# Understanding Optimization of ALGORITHMS

Let us quantify the quality of any particular set of weights W. The goal of optimization is to find "W" such that it minimize the loss function.

① finding set of param i.e W corresponding to global minima.

Strategy 01: A first VERY BAD idea try out many different values randomly. keep the track of the best. Procedure look like below

```
# assume X_train (each column) an example (3073 x 50,000)
# assume Y_train labels (1D array of 50,000)
# L evaluat my loss function

bestloss = float("inf")
for num in range(1000):
    W = np.random.randn(....)
    loss = L(X_train, Y_train, W)
    if loss < bestLoss
        bestLoss = loss
        bestW = W
```

This approach given 15% accuracy, where as completely Random will give ~10 % accuraly.

## Strategy #2 := (Random Local Search)

Try to take small step in random direction, take further step only if cost is getting lesser.

```
W = np.random(---)
bestloss = float('inf')
step_size = .0001
for(x=1 → 1000)
    Wtry = W + np.random(--) * step_size
    loss = L(X_tra, Y_tra, Wtry)
    if loss < bestloss:
        W = Wtry
        bestloss = loss
```

classification accuracy ≈ 21.4%.

## Strategy 03:- Stop guessing direction use "Gradient"

* The gradient tells us direction in which function has the steepest "Rate of Increase". ~~So to avoid~~ ~~more in opposite~~ (so -ve direction)
* Gradient tells only the direction not the step size.

Gradient Descent:- Compute gradient loss of the loss function. perform the parameter update.

```
while True:
    weights_grad = evaluate_grad(loss_fun, data, weights)
    weights += - step_size * weighs_grad
```

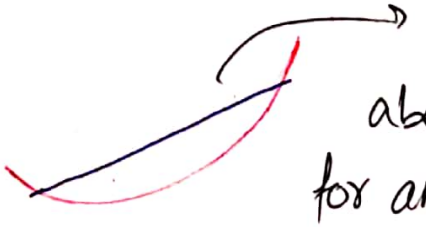
* Now it's clear step_size = learning_rate.

Mini_Batch_Gradient_Descent:- Perform the calculation over small batches rather than complete dataset.

```
while True:
    data_batch = sample_training_data(data, 256)   # 256 = batch_sz
    weights_grad = evaluate_gradient(loss_fn, data_batch, wts)
    weights += - weights_gradient * step_size.
```

challenges for plain Mini_Batch_Gradient Descent

Convex function:-



Line should above the curve for any two points on graph.

Saddle point:- Points where one dimension slopes up & another goes down. On these points Gradient is almost ≈ 0.

- Refer: blog.paperspace.com/intor to optimi -zation in deep learning - gradien descent.

- Do visit :- Complicated landscaped [cs.umd.edu/n tomg/projects/landscapes].

→ Surrounded by a plateau.

Plateau :- Also called high plain or tableland, is an area of highland.

Start Page

To Do

## Challenges Continue:-

Same learning rate applies to all parameter updates. If data is sparse & features have different frequencies, we might not want to update all of them with same extent. perform larger update to rarely occuring features.

**Gradient Descent + Momentum :-** The momentum term increases for dimensions whose gradients point in the same direction & reduces updates for dimensions whose gradient direction changes. As a result we converge faster & reduced oscillation. Lets take it by an example Visualize below loss function.

↓ what we want is

↕ slow learning rate VERTICAL

↔ fast learning rate in HORIZONTAL.

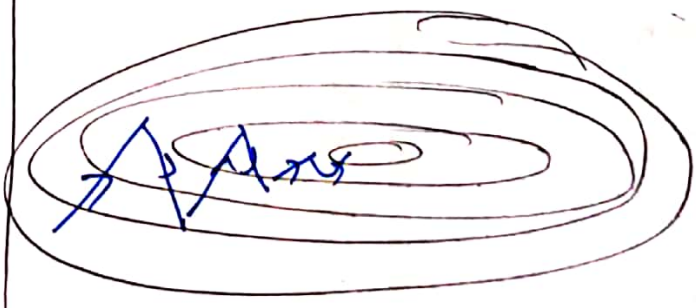
(P.T.O)

So $V_t = \gamma * V_{(t-1)} + \eta \nabla_\theta J(\theta)$
(Param Smooth)

$\theta = \theta - V_t$ ← (ParamUpdate) $\gamma = 0.9$

So by averaging over last iterations (gradients in the vertical direction will be ≈ 0 for prev example) ((bcaz in ONE itr it will go up (say +ve gradient in other go down -ve gradient hence averagin it will ≈ 0)) for vertical.
Where as in horizontal gradient is almost always (+ve) hence oscillation will reduce in that direction it will go faster horizontally with less oscillation.

