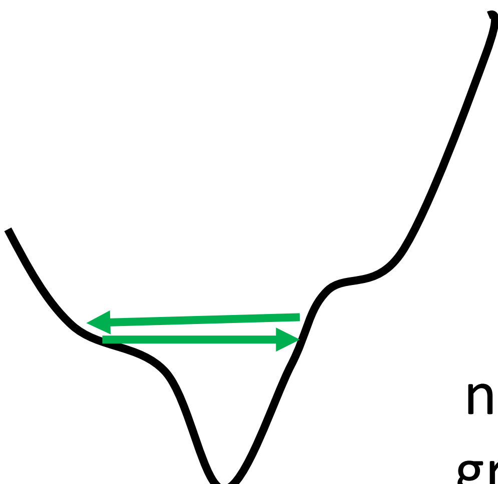
Error surface is rugged …

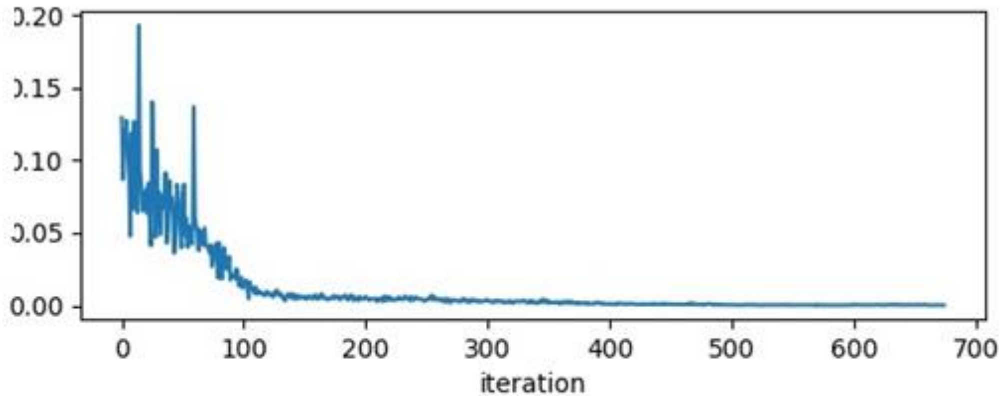
Tips for training: **Adaptive Learning Rate** https://docs.google.com/presentation/d/1siUFXARYRpNiMeSRwgFbt7mZVjkMPhR5od09w0Z8xaU/edit#slide=id.g3532c09be1_0_382

Training stuck ≠ Small Gradient

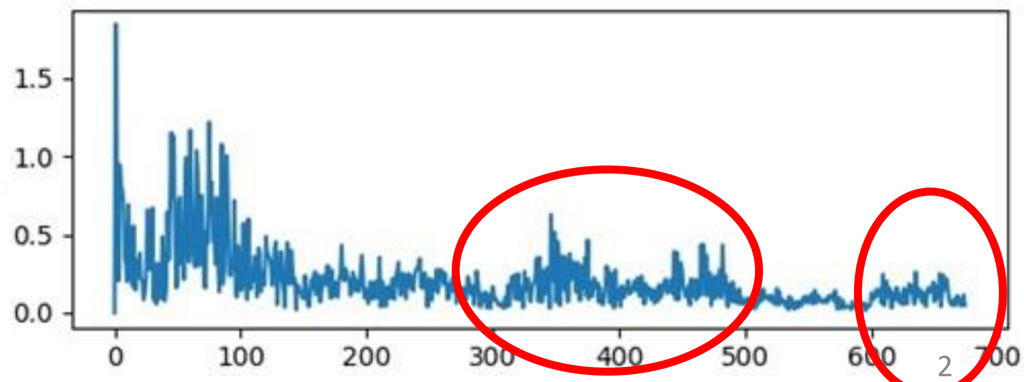
- People believe training stuck because the parameters are around a critical point …



