Mathematical Foundations of Data Analysis (MFDA) - II

Boqiang Huang

huang@math.uni-koeln.de

Institute of Mathematics, University of Cologne, Germany

2019.04.09-11

Chapter 1. Basics

- 1. Probablity and Information Theory
    - 1.1. Basic defintions and rules in probability theory
    - 1.2. Basic definitons and rules in information theory
- 2. Numerical computation
    - 2.1 Basic knowledge in numerical computation
    - 2.2 Basic knowledge in optimizations
        - 2.2.1 Gradient-based optimization
        - 2.2.2 Constrained optimization
- 3. Application: statitical model based data denoising (in tutorial after lecture this Thursday)

Reference:

[1] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, Chapter 3-4, MIT Press, 2016.
[2] A. Antoniou, W.-S. Lu, Practical optimization: algorithms and engineering applications, Springer, 2007.
[3] I. Cohen, Noise spectrum estimation in adverse environments: improved minima controlled recursive averaging, IEEE Trans. Acoust., Speech, Signal Processing, vol. 11, no. 5, pp. 466-475, 2003.

2. Numerical Computation

❖ If there is no analytical solution, approximate/estimate it via iteratively numerical process

2.1 Overflow and Underflow

- Underflow occurs when numbers near zero are rounded to zero

  rounding error, avoid division by zero, …

- Overflow occurs when numbers with large magnitude are approximated as ∞ or $-\infty$

  Example: the softmax function

$$\text{softmax}(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}$$

2. Numerical Computation

- If there is no analytical solution, approximate/estimate it via iteratively numerical process

2.2 Poor Conditioning

how rapidly a function changes with respect to small changes in its inputs

rounding errors in the inputs can result in large changes in the output

Example: consider a function $f(\boldsymbol{x}) = \boldsymbol{A}^{-1}\boldsymbol{x}$

condition number of matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ defined as ratio of the largest and smallest eigenvalue

$$\max_{i,j} \left| \frac{\lambda_i}{\lambda_j} \right|$$

when it is large, matrix inversion is sensitive to the input

Remark: poorly conditioned matrices amplify pre-existing errors when we multiply by its inverse

2. Numerical Computation

❖ If there is no analytical solution, approximate/estimate it via iteratively numerical process

2.3 Gradient-Based Optimization

$$\boldsymbol{x}^* = \arg\min f(\boldsymbol{x}) \qquad \boldsymbol{x} \in \mathbb{R}^n$$

minimizer objective/cost/loss/error function

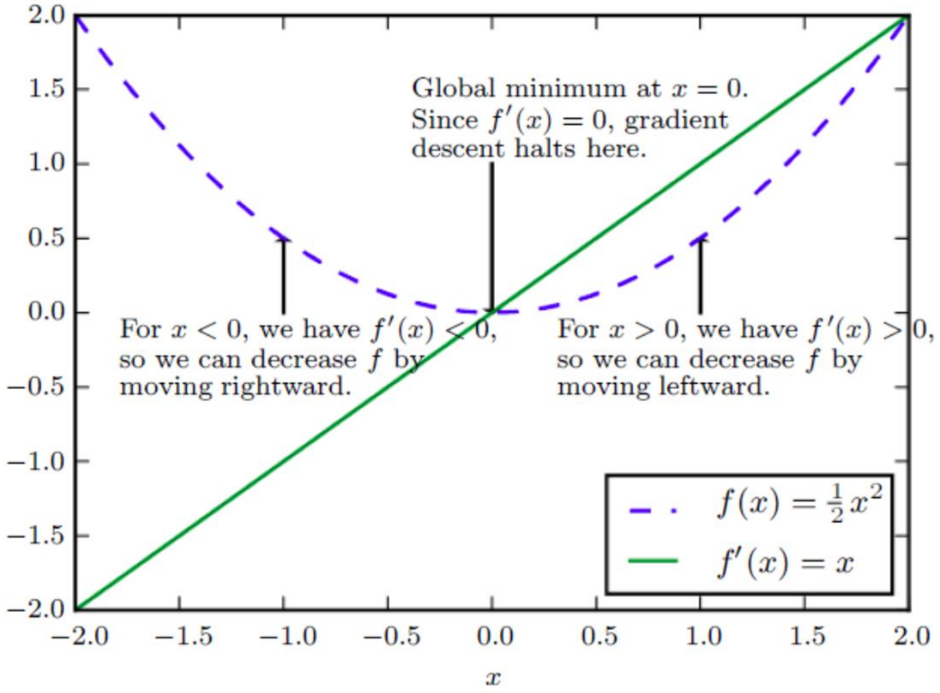
2. Numerical Computation

❖ If there is no analytical solution, approximate/estimate it via iteratively numerical process

