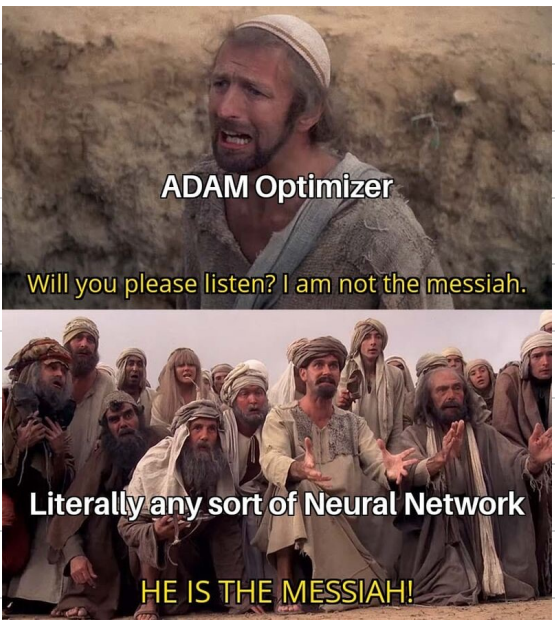
Session-8

NN- Optimizers

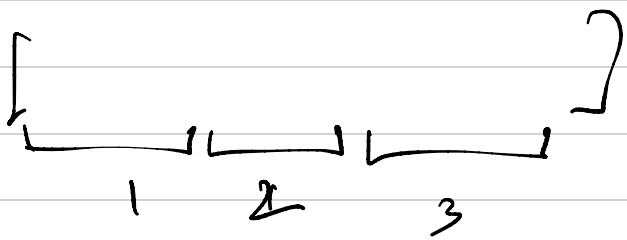
Feb 23, 2024



Agenda

1. Some problems with SGD
2. EMA
3. RMS Prop
4. Adam

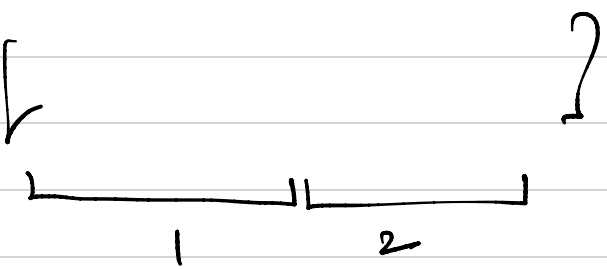
How SGD works



GPT → 4096

32

3.2 M



$$w_{ij}^{k} = w_{ij}^{k} - \eta \sum_{i=1}^{n} \frac{\partial L}{\partial w_{ij}^{k}}$$

n → no. of data points

w_{ij}^{k}

k → Layer

i → to Layer

j → From Layer

w_{34}^{2} → 2nd layer, 3rd neuron → 4th neuron

→ Stochastic Gradient descent

1 single sample (update weights)

→ Batch / Mini-batch G. descent (You take avg of gradients → beta)

a batch of dataset (3.2 M GPT)

32 - 64 - 128

(Replacement - not recommended)

