Learning

Optimization and gradient descent

Useful material

Math for ML: https://gwthomas.github.io/docs/math4ml.pdf

The matrix cookbook: https://www2.imm.dtu.dk/pubdb/pubs/3274-full.html

https://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html

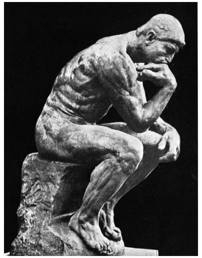
Learning

Machines make mistakes, but they can learn

$$\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & & \\ \vdots & \vdots & & \\ \vdots & \vdots & & \\ x_{M1} & x_{M2} & \cdots & x_{MN} \end{pmatrix} \longrightarrow \boxed{\text{ML method } f_w} \longrightarrow \begin{pmatrix} l_1 \\ l_2 \\ \vdots \\ \vdots \\ l_M \end{pmatrix}$$

true labels,
“ground truth”

Making mistakes and correcting them is learning.



Predicted label: $\hat{l}_1 = f_w\left(x_{11}, x_{12}, \cdots, x_{1N}\right)$

We want to tune the parameters w so that predicted label is close to the true one

$$f_w\left(x_{11}, x_{12}, \cdots, x_{1N}\right) = f\left(x_{11}, x_{12}, \cdots, x_{1N}; w_1, w_2, \cdots, w_k\right)$$

depends on w_i

The loss function

Measuring the prediction error made by the machine learning method

Predicted value for sample i: $\hat{l}_i = f_w\left(x_{i1}, x_{i2}, \cdots, x_{iN}\right)$

Prediction error for sample i: $E_i(l_i, \hat{l}_i)$

Example of squared error: $E_i = |l_i - \hat{l}_i|^2$ global error $E = \sum_i |l_i - \hat{l}_i|^2$

The error depends on the parameters of the machine learning model

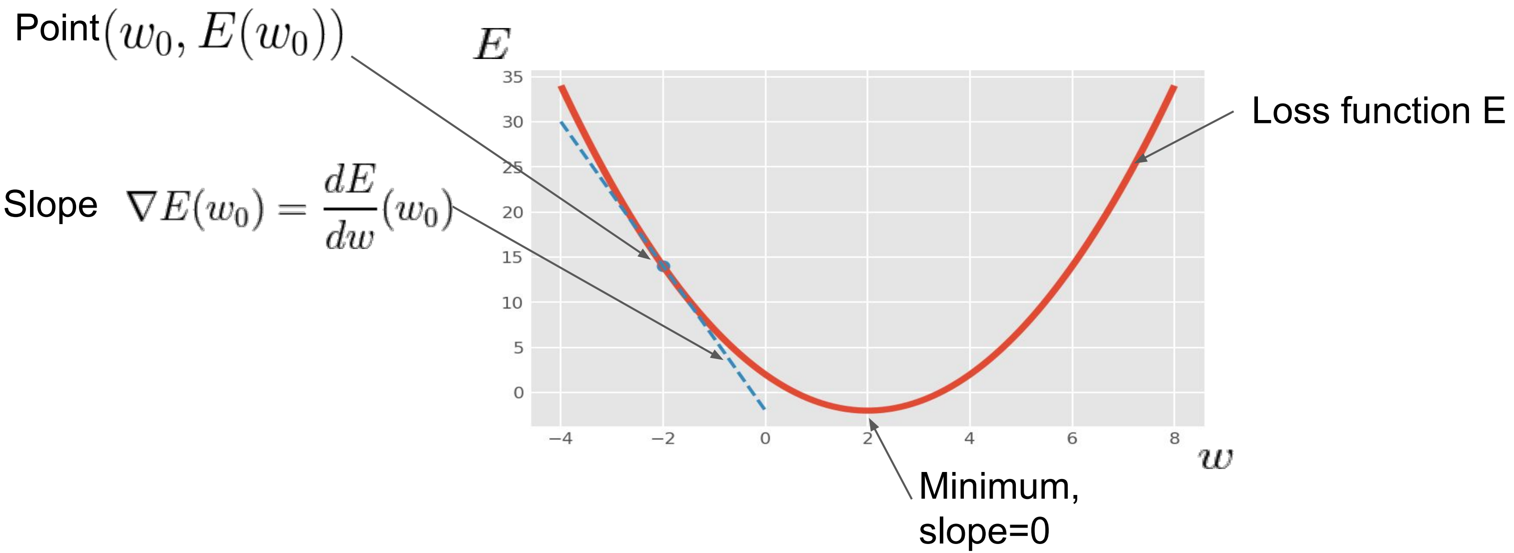
$$E(w) = \sum_i |l_i - \hat{l}_i(w)|^2$$