2.3 Gradient-Based Optimization

$$\boldsymbol{x}^* = \arg\min f(\boldsymbol{x}) \qquad \boldsymbol{x} \in \mathbb{R}^n$$

minimizer objective/cost/loss/error function



Gradient Descent Method

consider a real-valued function $y = f(x)$

its derivative gives the slope of $f(x)$ at point x

$$f(x + \epsilon) \approx f(x) + \epsilon f'(x)$$

$f(x - \epsilon\,\mathrm{sign}(f'(x)))$ is less than $f(x)$ for small enough ϵ

we can reduce $f(x)$ by moving x in small steps with opposite sign of the derivative

2. Numerical Computation

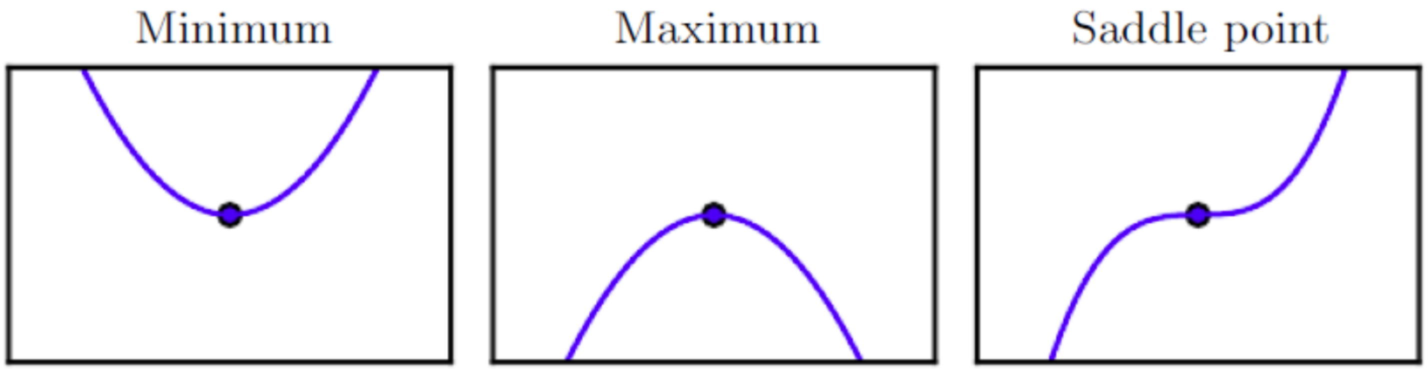
❖ If there is no analytical solution, approximate/estimate it via iteratively numerical process

2.3 Gradient-Based Optimization

$$\boldsymbol{x}^* = \arg\min f(\boldsymbol{x}) \qquad \boldsymbol{x} \in \mathbb{R}^n$$

