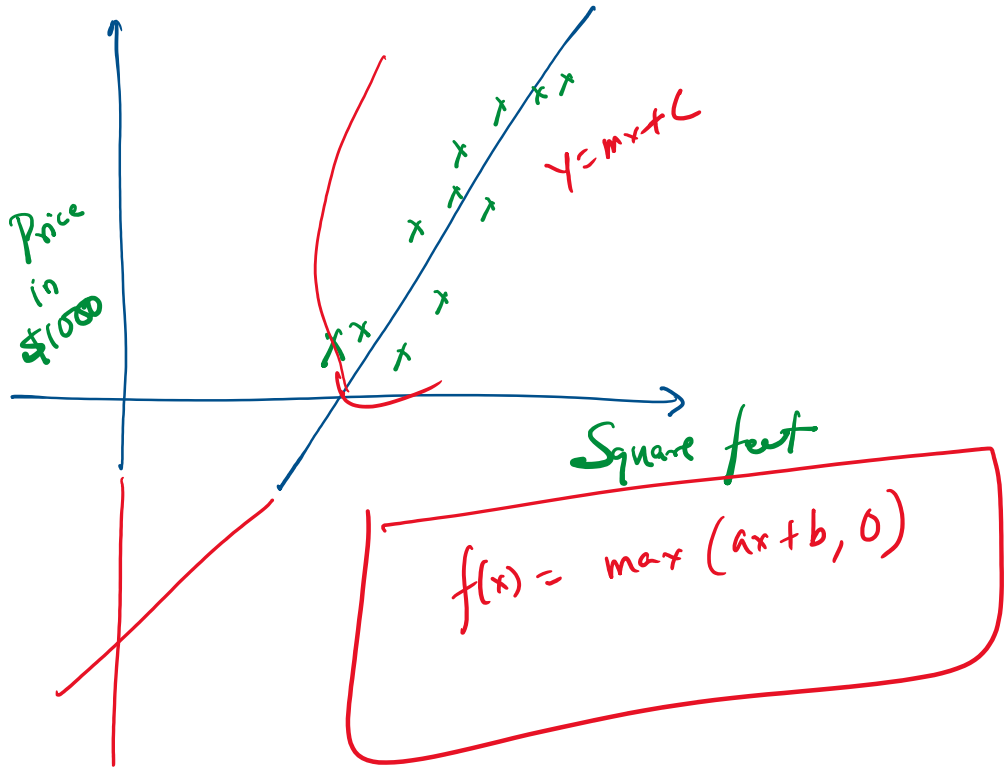
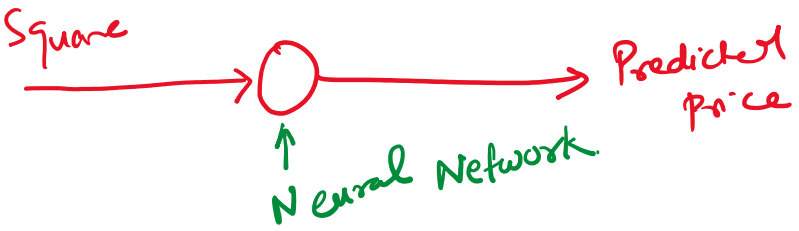
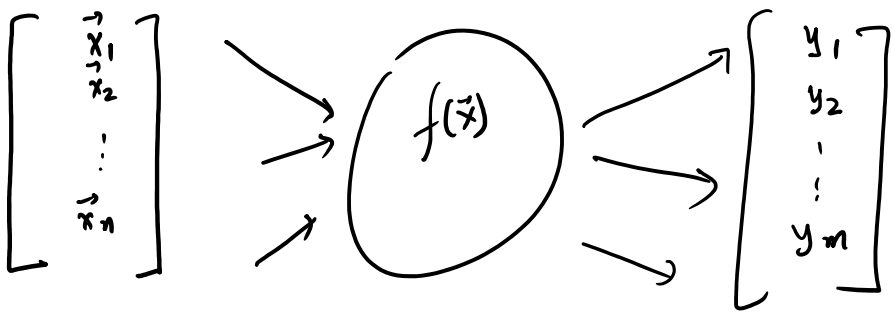
# Neural Network

