Exercise 1)

Iteration 1

- We first have to classify the data points with the current $\boldsymbol{w}$ and b

$$f(\mathbf{X}) = \boldsymbol{X}\boldsymbol{w} + b = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 0 & 0 \\ 0 & 2 \\ 1 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} + 0.1 = \begin{bmatrix} 0 \\ 0 \\ 0.1 \\ 0.3 \\ 0.4 \\ 0.4 \end{bmatrix}$$

Then, consider the classification errors for which $y_i f(\boldsymbol{x}_i) < 0$

$$\boldsymbol{y}f(\mathbf{X}) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \circ \begin{bmatrix} 0 \\ 0 \\ 0.1 \\ 0.3 \\ 0.4 \\ 0.4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0.1 \\ -0.3 \\ -0.4 \\ -0.4 \end{bmatrix}$$

The three last patterns are wrongly classified, so they will be used to update $\boldsymbol{w}$ and b

$$\nabla_{\boldsymbol{w}} L_p(\boldsymbol{w}, b) = -\sum_{i:y_i f(\boldsymbol{x}_i)<0} y_i \boldsymbol{x}_i = +1 \begin{bmatrix} 0 \\ 2 \end{bmatrix} + 1 \begin{bmatrix} 1 \\ 2 \end{bmatrix} + 1 \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$$

$$\nabla_b L_p(\boldsymbol{w}, b) = -\sum_{i:y_i f(\boldsymbol{x}_i)<0} y_i = 3$$

Parameter updates

$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla_{\boldsymbol{w}} L_p(\boldsymbol{w}, b)$ and $b \leftarrow b - \eta \nabla_b L_p(\boldsymbol{w}, b)$

$$\boldsymbol{w} = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} - 1 \begin{bmatrix} 3 \\ 5 \end{bmatrix} = \begin{bmatrix} -2.9 \\ -4.9 \end{bmatrix}$$

$$b = 0.1 - 1 * 3 = -2.9$$

Termination condition $\eta(|\nabla_{\boldsymbol{w}} L_p(\boldsymbol{w}, b)| + |\nabla_b L_p(\boldsymbol{w}, b)|) <= \theta$

$$t = 1 * (|3| + |5| + |3|) <= \theta = 0$$

Termination condition is *False* (then, loop continues)

iter	$\boldsymbol{y}f(\boldsymbol{x})$	L_p	$\nabla_{\boldsymbol{w}} L_p, \nabla_b L_p$	$\boldsymbol{w}, b$	t	θ
1.	[0. 0. 0.1 -0.3 -0.4 -0.4]	1.1	[3 5] 3	[-2.9 -4.9] -2.9	11.0	0
2.	[0. 2. -2.9 12.7 15.6 13.6]	2.9	[0 0] -1	[-2.9 -4.9] -1.9	1.0	0
3.	[1. 3. -1.9 11.7 14.6 12.6]	1.9	[0 0] -1	[-2.9 -4.9] -0.9	1.0	0
4.	[2. 4. -0.9 10.7 13.6 11.6]	0.9	[0 0] -1	[-2.9 -4.9] 0.1	1.0	0
5.	[3. 5. 0.1 9.7 12.6 10.6]	0.0	[0 0] 0	[-2.9 -4.9] 0.1	0.0	0

L1-norm

Exercise 2)

For each neuron of the current level the following formula must be calculated:

$$net_i = \sum_{j=1..d} w_{ji} \cdot in_j + w_{0i}$$

which includes a multiplication and a sum for each neuron of the previous level plus the final sum of the bias. Therefore:

Number of hidden layer operations: $8 \cdot (6 + 6 + 1) = 104$

Number of output layer operations: $5 \cdot (8 + 8 + 1) = 85$

104+85=189

Exercise 3)

$net(y_1) = 3{,}25 \cdot 0{,}21 + 2{,}08 \cdot 0{,}61 = 1{,}95$ $\quad out(y_1) = 0{,}88$

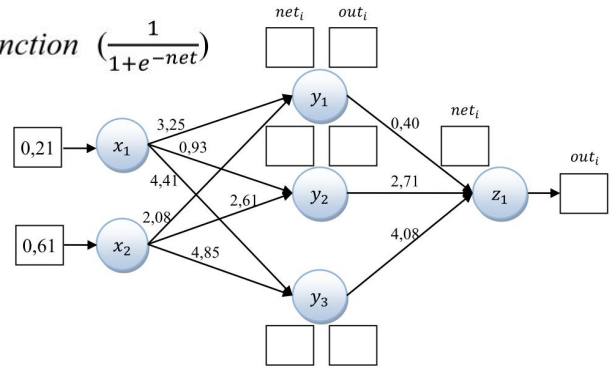
$net(y_2) = 0{,}93 \cdot 0{,}21 + 2{,}61 \cdot 0{,}61 = 1{,}79$ $\quad out(y_2) = 0{,}86$

$net(y_2) = 4{,}41 \cdot 0{,}21 + 4{,}85 \cdot 0{,}61 = 3{,}88$ $\quad out(y_3) = 0{,}98$

$net(z_1) = 0{,}40 \cdot 0{,}88 + 2{,}71 \cdot 0{,}86 + 4{,}08 \cdot 0{,}98 = 6{,}68$ $\quad out(z_1) = 1{,}00$

1/(1+exp(-1.95))

$y_1 = f_1(w_{(x1)1}x_1 + w_{(x2)1}x_2)$ $\quad$ *standard logistic function* $(\frac{1}{1+e^{-net}})$



Exercise 4)

	C_1	C_2	C_3	C_4	C_5	MV
p_1	1	2	1	1	3	**1**
p_2	1	2	2	3	4	**2**
p_3	3	2	1	3	3	**3**

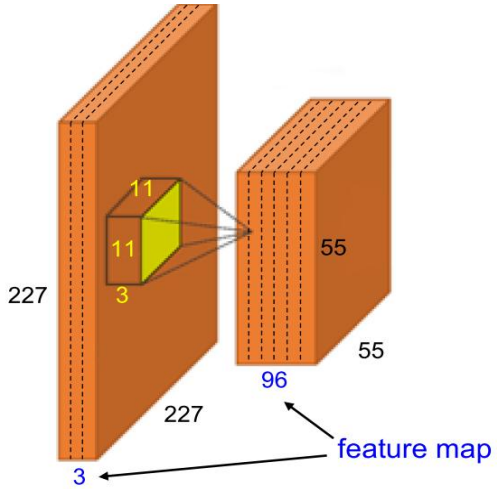
Exercise 5)

	C_1			C_2			C_3			sum rule				product rule				max rule				min rule			
	A	B	C	A	B	C	A	B	C	A	B	C	Out	A	B	C	Out	A	B	C	Out	A	B	C	Out
p_1	0,15	0,81	0,04	0,02	0,56	0,42	0,54	0,12	0,34	0,71	**1,49**	0,80	**B**	0,00	**0,05**	0,01	**B**	0,54	**0,81**	0,42	**B**	0,02	**0,12**	0,04	**B**
p_2	0,31	0,24	0,45	0,54	0,41	0,05	0,02	0,03	0,95	0,87	0,68	**1,45**	**C**	0,00	0,00	**0,02**	**C**	0,54	0,41	**0,95**	**C**	0,02	0,03	**0,05**	**C**
p_3	0,42	0,46	0,12	0,77	0,21	0,02	0,41	0,30	0,29	**1,6**	0,97	0,43	**A**	**0,13**	0,03	0,00	**A**	**0,77**	0,46	0,29	**A**	**0,41**	0,21	0,02	**A**

Attention to the min rule formula, I take the maximum of the minimums

Exercise 6)

Each neuron of the output level (96 x 55 x 55) is connected with many neurons of the input level equal to the size of the filter (3 x 11 x 11). Therefore the total number of connections (96 x 55 x 55) • (3 x 11 x 11) = 105 415 200.
The total number of weights, on the other hand, is much smaller since in a CNN the weights of each filter are shared by all the neurons contained in the same feature map. Since the number of feature maps equals 96, and the number of inputs for each filter equals (3 x 11 x 11), the total number of weights (without considering the bias) is (3 x 11 x 11) x 96 = 34 848.



Exercise 7)

		Predicted class: Spam	Predicted class: not Spam
Actual class	Spam	8	2
	not Spam	16	974

	Predicted Positive	Predicted Negative	
Actual Positive	TP *True Positive*	FN *False Negative*	Sensitivity $\frac{TP}{(TP+FN)}$
Actual Negative	FP *False Positive*	TN *True Negative*	Specificity $\frac{TN}{(TN+FP)}$
	Precision $\frac{TP}{(TP+FP)}$	Negative Predictive Value $\frac{TN}{(TN+FN)}$	Accuracy $\frac{TP+TN}{(TP+TN+FP+FN)}$

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$\text{precision} = \frac{8}{8+16} = \frac{1}{3}$$

$$\text{recall} = \frac{8}{8+2} = \frac{4}{5}$$

Exercise 8)

If the layers are "fully connected" as in MLP networks, the number of connections is equal to $L_i \times L_{i+1}$ therefore $10 \times 8 = 80$ total connections.
If the layers are of a CNN with "receptive field" equal to 3, then just multiply 3 by the number of neurons of the level (i + 1) so $3 \times 8 = 24$ total connections.
In the case of the MLP network the number of the weights is equal to the number of connections (therefore 80) while for the CNN it is equal to the size of the receptive field (since the weights are shared).

Exercise 9)

With K = 4 the 6000 patterns are divided into 4 folds of 1500 patterns each. Trainings step will be performed using each time a different partition as validation set and the remaining three fold as training set. Therefore, in each fold: the training set consists of 4500 patterns; the test set consists of 1500 patterns.

Exercise 10)

Yes	No
Yes | TP = 45 | FN = 5 | P = 50	
No | FP = 10 | TN = 40 | N = 50	
| P' = 55 | N' = 45 | P+N = 100	

Correct rate: (TP+TN)/(P+N) = 85/100 = 0.85

Precision (precision): TP/(TP+FP) = 45/55 = 0.818

Recall rate (recall): TP/P = 45/50 = 0.9

Exercise 11)

6 Input neurons
8 Intermediate (hidden) neurons
5 Output neurons

In the case of an MLP neural network, the number of weights is equal to the number of connections present. The number of connections (and therefore of weights) between two consecutive levels (i and i +1) can be calculated as the product of the number of neurons of the level *i* and the number of neurons of the level *i +1*. If you are using the bias, the number of neurons of each level will have to be increased by one.
Therefore the total number of weights will be equal to: (6 + 1) * 8 + (8 + 1) * 5 = 101.

Exercise 12)

Available information:
precision of 0.5;
recall of 0.35;
true positive + false negative = 20.

TP=x FP=y -> number of patterns classified as positive is 7+7=14

$$Precision = \frac{tp}{tp + fp}$$

$$Recall = \frac{tp}{tp + fn}$$

P = 0.50

R = 0.35

0.35 = x / 20

x = 20 * 0.35 = 7

0.50 = 7 / (x + y)

y = 7

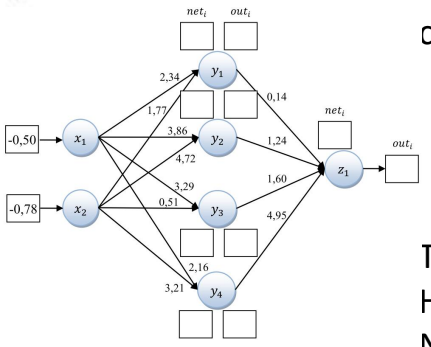
x+y=14

we know that recall=0.35 and that tp+fn=20, we can calculate the value of tp=x. Then we use the Precision formula to calculate the value of fp=y

Exercise 13)

$net(y_1) = 2{,}34 \cdot (-0{,}50) + 1{,}77 \cdot (-0{,}78) = -2{,}55$
$net(y_2) = 3{,}86 \cdot (-0{,}50) + 4{,}72 \cdot (-0{,}78) = -5{,}61$
$net(y_3) = 3{,}29 \cdot (-0{,}50) + 0{,}51 \cdot (-0{,}78) = -2{,}04$
$net(y_4) = 2{,}16 \cdot (-0{,}50) + 3{,}21 \cdot (-0{,}78) = -3{,}58$

$out(y_1) = 0{,}07$
$out(y_2) = 0{,}00$
$out(y_3) = 0{,}12$
$out(y_4) = 0{,}03$



$net(z_1) = 0{,}14 \cdot 0{,}07 + 1{,}24 \cdot 0{,}00 + 1{,}60 \cdot 0{,}12 + 4{,}95 \cdot 0{,}03 = 0{,}35$ $\quad out(z_1) = 0{,}59$

$$standard\ \ logistic\ \ function\ \ (\frac{1}{1+e^{-net}})$$

Exercise 14)

Each neuron of the output level (32 x 64 x 64) is connected with many neurons of the input level equal to the size of the filter (4 x 13 x 13). Therefore the total number of connections (32 x 64 x 64) * (4 x 13 x 13) = 88604672. The total number of weights, on the other hand, is much smaller since in a CNN the weights of each filter are shared by all the neurons contained in the same feature map. Since the number of feature maps equals 32, and the number of inputs for each filter equals (4 x 13 x 13), the total number of weights (considering the bias) is (4 x 13 x 13 + 1) x 32 = 21664.

Exercise 15)

	C_1	C_2	C_3	C_4	C_5	MV
p_1	3	1	1	3	3	**3**
p_2	2	3	1	4	2	**2**
p_3	4	1	4	2	3	**4**

Exercise 16)

The following formula is used to calculate the size of each output feature map

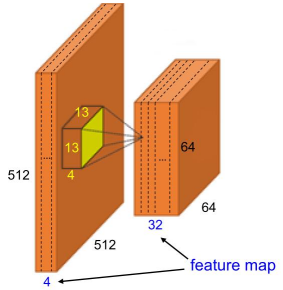
$$W_{out} = \frac{W_{in} - F + 2 \cdot Padding}{Stride} + 1$$

By substituting Filter size (F) = 6, Win = 28 or Win = 32 depending on the considered dimension, Padding = 3 and Stride = 2, we obtain:

$$Width = \frac{28 - 6 + 2 \cdot 3}{2} + 1 = 15$$

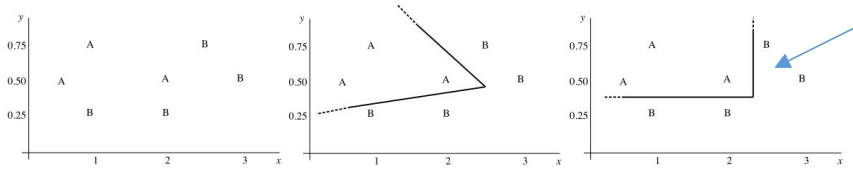
$$Height = \frac{32 - 6 + 2 \cdot 3}{2} + 1 = 17$$

Therefore the size of each output volume feature map will be 15 x 17 (Width x Height).
Note that the depth of the Input volume, which corresponds to the depth of the filter (in this case 4), is an independent value and therefore not useful for calculating the size of each Output volume feature map.



Exercise 17)

thus a perceptron cannot provide a consistent solution.

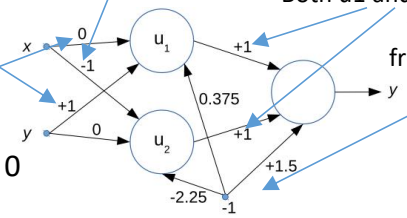


simply y> = 0.375 & x <= 2.25 allow you to divide the space into two areas, where in each area there are only patterns of one class.

weights of the input connections of the hidden units can be set such that their output is 1 in the respective half-planes containing the examples of class A, and therefore the output unit will have to implement the logical AND function. More precisely, denoting the hidden units with u_1 and u_2, their activation functions can output the value 1 when $y \geq 0.375$ (u_1) and when $x \leq 2.25$ (u_2). The corresponding perceptron network is shown below:

-1 represents <=
+1 represents >=

U1 isn't related to x -> weight= 0



Both u1 and u2 must be +1

the factor (-1 * 1.5) prevents output from being 1 if only one of u1 and u2 has an ouput of 1

Since the attribute values are continuous, infinitely many consistent hypotheses can be obtained

Exercise 18)

DocID	1	2	3	4	5	6	7	8	9	10	11	12
Judge 1	0	0	1	1	1	1	1	1	0	0	0	0
Judge 2	0	0	1	1	0	0	0	0	1	1	1	1

AI classifies as relevant {4, 5, 6, 7, 8}.
a document is considered relevant only if the two judges agree.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

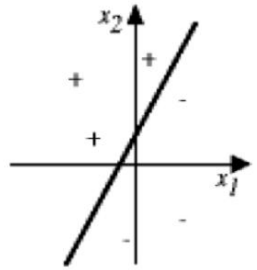
$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

P = 1/5 only document 4 is a true positive, the other retrieved documents are false positives
R = 1/2 the only false negative is {3} (it is not retrieved)

Exercise 19)

If the layer are "completely connected" as in MLP networks, the number of even connections $L_i \times L_{i+1}$ therefore $12 \times 9 = 108$ total connections.
If the layers are of a CNN with "receptive field" equal to 5 then just multiply 5 by the number of neurons of the layer (i + 1) so $5 \times 9 = 45$ total connections.
As for the weights, in the case of the MLP network the number is equal to the number of connections (therefore 108) while for the CNN it is equal to the size of the receptive field (5, the weights are shared).

Exercise 20)



The output of the perceptron is

$$o = sgn(w_0 + w_1 x_1 + w_2 x_2)$$

The equation of the decision surface (the line) is

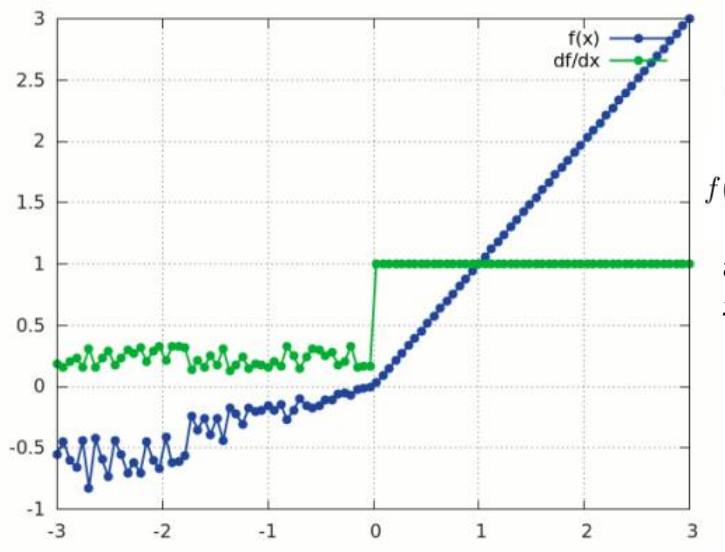
$$w_o + w_1 x_1 + w_2 x_2 = 0$$

We know the coordinates of 2 points of this line: A=(-1,0) and B=(0,2). Therefore, the equation of the line is

$$\frac{x_1 - x_{1A}}{x_{1B} - x_{1A}} = \frac{x_2 - x_{2A}}{x_{2B} - x_{2A}} \rightarrow \frac{x_1 - (-1)}{0 - (-1)} = \frac{x_2 - 0}{2 - 0} \rightarrow x_1 + 1 = \frac{x_2}{2} \rightarrow 2 + 2x_1 - x_2 = 0$$

So 2, 2, -1 are possible values for the weights w_0, w_1, and w_2, respectively. To check if their signs are correct, consider a point on one side of the line, for instance the origin O=(0,0). The output of the perceptron for this point has to be negative, but the output of the perceptron using the candidate weights is positive. Therefore, we need to negate the previous values and conclude that $w_0 = -2, w_1 = -2, w_2 = 1$.