Before Momentum



After Momentum.

$\gamma = 0.9$ most of the cases.

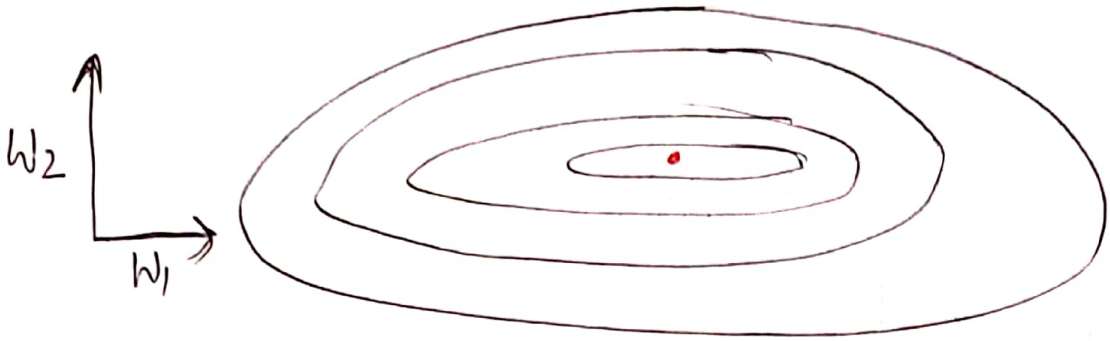
Adagrad :- It adapt learning rate to the parameters, performing larger updates fo infrequent parameters & smaller updates for frequent parameters. This is well suited for to deel with SPARSE DATA.

$g_{t,i} = \nabla_{\theta_t} J(\theta_{t,i})$ → gradient of objective function w.r.t to the parameter $\theta_i$ at the time step t:

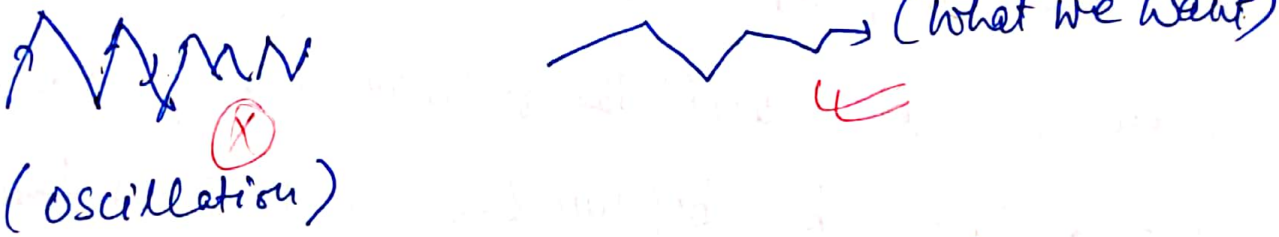
$$\theta_{t+1,i} = \theta_{t,i} - \eta \cdot g_{t,i}$$

P.T.O

# RMSProp




Assume above is our loss function & we want to make learning faster in ($W_1$) direction ($W_2$) slow. to avoid lot of oscillation.




(oscillation)

On iteration 't' will compute $dW_1$ $dW_2$ on mini batch

$$S^t_{dW1} = \beta S^{t-1}_{dW1} + (1-\beta)\, dW_1^2 \quad \text{(element wise square)}$$

$$S^t_{dW2} = \beta S^{t-1}_{dW2} + (1-\beta)\, dW_2^2$$

$$W1 = W1 - \alpha \frac{dW1}{\sqrt{S_{dW1} + \varepsilon}}$$

$$W2 = W2 - \alpha \frac{dW2}{\sqrt{S_{dW2} + \varepsilon}}$$

$\varepsilon \approx$ very small number to avoid divide by ZERO $\approx 10^{-8}$

## Intution why it works ?—

In horizontal direction we want fast learning, vertical we want slow down.

So we $dw_1^2$ will relatively small $\sqrt{s}dw_1$ where as $dw_2^2$ will be relatively big $\sqrt{s}dw_2$ by dividing by big number we slow down in vertical direction($w_2$). Hinton suggested $\beta = 0.9$ good default value for learning rate = .001.

