Optimizers in Neural Network:

NN → trained by — gradient descent

→ local minima.

→ saddle points → Global min ✓

Optimizers → we can atleast try to reach our global minima with more ~~use~~ probability

variants of GD.

(GD equation) → modify →

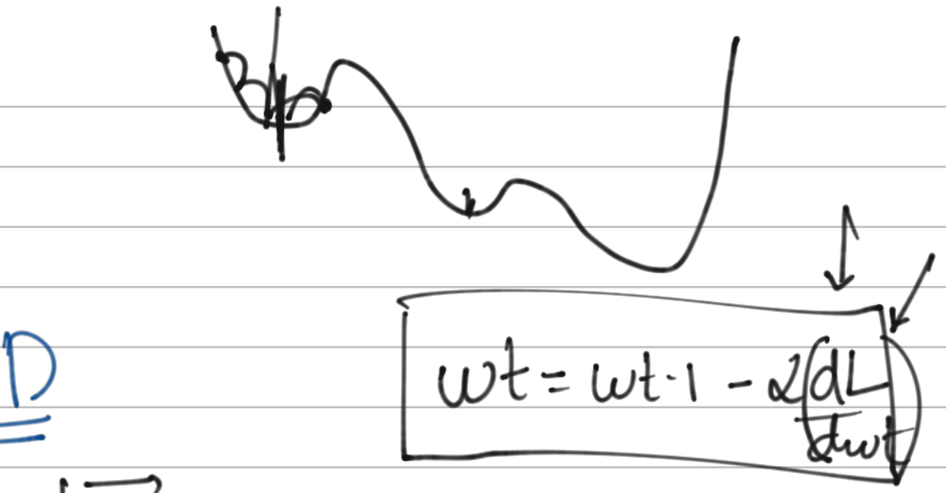
→ Momentum based gradient descent

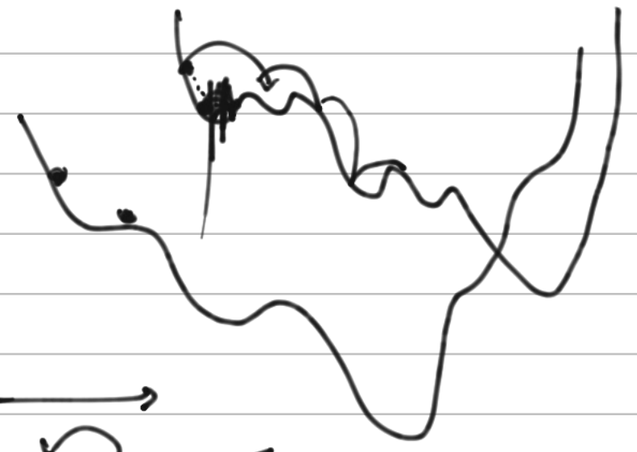
→ Adagrad ✓

→ RmsProp ✓

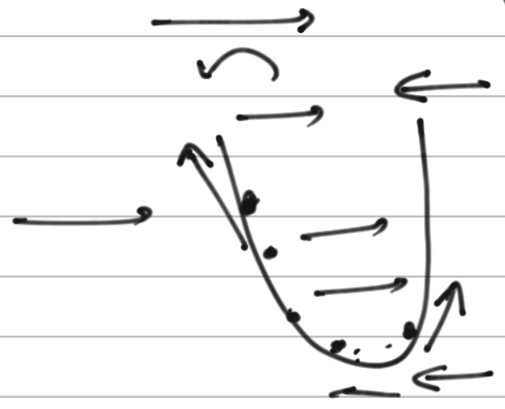
→ (Adam) ✓✓

GD





Momentum Based Gradient Descent: optimizer → better GD



momentum:

GD with momentum

Hyperparameter

Historical + point grad.