Monday, February 17, 2025 6:16 PM

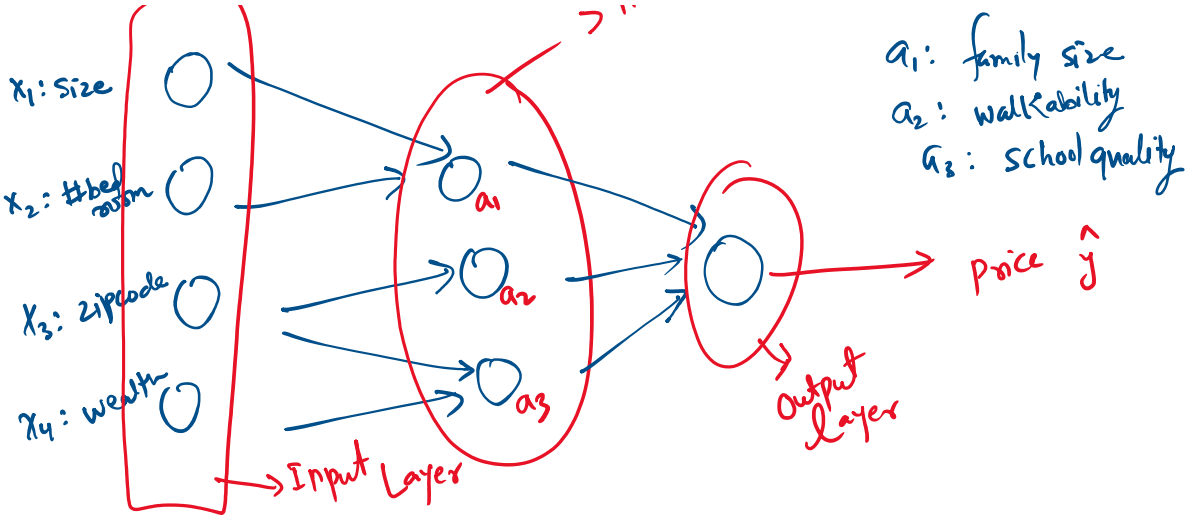



$$f(x) = \max(ax + b, 0)$$

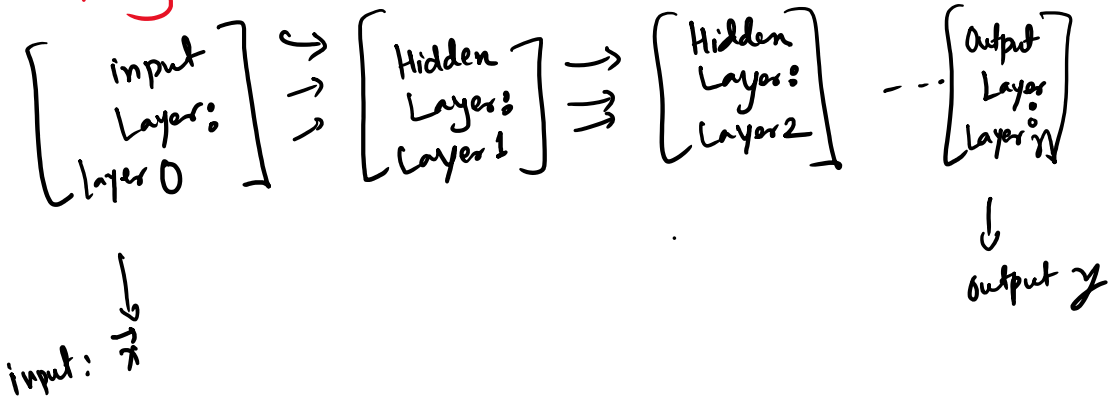



A neural network can process multiple inputs and provides multiple outputs.




## A general architecture of a neural network:




When intermediate features are also learnt then we call it end-to-end learning.

## Example to drive Motivation

Consider input features: $x_1, x_2, \ldots x_3$

Out put : $y$

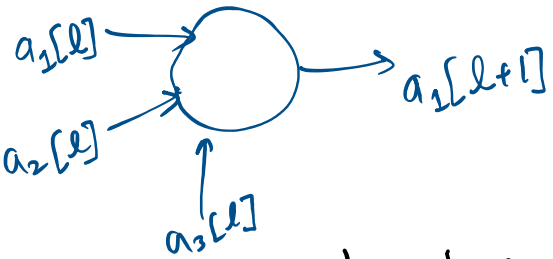
Hidden Layer contains 4 units

Output layer : 1 neurons.

A single hidden unit looks like this:

$l$ = index of layer

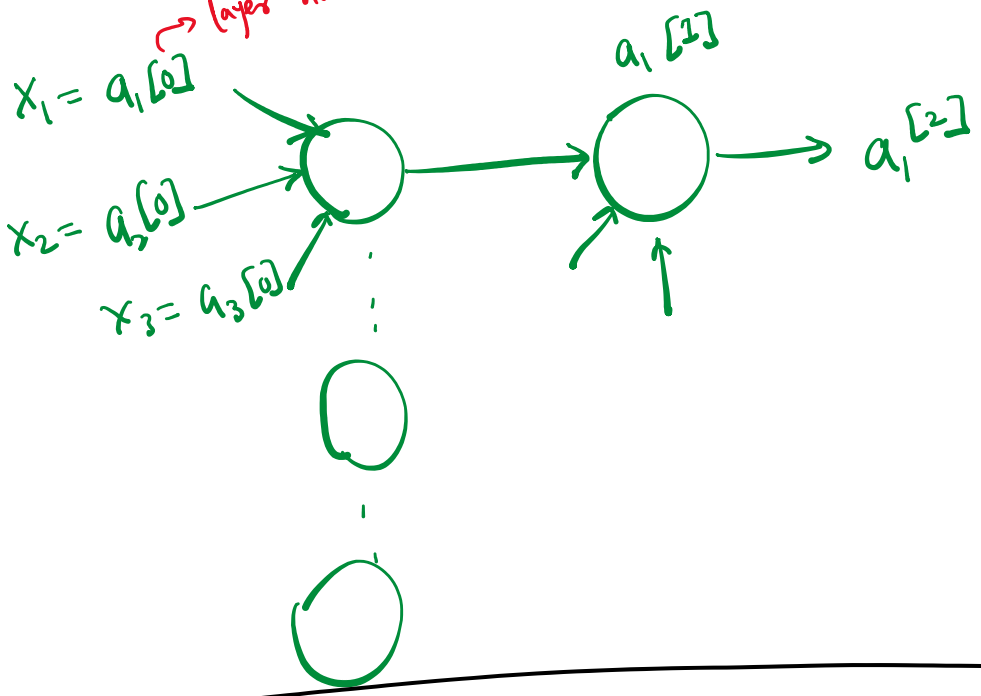
A single [illegible]




$l$ = index of layer

$a$ = activation of neuron.

This particular unit shows one hidden unit in $l^{th}$ layer.




Any $a_j^{[l]}$ is associated with the activation of the $j$th unit of layer $l$.