Minimization

we want to minimize the error

How? -> the error depends on the weights w

We want to find the w for which it is E minimal: $\underset{w}{\operatorname{argmin}} \; E(w)$



Finding the minimum using the gradient

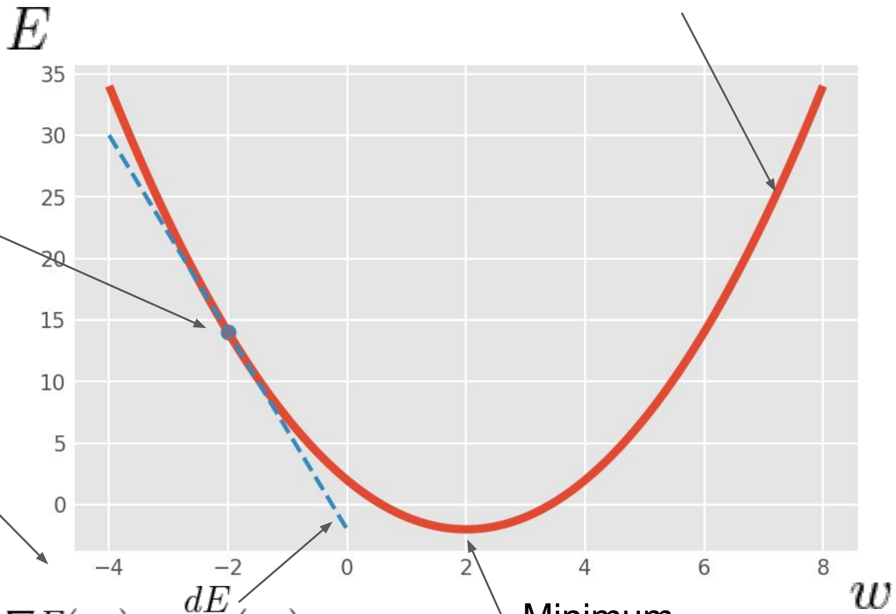
Gradient descent
The algorithm: an iterative process

- start at a random point w_0
- compute the gradient or derivative $\nabla E(w_0)$
- move one step following:

$$w_{n+1} = w_n - \alpha \nabla E(w_n)$$

The **step size** $\boldsymbol{\alpha}$ is a parameter to be chosen. It is also called the **learning rate**.

