HW1

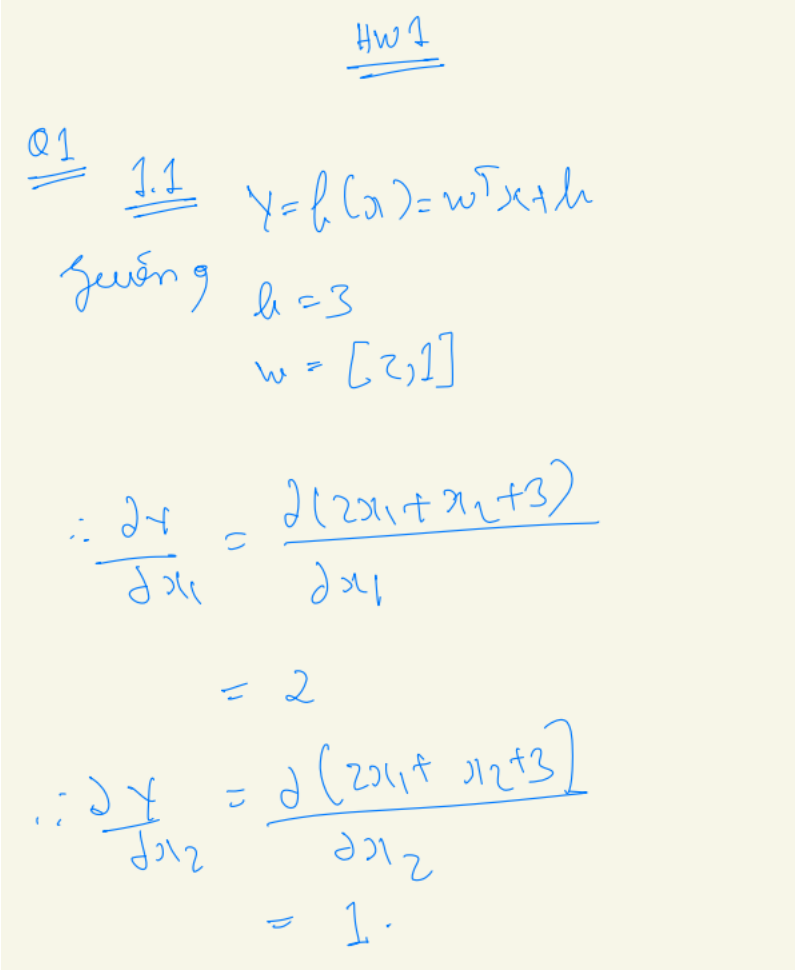
Submitted By: Parthiv Borgohain (pb25347)

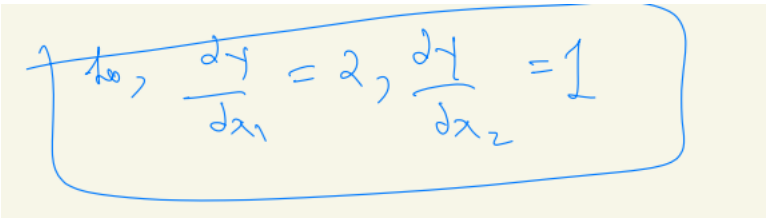
Note: Some of the questions have been answered in the attached ipynb file. Please refer to that file as well.

Question 1. In this question, we will define tensors in PyTorch and use them to find the derivative of a linear function w.r.t its variables. We then find the mean squared error and conclude with finding the optimum parameters of a linear model.

1.1 For the function y = f(x) = w.x + b, where w=[2,1] and b=3, find the partial derivates of y w.r.t the components of x (i.e.: dy/dx1 and dy/dx2) at x = [4,2] both on paper and using PyTorch.

Answer:





So, the gradients w.r.t x1 and x2 are 2 and 1 respectively.

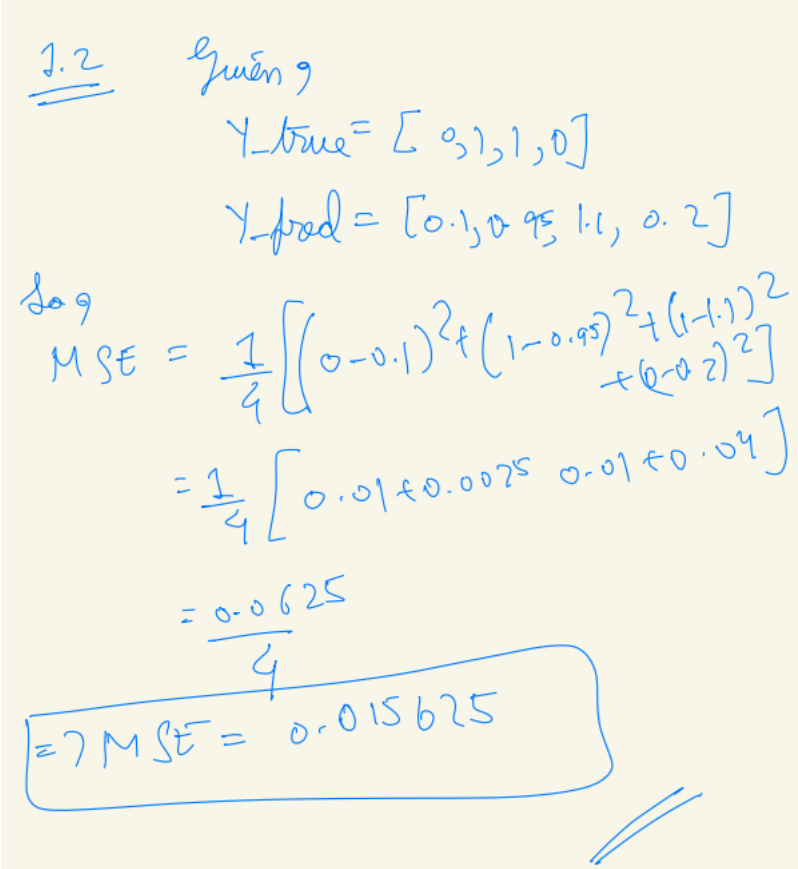
1.2 For a model y = f(x), the predicted and true values are as follows:

y_true = [0,1,1,0]

y_pred = [0.1,0.95,1.10,0.2]

Find the mean squared error both on paper and using PyTorch. In PyTorch, solve it using an inbuilt function and also by defining your own squared error function.