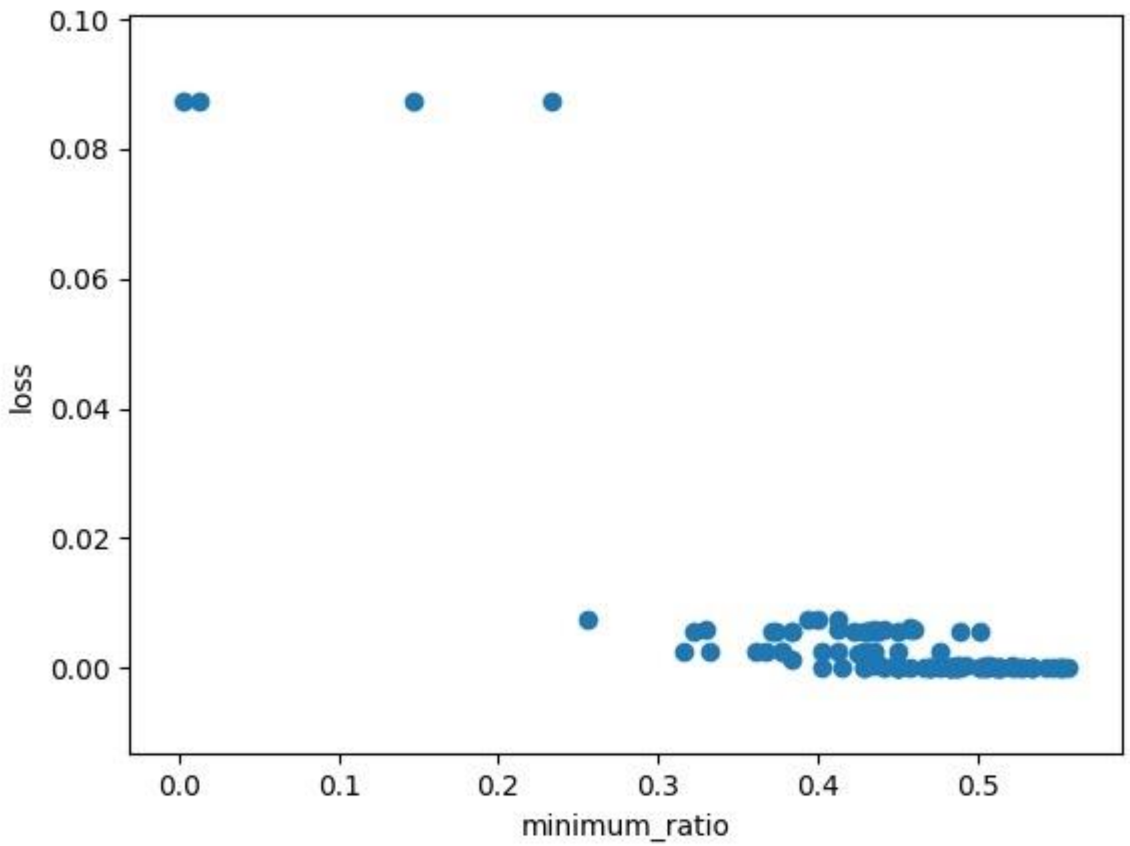
loss



norm of gradient

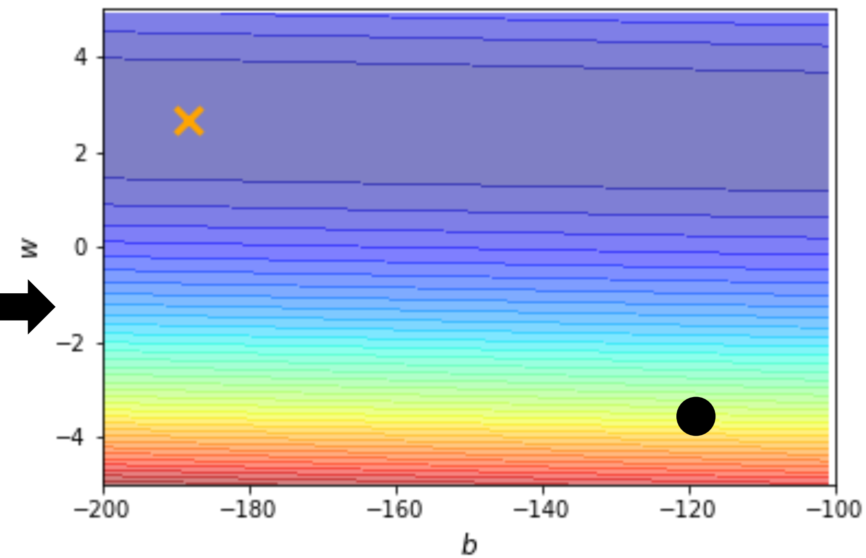


Wait a minute …

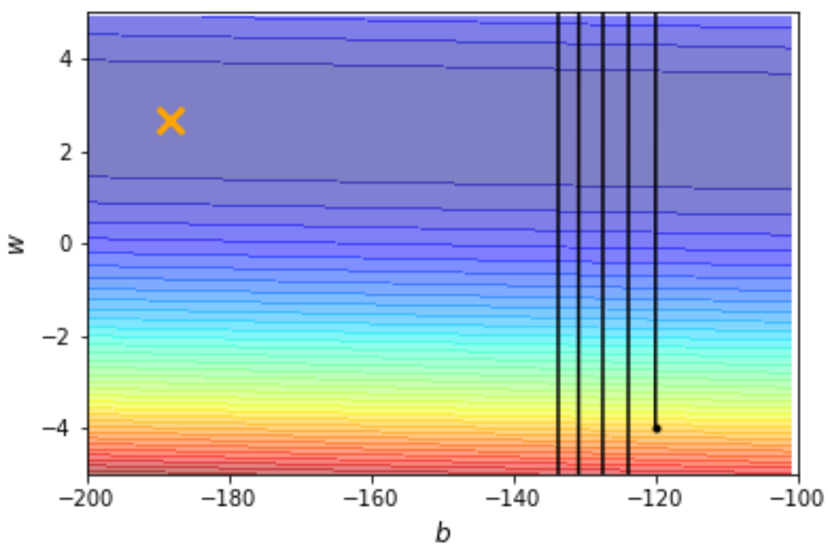


Training can be difficult even without critical points.

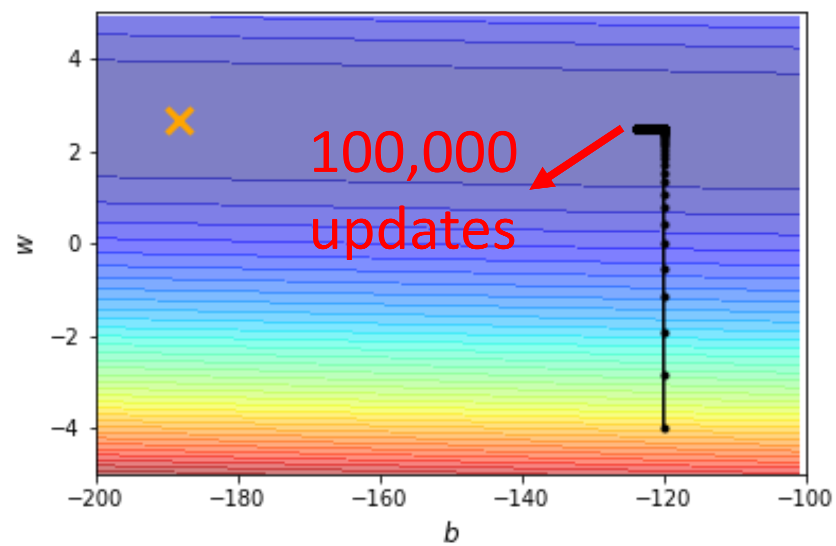
This error surface is convex.