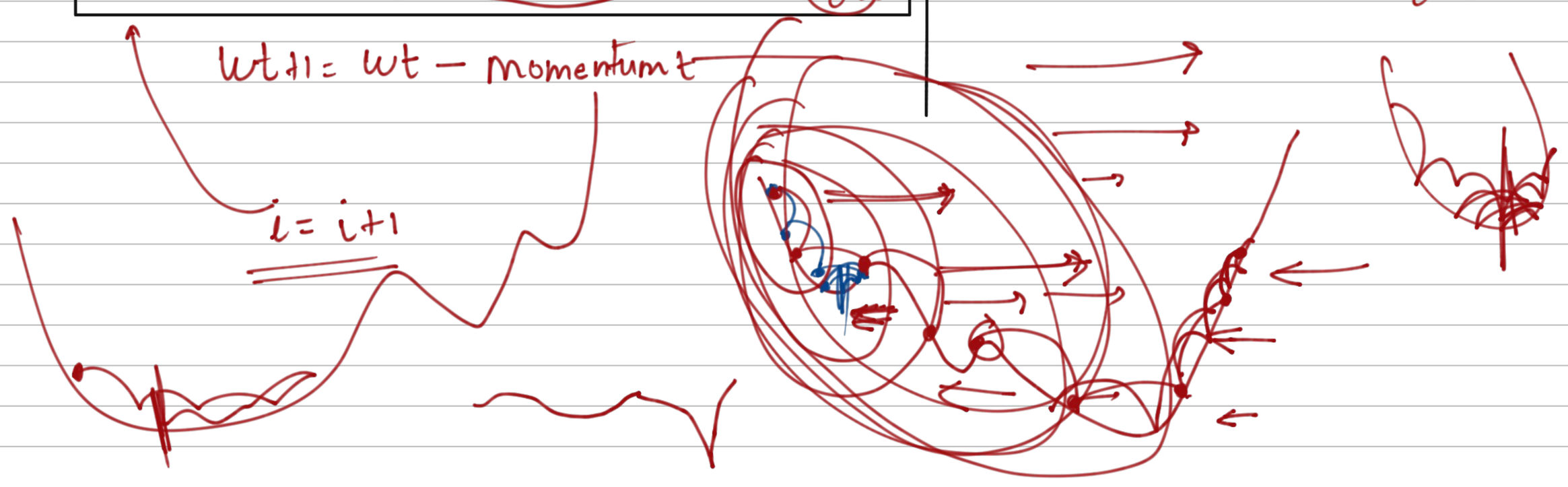
$$momentum_t = \gamma\, momentum_{t-1} + (1-\gamma)\,\alpha\frac{dL}{dw_t}$$

