Neural Network

* Neural Network :-

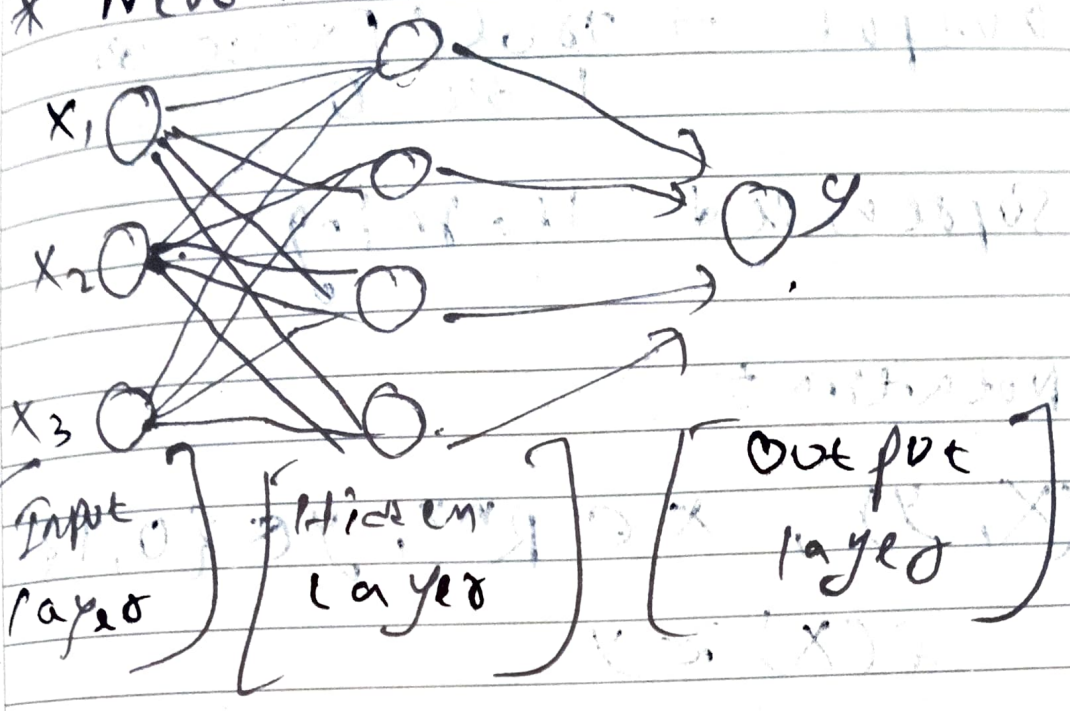


Fig : Standard NN

(NN → Image recognition, Object detection NLP, financial time series analysis

RNN → Sequential data → NLP, time - series analysis

Binary Classification

Output → True / False
1 or 0

Supervised learning

Notation :

(X, y), $x \in R^{n_x}$, $y \in \{0, 1\}$

$f(X) \approx y$

Logistic Regression

$$z = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$
$$= w^T x$$

$$\sigma(z) = \frac{1}{1 + e^{-z}} \Rightarrow \{0, 1\}$$

$$\hat{y} = \frac{1}{1 + e^{-(w^T x)}}$$

$x_1, x_2, x_3, \ldots x_n \rightarrow$ input

$w_1, w_2, \ldots, w_n \rightarrow$ weight

$w_0 \rightarrow$ bias (intercept)

* Loss function
* Gradient Descent update rule

Given x, $\hat{y} = P(y=1 \mid x)$

$x \in R^{n_x}$

parameter: $w \in R^{n_x}$, $b \in R$

* $\hat{y} = \sigma(w^T x + b)$

Loss function:

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

For m no. of data:

Cost function:

$$J(w, b) = \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$$

Gradient Descent Rule:

Repeat {

$$w := w - \alpha \frac{d J(w)}{d w}$$

}

$$w := w - \alpha \, dw$$

$\alpha \rightarrow$ learning rate

$$dw = X(\hat{y} - y)^T / m$$

$$w := w - \alpha \frac{dJ(w)}{dw}$$

$$\frac{dJ(w)}{d\hat{y}} = \frac{-y}{\hat{y}} + \frac{1-y}{1-\hat{y}}$$

$$dz = \frac{dJ(w)}{dz} = \hat{y} - y = db$$

$$\frac{dJ(w)}{dw} = (\hat{y} - y) \cdot X = dw$$

For multiple samples:

$$\frac{dJ(w)}{dw} = \frac{1}{m} X^T (\hat{y} - y)$$

$$\frac{dJ(w,b)}{db} = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right) = db$$

* Vectorized dot product:

$z = np.dot(w, x)$

$w^T x$

$w := w - \alpha\, dw$

$b := b - \alpha\, db$

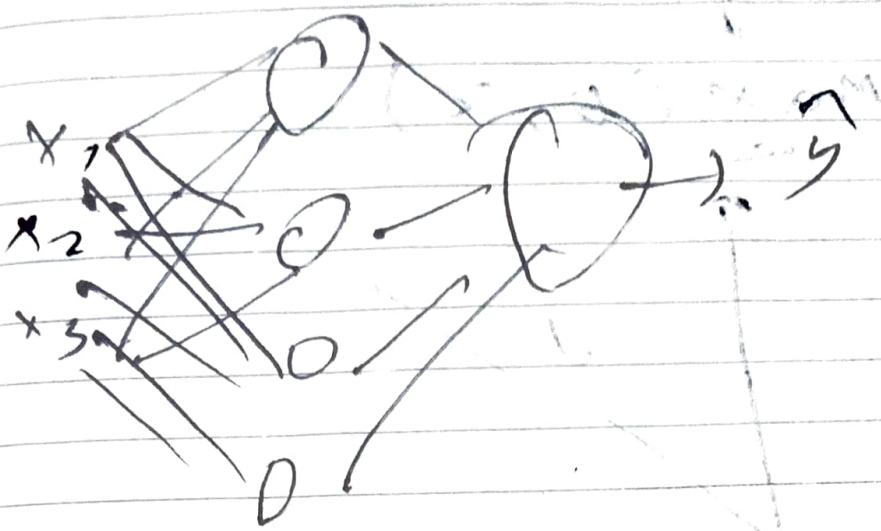
* Logistic regression cost function:

If $y = 1$: $p(y|x) = \hat{y}$

If $y = 0$: $p(y|x) = 1 - \hat{y}$

$p(y|x) = (\hat{y}^{y})(1-\hat{y})^{1-y}$

NN



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

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

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

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

Activation func :

Sigmoid func. is our activation func. for previous cases

(1) tanh func. works better for hidden layer.

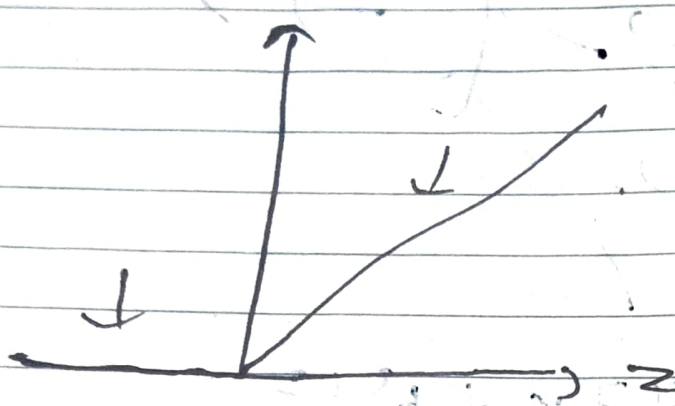
$$a = \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

For output layer → sigmoid is better.

