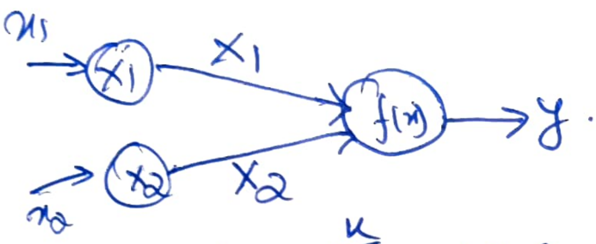
Problem 1

x_1	x_2	Class
1	1	+1
-1	-1	-1
0	0.5	-1
0.1	0.5	-1
0.2	0.2	+1
0.9	0.5	+1

$W^T x = 0$

W initial = [1, 1].



Let

$$f\,y(x) \rightarrow \begin{cases} 1, & y(x) > 0 \\ 0, & y(x) = 0 \\ -1, & y(x) < 0 \end{cases}$$

$$y = b + \sum_{i=0}^{k} w_i x_i .$$

(1) Let $b = 0$, Learning rate, $\alpha = 1$.

&

$y_{in} = w_1 x_1 + w_2 x_2$

1.1. The perceptron Learning Algorithm will Converge in 3 steps.

1.2

Step-1

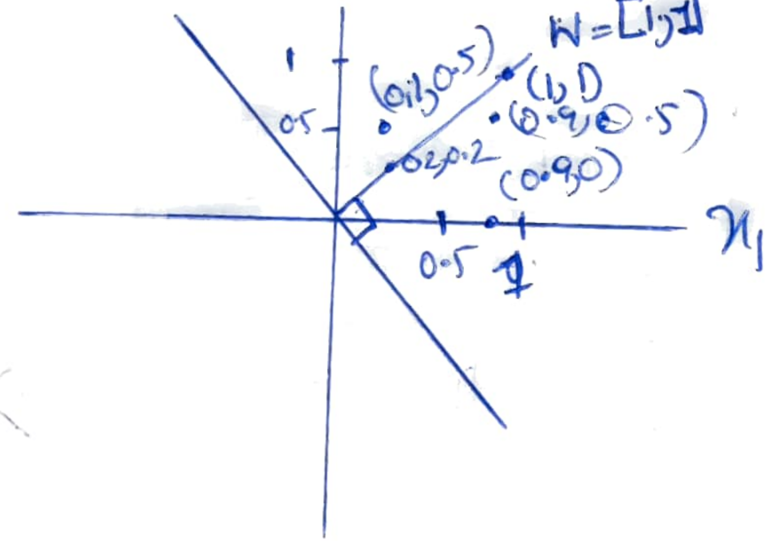
Class	x_1	x_2	w	$W^T x$	Correct	Corrected w	
1	1	1	[1, 1]	2	✓	—	$W^T x = 2$, $f(x) > 1$ Class > 0
-1	-1	-1	[1, 1]	-2	✓	—	$W^T x = -2$, $f(x) < 0$, Class < 0
-1	0	0.5	[1, 1]	0.5	✗	$W = W_1 - x$ = [1, 0.5]	$f(x) > 0$, Class < 0
-1	0.1	0.5	[1, 0.5]	0.35	✗	$W = W_1 - x$ = [0.9, 0]	$f(x) > 0$, Class < 0
1	0.2	0.2	[0.9, 0]	0.18	✓	—	$f(x) > 0$, Class > 0
1	0.9	0.5	[0.9, 0]	0.81	✓	—	$f(x) > 0$, Class > 0