Learning rate **cannot** be **one-size-fits-all**

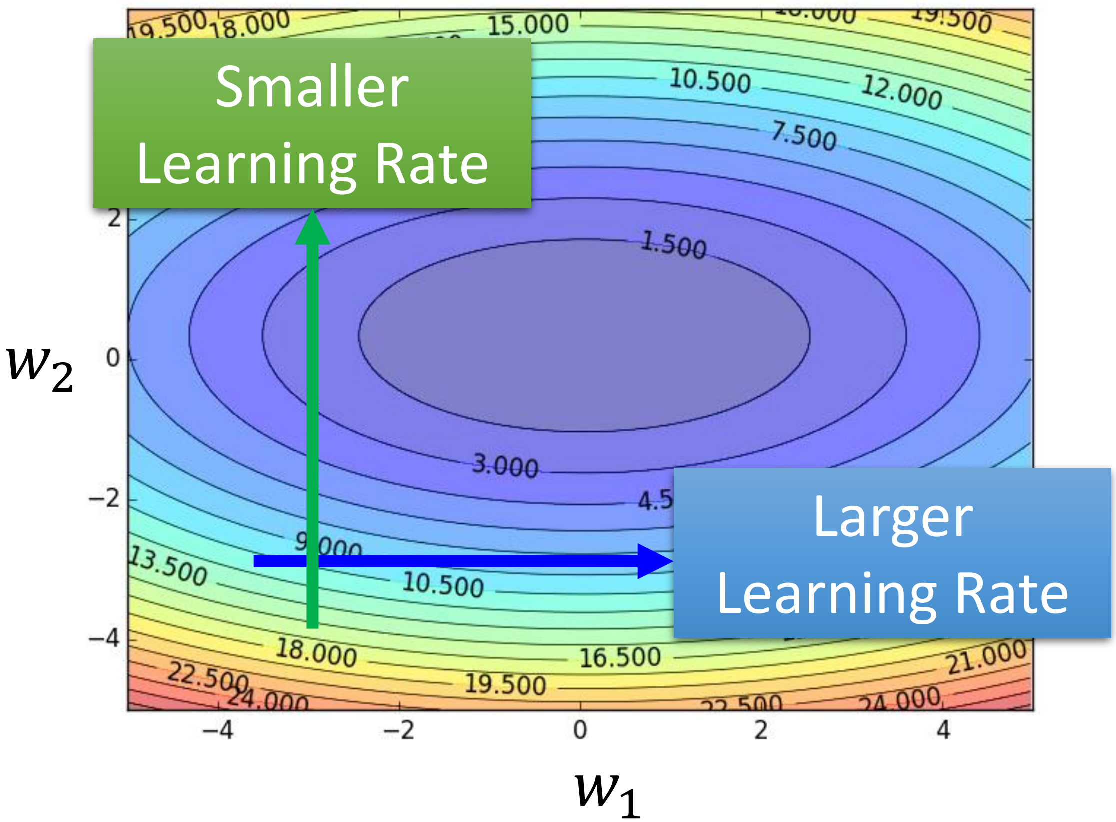


$\eta = 10^{-2}$



$\eta = 10^{-7}$

Different parameters needs different learning rate



Formulation for **one** parameter:

$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^{t} - \boxed{\eta}\,\boldsymbol{g}_i^{t}$$

$$\boldsymbol{g}_i^{t} = \frac{\partial L}{\partial \boldsymbol{\theta}_i}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^t}$$

$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^{t} - \boxed{\frac{\eta}{\sigma_i^t}}\,\boldsymbol{g}_i^{t}$$

Parameter dependent

均方根

Root Mean Square

$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^t - \boxed{\frac{\eta}{\sigma_i^t}} \boldsymbol{g}_i^t$$

$$\boldsymbol{\theta}_i^1 \leftarrow \boldsymbol{\theta}_i^0 - \frac{\eta}{\sigma_i^0} \boldsymbol{g}_i^0 \qquad \sigma_i^0 = \sqrt{\left(\boldsymbol{g}_i^0\right)^2} = \left|\boldsymbol{g}_i^0\right|$$

$$\boldsymbol{\theta}_i^2 \leftarrow \boldsymbol{\theta}_i^1 - \frac{\eta}{\sigma_i^1} \boldsymbol{g}_i^1 \qquad \sigma_i^1 = \sqrt{\frac{1}{2}\left[\left(\boldsymbol{g}_i^0\right)^2 + \left(\boldsymbol{g}_i^1\right)^2\right]}$$

$$\boldsymbol{\theta}_i^3 \leftarrow \boldsymbol{\theta}_i^2 - \frac{\eta}{\sigma_i^2} \boldsymbol{g}_i^2 \qquad \sigma_i^2 = \sqrt{\frac{1}{3}\left[\left(\boldsymbol{g}_i^0\right)^2 + \left(\boldsymbol{g}_i^1\right)^2 + \left(\boldsymbol{g}_i^2\right)^2\right]}$$

$$\vdots$$

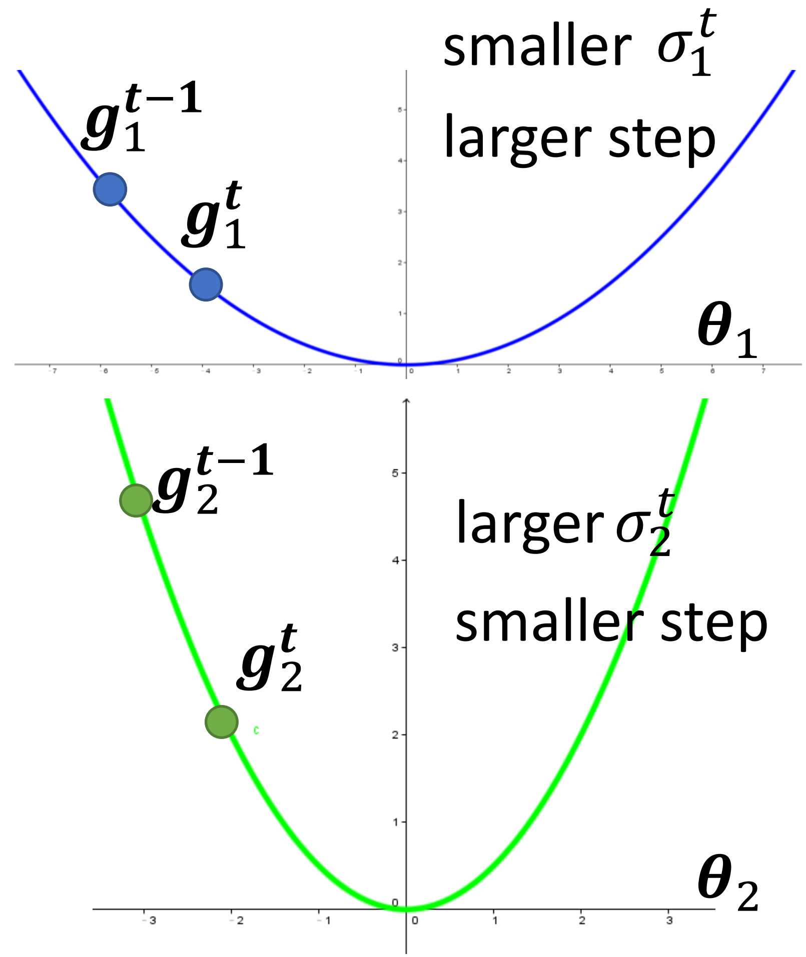
$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^t - \frac{\eta}{\sigma_i^t} \boldsymbol{g}_i^t \qquad \sigma_i^t = \sqrt{\frac{1}{t+1} \sum_{i=0}^{t} \left(\boldsymbol{g}_i^t\right)^2}$$