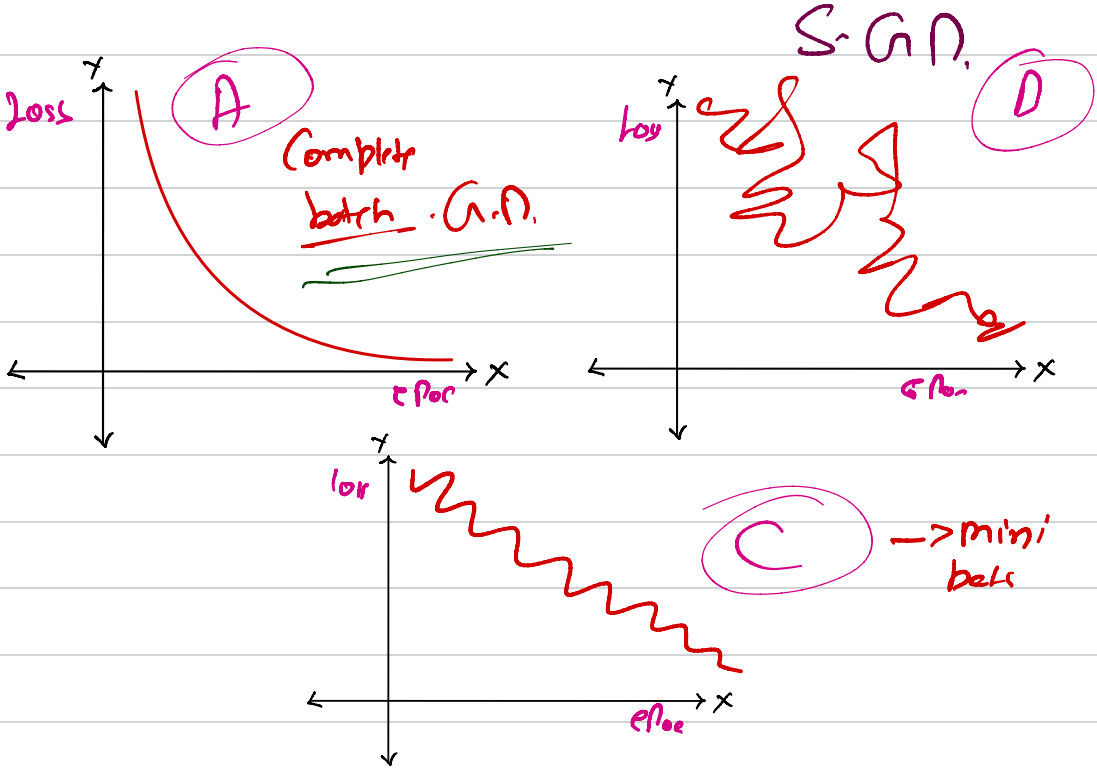
1MB = 1024 Kb
1GB = 1024 MB

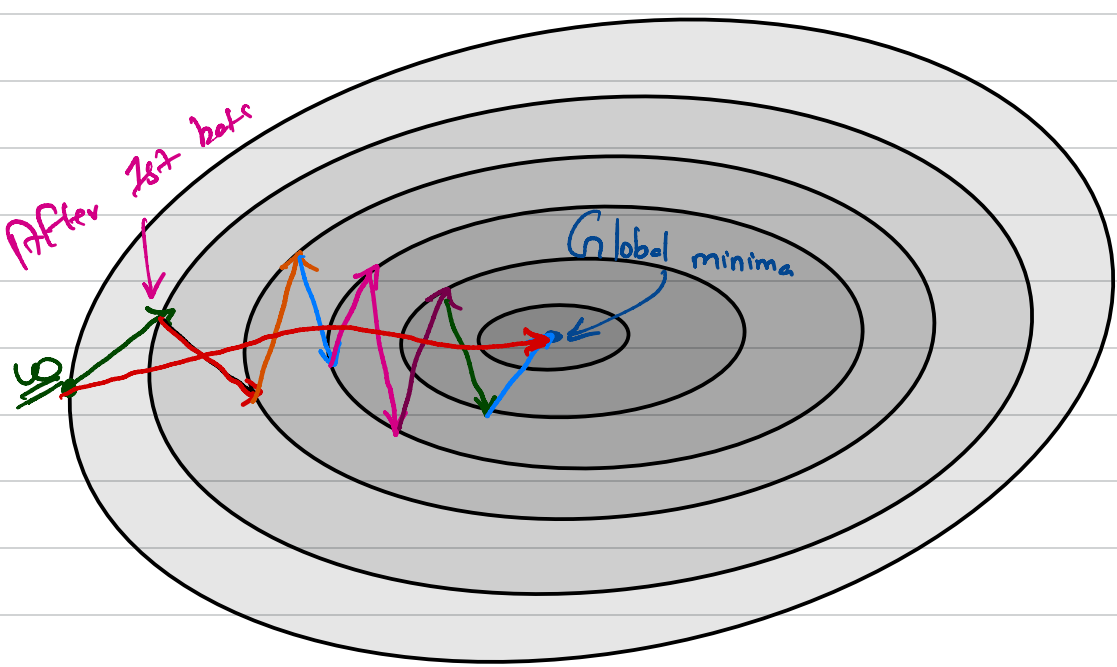
only handle amount of → 2^n

(

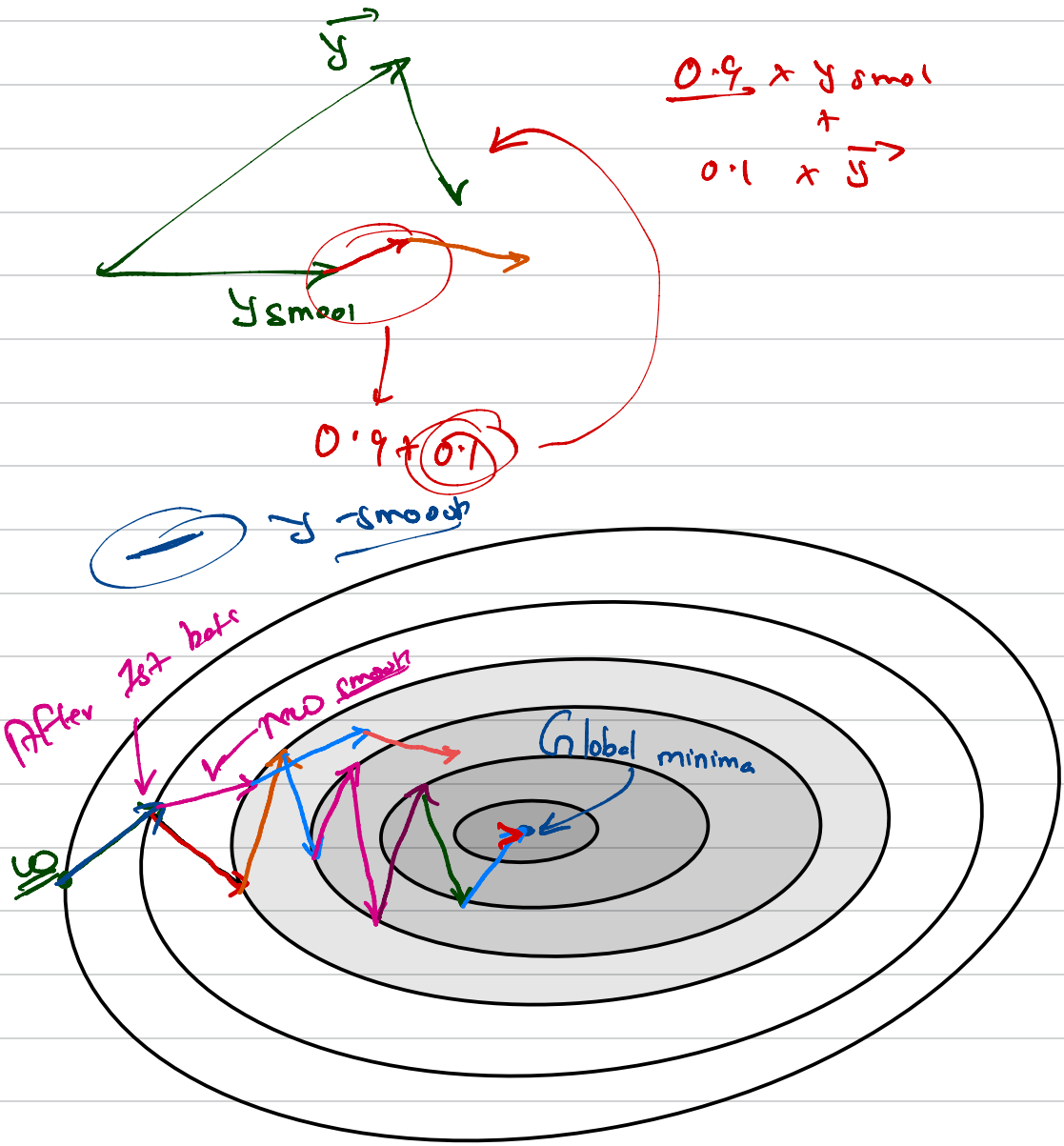
10 (GPU → 16 Samples

Using → 10 samples

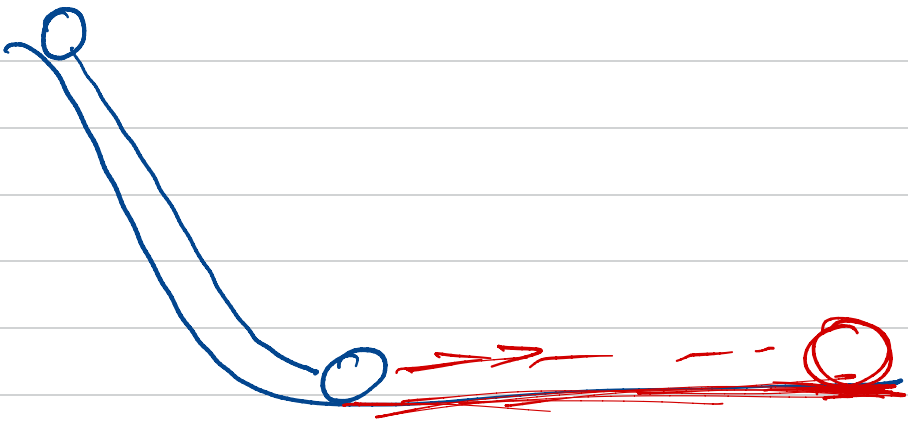

S.G.D.
Loss
A
Complete
batch G.D.
epoc
loy
D
epo
lor
C
->mini
bats
epoc

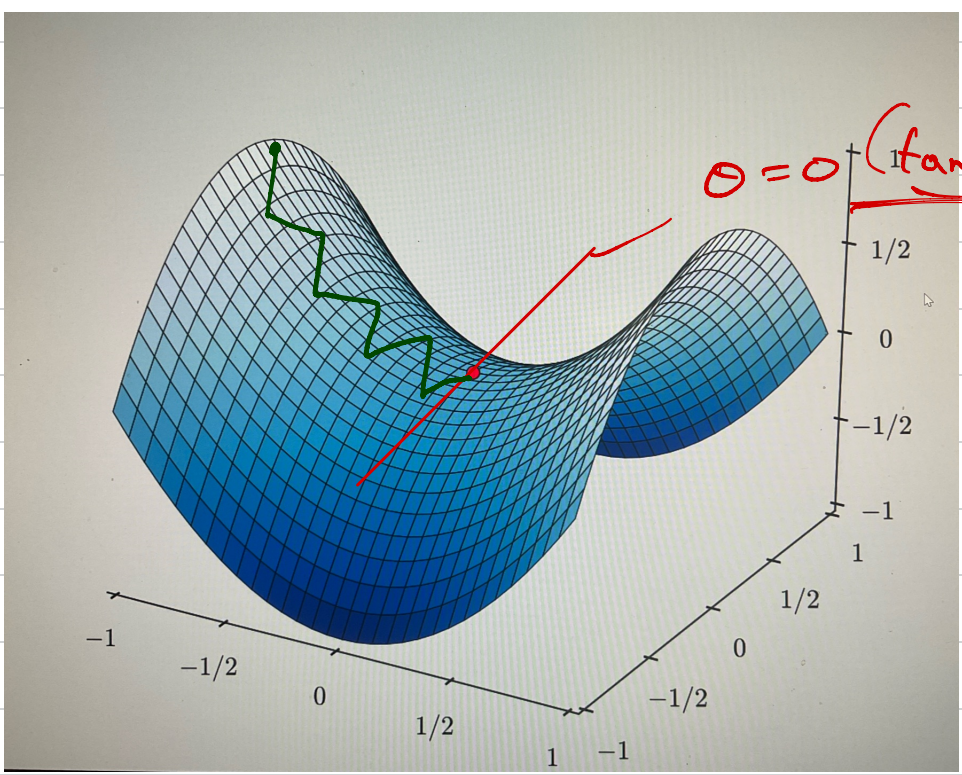

After
1st bats
Global minima

Exponential moving average



Momentum





When using EMA

$\frac{\partial L}{\partial w_{ij}^{k}}$ → at time step t → g^t

$$w^{t+1} = w^t - \eta \times g^t$$

In EMA