$W = [1,1]$

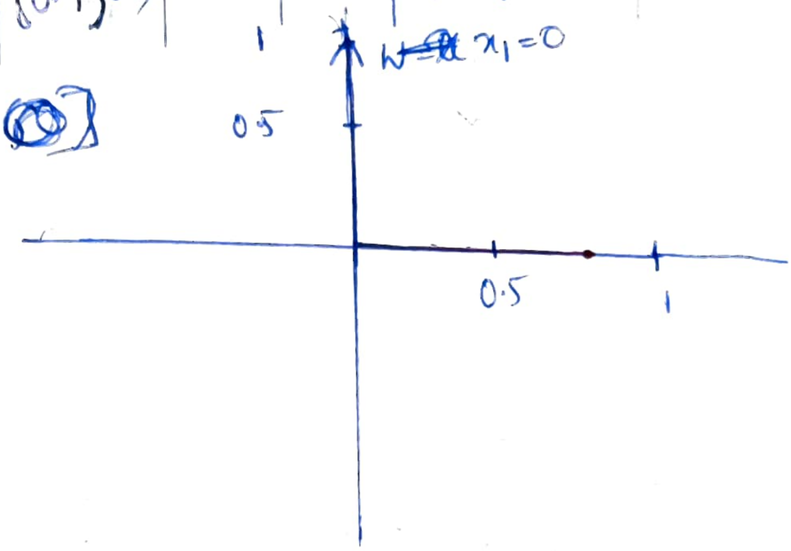
$$W^T x = [1,1]^T [x_1 + x_2] = x_1 + x_2 = 0$$

$+\ d$

$= 1 \times 1 + 1 \times 1 + 0 =$



Class	x_1	x_2	W	W^Tx	Correct	Corrected ω
1	1	1	(0.9, 0)	0.9	✓	$f(x)>0$, class>0
-1	-1	-1	(0.9, 0)	-0.9	✓	$f(x)>0$, class0
-1	0	0.5	(0.9, 0)	0	–	[0.9, -0.5]
-1	0.1	0.5	(0.9, -0.5)	-0.25	✓	$f(x)<$, class<0
1	0.2	0.2	(0.9, -0.5)	0.08	✓	$f(x)>0$ class>0
1	0.9	0.5	(0.9, 0.5)	0.46	✓	$f(x)>$class>0