Exercise 21)



α drawn from a uniform distribution

$$\alpha \sim \mathcal{U}(a,b)$$

$$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

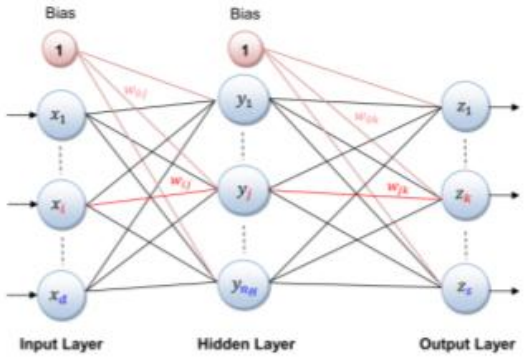
a=0.125, b=0.375

$x \in [-3,3]$

the blue curve represents the trend of the original function, as you can see if x is greater than zero the function assumes the value x, if the value of x is less than zero you have to multiply the value of x by a Random value between 0.125 and 0.375 . So the value of the function will no longer be equal to X, but will be closer to Zero.
the green curve represents the value of the derivative, if the value of x is greater than 0 the derivative is 1, therefore the function is always 1; if the value of x is less than 0 the value of the derivative is alpha, therefore the value of the derivative assumes a random value between A and B

Exercise 22)

In the MLP neural network the number of weights is equal to the number of connections. The number of connections (and therefore of weights) between two consecutive levels (*i* and *i + 1*) can be calculated as the product of the number of neurons of the level *i* by the number of neurons of the level *i + 1*. In the case of using the bias, the number of neurons of each level *i* will have to be increased by one.
Therefore the total number of weights will be equal to: (24 + 1) * 48 + (48 + 1) * 3 = 1347.



Exercise 23)

$$\text{Depth } 0 = 187 \cdot 0.89 + 22 \cdot 0.87 + 167 \cdot 0.3 + 237 \cdot 0.52 + 87 \cdot 0.38 = 391.97$$
$$\text{Depth } 1 = 248 \cdot 0.24 + 43 \cdot 0.9 + 12 \cdot 0.07 + 19 \cdot 0.64 + 42 \cdot 0.01 + 157 \cdot 0.41 = 176.01$$
$$\text{Depth } 2 = 187 \cdot 0.71 + 66 \cdot 0.4 + 88 \cdot 0.1 + 106 \cdot 0.3 = 199.77$$

result = $391.97 + 176.01 + 199.77 = 767.75$

we have to multiply element by element each channel, separately, of the filter with the image portion. Getting these three values; finally, we have to add these values to obtain the final result of the convolution

Input

Depth 0

197	103	42	252	27	78	205
114	57	2	195	7	1	130
97	71	179	60	187	22	21
86	84	187	229	208	167	237
25	177	236	250	25	9	87
217	175	190	175	23	10	69
67	127	246	142	4	125	87

Depth 1

124	164	158	18	229	152	110
19	111	22	75	167	224	88
136	21	201	237	248	43	136
151	245	140	163	12	207	19
212	197	87	203	42	149	157
12	78	232	52	113	232	198
64	167	99	112	42	236	186

Depth 2

105	45	160	12	81	207	228
174	18	111	216	200	91	62
170	191	128	124	74	187	123
199	224	184	134	66	193	87
77	41	50	226	226	88	106
151	182	191	216	198	184	93
60	120	91	168	141	136	150

Filtro

Depth 0

0.89	0.87	0
0	0.3	0.52
0	0	0.38

Depth 1

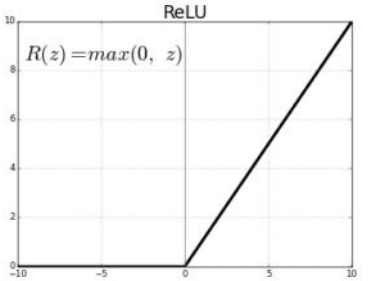
0.24	0.9	0
0.07	0	0.64
0.01	0	0.41

Depth 2

0	0.71	0
0.4	0	0
0	0.1	0.3

Exercise 24)

$$a^{(j)} = \max(W^{(1)}X + b^{(1)}, 0)$$



Exercise 25)

In the backward step we use the gradient of the activation function, therefore the output will be 0.01 (value of a when x <0) otherwise 1

$$y_i = f(x_i) = \begin{cases} ax_i, & x_i < 0 \\ x_i, & x_i \geq 0 \end{cases} \qquad \frac{dy_i}{dx_i} = f'(x_i) = \begin{cases} a, & x_i < 0 \\ 1, & x_i \geq 0 \end{cases}$$

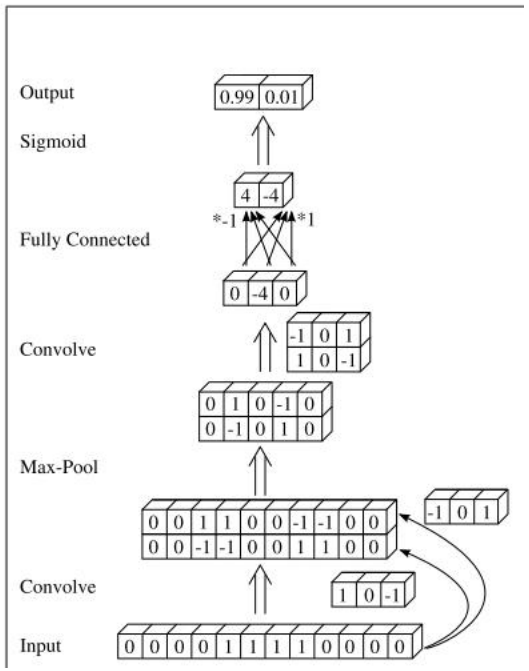
Exercise 26)

32x32x5 image and a 5x5x5 filter,
no padding; size output: 28x28x1

$$W_{out} = \frac{(W_{in} - F + 2 \cdot Padding)}{Stride} + 1$$

$Stride = 1,\ Padding = 0,\ W_{in} = 32,$
$F = 5 \rightarrow W_{out} = 28.$

Exercise 27)



- First convolutional layer with filters $F_0^1 = (-1, 0, 1)$ and $F_1^1 = (1, 0, -1)$ that generates two outp feature maps from a single input feature map. Use *valid* mode for convolutions.
- Max-pooling layer with stride 2 and filter size 2. Note that max-pooling pools each feature map separ tely.
- Convolutional layer with convolutional kernel $F_0^2 = ((-1, 0, 1), (1, 0, -1))$ of size $2 \times 3 \times 1$.
- Fully connected layer that maps all inputs to two outputs. The first output is calculated as the negati sum of all its inputs, and the second layer is calculated as the positive sum of all its inputs.
- Sigmoidal activation function $\phi(z) = \frac{1}{1+e^{-z}}$

Calculate the response of the CNN for the input $(0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0)$

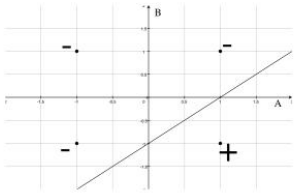
Exercise 28)

A	B	O = A∧¬B
-1	-1	-1
-1	1	-1
1	-1	1
1	1	-1

One of the correct decision surfaces (any line that separates the positive point from the negative points would be fine) is shown in the following picture.

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1}$$

$$x_1 \neq x_2 \quad y_1 \neq y_2$$



The line crosses the A axis at 1 and the B axis -1. The equation of the line is

$$\frac{A-0}{1-0} = \frac{B-(-1)}{0-(-1)} \rightarrow A = B + 1 \rightarrow 1 - A + B = 0$$

So 1, -1, 1 are possible values for the weights w_0, w_1, and w_2, respectively. Using this values the output of the perceptron for A=1, B=-1 is negative. Therefore, we need to negate the weights and therefore we can conclude that $w_0 = -1, w_1 = 1, w_2 = -1$.

The output of the perceptron is

$$o = sgn(w_0 + w_1 x_1 + w_2 x_2)$$

The equation of the decision surface (the line) is

$$w_o + w_1 x_1 + w_2 x_2 = 0$$

Exercise 29)

Perceptron's Learning Rule

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = \eta(t - o)x_i$$

Consider a Perceptron with only one input x_1, weight $w_1 = 0.5$, threshold $\theta = 0$ and learning rate $\eta = 0.6$. Consider also the training example $\{x_1 = -1,\ t = 1\}$. For now, let's temporarily ignore the learning of the threshold and consider it fixed.

Determine:

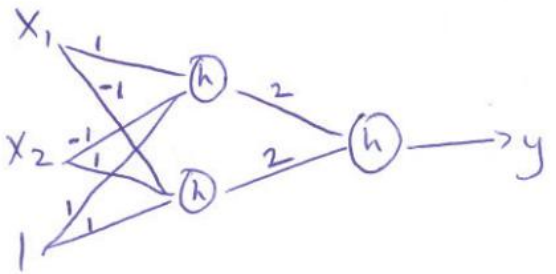
- The output of the Perceptron for the input -1.
  $w_1 x_1 = 0.5(-1) = -0.5 < \theta \rightarrow o = 0$
- The new weight w_1 after applying the learning rule.
  $\Delta w_1 = \eta(t - o)x_1 = 0.6(1 - 0)(-1) = -0.6 \rightarrow w_1 = 0.5 - 0.6 = -0.1$
- The new output of the Perceptron for the input -1.
  $w_1 x_1 = -0.1(-1) = 0.1 > \theta \rightarrow o = 1$

Exercise 30)

A	B	A **XOR** B
0	0	0
0	1	1
1	0	1
1	1	0

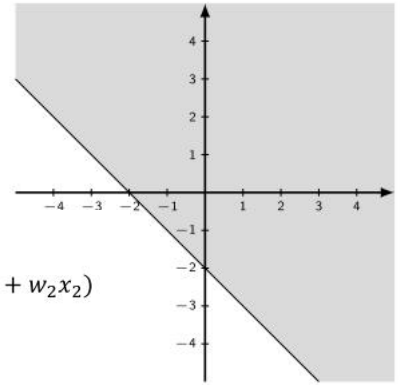
The activation function is the following (a=2):

$$h(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w}^T\mathbf{x} >= a \\ 0 & \text{otherwise} \end{cases}$$



three binary inputs X1, X2 and 1
output Y implements X1 XOR X2.

Exercise 31)



The output of the perceptron is

$$o = sgn(w_0 + w_1 x_1 + w_2 x_2)$$

The equation of the decision surface (the line) is

$$w_o + w_1 x_1 + w_2 x_2 = 0$$

if x1=0 then 2+x2=0 -> x2=-2;
if x2=0 then 2+x1=0 -> x1=-2
In the point (0,0) (remember that $w_0 = 2$; $w_1 = 1$; $w_2 = 1$) o=sgn(2)
therefore positive class

Exercise 32)

baseline: perceptron with the following weights (w0 = 2; w1=1; w2=1)

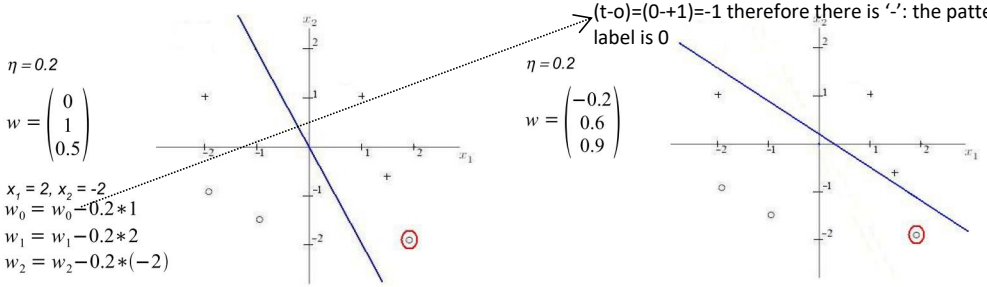
	$(w_0, w_1, w_2)^T$	same hyperplane	same classification
(I)	$(1, 0.5, 0.5)^T$	×	×
(II)	$(200, 100, 100)^T$	×	×
(III)	$(\sqrt{2}, \sqrt{1}, \sqrt{1})^T$		
(IV)	$(-2, -1, -1)^T$	×	

Exercise 33)

Perceptron's Learning Rule

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = \eta(t - o)x_i$$



(t-o)=(0-+1)=-1 therefore there is '-': the pattern is classified as positive (i.e. +1) but its true label is 0

Exercise 34)

S-shaped Rectified Linear Unit, or SReLU

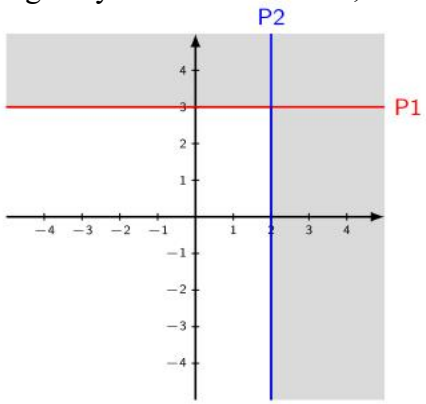
$$h(x_i) = \begin{cases} t_i^r + a_i^r(x_i - t_i^r), & x_i \geq t_i^r \\ x_i, & t_i^r > x_i > t_i^l \\ t_i^l + a_i^l(x_i - t_i^l), & x_i \leq t_i^l \end{cases}$$

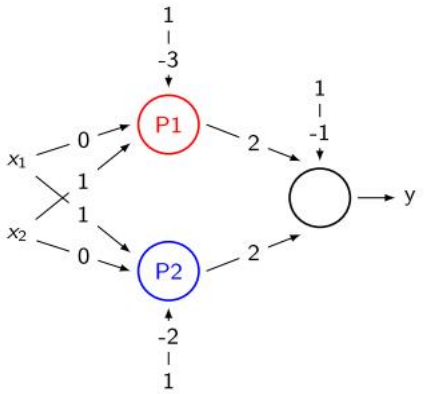
$$\frac{\partial h(x_i)}{\partial t_i^r} = I\{x_i \geq t_i^r\}(1 - a_i^r)$$
$$\frac{\partial h(x_i)}{\partial a_i^r} = I\{x_i \geq t_i^r\}(x_i - t_i^r)$$
$$\frac{\partial h(x_i)}{\partial t_i^l} = I\{x_i \leq t_i^l\}(1 - a_i^l)$$
$$\frac{\partial h(x_i)}{\partial a_i^l} = I\{x_i \leq t_i^l\}(x_i - t_i^l)$$

where $I\{\cdot\}$ is an indicator function and $I\{\cdot\} = 1$ when the expression inside holds true, otherwise $I\{\cdot\} = 0$.

Exercise 35)

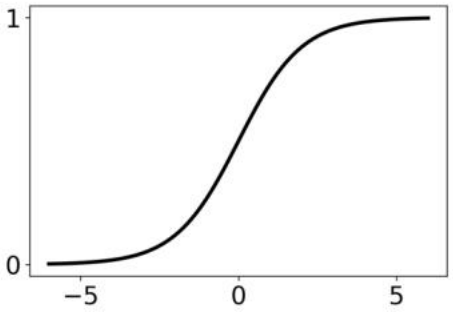
The network has to create the decision surface reported in the below figure.
P1: x2>3 it is managed by the first neuron, notice that the weight of the related bias is -3
P2: x1>2 it is managed by the second neuron, notice that the weight of the related bias is -2

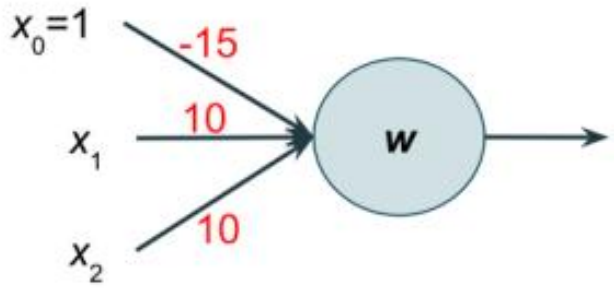




Exercise 36)

For the single unit in the output layer, if its three weight parameters are set to $\boldsymbol{w}$ = (-15, 10, 10) then the neural network models the **AND** function. Why? If either x_1 or x_2 is 0 then the input to the sigmoid activation function will be -5 or -15 and the output will be approximately 0. If both x_1 and x_2 are 1 then the input to the sigmoid activation function will be 5 and the output will be approximately 1.





Exercise 37)

The number of times that the weights are updated corresponds to the total number of performed iterations.
There are 1000 patterns divided into mini-batches of 25 patterns each, then 1000/25=40 iterations are performed. 8 epochs are performed, the total number of iterations can be calculated as 8 * 40 = 320.

Exercise 38)

$$D\left[\frac{f(x)}{g(x)}\right] = \frac{f'(x)\cdot g(x) - f(x)\cdot g'(x)}{[g(x)]^2}, \text{ con } g(x) \neq 0$$

- Standard logistic function

$$f(net) = \frac{1}{1+e^{-net}}$$

$$f(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x}$$

$$\frac{d}{dx}f(x) = \frac{e^x\cdot(1+e^x) - e^x\cdot e^x}{(1+e^x)^2} = \frac{e^x}{(1+e^x)^2} = f(x)(1-f(x))$$

$$f'(net) = \frac{\partial}{\partial net}\left(\frac{1}{1+e^{-net}}\right) = \frac{e^{-net}}{(1+e^{-net})^2} = f(net)\big(1-f(net)\big)$$

Exercise 39)

We define the following similarity between two patterns x and y: sim = f(A*(|g(x)-g(y)|)+b)
g: neural network, g(x) its output given input x
A: weights b: bias f: activation function
In case A has positive values, is sim a metric?