$$g^t = \beta \times g^{t-1} + (1-\beta) \times \frac{\partial L}{\partial w}$$

β → 0.9

g^{t-1} → gradient of prev step $t-1$

$\frac{\partial L}{\partial w}$ → batch current step

How momentum helps

g^0

$v_0 = 0$, $\beta = 0.9$, $w_0 = 0.1$

$\frac{\partial L}{\partial w}$

v_1 (θ^1) $= \beta \times v_0 + (1-\beta) \times w_0$

$= 0.9 \times 0 + (1-0.9) \times 0.1 = 0.01$

$g^{t-1} = 0.04$

$\beta = 0.9$

$g^t = 0.06$

$$g_t = \beta \times g^{t-1} + (1-\beta) \times \frac{\partial L}{\partial w}$$

$$g^t = \frac{\partial L}{\partial w} \quad (t=0)$$

$$0.06 = 0.9 \times 0.04 + (1-0.9) \times \Delta w$$

$$= 0.2 \Delta w$$

$$g^{t+1} = \frac{\partial L}{\partial w} \quad (t=1)$$

90% × g^t + 10% → g^{t+1}

0.06

$$w = w - \eta \times \frac{\partial L}{\partial w}$$

→ gradient

→ gradient at $t=0$

gradient at $t=1$ = $\beta \times g^{t-1} + (1-\beta) \frac{\partial L}{\partial w}$

↓ Gradient step

↓ Vanilla gr

$$v = u + at$$


SGD
SGD WITH MOMENTUM
imgflip.com

RMS PROP


t↑ α↓
gt →
g1
g2
g3
g4
Lowest loss

$$\Rightarrow \frac{\eta}{\sqrt{\sum_{i=1}^{t} g_t^2 + \epsilon}} \longrightarrow 10^{-8}$$

as time inc

$$\omega_{new} = \omega_{old} - \frac{\eta}{\sqrt{\sum_{i=1}^{n} g_t^2 + \epsilon}} \times \frac{\partial L}{\partial \omega}$$

