classification — Sigmoid, Tanh, ReLU
softmax

Regression — Linear Activation function
we use it on the o/p layer.

loss = $(Y - \hat{Y})$ — single datapoint

cost funct = $\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y})^2$ — Batch datapoint

Loss function —

ANN

classification

① Binary Cross Entropy

② categorical cross entropy

Regression

① MSE

② MAE

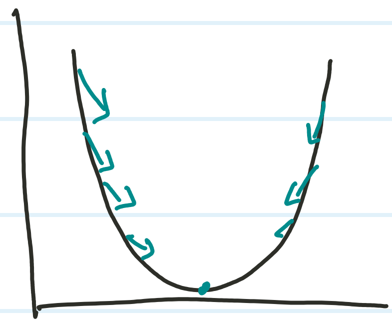
③ Huberloss

Regression

① MSE

$$loss = (y - \hat{y})^2$$

$$\text{cost fun} = \frac{1}{2n} \sum_{i=1}^{n} (y_i - \hat{y})^2$$



$$(a-b)^2 = a^2 - 2ab + b^2$$

$$\boxed{ax^2 + bx + c}$$

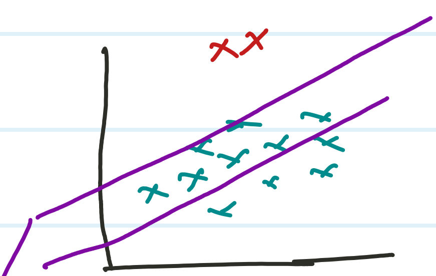
quadratic eqn

Adv:

① mse is differentiable
② It has one local and one global minima
③ It converg faster

Disadv

① Not Robust to outlier.

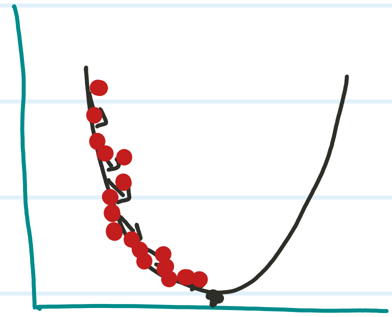


we are just penalizing error

② MAE

$$\text{Loss} = |y - \hat{y}|$$

$$\text{cost} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}|$$



Adv.

① Robust to outlies condition

Disadv.

① Convergen is slow

③ Huber Loss –

Combination of MSE and MAE

$$\text{cost fun.} = \begin{cases} \frac{1}{2n}\sum_{i=1}^{n} (y_i - \hat{y})^2 & \text{if } |y - \hat{y}| \le \delta \\ \delta|y - \hat{y}| - \frac{1}{2}\delta^2, \end{cases}$$

when outlier is present

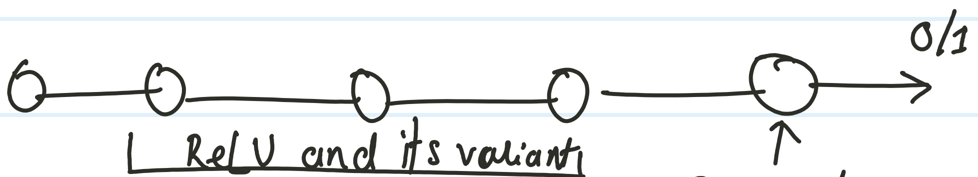
Loss and cost function for classification

① Binary cross Entropy -

$$\text{Loss fn} = -y * \log(\hat{y}) - (1-y) * \log(1-\hat{y})$$

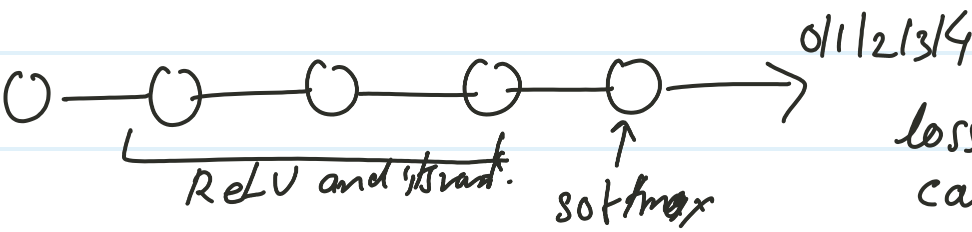
$$\text{Loss} = \begin{cases} -\log(1-\hat{y}), & \text{if } y = 0 \\ -\log(\hat{y}), & \text{if } y = 1 \end{cases}$$

