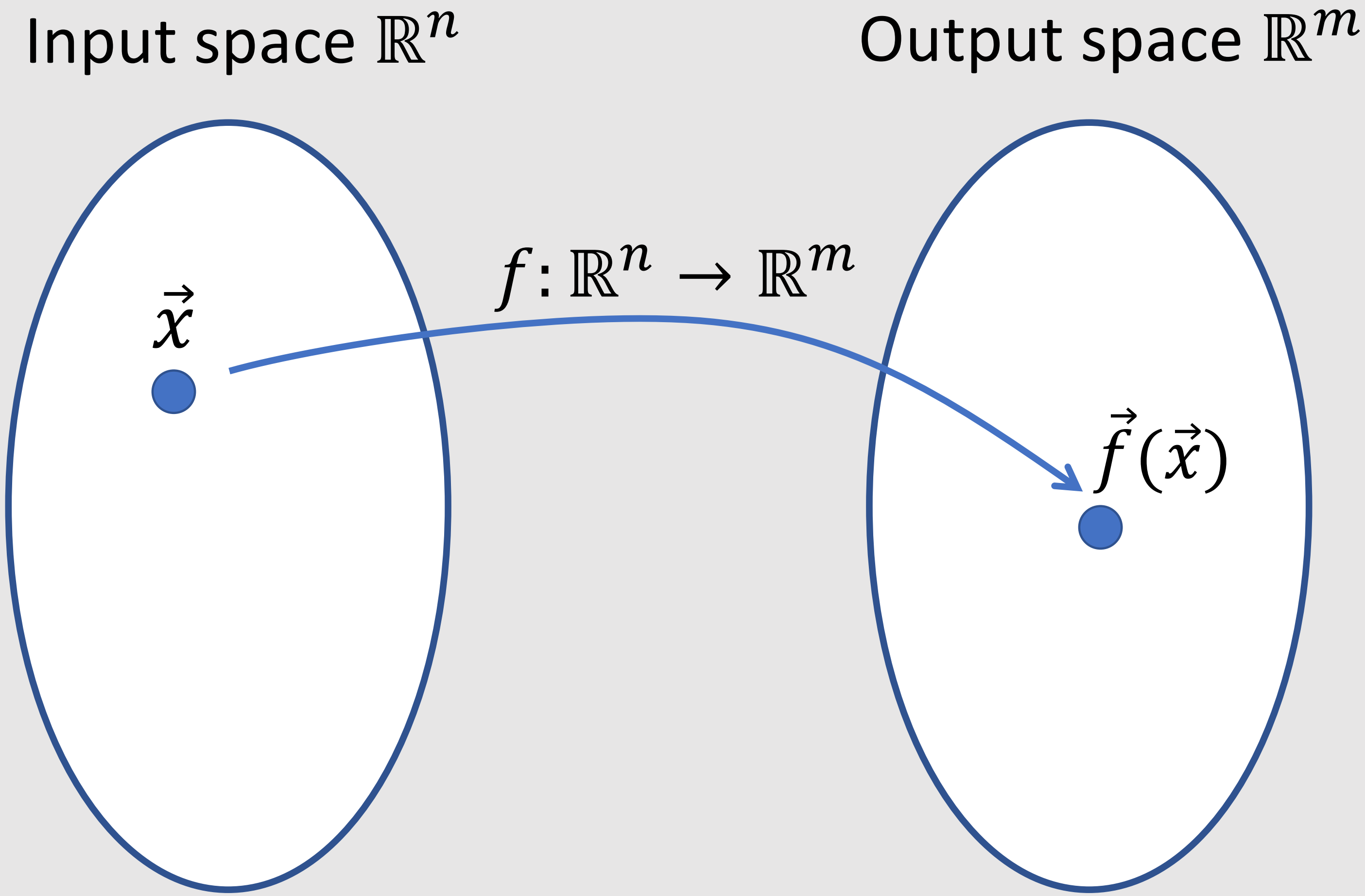
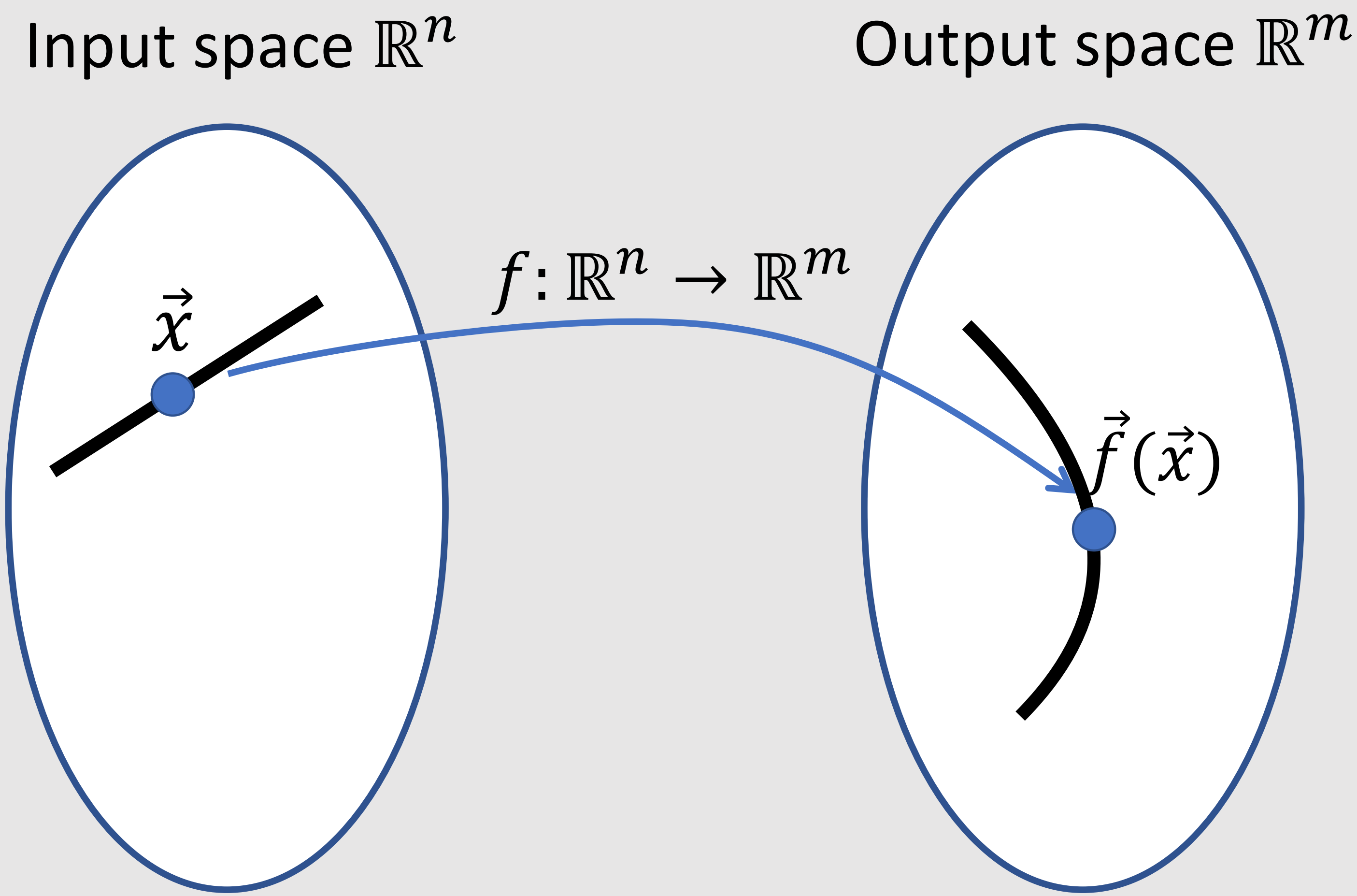
Jacobian & Hessian