Answer:



So, the MSE is 0.015625

1.3 You are given the following dataset (note that it describes an XOR function):

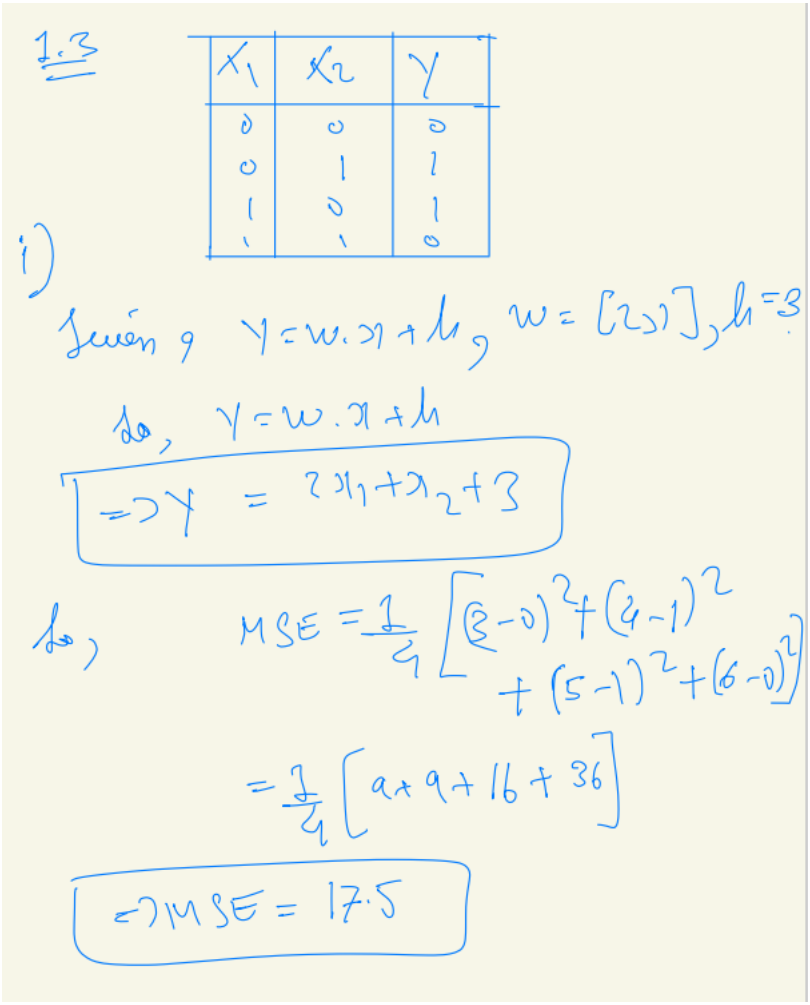
x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

Assume that we fit a linear function, y = w.x + b to this dataset with w = [2,1] and b=3.

i) Find the mean squared error (loss) over this dataset for the above weights and bias.
ii) Which direction should the weights move in to decrease the loss by maximum amount? Find that direction both on paper and using PyTorch.
iii) For what values of w and b is the loss minimum? Solve on paper only.

Answer:

i)


1.3
x_1 x_2 y
0 0 0
0 1 1
1 0 1
1 1 0
i)
Given, $y = w.x + b$, $w = [2,1]$, $b = 3$
So, $y = w.x + b$
$\Rightarrow y = 2x_1 + x_2 + 3$
So, $MSE = \frac{1}{4}[(3-0)^2 + (4-1)^2 + (5-1)^2 + (6-0)^2]$
$= \frac{1}{4}[9 + 9 + 16 + 36]$
$\Rightarrow MSE = 17.5$

ii)

ii)

$$MSE = \frac{1}{4}\left[(0-0-0-b)^2 + (1-0-w_2-b)^2 + (1-w_1-0-b)^2 + (0-w_1-w_2-b)^2\right]$$

$$\Rightarrow MSE = \frac{b^2 + (1-w_2-b)^2 + (1-w_2-b)^2 + (w_1+w_2+b)^2}{4}$$

$$\Rightarrow \frac{\partial(MSE)}{\partial w_1} = \frac{1}{4}\left[-2(1-w_1-b) - 2(-w_1-w_2-b)\right]$$

$$\boxed{\Rightarrow \frac{\partial(MSE)}{\partial w_1} = \frac{4w_1 + 2w_2 + 4b - 2}{4}}$$