for $W=(0.9, 0)$

$[0.9, 0]\begin{bmatrix}x_1\\x_2\end{bmatrix}=0$

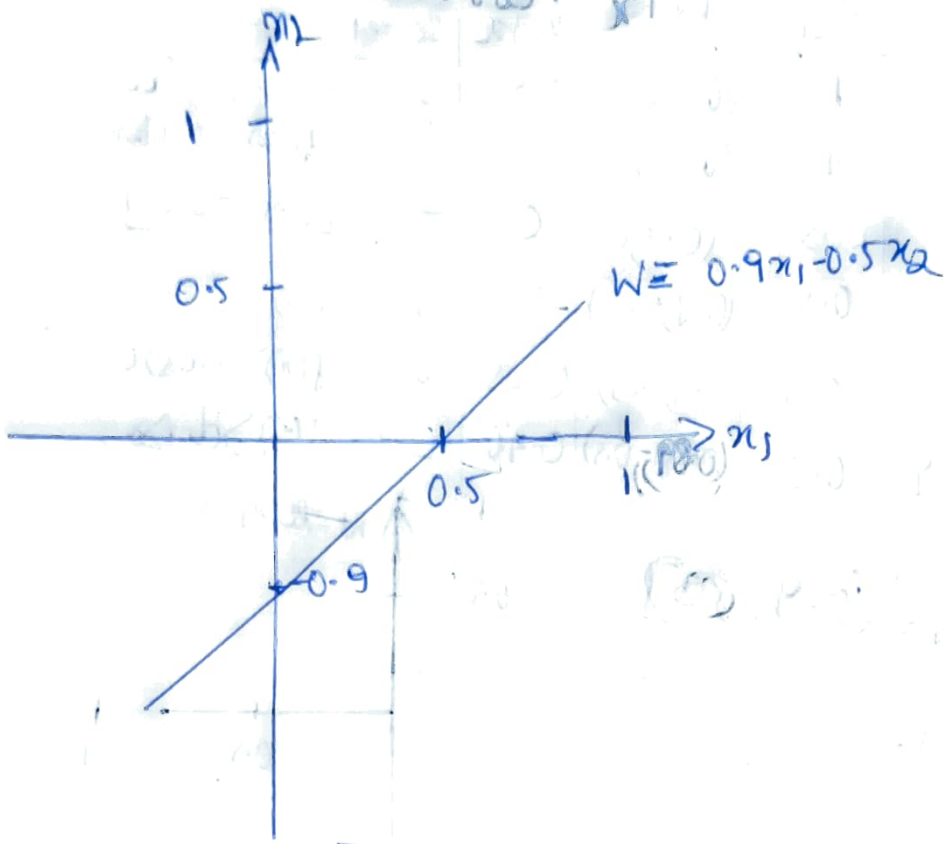
$\Rightarrow 0.9x_1=0$

$\Rightarrow x_1=0$

Step 3

Class	x_1	x_2	W	W^Tx	Correct	Correct ω	
1	1	1	(0.9, -0.5)	0.4	✓	–	$f(x)>0$, class>0
-1	-1	-1	(0.9, -0.5)	-0.4	✓	–	$f(x)<0$, class<0
-1	0	0.5	(0.9, -0.5)	-0.25	✓	–	$f(x)<0$, class<0
-1	0.1	0.5	(0.9, -0.5)	-0.16	✓	–	$f(x)<0$, class<0
1	0.2	0.2	(0.9, -0.5)	0.08	✓	–	$f(x)>0$, class>0
1	0.9	0.5	(0.9, -0.5)	0.56	✓		$f(x)>0$, class>0

for $W=(0.9, -0.5)$



$W = [0.9, -0.5]$

$W^T x = 0.9x_1 - 0.5x_2 = 0$

$\Rightarrow x_1$

The final Decision boundary is $(0.9, -0.5)$.

③ Prove that Perceptron Learning Algorithm... converges in a finite number of steps.

Sol:

<u>Definition</u>:- Two set P and N of points in an n-dimensional space are called absolutely linearly separable if n+1 real numbers $w_0, w_1, \dots, w_n$ exist such that every point $(x_1, x_2, \dots, x_n) \in P$ satisfies $\sum_{i=1}^{n} w_i * x_i > w_0$ and every point $(x_1, x_2, \dots x_n) \in N$ satisfies $\sum_{i=1}^{N} w_i * x_i < w_0$.

<u>Proposition</u> :- If the sets P and N are finite & linearly separable, the perceptron learning algorithm updates the weight vector w_T a finite number of times. In other words, if the vectors in P and N are tested cyclically one after the other, a weight w_t is found after a finite number of steps t which can separate the two sets.

<u>Setup</u> :-

* If $x \in N$ then $-x \in P$. ($\because w^T x < 0 \Rightarrow w^T(-x) \geq 0$)
* We can thus consider a single set $P' = P \cup N$ and for every element $p \in P'$ ensure that $w^T p \geq 0$.
* ~~Let~~ we will normalize all the p's so that $\|p\| = 1$.
* Let w^* ~~be~~ be the normalized solution vector.

Proofs:-

* Now suppose at time step t we inspected the point p_i, and found that $w^T \cdot p_i < 0$.
* we make a correction $w_{t+1} = w_t + p_i$
* Let β be the angle between w^* and w_{t+1}

$$\cos\beta = \frac{\cancel{w^* w_{t+1}}\ w^* w_{t+1}}{\|w_{t+1}\|}$$

$$\text{Numerator} = w^* \cdot w_{t+1} = w^*(w_t + p_i)$$

$$= w^* w_t + w^* p_i$$

$$\geq w^* w_t + \delta \quad (\delta = \min\{w^* \cdot p_i | \forall i\})$$

$$\geq w^*(w_{t-1} + p_j) + \delta$$

$$\geq w^* w_{t-1} + w^* p_j + \delta$$

$$\geq w^* w_{t-1} + 2\delta$$

Every time we make a correction a quantity δ is added. Say if K corrections are made. a quantity of $K\delta$ is added.