Loss function



Slope $\nabla E(w_0) = \dfrac{dE}{dw}(w_0)$

Minimum, slope=0

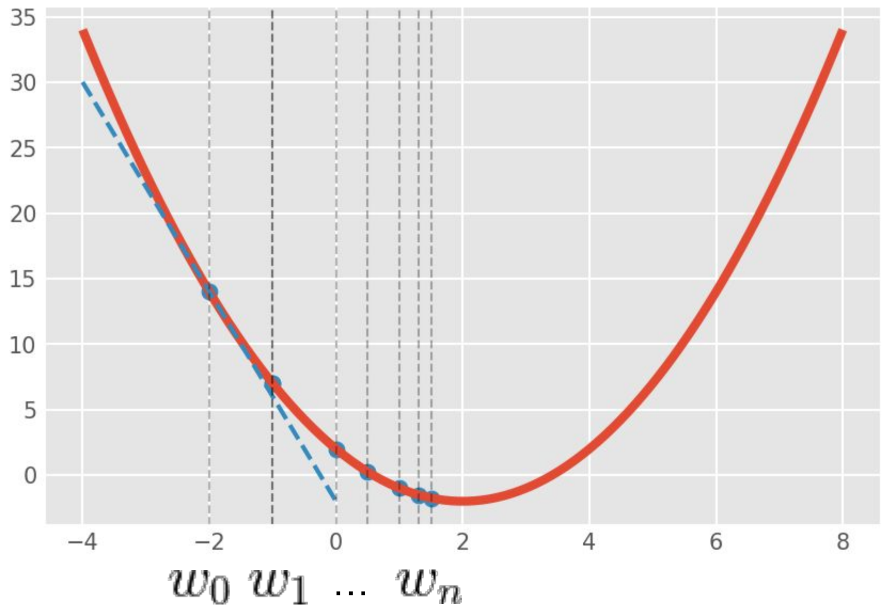
Gradient descent

$$w_{n+1} = w_n - \alpha \nabla E(w_n)$$

Iterate until minimum reached or

$$\left| w_{n+1} - w_n \right| \leq \varepsilon$$

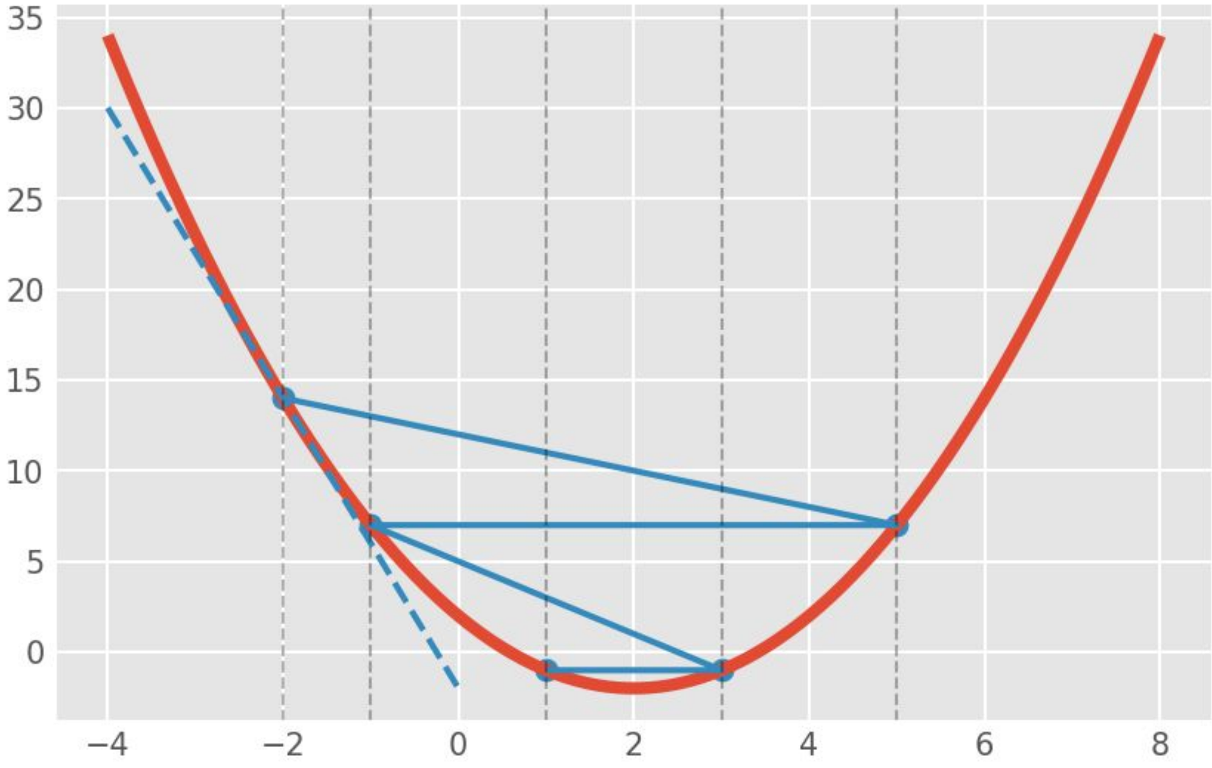
for a given small positive ε



Gradient descent

$$w_{n+1} = w_n - \alpha \nabla E(w_n)$$

When the step size α is too big

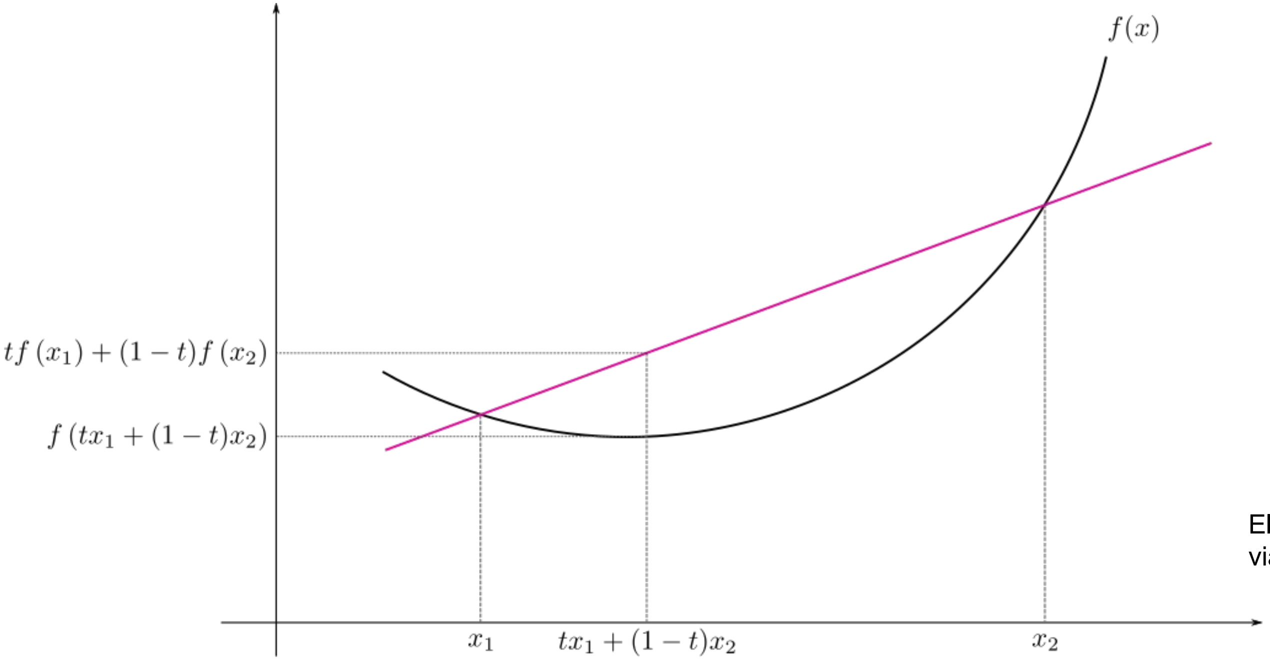


The step size α will depend on the application

- too small: slow convergence
