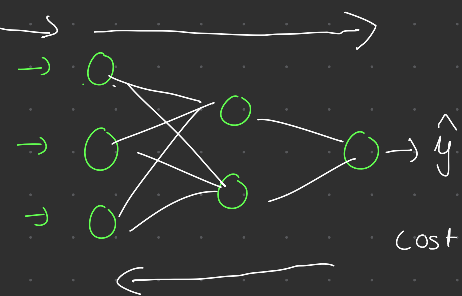
Optimizers

1. Batch Gradient Descent (a.k.a. "Vanilla" Gradient Descent)

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

- You send all 1 lakh rows at once.
- Compute cKerasunction on the entire dataset.
- Do 1 weight update per epoch.
- If you run 100 epochs → 100 weight updates in total.

100000 Rows



Advantages

- Smooth and stable updates (less noisy).
- Converges steadily because gradient is calculated from all data.

Disadvantages

- Slow when dataset is huge (computing gradients for all rows before updating).
- Requires a lot of RAM/VRAM to load the full dataset at once.

100 epochs → updating the weights 100 times

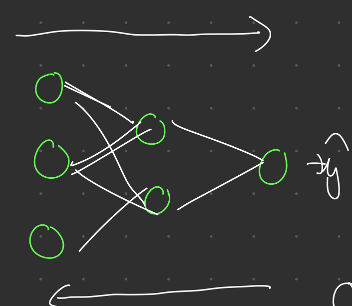
2. Stochastic Gradient Descent (SGD)

100000 Rows

1 Row

100 epoch

- You send 1 row at a time.
- Compute cost for that single row.
- Update weights immediately.
- With 1 lakh rows, in 1 epoch → 1 lakh updates.
- In 100 epochs → 1 crore updates.



= Update weight

100×100000
= 1 crore UP

2 Row
3 Row
4 Row
|
⋮
100000 Rows

Advantages

- Much faster to start learning (weights update after every row).
- Works well for very large datasets (you don't need all rows in memory at once).

Disadvantages

- Updates are very noisy → loss curve jumps around instead of smoothly decreasing.
- Can be unstable (harder to converge).
- Slower to reach the exact minimum compared to mini-batch.

3. The Practical Solution: Mini-Batch Gradient Descent

100000 ⇒ 100 Batch size

100 epoch

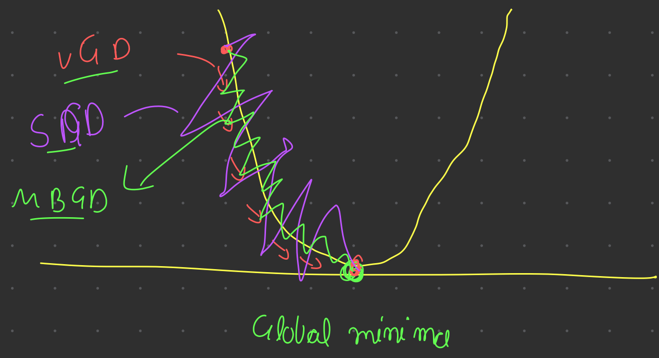
- A compromise between batch and SGD.
- Split data into small batches (e.g., 32, 64, 128 rows).
- Each batch → forward pass → backward pass → weight update.
- So: in 1 lakh rows, batch size 100 → 1000 updates per epoch.

$\hat{y}$ } iteration - 1

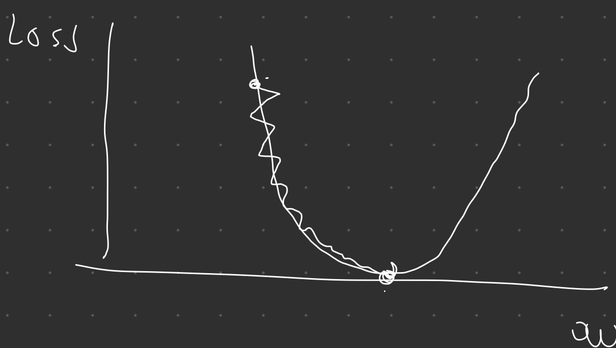
This is what almost all deep learning frameworks use today (including Keras/TensorFlow).

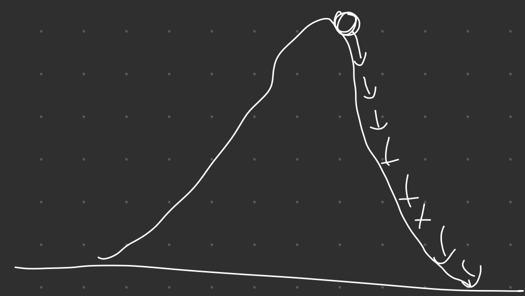
$$\frac{100000}{100} = 1000$$

$$100 \times 1000 = 100000$$



momentum with SGD





Momentum = $\boxed{\text{mass}}$ x velocity

$$W_{new} = W_{old} - \eta \boxed{\frac{\partial L}{\partial W_{old}}} \Rightarrow W_{T+1} = W_T - \eta \Delta W$$

$$W_{T+1} = W_T - \boxed{V_t}$$

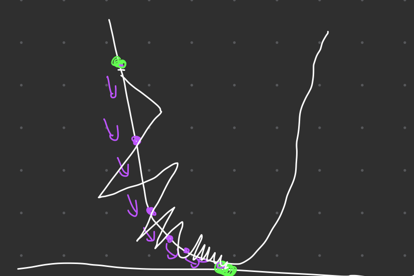
$\boxed{0.09}$

$$V_t = \beta \times V_{t-1} + \eta \Delta W_t$$

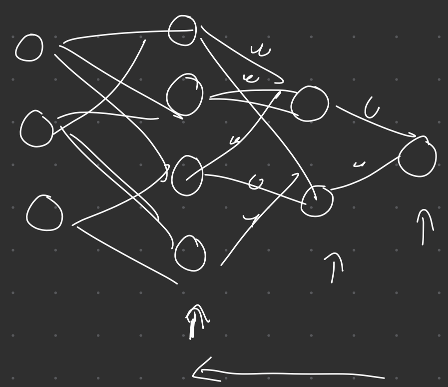
Adagrad (Adaptive Gradient descent)

$$W_T = W_{T-1} - \boxed{\eta} \frac{\partial L}{\partial W_{T-1}} \longrightarrow$$

0.01
constant



$$w_T = w_{T-1} - \boxed{\eta'} \frac{\partial L}{\partial w_{T-1}}$$



$$\eta' = \frac{\eta}{\sqrt{\boxed{\alpha_T} + \epsilon}}$$

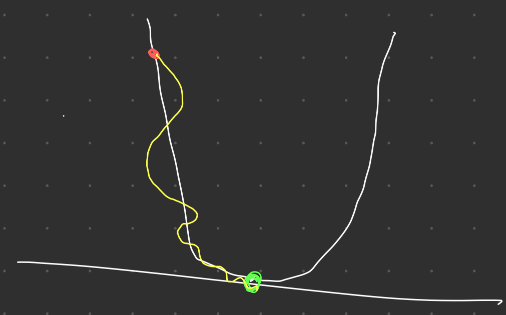
Epsilon

$$\alpha_T = \sum_{i=1}^{t} \left(\frac{\partial L}{\partial w_T}\right)^2 \Rightarrow$$

$0.01 \rightarrow 0.002 \rightarrow 0.001$

Adam optimizer

↳ Adagrad + Momentum



Decision Tree → white box model

Random Forest → Black box model

ANN → Black box model

LR - white box model