Similarly,

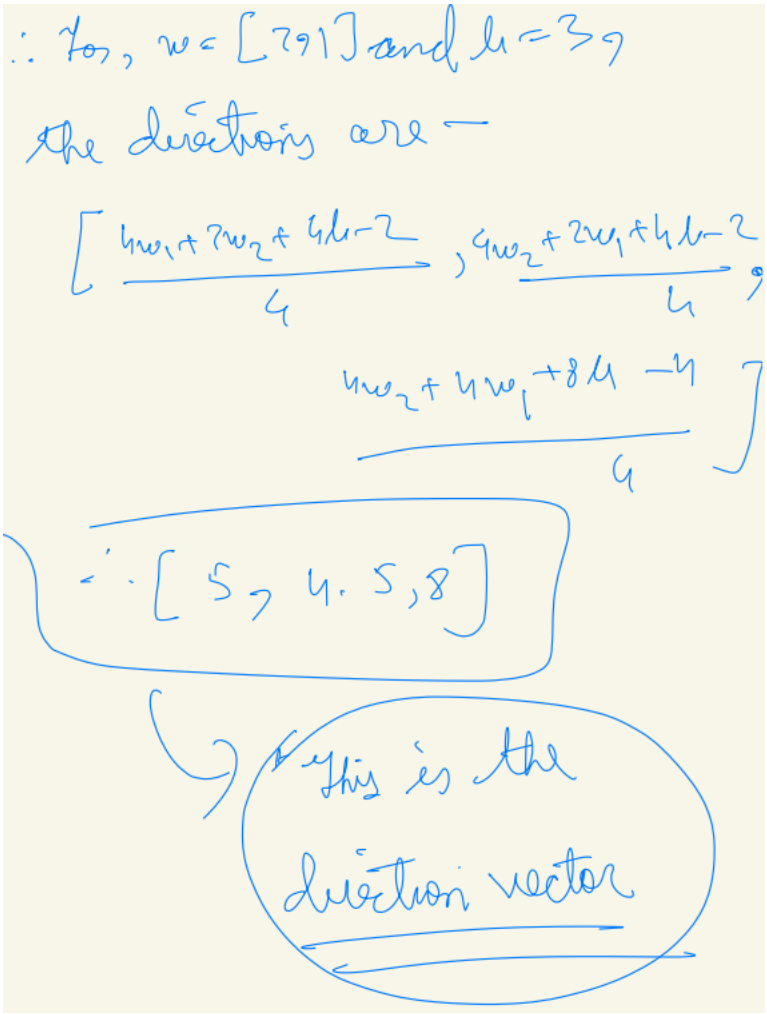
$$\frac{\partial(MSE)}{\partial w_2} = \frac{1}{4}\left[-2(1-w_2-b) - 2(w_1+w_2+b)\right]$$

$$\boxed{\Rightarrow \frac{\partial(MSE)}{\partial w_2} = \frac{4w_2 + 2w_1 + 4b - 2}{4}}$$

Similarly,

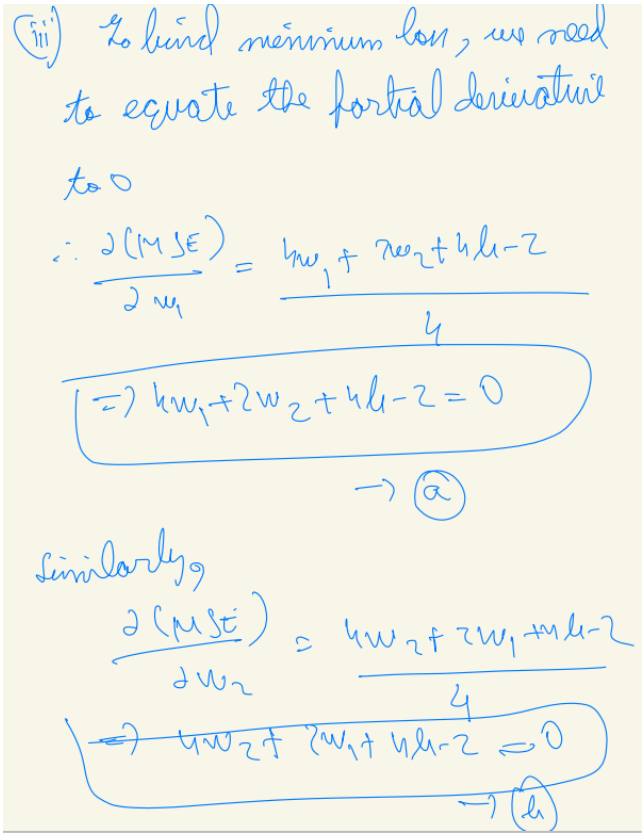
$$\frac{\partial(MSE)}{\partial b} = \frac{1}{4}\left[2b - 2(1-w_2-b) - 2(1-w_1-b) - 2(-w_1-w_2-b)\right]$$

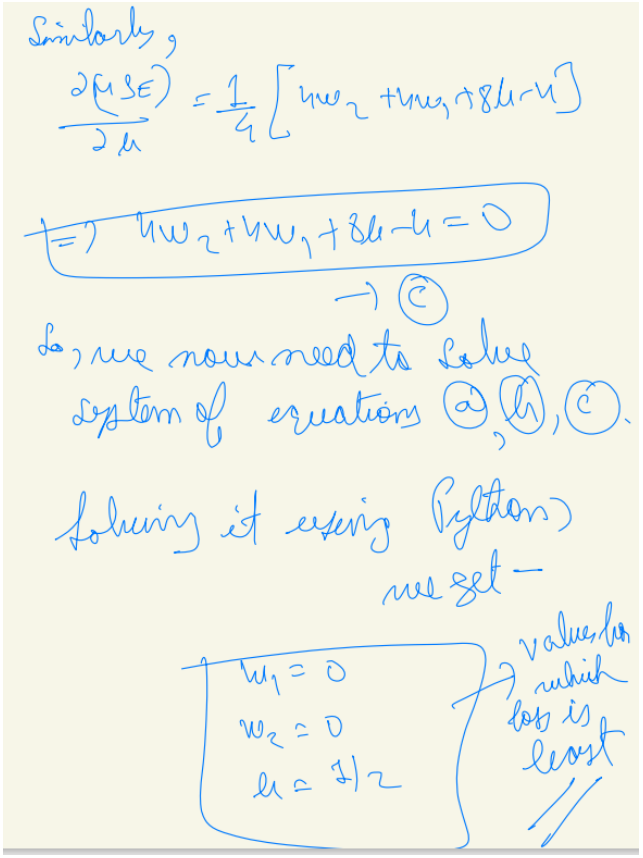
$$\boxed{\Rightarrow \frac{\partial(MSE)}{\partial b} = \frac{4w_2 + 4w_1 + 8b - 4}{4}}$$