Let's first consider the problem of classification

$$g(\vec{x}) = \frac{1}{1 + \exp(-\vec{w}^T \vec{x})} \qquad \vec{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$y = \begin{cases} 0 & g(x) < 0.5 \\ 1 & g(x) \geq 0.5 \end{cases}$$

$g(\vec{x})$ can be broken into two different computation

$g(\vec{x})$ can be ... into two different computation

for this $\vec{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$

① $\vec{z} = \vec{w}^T \vec{x} + \vec{b}$

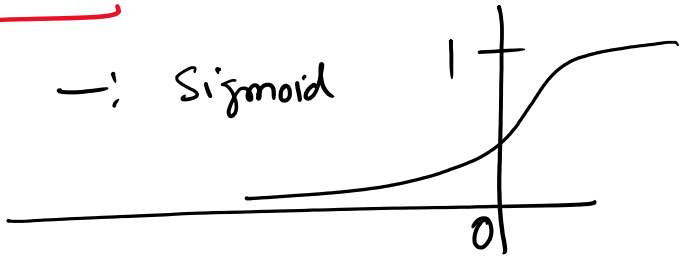
② $\vec{a} = \sigma(\vec{z})$ where $\sigma(\vec{z}) = \frac{1}{1+\exp(-\vec{z})}$

↳ activation function.

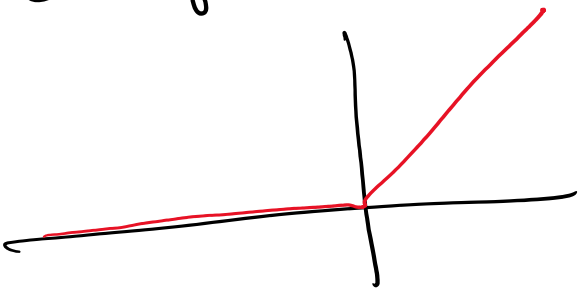
The sigmoid function shown is an example of an activation function.

## Activation function

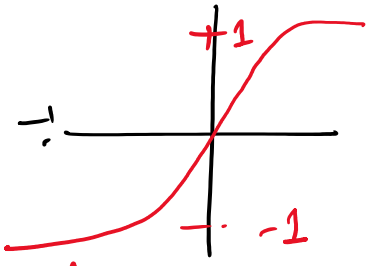
① $g(z) = \frac{1}{1+e^{-z}}$ —: Sigmoid



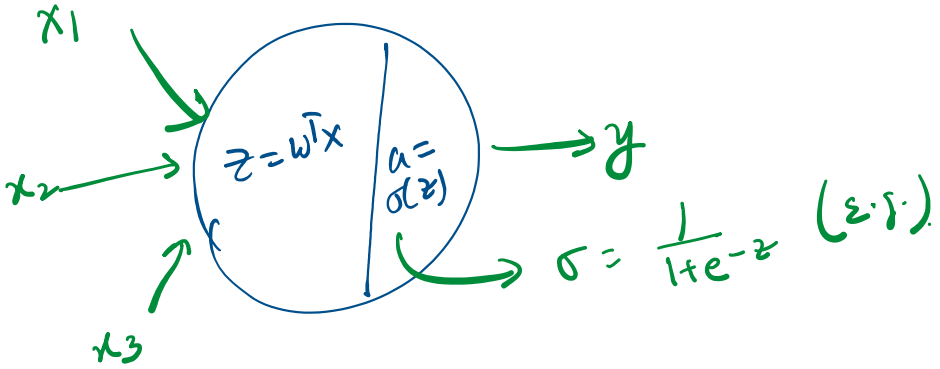

② $g(z) = \max(z, 0)$ → Rectified Linear Unit (ReLU)



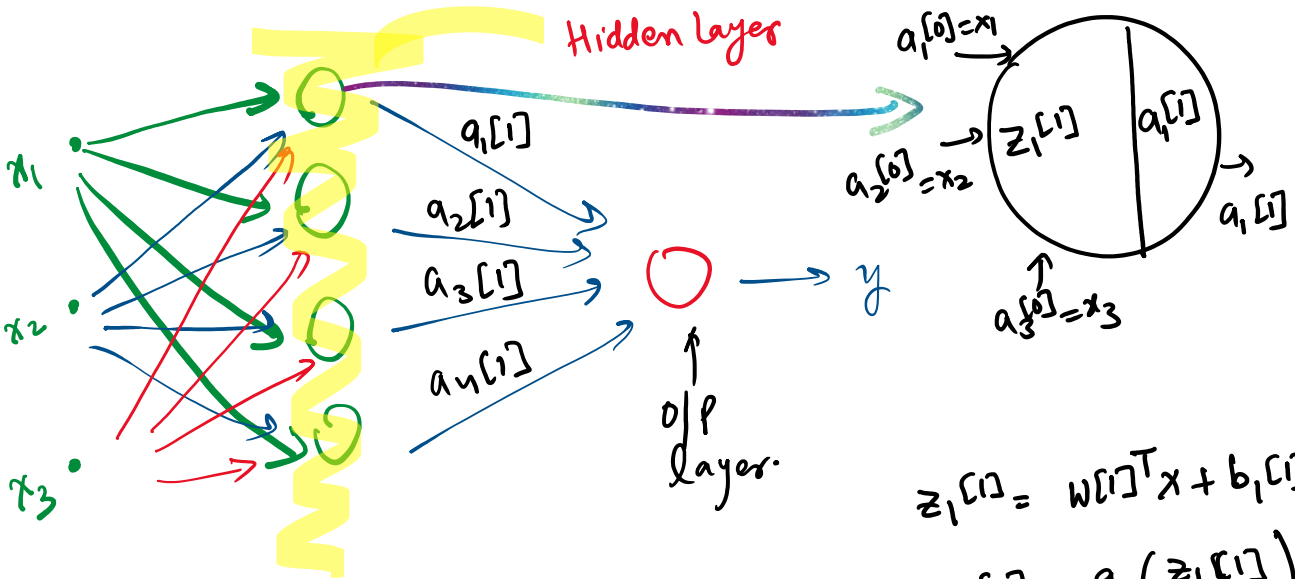
③ $g(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ —:




But in general $g(z)$ has to be nonlinear.