It only used to find loss in Binary classification



$(y-\hat{y})$

Binary cross entropy



loss

categorical cross entrop

⑪ Categorical cross Entropy

	f_1	f_2	f_3	O/p	J: $j=1$	$j=2$	$j=3$
i	2	3	4	Good	1	0	0
	5	6	7	Bad	0	1	0
	8	9	10	Neutral	0	0	1

No. of category $c = 3$

$$\text{Loss}(x_i\ y_i) = \sum_{j=1}^{c} y_{ij} * \log(\hat{y}_{ij})$$

Actual value $= y_{ij} = [y_1, y_2, y_3 - - - - - y_c]$

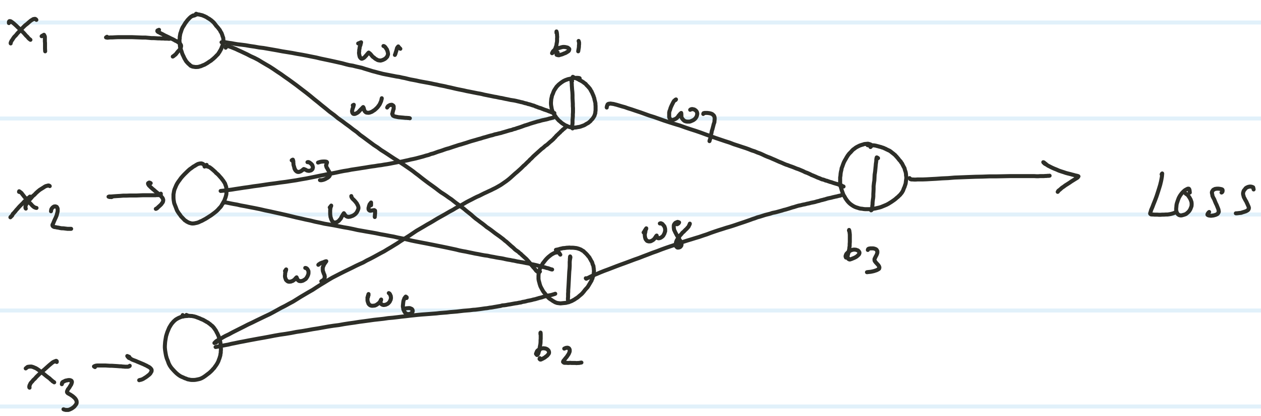
$[y_{21}, y_{22}, y_{23} - - - - 2_{2c}]$

$$y_{ii} = \begin{cases} 1, & \text{if the element is in the class} \\ 0, & \text{otherwise} \end{cases}$$

it used for multiclass classification

Optimizes

① Gradient Decent optimizes



weight update formula

$$w_{new} = w_{old} - \eta \boxed{\frac{\partial E}{\partial w_{old}}} \text{ — slop}$$

Epoch —

10000 datapoint

traning batch $\quad \frac{10000}{100} \Rightarrow 100$

Epoch-1 $\left\{ \begin{array}{l} \xrightarrow[]{100 \quad FP} \quad \text{cost} \\ \xleftarrow[100]{} \\ \quad BP \end{array} \right.$

Epoch = I reached on global minima

Adv.

1. Conversion will happen

Dis adv.