The direction vector is [5,4.5,8]

iii)





So, the values for which the loss is minimum are:

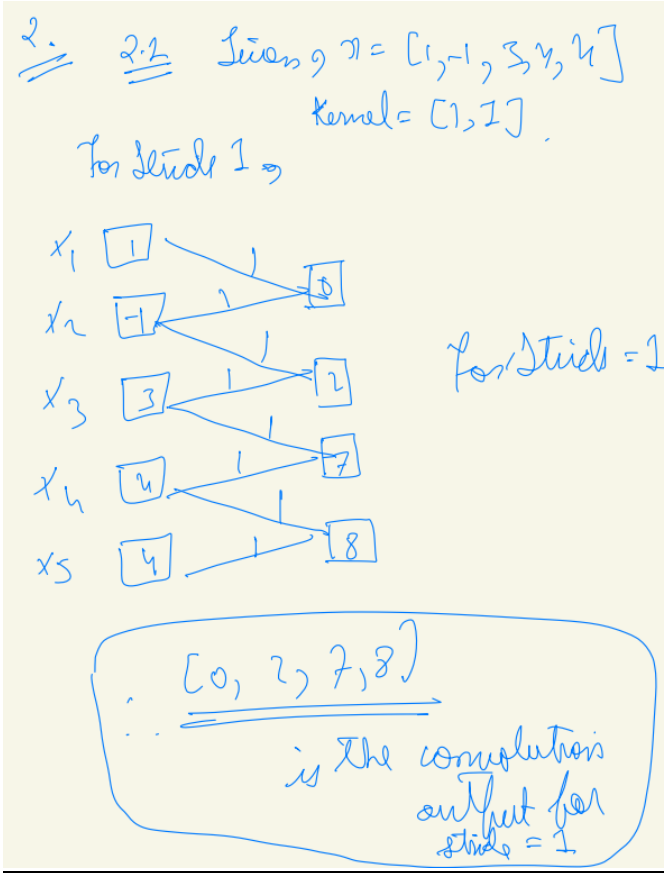
- The weights are w1=0,w2=0

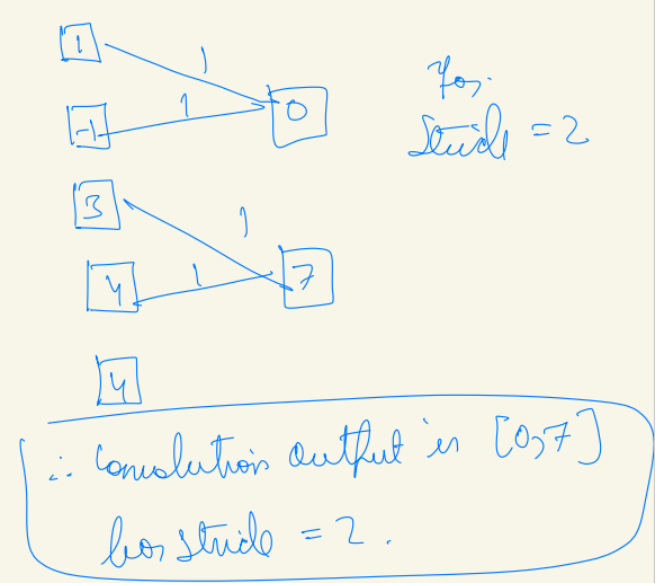
- The bias is ½ or 0.50

Q2. In this question, we will learn to perform 1-d and 2-d convolutions using different strides on a grayscale image.

2.1 Given a 1-d input x=[1,-1,3,4,4] and a kernel=[1,1], find the 1-d convolution for stride=1 and stride=2. Solve both on paper and using pyTorch.

Answer:





So, the convolution outputs for both the channels are [0,2,7,8] , [0,7]

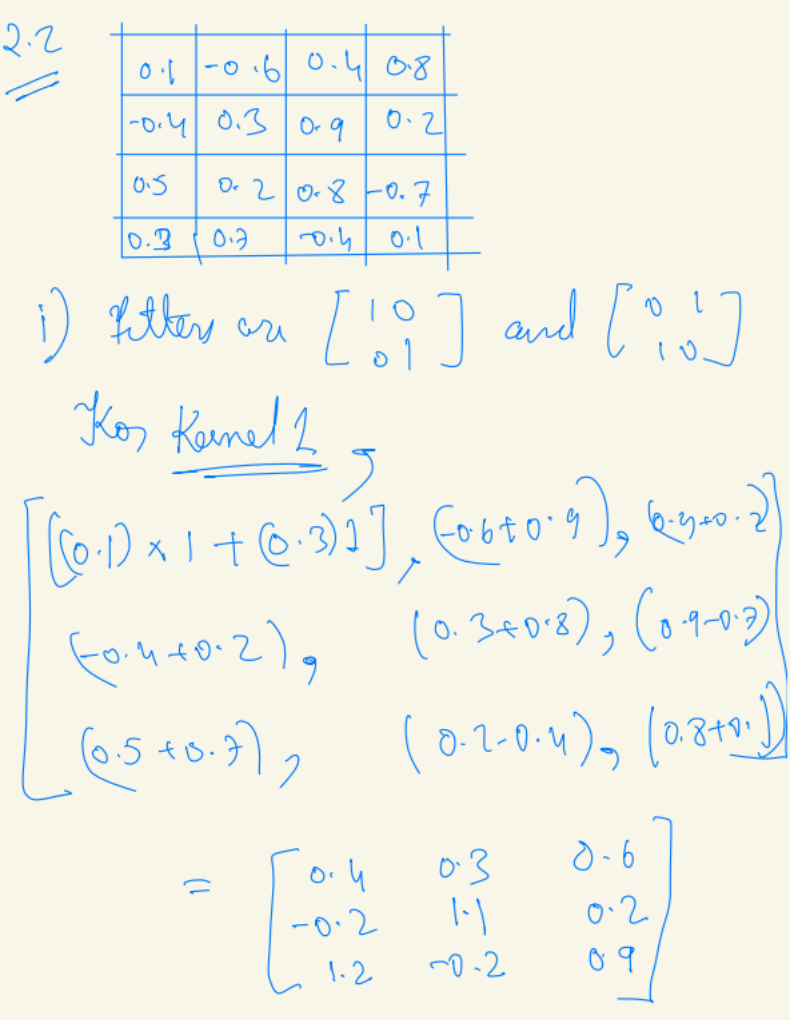
2.2 You're given the following grayscale image (any matrix in 2 dimensions is a grayscale image):

