1, dW2, db2 = activation_backward(da2, cache2, activation = "sigmoid")
        dX, dW1, db1 = activation_backward(da1, cache1, activation = "relu")

        # Update parameters with momentum
        ### START CODE HERE ###









        ### END CODE HERE ###
    return W1, W2, b1, b2
```

Question 4 (Backpropagation, 12 points)

Consider this three layer network:



$$Z^{[1]} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \end{bmatrix} = \begin{bmatrix} w_{11}^{[1]} & w_{12}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad , \quad A^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \end{bmatrix} = \begin{bmatrix} \sigma(z_1^{[1]}) \\ \sigma(z_2^{[1]}) \end{bmatrix}$$

$$Z^{[2]} = \begin{bmatrix} z_1^{[2]} \\ z_2^{[2]} \end{bmatrix} = \begin{bmatrix} w_{11}^{[2]} & w_{12}^{[2]} \\ w_{21}^{[2]} & w_{22}^{[2]} \end{bmatrix} \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \end{bmatrix} \quad , \quad A^{[2]} = \begin{bmatrix} a_1^{[2]} \\ a_2^{[2]} \end{bmatrix} = \begin{bmatrix} \sigma(z_1^{[2]}) \\ \sigma(z_2^{[2]}) \end{bmatrix}$$

Given that $f = w_1^{[3]} a_1^{[2]} + w_2^{[3]} a_2^{[2]}$, compute :

(i) (3 points) $\delta_1 = \frac{\partial f(x)}{\partial z_1^{[2]}}$

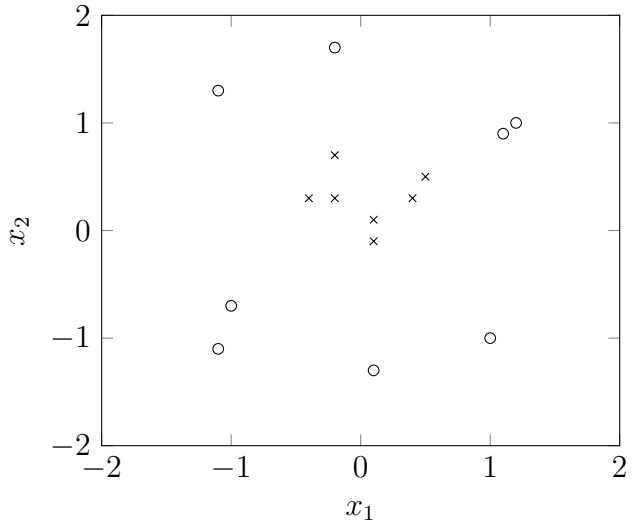
(ii) (3 points) $\delta_2 = \frac{\partial f(x)}{\partial Z^{[2]}}$

(iii) (3 points) $\delta_3 = \frac{\partial f(x)}{\partial Z^{[1]}}$

(iv) (3 points) $\delta_4 = \frac{\partial f(x)}{\partial w_{11}^{[1]}}$