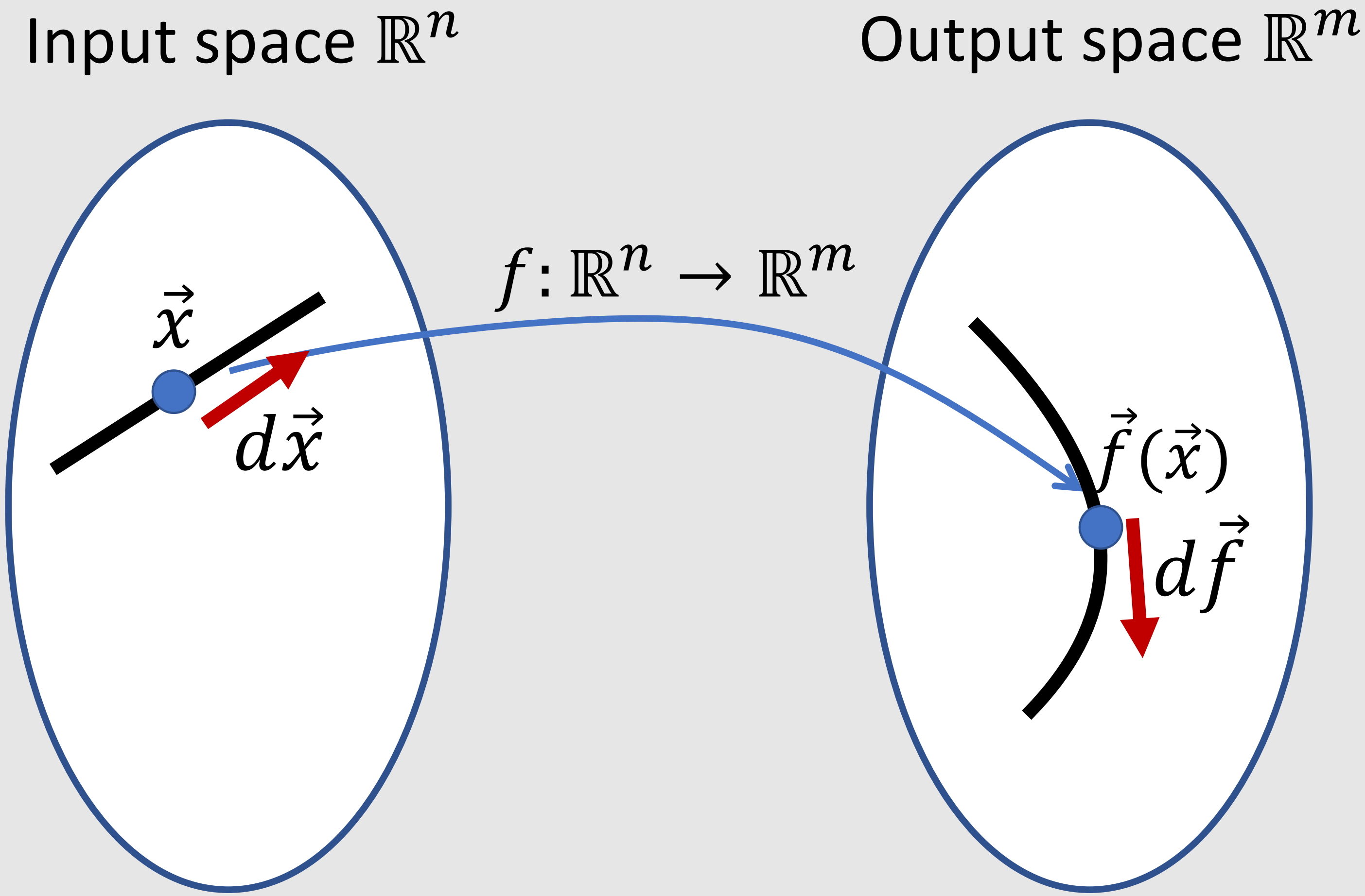
Multivariate Function: High Dimensional Map