$$\cos\beta = \frac{(w^* w_{t+1})}{\|w_{t+1}\|} \quad (\text{definition})$$

Numerator $\geq w^* w_0 + i\delta$ (proved by induction)

Denominator$^2 = \|w_{t+1}\|^2$

$$= (w_t + p_i)$$

$$= |w_t|^2 + 2 w_t p_i + |p_i|^2 \leq |w|^2 + |p|^2 \quad (\because w_t p_i \leq 0)$$

$$\leq (\|w_{t-1}\|^2 + 1$$

$$\leq \|w_{t-1}\|^2 + 1 + 1$$

Denominator $\leq \|w_{t-1}\|^2 + (K)$.

$$\cos\beta \geq \frac{w^* w_0 + K\delta}{\sqrt{\|w_0\|^2 + K}}.$$

$\cos\beta$ thus proportionat to $\sqrt{K}$

$\therefore$ K has to be finite so that $\cos\beta \leq 1$.

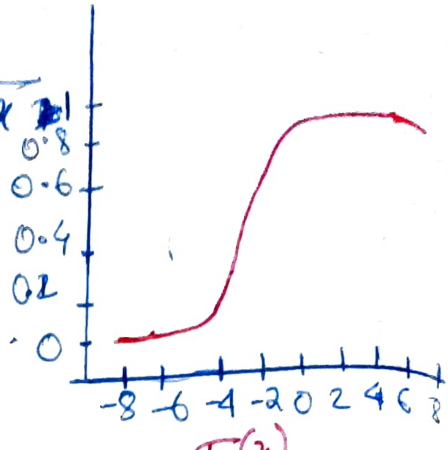
$\therefore$ Thus there can only be a finite number of corrections (k) to w and the perceptron algo will converge.

③ ① Compute Derivative of following activation functions:-

① Sigmoid.

$$\text{Sigmoid}(x) = \sigma(x) = \frac{1}{1+e^{-x}}$$

$$\frac{d}{dx}\sigma(x) = \frac{d}{dx}\frac{1}{1+e^{-x}}.$$



$\sigma(x)$

Applying quotient rule:-

$$\frac{d}{dx}\frac{f(x)}{g(x)} = \frac{\left(g(x)\frac{d}{dx}f(x)\right) - \left(f(x)\frac{d}{dx}g(x)\right)}{\{g(x)\}^2}$$

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

$$= \frac{e^{-x}}{(1+e^{-x})^2}$$

$$= \frac{1+e^{-x}-1}{(1+e^{-x})^2}$$

$$= \frac{1+e^{-x}}{(1+e^{-x})^2} - \frac{1}{(1+e^{-x})^2}$$

$$= \frac{1}{(1+e^{-x})} - \frac{1}{(1+e^{-x})^2}$$

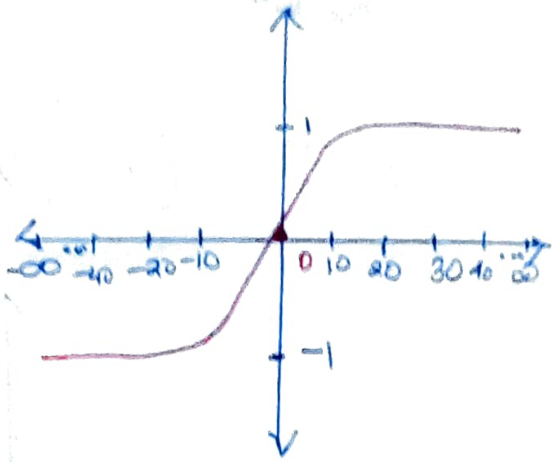
$$= \frac{1}{1+e^{-x}}\left\{1 - \frac{1}{(1+e^{-x})}\right\}$$

$$= \sigma(x)(1-\sigma(x))$$

$$\therefore \frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x))$$

(11) tanh

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$
$$= T(x)$$



$f(x) = \tanh(x)$

$$\frac{d}{dx}T(x) = \frac{d}{dx}\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