IDENTITY
d(x,x) = f(A*(|g(x)-g(x)|)+b) = f(A*0+b) = f(b)
we fix b =0 and choose f(0) = 0

SIMMETRY straightforward
https://en.wikipedia.org/wiki/Absolute_difference
|x-y|=|y-x|
TRIANGLE INEQUALITY
We impose A with positive elements, therefore A*v is increasing for each component

|g(x)-g(y)| <= |g(x)-g(z)| + |g(z)-g(y)|
so, since A has positive elements
A*(|g(x)-g(y)|) <= A*(|g(x)-g(z)| + |g(z)-g(y)|) = A*(|g(x)-g(z)|) + A*(|g(z)-g(y)|)
If f(a+b) <= f(a) + f(b) and f is increasing
d(x,y) = f(A*(|g(x)-g(y)|)) <= f(A*(|g(x)-g(z)|) + A*(|g(z)-g(y)|)) <= f(A*(|g(x)-g(z)|))
+ f(A*(|g(z)-g(y)|)) = d(x,z) + d(z,y)
Moreover, f() must have non negative values

A **metric space** is an ordered pair (M, d) where M is a set and d is a metric on M, i.e., a function

$$d : M \times M \to \mathbb{R}$$

such that for any $x, y, z \in M$, the following holds:[2]

1. $d(x, y) = 0 \iff x = y$ identity of indiscernibles
2. $d(x, y) = d(y, x)$ symmetry
3. $d(x, z) \leq d(x, y) + d(y, z)$ subadditivity or triangle inequality

$$d(x, y) \geq 0$$

$$|a - c| = |a - p + p - c| \leq |a - p| + |p - c|$$

Convex function:

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2) \quad \forall x_1, x_2 \in I \text{ e } t \in [0,1]$$

Exercise 40)

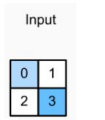
$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{j=0}^{T} \gamma^j r_{t+j+1}$$

γ = 0: only r_{t+1} is considered as reward
γ = 1: all the future rewards have the same weight

Exercise 41)

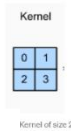
Transposed Convolutions

Transposed Convolutions are used to upsample the input feature map to a desired output feature map using some learnable parameters.
The basic operation that goes in a transposed convolution is explained below:
1. Consider a 2x2 encoded feature map which needs to be upsampled to a 3x3 feature map.



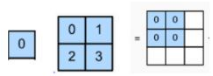
Input Feature Map

2. We take a kernel of size 2x2 with unit stride and zero padding.

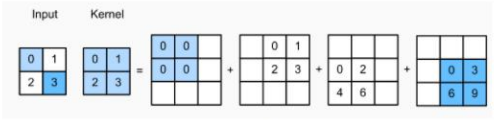


Kernel of size 2×2

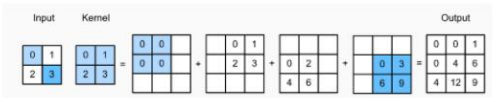
3. Now we take the upper left element of the input feature map and multiply it with every element of the kernel as shown in figure



4. Similarly, we do it for all the remaining elements of the input feature map as depicted in figure



5. As you can see, some of the elements of the resulting upsampled feature maps are over-lapping. To solve this issue, we simply add the elements of the over-lapping positions.



The Complete Transposed Convolution Operation

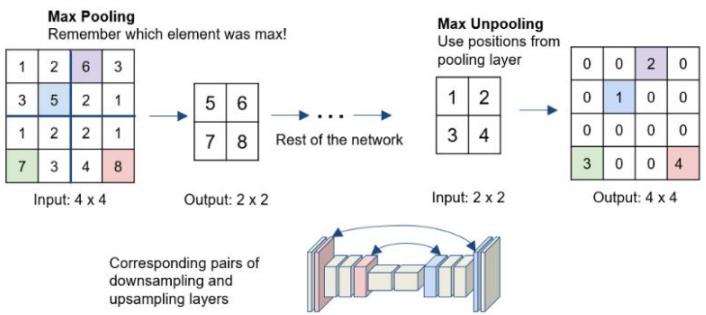
6. The resulting output will be the final upsampled feature map having the required spatial dimensions of 3x3.

Transposed convolution is also known as Deconvolution which is not appropriate as deconvolution implies removing the effect of convolution which we are not aiming to achieve.

It is also known as upsampled convolution which is intuitive to the task it is used to perform, i.e upsample the input feature map.

Exercise 42)

Max-Unpooling: The Max-Pooling layer in CNN takes the maximum among all the values in the kernel. To perform max-unpooling, first, the index of the maximum value is saved for every max-pooling layer during the encoding step. The saved index is then used during the Decoding step where the input pixel is mapped to the saved index, filling zeros everywhere else.



Exercise 43)

$$\frac{\partial H}{\partial p_i}(\mathbf{y},\mathbf{p}) = \frac{-\sum_{j=1}^{n} y_j \log p_j}{\partial p_i} = -y_i/p_i$$

$$\frac{d}{dx}[\log(x)] = \frac{1}{x} \quad \forall x > 0$$

$$\frac{\partial}{\partial x}\left((x-a)^2\right) = 2(x-a)$$

Exercise 44)

Consider an 1-layer neural network $\mathbf{y} = g(\mathbf{A}\mathbf{x})$ with input $\mathbf{x}$, output $\mathbf{y}$, network weight $\mathbf{A}$ and output function g. Let's first assume the input and the network weights are

$$\mathbf{A} = \begin{pmatrix} 3.0 & 2.0 \end{pmatrix} \in \mathbb{R}^{1\times 2} \qquad \mathbf{x} = \begin{pmatrix} 2.0 & 1.0 & -1.0 \\ 4.0 & -2.0 & 0.0 \end{pmatrix} \in \mathbb{R}^{2\times 3}$$

where $\mathbf{x}$ are 2D points with a **minibatch** size of 3.

i) Assume the function g is linear, i.e. $g(\mathbf{A}\mathbf{x}) = \mathbf{A}\mathbf{x}$. Given the target output $\mathbf{t}$ and the loss function $\mathcal{L}$, perform weight update. Assume a learning rate of 1. Please provide the loss *before* and *after* the weight update.

$$\mathbf{t} = \begin{pmatrix} 15 & 3 & 1 \end{pmatrix} \in \mathbb{R}^{1\times 3} \qquad \mathcal{L} = \frac{1}{2}\sum_j (y_j - t_j)^2$$

First calculate the forward pass and obtain the loss. Next, you can derive the gradients of loss wrt. to every element of the network weight $\frac{\partial \mathcal{L}}{\partial a_i}$. Make sure to include all steps of your derivation. Once the network weight is updated, calculate the forward pass again to acquire the loss.

Set $\mathbf{A} = \begin{pmatrix} a_1 & a_2 \end{pmatrix}$, $\mathbf{y} = \mathbf{A}\mathbf{x} = \begin{pmatrix} 2a_1 + 4a_2 & a_1 - 2a_2 & -a_1 \end{pmatrix}$. Therefore, the loss is:

$$\mathcal{L} = \frac{1}{2}(2a_1 + 4a_2 - 15)^2 + \frac{1}{2}(a_1 - 2a_2 - 3)^2 + \frac{1}{2}(-a_1 - 1)^2$$

At time step t, the loss is:

$$\mathcal{L}^t = \frac{1}{2}(1 + 16 + 16) = 16.5$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{A}} = \frac{\partial \mathcal{L}}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{A}}$$

where

$$\frac{\partial \mathcal{L}}{\partial \mathbf{y}} = \begin{pmatrix} \frac{\partial \mathcal{L}}{\partial y_1} & \frac{\partial \mathcal{L}}{\partial y_2} & \frac{\partial \mathcal{L}}{\partial y_3} \end{pmatrix} = \begin{pmatrix} y_1 - t_1 & y_2 - t_2 & y_3 - t_3 \end{pmatrix} = \begin{pmatrix} -1 & -4 & -4 \end{pmatrix}$$

$$\frac{\partial \mathbf{y}}{\partial \mathbf{A}} = \begin{pmatrix} \frac{\partial y_1}{a_1} & \frac{\partial y_1}{a_2} \\ \frac{\partial y_2}{a_1} & \frac{\partial y_2}{a_2} \\ \frac{\partial y_3}{a_1} & \frac{\partial y_3}{a_2} \end{pmatrix} = \begin{pmatrix} 2 & 4 \\ 1 & -2 \\ -1 & 0 \end{pmatrix}$$

so

$$\frac{\partial \mathcal{L}}{\partial \mathbf{A}} = \frac{\partial \mathcal{L}}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{A}} = \begin{pmatrix} -1 & -4 & -4 \end{pmatrix} \begin{pmatrix} 2 & 4 \\ 1 & -2 \\ -1 & 0 \end{pmatrix} = \begin{pmatrix} -2 & 4 \end{pmatrix}$$

$$a_1^{t+1} = a_1^t - \eta \cdot \frac{1}{3}\frac{\partial \mathcal{L}^t}{\partial a_1} = 3 + \frac{2}{3} = 3.667$$

$$a_2^{t+1} = a_2^t - \eta \cdot \frac{1}{3}\frac{\partial \mathcal{L}^t}{\partial a_2} = 2 - \frac{4}{3} = 0.667$$

The update $\mathbf{A}^{t+1} = \begin{pmatrix} 3.667 & 0.667 \end{pmatrix}$ and new $\mathcal{L}^{t+1}$ becomes:

$$\mathcal{L}^{t+1} = 23.611$$

to use the new values of A in the previous loss definition

Exercise 45) this exercise is based on the previous one

$\mathbf{A} = \begin{pmatrix} 3.0 & 2.0 \end{pmatrix} \in \mathbb{R}^{1\times 2}$ $\quad \mathbf{x} = \begin{pmatrix} 2.0 & 1.0 & -1.0 \\ 4.0 & -2.0 & 0.0 \end{pmatrix}$

$\mathcal{L} = \frac{1}{2}\sum_j (y_j - t_j)^2$ $\quad \mathbf{t} = \begin{pmatrix} 15 & 3 & 1 \end{pmatrix}$

The derivative w.r.t. $\mathbf{A}$: y=g(Ax), where g is ReLu

$$\frac{\partial \mathcal{L}^t}{\partial a_1} = \frac{1}{2}\sum_j \frac{\partial}{\partial a_1}(\max(0, \mathbf{A}x_j) - t_j)^2 = \sum_j (\max(0, \mathbf{A}x_j) - t_j) \cdot \frac{\partial}{\partial a_1} \max(0, \mathbf{A}x_j)$$

$$= (\max(0, \mathbf{A}x_1) - t_1) \cdot \frac{\partial}{\partial a_1} \max(0, \mathbf{A}x_1) = -2$$

$$\frac{\partial \mathcal{L}^t}{\partial a_2} = \frac{1}{2}\sum_j \frac{\partial}{\partial a_2}(\max(0, \mathbf{A}x_j) - t_j)^2 = \sum_j (\max(0, \mathbf{A}x_j) - t_j) \cdot \frac{\partial}{\partial a_2} \max(0, \mathbf{A}x_j)$$

$$= (\max(0, \mathbf{A}x_1) - t_1) \cdot \frac{\partial}{\partial a_2} \max(0, \mathbf{A}x_1) = (2a_1 + 4a_2 - 15) \cdot 4 = -4$$

You can notice that the gradients for x_3 did not contribute at all to the final gradients.

chain rule is applied within the summation

$\frac{\partial \mathcal{L}}{\partial a_i} = \frac{\partial \mathcal{L}}{\partial g}\frac{\partial g}{\partial a_i}$

$\frac{\partial}{\partial x}((x-a)^2) = 2(x-a)$

$dAx_1/da_1=2$ $dAx_2/da_1=1$

$$\mathbf{y} = \mathbf{A}\mathbf{x} = \begin{pmatrix} 2a_1 + 4a_2 & a_1 - 2a_2 & -a_1 \end{pmatrix}.$$

$dAx_2/da_2=-2$ $dAx_3/da_2=0$

$$(\max(0, \mathbf{A}x_1) - y_1) \cdot \frac{\partial}{\partial a_1} \max(0, \mathbf{A}x_1) = (2a_1 + 4a_2 - 15) \cdot 2 = -2$$

Both Ax_2 and Ax_3 have a value lower than 0, so the value $\max(0,Ax_j)$ is zero if j=2 or j=3

Only the component j=1 is not zero, for both a_1 and a_2

Exercise 47)

$\mathbf{v}^0 = 0$

$\mathbf{v}^1 = \beta_2\mathbf{v}^0 + (1-\beta_2)\left(\nabla_\mathbf{w}\mathcal{L}(\mathbf{w}^0) \odot \nabla_\mathbf{w}\mathcal{L}(\mathbf{w}^0)\right) = (1-\beta_2)\left(\nabla_\mathbf{w}\mathcal{L}(\mathbf{w}^0) \odot \nabla_\mathbf{w}\mathcal{L}(\mathbf{w}^0)\right)$

$\mathbf{v}^2 = \beta_2\mathbf{v}^1 + (1-\beta_2)\left(\nabla_\mathbf{w}\mathcal{L}(\mathbf{w}^1) \odot \nabla_\mathbf{w}\mathcal{L}(\mathbf{w}^1)\right)$

$= \beta_2(1-\beta_2)\left(\nabla_\mathbf{w}\mathcal{L}(\mathbf{w}^0) \odot \nabla_\mathbf{w}\mathcal{L}(\mathbf{w}^0)\right) + (1-\beta_2)\left(\nabla_\mathbf{w}\mathcal{L}(\mathbf{w}^1) \odot \nabla_\mathbf{w}\mathcal{L}(\mathbf{w}^1)\right)$

$\mathbf{v}^t = \sum_{i=1}^{t} \beta_2^{\,i-1}(1-\beta_2)\left(\nabla_\mathbf{w}\mathcal{L}(\mathbf{w}^{t-i}) \odot \nabla_\mathbf{w}\mathcal{L}(\mathbf{w}^{t-i})\right)$

$v^3=\beta_2 v^2$... we replace v^2 with the explicit form of v^2

$v^3=\beta_2($ *** $)+(1-\beta_2)(...)$

$$\mathbf{m}^{t+1} = \beta_1\mathbf{m}^t + (1-\beta_1)\nabla_\mathbf{w}\mathcal{L}(\mathbf{w}^t)$$
$$\mathbf{v}^{t+1} = \beta_2\mathbf{v}^t + (1-\beta_2)\left(\nabla_\mathbf{w}\mathcal{L}(\mathbf{w}^t) \odot \nabla_\mathbf{w}\mathcal{L}(\mathbf{w}^t)\right)$$
$$\hat{\mathbf{m}}^{t+1} = \frac{\mathbf{m}^{t+1}}{1-\beta_1^{t+1}}, \hat{\mathbf{v}}^{t+1} = \frac{\mathbf{v}^{t+1}}{1-\beta_2^{t+1}}$$
$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta\frac{\hat{\mathbf{m}}^{t+1}}{\sqrt{\hat{\mathbf{v}}_{t+1}}+\epsilon}$$

we know that v0 = 0, we report the formula of V1 taking it from the definition in Adam's formalization, Beta2*v(0)=0 as v0 = 0. V2 = Beta*V1 + (1 minus Beta2) * (gradients of the Loss) this formula is taken directly from Adam's formalization, we now substitute the explicit form of V1 into the definition of v2. Obtaining this and then proceeding in a similar way also for V3 I obtain the explicit representation of v3 by taking the explicit representation of V2 and so on. We finally obtain the explicit form of v(t).

Exercise 46)

$$\text{Softmax}(\mathbf{x}+c) = \frac{e^{\mathbf{x}+c}}{\sum_{i=1}^{n} e^{x_i+c}} = \frac{e^{\mathbf{x}} \cdot e^{c}}{e^{c} \cdot \sum_{i=1}^{n} e^{x_i}} = \frac{e^{\mathbf{x}}}{\sum_{i=1}^{n} e^{x_i}} = \text{Softmax}(\mathbf{x})$$

Exponent Rules

For $a \neq 0, b \neq 0$

Product Rule	$a^x \times a^y = a^{x+y}$
Quotient Rule	$a^x \div a^y = a^{x-y}$

Exercise 48)

$$u_1 = x * w = \begin{pmatrix} 2.8 & -1.9 & 5.6 & 13.2 \\ 6.1 & 29.8 & 14.7 & 4.6 \\ 4.4 & -0.5 & 8.5 & 13.0 \\ -5.3 & 7.6 & 6.8 & 8.2 \end{pmatrix} \quad x = \begin{pmatrix} 2 & 7 & 6 & 4 \\ 6 & 5 & 0 & 4 \\ 0 & 3 & 8 & 4 \\ 0 & 4 & 1 & 2 \end{pmatrix} \quad w = \begin{pmatrix} 0.5 & 0.3 & 0.5 \\ 0.8 & 1.1 & -1.7 \\ -1.0 & 1.0 & 1.3 \end{pmatrix}$$

$$u_2 = ReLU(u_1) = \begin{pmatrix} 2.8 & 0.0 & 5.6 & 13.2 \\ 6.1 & 29.8 & 14.7 & 4.6 \\ 4.4 & 0.0 & 8.5 & 13.0 \\ 0.0 & 7.6 & 6.8 & 8.2 \end{pmatrix}$$

$$u_3 = max\_pool(u_2) = \begin{pmatrix} 29.8 & 14.7 \\ 7.6 & 13.0 \end{pmatrix}$$

$$\mathcal{L}_{L1} = \frac{1}{N}\Sigma_{i,j}^{N}|u_{3_{i,j}} - t_{i,j}| = 15.275 \qquad t = \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix}$$

Exercise 49)

The gradient for a max function with respect to a single x_i will be

$$\frac{\partial \max(\mathbf{x})}{\partial x_i} = \begin{cases} 1 & x_i = \max(\mathbf{x}) \\ 0 & otherwise \end{cases}$$

We can see that the gradient will only be backpropagated through maximum values; for non-maximum values, the gradient is zero.

Max-pooling applies the max operations to several patches in the input. We split the input $\mathbf{x}$ into N patches $\mathbf{z}_k$. Then, the gradient for a single patch $\mathbf{z}_k$ wrt. to an input x_i is

$$\frac{\partial \max(\mathbf{z}_k)}{\partial x_i} = \begin{cases} 1 & x_i = \max(\mathbf{z}_k) \\ 0 & otherwise \end{cases}$$

We sum up the gradients for overlapping patches

$$\frac{\partial \text{max\_pool}(\mathbf{x})}{\partial x_i} = \sum_{\Sigma_{k \in \mathcal{R}(x_i)}} \frac{\partial \max(\mathbf{z}_k)}{\partial x_i}$$

where $\mathcal{R}(x_i)$ indexes all patches for which x_i is in the receptive field. This is necessary, as a single x_i can be the maximum of several patches, e.g. for a max pool operation with a kernel size 3 and stride 1.

In a deep learning context, the receptive field (RF) is defined as the size of the region in the input that produces the feature, in this case, the image area that is passed to each kernel of the pooling layer.

Exercise 50)

Derive the gradient for a 1D convolutional layer where the input is $\mathbf{x} = [x_1, x_2, x_3, x_4]^\top$, the weights are $\mathbf{w} = [w_1, w_2]^\top$, and the output is $\mathbf{y} = \mathbf{x} * \mathbf{w}$. Derive the gradient for the filter weights $\frac{\partial \mathbf{y}}{\partial \mathbf{w}}$.