So, in the first hidden unit of the first layer,




$$z_1[1] = w[1]^T x + b_1[1]$$

$$a_1[1] = g(z_1[1])$$

$$= W[1]^T x + b_1[1]$$

Here $g$ is unit function.

$g$ can be some function.

where $W[1]$ is a matrix of parameter for layer [1]

So that $W_1[1]^T$ is the first row of this matrix.

$$W[1] = \begin{bmatrix} \text{---} & W_1[1]^T & \text{---} \\ \text{---} & W_2[1]^T & \text{---} \\ & \vdots & \\ \text{---} & W_4[1]^T & \text{---} \end{bmatrix}$$

and thus, $w_1[1]^T = \left[ w_{11}[1], w_{21}[1], w_3[1] \right]$

$$z_1[1] = w_1[1] x + b_1[1]$$

$$= w_{11}[1] x_1 + w_{21}[1] x_2 + w_{31}[1] x_3 + b_1[1]$$

So $w_1[1] \in \mathbb{R}^3$

$b_1[1] \in \mathbb{R}^1$ then

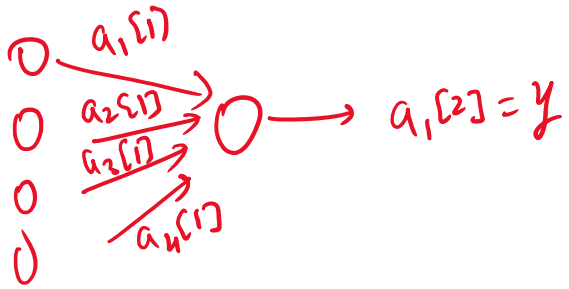
For each of the hidden units in layer 1.

$$z_1[1] = w_1[1] x + b_1[1] \qquad a_1[1] = g(z_1[1])$$

$$z_2[1] = w_2[1] x + b_2[1] \qquad \vdots$$

$$\vdots$$

$$z_4[1] = w_4[1] x + b_4[1] \qquad a_4[1] = g(z_4[1])$$




and then, at O/P

$$z_1[2] = w_1[2]^T a[1] + b_1[2]$$

and $a_1[2] = g(z_1[2])$

where $a[1] = \begin{bmatrix} a_1[1] \\ a_2[1] \\ a_3[1] \\ a_4[1] \end{bmatrix}$ ↳ Con catenation of all the first layer i.e. hidden.

↳ Can calculation of the first layer activation.

And we are free to choose $g(z)$ to be different depending on the task.

## Vectorize

$$z^{[1]} = W^{[1]T} \bar{x} + b^{[1]}$$

$$\begin{bmatrix} z_1^{[1]} \\ \vdots \\ z_4^{[1]} \end{bmatrix} = \begin{bmatrix} \text{---} & w_1^{[1]T} & \text{---} \\ \text{---} & w_2^{[1]T} & \text{---} \\ & \vdots & \\ \text{---} & & \text{---} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix}$$

$x \in \mathbb{R}^{3\times 1}$

$z^{[1]} \in \mathbb{R}^{4\times 1}$ $\quad W^{[1]} \in \mathbb{R}^{4\times 3}$ $\quad b^{[1]} \in \mathbb{R}^{4\times 1}$

What about $g(z)$ as a vector?

Although $g(z)$ is a scalar operation.

for $g(z) = \frac{1}{1+\exp(-z)}$, we can define the vector as elementwise computation which can be executed concurrently.

Then the output layer comes out as

$$\underset{1\times 1}{z^{[2]}} = \underset{1\times 4}{W^{[2]}} \underset{4\times 1}{a^{[1]}} + \underset{1\times 1}{b^{[2]}}$$

$$\underset{1\times 1}{a^{[2]}} = \underbrace{g(z^{[2]})}_{1\times 1}$$

1×1 1×1

When including more than one training sample (Consider 3 training samples)

$$z^{[1](1)} = W^{[1]}\vec{x}^{(1)} + b^{[1]}$$
$$z^{[1](2)} = W^{[1]}\vec{x}^{(2)} + b^{[1]}$$
$$z^{[1](3)} = W^{[1]}\vec{x}^{(3)} + b^{[1]}$$

↳ $\vec{x}$ contains all features.

In matrix representation:

$$Z^{[1]} = \begin{bmatrix} | & | & | \\ z^{[1](1)} & z^{[1](2)} & z^{[1](3)} \\ | & | & | \end{bmatrix}$$

$$\vec{X} = \begin{bmatrix} | & | & | \\ \vec{x}^{(1)} & \vec{x}^{(2)} & \vec{x}^{(3)} \\ | & | & | \end{bmatrix}$$

Then

$$Z^{[1]} = \begin{bmatrix} | & | & | \\ z_1^{[1](1)} & z^{[1](2)} & z^{[1](3)} \\ | & | & | \end{bmatrix}$$
$$= W^{[1]} \begin{bmatrix} | & | & | \\ \vec{x}^{(1)} & \vec{x}^{(2)} & \vec{x}^{(3)} \\ | & | & | \end{bmatrix} + b^{[1]}$$

Training Samples are columns.