Question 5 (Universal Approximation, (19 points))

Consider the following binary classification task:



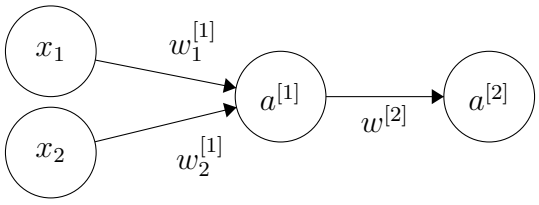
(a.) Let's begin by modeling this problem with a simple 2 layer network: with an activation function $g^{[1]}$ and a sigmoid output unit ($\sigma(z) = \frac{1}{1+e^{-z}}$):

$$z^{[1]} = w_1^{[1]} x_1 + w_2^{[1]} x_2 + b^{[1]}$$

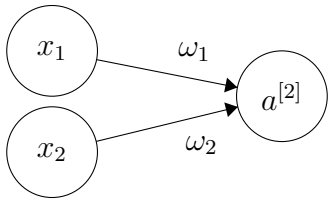
$$a^{[1]} = g^{[1]}(z^{[1]})$$

$$z^{[2]} = w^{[2]} a^{[1]} + b^{[2]}$$

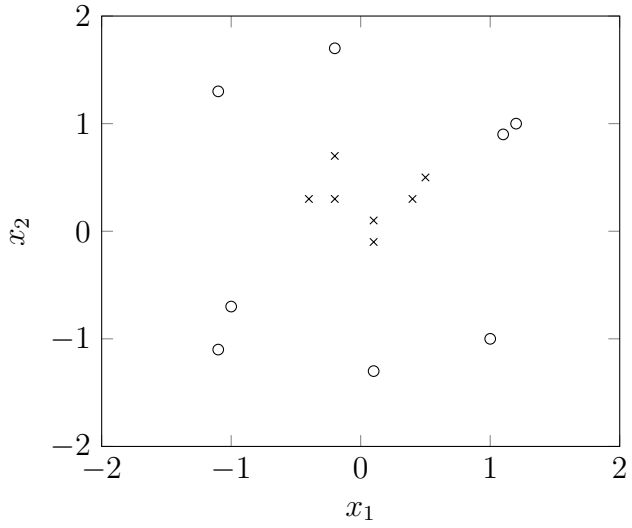
$$a^{[2]} = \sigma(z^{[2]})$$



(i.) (2 points) Show that if $g^{[1]}$ is a linear activation function, $g^{[1]}(z) = \alpha z$, then the above network can be reduced to the single layer network shown below. Give the new weights and bias for this network $\omega_1, \omega_2, \beta_1$.

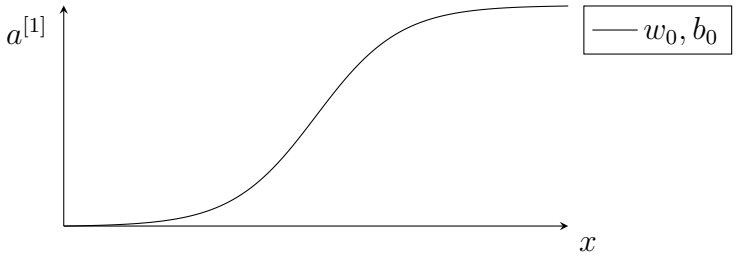


(ii.) (2 points) If we use a threshold of $a^{[2]} > 0.5$, what is the form of decision rules that can be learned using a $g(z) = \alpha z$. Draw an example of a possible decision rule on the plot below, and write down the equation of the decision rule.

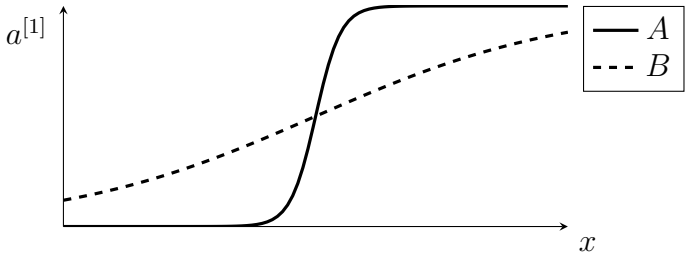


(b.) Instead of a linear activation function, let's examine the effects of defining $g^{[1]}$ as a sigmoid activation: $g^{[1]}(z) = \sigma(z) = \frac{1}{1+e^{-z}}$. In order to simplify the graphical representation, assume that you have a single neuron activation $a^{[1]} = g^{[1]}(z^{[1]}) = \sigma(wx + b)$ where w, x and b are scalars.