$$w_{t+1} = w_t - momentum_t$$

$i = i+1$

point gradient

$$w_{t+1} = w_t - \alpha\frac{dL}{dw_t}$$



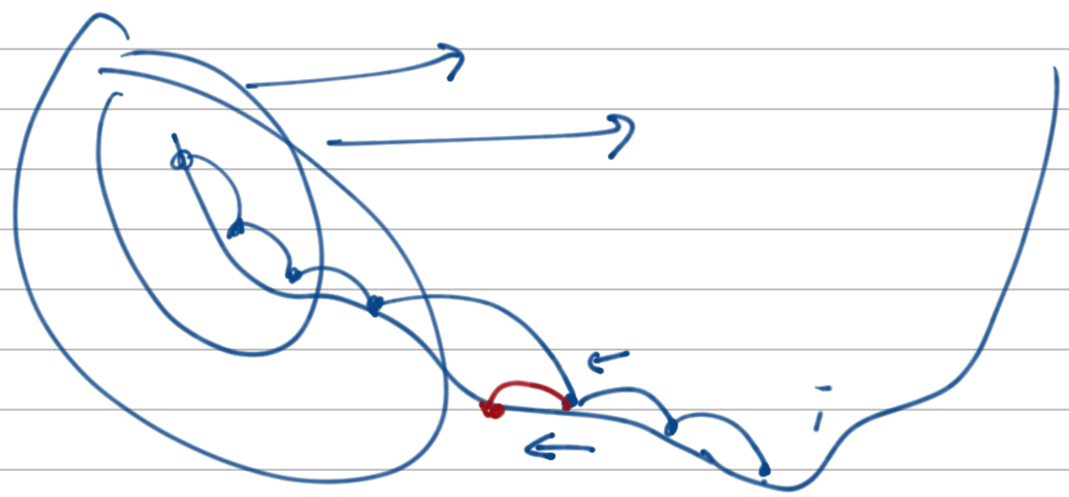
momentum = 0

$$momentum_t = \gamma\, momentum_{t-1} + (1-\gamma)\,\alpha \frac{dL}{dw_t}$$

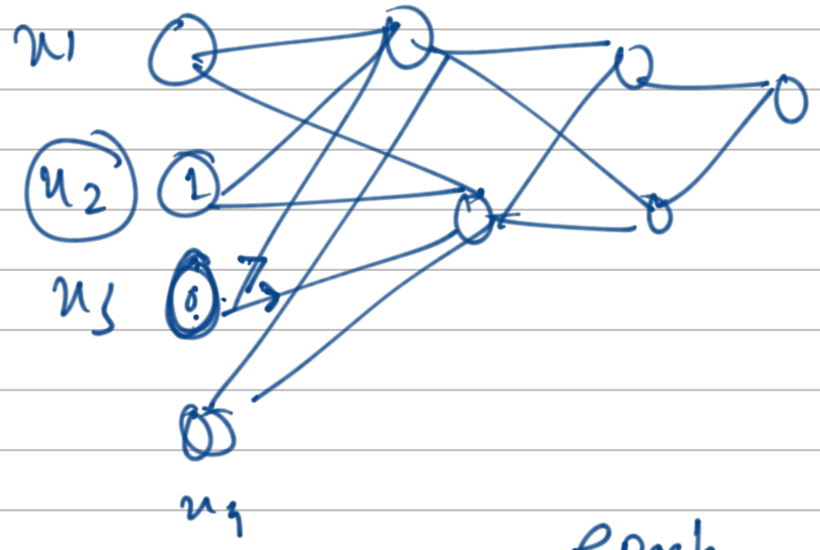
$\frac{dL}{dw_t}$ → point gradient

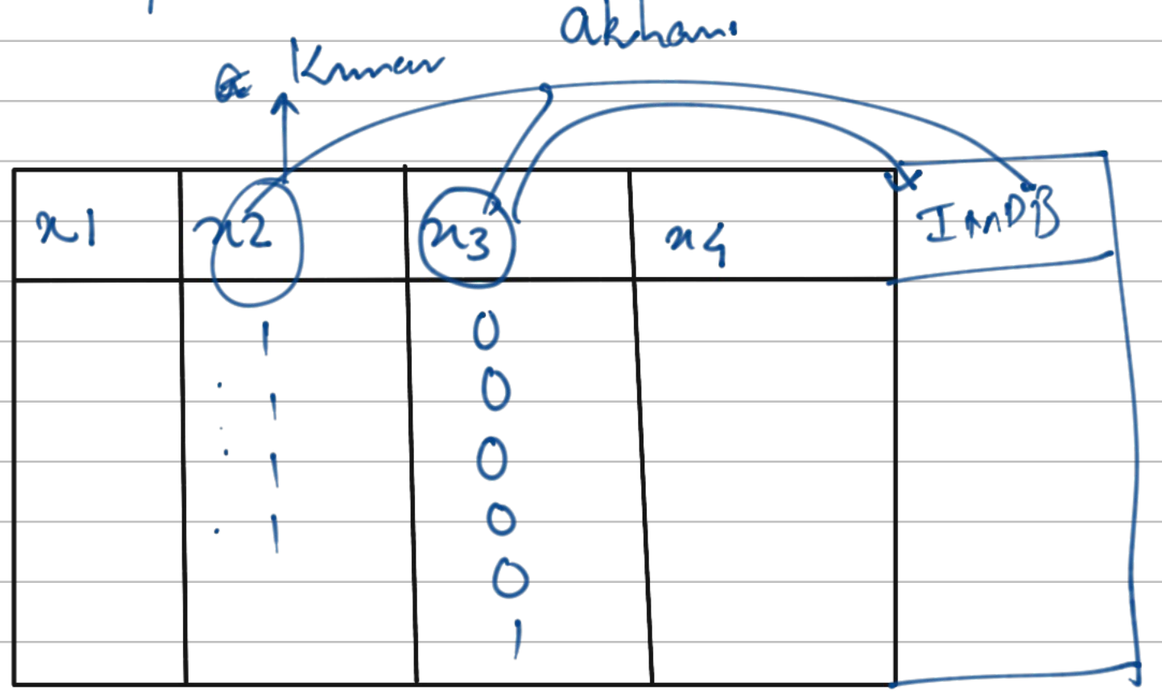
momentum → Accumulator

$$w_{t+1} = w_t - (momentum_t)$$



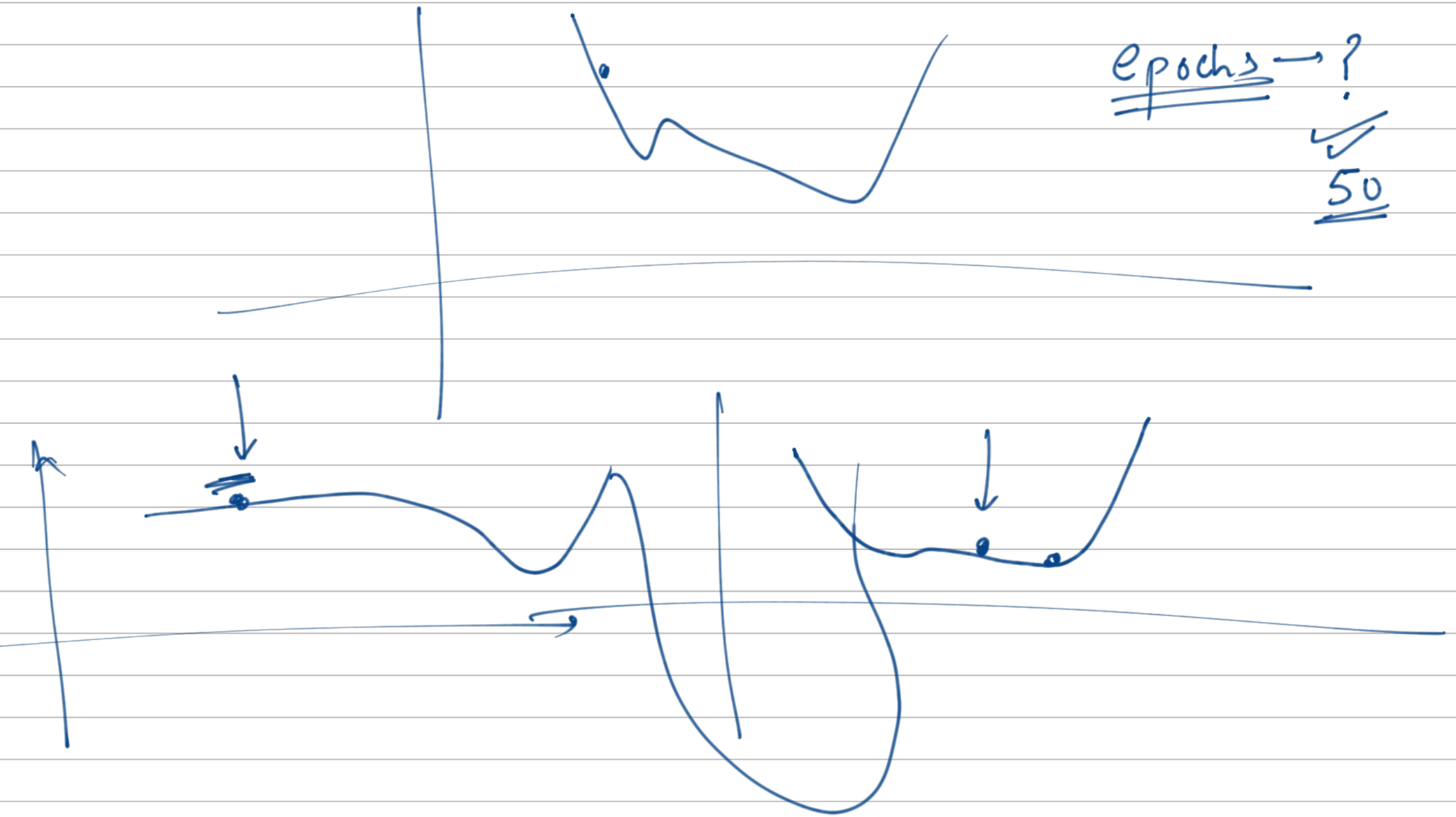
Adagrad → Adaptive Gradient → optimizer



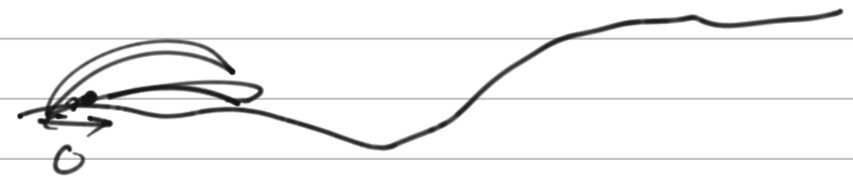


epoch — weights associated to x_3 will not have equal opportunity to get trained as much as x_2