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

To train, the process is:

1. initialize $w^{[1]}$, $b^{[1]}$, $w^{[2]}$, $b^{[2]}$ to small random numbers.
2. for all $i \in [1, n]$, compute the o/p probability from $a^{[2](i)}$ and then

then

$$\max_{w,b} \sum_{i=1}^{n} \left( y^{(i)} \log a^{[2](i)} + (1 - y^{(i)}) \log (1 - a^{[2](i)}) \right)$$

↳ use gradient ascent

find $w, b$ that maximizes the above

Cross entropy function means training a neural network.

---

## Back propagation

Neural network has 2 related components

1. Network architecture
   - a. # hidden layers.
   - b. # neurons

(b) # neurons

(c) # connections b/w neurons

(d) activation functions

(2) Parameters / Weights

For individual neurons (aka, for individual optimizations), we've determined how to select parameters.

But activations of layer $l$ change the training set for layer $l+1$. So how do we find $w, b$!

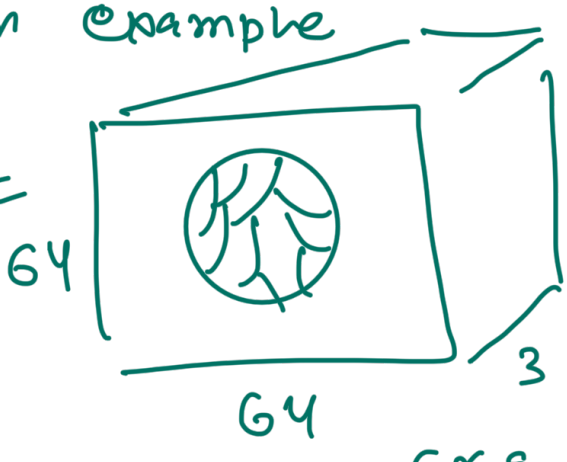
We need a new kind of solution, that can help in find the concatenated solution:

One such solution is the "Back propagation algorithm" because in this weights are updated backward from the last layer to the first layer.

---

Consider an example

$x^{(i)}$ =




Flatten ⇒

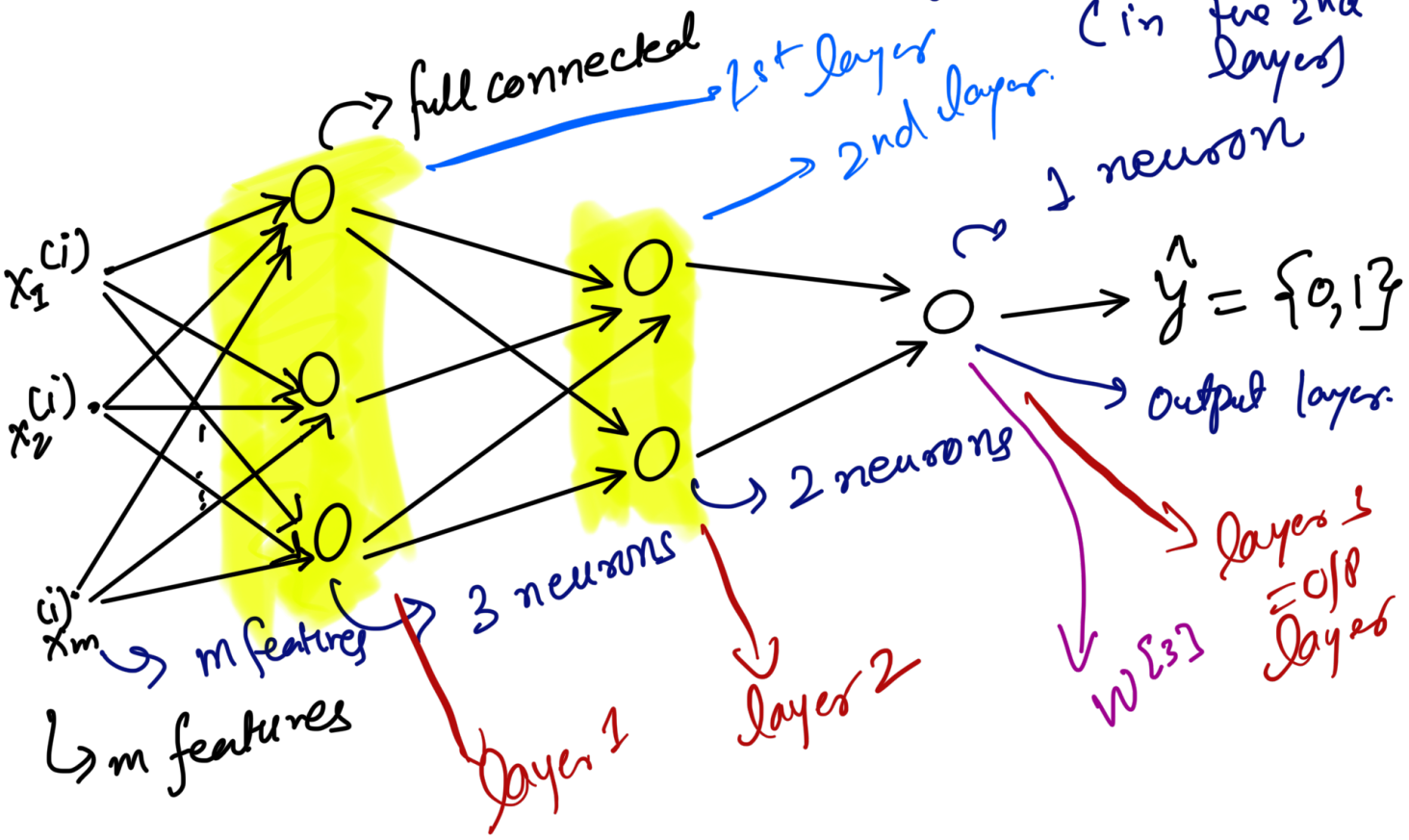
$$\begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_m^{(i)} \end{bmatrix}$$

$m = 64 \times 64 \times 3$

NN → {0, 1}

For this kind of data, it will require a lot of neurons (Hidden Layer).