minimizer objective/cost/loss/error function

critical/stationary point exists at $f'(x) = 0$



2. Numerical Computation

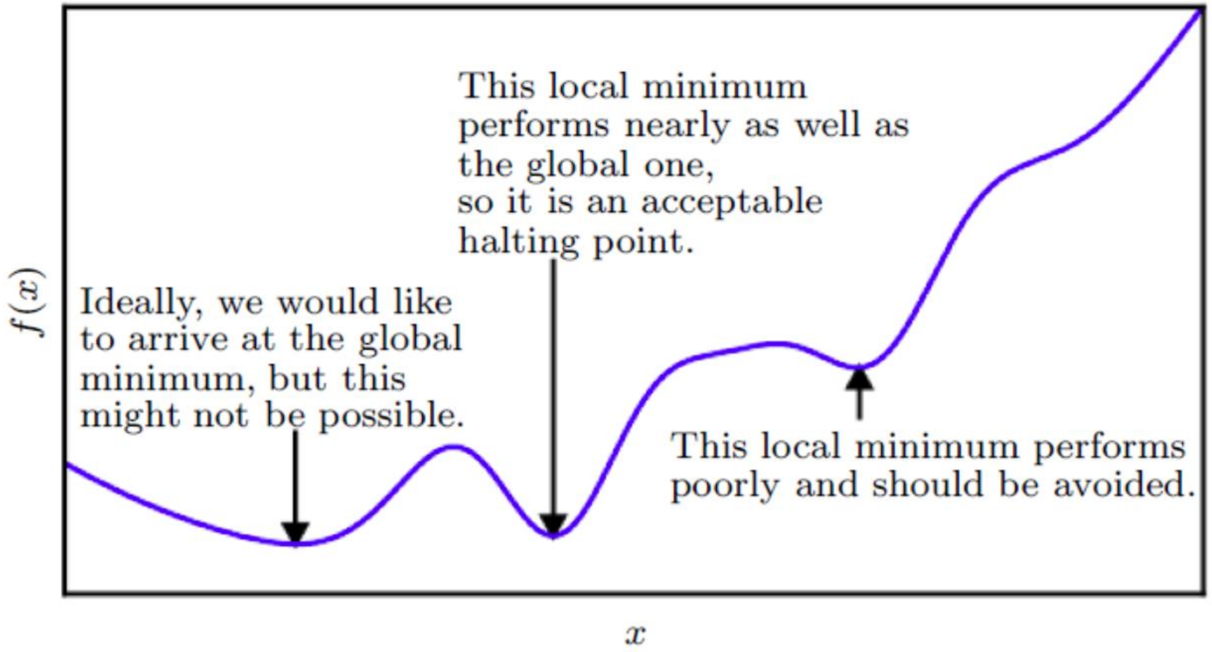
❖ If there is no analytical solution, approximate/estimate it via iteratively numerical process

2.3 Gradient-Based Optimization

$$\boldsymbol{x}^* = \arg\min f(\boldsymbol{x}) \qquad \boldsymbol{x} \in \mathbb{R}^n$$

minimizer   objective/cost/loss/error function

critical/stationary point exists at $f'(x) = 0$



2. Numerical Computation

❖ If there is no analytical solution, approximate/estimate it via iteratively numerical process

2.3 Gradient-Based Optimization

$$\boldsymbol{x}^* = \arg\min f(\boldsymbol{x}) \qquad \boldsymbol{x} \in \mathbb{R}^n$$

minimizer objective/cost/loss/error function

critical/stationary point exists at $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix}^T = 0$

gradient

directional derivative in direction $\boldsymbol{u}$ (a unit vector) is the slope of the function f in direction $\boldsymbol{u}$, $\boldsymbol{u} \in \mathbb{R}^n$

$$\nabla_{\boldsymbol{u}} f(\boldsymbol{x}) = \lim_{\alpha \to 0} \frac{f(\boldsymbol{x} + \alpha \boldsymbol{u}) - f(\boldsymbol{x})}{\alpha} \qquad \left. \frac{\partial f(\boldsymbol{x} + \alpha \boldsymbol{u})}{\partial \alpha} \right|_{\alpha = 0} = \boldsymbol{u}^T \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$$

2. Numerical Computation

❖ If there is no analytical solution, approximate/estimate it via iteratively numerical process

2.3 Gradient-Based Optimization

$$\boldsymbol{x}^* = \arg\min f(\boldsymbol{x}) \qquad \boldsymbol{x} \in \mathbb{R}^n$$

minimizer objective/cost/loss/error function

critical/stationary point exists at $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \left[\frac{\partial f}{\partial x_1} \quad \cdots \quad \frac{\partial f}{\partial x_n}\right]^T = 0$

gradient

directional derivative in direction $\boldsymbol{u}$ (a unit vector) is the slope of the function f in direction $\boldsymbol{u}$, $\boldsymbol{u} \in \mathbb{R}^n$

$$\nabla_{\boldsymbol{u}} f(\boldsymbol{x}) = \lim_{\alpha \to 0} \frac{f(\boldsymbol{x} + \alpha \boldsymbol{u}) - f(\boldsymbol{x})}{\alpha} \qquad \left.\frac{\partial f(\boldsymbol{x} + \alpha \boldsymbol{u})}{\partial \alpha}\right|_{\alpha=0} = \boldsymbol{u}^T \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$$

$$\min_{\boldsymbol{u},\, \boldsymbol{u}^T \boldsymbol{u}=1} \boldsymbol{u}^T \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \min_{\boldsymbol{u},\, \boldsymbol{u}^T \boldsymbol{u}=1} \|\boldsymbol{u}\|_2 \|\nabla_{\boldsymbol{x}} f(\boldsymbol{x})\|_2 \cos\theta \qquad \min_{\boldsymbol{u},\, \boldsymbol{u}^T \boldsymbol{u}=1} \cos\theta$$

$\boldsymbol{u} = -\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$ steepest/gradient descent