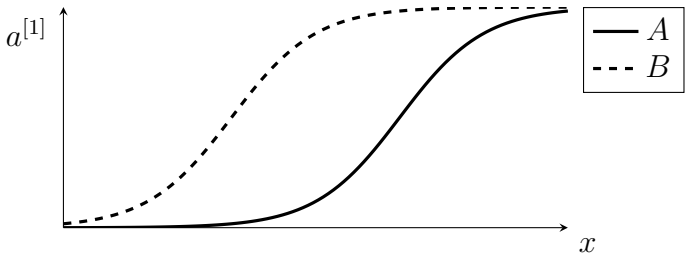
Let's plot $a^{[1]}$ against the input x for parameters $(w, b) = (w_0, b_0)$:



(i.) (1 point) For a $\Delta > 0$, which line corresponds to the change w to $w_0 + \Delta$? Circle **A** or **B**, given that we hold b at b_0.



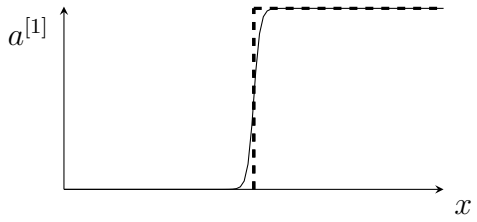
(ii.) (1 point) For a $\Delta > 0$, which line corresponds to the change b to $b_0 + \Delta$? Circle **A** or **B**, given that we hold w at w_0.



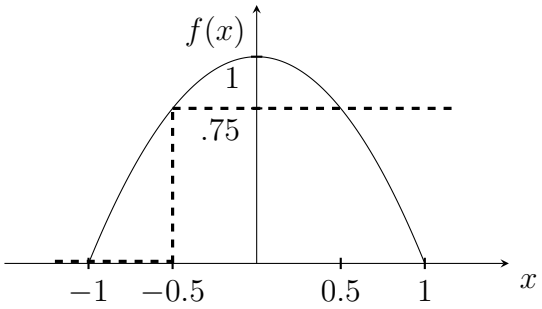
(c.) Given the responses above, it can be shown that for certain choices of w and b, the sigmoid response can closely approximate a step function:

$$a^{[1]} = \sigma(wx + b) \approx step(x) = 1\{x \geq s\}$$

with $s = -b/w$.

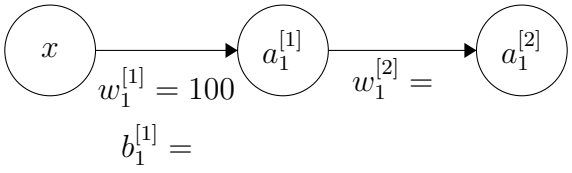


(i.) (2 points) For a scalar input x, you wish to approximate $f(x) = 1 - x^2$ with the dotted line below (a step function),



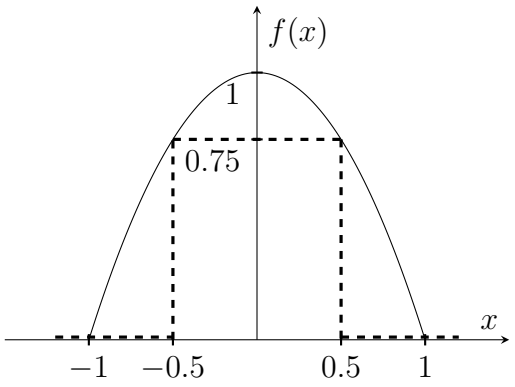
using the network,

$$z_1^{[1]} = w_1^{[1]}x + b_1^{[1]}$$
$$a_1^{[1]} = \sigma(z_1^{[1]})$$
$$z_1^{[2]} = w_1^{[2]}a_1^{[2]}$$
$$a_1^{[2]} = z_1^{[2]}$$



fill in the above values of $b_1^{[1]}$ and $w_1^{[2]}$ that result in the dotted approximation.