Consider a 2-layer hidden network (a fully connected) with 3 neurons (in the first layer), 2 neurons (in the 2nd layer) and 1 neuron (output layer).




Let's write the parameters by hand:

$$z^{[1]} = W^{[1]} \cdot x^{(i)} + b^{[1]}$$

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

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

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

$$z^{[3]} = W^{[3]} a^{[2]} + b^{[3]}$$

$$\hat{y}^{(i)} = a^{[3]} = g(z^{[3]})$$

$z^{[1]}, a^{[1]} \in \mathbb{R}^{3\times 1}$

$z^{[2]}, a^{[2]} \in \mathbb{R}^{2\times 1}$

$z^{[3]}, a^{[3]} \in \mathbb{R}^{2\times 1}$

What about the dimension of $W$

$$z^{[1]} = W^{[1]} x^{(i)} + b^{[1]}$$

$3\times 1$ $\quad (?\times ?)$ $\quad m\times 1$ $\quad 3\times 1$

$(?\times ?)$: $?$ → $3$, $?$ → $m$

$\hookrightarrow W^{[1]} \in \mathbb{R}^{3\times m}$.

Similarly, $W^{[2]} \in \mathbb{R}^{2\times 3}$ $\quad b^{[2]} \in \mathbb{R}^{2\times 1}$

$W^{[3]} \in \mathbb{R}^{1\times 2}$ $\quad b^{[3]} \in \mathbb{R}^{1\times 1}$

In Total:

Layer 1: $W^{[1]}, b^{[1]}$ $\quad 3\cdot m + 3\cdot 1 = 3m+3$

Layer 2: $W^{[2]}, b^{[2]}$ $\quad 2\cdot 3 + 2\cdot 1 = 8$

Layer 3 (O/P Layer):

$W^{[3]}, b^{[3]}$ $\quad 1\cdot 2 + 1\cdot 1 = 3$

$3m+3+8+3$

$= 3m+14$ Parameters.

How do we initialize these parameters?

Can be set them to 0?

$$z^{[1]} = W^{[1]} \cdot x^{(i)} + b^{[1]}$$

$$= [0] \cdot x^{(i)} + 0 = 0$$

In practice, we do initialization near zero using a normal distribution, $N(0, 0.1)$ with a small standard deviation.

Our loss is

$$\mathcal{L}(\hat{y}, y) = -[(1-y)\log(1-\hat{y}) + y \log \hat{y}]$$

$\log(1-(\hat{y}+\epsilon))$ ; $(\hat{y}+\epsilon)$

As, $\hat{y} = g(z^{[2]})$

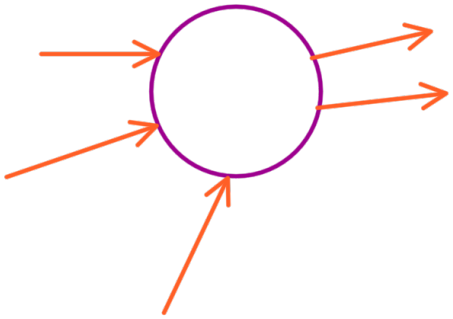
So, we see that it will never be exactly 0 or 1.

Hence, we see that we never calculate $\log(1)$, rather we calculate $\log(1-\epsilon)$, $0 < \epsilon << 1$

With this idea of our loss function, and what we learnt in regression, what can we do about N.N.?

Let's look at the updater function during gradient descent.

Consider that we are at layer $l$:



$$z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$$

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

at layer $l$.

From our logistics regression

$$\theta_j = \theta_j - \alpha \frac{\partial L}{\partial \theta_j}$$

$$W^{[l]} = W^{[l]} - \alpha \frac{\partial L}{\partial W^{[l]}}$$

$$b^{[l]} = b^{[l]} - \alpha \frac{\partial L}{\partial b^{[l]}}$$

For linear regression $\theta^T = [w_0, w_1 \dots w_n]$

$x = [\underbrace{1}_{b}, x_1 x_2 \dots x_n]$

Then, if we can calculate $\frac{\partial L}{\partial W^{[l]}}$ & $\frac{\partial L}{\partial b^{[l]}}$

then we can can use gradients-based method.

Pick layer $l=3$ (last layer)

$$L = -[(1-y)\log(1-\hat{y}) + y\log(\hat{y})]$$

$$\hat{y} = g(z^{[3]}) \quad g(\cdot) = \sigma(z^{[3]})$$

a sigmoid

$$\frac{\partial L}{\partial w^{[3]}} = -\frac{\partial}{\partial w^{[3]}}\left[(1-y)\log(1-g(z^{[3]})) + y\log(g(z^{[3]}))\right]$$

$$= -(1-y)\left[\frac{\partial}{\partial w^{[3]}}\log(1-g(z^{[3]}))\right] - y\frac{\partial}{\partial w^{[3]}}\log(g(z^{[3]}))$$