- too big: no convergence

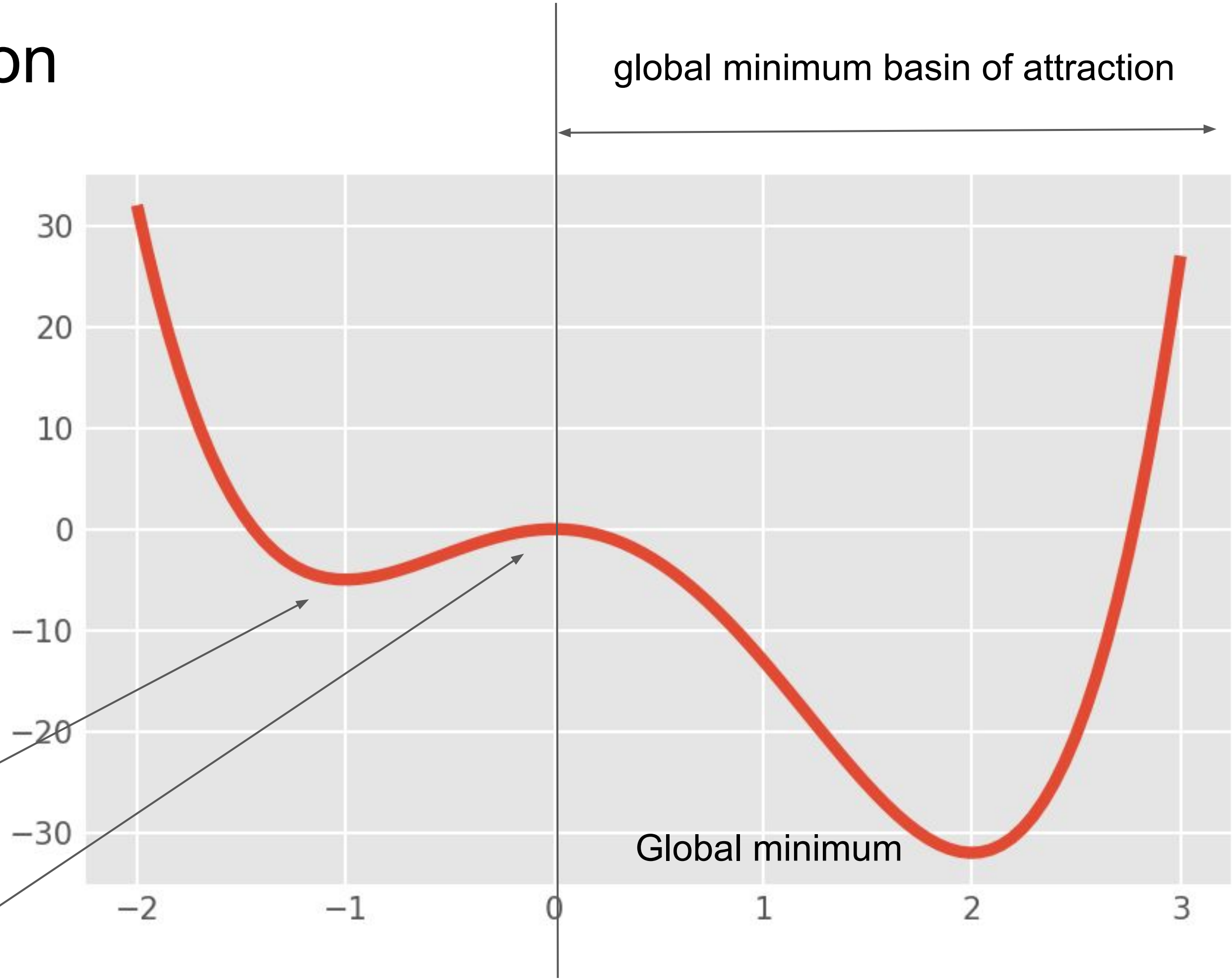
Convex function

f convex: $\forall t \in [0,1], \quad f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$
$\forall x_1, x_2 \in \operatorname{dom} f$



Non-convex function

- The solution is not unique or there may not be any solution
- This is the case in machine learning: many local minima
- Final solution depends on the initial conditions



Several coordinates

The gradient is a vector

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_N} \end{pmatrix}$$