(ii.) (4 points) You now want to go a step further and approximate using a pulse as shown below:



If you use the network below,

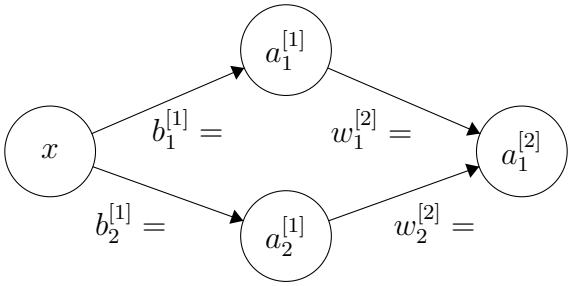
$$Z^{[1]} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \end{bmatrix} = \begin{bmatrix} w_1^{[1]} & 0 \\ 0 & w_2^{[1]} \end{bmatrix} \begin{bmatrix} x \\ x \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \end{bmatrix}$$

$$A^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \end{bmatrix} = \begin{bmatrix} \sigma(z_1^{[1]}) \\ \sigma(z_2^{[1]}) \end{bmatrix}$$

$$Z^{[2]} = \begin{bmatrix} z_1^{[2]} \\ z_2^{[2]} \end{bmatrix} = \begin{bmatrix} w_1^{[2]} & 0 \\ 0 & w_2^{[2]} \end{bmatrix} \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \end{bmatrix}$$

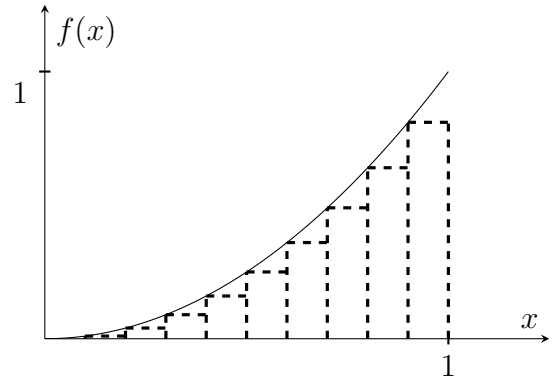
$$a_1^{[2]} = z_1^{[2]} + z_2^{[2]}$$

with $w_1^{[1]} = w_2^{[1]} = 100$,



Fill in the parameters $b_1^{[1]}$,$b_2^{[1]}$ and $w_1^{[2]}$,$w_2^{[2]}$ to approximate the dotted pulse function.

(d.) You can extend the above scheme to closely approximate almost any function. Here you will approximate $f(x) = x^2$ over [0,1) using step impulses plotted below,



with each step impulse as

$$\tilde{f}(x; w, s_1, s_2) = w * 1\{s_1 \leq x < s_2\}$$

and your approximation as a sum of impulses,

$$\hat{f}(x) = \sum_{\hat{x} \in S} \tilde{f}(x; \hat{x}^2, \hat{x}, \hat{x} + \epsilon)$$

We define $S = \{0, \epsilon, 2\epsilon, ..., 1 - \epsilon\}$ where ϵ is a small positive scalar such that all bins have equal width.

(i.) (5 points) What is the maximum error $|f(x) - \hat{f}(x)|$ of your approximation over [0,1)? Give your answer as a function of ϵ, the impulse width.
Hint: for any $x \in [0, 1)$ consider $\hat{x}_n$ to be the smallest element of S such that $\hat{x}_n + \epsilon > x$, and find an upper bound to $|f(x) - \hat{f}(x)|$ that depends only on ϵ.

(ii.) (2 points) As you've just seen, you can approximate functions arbitrarily well with 1 hidden layer. State a reason and explain why, in practice, you would use deeper networks.