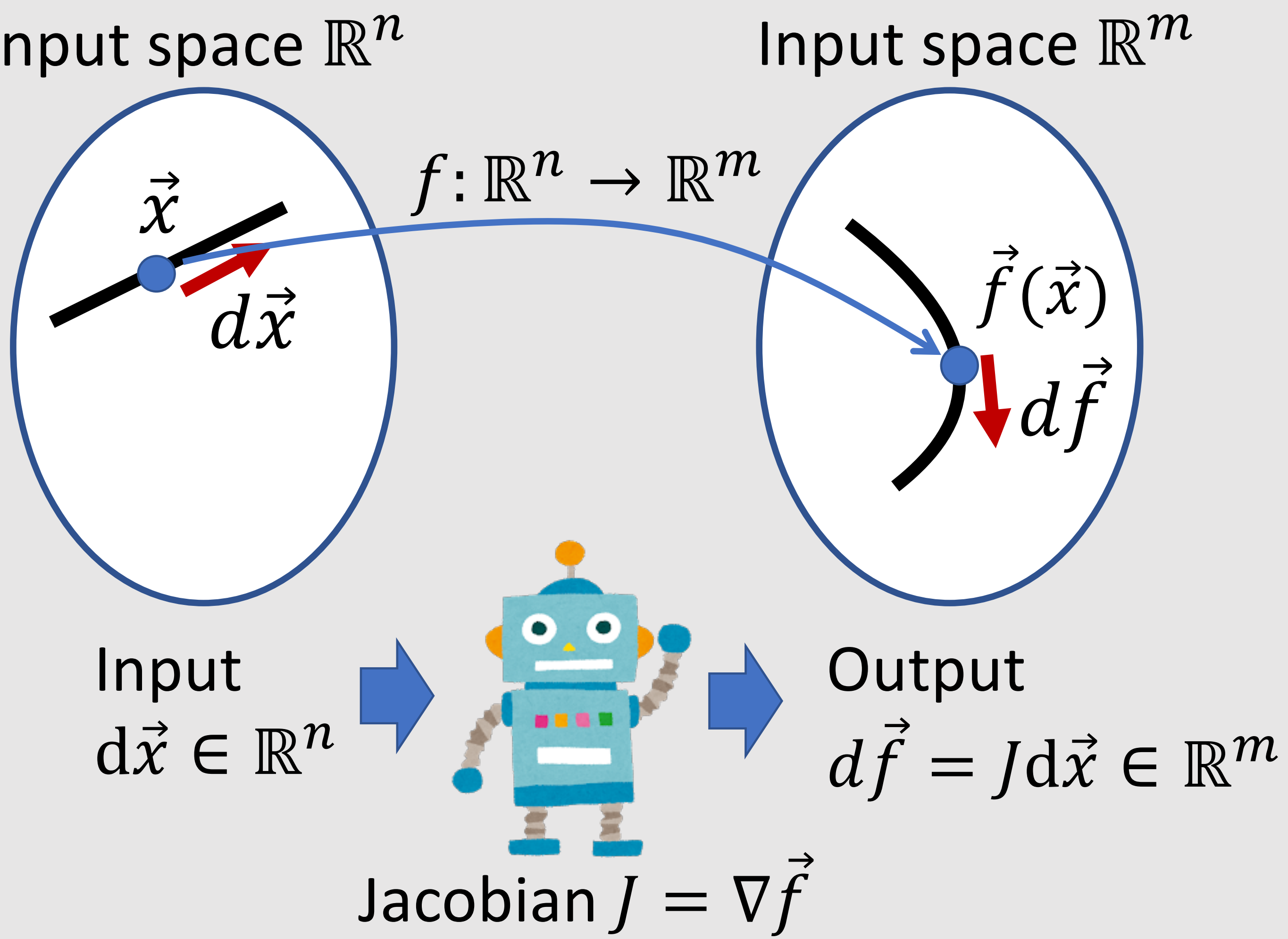
Trajectory of the Function



Differentiation of the Map

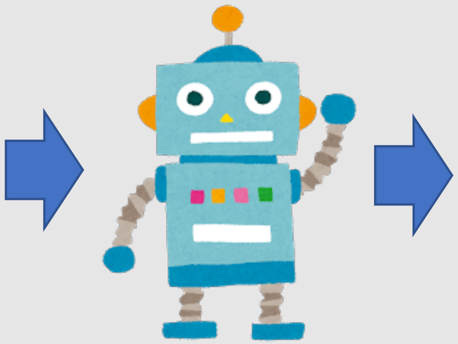


Jacobian Matrix: Gradient of Map



Jacobian Matrix: Gradient of Map

Input
$\mathrm{d}\vec{x} \in \mathbb{R}^n$

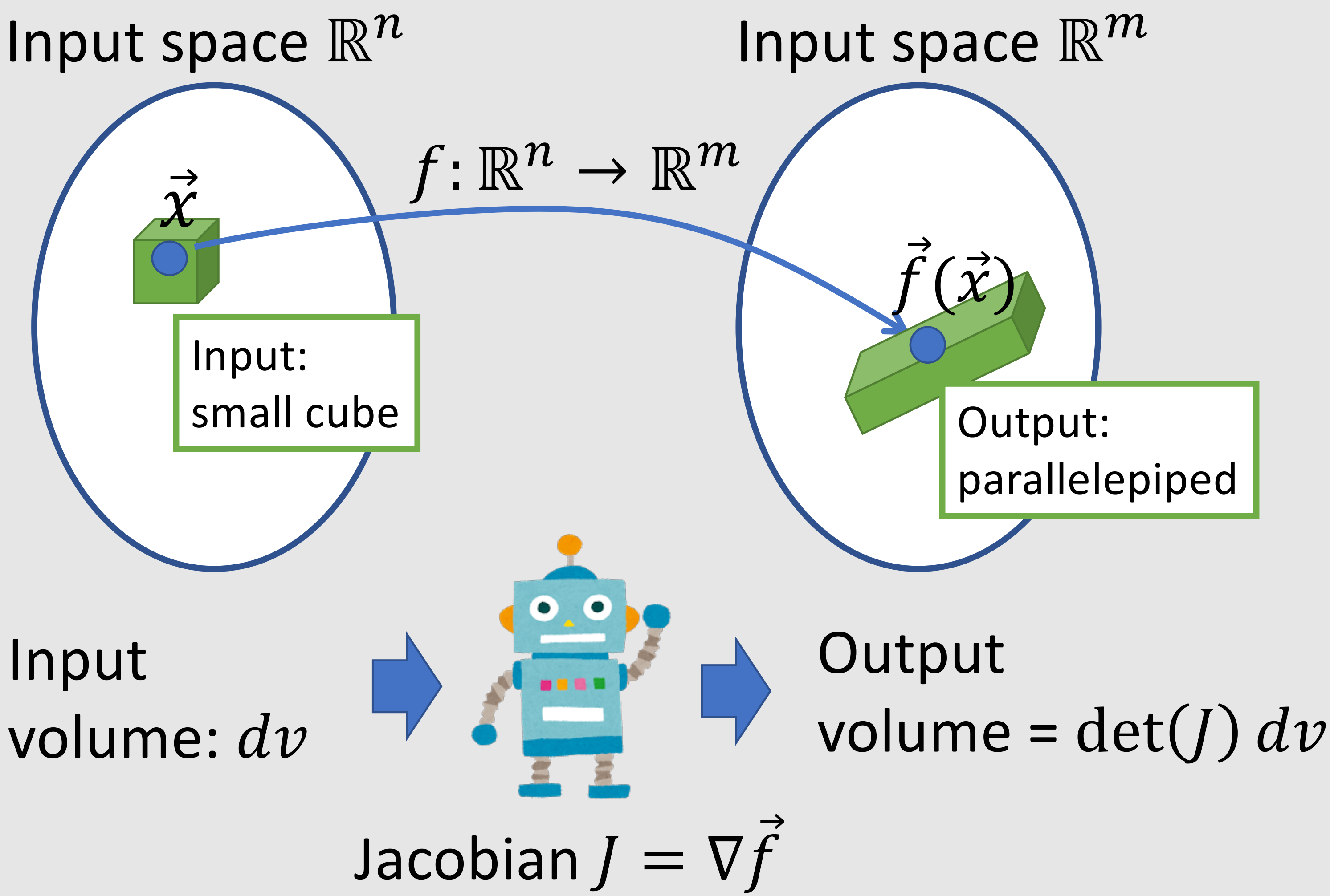


Output
$d\vec{f} = J\mathrm{d}\vec{x} \in \mathbb{R}^m$

Jacobian $J = \nabla \vec{f}$

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial \mathbf{f}}{\partial x_1} & \cdots & \dfrac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \nabla^{\mathrm{T}} f_1 \\ \vdots \\ \nabla^{\mathrm{T}} f_m \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \cdots & \dfrac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_m}{\partial x_1} & \cdots & \dfrac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Jacobian Determinant: Volume Change Ratio



Hessian Matrix: Jacobian Matrix for Gradient

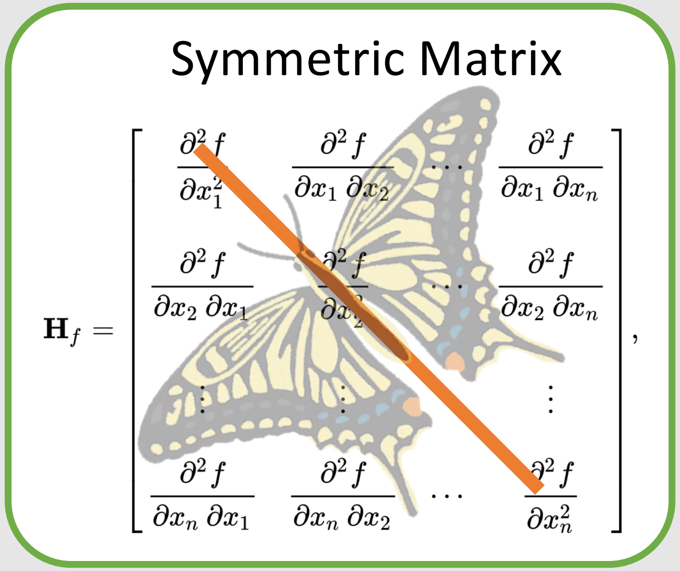
- Second derivative of a scalar function $f(\vec{x})$

$$\boldsymbol{H}_f = \boldsymbol{J}(\nabla f(\vec{x}))$$

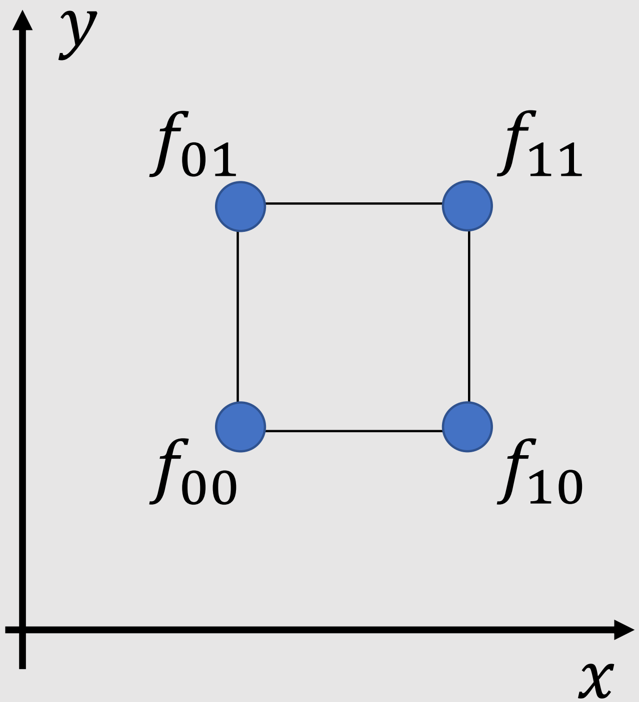
$$(\mathbf{H}_f)_{i,j} = \frac{\partial^2 f}{\partial x_i \, \partial x_j}.$$

$$\mathbf{H}_f = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \, \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \, \partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2 \, \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2 \, \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n \, \partial x_1} & \dfrac{\partial^2 f}{\partial x_n \, \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix},$$

Symmetricity of Hessian



- Hessian is symmetric if $f(\vec{x})$ is continuous



$$\frac{\partial}{\partial x}\left(\frac{\partial f}{\partial y}\right) \approx (f_{11} - f_{10}) - (f_{01} - f_{00})$$

equal

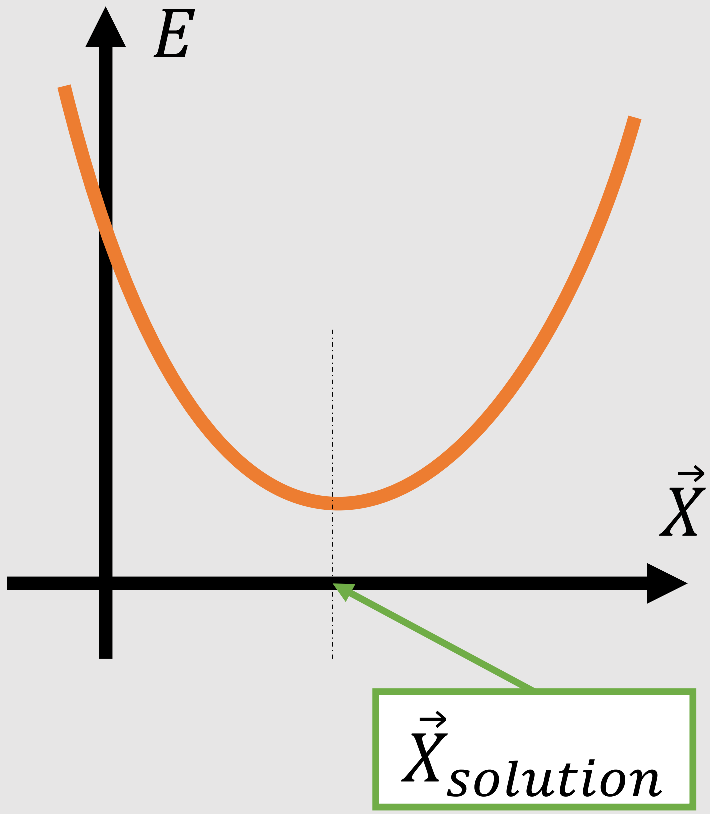
$$\frac{\partial}{\partial y}\left(\frac{\partial f}{\partial x}\right) \approx (f_{11} - f_{01}) - (f_{10} - f_{00})$$

Numerical Optimization

What is Optimization?

- Find input parameter $\vec{X}$ where a cost function $W(\vec{X})$ is minimized

$$\vec{X}_{solution} = \underset{\vec{X}}{\operatorname{argmin}}\, W(\vec{X})$$



Optimization Solve Many Problems

- What typical computer science paper looks like:

a sketch or a parameter sample, and (iii) the reconstruction error of a parameter sample from itself in an auto-encoder fashion. Thus, the combined loss function is defined as:

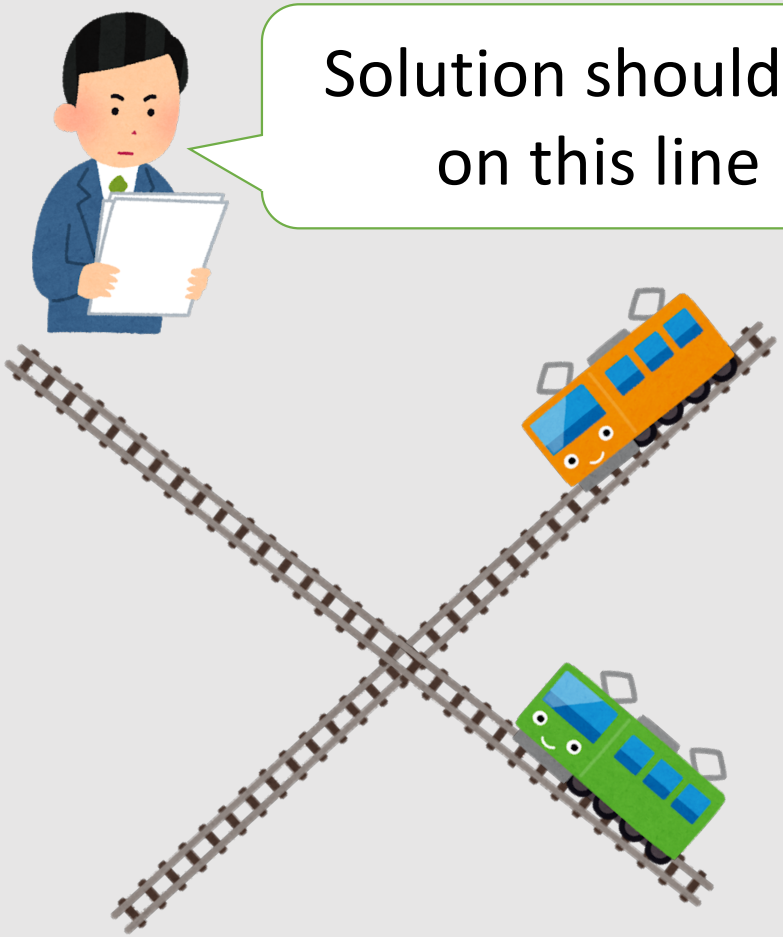
$$\mathscr{L}(\mathbf{P}, \mathbf{M}, \mathbf{S}) = \omega_1 \|P - f_{L2P}(f_{S2L}(S))\|_2 + \omega_2 \|M - f_{L2M}(f_{S2L}(S))\|_2 + \omega_3 \|M - f_{L2M}(f_{P2L}(P))\|_2 + \omega_4 \|P - f_{L2P}(f_{P2L}(P))\|_2, \quad (1)$$

where $\{\omega_1, \omega_2, \omega_3, \omega_4\}$ denote the relative weighting of the individual errors. We set these weights such that the average gradient of

Tuanfeng Y. Wang, Duygu Ceylan, Jovan Popović, and Niloy J. Mitra. 2018. Learning a shared shape space for multimodal garment design. ACM Trans. Graph. 37, 6, Article 203 (November 2018), 13 pages. DOI:https://doi.org/10.1145/3272127.3275074