# Adam Optimization

It is a combination of (Gradient Descent with Momentum + RMS Prop) Algorithm.

<Steps>

1. $V_{dW1}, S_{dW1}, V_{dW2}, S_{dW2} = 0, 0, 0, 0$

On iteration 'T' compute $dW_1$ & $dW_2$ with current mini-batch.

$$V_{dW1} = \beta_1 \times V_{dW1} + (1-\beta_1) \times dW_1$$
$$V_{dW2} = \beta_1 \times V_{dW2} + (1-\beta_1) \times dW_2$$

} Momentum like update

$$S_{dW1} = \beta_2 \times S_{dW1} + (1-\beta_2) dW_1^2$$
$$S_{dW2} = \beta_2 \times S_{dW2} + (1-\beta_2) dW_2^2$$

} RMS Prop like Update

Bias Correction

$$V_{dW1}^{Corrected} = V_{dW1} / (1-\beta_1^t)$$
$$V_{dW2}^{Corrected} = V_{dW2} / (1-\beta_1^t)$$
$$S_{dW1}^{Corrected} = S_{dW1} / (1-\beta_2^t)$$
$$S_{dW2}^{Corrt} = S_{dW2} / (1-\beta_2^t)$$

P.T.O

$$W1 = W_1 - \alpha * \frac{V_{dw_1}^{Corrected}}{\sqrt{S_{dw_1}^{Corrected}} + \varepsilon}$$

$$W_2 = W_2 - \alpha * \frac{V_{dw_2}^{Corrected}}{\sqrt{S_{dw_2}^{Corrected}} + \varepsilon}$$

$\alpha$ = learning rate

$\beta_1 = .9$

$\beta_2 = .99$

$\varepsilon = 10^{-8}$

//

# RADAM

- Recently many new methods have been proposed to accelerate optimization by applying "adaptive learning rate". Example ADAGRAD (2011), RMSProp
- RADAM proposed new variant of ADAM. which explicitly rectifies the variance of the "adaptive learning rate" based on derivation.
- Contribution:- Convergence issue is due to undesirable big variance of the adaptive learning rate in the early stage of model training.
- Proposing RADAM as solution.

$$\phi(g_1, g_2 \dots g_t) = \frac{(1-\beta_1)\sum_{i=1}^{t} \beta_1^{t-i} g_i}{1-\beta_1^t}$$

[$\phi$ momentum type] update.

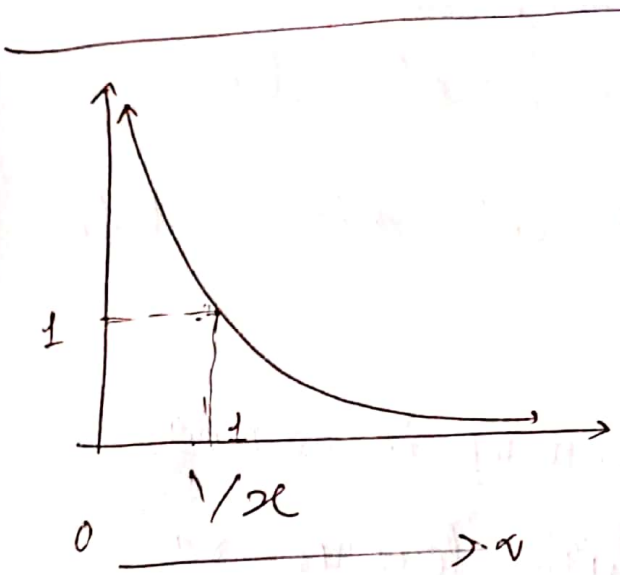
$$\psi(g_1 g_2 \dots g_t) = \sqrt{\frac{1-\beta_2^t}{\varepsilon + \sqrt{(1-\beta_2)\left(\sum_{i=1}^{t} \beta_2^{t-i} g_i^2\right)}}}$$