Question 6 (Optimization, 9 points)

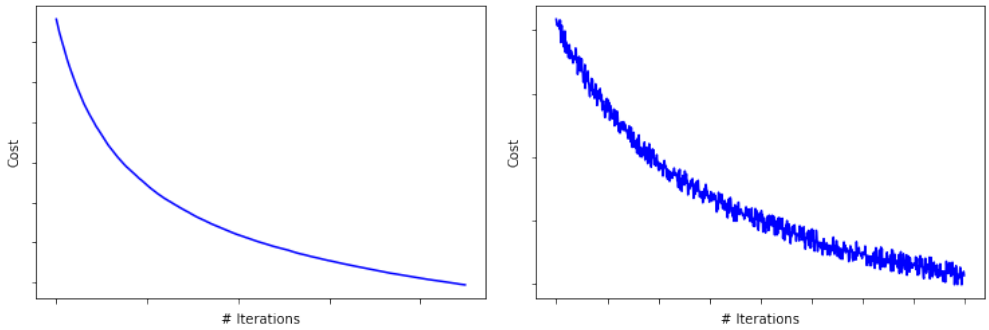
For these questions, we expect you to be concise and precise in your answers.

(a) (2 points) What problem(s) will result from using a learning rate that's too high? How would you detect these problems?

(b) (2 points) What problem(s) will result from using a learning rate that's too low? How would you detect these problems?

(c) (2 points) What is a saddle point? What is the advantage/disadvantage of Stochastic Gradient Descent in dealing with saddle points?

(d) (1 point) Figure 2 below shows how the cost decreases (as the number of iterations increases) when two different optimization algorithms are used for training. Which of the graphs corresponds to using batch gradient descent as the optimization algorithm and which one corresponds to using mini-batch gradient descent? Explain.



(a) Graph A (b) Graph B

Figure 2

(e) (2 points) Figure 3 below shows how the cost decreases (as the number of iterations increases) during training. What could have caused the sudden drop in the cost? Explain one reason.

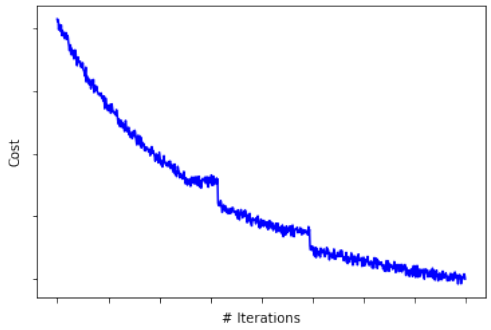


Figure 3

Question 7 (Case Study: semantic segmentation on microscopic images, 25 points)

You have been hired by a group of health-care researchers to solve one of their major challenges dealing with cell images: determining which parts of a microscope image corresponds to which individual cells.
In deep learning, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics. In your case, you want to locate the cells and their boundaries in microscopic images.
Here are three examples of input images and the corresponding target images:

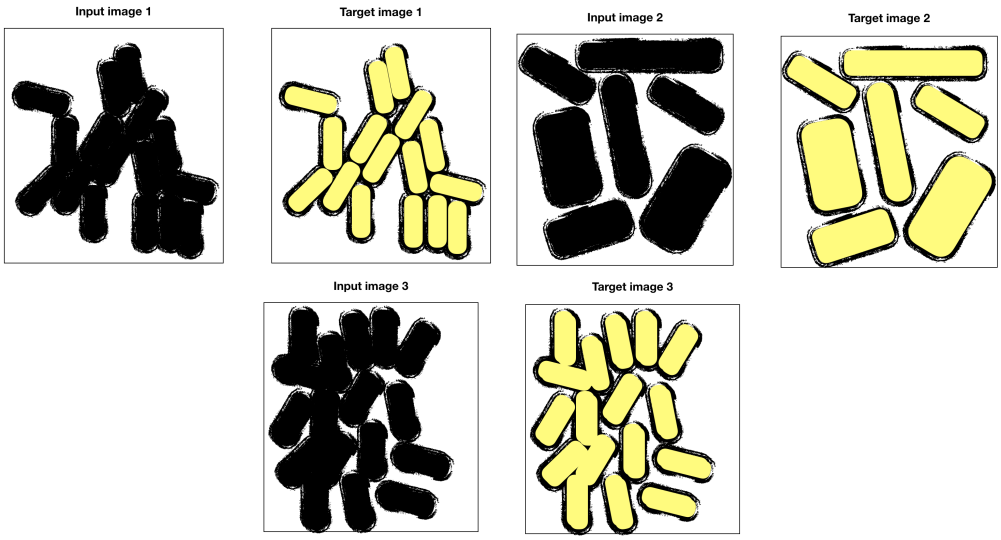


Figure 4: The input images are taken from a microscope. The target images have been created by the doctors, they labeled the pixels of the input image such that 1 represents the presence of a cell while 0 represents the absence of a cell. The target image is the superposition of the labels and the input image (light grey pixels you can see inside the cells correspond to label 1, indicating the pixel belongs to a cell). A good algorithm will segment the data the same way the doctors have labeled it.