All weights are randomly initialized.


epochs → ?
50

Adagrad:



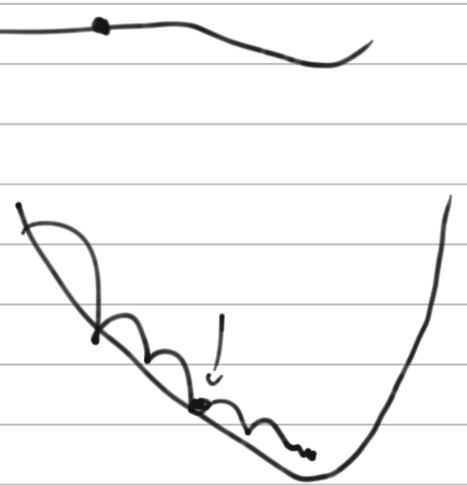
$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{v_t + \varepsilon}} \times \frac{dL}{dw_t}$$

$v_0 = 0$

$$v_t = v_{t-1} + \left(\frac{dL}{dw_t}\right)^2$$

Square

People/wts who have done very big steps



RmsProp:

$$v_t = \beta v_{t-1} + (1-\beta)\left(\frac{\partial h}{\partial w_t}\right)^2$$

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{v_t + \varepsilon}} \times \frac{\partial h}{\partial w_t}$$

→ momentum - ?

Combination of RmsProp + momentum → Adam

Adaptive moments

Adam → Adaptive moments

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) \times \frac{\partial L}{\partial w_t}$$

momentum

Velocity

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) \times \left(\frac{\partial L}{\partial w_t}\right)^2$$

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{v_t + \varepsilon}} \times m_t$$

$\beta_1 = 0.9$ $\beta_2 = 0.999$

ohe = onehotencoder()

ohe.fit_transform

model (Training)

onehot encoder

x_1	x_2_A	x_2_B	x_2_C	x_2D
	A			
	B			
	C			
	D			
	A			

Onehotencoder.

production

model.Predict( )

x_1	x_2
	A
	A

error.

ohe.transform

x_1	x_2A	x_2B	x_2C	x_2D
	1			