2. Numerical Computation

❖ If there is no analytical solution, approximate/estimate it via iteratively numerical process

2.3 Gradient-Based Optimization

$$\boldsymbol{x}^* = \arg\min f(\boldsymbol{x}) \qquad \boldsymbol{x} \in \mathbb{R}^n$$

minimizer objective/cost/loss/error function

critical/stationary point exists at $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \left[\frac{\partial f}{\partial x_1} \quad \cdots \quad \frac{\partial f}{\partial x_n}\right]^T = 0$

gradient

directional derivative in direction $\boldsymbol{u}$ (a unit vector) is the slope of the function f in direction $\boldsymbol{u}$, $\boldsymbol{u} \in \mathbb{R}^n$

$\boldsymbol{u} = -\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$ steepest/gradient descent

iteration: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \epsilon \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k)$

ϵ : learning rate

❖ If there is no analytical solution, approximate/estimate it via iteratively numerical process

2.3 Gradient-Based Optimization

$$x^* = \arg\min f(x) \qquad x$$

minimizer objective/cost/loss/

critical/stationary point exists at $\nabla_x f(x) = \left[\frac{\partial f}{\partial x_1} \quad \cdots \quad \frac{\partial f}{\partial x_n}\right]^T$

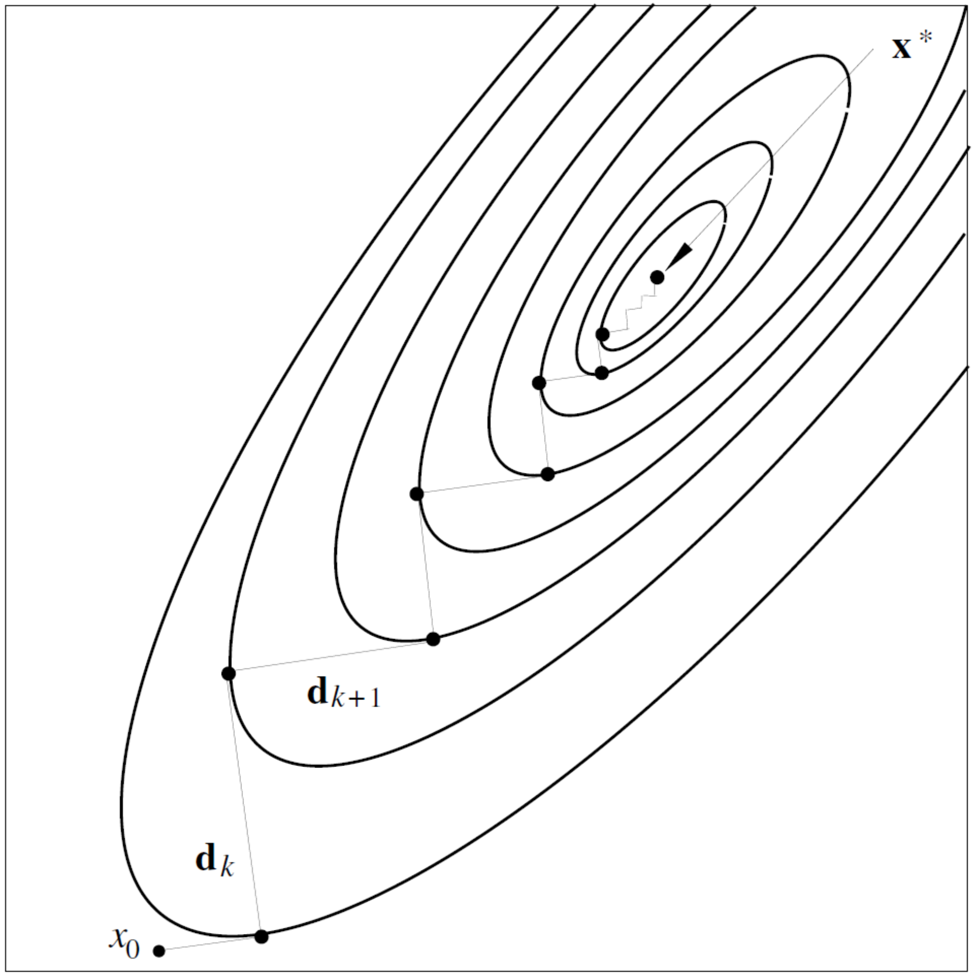
gradient

directional derivative in direction $\boldsymbol{u}$ (a unit vector) is the slop

$\boldsymbol{u} = -\nabla_x f(x)$ steepest/gradient descent

iteration: $x_{k+1} = x_k - \epsilon \nabla_x f(x_k)$

ϵ : learning rate



2. Numerical Computation

❖ If there is no analytical solution, approximate/estimate it via iteratively numerical process