Root Mean Square

$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^{t} - \boxed{\frac{\eta}{\sigma_i^t}} \boldsymbol{g}_i^t$$

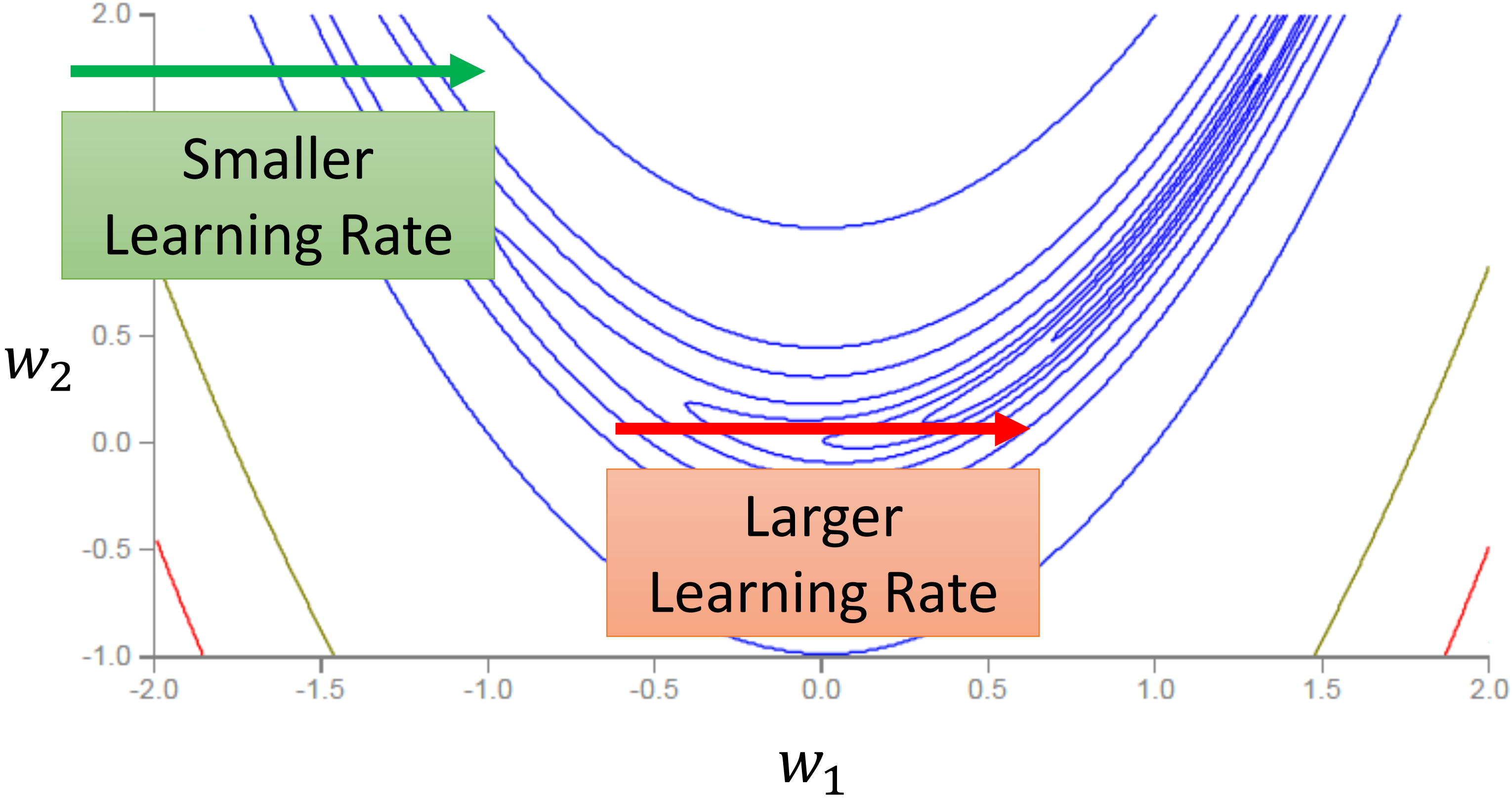
$$\sigma_i^t = \sqrt{\frac{1}{t+1}\sum_{i=0}^{t}\left(\boldsymbol{g}_i^t\right)^2}$$

Used in **Adagrad**



Learning rate adapts dynamically

Error Surface can be very complex.



RMSProp

$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^{t} - \boxed{\frac{\eta}{\sigma_i^t}} \boldsymbol{g}_i^t$$

$$\boldsymbol{\theta}_i^{1} \leftarrow \boldsymbol{\theta}_i^{0} - \frac{\eta}{\sigma_i^0} \boldsymbol{g}_i^{0} \qquad \sigma_i^0 = \sqrt{\left(\boldsymbol{g}_i^{0}\right)^2}$$

$$0 < \alpha < 1$$

$$\boldsymbol{\theta}_i^{2} \leftarrow \boldsymbol{\theta}_i^{1} - \frac{\eta}{\sigma_i^1} \boldsymbol{g}_i^{1} \qquad \sigma_i^1 = \sqrt{\alpha\left(\sigma_i^0\right)^2 + (1-\alpha)\left(\boldsymbol{g}_i^{1}\right)^2}$$

$$\boldsymbol{\theta}_i^{3} \leftarrow \boldsymbol{\theta}_i^{2} - \frac{\eta}{\sigma_i^2} \boldsymbol{g}_i^{2} \qquad \sigma_i^2 = \sqrt{\alpha\left(\sigma_i^1\right)^2 + (1-\alpha)\left(\boldsymbol{g}_i^{2}\right)^2}$$

$$\vdots$$

$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^{t} - \frac{\eta}{\sigma_i^t} \boldsymbol{g}_i^{t} \qquad \sigma_i^t = \sqrt{\alpha\left(\sigma_i^{t-1}\right)^2 + (1-\alpha)\left(\boldsymbol{g}_i^{t}\right)^2}$$