— ①

Some rules

① $\sigma' = \sigma(1-\sigma)$ for $\sigma$ = sigmoid

② $\frac{\partial}{\partial x}\log(f(x)) = \frac{1}{f(x)}\cdot f'(x)$

③ $f(x) = h(g(x)) \Rightarrow f'(x) = h'(g(x))\, g'(x)$

with this in mind:

$$\frac{\partial}{\partial w^{[3]}}\log(1-g(z^{[3]})) = \frac{1}{1-g(z^{[3]})}\cdot\frac{\partial(1-g(z^{[3]}))}{\partial w^{[3]}}\cdot\frac{\partial z^{[3]}}{\partial w^{[3]}}$$

$$= \frac{1}{1-g(z^{[3]})}(-1) \underbrace{\frac{\partial g(z^{[3]})}{\partial w^{[3]}}}_{g'(z^{[3]})} \cdot \frac{\partial z^{[3]}}{\partial w^{[3]}}$$

Rewriting (1)

$$-(1-y)(1A) - y(1B)$$

let's calculate a few items:

$$z^{[3]} = w^{[3]} a^{[2]} + b^{[3]}$$

$$\frac{\partial z^{[3]}}{\partial w^{[3]}} = \frac{\partial (w^{[3]} a^{[2]} + b^{[3]})}{\partial w^{[3]}}$$

✓

$$= \frac{\partial (w^{[3]} \cdot a^{[2]})}{\partial w^{[3]}} = a^{[2]T}$$

✓ $$\frac{\partial z^{[3]}}{\partial b^{[3]}} = \frac{\partial (w^{[3]} \cdot a^{[2]} + b^{[3]})}{\partial b^{[3]}} = 1$$

and $$g'(z^{[3]}) = g(z^{[3]})(1-g(z^{[3]}))$$

(1A) becomes:

$$= (-1) \frac{1}{1-g(z^{[3]})} g(z^{[3]})(1-g(z^{[3]})) \cdot a^{[2]T}$$

$$= -g(z^{[3]}) a^{[2]T}$$

(1B) becomes.

$$\frac{1}{g(z^{[3]})}\, g(z^{[3]})\,(1-g(z^{[1]}))\,a^{[2]T}$$

$$= (1-g(z^{[2]}))\,a^{[2]T}$$

Hence eqn (1)

$$\frac{\partial L}{\partial W^{[3]}} = -(1-y)(-1)\,g(z^{[3]})\,a^{[2]T} - y\,(1-g(z^{[3]}))\,a^{[2]T}$$

$$= [\,g(z^{[3]}) - y\,g(z^{[3]}) - y + y\,g(z^{[3]})\,]\cdot a^{[2]T}$$

$$= [\,g(z^{[3]}) - y\,]\,a^{[2]T}$$

(Note: $a^{[3]} = g(z^{[3]})$)

$$(1) \Rightarrow \frac{\partial L}{\partial W^{[3]}} = (a^{[3]} - y)\,a^{[2]T}$$

Given this, can we calculate $\frac{\partial L}{\partial W^{[2]}}$?

Chain Rule:

$$\frac{\partial L}{\partial W^{[3]}} = \frac{\partial L}{?}\cdot\frac{?}{\partial W^{[2]}}\cdot ?$$

$$a^{[3]} = g(z^{[3]})$$

Some important relationship:

(I) $z^{[3]}$ is related to $a^{[2]}$, $\frac{\partial z^{[3]}}{\partial a^{[2]}}$

$$\left( z^{[3]} = w^{[3]} \cdot a^{[2]} + b^{[3]} \right)$$

(II) $a^{[2]}$ depends on $z^{[2]}$

$$a^{[2]} = g(z^{[3]}) \Rightarrow \frac{\partial a^{[2]}}{\partial z^{[2]}}$$

(III) $z^{[2]}$ depends on $w^{[2]}$

$$z^{[2]} = w^{[2]} a^{[1]} + b^{[2]} \Rightarrow \frac{\partial z^{[2]}}{\partial w^{[2]}}$$

(IV) $\mathcal{L}$ depends on $a^{[3]}$ $\Rightarrow \frac{\partial \mathcal{L}}{\partial a^{[3]}}$

(V) $a^{[3]}$ depends on $z^{[3]}$. $\Rightarrow \frac{\partial a^{[3]}}{\partial z^{[3]}}$

So

$$\frac{\partial \mathcal{L}}{\partial w^{[2]}} = \underbrace{\frac{\partial \mathcal{L}}{\partial a^{[3]}} \cdot \frac{\partial a^{[3]}}{\partial z^{[3]}}}_{\text{IV \& V}} \cdot \underbrace{\frac{\partial z^{[3]}}{\partial a^{[2]}}}_{\text{I}} \cdot \underbrace{\frac{\partial a^{[2]}}{\partial z^{[2]}}}_{\text{II}} \cdot \underbrace{\frac{\partial z^{[2]}}{\partial w^{[2]}}}_{\text{III}}$$

Since $\frac{\partial \mathcal{L}}{\partial w^{[3]}} = (a^{[3]} - y)\, a^{[2]}$ &

Recall (IA) $\frac{\partial z^{[3]}}{\partial w^{[3]}} = a^{[2]T}$

$$\frac{\partial L}{\partial W^{[2]}} = (a^{[2]} - y)\frac{\partial z^{[3]}}{\partial W^{[2]}}$$

$$\frac{\partial L}{\partial z^{[3]}} = \frac{\partial L}{\partial a^{[3]}} \cdot \frac{\partial a^{[3]}}{\partial z^{[3]}}$$

$$\frac{\partial L}{\partial W^{[2]}} = (a^{[3]} - y) \cdot \frac{\partial z^{[3]}}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial W^{[1]}}$$