2.3 Gradient-Based Optimization

$$\boldsymbol{x}^* = \arg\min f(\boldsymbol{x}) \qquad \boldsymbol{x} \in \mathbb{R}^n$$

minimizer objective/cost/loss/error function

critical/stationary point exists at $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix}^T = 0$

gradient

directional derivative in direction $\boldsymbol{u}$ (a unit vector) is the slope of the function f in direction $\boldsymbol{u}$, $\boldsymbol{u} \in \mathbb{R}^n$

$\boldsymbol{u} = -\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$ steepest/gradient descent

iteration: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \epsilon_k \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k)$

line search: $\epsilon_k = \operatorname{argmin} f\left(\boldsymbol{x}_k + \epsilon \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k)\right)$

ϵ_k: learning rate

2. Numerical Computation

❖ If there is no analytical solution, approximate/estimate it via iteratively numerical process

2.3 Gradient-Based Optimization

2.3.1 Beyond the Gradient: Jacobian and Hessian Matrices

$\boldsymbol{f}\colon \mathbb{R}^m \to \mathbb{R}^n$ Jacobian Matrix $\boldsymbol{J} \in \mathbb{R}^{n\times m}$, where $\boldsymbol{J}_{i,j} = \frac{\partial}{\partial x_j} f_i(\boldsymbol{x})$

$f\colon \mathbb{R}^m \to \mathbb{R}$ Hessian Matrix $\boldsymbol{H} \in \mathbb{R}^{m\times m}$, where $\boldsymbol{H}_{i,j} = \frac{\partial^2}{\partial x_i \partial x_j} f(\boldsymbol{x})$

2. Numerical Computation

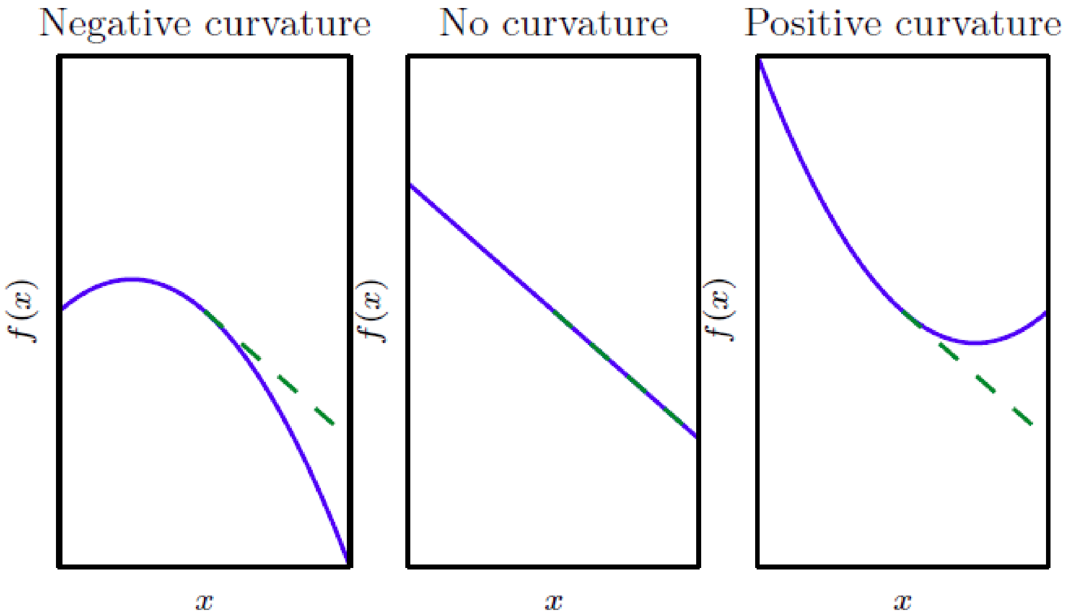
❖ If there is no analytical solution, approximate/estimate it via iteratively numerical process

2.3 Gradient-Based Optimization

2.3.1 Beyond the Gradient: Jacobian and Hessian Matrices

$\boldsymbol{f}: \mathbb{R}^m \to \mathbb{R}^n$ Jacobian Matrix $\boldsymbol{J} \in \mathbb{R}^{n\times m}$, where $\boldsymbol{J}_{i,j} = \frac{\partial}{\partial x_j} f_i(\boldsymbol{x})$