In other words, this is a classification task where each pixel of the target image is labeled as 0 (this pixel is not part of a cell) or 1 (this pixel is part of a cell).

Dataset: Doctors have collected 100,000 images from microscopes and gave them to you. Images have been taken from three types of microscopes: A (50,000 images), B (25,000 images) and C (25,000 images). The doctors who hired you would like to use your algorithm on images from microscope C.

(a) (3 points) Explain how you would split this dataset into train, dev and test sets. Give the exact percentage split, and give reasons to your choices.

(b) (2 points) Can you augment this dataset? If yes, give only 3 distinct methods you would use. If no, explain why (give only 2 reasons).

You have finished the data processing, and are wondering you could solve the problem using a neural network. Given a training examples $x^{(i)}$ (flattened version of an RGB input image, of shape $(n_x, 1)$ and its corresponding label $y^{(i)}$ (flattened version of labels of shape $(n_y, 1)$) answer the following questions:

(c) (2 points) What is the mathematical relation between n_x and n_y?

(d) (2 points) Write down the cross-entropy loss function $\mathcal{L}^{(i)}$ for one training example.

(e) (1 point) Write down the cost function $\mathcal{J}$, i.e. generalize the loss function to a batch of m training examples.

You have coded your neural network (model M1) and have trained it for 1000 epochs.

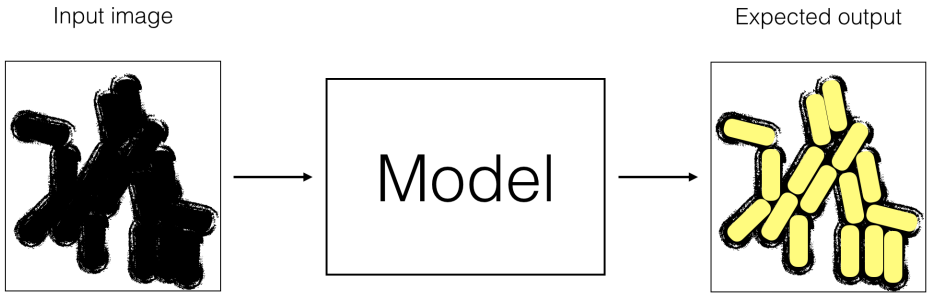


Figure 5: Your model (M1) takes as input a cell image taken from a microscope and outputs a matrix of 0s and 1s indicating for each pixel if it is part of a cell or not. You then superpose this matrix on the input image to see the results. For the given input image, M1 should ideally output the target image above.

Your model is not performing well. One of your friends suggested to use transfer learning using another labeled dataset made of 1,000,000 microscope images for skin disease classification. A model (M2) has been trained on this dataset on a 10-class classification. The images are the same size as those of the dataset the doctors gave you. Here is an example of input/output of the model M2.