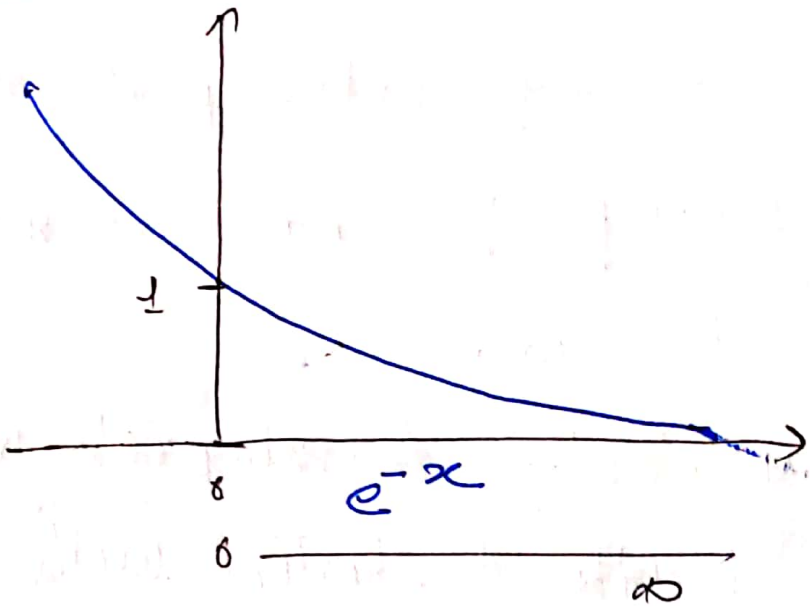
$\psi$ = RMS type update.

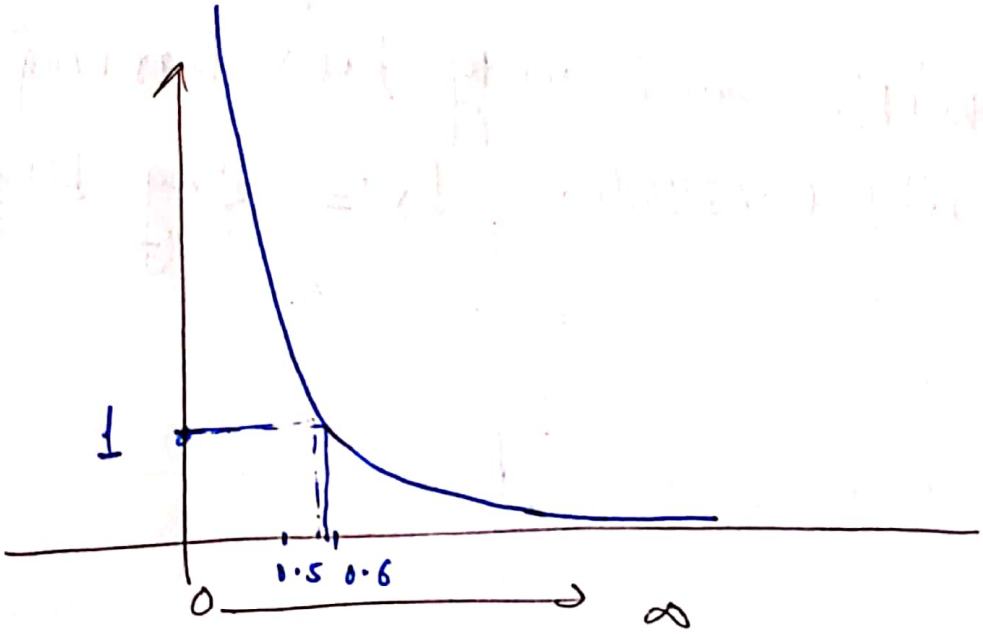
Learning Rate Warm Up :- Instead of setting learning rate ($\alpha_t$) as constant or in decreasing order. Learning Rate Warmup Strategy sets $\alpha_t$ as some small value.

Example :- Linear warmup set $\alpha_t = t\alpha_0$ $t < T_w$.

My Notes :- learning rate warm up is saying inspite of starting with high learning rate start with some very low learning rate & gradually increase. Ex :- Jog before Run.




$(x^{-1}e^{-x})$

# Variance of ADAPTIVE RATE

"Due to lack of samples in the early stage, the adaptive learning rate has an undesirably large variance."

Say $t=1$, $\psi(g_1) = \sqrt{\frac{1}{g_1^2}}$, we view $\{g_1, g_2 \dots g_t\}$ as random variables drawn from Normal Distribution $N(0, \sigma^2)$. Therefore $\frac{1}{g_1^2}$ is subject to scale-inv-$\chi^2(1, 1/g_1^2)$. Note that

$$Var\left[\sqrt{1/g_1^2}\right] \propto \int_0^\infty x^{-1} e^{-x}.$$ This means adaptive learning rate can have very high variance ~~for~~ first stage of learning & setting small learning rate can reduce variance $\Rightarrow$ why ?

$$Var[\alpha x] = \alpha^2 Var[x]$$

Therefore it is unbounded variance that cause problem.