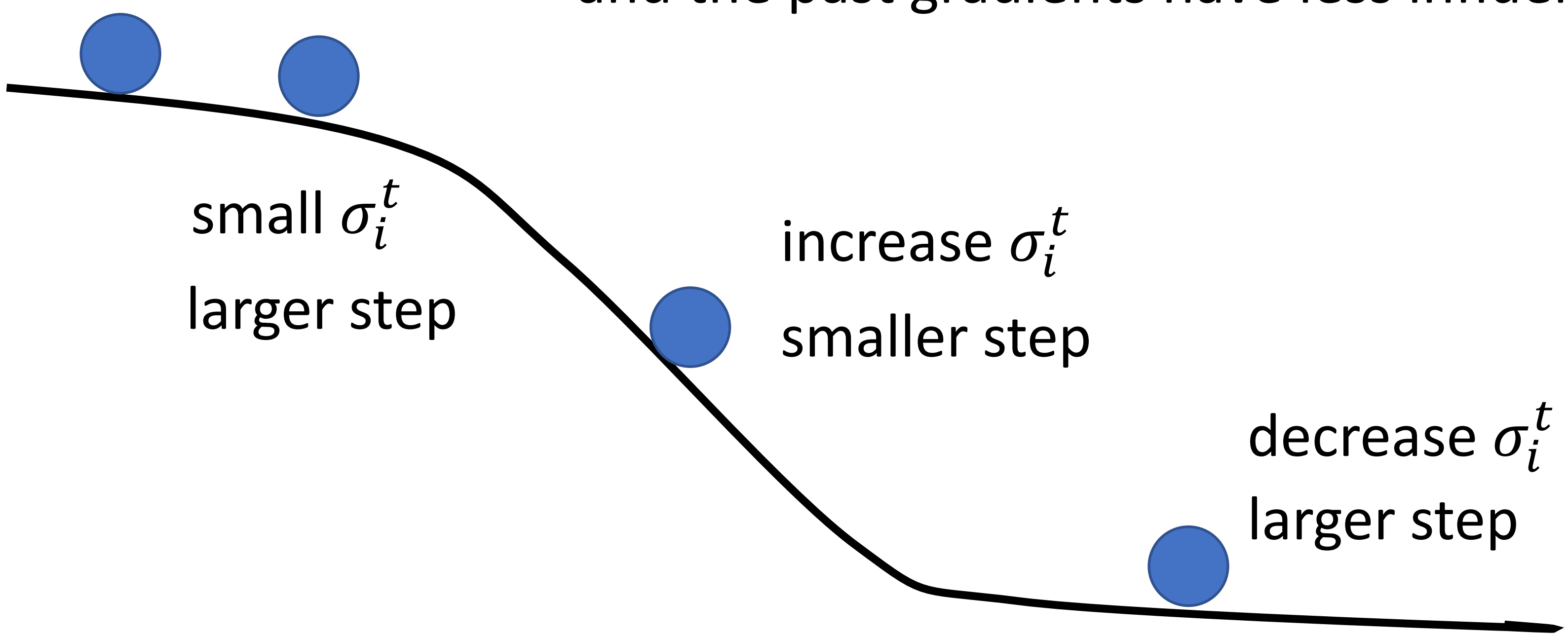
RMSProp

$$\boldsymbol{g}_i^{\boldsymbol{1}} \ \boldsymbol{g}_i^{\boldsymbol{2}} \ \cdots\cdots \ \boldsymbol{g}_i^{\boldsymbol{t-1}}$$

$$0 < \alpha < 1$$

$$\boldsymbol{\theta}_i^{\boldsymbol{t+1}} \leftarrow \boldsymbol{\theta}_i^{\boldsymbol{t}} - \boxed{\frac{\eta}{\sigma_i^t}} \boldsymbol{g}_i^{\boldsymbol{t}} \qquad \sigma_i^t = \sqrt{\alpha\left(\sigma_i^{t-1}\right)^2 + (1-\alpha)\left(\boldsymbol{g}_i^{\boldsymbol{t}}\right)^2}$$

The recent gradient has larger influence, and the past gradients have less influence.



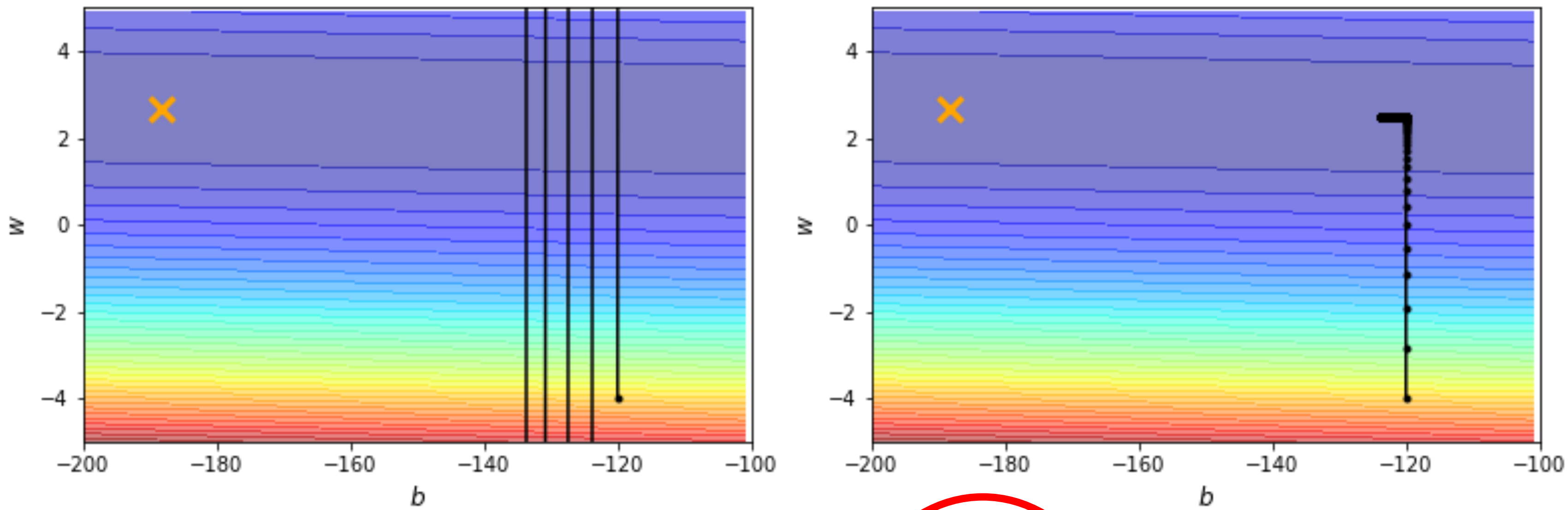
Original paper: https://arxiv.org/pdf/1412.6980.pdf

