Q1)

1. T
2. MLP, since the MLP can produce non-linear decision boundaries while the LMS has the assumption that the data is linearly separable
3. $$y_1 = w^t x_1 = [w_1, w_2]\,[1, -1] = w_1 - w_2 \quad (1)$$
   $$y_2 = w^t x_2 = [w_1, w_2]\,[0, 1] = w_2 \quad (2)$$

   $$e = \frac{1}{2}\left[(y_1 - t_1)^2 + (y_2 - t_2)^2\right]$$
   $$e = \frac{1}{2}\left[(w_1 - w_2 - 0)^2 + (w_2 - 1)^2\right]$$
   $$e = \frac{1}{2}\left[w_1^2 + w_2^2 - 2w_1w_2 + w_2^2 + 1 - 2w_2\right]$$
   $$e = \frac{1}{2}\left[w_1^2 + 2w_2^2 - 2w_1w_2 - 2w_2 + 1\right]$$

4. a

   y_1 : $sigmoid\ is\ between\ ]0,\ 1[$

   y_2: $linear\ has\ no\ limit$

   y_3: $threshold\ output\ is\ \ either\ 0\ or\ 1$

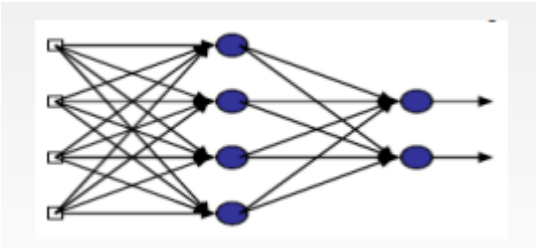
5. علينا؟

	Batch learning	On-line Learning
Weight adjustment	after a whole pass on the dataset (Epoch)	After each new sample
Cost Function	Averaged on the whole dataset	Calculated for each sample
Ability to parallelize	Easily parallelizable	Can't parallelize since not all data is present
Required storage	Large storage to contain the whole dataset	Minimal storage since samples are passed one by one

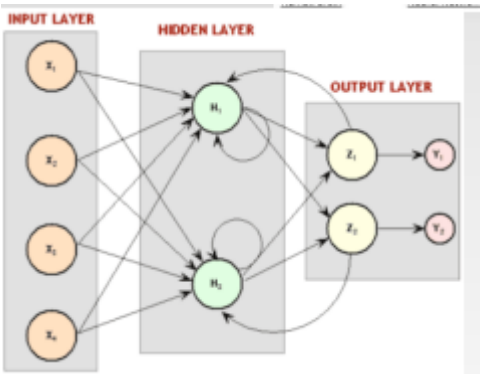
Q2)

1)

Feedforward: Each layer output is passed forward to the next layer with no feedback connections, can be used for regression or classification problems



Recurrent: The layer's output can be fed back to itself or previous layers, used where the ordering aspect is important such as in NLP or Time series tasks



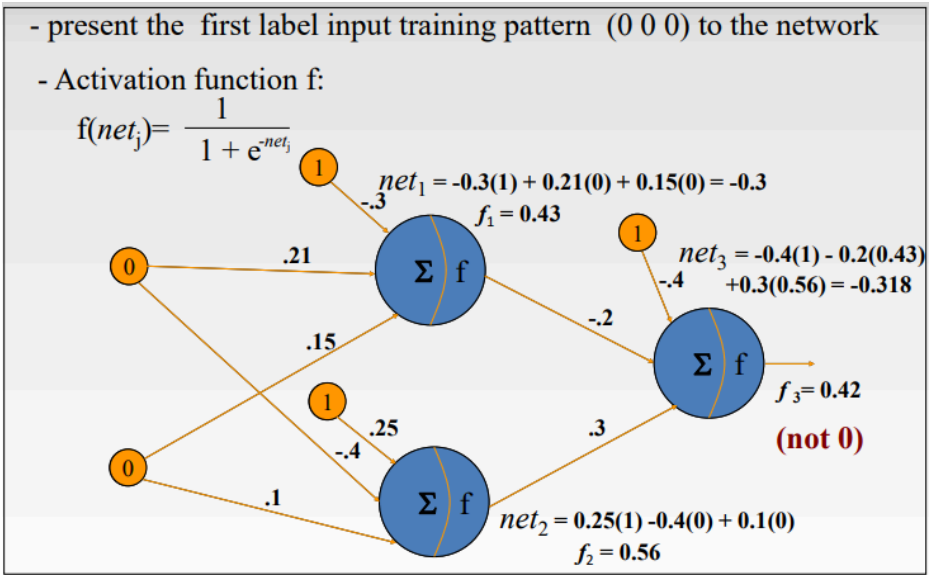
2)

