# Gradient Descent

Wednesday, 19 February 2025 11:05

Optimization algo used to find $min^m$ of a $func^n$. → $f(x)$

$$x_{new} = x_{old} - \underbrace{\alpha}_{\text{Learning rate (step size)}} \vec{\nabla} f(x_{old})$$

Epoch = One full pass over the entire dataset.
Batch = A small subset of dataset used in one step of training
Iteration = One update of the model using a batch.

ex:- 1000 samples ; batch size = 100

1 epoch = 10 iterations ( 1000 / 100 = 10)

| | Batch GD | Stochastic GD | Mini-batch GD |
|---|---|---|---|
| Batch Size = | All data | 1 sample | Small batch |
| Updates per epoch = | 1 update per epoch | Many updates | Medium updates |
| Speed = | Slow | Fast | Balanced |
| Stability = | High | Low (noisy) | Balanced |

$$1 \text{ epoch} = \frac{\text{dataset size}}{\text{batch size}}$$

https://colab.research.google.com/drive/1PKOoRT76HTA-5WrbDY6jAiH87c5KVw_t?authuser=2#scrollTo=q3zASsyttNsN (For code and visualization purpose)

→ Code of all 5 optimizers, we studied.

# SGD with Momentum

Wednesday, 19 February 2025 11:30

Why we need momentum?

ⓐ non-convex cost funcⁿ.

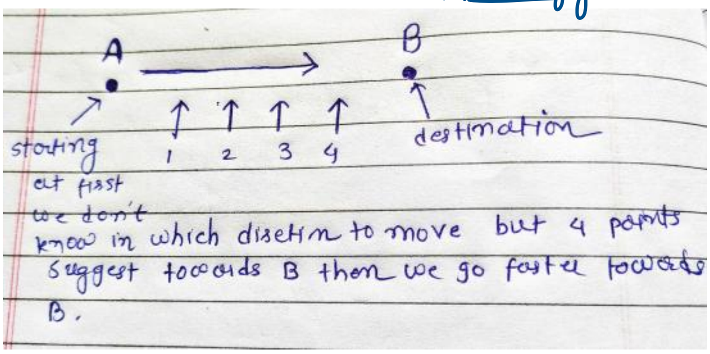
ⓑ Saddle pts

A saddle point is a point where in one direction the surface goes in the upward direction and in another direction it goes downwards

From <https://paperswithcode.com/method/sgd-with-momentum>

ⓒ slow performance
‖
Speed + Stability

# Analogy




EWMA (Expo Wt Moving Avg)

↳ Tech used to find trends in time-series data.

"velocity" $\boxed{v_t = \beta v_{t-1} + (1-\beta)\theta_t}$

$\beta$ ↓ wt assigned to past value

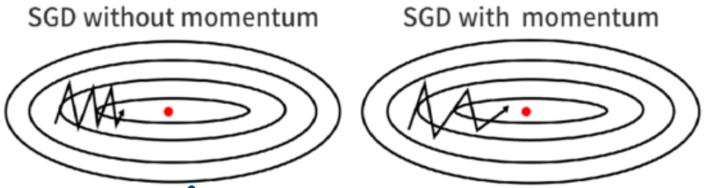
$0 < \beta < 1$

ex:- if $\beta = 0.5$ then $\frac{1}{1-\beta} = 2$
⇓
calc. avg was from previous 2 readings.

High $\beta$ ⇒ Avg of more past data.

where,

$$\boxed{\begin{aligned} w_{t+1} &= w_t - v_t \\ v_t &= \underbrace{\beta}_{\text{Decaying factor}} v_{t-1} + \eta \Delta w_t \end{aligned}}$$




$\beta = 0$

# Advantages :

1. Faster convergence.
2. Escaping local minima (goal → global minima)

# Adagrad adapts the learning rate individually for each parameter based on past gradients.

Wednesday, 19 February 2025 11:57

Let, $\epsilon = 10^{-8}$ (to avoid div. by zero)

Let $G_0 = 0$

Sum of squared gradients : $\boxed{G_t = G_{t-1} + g_t^2}$

$g_t^2$ : element-wise sq. of grad vec

$$\boxed{\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{G_t + \epsilon}} g_t}$$

$\eta$ : Initial learning rate

$\sqrt{G_t + \epsilon}$ : Adaptive term that scales the learning rate.

* Large gradients ⇒ small l.r.
* Small " ⇒ large l.r.
* Good for sparse data.

# RMSProp

Wednesday, 19 February 2025 11:54

Root Mean Square Propagation

↳ Improvement over AdaGrad

Instead of accumulating all past squared gradients (which causes the learning rate to decay too much), RMSProp uses a moving average of squared gradients. This makes it more suitable for **non-convex optimization problem**s, such as training deep neural networks.

expo dec. avg $\Rightarrow$ More wt. to recent gradients.

Initialize $\theta_0$ ; $E[g^2]_0 = 0$ "Moving avg of sq gradients"

$\gamma \rightarrow$ expo decay factor (Typically, $\gamma \simeq 0.9$)

$$\boxed{\begin{aligned} E[g^2]_t &= \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2 \\ \theta_t &= \theta_{t-1} - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \end{aligned}}$$

# ADAM = Mom + RMSprop

Wednesday, 19 February 2025 12:01

(Adaptive Moment Estimation)

* ADAM uses 2 moving averages :⇒
  - ⓐ mom term ($m_t$)
  - ⓑ Adaptive scaling ($v_t$)

$g_t \rightarrow$ gradient

$$\boxed{\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\ \theta_t &= \theta_{t-1} - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \end{aligned}}$$

Bias correction :⇒

→ since $m_0$ & $v_0 = 0$ ⇒ biased towards smaller values in early steps.

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

⇓ Too small updates in beginning.

$\eta \sim 0.001$ ; $\beta_1 \sim 0.9$ ; $\beta_2 \sim 0.999$

$\epsilon \sim 1e-8$