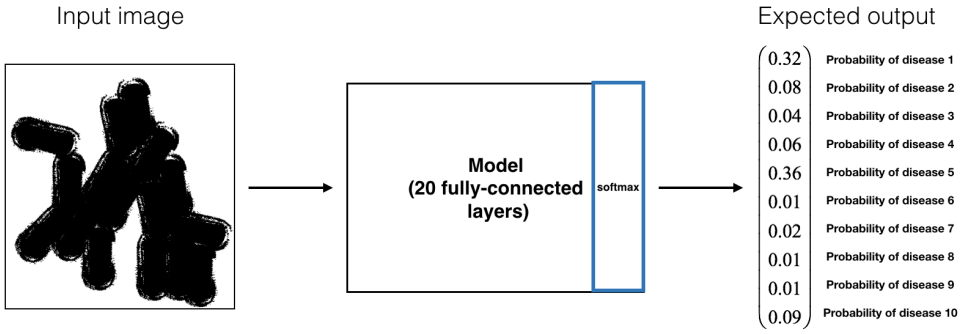


Figure 6: The model (M2) takes as input a cell image taken from a microscope and outputs a 10-dimensional vector of values indicating the probabilities of presence of each of the 10 considered deceases in the image.

(f) (3 points) Explain what transfer learning means in this case. If there are hyperparameters specific to the transfer learning process, explain how you would choose them.

You now have a trained model, you would like to define a metric computing the per-pixel accuracy. An accuracy of 78% means that 78% of the pixels have been classified correctly by the model, while 22% of the pixels have been classified incorrectly.

(g) (2 points) Write down a formula defining your accuracy metric.

Your model's accuracy on the test set is 96%. You thus decide to present the model to the doctors. They are very unhappy, and argue that they would like to visually distinguish cells. They show you the following prediction output, which does not clearly separate distinct cells.

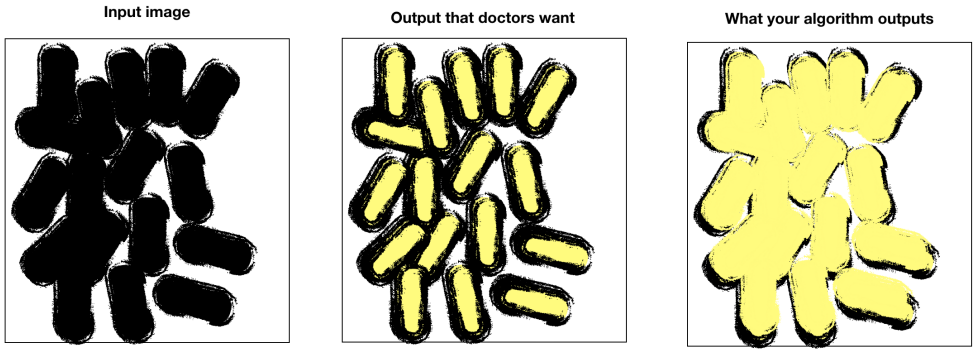


Figure 7: On the left, the input image. In the middle, the type of segmentation that would have satisfied the doctors. On the right, the output of your algorithm which struggles to separate distinct cells.

(h) (4 points) How can you correct your model and/or dataset to satisfy the doctors' request? Explain in details.

You have solved the problem, the doctors are really satisfied. They have a new task for you: a binary classification. They give you a dataset containing images similar to the previous ones. The difference is that each image is labeled as 0 (there are no cancer cells on the image) or 1 (there are cancer cells on the image).
Strong from your previous experience with neural networks, you easily build a state-of-the-art model to classify these images with 99% accuracy. The doctors are astonished and surprised, they ask you to explain your network's predictions. More specifically,

(i) (3 points) Given an image classified as 1 (cancer present), can you figure out based on which cell(s) the model predicted 1? Explain.

Your model detects cancer on cells (test set) images with 99% accuracy, while a doctor would on average perform 97% accuracy on the same task.

(j) (3 points) Is this possible? Explain.