# Estimation of $\rho$ or $\rho_t$ :-

Find the length is SMA( Simple Moving Average) such it has same "center of mass" with EMA( Exponential Moving Average). In other length $f(t, \beta_2)$ satisfies below equation.

$$\frac{(1-\beta_2)\sum_{i=1}^{t} \beta_2^{t-1}(t+1-i)}{1-\beta_2^t} = \sum_{i=1}^{f(t,\beta_2)} \frac{(t+1-i)}{f(t,\beta_2)}$$

Solve above equation :-

$$f(t,\beta_2) = \frac{2}{1-\beta_2} - 1 - \frac{2t\beta_2^t}{1-\beta_2^t} = (\rho_t) \leftarrow \text{Reference}$$

## Rectification Term :-

$$r_t = \sqrt{\frac{(\rho_t - 4)(\rho_t - 2)\rho_\infty}{(\rho_\infty - 4)(\rho_\infty - 2)\rho_t}}$$

## < RADAM >

while t = {1 ... T} do

$g_t \leftarrow \Delta_\theta f_t(\theta_{t-1})$ (gradient at t)

$v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$ (2nd moment)

$m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$ (1st moment)

$\hat{m}_t \leftarrow m_t / (1-\beta_1^t)$ (Bias correction)

calculate $\rho_t$

$\leftarrow \rho_\infty - 2t\beta_2^t / 1-\beta_2^t$

if $\rho_t > 4$ :

$\hat{v}_t \leftarrow \sqrt{v_t / 1-\beta_2^t}$ (Bias Correction)

$r_t \leftarrow$ calculate variance rectification

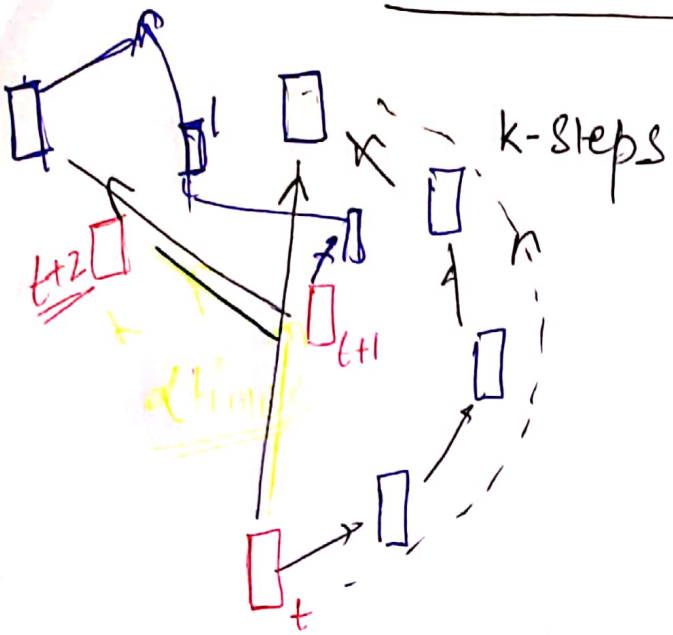
$\theta_t \leftarrow \theta_{t-1} - \alpha_t r_t \hat{m}_t / \hat{v}_t$ (with adaptive moment)

else :

$\theta_t \leftarrow \theta_{t-1} - \alpha_t \hat{m}_t$

(un adapted moment).

# Look Ahead Optimizer




Fast weight
Slow weight

# RADAM

- Recently many new methods have been proposed to accelerate optimization by applying "adaptive learning rate". Example ADAGRAD (2011), RMSProp
- RADAM proposed new variant of ADAM. which explicitly rectifies the variance of the "adaptive learning rate" based on derivation.
- Contribution:- Convergence issue is due to undesirable big variance of the adaptive learning rate in the early stage of model training.
- Proposing RADAM as solution.

$$\phi(g_1, g_2 \dots g_t) = \frac{(1-\beta_1)\sum_{i=1}^{t} \beta_1^{t-i} g_i}{1-\beta_1^t}$$

[$\phi$ momentum type] update.

$$\psi(g_1 g_2 \dots g_t) = \sqrt{\frac{1-\beta_2^t}{\varepsilon + (1-\beta_2)\left(\sum_{i=1}^{t} \beta_2^{t-i} g_i^2\right)}}$$

$\psi$ = RMS type update.