Since the gradient descent will stop as it has found a minima and moving in any other direction will cause an increase in the cost function, but there might be a lower point (global minima) that can be found.
To overcome these difficulties you can try:

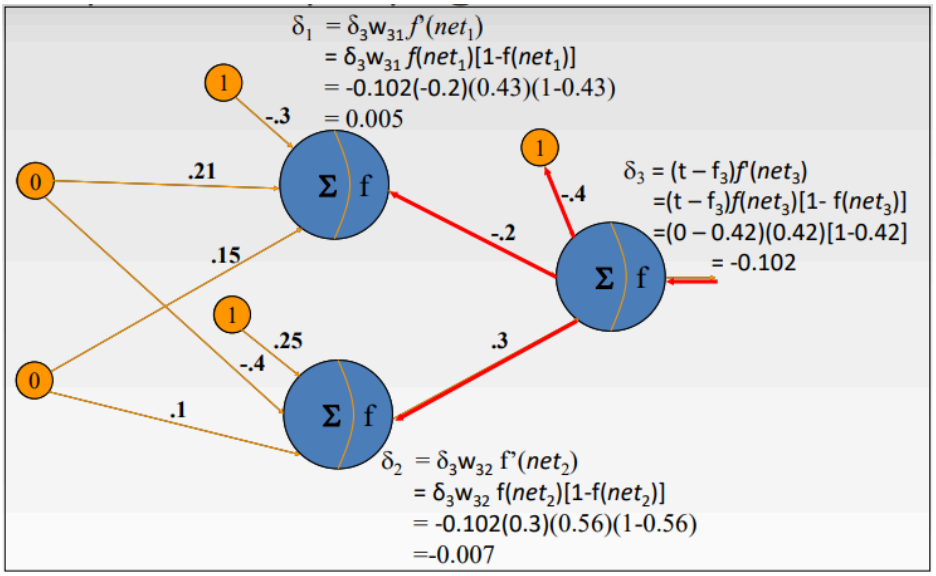
- Attempting different network architectures and initialization methods
- Train and test an ensemble of networks then use the average output of them
- Using Momentum in the GD

3)

1.



2. 3.



4.

$$w'_{1(x_0)} = w_{1(x_0)} + \eta * \delta_1 * x_0$$

$$w'_{1(x_1)} = w_{1(x_1)} + \eta * \delta_1 * x_1$$

$$w'_{1(x_2)} = w_{1(x_2)} + \eta * \delta_1 * x_2$$

$$w_{1(x_0)} = (-0.3) + 0.6 \times 0.005 \times 1 = -0.297$$

$$w_{1(x_1)} = (0.21) + 0.6 \times 0.005 \times 0 = 0.21$$

$$w_{1(x_2)} = (0.15) + 0.6 \times 0.005 \times 0 = 0.15$$

$$w'_{2(x_0)} = w_{2(x_0)} + \eta * \delta_2 * x_0$$
$$w'_{2(x_1)} = w_{2(x_1)} + \eta * \delta_2 * x_1$$
$$w'_{2(x_2)} = w_{2(x_2)} + \eta * \delta_2 * x_2$$

$w_{2(x_0)} = (0.25) + 0.6 \times (-0.007) \times 1 = 0.2458$

$w_{2(x_1)} = (-0.4) + 0.6 \times (-0.007) \times 0 = -0.4$

$w_{2(x_2)} = (0.1) + 0.6 \times (-0.007) \times 0 = 0.1$

$$w'_{30} = w_{30} + \eta * \delta_3 * f_0$$
$$w'_{31} = w_{31} + \eta * \delta_3 * f_1$$
$$w'_{32} = w_{32} + \eta * \delta_3 * f_2$$