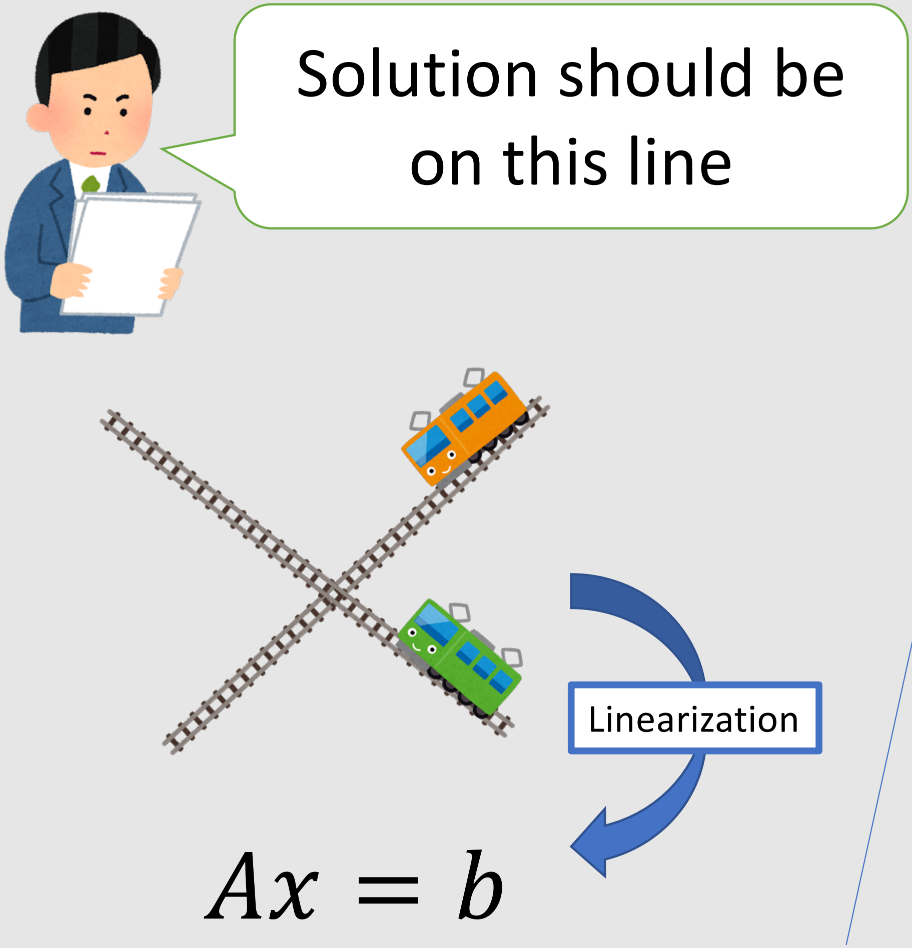
Solving Constraints v.s. Optimization

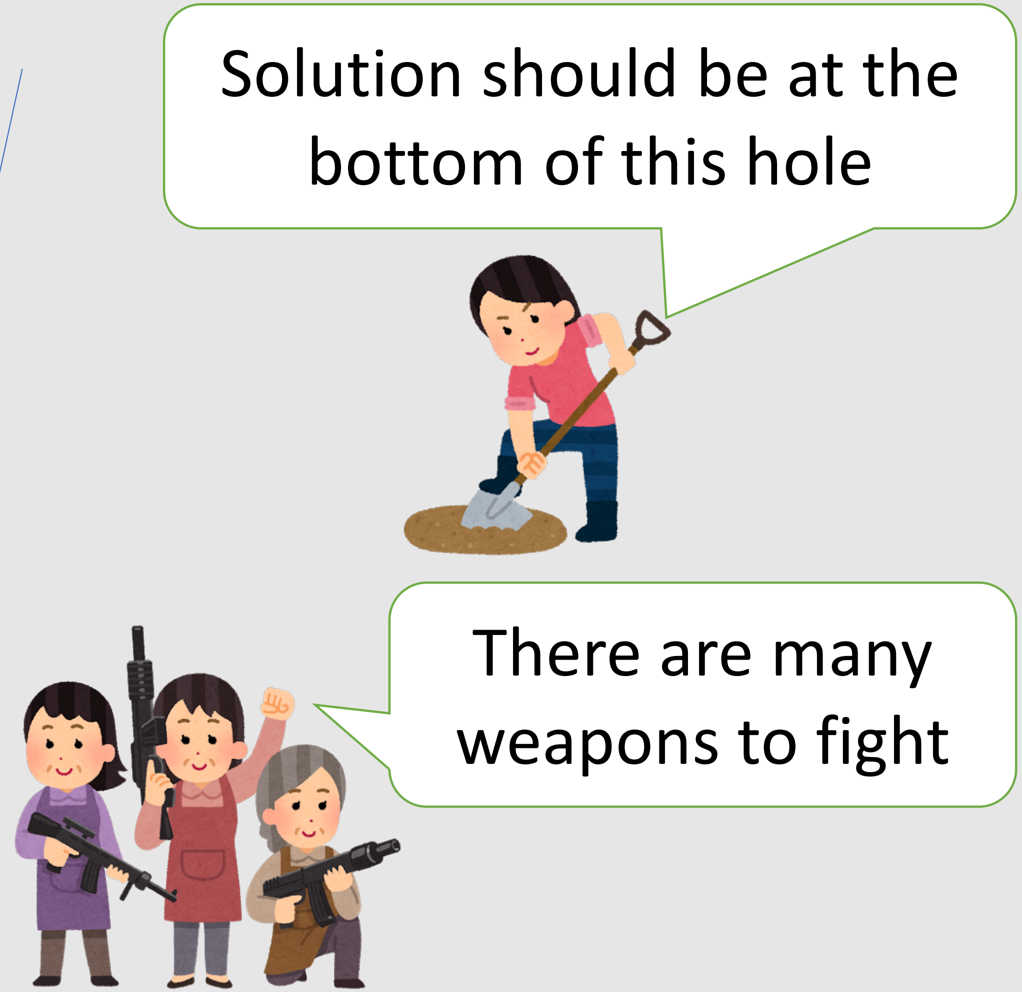
Solution should be on this line



Solution should be at the bottom of this hole

Solving Constraints v.s. Optimization





Three Optimization Approaches

- Stochastic Optimization

Requires value $W(\vec{X})$

- Gradient Descent

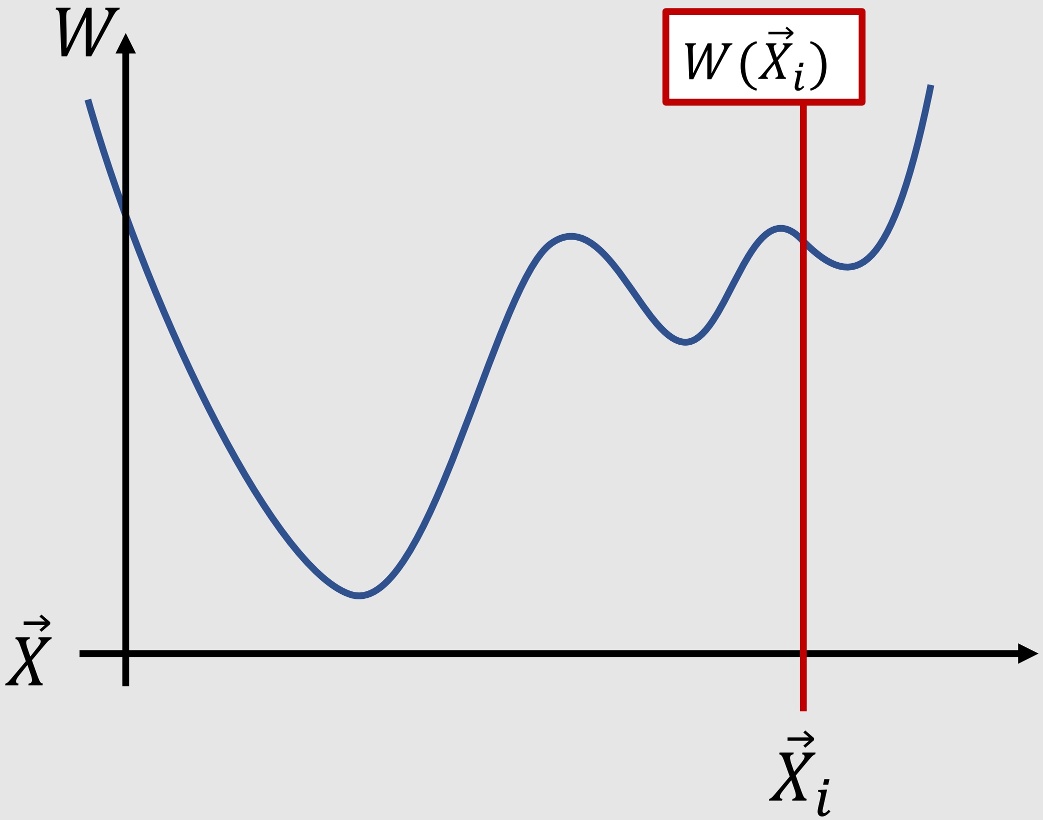
Requires gradient $\nabla W(\vec{X})$

- Newton Method

Requires gradient & hessian
$\nabla W(\vec{X}), \nabla^2 W(\vec{X})$

Stochastic Optimization

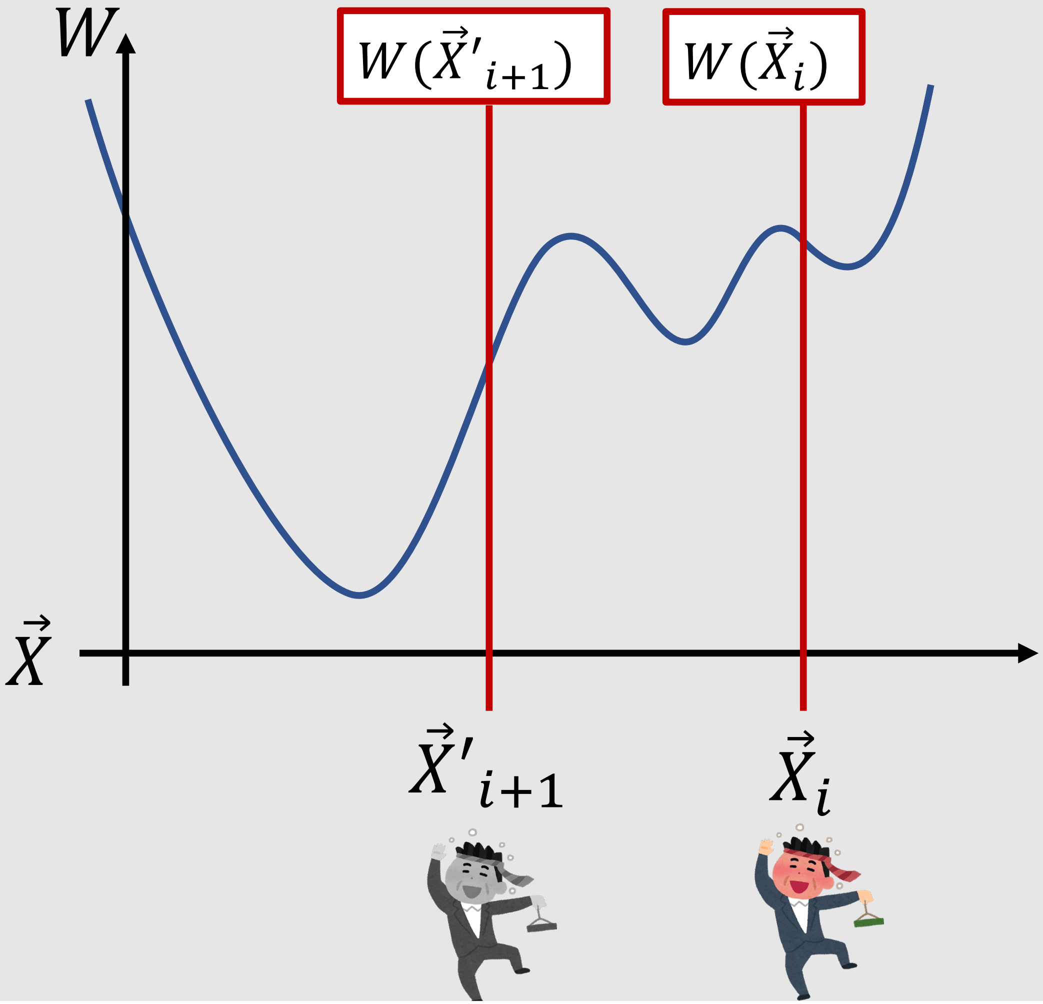
Find Minimum by Random Sampling 1



1. Starting from an initial guess $\vec{X}_0$
2. Evaluate $W(\vec{X}_i)$

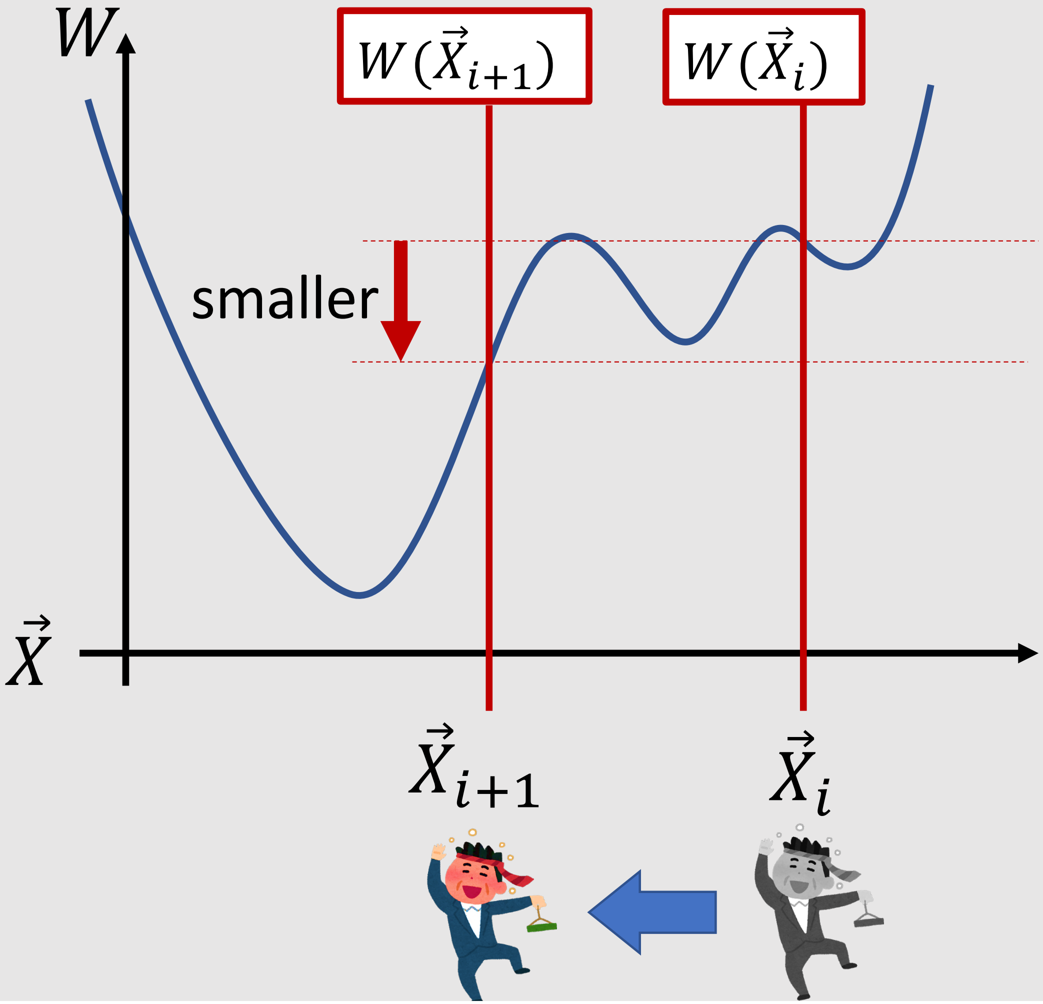
1. Starting from an initial guess $\vec{X}_0$
2. Evaluate $W(\vec{X}_i)$