Learning Rate Warm Up :- Instead of setting learning rate ($\alpha_t$) as constant or in decreasing order. Learning Rate Warmup Strategy sets $\alpha_t$ as some small value.

Example :- Linear warmup set $\alpha_t = t\alpha_0$ $t < T_w$.

= My Notes :- learning rate warm up is saying inspite of starting with high learning rate start with some very low learning rate & gradually increase. Ex :- Jog before Run.

# Variance of ADAPTIVE RATE $\alpha$

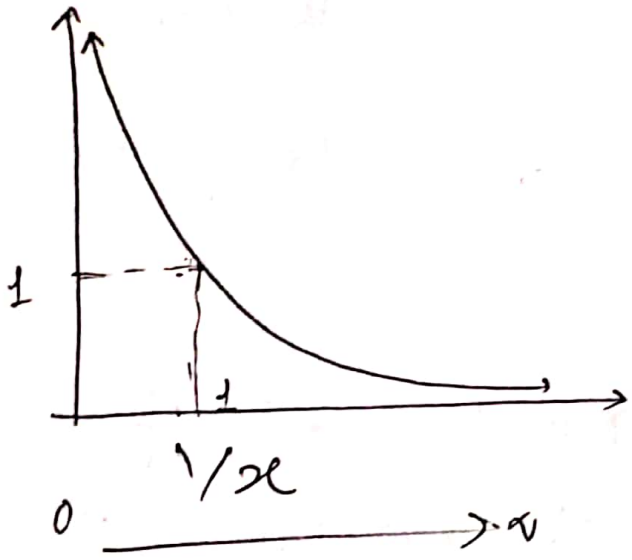
"Due to lack of samples in the early stage, the adaptive learning rate has an undesirably large variance."

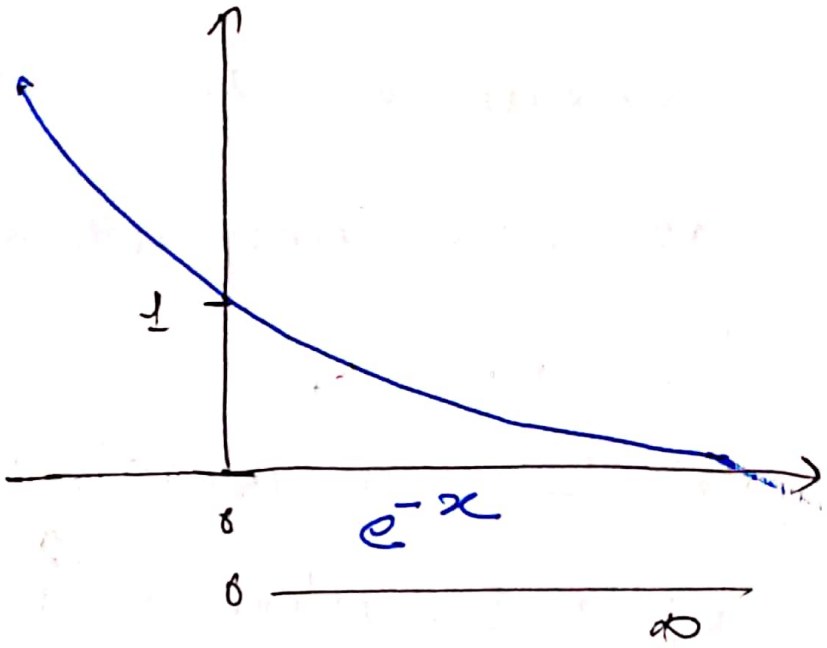
Say $t=1$, $\psi(g_1) = \sqrt{\frac{1}{g_1^2}}$, we view $\{g_1, g_2 \dots g_t\}$ as random variables drawn from Normal Distribution $N(0, \sigma^2)$. Therefore $\frac{1}{g_1^2}$ is subject to scale Scale-inv-$\chi^2(1, 1/g_2)$. Note that