$w_{3(0)} = (-0.4) + 0.6 \times (-0.102) \times 1 = -0.4612$

$w_{3(1)} = (-0.2) + 0.6 \times (-0.102) \times 0.43 = -0.2263$

$w_{3(2)} = (0.3) + 0.6 \times (-0.102) \times 0.56 = 0.2657$

Q3)

1. ?
2. مش علينا

3. Yes, γ and β which control the mean and variance of the normalized data, they allow the backpropagation to control the scale and shift of the output to best represent the activations

4. $YOLO\ output\ size = S \times S \times (5 * B + C)$ (lec 9, slide 7)
   where:
   • S : $Grid\ cells = 5$
   • B : $Bounding\ Boxes\ count = 2$
   • C : $Classes\ Count = 80$

   a. Output Dimensions = $5 \times 5 \times (5 \times 2 + 80) = 5 \times 5 \times 90$

   b. ?

   c. Softmax for classification part

5. Batch GD converges steadily towards the local optima however if the data is large the updating steps can take long.
   while the SGD suffers from oscillations in the descent and is more prone to noise in data since weights are updated after each sample.
   A moderate approach is the Mini-Batch which updates weights after each batch of samples

6. مش علينا

7. Image Detection?

8. a. T
   b. T
   c. F, Yolo is faster
   d. F, Fully connected layers are at the end
   e. مش علينا
   f. T

Q4)

$Input = 28 \times 28 \times 3(colored\ image)$

Architecture:

	N	K	S	P
$layer1$	4	4×4	2	"same"
$layer2$	5	4×4	1	1
$layer3$ (Pool)	-	2×2	-	-
$layer4$	6	3×3	1	0

1) $Output = \frac{Input - K + 2P}{S} + 1$

   $28 = \frac{28 - 4 + 2P}{2} + 1$

   $P = 15$

2) ?

3)

	N	K	S	P	Output $O = \frac{Input - K + 2P}{S} + 1$ $Dims: O \times O \times N$
$layer1$	4	4×4	2	"same"	$28 \times 28 \times 4$
$layer2$	5	4×4	1	1	$27 \times 27 \times 5$
$layer3$ (Pool)	-	2×2	2 (stride is the same as the kernel size)	-	$O = floor(\frac{27-2}{2} + 1) = 13$ $13 \times 13 \times 5$
$layer4$	6	3×3	1	0	$11 \times 11 \times 6$

4)

	Weights $K \times K \times Depth \times N + N\ biases$
$layer1$	$4 \times 4 \times 3 \times 4 + 4 = 196\ weight$
$layer2$	$4 \times 4 \times 4 \times 5 + 5 = 325\ weight$
$layer3$ (Pool)	0
$layer4$	$3 \times 3 \times 5 \times 6 + 6 = 276\ weighs$

5)

To produce the same output volume:

$neurons = 28 \times 28 \times 4 = 3136$

Weights: input image pixels count x output neurons count + biases

$Weights = (28 \times 28 \times 3) \times 3136 + 3136 = 7,379,008$

6)

The weights in the fully connected are larger

- Since in convolution layers the neuron will connect to 5x5x3 chunk and have 5x5x3 weights
- Convolutional layers allow the sharing of parameters between neurons

---

7)

Adding a momentum term to the update rule of the backpropagation can decrease the oscillations by using the previous gradients to update the current gradients

$$\Delta w_{ij}(t) = -\eta \frac{\delta E}{\delta w_{ij}} + \alpha\left(w_{ij}(t) - w_{ij}(t-1)\right)$$

(lec 5, slide 17)

8) ReLU
9) Fully Connected layer
10) 1
11) Sigmoid
12) Binary Cross Entropy
13) The network's parameters might not be enough to achieve good performance, this can be solved by increasing the number of layers and the number of neurons in each layer
14) مش علينا
15) The IoU