The forward pass yields

$$\mathbf{y} = \begin{pmatrix} w_1 x_1 + w_2 x_2 \\ w_1 x_2 + w_2 x_3 \\ w_1 x_3 + w_2 x_4 \end{pmatrix}$$

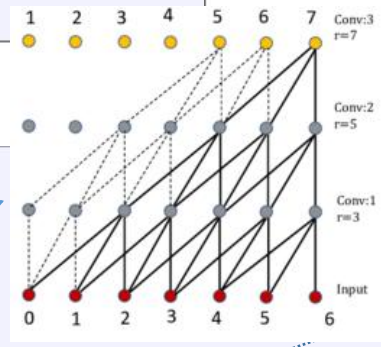
The Jacobian is

$$\frac{\partial \mathbf{y}}{\partial \mathbf{w}} = \begin{pmatrix} \frac{\partial y_1}{\partial w_1} & \frac{\partial y_1}{\partial w_2} \\ \frac{\partial y_2}{\partial w_1} & \frac{\partial y_2}{\partial w_2} \\ \frac{\partial y_3}{\partial w_1} & \frac{\partial y_3}{\partial w_2} \end{pmatrix} = \begin{pmatrix} x_1 & x_2 \\ x_2 & x_3 \\ x_3 & x_4 \end{pmatrix}$$

Exercise 51)

Examples:

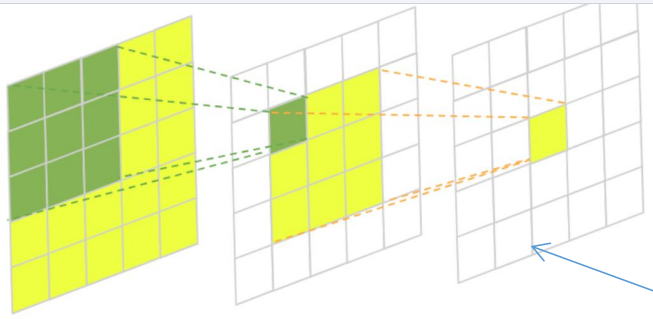
$\text{Conv}(K = 2, S = 1) : \mathcal{R}_1 = 2, \mathcal{R}_2 = 3, \mathcal{R}_3 = 4$
$\text{Conv}(K = 2, S = 2) : \mathcal{R}_1 = 2, \mathcal{R}_2 = 4, \mathcal{R}_3 = 8$
$\text{Conv}(K = 3, S = 1) : \mathcal{R}_1 = 3, \mathcal{R}_2 = 5, \mathcal{R}_3 = 7$
$\text{Conv}(K = 3, S = 2) : \mathcal{R}_1 = 3, \mathcal{R}_2 = 7, \mathcal{R}_3 = 15$



The formula to compute the receptive field R_k for a convolution/pooling layer at the depth k is

$$\mathcal{R}_k = \mathcal{R}_{k-1} + (K-1)\prod_{i=1}^{k-1} S_i$$

The same formula holds for higher-dimensional inputs (e.g., images) as we can apply the derivations independently for each dimension.



In this image, we have a two-layered fully-convolutional neural network with a 3×3 kernel in each layer. The green area marks the receptive field of one pixel in the second layer and the yellow area marks the receptive field of one pixel in the third final layer.

$$R_0 = 1$$
$$R_1 = 1 + (9-1)(1) = 9$$
$$R_2 = 9 + (2-1)(1) = 10$$
$$R_3 = 10 + (9-1)(2) = 26$$
$$R_4 = 26 + (2-1)(2) = 28$$
$$R_5 = 28 + (9-1)(2*2) = 60$$

Notice that these values are related to each dimension, so R_k=X means a square X×X in 2D

In this example, K=3 - S=1, there are another two overlapped filters (stride=1), so R_2= 3+2*1=5 (for each dimension)
This means a 5×5 area = 25 pixels

Exercise 52)

$\text{Conv}(C_{in} = 3, C_{\text{out}} = 16, K = 5, S = 2, P = 1) \equiv \text{Conv}(3, 16, 5, 2, 1)$
$\text{MaxPool}(K = 2, S = 1)$: max-pooling with a (square) kernel size of 2 with stride

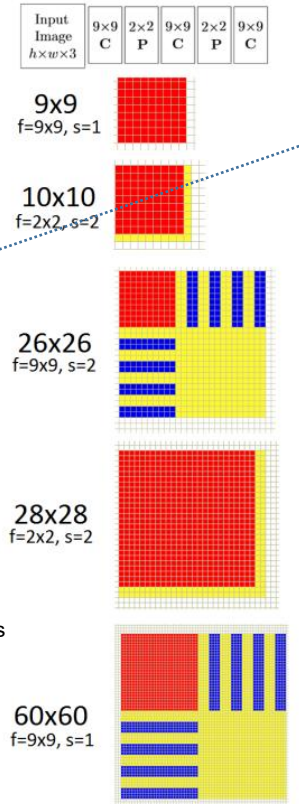
$$\mathcal{R}_k = \mathcal{R}_{k-1} + (K-1)\prod_{i=1}^{k-1} S_i$$

i) $\text{Conv}(32, 128, 3, 1, 1)$ - Relu - $\text{Conv}(128, 128, 4, 4, 1)$
ii) $(\text{Conv}(3, 3, 3, 2, 1)$ - Relu$)*6$ (The same block 6 times)
iii) $\text{Conv}(3, 64, 3, 2, 1)$ - Relu - $\text{MaxPool}(4, 3)$ - $\text{Conv}(64, 128, 3, 1, 1)$ - Relu - $\text{MaxPool}(2, 2)$

We do not index the Relu functions for brevity as they are elementwise functions; hence, they don't change the receptive field.

i) $\mathcal{R}_1 = 3, \mathcal{R}_2 = 6$
ii) $\mathcal{R}_1 = 3, \mathcal{R}_2 = 7, \mathcal{R}_3 = 15, \mathcal{R}_4 = 31, \mathcal{R}_5 = 63, \mathcal{R}_6 = 127$
iii) $\mathcal{R}_1 = 3, \mathcal{R}_2 = 9, \mathcal{R}_3 = 21, \mathcal{R}_4 = 27$

$R_1=R_0+(3-1)=1+2=3$
$R_2=R_1+(4-1)*1=3+3=6$
$R_1=R_0+(3-1)=1+2=3$
$R_5=R_4+(3-1)*(2*2*2*2)=31+32=63$
$R_2=R_1+(4-1)*2=3+3*2=9$
R4=R3+(2-1)*2 (stride of Conv)*3(Stride Max Pool)*1(stride Conv)
=21+6=27



i.e. if stride=1 there are f_k-1 overlapped filters; if the stride increases. the non overlap area, between filters, increases.

The bottom-up approach is the method to calculate the ERF of a neuron at layer k projected on the input image. Let R_k be the ERF of a neuron at layer k. Given the ERF of the previous layer R_{k-1}, where $R_0 = 1$ is the ERF at input image layer, the ERF for a neuron at current layer R_k can be computed by adding the non-overlapped-area A to R_{k-1}:

$$R_k = R_{k-1} + A \quad (1)$$

Let f_k represent the filter size of layer k. There are $(f_k - 1)$ filters overlap with each other. Since a filter can be convolved with a stride greater than one, it can significantly increase the non-overlapped area. Thus, it is necessary to account for the number of pixels each extra filter contributes to the ERF. Since the stride of the lower layer also affects the ERF of the higher layer, the pixel contributions of all layers must be accumulated. Therefore the non-overlapped area is calculated as:

$$A = (f_k - 1)\prod_{i=1}^{k-1} s_i \quad (2)$$

where s_i is the stride of the layer i. Combining equation 1 and equation 2, the ERF can be computed as:

$$R_k = R_{k-1} + (f_k - 1)\prod_{i=1}^{k-1} s_i \quad (3)$$

Example of Bottom-up approach for ERF calculation for the network shown in the top row. Red area is the ERF of the lower layer. Yellow and blue are non-overlapped areas, used to indicate how stride affects the calculation of the additional area. In this example, after the first pooling, each additional filter adds 2 pixels to the ERF. After the second pooling, each additional filter adds 4 pixels to the ERF.

Exercise 53)

$\text{Conv}(C_{in} = 3, C_{\text{out}} = 16, K = 5, S = 2, P = 1) \equiv \text{Conv}(3, 16, 5, 2, 1)$

MaxPool($K = 2, S = 1$): max-pooling with a (square) kernel size of 2 with stride

We pass an image of size $C = 3, W = 320, H = 320$ through the network below. Fill in values for the ?'s so that the architecture is valid. Then report the tensor size (C_{out}, H, W) after each layer when the image is passed through the network.

Layer
Conv(?, ?, 3, 1, 1)
ReLU
MaxPool(2, 2)
Conv(64, ?, 3, 1, 1)
ReLU
MaxPool(2, 2)
Conv(128, 256, 3, 1, 1)
ReLU
MaxPool(2, 2)
Conv(?, 512, 3, 1, 1)
ReLU
MaxPool(2, 2)

Layer	Output Shape
Conv(3, 64, 3, 1, 1)	(64, 320, 320)
ReLU	(64, 320, 320)
MaxPool(2, 2)	(64, 160, 160)
Conv(64, 128, 3, 1, 1)	(128, 160, 160)
ReLU	(128, 160, 160)
MaxPool(2, 2)	(128, 80, 80)
Conv(128, 256, 3, 1, 1)	(256, 80, 80)
ReLU	(256, 80, 80)
MaxPool(2, 2)	(256, 40, 40)
Conv(256, 512, 3, 1, 1)	(512, 40, 40)
ReLU	(512, 40, 40)
MaxPool(2, 2)	(512, 20, 20)

Exercise 54)

number of filters

Convolutional layer: $\#params = C_{out} * (K * K * C_{in} + 1)$

- $\text{Conv}(C_{in} = 3, C_{\text{out}} = 16, K = 5, S = 2, P = 1)$: a convolutional layer with 3 input channels and 16 output channels, kernel size 5, stride 2, and padding 1

i) $\text{Conv}(3, 64, 3, 1, 1) : \#params =$ 64*(3*3*3*+1)=1728

ii) $\text{Conv}(64, 128, 3, 1, 1) : \#params =$ 128*(3*3*64+1)=73856

iii) $\text{FC}(25088, 4096) : \#params = 102,764,544$ 25088*4096+4096 (bias)

fully connected layer with 25088 input and an output of 4096

The advantage of weight sharing becomes apparent.

Exercise 55)

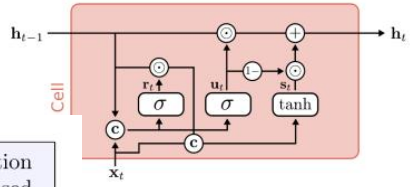
see page 273 of the lecture notes: If the gradient gets too large, we rescale it to keep it small. More precisely, if $\|g\| \geq c$, then

$g = c \cdot g/\|g\|$

$$\|\mathbf{g}\|_2 = \sqrt{0.1^2 + 0.2^2 + 0.0^2 + 1.2^2 + 10.0^2 + 0.2^2} \approx 10.076$$

$$g_{norm} = \tau \frac{\mathbf{g}}{\|\mathbf{g}\|_2} \approx \begin{bmatrix} 0.02 & 0.04 & 0.0 & 0.238 & 1.985 & 0.4 \end{bmatrix}^\top$$

Exercise 56)



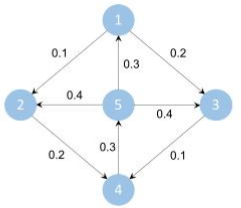
The changes are additional equations for the reset gate R_t and the more complicated equation for the target state S_t. The reset gate can choose which parts of the previous state are used to compute the current state.

$$R_t[b, c_{\text{out}}] = \sigma\left(A_{rh}[c_{\text{out}}, C_{\text{in}}] H_{t-1}[b, C_{\text{in}}] + A_{rx}[c_{\text{out}}, C_{\text{in}}] X_t[b, C_{\text{in}}] + b_r[c_{\text{out}}]\right)$$
$$R'_t[b, c_{\text{out}}] = R_t[b, c_{out}] H_{t-1}[b, c_{\text{out}}]$$
$$U_t[b, c_{\text{out}}] = \sigma\left(A_{uh}[c_{\text{out}}, C_{\text{in}}] H_{t-1}[b, C_{\text{in}}] + A_{ux}[c_{\text{out}}, C_{\text{in}}] X_t[b, C_{\text{in}}] + b_u[c_{\text{out}}]\right)$$
$$S_t[b, c_{\text{out}}] = \tanh\left(A_{sh}[c_{\text{out}}, C_{\text{in}}] R'_t[b, C_{\text{in}}] + A_{sx}[c_{\text{out}}, C_{\text{in}}] X_t[b, C_{\text{in}}] + b_s[c_{\text{out}}]\right)$$
$$H_t[b, c_{\text{out}}] = U_t[b, c_{\text{out}}] H_{t-1}[b, c_{\text{out}}] + (1 - U_t[b, c_{\text{out}}]) S_t[b, c_{\text{out}}]$$

Exercise 57)

For the graph on the left, write down the adjacency matrix and for the adjacency matrix on the right, draw the graph. In both cases, also compute the degree matrix and graph Laplacian.

In this exercise $A_{ji}=w_{ij}$ instead in the lecture notes $A_{ij}=w_{ij}$



(i)

$$\mathbf{A} = \begin{pmatrix} 0 & 0.1 & 0 & 0.3 \\ 0 & 0 & 0.5 & 0 \\ 0.2 & 0 & 0 & 0 \\ 0 & 0 & 0.4 & 0 \end{pmatrix}$$

(ii)

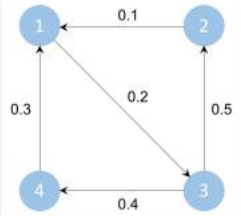
i)

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0.3 \\ 0.1 & 0 & 0 & 0 & 0.4 \\ 0.2 & 0 & 0 & 0 & 0.4 \\ 0 & 0.2 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0.3 & 0 \end{pmatrix} \quad \mathbf{D} = \begin{pmatrix} 0.3 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0.6 & 0 & 0 \\ 0 & 0 & 0 & 0.3 & 0 \\ 0 & 0 & 0 & 0 & 0.3 \end{pmatrix}$$

The **degree** d_i of node i is the sum of weights of its incident edges:

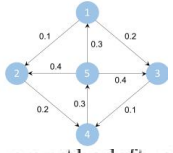
$$\mathbf{L} = \begin{pmatrix} 0.3 & 0 & 0 & 0 & -0.3 \\ -0.1 & 0.5 & 0 & 0 & -0.4 \\ -0.2 & 0 & 0.6 & 0 & -0.4 \\ 0 & -0.2 & -0.1 & 0.3 & 0 \\ 0 & 0 & 0 & -0.3 & 0.3 \end{pmatrix} \quad \mathbf{L} = \mathbf{D} - \mathbf{A}.$$

ii)



$$\mathbf{D} = \begin{pmatrix} 0.4 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 0.4 \end{pmatrix} \quad \mathbf{L} = \begin{pmatrix} 0.4 & -0.1 & 0 & -0.3 \\ 0 & 0.5 & -0.5 & 0 \\ -0.2 & 0 & 0.2 & 0 \\ 0 & 0 & -0.4 & 0.4 \end{pmatrix}$$

Exercise 58)



For the graph shown above on the left, execute the diffusion process 3 times. Consider the Laplacian as the graph shift operator and the input graph signal $\mathbf{x} = [5, 4, 3, 2, 1]^\top$.

$$\mathbf{x}_0 = [5, 4, 3, 2, 1]^\top$$
$$\mathbf{x}_1 = \mathbf{S}\mathbf{x}_0 = [1.2, 1.1, 0.4, -0.5, -0.3]^\top$$
$$\mathbf{x}_2 = \mathbf{S}\mathbf{x}_1 = [0.45, 0.55, 0.12, -0.41, 0.06]^\top$$
$$\mathbf{x}_3 = \mathbf{S}\mathbf{x}_2 = [0.117, 0.206, -0.042, -0.245, 0.141]^\top$$

$$\mathbf{L} = \begin{pmatrix} 0.3 & 0 & 0 & 0 & -0.3 \\ -0.1 & 0.5 & 0 & 0 & -0.4 \\ -0.2 & 0 & 0.6 & 0 & -0.4 \\ 0 & -0.2 & -0.1 & 0.3 & 0 \\ 0 & 0 & 0 & -0.3 & 0.3 \end{pmatrix}$$

In the first case, it is sufficient to multiply S by x_0, where x_0 is the initial input and S the Laplacian, obtaining x_1. now we multiply the Laplacian by x_1, getting x_2. after the diffusion process, the initial signal x_0 is transformed into the signal x_3

Exercise 59)

h_0=0.2,h_1=0.3,h_2=0.5

For the same graph, compute the output of the graph convolution with the filter $\mathbf{h} = \{0.2, 0.3, 0.5\}$ for the graph signal $\mathbf{x} = [1, 2, 3, 4, 5]^\top$. Consider the Laplacian as the shift operator.

$$\mathbf{y} = h_0\mathbf{S}^0\mathbf{x} + h_1\mathbf{S}^1\mathbf{x} + h_2\mathbf{S}^2\mathbf{x} = [-0.385, -0.205, 0.42, 1.155, 1.06]^\top \qquad \mathbf{y} = \mathbf{H}(\mathbf{S})\,\mathbf{x} = \sum_{k=0}^{\infty} h_k\,\mathbf{S}^k\,\mathbf{x}.$$

h(0)*(S^0)*x+h(1)*S*x+h(2)*(S^2)*x

$$\mathbf{A}^0 = \mathbf{I},$$
$$\mathbf{A}^1 = \mathbf{A},$$
$$\mathbf{A}^k = \underbrace{\mathbf{A}\mathbf{A}\cdots\mathbf{A}}_{k \text{ times}}.$$

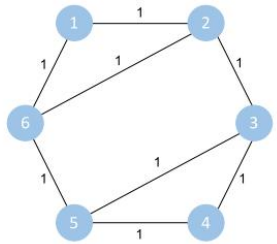
```
>> S^0

ans =

     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
```

Exercise 60)

Consider the graph shown below. Compute the output of the graph convolution for the filter $\mathbf{h} = \{1, 1\}$ for graph signals $\mathbf{x}_1 = [1, 2, 0, 0, 0, 3]^\top$ and $\mathbf{x}_2 = [0, 0, 3, 1, 2, 0]^\top$. Which property of graph convolutions is illustrated based on the nature of the two graph signals? Assume the adjacency matrix as the shift operator.



$$\mathbf{S} = \mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\mathbf{y}_1 = h_0\mathbf{S}^0\mathbf{x}_1 + h_1\mathbf{S}^1\mathbf{x}_1 = [6, 6, 2, 0, 3, 6]^\top$$
$$\mathbf{y}_2 = h_0\mathbf{S}^0\mathbf{x}_2 + h_1\mathbf{S}^1\mathbf{x}_2 = [0, 3, 6, 6, 6, 2]^\top$$

Output is shifted in the same way as input ⇒ Equivariance

```
L =

    0.3000         0         0         0   -0.3000
   -0.1000    0.5000         0         0   -0.4000
   -0.2000         0    0.6000         0   -0.4000
         0   -0.2000   -0.1000    0.3000         0
         0         0         0   -0.3000    0.3000

>> x

x =

     5
     4
     3
     2
     1

>> L*x

ans =

    1.2000
    1.1000
    0.4000
   -0.5000
   -0.3000
```