$\frac{\partial a^{[2]}}{\partial z^{[2]}} \rightarrow g'(z^{[2]})$

$a^{[1]T}$ (from 2A)

$W^{[3]}\left(\frac{\partial z^{[2]}}{\partial a^{[2]}} = \frac{\partial}{\partial a^{[2]}}\left[W^{[3]} a^{[2]} + b^{[3]}\right)\right)$

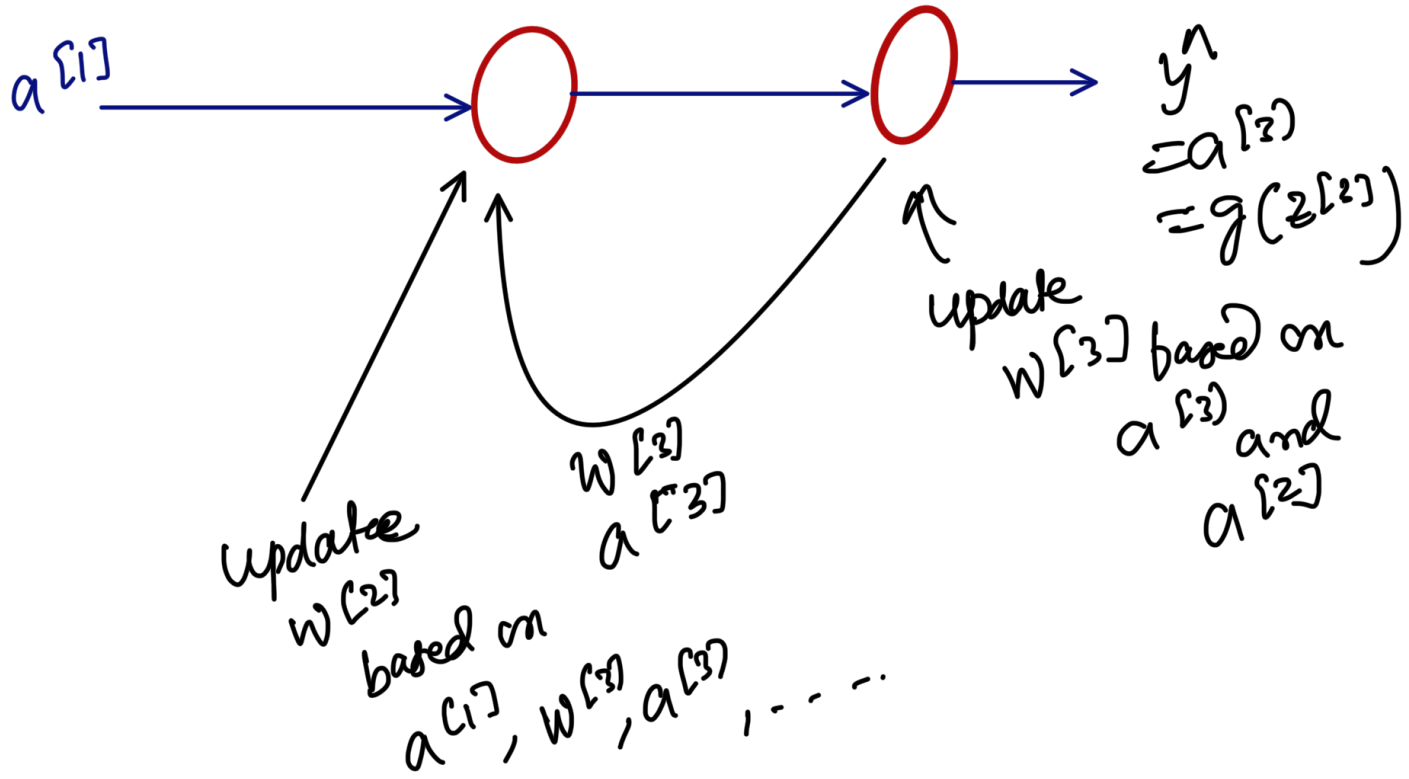
$$\Rightarrow \frac{\partial L}{\partial W^{[2]}} = (a^{[3]} - y)\, W^{[2]}\, g'(z^{[2]})\, a^{[1]T}$$

$1\times1 \quad 1\times2 \quad 2\times1 \quad 1\times3$

$$\Rightarrow \frac{\partial L}{\partial W^{[2]}} = W^{[3]T} \cdot g'(z^{[2]}) \cdot (a^{[3]} - y) \cdot a^{[1]T}$$

$2\times1 \quad 2\times1 \quad 1\times1 \quad 1\times3$

Or in essence, changes to the cost cost $W^{[2]}$ are based on the downstream weights $W^{[2]}$




But $w^{[2]}$ and $a^{[2]}$ propagate backward to do this optimization!

## Full Gradient descent

For any layer

$$w^{[l]} = w^{[l]} - \alpha \frac{\partial J}{\partial w^{[l]}}$$

where $J = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}^{(i)}$

$m$ = #features

$i$ = training samples

In this kind of scenario, $\mathcal{L}^{(i)}$ will be noisy.

$J$ will be computationally difficult.

Solution: Mini batch Gradient Descent

$$J_{MB} = \frac{1}{B} \sum_{i=1}^{B} \mathcal{L}^{(i)}$$

where $B << m$ is the no. of samples in the mini batch

---

Momentum

For any layer $l$

$$v_{dw}[l] = \beta v_{dw}[l] + (1-\beta)\frac{\partial J}{\partial w[l]}$$

($\beta$: momentum parameter.)

$$w[l] = w[l] - \alpha v_{dw}[l].$$

($\alpha$: learning rate)