Applying quotient rul

$$\frac{d}{dx}\frac{f(x)}{g(x)} = \frac{g(x)\frac{d}{dx}f(x) - f(x)\frac{d}{dx}g(x)}{g(x)2}$$

$$\frac{d}{dx}\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} = \frac{e^{x}+e^{-x}\left(\frac{d}{dx}\left(e^{x}-e^{-x}\right)\right) - e^{x}-e^{-x}\frac{d}{dx}\left(e^{x}+e^{-x}\right)}{\left(e^{x}+e^{-x}\right)^{2}}$$

$$= \frac{\left(e^{x}+e^{-x}\right)\left(e^{x}+e^{-x}\right) - \left\{\left(e^{x}-e^{-x}\right)\left(e^{x}-e^{-x}\right)\right\}}{\left(e^{x}+e^{-x}\right)^{2}}$$

$$= \frac{\left(e^{x}+e^{-x}\right)^{2}}{\left(e^{x}+e^{-x}\right)^{2}} - \frac{\left(e^{x}-e^{-x}\right)^{2}}{\left(e^{x}+e^{-x}\right)^{2}}$$

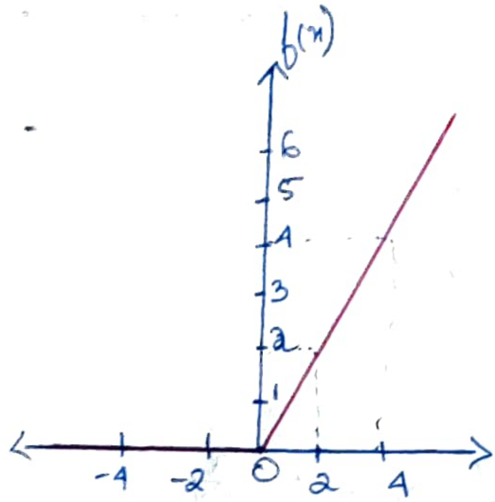
$$= 1 - \frac{\left(e^{x}-e^{-x}\right)^{2}}{\left(e^{x}+e^{-x}\right)^{2}}$$

$$\frac{d}{dx}T(x) = 1 - \left(T(x)\right)^{2}$$

$$\therefore \frac{d}{dx}\tanh = 1 - (\tanh)^{2}$$

(1.4) ReLU = max(0, x)

$$ReLU = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases}$$



f(x) = ReLU.

~~At x > 0~~

$$f'(x) = \lim_{x \to 0} \frac{\max(0, x+\Delta x) - \max(0, x)}{\Delta x}.$$

At $x > 0$

$$f'(x) = \lim_{x \to 0} \frac{x + \Delta x - x}{\Delta x} = 1$$

At $x < 0$

$$f'(x) = \lim_{x \to 0} \frac{0 - 0}{\Delta x} = 0.$$

At $x = 0$

$$f'(0) = \lim_{x \to 0^+} \frac{0 + \Delta x - 0}{\Delta x} = 1$$

$$f'(0) = \lim_{x \to 0^-} \frac{0 - 0}{\Delta x} = 0$$

LHL ≠ RHL.