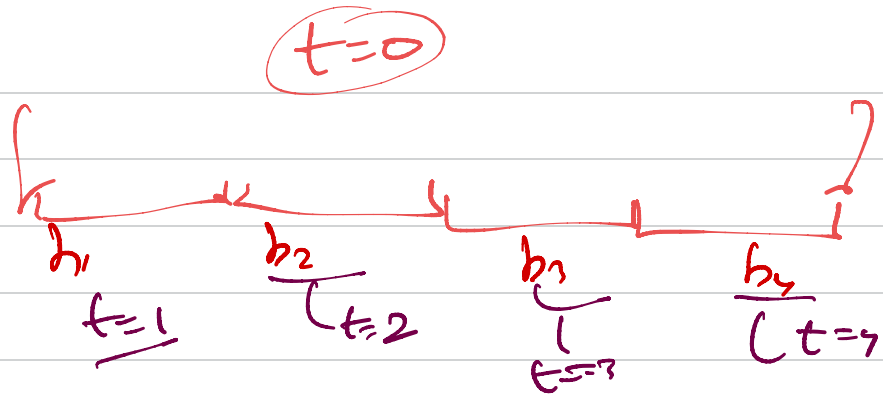
$$\rightarrow \quad t \uparrow \Rightarrow \left(\sum_{i=1}^{n} g_i^2 \right) \longrightarrow \alpha \downarrow$$

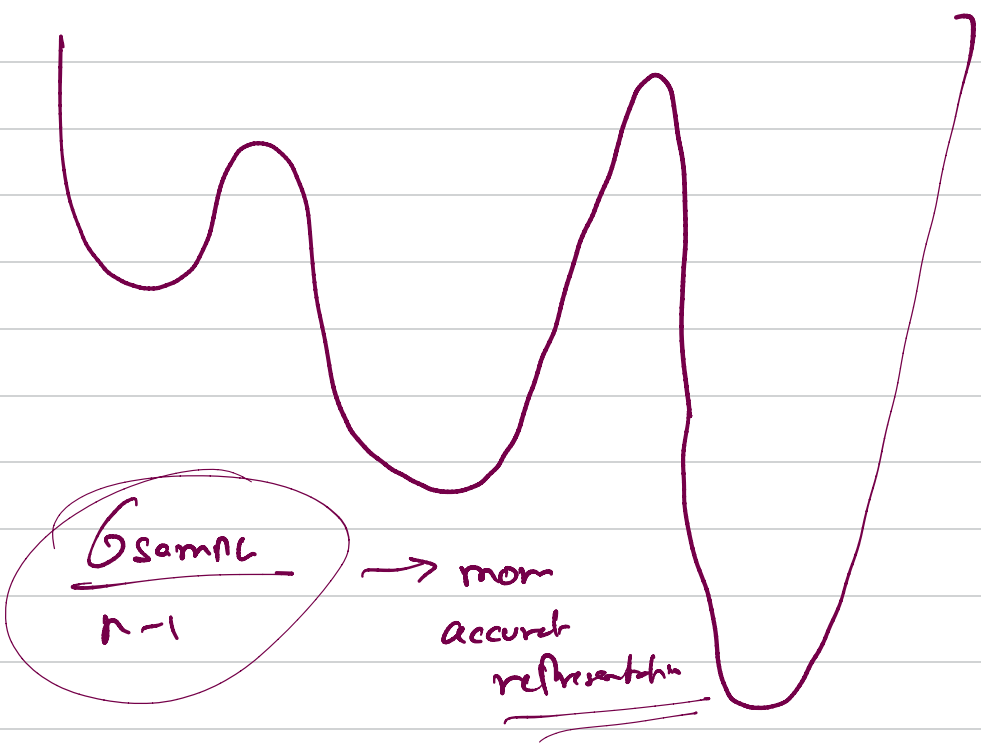
Adam

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize → Learning Rate
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ → loss function
Require: θ_0: Initial parameter vector → weight Param
$m_0 \leftarrow 0$ (Initialize 1st moment vector) → momentum vector
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep) → learning rate
while θ_t not converged **do**
  $t \leftarrow t + 1$
  $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t) → $\frac{\partial L}{\partial w_t}$
  $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
  $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
  $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
  $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
  $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

$g_t = \beta \times g_{t-1} + (1-\beta) \frac{\partial L}{\partial w_t}$


t=0
h1
h2
h3
h4
t=1
t=2
t=3
t=4


G sample
n-1
→ more
accurate
representation

train → Standard

$\frac{x - \mu}{\sigma}$ → Z-score

fit → train → μ & σ

transform test