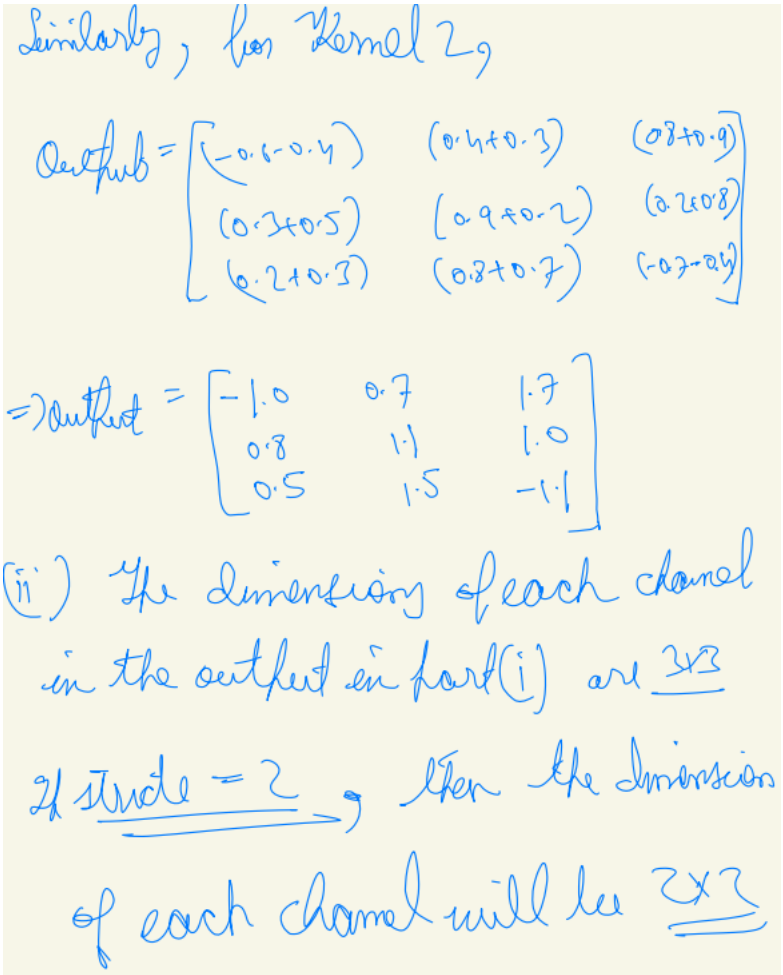
0.1	-0.6	0.4	0.8
-0.4	0.3	0.9	0.2
0.5	0.2	0.8	-0.7
0.3	0.7	-0.4	0.1

i) You're given 2 2*2 filters (kernels): [[1,0],[0,1]] and [[0,1],[1,0]]. Find the output when the image is convolved using each filter with stride=1. Solve both on paper and by using pyTorch. Note that you'll get 2 outputs - one for each filter - each output is known as a channel.

ii) What are the dimensions of each channel in (i)? What will be the dimensions of the output when stride=2?

Answer:

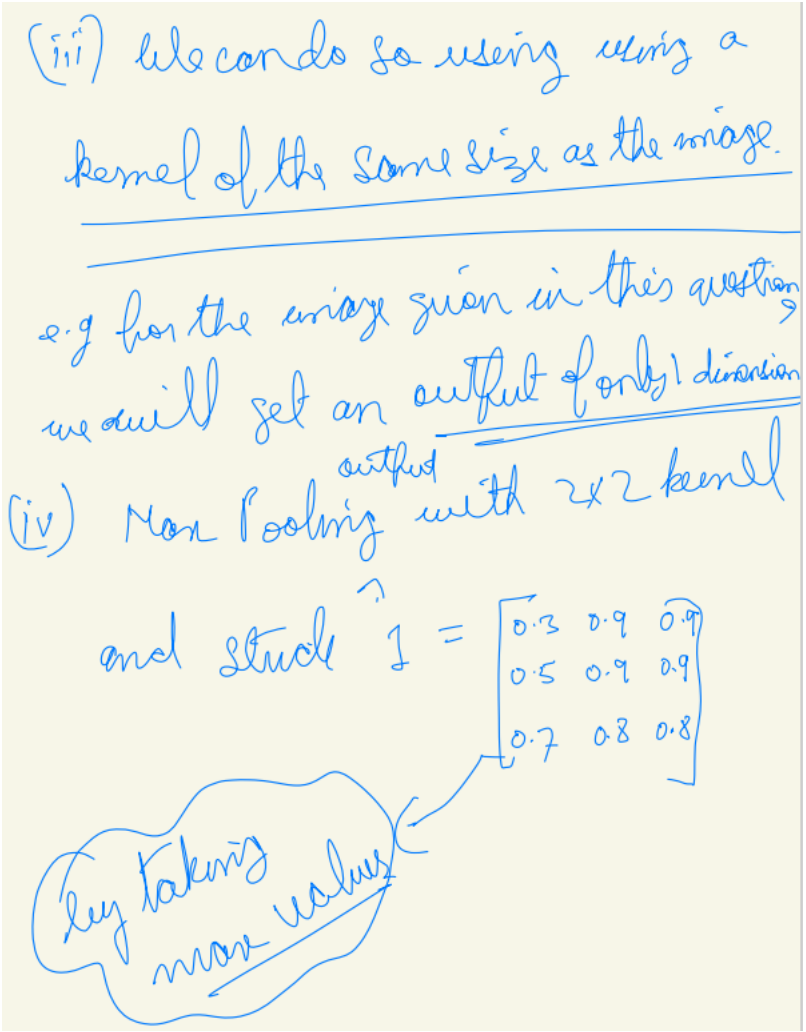




iii)The given image is a 2-d matrix. How can you convolve it so that the output channel has only one dimension?

iv) Perform a 2*2 max-pooling with stride=1 on the image both on paper and using PyTorch. What are the dimensions of the channel after max-pooling?

Answer:



Extra-credit: If the size of the output channel after performing a 2*2 convolution (stride=1) and a 2*2 max-pooling (stride=1) is the same, why do you think we needed max-pooling when we could perform convolution to decrease the size of the image. Think in terms of the advantages that max-pooling may offer over convolution. When do you think a max-pooling operation may not be advantageous (and in fact may hurt the network)?

Answer:

Max pooling is a widely used operation in convolutional neural networks (CNNs) that aids in the reduction of the input's spatial dimensions. This reduction can significantly reduce the model's computational cost and prevent overfitting. Max pooling does not introduce any trainable parameters during training, which means that it does not add any complexity to the model. Conversely, convolutional layers possess trainable parameters known as convolutional kernels, which are used to learn crucial features from the input data. During training, the kernel weights are modified using backpropagation to minimize the loss function. This process can add complexity to the model and increase the computational cost of training.

While both convolution and max pooling are effective dimensionality reduction techniques, they serve distinct purposes. Convolution captures all the information present in the input, whereas max pooling selects the most significant information. Max pooling has its advantages, such as extracting

the most vital features from the input and discarding less important ones. It also introduces some level of invariance to small translations or rotations in the input, which can be beneficial for tasks like object recognition.

However, the use of the max-pooling operation may not be beneficial in certain situations, such as when the input image is small, the task requires precise localization, or the features are already well-extracted. In such cases, using max pooling can result in the loss of crucial information and important details that could be vital to the task at hand. For instance, in image recognition tasks where the input image is small, max pooling may cause the image to become too small and lose vital features. Similarly, in tasks that require precise localization, max pooling may cause the loss of spatial information, making it difficult to pinpoint the exact location of the object of interest.

Question 3 This question deals with the addition of perturbations to an image x to create an adversarial example from that image.

3.1 Identify True/False i) In FGSM method, the parameters of the trained model change.

ii) In FGSM method, the pixel values of input image changes.

iii)FGSM can only be used for undirected adversarial attacks (An undirected adversarial attack is one in which the aim is to only perturb the original image in the direction of maximum loss. The attack does not care about which incorrect class the input example is classied into after perturbation).

iv) PGD always finds points inside the threat model.

Answer:

i)False. In FGSM, the parameters of the trained model are not directly modified. However, the input data is perturbed to generate adversarial examples that can fool the model's predictions. This means that the model's output can be different for these adversarial inputs, even though its parameters remain the same.

ii) True. The Fast Gradient Sign Method (FGSM) involves perturbing the pixel values of an input image by a small amount in the direction of the sign of the gradient of the loss function with respect to the input. This perturbation is designed to create an adversarial example that can mislead the model's predictions.

iii) False. The Fast Gradient Sign Method (FGSM) is a form of untargeted attack in which the perturbation is not specifically designed to target certain features of the input data that are most important for classification.

iv)True. The PGD algorithm ensures that the final point is within the allowable threat model and close to the original input.

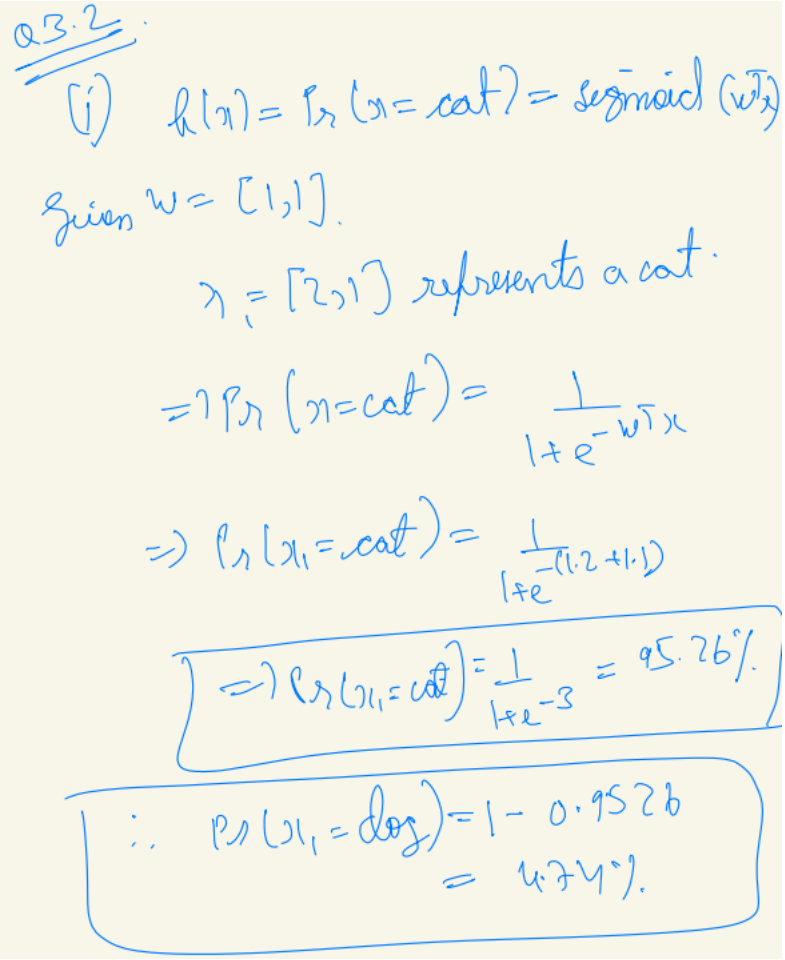
3.2 Solve this question assuming that you're the attacker. You're given a linear classifier which classifies each input into a dog or a cat:

$$f(x) = Pr(x = cat) = sigmoid(w'x)$$

Given w = [1,1].

i) It is given that x = [2, 1] represents a cat. What is $Pr(x = dog)$ and $Pr(x = cat)$?

Answer:

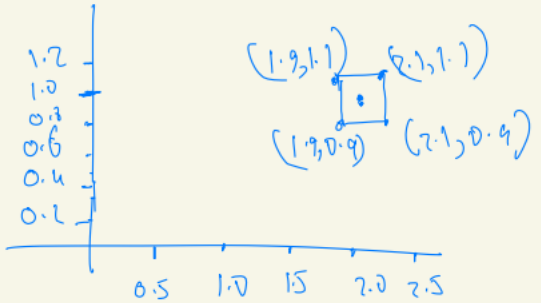


ii) Now you want to change the components of x (i.e, change from [2,1] to something else) so that the probability of dog increases. However, you can only change x acc. to the threat model $||x - x$. 1 $|\ inf$ <= 0. 1 The threat model specifies the region in which the input x is allowed to vary. 1 Plot the threat model on a 2-d graph and specify the co-ordinates of the corners of the quadrilateral thus formed.

Answer:

(ii) Given, $x_1 = [2, 1]$

$||x - x_1||_{inf} \leq 0.1$



∴ Centre of the square is (2, 1)

∴ Co-ordinates of the corners are

(1.9, 0.9), (2.1, 0.9), (2.1, 1.1), (1.9, 2.1)

iii) Write down the optimization equation and constraints for finding the optimum x (optimum x is that x which assigns maximum probability to x being a dog). Denote optimum x by $x\ optim$

Answer:

(iii) This the optimization problem –

Max $\sigma(w^T \cdot x)$

such that $||x - x_1||_{inf} \leq 0.1$

∴ we have –

Max. $\sigma(w^T \cdot x)$

s.t $\max(|x - x_1|, |y - y_1|) \leq 0.1$

As seen in the plot in part (ii),

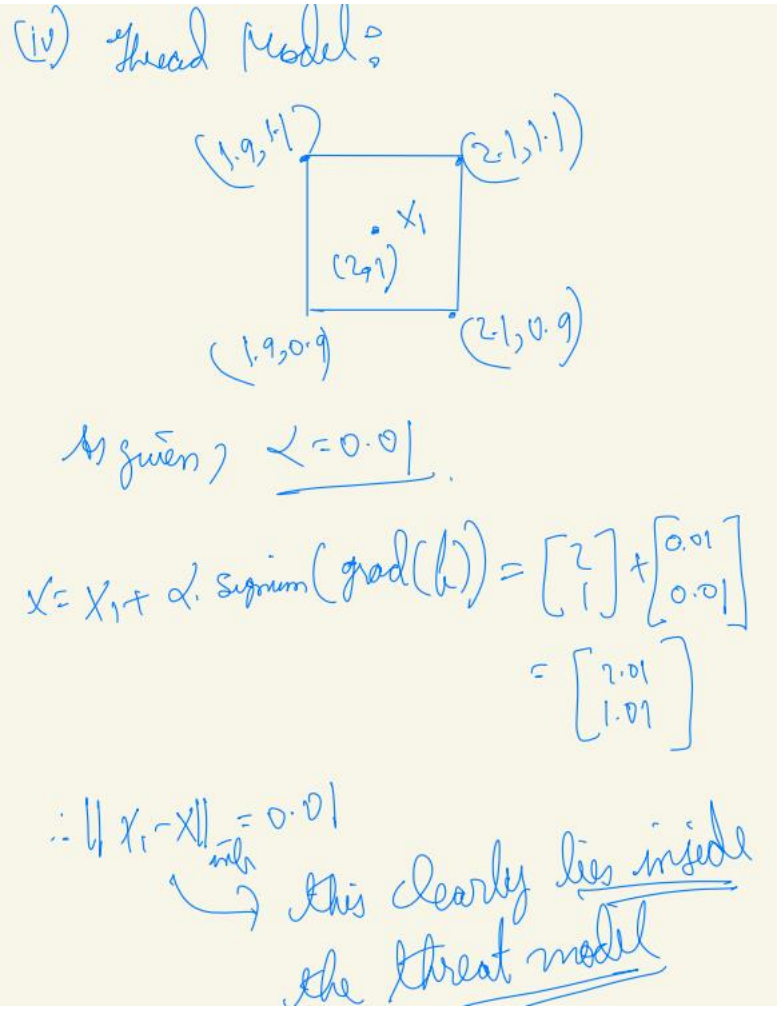
optimum value of x is

$x_{optimum} = (1.9, 0.9)$

for linear model

iv) Find an FGSM attack using these values of step size, α: 0.01, 0.1, 0.25, 0.5 (finding FGSM attack means finding the updated values of x). 1 Which of these FGSM attack lies in the threat model?