Find Minimum by Random Sampling 2



1. Starting from an initial guess $\vec{X}_0$
2. Evaluate $W(\vec{X}_i)$
3. Make a candidate $\vec{X}'_{i+1} = \vec{X}_i + Random$
4. Evaluate $W(\vec{X}'_{i+1})$

Find Minimum by Random Sampling 3

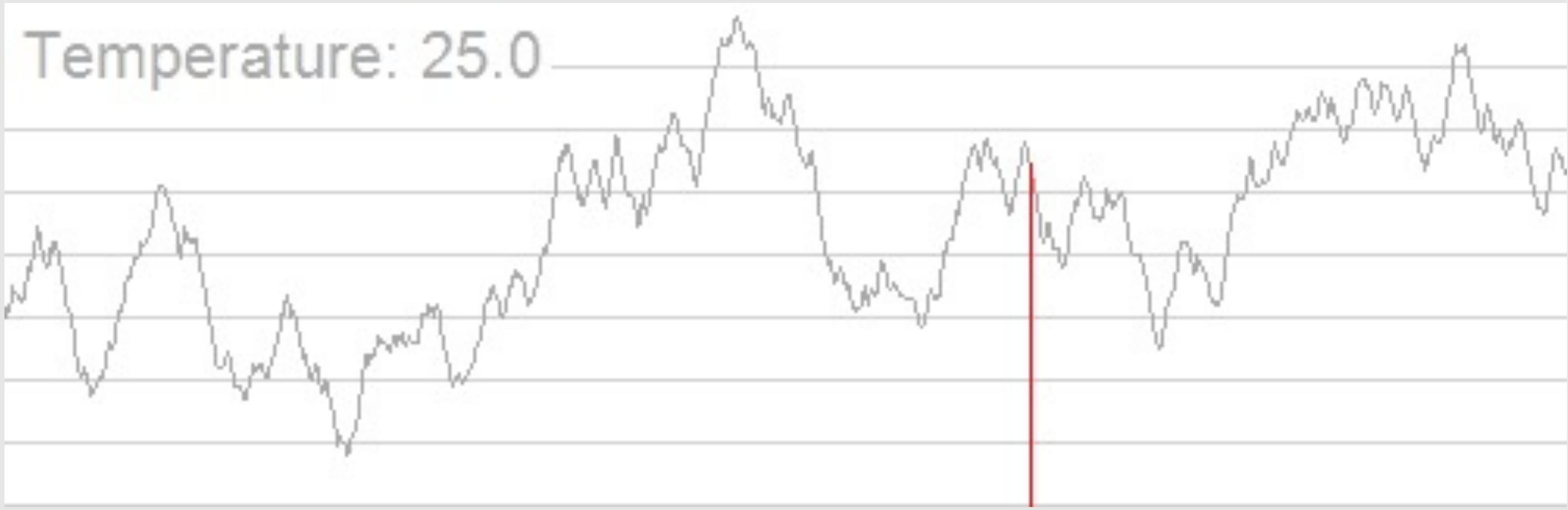


1. Starting from an initial guess $\vec{X}_0$
2. Evaluate $W(\vec{X}_i)$
3. Make a candidate $\vec{X}'_{i+1} = \vec{X}_i + Random$
4. Evaluate $W(\vec{X}'_{i+1})$
5. Move $\vec{X}$ to the candidate if $W(\vec{X}'_{i+1}) < W(\vec{X}_i)$
6. Go to 3

Simulated Annealing Method

Gradually make the random update small during iteration

→ Make the optimization robust to local minima



Credit: Kingpin13 @ Wikipedia

Stochastic Optimization: Blinded Golf

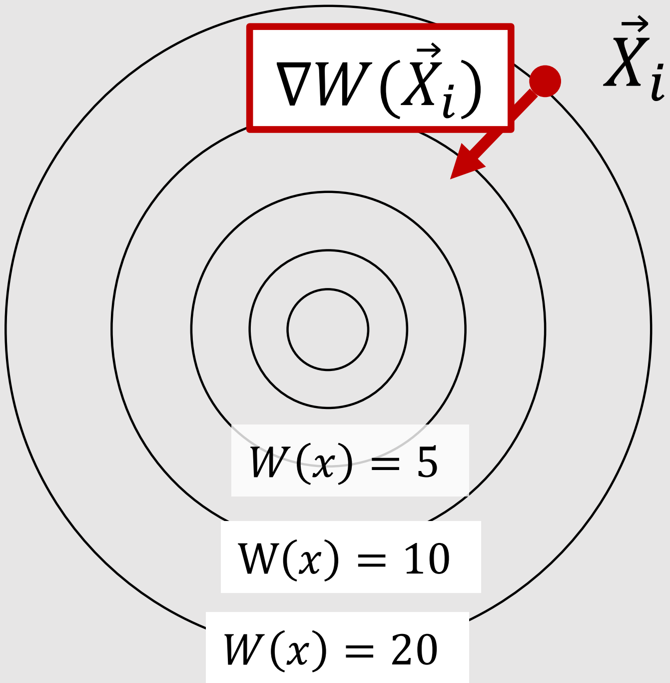
- Optimizer do not know the direction & strength to hit

Gradient Descent Method

最急降下法

Gradient Descent Method

- A.k.a “steepest descent method” or “hill climbing method”



$$\vec{X}_{i+1} = \vec{X}_i - \alpha \nabla W(\vec{X}_i)$$

Learning rate

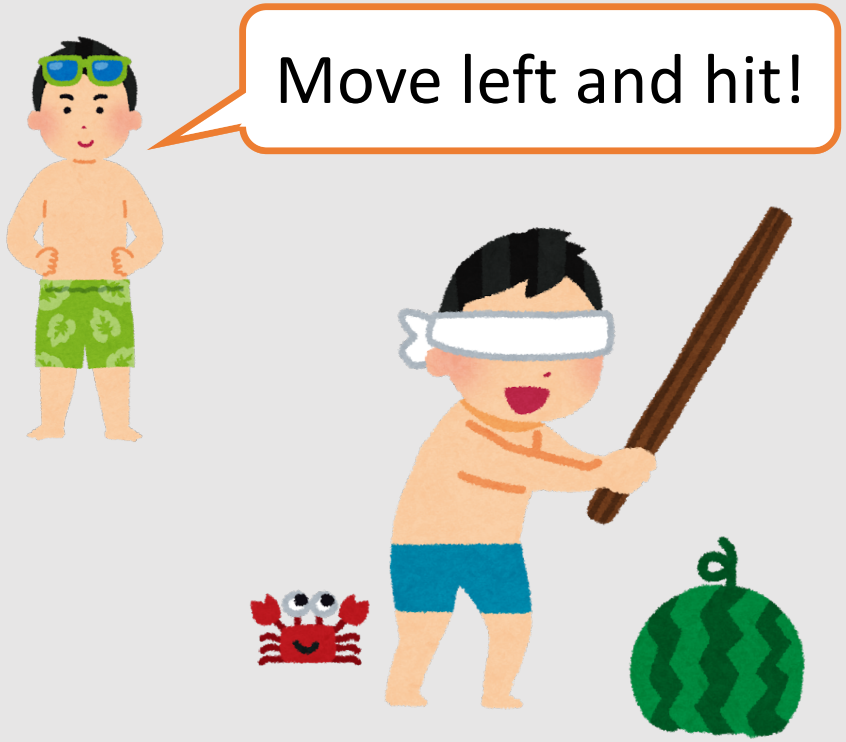
Gradient Descent: Blinded Golf with a Guide

- Optimizer know the direction, but do not know strength to hit

Japanese Version of “Pinata”