Question 8 (AlphaTicTacToe Zero)

DeepMind recently invented AlphaGo Zero – a Go player that learns purely through self-play and without any human expert knowledge. AlphaGo Zero was able to impressively defeat their previous player AlphaGo, which was trained on massive amounts of human expert games. AlphaGo in its turn had beat the human world Go champion- a task conceived nearly impossible 2 years ago! Unsurprisingly, at its core, AlphaGo Zero uses a neural network.

Your task is to build a neural network that we can use as a part of AlphaTicTacToe Zero. The board game of TicTacToe uses a grid of size 3×3, and players take turns to mark an $\times$ (or $\bigcirc$) at any unoccupied square in the grid until either player has 3 in a row or all nine squares are filled.

(a) (2 points) The neural network we need takes the grid at any point in the game as the input. Describe how you can convert the TicTacToe grid below into an input for a neural network.

$\times$	$\bigcirc$	
	$\times$	$\times$
	$\bigcirc$	

(b) (3 points) The neural network we require has 2 outputs. The first is a vector $\vec{\boldsymbol{a}}$ of 9 elements, where each element corresponds to one of the nine squares on the grid. The element with the highest value corresponds to the square which the current player should play next. The second output is a single scalar value $\boldsymbol{v}$, which is a continuous value in [-1,1]. A value closer to 1 indicates that the current state is favorable for the current player, and -1 indicates otherwise.

Roughly sketch a fully-connected single hidden layer neural network (hidden layer of size 3) that takes the grid as input (in its converted form using the scheme described in part (a)) and outputs $\vec{\boldsymbol{a}}$ and $\boldsymbol{v}$. In your sketch, clearly mark the input layer, hidden layer and the outputs. You need not draw all the edges between two layers, but make sure to draw all the nodes. Remember, the same neural network must output both $\vec{\boldsymbol{a}}$ and $\boldsymbol{v}$.

(c) (i) (1 point) As described above, each element in the output $\vec{\boldsymbol{a}}$ corresponds to a square on the grid. More formally, $\vec{\boldsymbol{a}}$ defines a probability distribution over the 9 possible moves, with higher probability assigned to the better move. What activation function should be used to obtain $\vec{\boldsymbol{a}}$?

(ii) (1 point) The output $\boldsymbol{v}$ is a single scalar with value in [-1,1]. What activation function should be used to obtain $\boldsymbol{v}$?

(d) (4 points) During the training, given a state t of the game (grid), the model predicts $\vec{\boldsymbol{a}}^{<t>}$ (vector of probabilities) and $\boldsymbol{v}^{<t>}$. Assume that for every state t of the game, someone has given you the best move $\vec{\boldsymbol{y_a}}^{<t>}$ to do (one-hot vector) and the corresponding value $\boldsymbol{y_v}^{<t>}$ for $\boldsymbol{v}^{<t>}$

In terms of $\vec{\boldsymbol{a}}^{<t>}$, $\boldsymbol{v}^{<t>}$, $\vec{\boldsymbol{y_a}}^{<t>}$ and $\boldsymbol{y_v}^{<t>}$, propose a valid loss function for training on a single game with T steps. Explain your choice.

Question 9 (Practical industry-level questions)

You want to solve the following problem: *build a trigger word detection algorithm to spot the word* ***cardinal*** *in a 10 second long audio clip.*

(a.) (4 points) Explain in a short paragraph what is the best practice for building the dataset.

(b.) (2 points) Give 2 pros and 2 cons of embedding your model on a smartphone device instead of using it on a server.

(c.) (2 points) You have coded a model and trained on the audio dataset you have built, the training accuracy indicates that there is a problem. Other than spending time checking your code, what is a good strategy to quickly know if the problem is due to an error in your code, or from the fact that your model is not complex/deep enough to understand the dataset?

Extra Page 1/3

Extra Page 2/3

Extra Page 3/3

END OF PAPER