Exercise 61)

Solution: Don't listen to them! Although this is *technically* a non-linearity, specifically a *discontinuous nonlinear step function*, the gradient is 0 everywhere but the origin. Thus it would pass almost no gradient back during backprop, which is crucial when optimizing with ADAM or any other descent-based optimizer.

$$\frac{d}{dx}\left(\begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}\right) = \begin{cases} 0 & x \neq 0 \\ \text{indeterminate} & \text{(otherwise)} \end{cases}$$

If the function f is differentiable at $x_0 \in$ Dom(f), then

$$\lim_{h \to 0^+} \frac{f(x_0 + h) - f(x_0)}{h} = \lim_{h \to 0^-} \frac{f(x_0 + h) - f(x_0)}{h} = c \in \mathbb{R}$$

Exercise 62)

Solution:

The network isn't learning anything, and the weights are randomly distributed *close to zero*. The cross-entropy loss evaluates the *logarithm* of probability for the *correct* class. The softmax final layer has 99 output nodes, and with a perfectly balanced training set, the 'correct' predictions will be randomly, *evenly* distributed across all these nodes. Thus, the *expected* loss value would be

$$L(\hat{y}, y) = -\log(\frac{1}{99})$$

random classification: we have n classes the probability of a correct classification is (1/n), each output neuron assign a probability of 1/99 on average (random output)

see page 141 of the lecture notes for an example of cross entropy

Exercise 63)

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 activation shapes in the format (H, W, C), where H, W, C are the *height, width* and *channel* dimensions, respectively. Unless specified, *assume padding 1, stride 1 where appropriate.*
Notation:

- CONVx-N denotes a convolutional layer with N filters with height and width equal to x.
- POOL-n denotes a n×n max-pooling layer with stride of n and 0 padding.
- FLATTEN flattens its inputs, identical to `torch.nn.flatten` / `tf.layers.flatten`
- FC-N denotes a fully-connected layer with N neurons

Layer	Activation Volume Dimensions	Number of parameters
Input	32 × 32 × 3	0
CONV3-8		
Leaky ReLU		
POOL-2		
BATCHNORM		
CONV3-16		
Leaky ReLU		
POOL-2		
FLATTEN		
FC-10		

we have 8 filters

bias

Layer	Activation Volume Dimensions	Number of parameters
Input	32 × 32 × 3	0
CONV3-8	32 x 32 x 8	8*(3x3x3 + 1) = 224
Leaky ReLU	32 x 32 x 8	0
POOL-2	16 x 16 x 8	0
BATCHNORM	16 x 16 x 8	2*8
CONV3-16	16 x 16 x 16	16*(3x3x8 + 1) = 1168
Leaky ReLU	16 x 16 x 16	0
POOL-2	8 x 8 x 16	0
FLATTEN	16*8*8	0
FC-10	10	(8x8x16 + 1) * 10 = 10250

see page 233 of the lecture notes

Exercise 64)

You decide to use cross entropy loss to train your network. Recall that the cross-entropy loss for a single example is defined as follows: $L_{CE}(\hat{y}, y) = -\sum_{i=1}^{n_y} y_i log(\hat{y}_i)$. where $\hat{y} = (\hat{y_1}, \hat{y_2}...\hat{y_{n_y}})^T$ represents the predicted probability distribution over the classes and $y = (y_1, y_2...y_{n_y})^T$ represents the ground truth vector, which is zero everywhere except for the correct class (eg $y = (1, 0, 0, 0)^T$ for comedy, and $y = (0, 0, 1, 0)^T$ for action).

Suppose you're given an example poster of a horror movie. If the model correctly predicts the resulting probability distribution as $\hat{y} = (0.1, 0.4, 0.3, 0.2)$, what is the value of the cross-entropy loss? You can give an answer in terms of logarithms.

to build a classifier that takes in an image of a movie poster and classifies it into one of four genres: comedy, horror, action, and romance. You have been provided with a large dataset of movie posters where each movie poster corresponds to a move with exactly one of these genres.

Solution: $-log0.4$

After some training, the model now incorrectly predicts romance with distribution ⟨0, 0.4, 0, 0.6⟩ for the same poster. What is the new value of the cross-entropy loss for this example?

Solution: $-log0.4$

Exercise 65)

Prove the following lower bound on the cross-entropy loss for an example with K correct classes:

$$L_{CE}(\hat{y}, y) \geq KlogK$$

Assume you are still using softmax activation with cross-entropy loss with ground truth vector $y \in \{0, 1\}^{n_y}$ with K nonzero components.

$$\log\left(\frac{\sum_{i=1}^{n} x_i}{n}\right) \geq \frac{\sum_{i=1}^{n} \log(x_i)}{n} \quad \text{or} \quad \frac{x_1 + x_2 + \cdots + x_n}{n} \geq \sqrt[n]{x_1 \cdot x_2 \cdots x_n}$$

notice that there is a '-'

Solution:

L should be K

Let S denote the set of classes the given example belongs to (note $|S| = L$). Then,

$$\begin{aligned}
\mathcal{L}_{CE}(\hat{y}, y) &= -\sum_{i \in S}^{n_y} \log \hat{y}_i \\
&= (-L) \sum_{i \in S}^{n_y} \frac{1}{L} \log \hat{y}_i \\
&\geq (-L) \log \left(\sum_{i \in S}^{n_y} \frac{1}{L} \hat{y}_i \right) \quad \text{(by Jensen's Inequality)} \\
&= (-L) \log \frac{1}{L} \quad \text{(softmax sums to 1)} \\
&= L \log L
\end{aligned}$$

$\log_a m^n = n \log_a m$

even without this strong assumption the proof works, remember that: e.g., the sum of logits relative to the pattern classes is 0.8 and that L=4)
-L*log((1/L)*0.8) > -L*log((1/L))

Exercise 66)

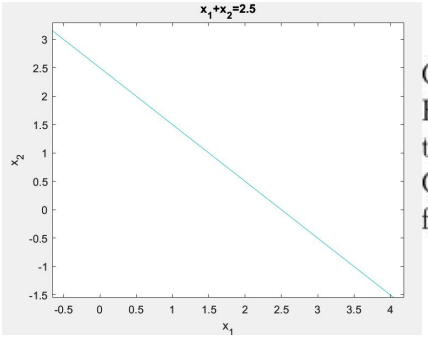
Given the following data

Item	x1	x2	Class
A	1	2	yes =1
B	2	1	yes =1
C	1	1	no =0
D	1	0	no =0

a) Are the data linearly separable? State reasons for your answer.

b) We will train a perceptron on the data. We add a bias $x_0 = -1$ to each of the data points. Suppose the current weigths to be $\boldsymbol{w} = (0, -1, 1)$. Assume a learning rate of 0.1. How should the weights be updated if point A is considered? How would the weights have been updated if the algorithm instead had considered point B?

a) The datapoints can be separated e.g. by the line $x_1 + x_2 = 2.5$



b) For A, we get $z = w_0x_0 + w_1x_1 + w_2x_2 = 0(-1) - 1(1) + 1(2) = 1$.
Since $z > 0$, we get the prediction $y = 1$.
Since $y = t$, there is no change to $\boldsymbol{w}$.

For B, we get $z = w_0x_0 + w_1x_1 + w_2x_2 = 0(-1) - 1(2) + 1(1) = -1$.
Since $z < 0$, we get the prediction $y = 0$. The update
$\boldsymbol{w} = \boldsymbol{w} - \eta(y - t)\boldsymbol{x} = (0, -1,1) - 0.1(0 - 1)(-1,2,1) = (-0.1, -0.8,1.1)$

Perceptron's Learning Rule

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = \eta(t - o)x_i$$

be careful, in the lecture notes we have seen a different formula, the signs are different but the output is the same:
[0 -1 1]+0.1*(1-0).*[-1 2 1]=[-0.1000 -0.8000 1.1000]

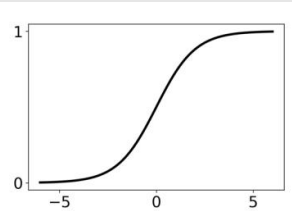
Exercise 67)

You are solving the binary classification task of classifying images as cat vs. non-cat. You design a CNN with a single output neuron. Let the output of this neuron be z. The final output of your network, $\hat{y}$ is given by:

$$\hat{y} = \sigma(ReLU(z))$$

You classify all inputs with a final value $\hat{y} \geq 0.5$ as cat images. What problem are you going to encounter?

Solution: Using ReLU then sigmoid will cause all predictions to be positive $(\sigma(ReLU(z)) \geq 0.5 \quad \forall z)$.



X = 0
Y = 1 / (1 + exp(-X))
Y = 0.5000

Exercise 68)

Given the gradient calculated at a point, the Adam optimizer has three distinct steps. First, update the moving averages. Second, apply the bias correction. Third, update the parameters.
Consider the moving average of the square of the gradients. It is given by the recursive formula:

$$s_t = \beta_2 s_{t-1} + (1 - \beta_2)g_t^2$$

Write down the expression for s_t only in terms of the gradients $g_0, g_1, \ldots, g_t$

Solution: Since $s_t = \beta_2 s_{t-1} + (1 - \beta_2)g_t^2$, this implies that
$s_{t-1} = \beta_2 s_{t-2} + (1 - \beta_2)g_{t-1}^2$ and so on.
Therefore, replacing s_{t-1} in the first equation gives us:
$s_t = \beta_2 \left(\beta_2 s_{t-2} + (1 - \beta_2)g_{t-1}^2\right) + (1 - \beta_2)g_t^2$
$s_t = \beta_2^2 s_{t-2} + (1 - \beta_2)\left(g_t^2 + \beta_2 g_{t-1}^2\right)$ Therefore, replace $s_{t-2}=\beta_2 s_{t-3}+(1-\beta_2)g_{t-2}^2$
$s_t = (1 - \beta_2)\left(g_t^2 + \beta_2 g_{t-1}^2 + \cdots + \beta_2^{t-1} g_1^2\right)$ remember that $s_0=0$
Or equivalently
$s_t = (1 - \beta_2)\sum_{i=1}^{t} \beta_2^{t-i} g_i^2$

Exercise 69)

$\cos(5x^2)$	expression	value	derivative	value
w1	x	2	1	1
w2	$w1^2$	4	2w1	2*2=4
w3	5w2	20	5	5
w4	cos(w3)	0.4081	-sin(w3)	-sin(20)
z	w4	0.4081	1	1

-sin(20)*5*4

-18.2589

First, decompose the function into elementary operations:
Let w1=x
Let w2 = x^2 = $w1^2$
Let w3 = 5*w2
Let w4 = cos(w3)
Here w4 is our final output, and we want to find the derivative of w4 with respect to x (d(w4)/dx).

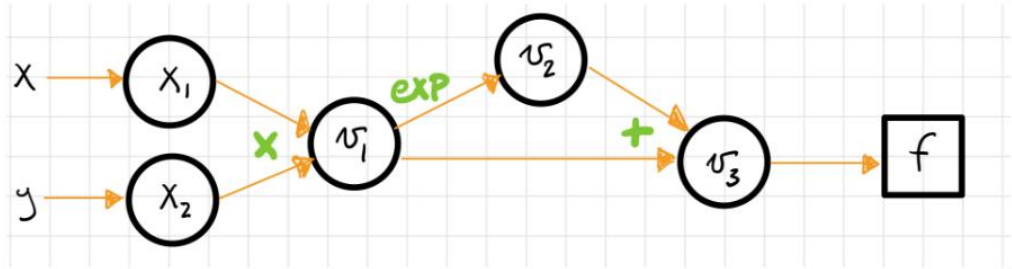
We know that:
dw1/dx = 2
dw2/dw1 = 2*w1
dw3/dw2 = 5
dw4/dw3 = -sin(w3)
Using the chain rule for each operation

Using "standard chain rule": if we have a composite function f(g(x)), then its derivative is given by f'(g(x)) * g'(x).
In our case, our outer function is cos(u), whose derivative is -sin(u), and our inner function is $5x^2$, whose derivative is 10x. Thus, using the chain rule, the derivative of our activation function f(x) = cos($5x^2$) is:
f'(x) = -sin($5x^2$) * 10x
Now, to find the derivative at the point x = 2, just plug in 2 for x:
f'(2) = -sin(5*$(2)^2$) * 10*(2) = -sin(20) * 20
Please note that the value of sin(20) would be a numerical value dependent on whether you're calculating in degrees or radians. The value of sin(20) in radians is approximately 0.9129452507276277.
So, f'(2) = -0.9129452507276277 * 20 = -18.258905014552554.

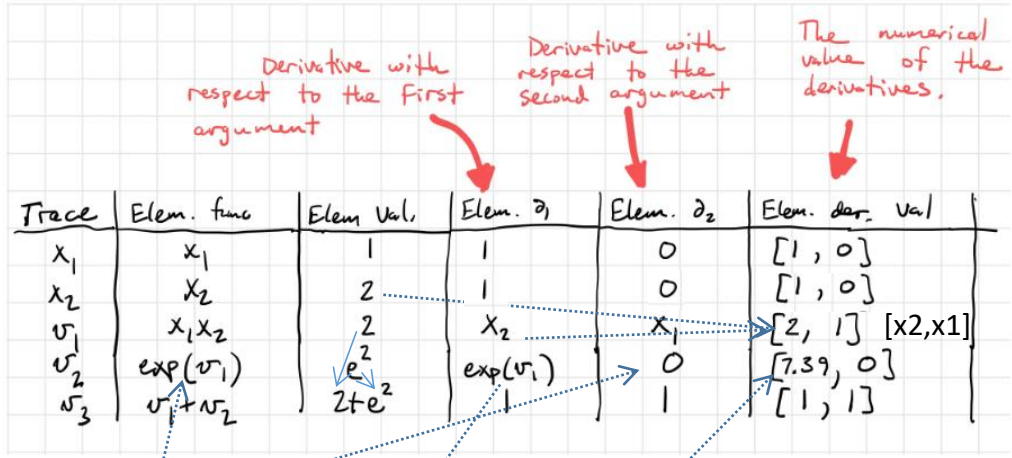
Exercise 70)

$$f(x, y) = xy + \exp(xy)$$

and its gradient at the point $(1, 2)$. We'll use reverse mode this time. Here is a picture of the computational graph,



and the corresponding table,



It has only one component (v_1) $y=e^x \rightarrow y'=e^x$ exp(2) = 7.3891

Step 1

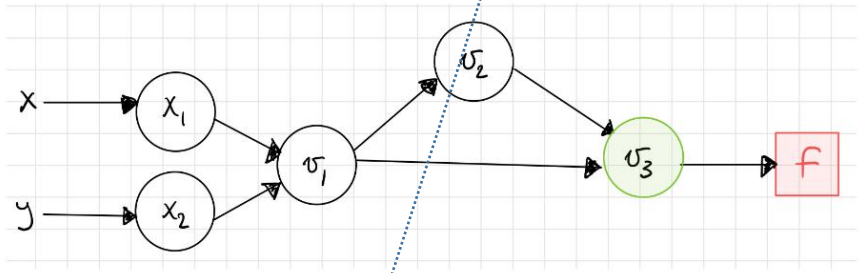
The first step is to generate the forward trace and calculate the partial derivatives of a node with respect to its children. Note that this time we must save the graph.

Step 2

Next, we start at v_3 and start calculating the chain rule. We have

$$\bar{v}_3 = \frac{\partial f}{\partial v_3} = 1.$$

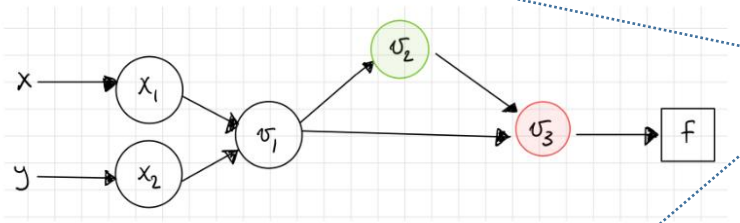
The reverse pass starts with $\bar{v}_N = \frac{\partial f}{\partial v_N} = 1$ (since f is v_N).



Step 3

$$\bar{v}_{N-1} = \frac{\partial f}{\partial v_N}\frac{\partial v_N}{\partial v_{N-1}}$$

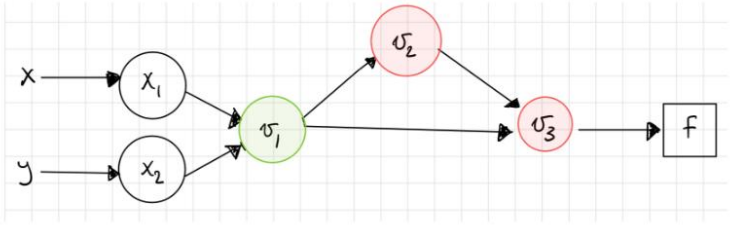
$$\bar{v}_2 = \frac{\partial f}{\partial v_3}\frac{\partial v_3}{\partial v_2} = 1 \cdot 1 = 1.$$



Step 4

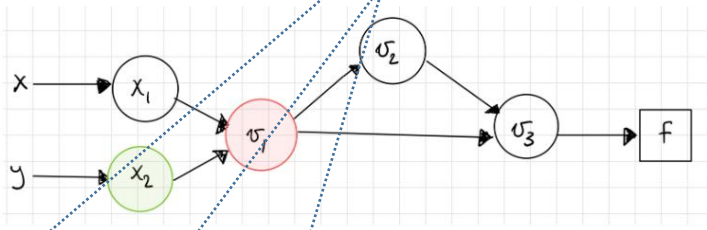
$$\bar{v}_1 = \frac{\partial f}{\partial v_3}\frac{\partial v_3}{\partial v_1} + \frac{\partial f}{\partial v_2}\frac{\partial v_2}{\partial v_1} = 1 \cdot 1 + 1 \cdot e^2 = 1 + e^2.$$

Note that we had to do a sum over the childen this time!