Answer:



Now, given $\alpha = 0.1$ -

$$x = x_1 + \alpha \cdot \text{Signum}(\text{grad}(l_1)) = \begin{bmatrix} 2 \\ 1 \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} = \begin{bmatrix} 2.1 \\ 1.1 \end{bmatrix}$$

$\|x_1 - x\|_{inf} = 0.1 \leq 0.1$

$\Rightarrow$ The FGSM attack clearly lies inside the threat model

Now, given $\alpha = 0.25$ -

$$x = x_1 + \alpha \cdot \text{Signum}(\text{grad}(l_1)) = \begin{bmatrix} 2 \\ 1 \end{bmatrix} + \begin{bmatrix} 0.25 \\ 0.25 \end{bmatrix} = \begin{bmatrix} 2.25 \\ 1.25 \end{bmatrix}$$

$\therefore \|x - x_1\|_{inf} = 0.25 > 0.1$

$\therefore$ The FGSM attack lies outside the threat model

Now given, $\alpha = 0.5$ -

$$\therefore x = x_1 + \alpha \cdot \text{Signum}(\text{grad}(l_1)) = \begin{bmatrix} 2 \\ 1 \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 2.5 \\ 1.5 \end{bmatrix}$$

$\therefore \|x_1 - x\|_{inf} = 0.5 > 0.1$

This attack lies outside the threat model

v) Find a PGD attack using α: 0.01.

Answer:

(v) For $\alpha = 0.001$,

$$x_b = x_1 + \alpha(\text{grad}(h)) = \begin{bmatrix} 2 \\ 1 \end{bmatrix} + 0.001 \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2.002 \\ 1.001 \end{bmatrix}$$

Clearly, x_b lies within

$$\varepsilon = 0.1$$

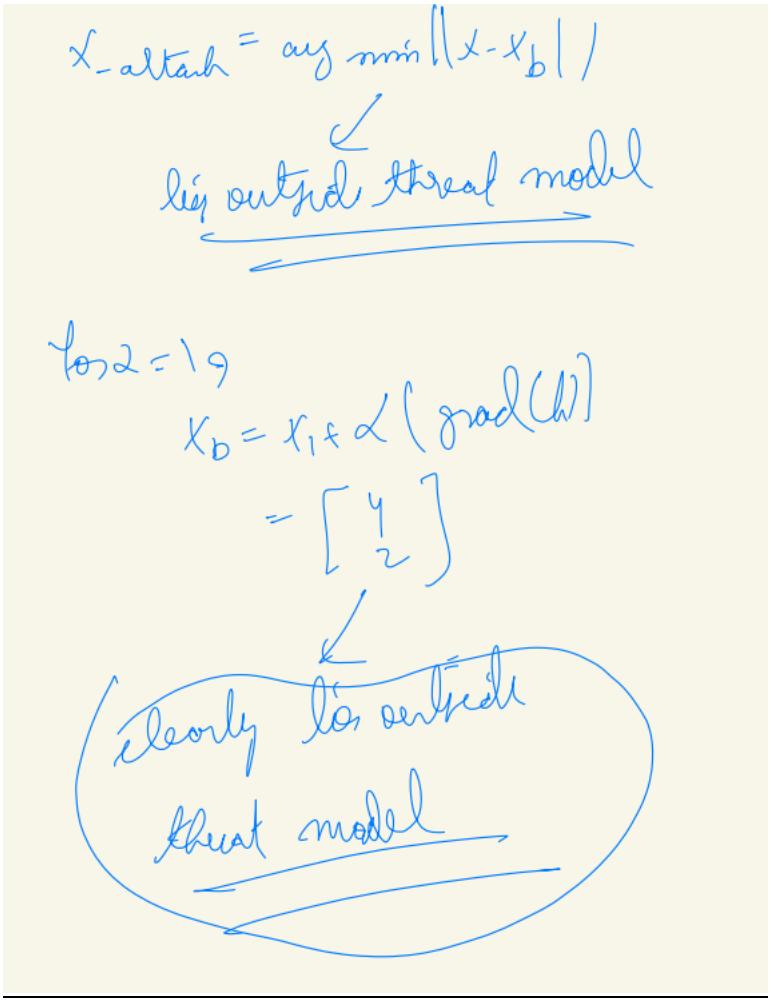
$\therefore$ PGD attack $= x\_attack = [2.002, 1.001]$

For $\alpha = 0.1$,

$$x_b = x_1 + \alpha(\text{grad}(h))$$

$$= \begin{bmatrix} 2 \\ 1 \end{bmatrix} + 0.1 \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2.2 \\ 1.1 \end{bmatrix}$$



vi) What are your observations from the results in (iv) and (v)?

Answer:

When using the actual values of gradient descent in generating adversarial examples, it can be observed that all the points for all given step sizes lie within the allowed threat model. This indicates that the resulting images are not significantly different from the original, non-adversarial images. In contrast, when we use only the signs of the gradient descent, it can lead to a stronger and more effective adversarial attack. This can result in generating new images that are much closer to the targeted image and can significantly mislead the model's predictions. However, when the model is only dealing with two classes, the difference between an untargeted and a targeted attack might not have a significant impact on the overall results. Nonetheless, it is important to consider and evaluate the potential impact of such attacks in the context of the specific task or application.