- Breaking a watermelon with a stick on a beach

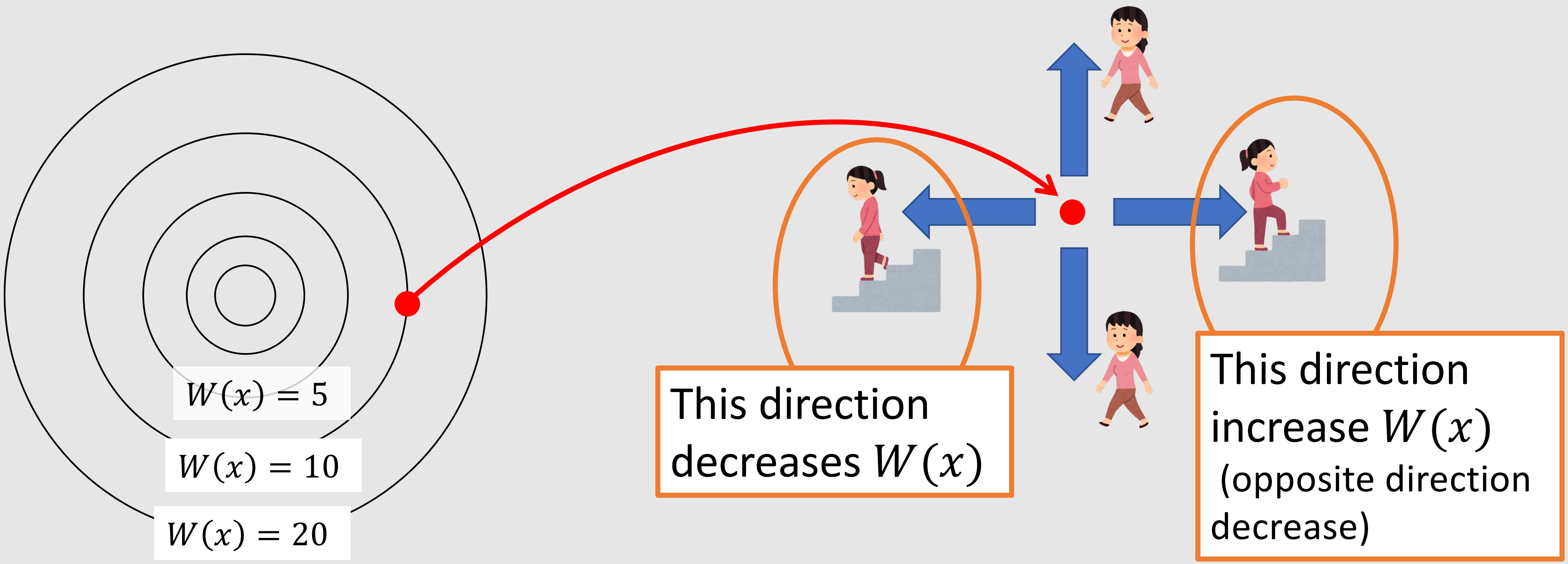
Credit: BeenAroundAWhile @ Wikipedia



Newton-Raphson Method

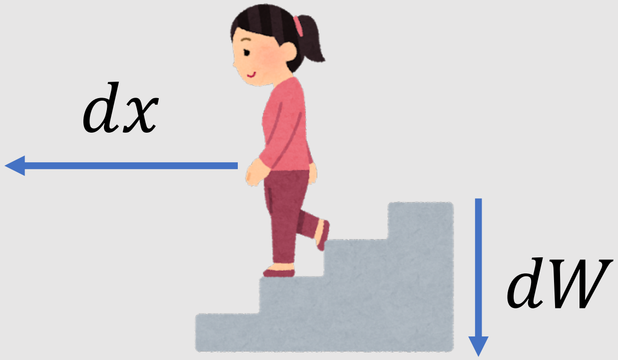
What is not Minimum

- A point is not minimum if there is a direction changing $W(x)$

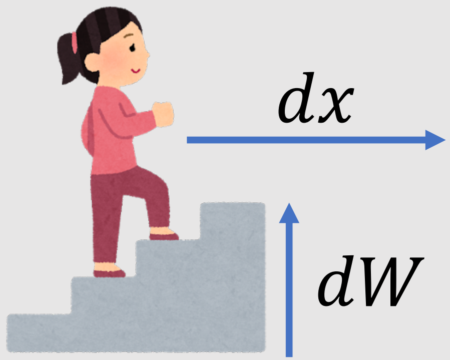


What is not Minimum

- A point is not minimum if $\exists dx \neq 0 \; s.t. \; \nabla W(x) \cdot dx \neq 0$



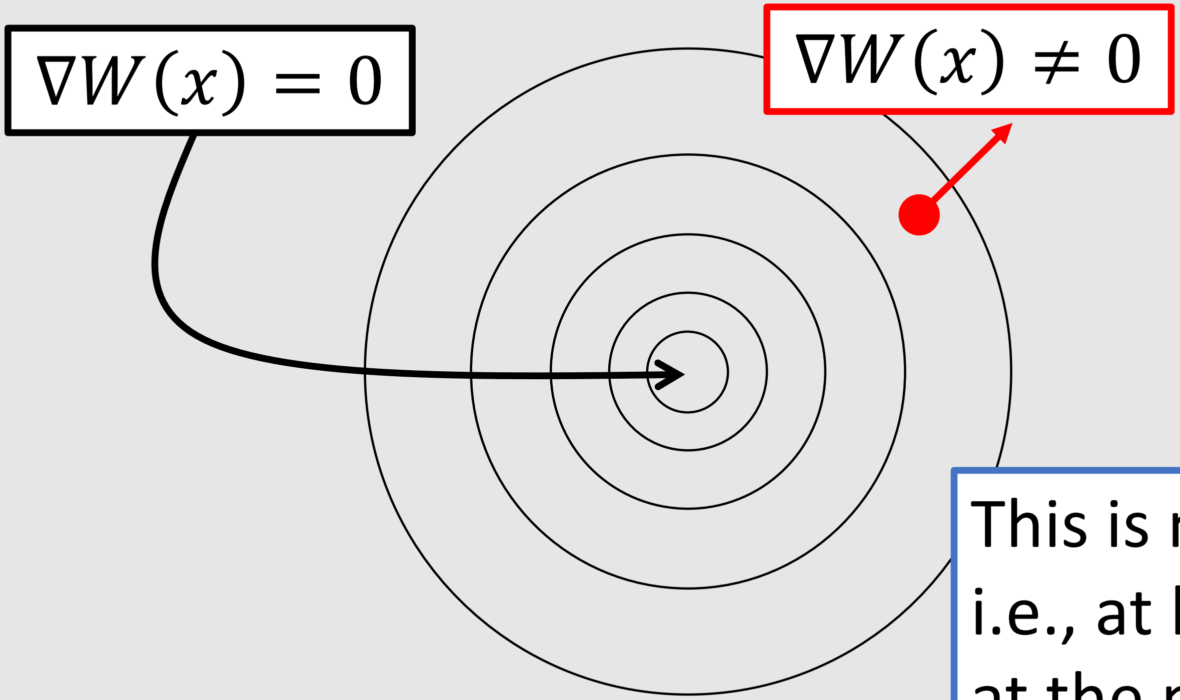
$$dW = \nabla W(x) \cdot dx < 0$$



$$df = \nabla f(x) \cdot dx > 0$$

What Might be Minimum: Zero Gradient

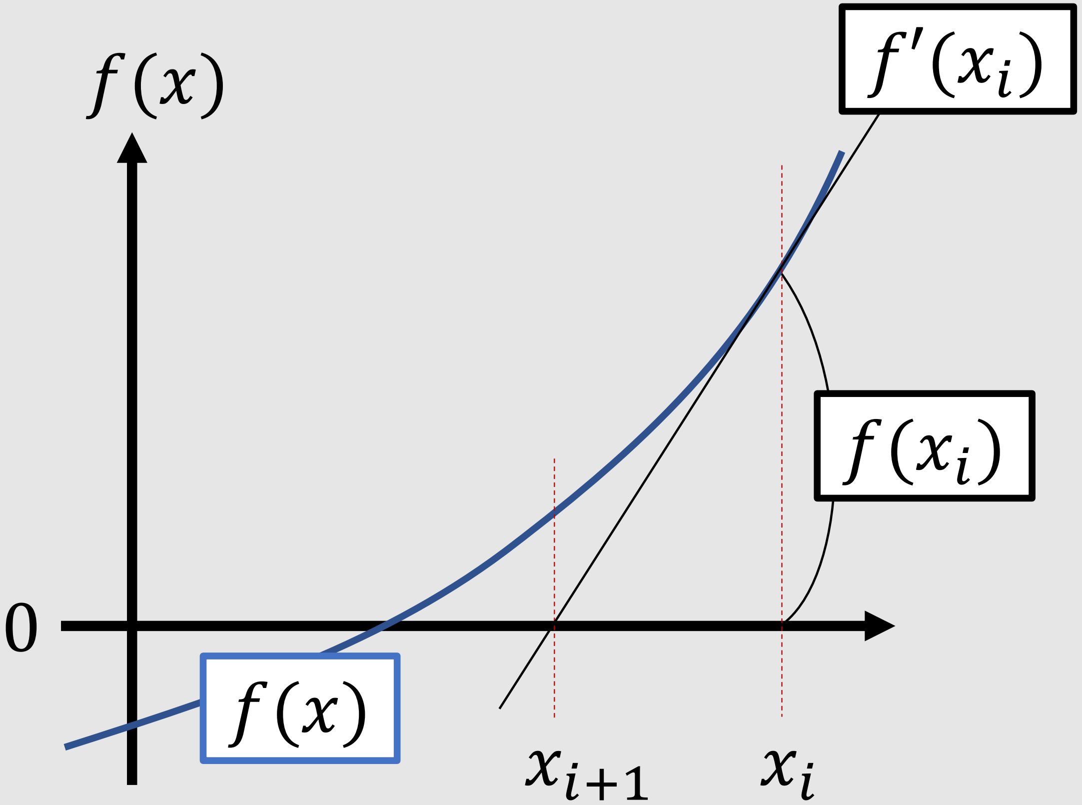
$$\nabla W(x) = 0$$



Find the root of $\nabla W(x)$

This is necessary condition (not sufficient)
i.e., at least $\nabla W(x)$ needs to be zero at the minimum