Left hand limit ≠ Right hand limit.

$$\therefore \frac{d}{dx} ReLU = \frac{d}{dx} \max(0, x)$$

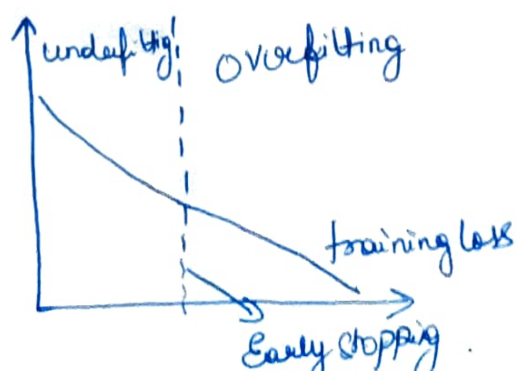
$$= \begin{cases} 1, & x > 0 \\ 0, & x < 0 \\ \text{Doesnot exist}, & x = 0 \end{cases}$$

(2) What are the Strategies you will follow to avoid over fitting in a neural network.

Ans Overfitting is a situation where the model of neural network learns the training data very well but doesnot perform well on a testing dataset.

(1) Simplifying the model by reducing the number of Internal layers and making the network smaller Accordingly change the input we reduction of Internal layers.

(2) Early Stopping:- Stopping the learning or training process of the model before it ~~loss~~ crosses over -fitting.



However it is important to find the right point to stop to avoid underfitting or overfitting.

③ Data Augmentation :- By making small changes in the existing data set, we make the model consider each augmented model as a distinct data set. This will make the neural network avoid overfitting.

④ Regularization:- It can be used to reduce the complexity of a model. The ~~most com~~ best way is to add penalty to the loss function in proportion to the weight of the model.

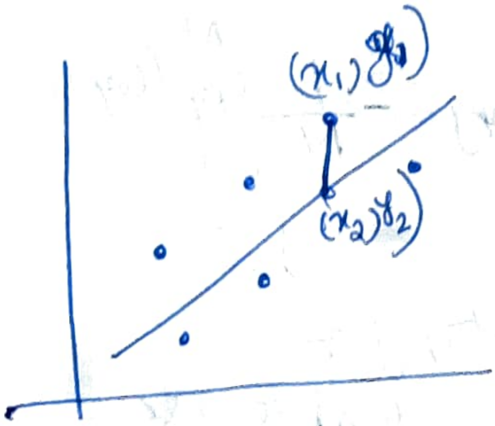
(4) Define following loss functions in brief.

(1) Mean Squared Error (MSE)

~~A loss~~

Mean Squared Error (MSE) Calculate the mean of distance of set data points from a regression line.

It is basically the average of errors squared from data.



$Error = y_1 - y_2$.

$Squared\ Error = (y_1 - y_2)^2$

Mean Squared error

$$= \frac{1}{n} (y - \hat{y})^2.$$

$$\cancel{= \frac{1}{n} \sum_{i=1}^{n} (y - y_i)^2}$$

(2) Binary Cross-Entropy (BCE).

It compares each predicted probabilities to actual class output which can either be 0 or 1. It than calculates the score that penalizes the probabilities based on the distance from the expected value. Binary entropy is the negative of the log of corrected predicted probabilities.

Eg ID	Actual ~~probability~~	Predicted prob	Corrected
1	1	P1	P1
2	1	P2	P2
3	1	P3	P3
4	0	P4	P4
5	0	P5	P5

~~log P1~~

$$\text{Binary Cross Entropy} = -\frac{1}{N}\sum_{i=1}^{N}\left(\log(P_i)\right)$$

Q(III) Categorical Cross Entropy :-

It is a loss function that is used in multi class classification tasks. These are tasks where an example can only belong to one out of many possible categories and the model must decide which one.

→ Softmax → Cross ~~entropy~~ loss

$f(s)$ / Softmax $= \dfrac{e^{s_i}}{\sum_i^{c} e^{s_j}}$

Cross Entropy $= -\sum_{i=}^{c} t_i \log(f(s)_i)$

$= -\log\left(\dfrac{e^{s_p}}{\sum_j^{c} e^{s_j}}\right)$.

Eg :- True label : Rabbit

Prediction :- Dog = 1, Cat = 4, Rabbit = 8, Squirrel = 2.

