* Linear Regression:

$$y = ax + b \quad \text{OR} \quad y = \theta_0 + \theta_1 x$$

slope ↑ intercept ↑

$$\theta_1 = b = \frac{n\Sigma(xy) - \Sigma x \Sigma y}{n\Sigma x^2 - (\Sigma x)^2}, \quad \theta_0 = a = \bar{y} - bx$$

cost function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} [\hat{y}_i - y_i]^2$ ($\hat{y}_i$ ← predicted, y_i ← actual)

mean squared error

gradient descent: (α ← learning rate)

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

$$\theta_j = \theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} [(\hat{y}_i - y_i) x_i] \quad \text{(final formula)}$$

* Logistic Regression:

when we use sigmoid function of linear regression, we get logistic regression.

(Linear Reg) $y = \theta_0 + \theta_1 x$

↓

(sigmoid func) $p = \frac{1}{1 + e^{-y}}$

↓

(logistic reg) $p = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}}$

OR

$$\ln\left(\frac{p}{1-p}\right) = \theta_0 + \theta_1 x$$

Date:

Cost func:

$$J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left[-y_i \log(\hat{y}_i) + (1 - y_i) \log(\hat{y}_i) \right]$$

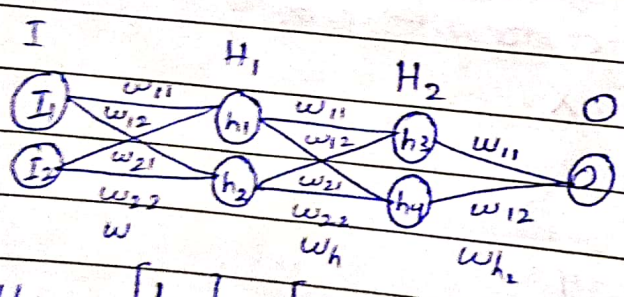
* we used mean squared error in linear regression, but we use a different error calculation method in logistic regression.

gradient descent:

$$\theta_j = \theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} \left[(\hat{y}_i - y_i) x_i \right]$$

* Neural Networks

Forward pass:



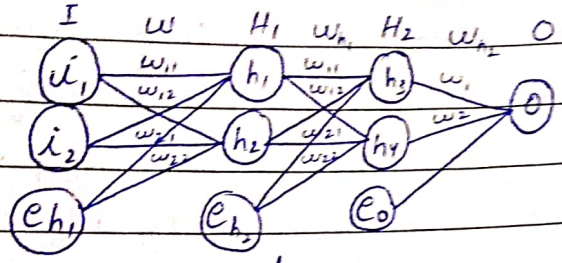
$$H_1 = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} i_1 w_{11} + i_2 w_{21} \\ i_1 w_{12} + i_2 w_{22} \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}^T \begin{bmatrix} i_1 \\ i_2 \end{bmatrix}$$

$$H_1 = \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = W^T I$$

$$H_2 = W_h^T H_1 \quad , \quad O = W_{h_2}^T H_2$$

Date:

Backward Pass:



$e_o = O - Y$ ← predicted, ← actual

$$e_{h_2} = \begin{bmatrix} e_{h_2 1} \\ e_{h_2 2} \end{bmatrix} = \begin{bmatrix} w_{h_1 1} / (w_{h_1 1} + w_{h_1 2}) \times e_o \\ w_{h_1 2} / (h_1 2 + w_{h_1 1}) \times e_o \end{bmatrix} = \underline{w_{h_1}} e_o$$

sum of diagonals

$$e_{h_1} = \begin{bmatrix} e_{h_{11}} \\ e_{h_{12}} \end{bmatrix} = \begin{bmatrix} w_{11} / (w_{11} + w_{12}) \times e_{h_{21}} + w_{12} / (w_{11} + w_{12}) \times e_{h_{22}} \\ w_{21} / (w_{21} + w_{22}) \times e_{h_{21}} + w_{22} / (w_{21} + w_{22}) \times e_{h_{22}} \end{bmatrix}$$

$e_{h_1} = \underline{w}\, e_{h_2}$

s of diag.

gradient descent: $w = w - \alpha \Delta w$

Δw = error of next layer × cost function × input layer^T

cost function = $y(1-y) \times X^T$

↑ transpose of input layer. ↑ sigmoid of output

$\Delta w_{11} = e_{h_1} \times h_1 (1 - h_1) \times i_1$

$\Delta w_{12} = e_{h_2} \times h_2 (1 - h_2) \times i_1$

$\Delta w_{21} = e_{h_1} \times h_1 (1 - h_1) \times i_2$

$\Delta w_{22} = e_{h_2} \times h_2 (1 - h_2) \times i_2$

$\Delta w = [E_{H_1} \times H_1 \times (1 - H_1)] \times I^T$

$(E_{H_1} \times H_1 \times 1 - H_1) \times I^T$

$\begin{bmatrix} e_{h_1} \\ e_{h_2} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} \begin{bmatrix} 1-h_1 \\ 1-h_2 \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \end{bmatrix}$