Adam: RMSProp + Momentum

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.

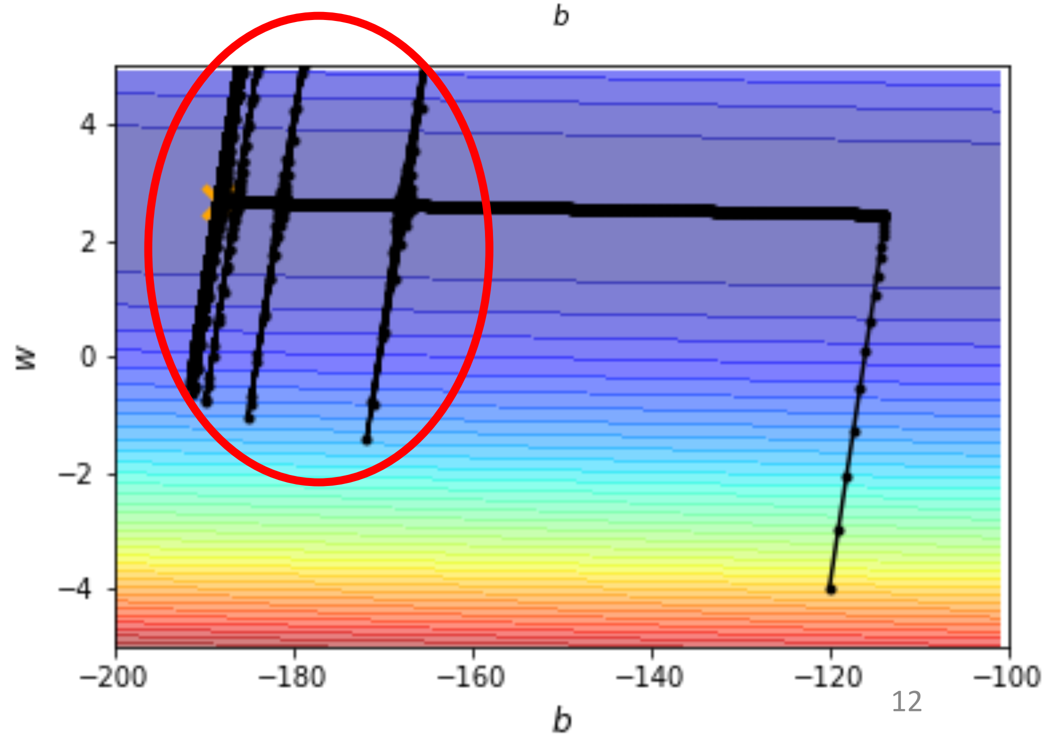
Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector) → for momentum
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector) → for RMSprop
  $t \leftarrow 0$ (Initialize timestep)
  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)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (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)

Without Adaptive Learning Rate



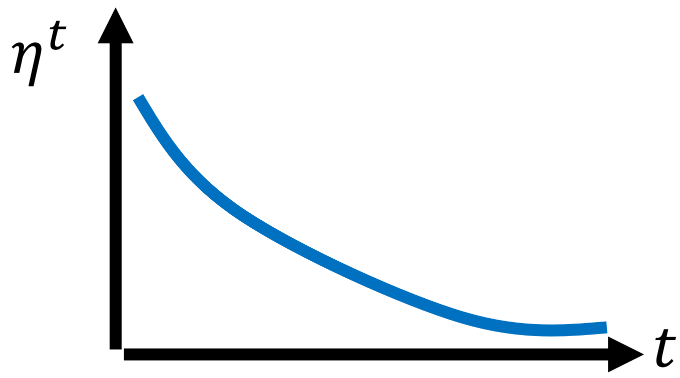
$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^{t} - \boxed{\frac{\eta}{\sigma_i^t}} \boldsymbol{g}_i^t$$

$$\sigma_i^t = \sqrt{\frac{1}{t+1}\sum_{i=0}^{t}\left(\boldsymbol{g}_i^t\right)^2}$$