Softmax = Dog = e^1/Sum, Cat = e^4/SUM, Rabbit = e^8/Sum

Squirrel = e^2/Sum. // Sum = $e^1 + e^4 + e^8 + e^2$

Cross Entropy = $-(0 \times \ln(\text{Dog}) + 0 \times \log(\text{Cat})$

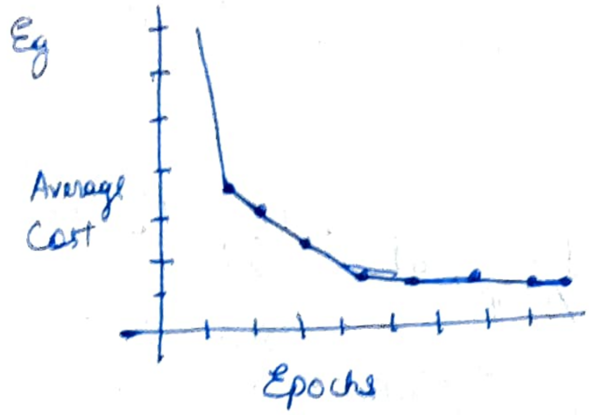
$+ 1 \times \log(\text{Rabbit}) + 0 \times \log(\text{Squirrel})$

$= \log(\text{Rabbit})$.

5 ① Batch Gradient Descent:

Gradient descent is an optimization technique for minimising a function. In Batch Gradient Descent, all the training examples and then use that mean gradient to update our parameters. It gives the best ~~result in case~~ result in case of convex or relatively smooth function.

Eg



The graph of cost vs epochs is also quite smooth because we are averaging over all the gradients of training data for a single step.

(ii) Stochastic Gradient Descent

Stochastic gradient descent attempts to find the global minimum by adjusting the configuration of the network after each training point. Instead of decreasing the error, or find the gradient, for the entire data set, this method merely decreases the error by approximating the gradient for a randomly selected

It can adjust the network parameters in such a way as to move the model out of a local minimum and toward a global minimum.

③ Mini Batch Gradient Descent:-

Mini Batch Gradient Descent is a variation of the gradient descent algorithm that splits the training dataset into small batches that are used to ~~collect~~ calculate model error and update model coefficients.

Implementation may choose to sum the gradient over the mini batch which further reduces the variance of the gradient.

Mini Batch gradient descent seeks to find a balance between the robustness of stochastic gradient descent and efficiency of batch gradient descent. It is the most common implementation of gradient descent. ~~used in the field of deep learning~~.

(IV) Momentum based Gradient Descent:-

(V) Adam Solver :

Adam Solver optimization algorithm is an extension to Stochastic gradient descent that recently seen broader adoption for deep learning application.

· the learning rate is maintained for each network weight and separately adapted as learning unfolds.

It used both ~~has~~ Adaptive gradient algorithm and Root mean square propagation ~~It is also make~~.

The algorithm calculate an exponential moving average of the gradient and the squared gradient, and the parameters control the decay rates of these these moving averages.

③ Let $x = [1, 1]^T$, $y = [1, 1]^T \in R^2$ and let $f : R^2 \Rightarrow R^2$ with $f(z) = z_1 x + z_2 y$ for any $z = [z_1, z_2]^T \in R^2$ further $z = g(r) = [r^2, r^3]$ where $R \in R$. Show how chain rule is applied. write down the expression for $\frac{\partial f}{\partial r}$ and also evaluate $\frac{\partial f}{\partial r}$ at $r = 2$.

$$\frac{\partial f}{\partial r} = \frac{\partial f}{\partial z} \frac{\partial z}{\partial r}.$$

$$z = [r^2, r^3]$$

$$\frac{\partial z}{\partial r} = [2r, 3r^2]^T$$

$$\frac{\partial f}{\partial z} = \cancel{\begin{bmatrix} z_1 x \\ z_2 y \end{bmatrix}}$$