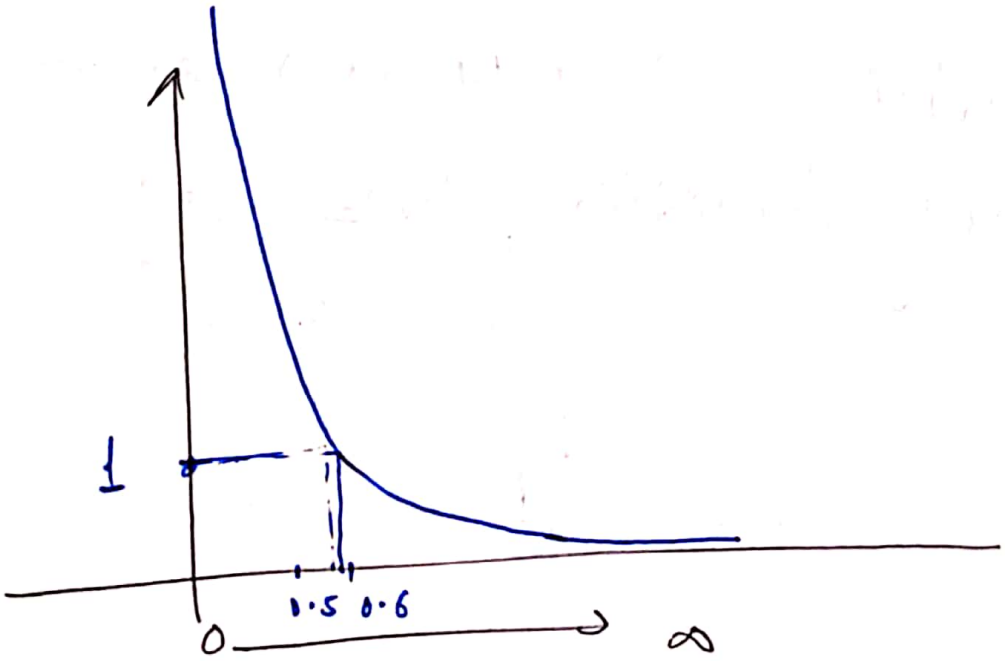
$$\text{Var}\left[\sqrt{1/g_1^2}\right] \propto \int_0^\infty x^{-1} e^{-x}.$$ This means adaptive learning rate can have very high variance ~~(x)~~ first stage of learning & setting small learning rate can reduce variance $\Rightarrow$ why ?

$$\text{Var}[\alpha x] = \alpha^2 \text{Var}[x]$$

Therefore it is unbounded variance that cause problem.




$(x^{-1}e^{-x})$

# Estimation of $\rho$ or $\rho_t$ :-

Find the length is SMA( Simple Moving Average) such it has same" center of mass" with EMA( Exponential Moving Average). In other length $f(t, \beta_2)$ satisfies below equation.

$$\frac{(1-\beta_2)\sum_{i=1}^{t} \beta_2^{t-1}(t+1-i)}{1-\beta_2^t} = \sum_{i=1}^{t} \frac{(t+1-i)}{f(t,\beta_2)}$$

$f(t,\beta_2)$

Solve above equation :-

$$f(t,\beta_2) = \frac{2}{1-\beta_2} - 1 - \frac{2t\beta_2^t}{1-\beta_2^t} = (\rho_t) \leftarrow \text{Refer as}$$

## Rectification Term :-

$$r_t = \sqrt{\frac{(\rho_t - 4)(\rho_t - 2)\rho_\infty}{(\rho_\infty - 4)(\rho_\infty - 2)\rho_t}}$$

< RADAM >

while $t = \{1 \ldots T\}$ do

$g_t \leftarrow \Delta_\theta f_t(\theta_{t-1})$ (gradient at t)

$v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$ ($2^{nd}$ moment)

$m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$ ($1^{st}$ moment)

$\hat{m}_t \leftarrow m_t / (1-\beta_1^t)$ (Bias correction)

calculate $\rho_t$

$\leftarrow \rho_\infty - 2t\beta_2^t / 1-\beta_2^t$

if $\rho_t > 4$ :

$\hat{v}_t \leftarrow \sqrt{v_t / 1-\beta_2^t}$ (Bias Correction)

$r_t \leftarrow$ calculate variance rectification

$\theta_t \leftarrow \theta_{t-1} - \alpha_t r_t \hat{m}_t / \hat{v}_t$ (with adaptive moment)

else :

~~$\theta_t \leftarrow \theta_{t-1}$~~

$\theta_t \leftarrow \theta_{t-1} - \alpha_t \hat{m}_t$

(un adapted moment).