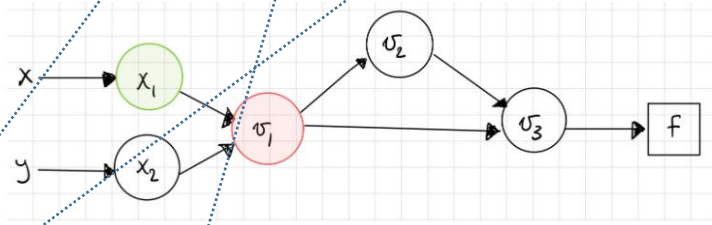
Step 5

$$\bar{x}_2 = \frac{\partial f}{\partial v_1}\frac{\partial v_1}{\partial x_2} = (1 + e^2)x_1 = 1 + e^2 = \frac{\partial f}{\partial y}.$$



Step 6

$$\bar{x}_1 = \frac{\partial f}{\partial v_1}\frac{\partial v_1}{\partial x_1} = (1 + e^2)x_2 = 2 + 2e^2 = \frac{\partial f}{\partial x}.$$



Trace	Elem. func	Elem Val	Elem. ∂_1	Elem. ∂_2	Elem. der. Val
x_1	x_1	1	1	0	[1, 0]
x_2	x_2	2	1	0	[1, 0]
v_1	$x_1 x_2$	2	x_2	x_1	[2, 1]
v_2	$\exp(v_1)$	e^2	$\exp(v_1)$	0	[7.39, 0]
v_3	$v_1 + v_2$	$2+e^2$	1	1	[1, 1]

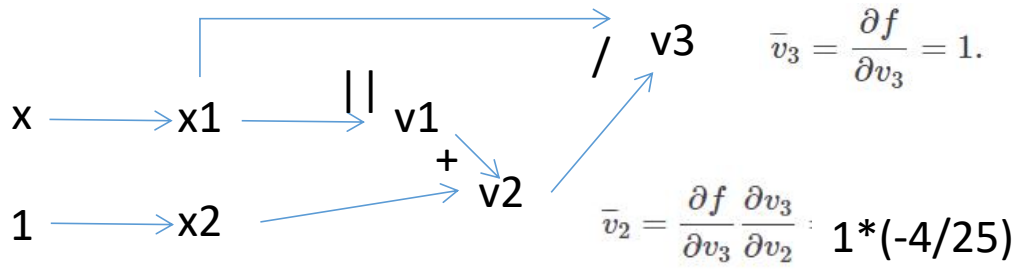
that case, we sum the two paths. For example, if v_3 has v_4 and v_5 as children, then we do:

$$\bar{v}_3 = \frac{\partial f}{\partial v_3} = \frac{\partial f}{\partial v_4}\frac{\partial v_4}{\partial v_3} + \frac{\partial f}{\partial v_5}\frac{\partial v_5}{\partial v_3}$$

Exercise 71) $\frac{d}{dx}\left(\frac{x}{1+|x|}\right)=\frac{1}{(1+|x|)^2}$ $1/(1+4)^2=1/25=0.04$

x/(1+|x|) in x=4

	Element funct.	Element Value	Derivative first argument	Derivative second argument	Derivative values
x1	x	4	1	0	1 0
x2	1	1	0	0	0 0
v1	|x|	4	|x|/x	0	1 0
v2	v1+x2	5	1	1	1 1
v3	x1/v2	4/5	1/v2	-x1/(v2^2)	1/5 -4/25



$\bar{v}_3=\frac{\partial f}{\partial v_3}=1.$

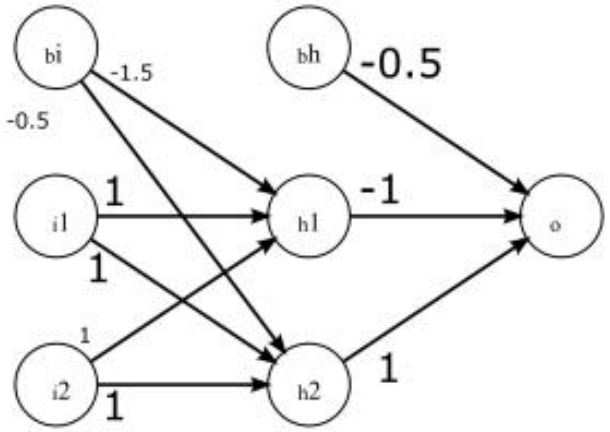
$\bar{v}_2=\frac{\partial f}{\partial v_3}\frac{\partial v_3}{\partial v_2}$ 1*(-4/25)

$\bar{v}_1=\frac{\partial f}{\partial v_2}\frac{\partial v_2}{\partial v_1}$ 1*(-4/25)*1

$\bar{x}_1=\frac{\partial f}{\partial v_1}\frac{\partial v_1}{\partial x_1}=$ 1*(-4/25)*1*1

$+\ \frac{\partial f}{\partial v_3}\frac{\partial v_3}{\partial x_1}$ + 1*(1/5)=0.04

Exercise72)

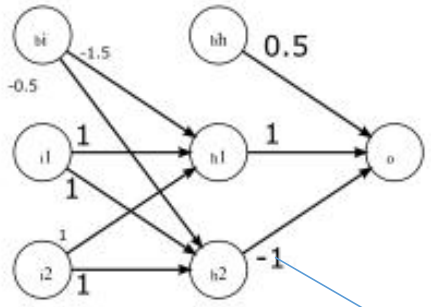


Input		Output
A	B	A XNOR B
0	0	1
0	1	0
1	0	0
1	1	1

https://en.wikipedia.org/wiki/XNOR_gate

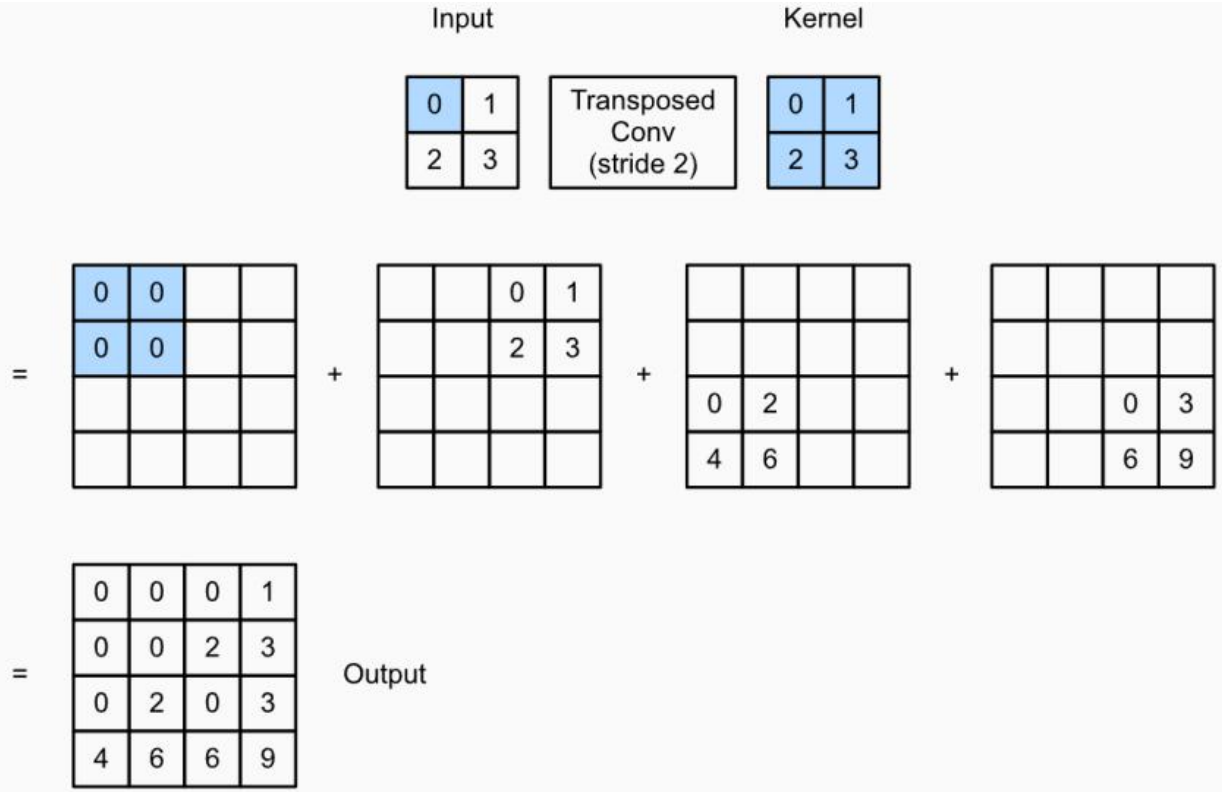
The activation function of each neuron has an output of 1 if its input is >=0.5 (otherwise output=0)
bi=bh=1;

Answer:
If we just want it to work in the binary case, we can simply switch signs on all the output weights to create the opposite function:



I1	I2	H1 input	H1 output	H2 input	H2 output	O input	O output
0	0	-1.5	0	-0.5	0	0.5	1
0	1	1-1.5=-0.5	0	1-0.5=0.5	1	-1+0.5=-0.5	0
1	0	1-1.5=-0.5	0	1-0.5=0.5	1	-1+0.5=-0.5	0
1	1	1+1-1.5=0.5	1	1+1-0.5=1.5	1	1-1+0.5=0.5	1

Exercise73)



Exercise74)

Consider an initial parameter vector $\mathbf{w}^{(0)} = [1\ 0\ 2]^T$ and a loss function of the form:

$$J(\mathbf{w}, x_i, t_i) = \frac{1}{4}(\mathbf{x_i}^T\mathbf{w} - t_i)^4$$

for a linear regression model $\hat{t}_i = \mathbf{x_i}^T\mathbf{w}$.

1. Derive the generic update formula for the parameter vector $\mathbf{w}^{(i)}$, given the datum $(\mathbf{x_i}, t_i)$, provided by the gradient descent.
2. Apply the derived formula to the datum $x_1 = [2\ 1\ 3]^T$, $t_1 = 7$ and a learning rate $\alpha = 0.5$.

$$\begin{aligned}
\mathbf{w}^{(i)} &= \mathbf{w}^{(i-1)} - \alpha \frac{\partial J(\mathbf{w}, x_i, t_i)}{\partial \mathbf{w}} \\
&= \mathbf{w}^{(i-1)} - \alpha\frac{1}{4} \cdot 4(\mathbf{x_i}^T\mathbf{w}^{(i-1)} - t_i)^3\mathbf{x_i} \\
&= \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} - 0.5 \left([2\ 1\ 3] \cdot \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} - 7 \right)^3 \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix} \\
&= \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} - 0.5 \cdot 1 \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix} \\
&= \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} - \begin{bmatrix} 1 \\ 0.5 \\ 1.5 \end{bmatrix} = \begin{bmatrix} 0 \\ -0.5 \\ 0.5 \end{bmatrix}
\end{aligned}$$

$$\frac{\partial}{\partial w}\left(0.25\,(x\,w - t)^4\right) = x\,(w\,x - t)^3$$

Factor out constants:

$$= 0.25\left(\frac{\partial}{\partial w}\left((-t + w\,x)^4\right)\right)$$

Using the chain rule, $\frac{\partial}{\partial w}\left((w\,x - t)^4\right) = \frac{\partial u^4}{\partial u}\,\frac{\partial u}{\partial w}$, where $u = w\,x - t$ and

$$\frac{\partial}{\partial u}\left(u^4\right) = 4\,u^3$$

Exercise75)

		Actual	
		Positive	Negative
	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

N=FP+TN I have both N and FP, therefore I calculate TN, then I find TP from accuracy, then we can calculate the recall

50 patterns of class '1'
and 50 of class '0'
Accuracy=0.85;
Precision=0.818;
False positives=10

Yes	No
Yes	TP = 45	FN = 5	P = 50
No	FP = 10	TN = 40	N = 50
	P' = 55	N' = 45	P+N = 100

Correct rate: (TP+TN)/(P+N) = 85/100 = 0.85

Precision (precision): TP/(TP+FP) = 45/55 = 0.818

Recall rate (recall): TP/P = 45/50 = 0.9

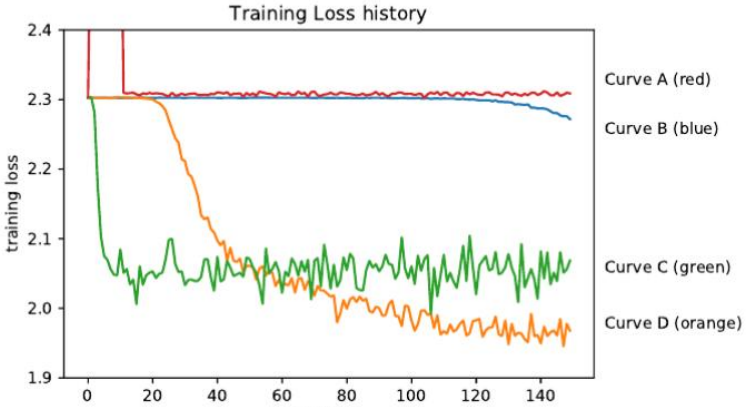
Exercise 76)

Curve A: 4e-1 = 0.4 (Learning Rate is way too high)

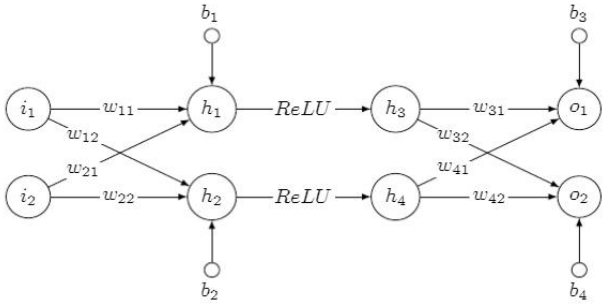
Curve B: 2e-5 = 0.00002 (Learning Rate is too low)

Curve C: 8e-3 = 0.008 (Learning Rate is too high)

Curve D: 3e-4 = 0.0003 (Good Learning Rate)



Exercise 77)



The values of variables are given in the following table:

Variable	i_1	i_2	w_{11}	w_{12}	w_{21}	w_{22}	w_{31}	w_{32}	w_{41}	w_{42}	b_1	b_2	b_3	b_4	t_1	t_2
Value	2.0	-1.0	1.0	-0.5	0.5	-1.0	0.5	-1.0	-0.5	1.0	0.5	-0.5	-1.0	0.5	1.0	0.5

Forward pass:

$$h_1 = i_1 \times w_{11} + i_2 \times w_{21} + b_1 = 2.0 \times 1.0 - 1.0 \times 0.5 + 0.5 = 2.0$$

$$h_2 = i_1 \times w_{12} + i_2 \times w_{22} + b_2 = 2.0 \times -0.5 + -1.0 \times -1.0 - 0.5 = -0.5$$

$$h_3 = max(0, h_1) = h_1 = 2$$

$$h_4 = max(0, h_2) = 0$$

$$o_1 = h_3 \times w_{31} + h_4 \times w_{41} + b_3 = 2 \times 0.5 + 0 \times -0.5 - 1.0 = 0$$

$$o_2 = h_3 \times w_{32} + h_4 \times w_{42} + b_4 = 2 \times -1.0 + 0 \times 1.0 + 0.5 = -1.5$$

$$MSE = \frac{1}{2} \times (t_1 - o_1)^2 + \frac{1}{2} \times (t_2 - o_2)^2 = 0.5 \times 1.0 + 0.5 \times 4.0 = 2.5$$

$\frac{\partial}{\partial o}\left((t-o)^2\right) = 2\,(o-t)$ ReLU activation $R = max(0, Z)$ $R'(Z) = \begin{Bmatrix} 0 & Z < 0 \\ 1 & Z > 0 \end{Bmatrix}$

Backward pass (Applying chain rule):

$$\frac{\partial MSE}{\partial w_{21}} = \frac{\partial \frac{1}{2}(t_1 - o_1)^2}{\partial o_1} \times \frac{\partial o_1}{\partial h_3} \times \frac{\partial h_3}{\partial h_1} \times \frac{\partial h_1}{\partial w_{21}} + \frac{\partial \frac{1}{2}(t_2 - o_2)^2}{\partial o_2} \times \frac{\partial o_2}{\partial h_3} \times \frac{\partial h_3}{\partial h_1} \times \frac{\partial h_1}{\partial w_{21}}$$

$$= (o_1 - t_1) \times w_{31} \times 1.0 \times i_2 + (o_2 - t_2) \times w_{32} \times 1.0 \times i_2$$

$$= (0 - 1.0) \times 0.5 \times -1.0 + (-1.5 - 0.5) \times -1.0 \times -1.0$$

$$= 0.5 + -2.0 = -1.5$$

notice that h_4=0 therefore $d(h_4)/d(h_2)$=0

Update using gradient descent:

$$w_{21}^{+} = w_{21} - lr * \frac{\partial MSE}{\partial w_{21}} = 0.5 - 0.1 * -1.5 = 0.65$$

for calculating the partials see the equations that describe h and o

Exercise 78)

$$\mu = \frac{1}{N}\sum_{i=1}^{N} X_i$$

The Jacobian $J_Y(X)$ is the matrix of partial derivatives $\frac{\partial Y_i}{\partial X_j}$. For $i = j$, the derivative is

$$Y_i = X_i - \mu \quad \text{for} \quad i = 1, \ldots, N$$

$$\frac{\partial Y_i}{\partial X_i} = \frac{\partial}{\partial X_i} X_i - \frac{1}{N}\sum_{k=1}^{N} \frac{\partial}{\partial X_i} X_k = 1 - \frac{1}{N}\frac{\partial}{\partial X_i} X_i = 1 - \frac{1}{N}$$

and for $i \neq j$, it is

$$\frac{\partial Y_i}{\partial X_j} = 0 - \frac{1}{N}\sum_{k=1}^{N} \frac{\partial}{\partial X_j} X_k = -\frac{1}{N}$$

Putting together the pieces, the Jacobian is

$$J_Y(X) = \begin{bmatrix} 1 - \frac{1}{N} & -\frac{1}{N} & \cdots & -\frac{1}{N} \\ -\frac{1}{N} & 1 - \frac{1}{N} & \ddots & -\frac{1}{N} \\ \vdots & \ddots & \ddots & \vdots \\ -\frac{1}{N} & -\frac{1}{N} & \cdots & 1 - \frac{1}{N} \end{bmatrix}$$

Exercise 79)

Suppose the first qubit is in the state $3/5\,|0\rangle + 4/5\,|1\rangle$ and the second qubit is in the state $1/\sqrt{2}\,|0\rangle - 1/\sqrt{2}\,|1\rangle$, then the joint state of the two qubits is $(3/5\,|0\rangle + 4/5\,|1\rangle)(1/\sqrt{2}\,|0\rangle - 1/\sqrt{2}\,|1\rangle) = 3/5\sqrt{2}\,|00\rangle - 3/5\sqrt{2}\,|01\rangle + 4/5\sqrt{2}\,|10\rangle - 4/5\sqrt{2}\,|11\rangle$

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

$$|\psi\rangle \otimes |\psi\rangle = \alpha^2|00\rangle + \alpha\beta|01\rangle + \alpha\beta|10\rangle + \beta^2|11\rangle = \begin{bmatrix} \alpha^2 \\ \alpha\beta \\ \alpha\beta \\ \beta^2 \end{bmatrix}$$

Suppose the first qubit is in the state $3/5\,|0\rangle + 4/5\,|1\rangle$ and the second qubit is in the state $1/\sqrt{2}\,|0\rangle - 1/\sqrt{2}\,|1\rangle$, then the joint state of the two qubits is $(3/5\,|0\rangle + 4/5\,|1\rangle)(1/\sqrt{2}\,|0\rangle - 1/\sqrt{2}\,|1\rangle) = 3/5\sqrt{2}\,|00\rangle - 3/5\sqrt{2}\,|01\rangle + 4/5\sqrt{2}\,|10\rangle - 4/5\sqrt{2}\,|11\rangle$

Can every state of two qubits be decomposed in this way? Our classical intuition would suggest that the answer is obviously affirmative. After all each of the two qubits must be in some state $\alpha\,|0\rangle + \beta\,|1\rangle$, and so the state of the two qubits must be the product. In fact, there are states such as $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ which cannot be decomposed in this way as a state of the first qubit and that of the second qubit. Can you see why? Such a state is called an entangled state. When the two qubits are entangled, we cannot determine the state of each qubit separately. The state of the qubits has as much to do with the relationship of the two qubits as it does with their individual states.