Rectified Linear Unit

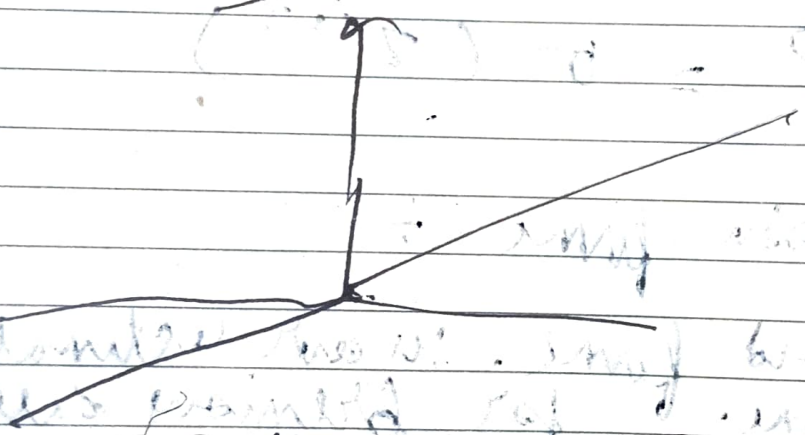
(ii) ReLU func.

$a = Max(0, z)$



mostly used

(iii) Leaky ReLU



$a = Max(0.01z, z)$

* Derivative of activation func:

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

$$d g(z)/dz = g(z)(1-g(z))$$

(ii) Tanh: $g(z) = \tanh(z)$

$$dg(z)/dz = 1 - (g(z))^2$$

(iii) ReLU: $g(z) = \max(0, z)$

$$g'(x) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z > 0 \end{cases}$$

Leaky ReLU

$$g'(x) = \begin{cases} 0.01 & \text{if } z < 0 \\ 1 & \text{if } z > 0 \end{cases}$$

* Gradient descent:

$$\theta := \theta - \alpha \frac{dL}{d\theta}$$

L → cost

θ → parameter (weight, bias)

α → Learning rate

* Forward - Propagation :

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

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

$$z^{[2]} = w^{[2]} A^{[1]} + b^{[2]}$$

$$A^{[2]} = g^{[2]}(z^{[2]}) = \sigma(z^{[2]})$$

* Back propagation :

$$dz^{[2]} = A^{[2]} - Y$$

$$Y = [y^{(1)}, y^{(2)}, \ldots, y^{(m)}]$$

$$dw^{[2]} = \frac{1}{m} dz^{[2]} A^{[1]T}$$

$$db^{[2]} = \frac{1}{m} np.sum(dz^{[2]}, axis=1, keepdims=True)$$

$$dz^{[1]} = w^{[2]T} dz^{[2]} * g^{[1]'}(z^{[1]})$$

(element wise product

$$dW^{[l]} = \frac{1}{m} dZ^{[l]} X^T$$

$$db^{[l]} = \frac{1}{m} np.sum(dZ^{[l]}, axis=1, keepdims=True)$$

Dimension of weights & bias:

m → input

n → neuron

Weight → (n, m)

bias → (n, 1)

Random initialization:

```
W = np.random.randn((n, m)) * 0.01

b = np.zero((n, 1))
```

DNN

$L \rightarrow$ No. of layer

$n^{[l]} \rightarrow$ #r, Neurons in layer l / units /

$a^{[l]} \rightarrow$ Activation in layer l

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

$w^{[l]}$, $b^{[l]} \rightarrow$ weight for $z^{[l]}$

$X \rightarrow a^{[0]} \rightarrow$ input

* Forward Propagation in DNN:

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

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

Back Propagation in DNN :-

For layer l

Input $da^{[l]}$

Output $da^{[l-1]}, dw^{[l]}, db^{[l]}$

$$dz^{[l]} = da^{[l]} * g^{[l]'}(z^{[l]})$$

$$dw^{[l]} = dz^{[l]} \cdot a^{[l-1]}$$

$$db^{[l]} = dz^{[l]}$$

$$da^{[l-1]} = w^{[l]T} \cdot dz^{[l]}$$

$$dz^{[l]} = w^{[l+1]T} \cdot dz^{[l+1]} * g^{[l]'}(z^{[l]})$$

$$dw^{[l]} = \frac{1}{m} dz^{[l]} \cdot A^{[l-1]T}$$

$$db^{[l]} = \frac{1}{m} np.sum(dz^{[l]}, axis=1, keepdims=True)$$

$$dA^{[l-1]} = w^{[l]T} dz^{[l]}$$