$$\frac{\partial E}{\partial w_i}(w)$$

The effect of a slight change of the parameter i on the error.

Stochastic gradient descent (SGD)

The loss function takes into account all the training set $$E = \sum_i |l_i - \hat{l}_i|^2$$

- slow to compute
- may need a lot of memory if dataset is large

Instead: compute the gradient for each sample $$E_i = |l_i - \hat{l}_i|^2$$

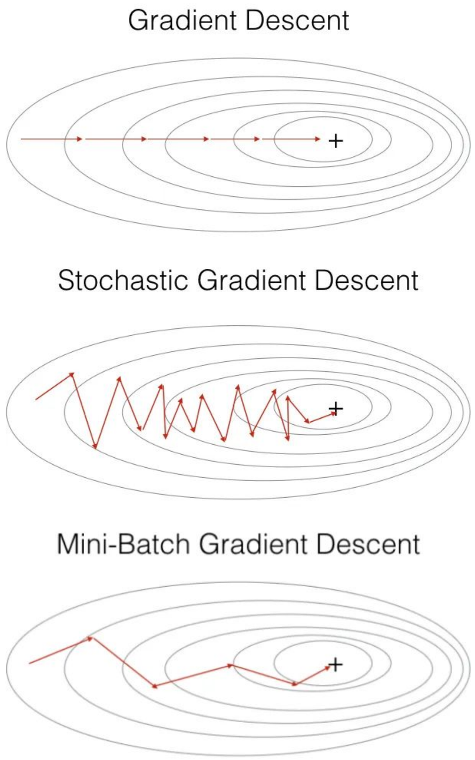
Batch or mini-batch gradient descent: compromise, compute the gradient for a small set of samples -> “batch size” in deep learning