Finding the Root of a Scalar Function

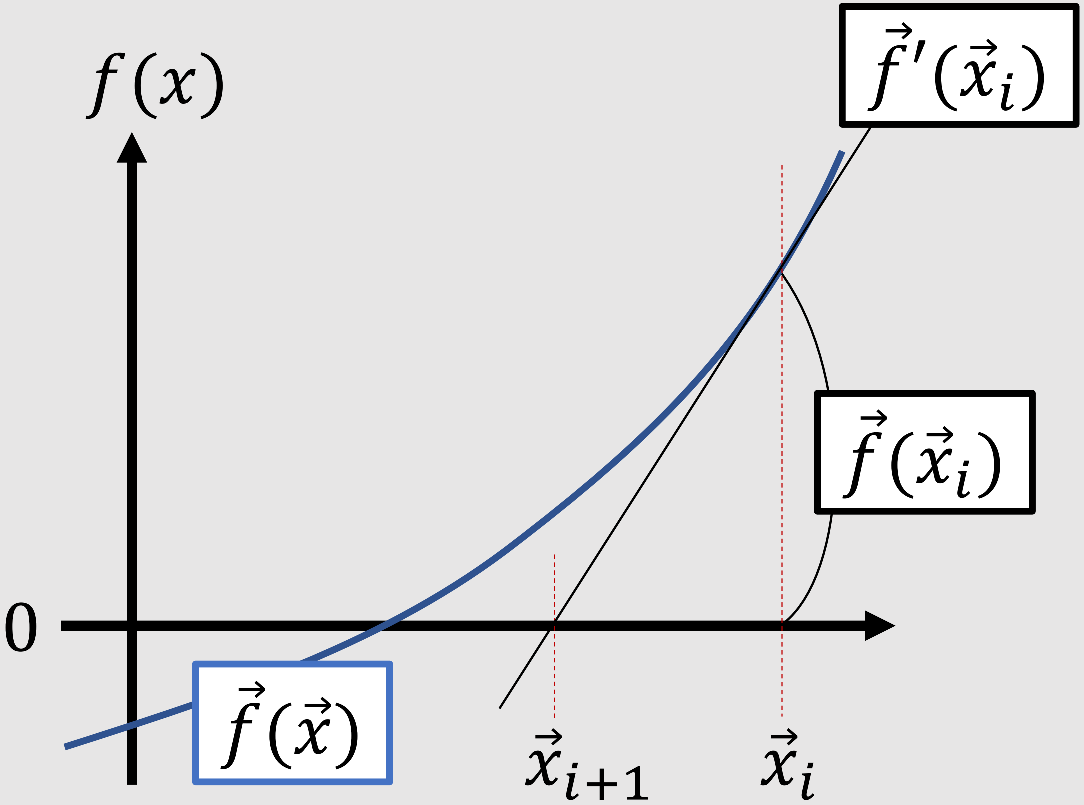


To find x where $f(x) = 0$

Iterate:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Finding the Root of a Multivariate Function



To find $\vec{x}$ where $\vec{f}(\vec{x}) = 0$

Iterate:

$$\vec{x}_{i+1} = \vec{x}_i - \left[\nabla \vec{f}(\vec{x}_i)\right]^{-1} f(\vec{x}_i)$$

Jacobian matrix

* $\nabla \vec{f}(\vec{x}_i)$ need to be invertible

Finding the Root of Gradient $\nabla W(x) = 0$

- Gradient of gradient is called hessian

$\vec{f} = \nabla W$

To find $\vec{x}$ where $\vec{f}(\vec{x}) = 0$

Iterate:

$$\vec{x}_{i+1} = \vec{x}_i - \left[\nabla \vec{f}(\vec{x}_i)\right]^{-1} f(\vec{x}_i)$$

To find $\vec{x}$ where $\nabla W(\vec{x}) = 0$

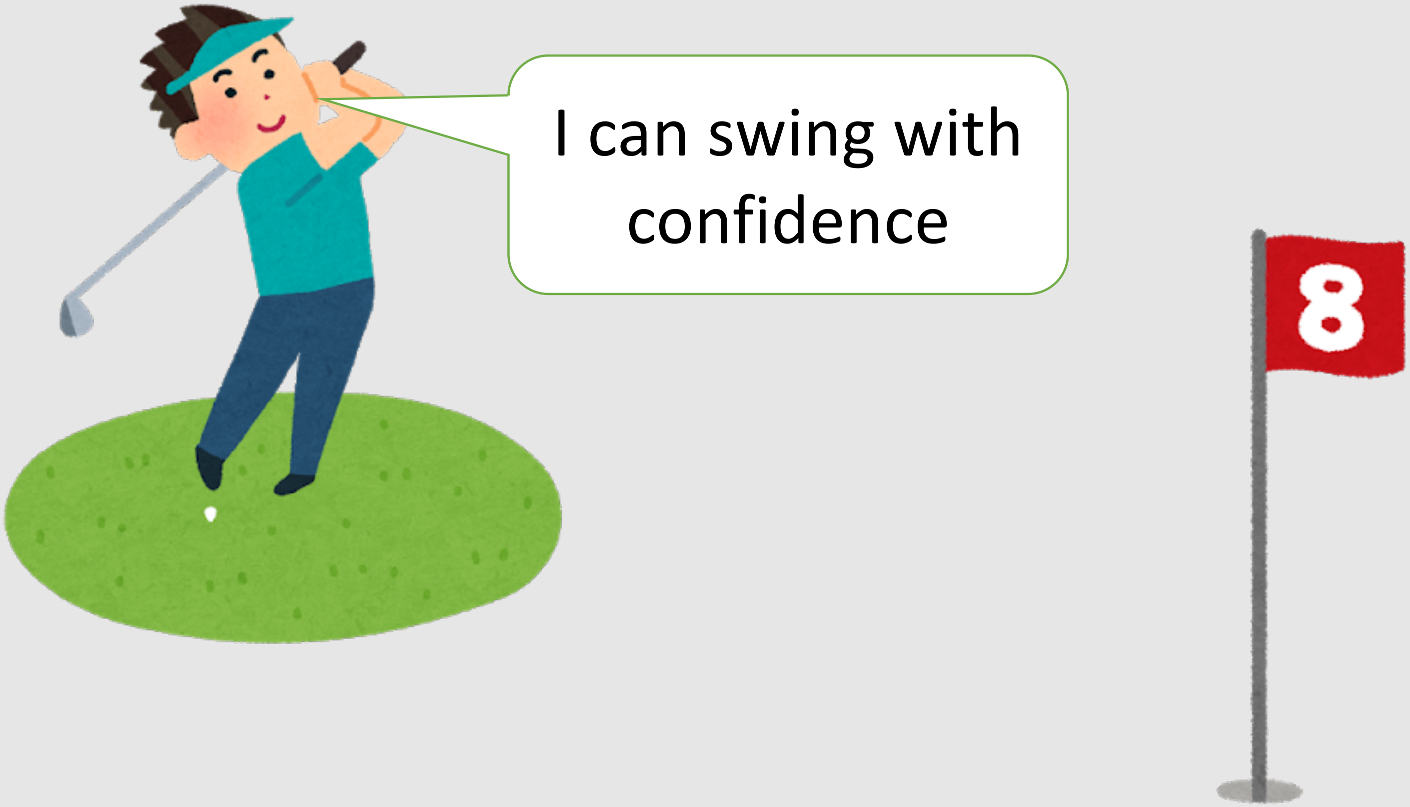
Iterate:

$$\vec{x}_{i+1} = \vec{x}_i - [\nabla^2 W(\vec{x}_i)]^{-1} \nabla W(\vec{x}_i)$$

hessian

Gradient Descent: Golf without Blindfold

- Optimizer know the direction & strength to hit



Comparison of Three Approaches

Stochastic Optimization

☺ Only evaluation of a function is necessary

☹ Very slow
☹ Not scalable
☹ Heuristics

Gradient Descent

☺ Only gradient is necessary
☺ Very scalable

☹ Slow
☹ Parameter tuning

Newton Method

☺ Very fast for almost quadratic problem

☹ Require Hessian
☹ Complicated Code

Advanced Topics

- Stochastic Optimization
  - Metropolis Hasting Method
  - Meta-heuristic Optimization (Particle Swarm, Evolutionary Algorithm)

- Gradient Descent
  - Stochastic Gradient Descent

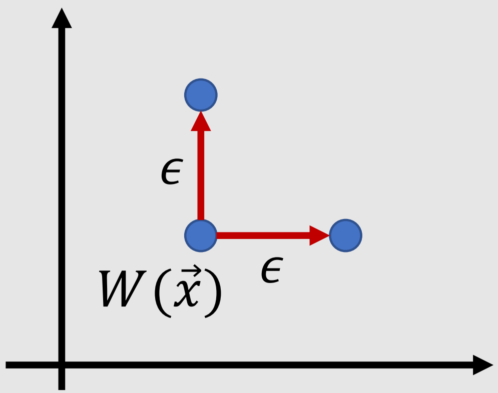
- Newton Method
  - Levenberg–Marquardt method
  - L-BFGS method

Typical Mistakes in Optimization

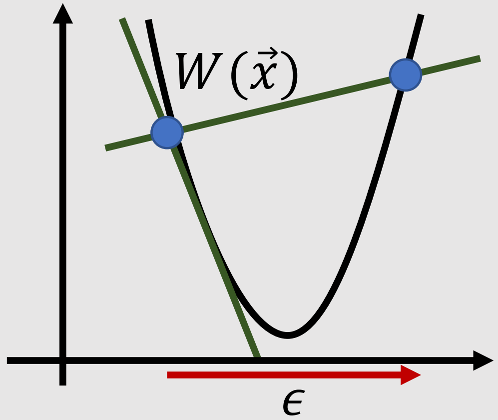
- Don't use numerical difference in gradient or Newton method

$$(\nabla W)_i = \frac{W(\vec{x} + \epsilon \vec{e}_i) - W(\vec{x})}{\epsilon}$$

Not scalable for large DoFs



Inaccurate around convergence



End