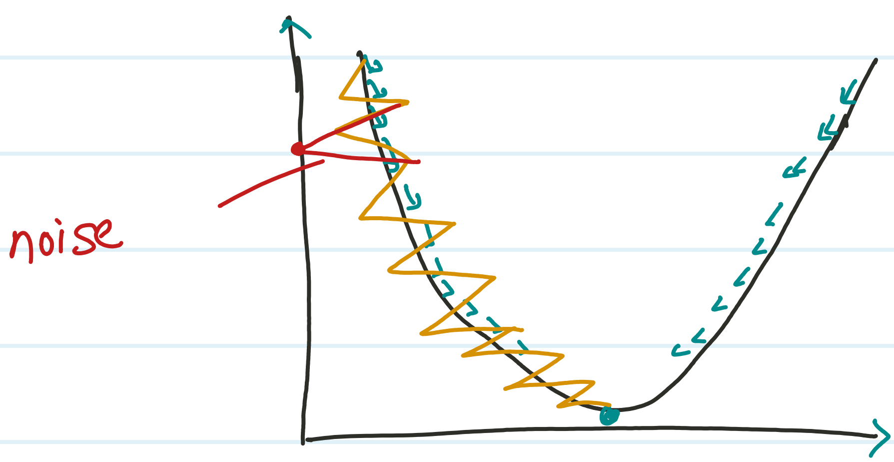
1. Require huge ram and GPU

100 Data — 10000 Data

20% over capacity

2. SGD (Stochastic Gradient Decent)

It use epoch to train batch data as a optimizer



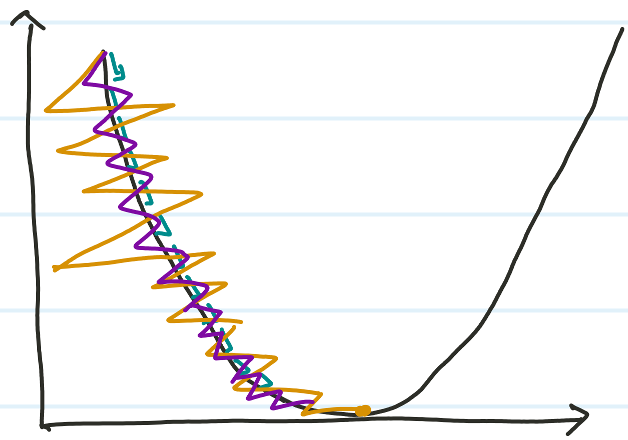
Adv

1. solve Resource issue

Disadv.

1. Convergen will take time
2. noise will get introduce.

③ mini batch SGD



Adv.

① conversion speed will increas
② noise will be less
③ Efficient for resource

Dis adv.

① Noise still exist

④ SGD with momentum

$$w_{new} = w_{old} - \eta \frac{\partial L}{\partial w_{old}}$$

$$b_{new} = b_{old} - \eta \frac{\partial L}{\partial b_{old}}$$

original formal

SGD

$$w_t = w_{t-1} - \eta \frac{\partial L}{\partial w_{t-1}}$$

$$b_t = b_{t-1} - \eta \frac{\partial L}{\partial t_{-1}}$$

✱ Exponetial weight Averuge

suppose

time	t_1	t_2	t_3	t_4	t_5	-------	t_n
value	a_1	a_2	a_3	a_4	a_5	-------	a_n

time value $V_{t_1} = a_1$

$$V_{t_2} = \beta * V_{t_1} + (1-\beta)\, a_2$$

$\beta = 0.95$

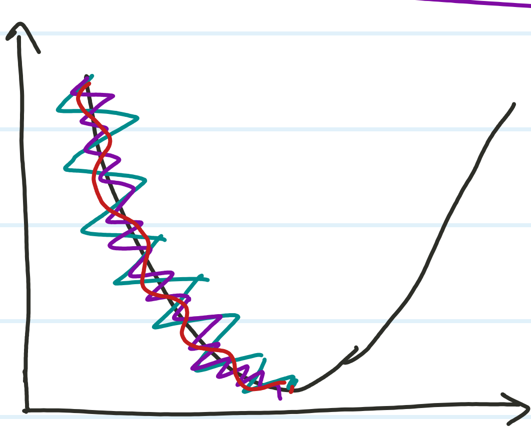
$$= 0.95 * a_1 + (1 - 0.95)\, a_2$$

$$V_{t_3} = \beta * V_{t_2} + (1-\beta)\, a_3$$

Exponetial weight Averaye

$$w_t = w_{t-1} - \eta\, V_{dw}$$

$$\boxed{V_{dw_t} = \beta * V_{dw_t} + (1-\beta) * \frac{\partial L}{\partial w_{t-1}}}$$



Adv.

1. Reduce the noise
2. smoothen the noise
3. Quick conversion
4. working for mini batch

Disadv.

1. We do not have dynamic learning rate

5. Adagrad (Adaptive Gradient Decent)

$$w_t = w_{t-1} - \eta \frac{\partial L}{\partial w_{t-1}}$$

$$w_t = w_{t-1} - \eta' \frac{\partial Loss}{\partial w_{t-1}}$$

$$\eta' = \frac{\eta}{\sqrt{\alpha_t + \epsilon}}$$ denominator should not become zero

$$\alpha_t = \sum_{i=1}^{t} \left(\frac{\partial L}{\partial w_t} \right)^2$$

t is current time stamp

Disadv.

1. Initialy faster conversion after few time become slow.
2. η' = possibility to become small value ≈ 0

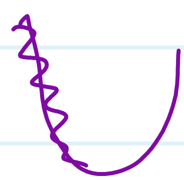
⑥ Adadelta and RMS prop

We are bringing Exponetial weighted Avg. in learning.

⑦ Adam optimizer

This is combination of SGD with momentum and RMS prop

$$w_t = w_{t-1} - \eta' V_{dw}$$



$$V_{dw_t} = \beta * V_{dw_{t-1}} + (1-\beta) \frac{\partial Loss}{\partial w_{t-1}}$$