Example: gradient for the logistic regression $$\frac{\partial}{\partial a} L = X^T(\hat{y} - y)$$

Exercise: find the expression for the SGD and batch GD

Convergence of the gradient descent

SGD is faster to compute but has slower convergence.



from https://www.coursera.org/learn/deep-neural-network/

Gradient descent with momentum

Severals improvement for the SGD

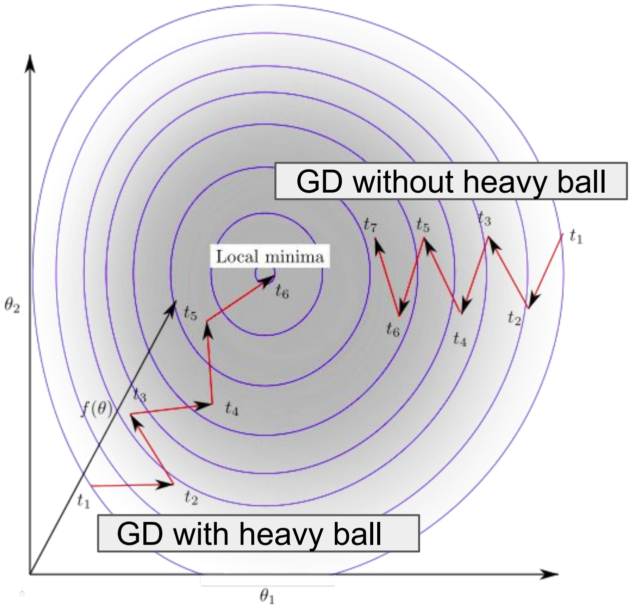
Example: heavy ball

$$w_{t+1} = w_t - \alpha \nabla E(w_t) + \beta(w_t - w_{t-1})$$

Momentum

$(w_t - w_{t-1})$ is the variation at the previous step

2 stochastic gradient descents

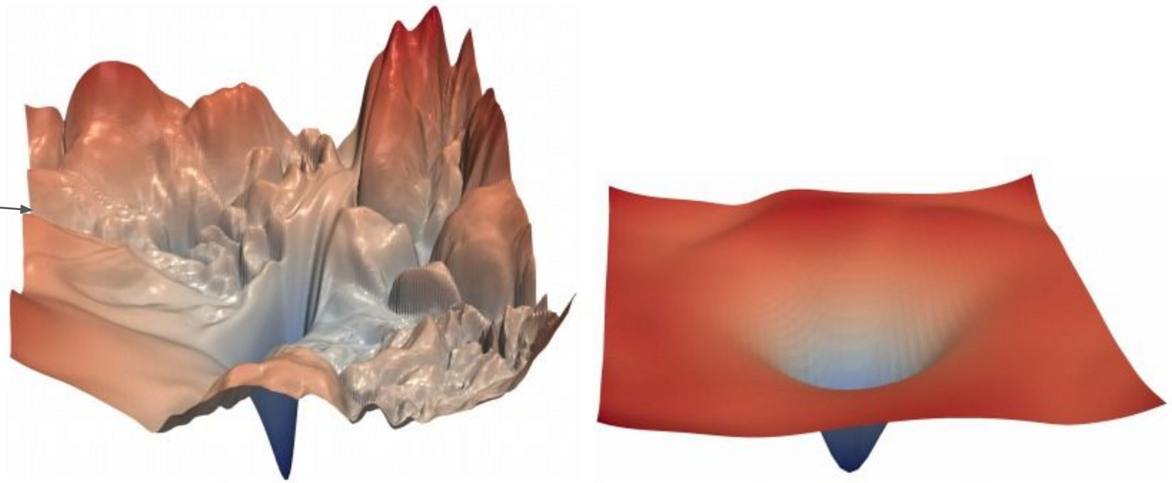


From https://commons.wikimedia.org/wiki/File:Gradient_descent_with_momentum.svg
Public domain

Loss landscape

Loss functions in neural nets are non-convex

An example of loss function of a deep neural network.



(a) without skip connections (b) with skip connections

Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

Gradient descent may be a challenge!

From: Visualizing the Loss Landscape of Neural Nets, https://arxiv.org/pdf/1712.09913.pdf