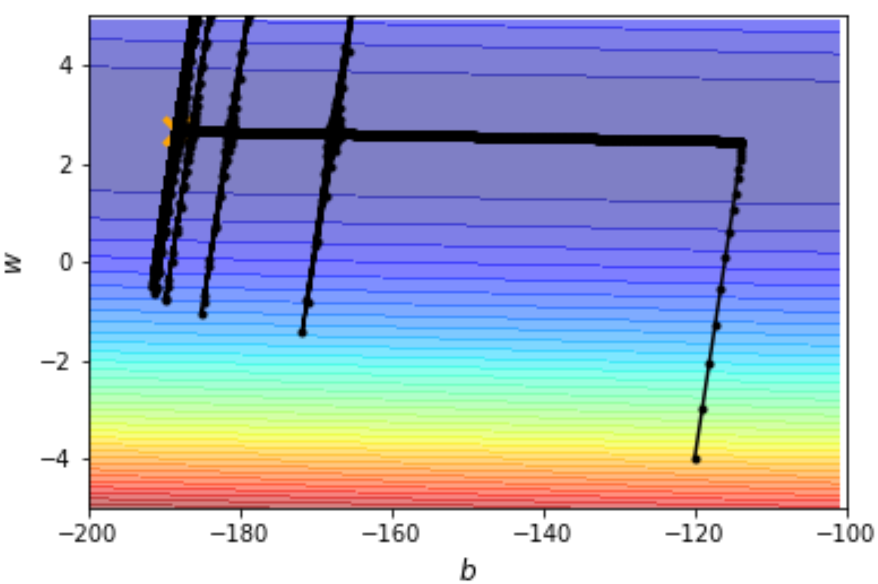
Learning Rate Scheduling

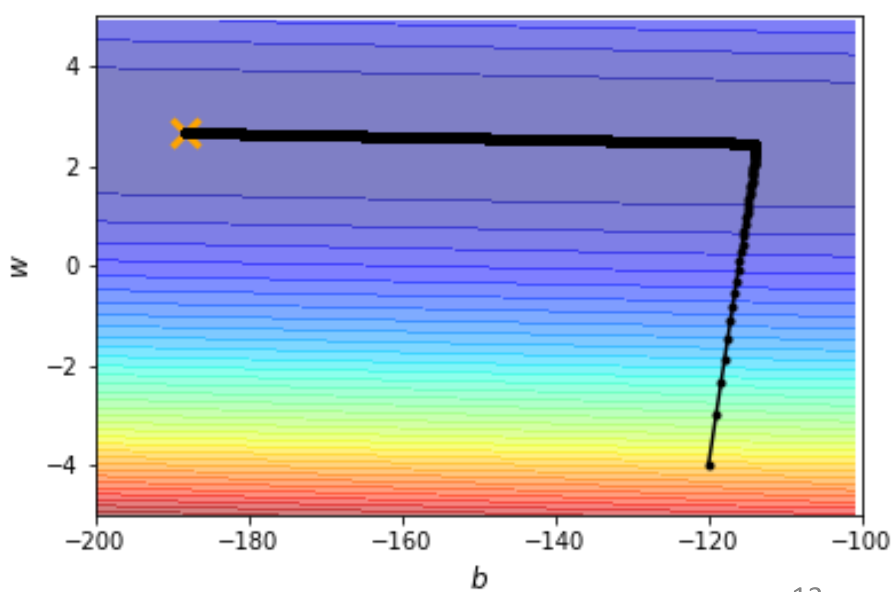
$$\theta_i^{t+1} \leftarrow \theta_i^t - \frac{\boxed{\eta^t}}{\sigma_i^t} g_i^t$$



Learning Rate Decay

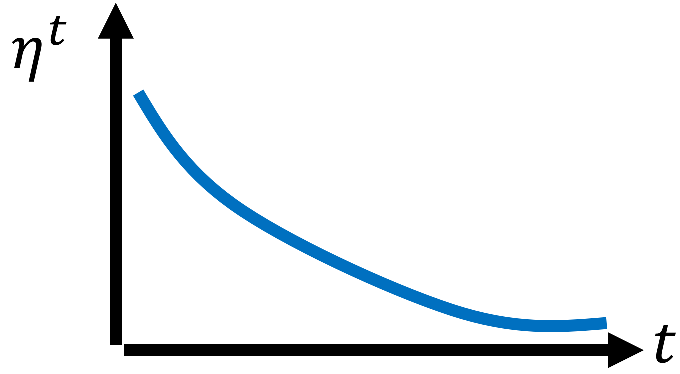
As the training goes, we are closer to the destination, so we reduce the learning rate.





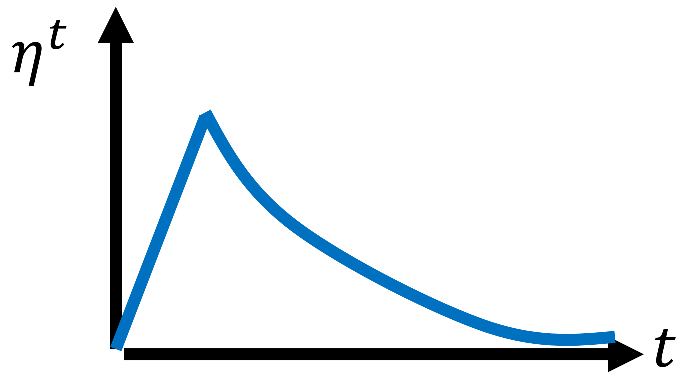
Learning Rate Scheduling

$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^{t} - \frac{\boxed{\eta^t}}{\sigma_i^t} \boldsymbol{g}_i^t$$



Learning Rate Decay

As the training goes, we are closer to the destination, so we reduce the learning rate.



Warm Up

Increase and then decrease?

We further explore $n = 18$ that leads to a 110-layer ResNet. In this case, we find that the initial learning rate of 0.1 is slightly too large to start converging[5]. So we use 0.01 to warm up the training until the training error is below 80% (about 400 iterations), and then go back to 0.1 and continue training. The rest of the learning schedule is as done previously. This 110-layer network converges well (Fig. 6, middle). It has *fewer* parameters than other deep and thin

[5] With an initial learning rate of 0.1, it starts converging (<90% error) after several epochs, but still reaches similar accuracy.

Residual Network

https://arxiv.org/abs/1512.03385

5.3 Optimizer

We used the Adam optimizer [17] with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. We varied the learning rate over the course of training, according to the formula:

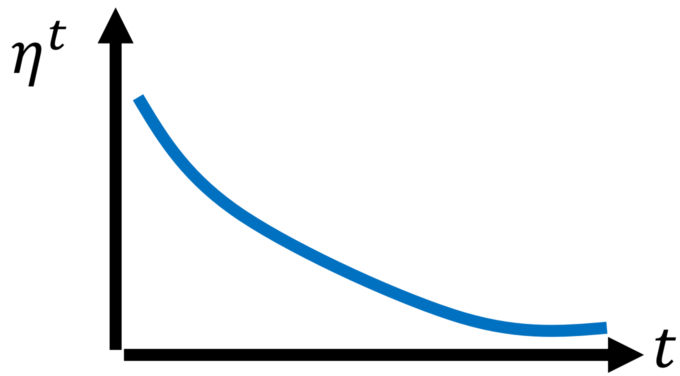
$$lrate = d_{\text{model}}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5}) \tag{3}$$

This corresponds to increasing the learning rate linearly for the first $warmup\_steps$ training steps, and decreasing it thereafter proportionally to the inverse square root of the step number. We used $warmup\_steps = 4000$.