$[e_{h_1} \times h_1 + e_{h_1} \times h_2](1-h_1) + [e_{h_1} \times h_1 + e_{h_1} \times h_2](1-h_2)$ $e_{h_1} \times h_1 + e_{h_1} \times h_2$

$[e_{h_2} \times h_1 + e_{h_2} \times h_2](1-h_1) + [e_{h_2} \times h_1 + e_{h_2} \times h_2](1-h_2)$ $e_{h_2} \times h_1 + e_{h_2} \times h_2$

ate:_____

* Logistic Regression

$$\log(\text{odds}) = \log\left(\frac{P}{1-P}\right)$$

P ← probability (p-value) of success

$1-P$ ← prob of failure

logit function

↓

bases of logistic regression

Concept:

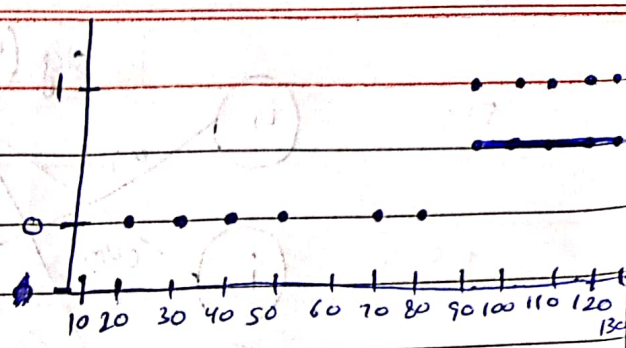
odds are simply the ratio of the probability of success to probability of failure. log(odds) makes the distribution symmetrical & easy to interpret, so we use that.

The typically used p-value is 0.05. Its basically a threshold to differ classes.

Date:

Logistic Regression

x weight	y obese
50	0
100	1
20	0
30	0
110	1
90	1
80	0
70	0
40	0
120	1
130	1



$y = bx + a$

$b = \frac{n\Sigma XY - \Sigma X \Sigma Y}{n\Sigma(X)^2 - (\Sigma X)^2} \Rightarrow \frac{11(550) - 840(5)}{11(78200) - (840)^2}$

$\Rightarrow 0.011966\,36481$

$b = \frac{1850}{154600} \Rightarrow \frac{37}{3092}$

$a = \bar{y} - b\bar{x}$

$= \frac{5}{11} - \left(\frac{37}{3092}\right)\left(\frac{840}{11}\right)$

$= 0.4545 - 0.91379 \Rightarrow -0.4592445.$

$\text{sigmoid} = \frac{1}{1+e^{-x}} \Rightarrow \frac{1}{1+e^{-(-0.4592445)}} \Rightarrow \frac{1}{1+1.58287} \Rightarrow \frac{1}{2.582} \Rightarrow 0.387$

$\text{sig}\left(y = 0.01196x - 0.4592\right)$

plug in a value to test:

$x = 150$ (clearly obese) Let pval = 0.05

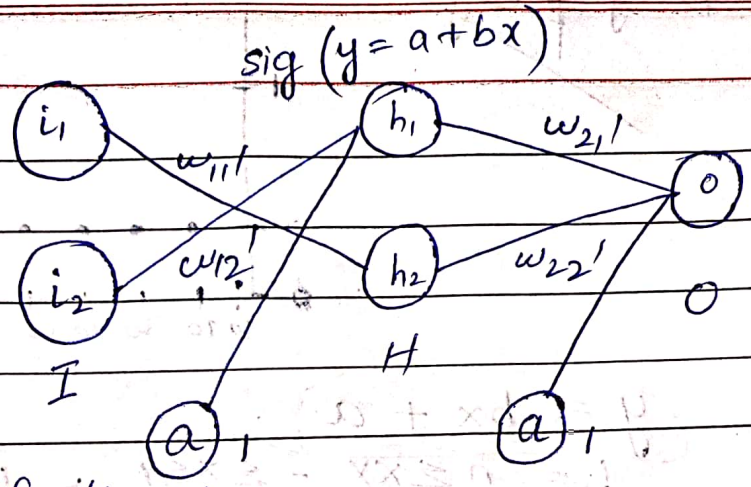
$\text{sig}(0.01196(150) - 0.4592) = 0.208$