Exercise 80)

For the first question, the intuitive answer is this: take the sum of squares of all amplitudes associated with the value for which you want to find the probability of collapse. So, if you want to know the probability of the measured qbit collapsing to $|0\rangle$, you'd look at the amplitudes associated with cases $|00\rangle$ and $|01\rangle$, because those are the cases where the measured qbit is $|0\rangle$. Thus:

$$P[|0\rangle] = |\alpha_{00}|^2 + |\alpha_{01}|^2$$

Similarly, for $|1\rangle$ you look at the amplitudes associated with cases $|10\rangle$ and $|11\rangle$, so:

$$P[|1\rangle] = |\alpha_{10}|^2 + |\alpha_{11}|^2$$

As for the state of the 2-qbit system after measurement, what you do is cross out all the components of the superposition which are inconsistent with the answer you got. So, if you measured $|0\rangle$, then the state after measurement is:

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \cancel{\alpha_{10}|10\rangle} + \cancel{\alpha_{11}|11\rangle} = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle$$

However, this state is not normalized - the sum of squares does not add up to 1, and so you have to normalize it:

$$|\psi\rangle = \frac{\alpha_{00}|00\rangle + \alpha_{01}|01\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}$$

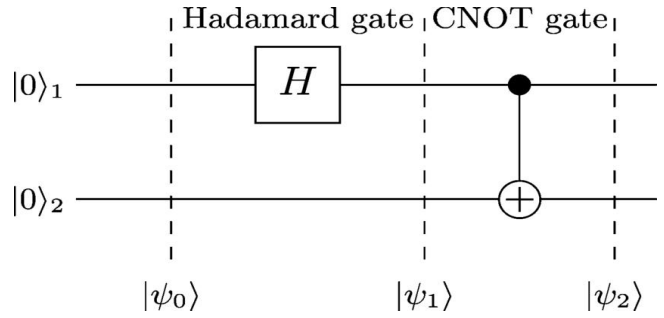
Similarly, if you measured $|1\rangle$ then you'd get:

$$|\psi\rangle = \cancel{\alpha_{00}|00\rangle} + \cancel{\alpha_{01}|01\rangle} + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle = \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

Normalized:

$$|\psi\rangle = \frac{\alpha_{10}|10\rangle + \alpha_{11}|11\rangle}{\sqrt{|\alpha_{10}|^2 + |\alpha_{11}|^2}}$$

Exercise 81)



The output of this gate is the quantum state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$. This entangled state is call an *EPR pair*, named after Einstein, Podolsky, and Rosen

$$|\psi_0\rangle = |00\rangle = |0\rangle \otimes |0\rangle$$

The Hadamard gate acts on 1-qubit and maps the basis states $|0\rangle$ and $|1\rangle$ to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ respectively.

The first qubit is passed through a Hadamard gate and then both qubits are entangled by a CNOT gate.

If the input to the system is $|0\rangle \otimes |0\rangle$, then the Hadamard gate changes the state to

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|10\rangle \ ,$$

and after the CNOT gate the state becomes $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, the Bell state $|\Phi^+\rangle$.

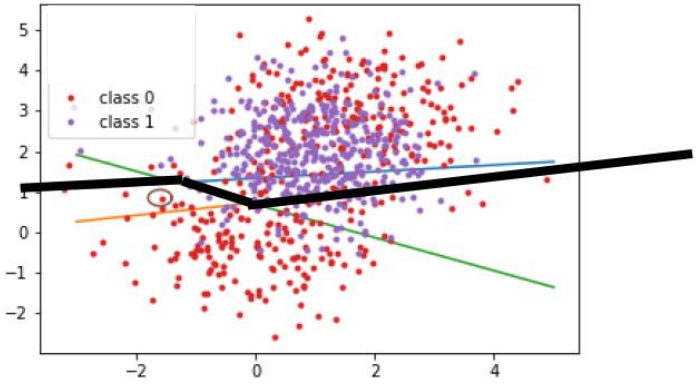
The input of CNOT gate is $|\psi_1\rangle$ 10 -> the first bit is 1 so the second bit is flipped -> 11

it applies the Hadamard gate to the first qubit.

Exercise 82)

Suggested Solution

A voting classifier consists of several different classifiers trained on the same problem and (possibly subsets of) the same data. To make a prediction, the voting classifier applies all the classifiers on the datapoint and let them each predict a class; collect the predictions and choose the majority class.



Exercise 83)

With padding 0 and a stride of 1, the output will be 194 x 194 x 1. There are 194 values because a 7x7 stride, when we view strides as defined by their top left corners, they extend all the way across the 200 dimensions until hitting the 194th dimension, upon which time they can't extend further. There are 1 depth channels as there are 1 filter of depth 128.

Exercise 84)

Layer	Activation map dimensions	Number of weights
INPUT	$128 \times 128 \times 3$	0
CONV-9-32		
POOL-2		
CONV-5-64		
POOL-2		
CONV-5-64		
POOL-2		
FC-3		

Solution: Successively:

- $120 \times 120 \times 32$ and $32 \times (9 \times 9 \times 3 + 1)$
- $60 \times 60 \times 32$ and 0
- $56 \times 56 \times 64$ and $64 \times (5 \times 5 \times 32 + 1)$
- $28 \times 28 \times 64$ and 0
- $24 \times 24 \times 64$ and $64 \times (5 \times 5 \times 64 + 1)$
- $12 \times 12 \times 64$ and 0
- 3 and $3 \times (12 \times 12 \times 64 + 1)$

Exercise 85) $f(x,y) = (x^4 + 3y^2x \ , \ 5y^2 - 2xy + 1)$

$$\frac{\partial f_1}{\partial x} = 4x^3 + 3y^2 \qquad \frac{\partial f_1}{\partial y} = 6yx \qquad \frac{\partial f_2}{\partial x} = -2y \qquad \frac{\partial f_2}{\partial y} = 10y - 2x$$

$$J_f(x,y) = \begin{pmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{pmatrix} = \begin{pmatrix} 4x^3 + 3y^2 & 6yx \\ -2y & 10y - 2x \end{pmatrix}$$

$$J_f(1,2) = \begin{pmatrix} 4 \cdot 1^3 + 3 \cdot 2^2 & 6 \cdot 2 \cdot 1 \\ -2 \cdot 2 & 10 \cdot 2 - 2 \cdot 1 \end{pmatrix}$$

$$J_f(1,2) = \begin{pmatrix} 16 & 12 \\ -4 & 18 \end{pmatrix}$$

Exercise 86)

$-H(z_2)$, where H is the Heaviside step function. $H(x) := \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$

Exercise 87)

$$L(y,\hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 = \frac{1}{n}(\hat{y}-y)^T(\hat{y}-y)$$

$$= \frac{1}{n}\left(\hat{y}^T\hat{y} - 2\hat{y}^T y + y^T y\right)$$

$$\hat{y} = Xw \Rightarrow L(y,\hat{y}) = \frac{1}{n}\left(w^T X^T X w - 2 w^T X^T y + y^T y\right)$$

Thus:

$$\frac{\partial L}{\partial w} = \frac{1}{n}\left(2X^T X w - 2X^T y + 0\right) = 0$$

$$\Rightarrow \quad 2X^T X w - 2X^T y = 0$$

$$\Rightarrow \quad X^T X w = X^T y \Rightarrow w = (X^T X)^{-1} X^T y$$

(if $|X^T X| \neq 0$)

↑ det

Exercise 88)

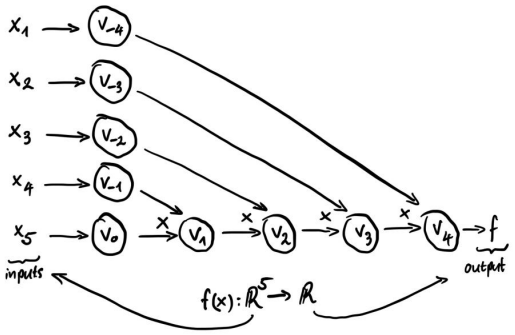
Imagine that we pass an EPR pair $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ through the quantum gate $\mathrm{H}\otimes\mathrm{H}$; that is, we apply a separate Hadamard gate to each qubit. Recall that $\mathrm{H}|0\rangle = |+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ and $\mathrm{H}|1\rangle = |-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$. Thus, we have that

$$\begin{aligned}
(\mathrm{H}\otimes\mathrm{H})\left(\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle\right) &= \frac{1}{\sqrt{2}}(|+\rangle\otimes|+\rangle) + \frac{1}{\sqrt{2}}(|-\rangle\otimes|-\rangle) \\
&= \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)\otimes\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) \\
&\quad + \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right)\otimes\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) \\
&= \frac{1}{2\sqrt{2}}|00\rangle + \frac{1}{2\sqrt{2}}|01\rangle + \frac{1}{2\sqrt{2}}|10\rangle + \frac{1}{2\sqrt{2}}|11\rangle \\
&\quad + \frac{1}{2\sqrt{2}}|00\rangle - \frac{1}{2\sqrt{2}}|01\rangle - \frac{1}{2\sqrt{2}}|10\rangle + \frac{1}{2\sqrt{2}}|11\rangle \\
&= \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle,
\end{aligned}$$

which is our original EPR pair! In general, the bookkeeping of quantum states can be rather unintuitive.

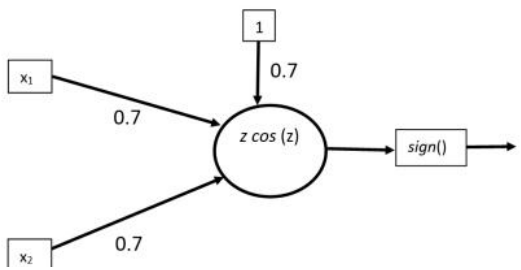
Exercise 89)

this tool can be useful https://www.symbolab.com/solver/gradient-calculator/gradient%20x%5Ccdot%20y%5Ccdot%20z%5Ccdot%20t%5Ccdot%20u%2C%20%5Cat%5Cleft(2%2C1%2C1%2C1%2C1%5Cright)?or=input



Forward primal trace	Forward tangent trace	Pass with $p^{(1)}$	Pass with $p^{(2)}$	Pass with $p^{(3)}$	Pass with $p^{(4)}$	Pass with $p^{(5)}$
$v_{-4} = x_1 = 2$	$D_p v_{-4} = p_1^{(j)}$	$D_p v_{-4} = 1$	$D_p v_{-4} = 0$	$D_p v_{-4} = 0$	$D_p v_{-4} = 0$	$D_p v_{-4} = 0$
$v_{-3} = x_2 = 1$	$D_p v_{-3} = p_2^{(j)}$	$D_p v_{-3} = 0$	$D_p v_{-3} = 1$	$D_p v_{-3} = 0$	$D_p v_{-3} = 0$	$D_p v_{-3} = 0$
$v_{-2} = x_3 = 1$	$D_p v_{-2} = p_3^{(j)}$	$D_p v_{-2} = 0$	$D_p v_{-2} = 0$	$D_p v_{-2} = 1$	$D_p v_{-2} = 0$	$D_p v_{-2} = 0$
$v_{-1} = x_4 = 1$	$D_p v_{-1} = p_4^{(j)}$	$D_p v_{-1} = 0$	$D_p v_{-1} = 0$	$D_p v_{-1} = 0$	$D_p v_{-1} = 1$	$D_p v_{-1} = 0$
$v_0 = x_5 = 1$	$D_p v_0 = p_5^{(j)}$	$D_p v_0 = 0$	$D_p v_0 = 0$	$D_p v_0 = 0$	$D_p v_0 = 0$	$D_p v_0 = 1$
$v_1 = v_{-1} v_0 = 1$	$D_p v_1 = v_0 D_p v_{-1} + v_{-1} D_p v_0$	$D_p v_1 = 0$	$D_p v_1 = 0$	$D_p v_1 = 0$	$D_p v_1 = 1$	$D_p v_1 = 1$
$v_2 = v_{-2} v_1 = 1$	$D_p v_2 = v_1 D_p v_{-2} + v_{-2} D_p v_1$	$D_p v_2 = 0$	$D_p v_2 = 0$	$D_p v_2 = 1$	$D_p v_2 = 1$	$D_p v_2 = 1$
$v_3 = v_{-3} v_2 = 1$	$D_p v_3 = v_2 D_p v_{-3} + v_{-3} D_p v_2$	$D_p v_3 = 0$	$D_p v_3 = 1$	$D_p v_3 = 1$	$D_p v_3 = 1$	$D_p v_3 = 1$
$v_4 = v_{-4} v_3 = 2$	$D_p v_4 = v_3 D_p v_{-4} + v_{-4} D_p v_3$	$D_p v_4 = 1$	$D_p v_4 = 2$	$D_p v_4 = 2$	$D_p v_4 = 2$	$D_p v_4 = 2$

Exercise 90) It solves the XOR problem, but only using input ϵ{-1,1}



$$D = \left\{ \left(\begin{bmatrix} -1 \\ -1 \end{bmatrix}, -1 \right), \left(\begin{bmatrix} 1 \\ -1 \end{bmatrix}, 1 \right), \left(\begin{bmatrix} -1 \\ 1 \end{bmatrix}, 1 \right), \left(\begin{bmatrix} 1 \\ 1 \end{bmatrix}, -1 \right) \right\}$$

Exercise 91)

By assumption, $z_1 = \mathbf{w}^T \mathrm{x}^1 + b > 0$ AND $z_2 = \mathbf{w}^T \mathrm{x}^2 + b > 0$. Also $d_1 < d_2$
Using the formula for $d(\mathbf{x},\mathbf{H})$:

$$\frac{|\mathbf{w}^T \mathbf{x}^1 + b|}{\|w\|} < \frac{|\mathbf{w}^T \mathbf{x}^2 + b|}{\|w\|}$$

$$\frac{\mathbf{w}^T \mathbf{x}^1 + b}{\|w\|} < \frac{\mathbf{w}^T \mathbf{x}^2 + b}{\|w\|}$$

$$\mathbf{w}^T \mathbf{x}^1 + b < \mathbf{w}^T \mathbf{x}^2 + b$$

$$z_1 < z_2$$

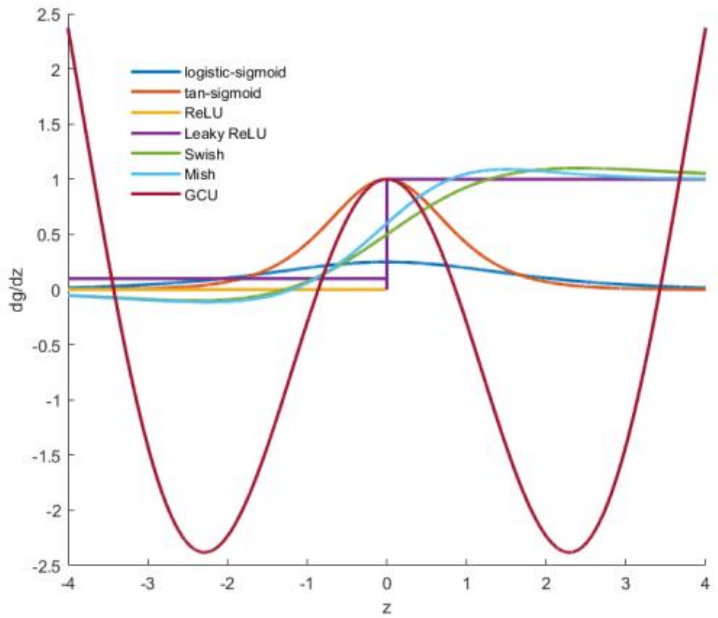
$0 < z_1$ by assumption, thus $0 < z_1 < z_2$.
Since g is strictly increasing and $g(0) = 0 : 0 < g(z_1) < g(z_2)$

$$g(z_2) > 0$$

$$Class(\mathbf{x}^2) = sign(\mathbf{w}^T \mathbf{x}^2 + b) = sign(g(z_2)) = 1$$

Thus $x_2 \in H_+$ and hence x_2 belongs to the same class as x_1.