Transformer https://arxiv.org/abs/1706.03762

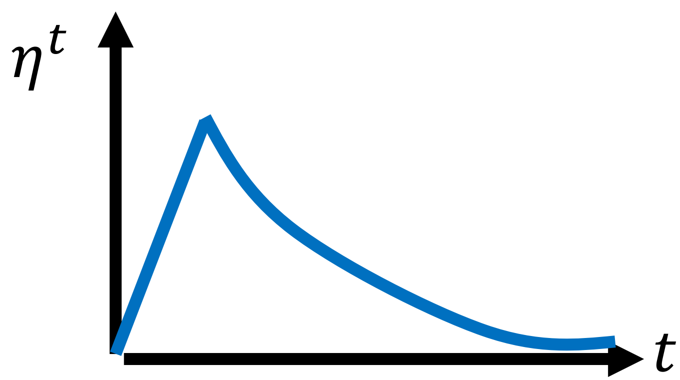
Learning Rate Scheduling

$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^t - \frac{\boxed{\eta^t}}{\sigma_i^t} \boldsymbol{g}_i^t$$



Learning Rate Decay

After the training goes, we are close to the destination, so we reduce the learning rate.



Warm Up

Increase and then decrease?

At the beginning, the estimate of σ_i^t has large variance.

Please refer to **RAdam** https://arxiv.org/abs/1908.03265

Summary of Optimization

(Vanilla) Gradient Descent

$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^{t} - \eta \boldsymbol{g}_i^{t}$$

Various Improvements

Learning rate scheduling

$$\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^{t} - \frac{\eta^t}{\sigma_i^t} \boldsymbol{m}_i^{t}$$

Momentum: weighted sum of the previous gradients

Consider direction

root mean square of the gradients

only magnitude

To Learn More ……

https://youtu.be/4pUmZ8hXlHM

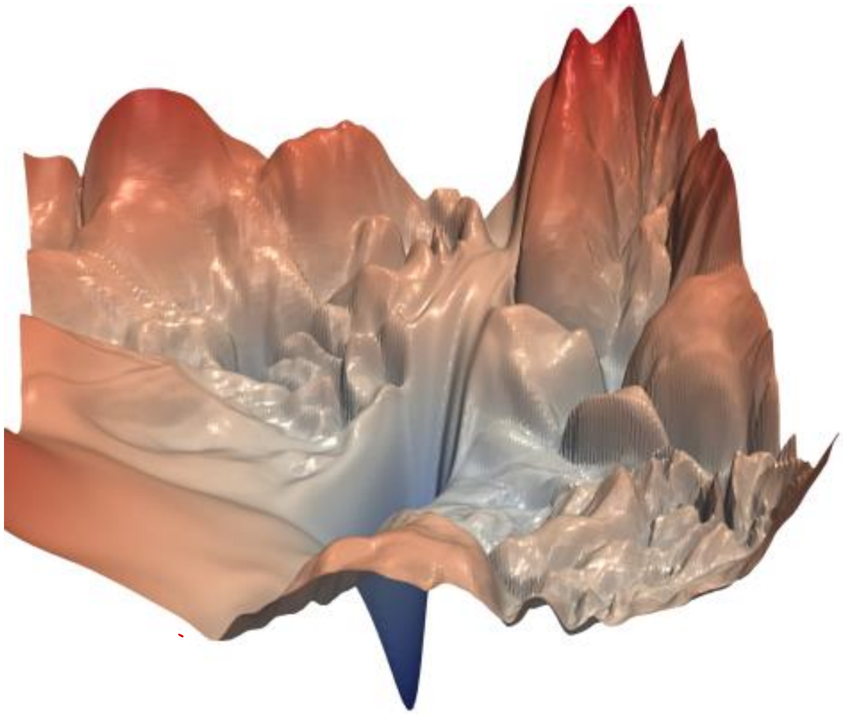
(in Mandarin)

https://youtu.be/e03YKGHXnL8

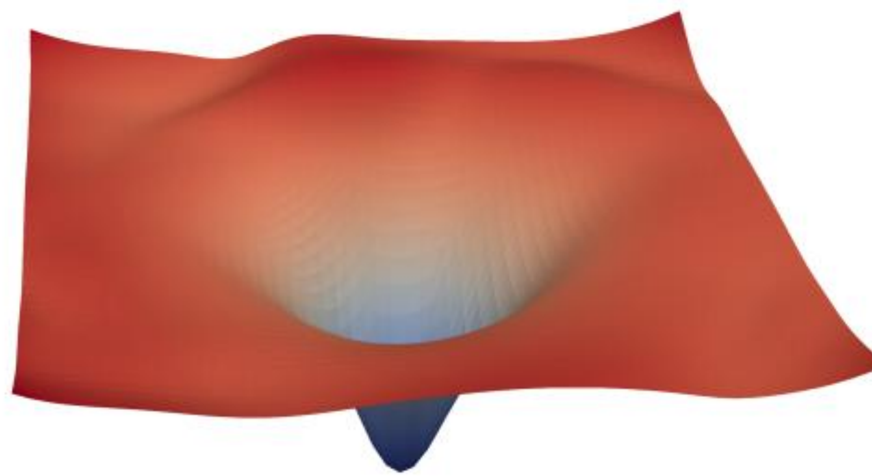
(in Mandarin)

Next Time

Source of image: https://arxiv.org/abs/1712.09913



Next time



Better optimization strategies: If the mountain won't move, build a road around it.

Can we change the error surface? Directly move the mountain!

Adadelta 算法

$$s_t = \rho s_{t-1} + (1-\rho) g_t^2$$

$$x_t = x_{t-1} - g'$$

$$g' = \frac{\sqrt{\Delta x_{t-1} + \varepsilon}}{\sqrt{s_t + \varepsilon}} \odot g_t$$

$$\Delta x_t = \rho \Delta x_{t-1} + (1-\rho) g_t'^2$$