$0.208 > 0.05$ thus $y = 1$

I_1	I_2	O
1	1	1
1	0	0
0	1	0
0	0	0

Date:

neural network



forward pass for (1, 1):

hidden layer

$h_1 = 1 + 1(1) \Rightarrow 2$

$sig(2) = \frac{1}{1+e^{-2}} \Rightarrow 0.88$

$h_2 = 0.88$

output layer

$O = a + w_1 h_1 + w_2 h_2$

$O = 1 + 1(0.88) + 1(0.88)$

$sig(O) = \frac{1}{1+e^{-(2.76)}} \Rightarrow 0.94$

forward pass for (1, 0):

hidden layer

$h_1 = 1 + 1(0) \Rightarrow 1$

$sig(1) = \frac{1}{1+e^{-1}} \Rightarrow 0.73$

$h_2 = 1 + 1(1) \Rightarrow 2$

$sig(2) = \frac{1}{1+e^{-2}} \Rightarrow 0.88$

output

$O = 1 + 1(0.73) + 1(0.88)$

$sig(O) = 0.93$

forward pass for (0, 1):

hidden

$h_1 = 0.88, \; h_2 = 0.73$

output

$O = 1 + 1(0.88) + 1(0.73)$

$sig(O) = 0.93$

forward pass for (0, 0):

hidden

$h_1 = 0.73, \; h_2 = 0.73$

output

$O = 1 + 1(0.73) + 1(0.73)$

$sig(O) = 0.92$

Date:_____

weight updation using batch gradient descent:

costs:

for a: Let $\alpha = 0.01$, $m = 4$

$$cost_a = \alpha \frac{1}{2m} \sum (\hat{y} - y)^2$$

$$= 0.01 \left(\frac{1}{2(4)} (0.94-1)^2 + (0.93-0)^2 + (0.93-0)^2 + (0.92-0)^2 \right)$$

$$= 0.01 \left(\frac{0.0036 + 0.8649 + 0.8649 + 0.8464}{8} \right)$$

$$= 0.01 (0.1896) \Rightarrow 0.00189$$

for w_{21}:

$$cost_{w_{21}} = 0.00189$$

for w_{22}:

$$cost_{w_{22}} = 0.00189.$$

new values:

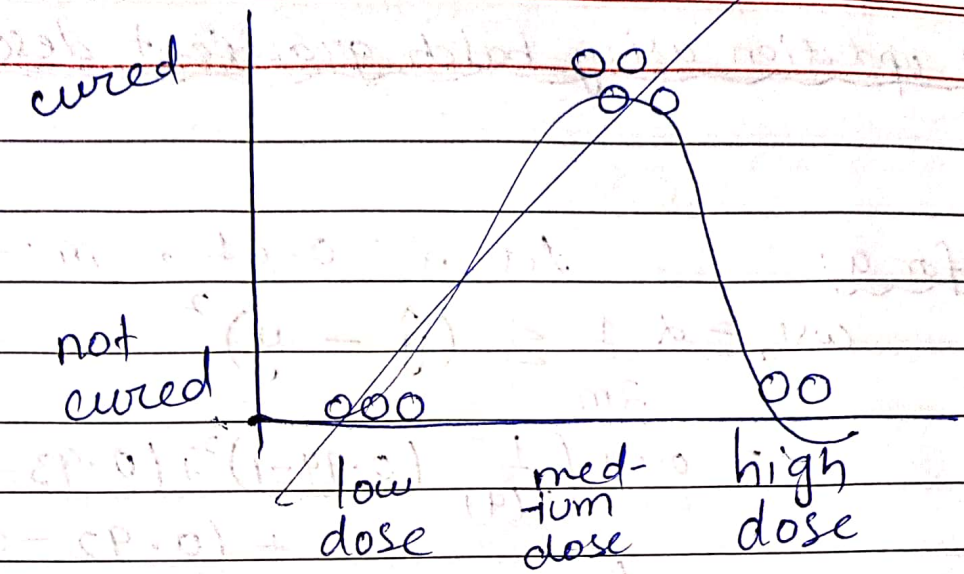
$$a = 1 - 0.00189 = 0.999$$

$$w_{21} = 1 - 0.00189 = 0.999$$

$$w_{22} = 1 - 0.00189 = 0.999$$

$$cost = -\frac{1}{m} \sum \left(y \times \log(\hat{y}) + (1-y) \times (\log(\hat{y})) \right)$$

Date:



convolutional neural network:

1. choose a filter
2. apply filter on image
3. apply suitable activation function (ReLU : convert negs to zero)
4. apply max pool filter
5. flatten the image
6. apply normal neural network:

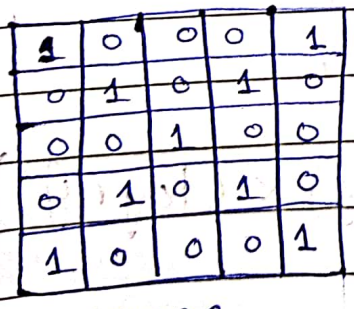
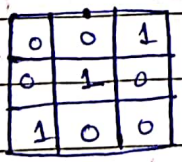
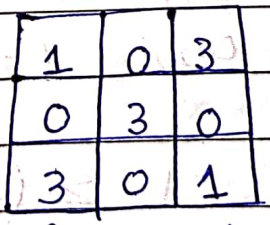


image 5×5



filter 3×3

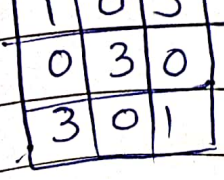
after applying →



filtered image 3×3

→ apply activation function

ReLU →



apply max pool filter 3×3

flattened → 

send into neural network

3(1) (2) bias

1 →

(1) ×

(0) 0

Date:

* Recurrent Neural Network :

① Its like a normal neural network except there's a thing called feedback loop embeded in it. This loop makes considering previous data in the output of a new data point possible for the neural network.

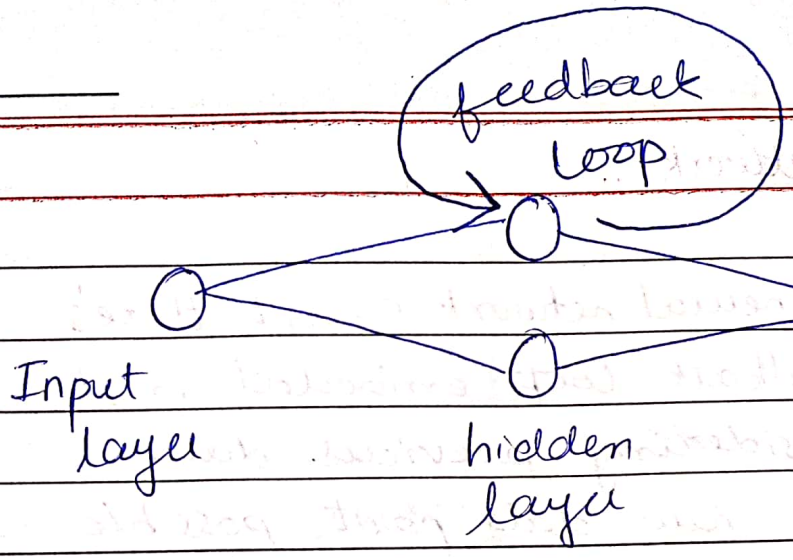
② The feedback loop picks the output of hidden layers and feeds it right back to the first hidden layer.

③ This could cause two types of problems;
- → Exploding gradient problem
- → Vanishing gradient problem

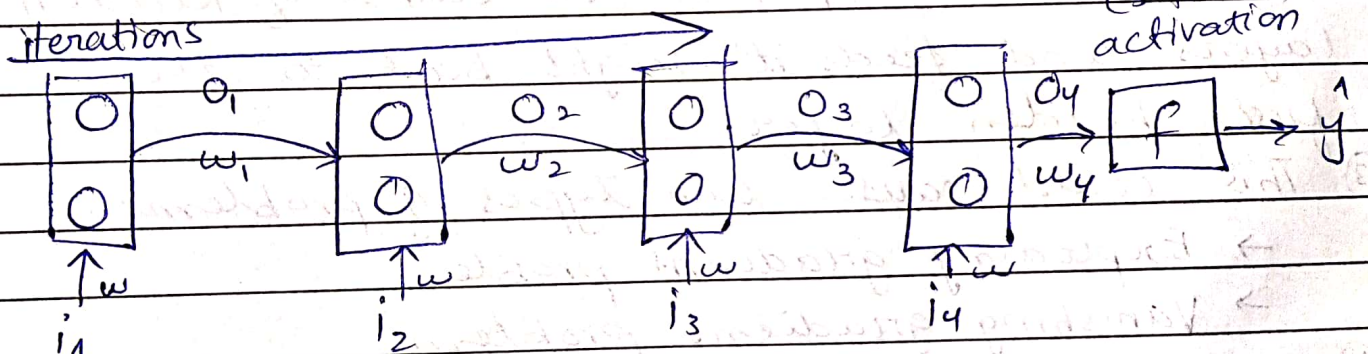
④ Exploding gradient problem occurs when the weight associated with the feedback loop grows exponentially. Its value becomes so large that its nearly impossible to converge in the gradient descent.

⑤ Vanishing gradient problem occurs when the weight associated with the feedback loop grows exponentially 'small' (near to zero). Its value is so small that we fail to reach the gradient optimal as the convergence is too slow.

Date:



* if inputs are 4, feedback loop runs 4 times.



activation function.

$$O_1 = f(i_1 w)$$

$$O_2 = f(i_2 w + O_1 w_1)$$

$$O_3 = f(i_3 w + O_2 w_2)$$

$$O_4 = f(i_4 w + O_3 w_3)$$

$$\hat{y} = \text{softmax}(O_4)$$