Exercise 92)



Plot of first derivative of different activation functions

Exercise 93)

$\delta_3 = \mathrm{t} - \mathrm{out}_3 = 1.79 \quad \frac{\partial J}{\partial \mathrm{w_i}} = -(\delta_3 \cdot \mathrm{out}_i)$ ×2

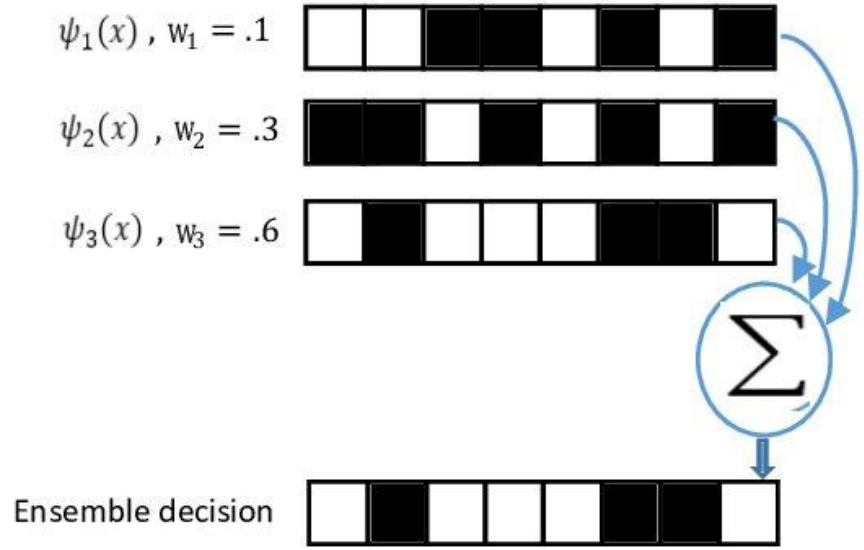
$\frac{\partial J}{\partial \mathrm{w}_1} = -(1.79 \cdot 0.3)$ ×2=-1.07

$\frac{\partial J}{\partial \mathrm{w}_2} = -(1.79 \cdot 0)$ ×2=0

$\mathrm{w}_1' = 0.7 - 0.1$ × -1.07 = 0.8070

$\mathrm{w}_2' = 0.6 - 0.1 \cdot 0 = 0.6$

Exercise 94)



Exercise 95)

$$Loss(y, z) = max(z, 0) - zy + \log(1 + e^{-|z|})$$

Exercise 96)

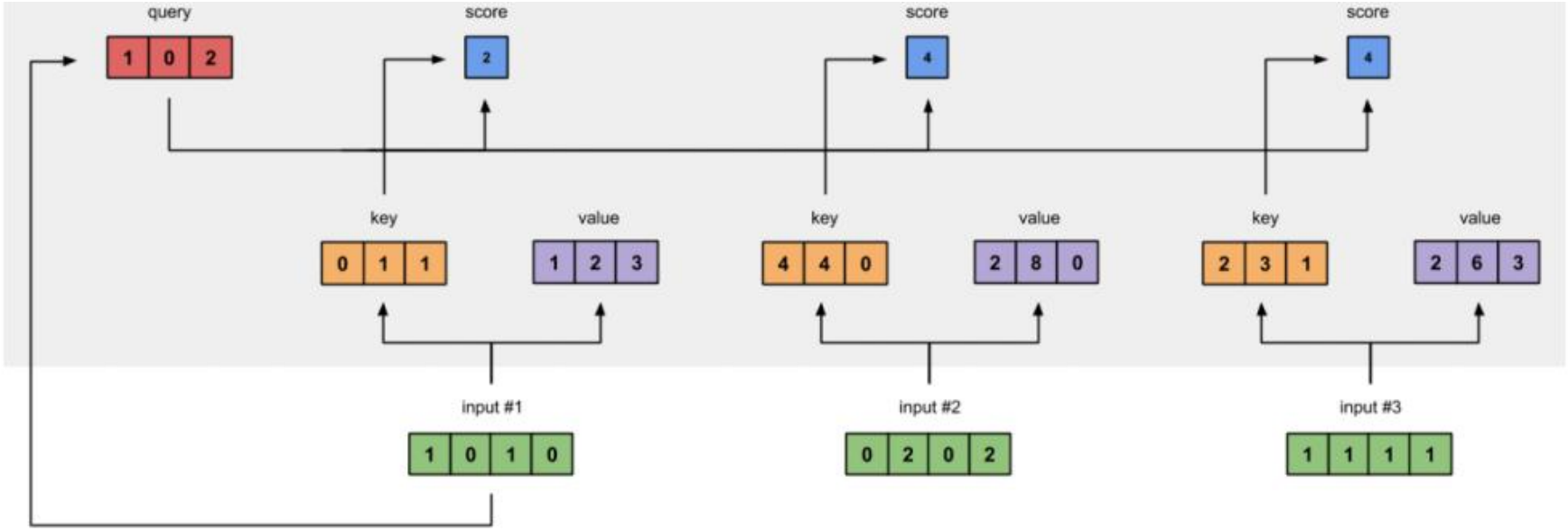
$$\frac{\partial J}{\partial w_j} = -\frac{1}{N}\sum_{i=1}^{N} \frac{\partial J}{\partial p_i}\frac{\partial p_i}{\partial z_i}\frac{\partial z_i}{\partial w_j} = -\frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i}{p_i} + \frac{1-y_i}{1-p_i}(-1)\right]\left[p_i(1-p_i)\right] x_j =$$

$$= -\frac{1}{N}\sum_{i=1}^{N}\left[y_i(1-p_i) - (1-y_i)p_i\right] x_j =$$

$$= -\frac{1}{N}\sum_{i=1}^{N}(y_i - p_i)\, x_j =$$

$$= \frac{1}{N}\sum_{i=1}^{N}(p_i - y_i)\, x_j$$

Exercise 97)



Calculating attention scores (blue) from query 1

```
                [0, 4, 2]
[1, 0, 2] x [1, 4, 3] = [2, 4, 4]
                [1, 0, 1]
```

Exercise 98)

We consider the function signature of the network step-by-step:

$$
\begin{aligned}
l_{\boldsymbol{A}} &: \mathbb{R}^{10} \to \mathbb{R}^{10} \\
\mathrm{ReLU} \circ l_{\boldsymbol{A}} &: \mathbb{R}^{10} \to (\mathbb{R}^{\geq 0})^{10} \\
l_{\boldsymbol{B}} \circ \mathrm{ReLU} \circ l_{\boldsymbol{A}} &: \mathbb{R}^{10} \to \mathbb{R}^{10} \\
\mathrm{ReLU} \circ l_{\boldsymbol{B}} \circ \mathrm{ReLU} \circ l_{\boldsymbol{A}} &: \mathbb{R}^{10} \to (\mathbb{R}^{\geq 0})^{10} \\
l_{\boldsymbol{C}} \circ \mathrm{ReLU} \circ l_{\boldsymbol{B}} \circ \mathrm{ReLU} \circ l_{\boldsymbol{A}} &: \mathbb{R}^{10} \to \mathbb{R}^{3}
\end{aligned} \tag{2}
$$

For $\boldsymbol{x} \in \mathbb{R}^n$ the signature of the softmax function is:

$$
\mathrm{softmax}(\boldsymbol{x}) : \mathbb{R}^n \to [0,1]^n \qquad \text{and} \qquad \sum_{i=1}^{n} \mathrm{softmax}(\boldsymbol{x})_i = 1 \tag{3}
$$

We generally call the inputs to the softmax function logits. So N outputs the class logits and $\mathrm{softmax} \circ N$ outputs the class probabilities.

We thus answer the questions:

1. $\mathbb{R}^{10}$ by eq. (2).
2. $\mathbb{R}^3$ by eq. (2). No. For an arbitrary $\boldsymbol{y} \in \mathbb{R}^3$ normalization is not guaranteed, i.e. there exists $\boldsymbol{y}$ with $\sum_{i=1}^{3} \boldsymbol{y}_i \neq 1$.
3. $[0,1]^3$ by eqs. (2) and (3). Due to the properties of the softmax function (see eq. (3)) any vector $\boldsymbol{y}$ output by a softmax function *can* be considered a probability distribution, and is often treated as such in calculation.

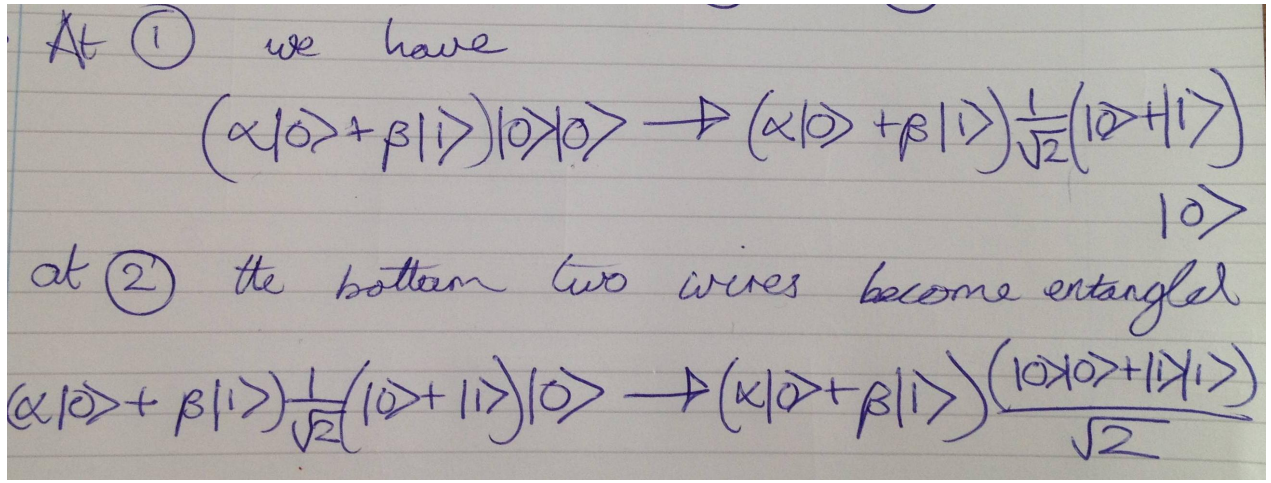
Exercise 99)

$$\frac{\partial}{\partial \boldsymbol{\eta}_i} g(\boldsymbol{\eta}) = \begin{cases} 1 & \text{if } \boldsymbol{\eta}_i > \tau \\ 0 & \text{else} \end{cases}$$

Exercise 100)

$$\begin{aligned}
&\tfrac{1}{\sqrt{2}}(|vv\rangle + |v^{\perp}v^{\perp}\rangle) \\
&= \tfrac{1}{\sqrt{2}}(\alpha^2|00\rangle + \alpha\beta|01\rangle + \alpha\beta|10\rangle + \beta^2|11\rangle) + \tfrac{1}{\sqrt{2}}(\beta^2|00\rangle - \alpha\beta|01\rangle - \alpha\beta|10\rangle + \alpha^2|11\rangle) \\
&= \tfrac{1}{\sqrt{2}}(\alpha^2+\beta^2)(|00\rangle + |11\rangle) \\
&= \tfrac{1}{\sqrt{2}}(|00\rangle + |11\rangle)
\end{aligned}$$

Exercise 101)



Exercise 102)

$$\frac{\partial}{\partial \boldsymbol{\eta}_i} h(\boldsymbol{\eta}) = \begin{cases} \text{sgn}(\eta_i) & \text{if } i \in \text{argmax}_k(|\boldsymbol{\eta}_k|) \\ 0 & \text{else} \end{cases} \quad (\eta_i \neq 0)$$

$$\frac{d}{dx}|x| = \frac{d}{dx}[x \cdot \text{sgn}(x)] = \frac{d}{dx}[x] \cdot \text{sgn}(x) + x \cdot \frac{d}{dx}[\text{sgn}(x)] = 1 \cdot \text{sgn}(x) + x \cdot 0 = \text{sgn}(x) \ (x \neq 0)$$

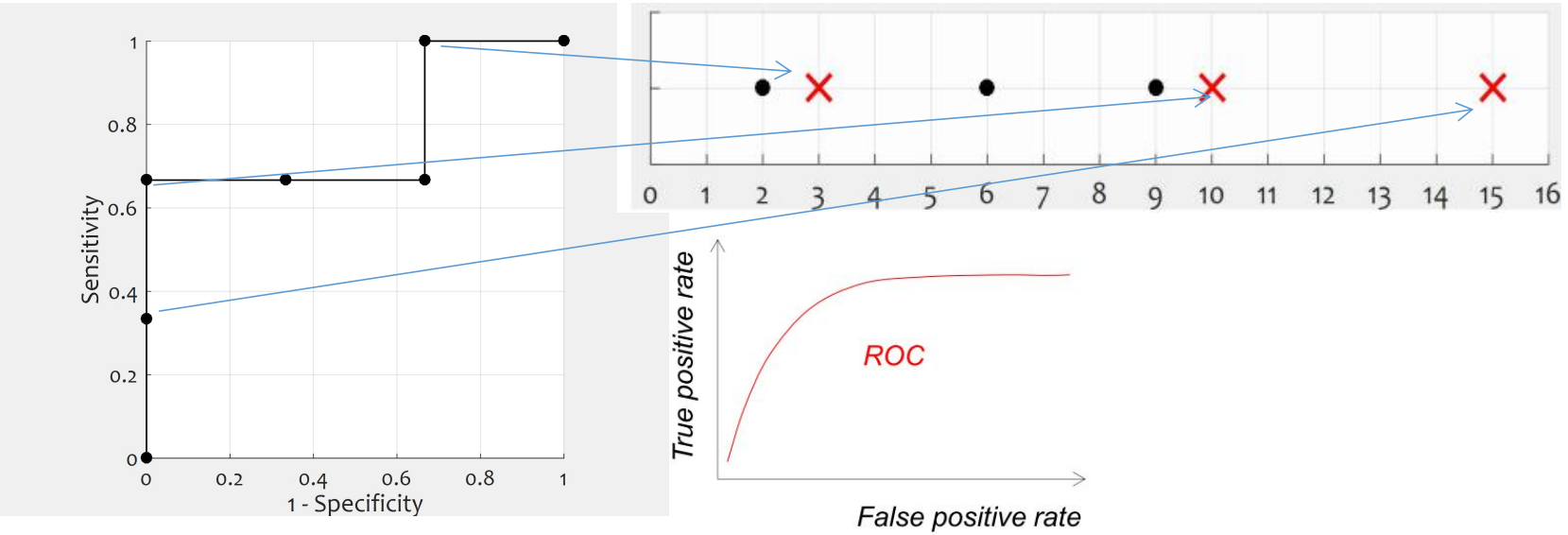
Exercise 103)

- Choose a specific interval $[a, b]$ over which you want to calculate the integral.
- Generate a dataset of input-output pairs. For each input x in the interval, calculate the true value of the integral of $f(x)$ using a reliable numerical integration method (e.g., Simpson's rule, trapezoidal rule, or a symbolic math library). These values will be your ground truth labels.
- Create a dataset containing pairs of input values x and their corresponding true integral values.

Exercise 104)

it is a dropout layer

Exercise 105)



Exercise 106)

$$\begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}\alpha\\\beta\end{pmatrix} = \begin{pmatrix}\cos(\theta/2)\\e^{i\phi}\sin(\theta/2)\end{pmatrix}$$

$$\cos(\theta/2) = 0$$

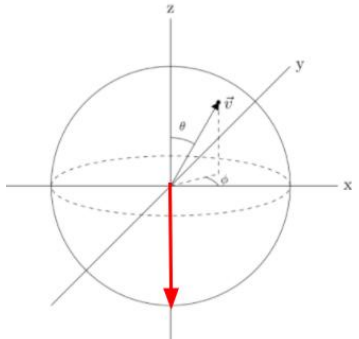
$$e^{i\phi}\sin(\theta/2) = 1$$

$$\cos(\theta/2) = 0 \rightarrow \theta = \pi$$

$$e^{i\phi}\sin(\pi/2) = 1 \rightarrow e^{i\phi}(1) = 1$$

$$i\phi = ln(1) \rightarrow \phi = 0$$

$$\begin{pmatrix}\cos(\theta/2)\\e^{i\phi}\sin(\theta/2)\end{pmatrix} = \begin{pmatrix}\cos(\pi/2)\\e^{i(0)}\sin(\pi/2)\end{pmatrix} = \begin{pmatrix}0\\1\end{pmatrix}$$



Exercise 107)

	000 100	000 010	000 001	100 000	010 000	001 000
⇧	0.2	0.3	1.0	-0.22	-0.3	0.0
⇩	-0.5	-0.4	-0.2	-0.04	-0.02	0.0
⇨	0.21	0.4	-0.3	0.5	1.0	0.0
⇦	-0.6	-0.1	-0.1	-0.31	-0.01	0.0

	000 100	000 010	000 001	100 000	010 000	001 000
⇧	0.2	0.3	1.0	-0.22	-0.3	0.0
⇩	-0.5	-0.4	-0.2	-0.04	-0.02	0.0
⇨	0.21	**0.95**	-0.3	0.5	1.0	0.0
⇦	-0.6	-0.1	-0.1	-0.31	-0.01	0.0

New Q(s,a) = r + γ * maxQ(s') = 0 + 0.95 * 1 = **0.95**

Exercise 108)

a)

transition $<s_t, a_t, r_{t+1}, s_{t+1}>$ (1,3,4,3)

	a1	a2	a3	a4
S1	0	0	0	0
S3	0	0	0	0

$Q(1, 3) = Q(1, 3) + \alpha (R3 + \gamma * Q(3, 2) - Q(1, 3))$
$Q(1, 3) = 0 + (4 + 0 - 0)$, taking $\alpha = \gamma = 1$ for simplicity
$Q(1, 3) = 4$

	a1	a2	a3	a4
S1	0	0	4	0
S3	0	0	0	0

---

transition $<s_t, a_t, r_{t+1}, s_{t+1}>$ (1,3,2,2)

	a1	a2	a3	a4
S1	0	0	4	0
S2	0	0	0	0
S3	0	0	0	0

$Q(3, 1) = Q(3, 1) + \alpha (R1 + \gamma * Q(2, 1) - Q(3, 1))$
$Q(3, 1) = 0 + (2 + 0 - 0)$, taking $\alpha = \gamma = 1$ for simplicity
$Q(3, 1) = 2$

	a1	a2	a3	a4
S1	0	0	4	0
S2	0	0	0	0
S3	2	0	0	0

---

transition $<s_t, a_t, r_{t+1}, s_{t+1}>$ (3,1,1,2)

b)

	a1	a2	a3	a4
S2	2	0	1	3
S3	2	1	6	2

$Q(3, 1) = Q(3, 1) + \alpha (R1 + \gamma * Q(2, 4) - Q(3, 1))$
$Q(3, 1) = 2 + (1 + 3 - 2)$, taking $\alpha = \gamma = 1$ for simplicity
$Q(3, 1) = 4$

	a1	a2	a3	a4
S2	2	0	1	3
S3	4	1	6	2

Exercise 109)

$$\mathbf{Q}(state, action) = \mathbf{R}(state, action) + \gamma \cdot Max\left[\mathbf{Q}(next\ state, all\ actions)\right]$$

$$\mathbf{Q}(B,F) = \mathbf{R}(B,F) + 0.8 \cdot Max\left\{\mathbf{Q}(F,B), \mathbf{Q}(F,E), \mathbf{Q}(F,F)\right\} = 100 + 0.8 \cdot 0 = 100$$

The next state is F, now become the current state. Because F is the goal state, we finish one episode. Our agent's brain now contain updated matrix **Q** as

$$\mathbf{Q} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

For the next episode, we start with initial random state. This time for instance we have state D as our initial state.

Look at the fourth row of matrix **R;** it has 3 possible actions, that is to go to state B, C and E By random selection, we select to go to state B as our action.

Now we imagine that we are in state B. Look at the second row of reward matrix **R** (i.e. state B). It has 2 possible actions to go to state D or state F. Then, we compute the **Q** value

$$\mathbf{Q}(state, action) = \mathbf{R}(state, action) + \gamma \cdot Max\left[\mathbf{Q}(next\ state, all\ actions)\right]$$

$$\mathbf{Q}(D,B) = \mathbf{R}(D,B) + 0.8 \cdot Max\left\{\mathbf{Q}(B,D), \mathbf{Q}(B,F)\right\} = 0 + 0.8 \cdot Max\left\{0, 100\right\} = 80$$

We use the updated matrix **Q** from the last episode. $\mathbf{Q}(B,D) = 0$ and $\mathbf{Q}(B,F) = 100$. The result of computation $\mathbf{Q}(D,B) = 80$ because of the reward is zero. The **Q** matrix becomes

$$\mathbf{Q} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Now, in state B the best action is F

$$\mathbf{Q}(state, action) = \mathbf{R}(state, action) + \gamma \cdot Max\left[\mathbf{Q}(next\ state, all\ actions)\right]$$

$$\mathbf{Q}(B,F) = \mathbf{R}(B,F) + 0.8 \cdot Max\left\{\mathbf{Q}(F,B), \mathbf{Q}(F,E), \mathbf{Q}(F,F)\right\}$$

$$= 100 + 0.8 \cdot Max\left\{0, 0, 0\right\} = 100$$

Because F is the goal state, we finish this episode. Our agent's brain now contain updated matrix **Q** as

$$\mathbf{Q} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$