$f: \mathbb{R}^m \to \mathbb{R}$ Hessian Matrix $\boldsymbol{H} \in \mathbb{R}^{m\times m}$, where $\boldsymbol{H}_{i,j} = \frac{\partial^2}{\partial x_i \partial x_j} f(\boldsymbol{x})$



second derivative, curvature

positive/negative (semi-) definite

function f is convex/concave

2. Numerical Computation

❖ If there is no analytical solution, approximate/estimate it via iteratively numerical process

2.3 Gradient-Based Optimization

2.3.1 Beyond the Gradient: Jacobian and Hessian Matrices

$\boldsymbol{f}: \mathbb{R}^m \rightarrow \mathbb{R}^n$ Jacobian Matrix $\boldsymbol{J} \in \mathbb{R}^{n \times m}$, where $\boldsymbol{J}_{i,j} = \frac{\partial}{\partial x_j} f_i(\boldsymbol{x})$

$f: \mathbb{R}^m \rightarrow \mathbb{R}$ Hessian Matrix $\boldsymbol{H} \in \mathbb{R}^{m \times m}$, where $\boldsymbol{H}_{i,j} = \frac{\partial^2}{\partial x_i \partial x_j} f(\boldsymbol{x})$

consider a second-order Taylor approximation of $f(\boldsymbol{x})$

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}^{(0)}) + (\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{g} + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}^{(0)})$$

where $\boldsymbol{g}$ is the gradient and $\boldsymbol{H}$ is the Hessian at $\boldsymbol{x}^{(0)}$. If we set $\boldsymbol{x} = \boldsymbol{x}^{(0)} - \epsilon\, \boldsymbol{g}$

$$f(\boldsymbol{x}^{(0)} - \epsilon \boldsymbol{g}) \approx f(\boldsymbol{x}^{(0)}) - \epsilon \boldsymbol{g}^\top \boldsymbol{g} + \frac{1}{2}\epsilon^2 \boldsymbol{g}^\top \boldsymbol{H} \boldsymbol{g}.$$

if $\boldsymbol{g}^T \boldsymbol{H} \boldsymbol{g}$ is positive

$$\epsilon^* = \frac{\boldsymbol{g}^\top \boldsymbol{g}}{\boldsymbol{g}^\top \boldsymbol{H} \boldsymbol{g}}$$

2. Numerical Computation

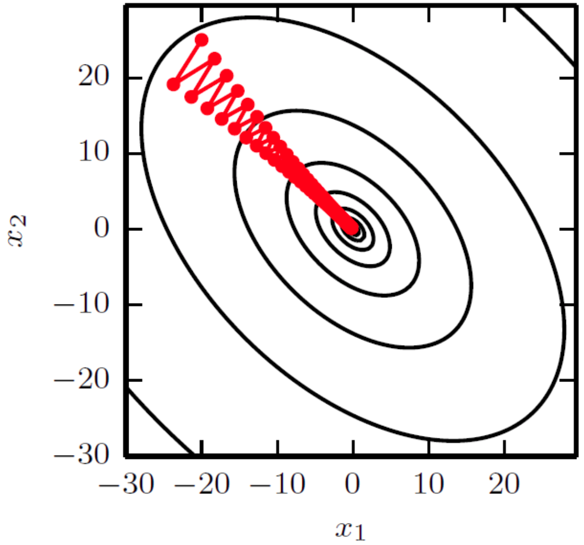
❖ If there is no analytical solution, approximate/estimate it via iteratively numerical process

2.3 Gradient-Based Optimization

2.3.1 Beyond the Gradient: Jacobian and Hessian Matrices

$\boldsymbol{f}: \mathbb{R}^m \to \mathbb{R}^n$ Jacobian Matrix $\boldsymbol{J} \in \mathbb{R}^{n \times m}$, where $\boldsymbol{J}_{i,j} = \frac{\partial}{\partial x_j} f_i(\boldsymbol{x})$

$f: \mathbb{R}^m \to \mathbb{R}$ Hessian Matrix $\boldsymbol{H} \in \mathbb{R}^{m \times m}$, where $\boldsymbol{H}_{i,j} = \frac{\partial^2}{\partial x_i \partial x_j} f(\boldsymbol{x})$



consider a second-order Taylor approximation of $f(\boldsymbol{x})$

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}^{(0)}) + (\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{g} + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}^{(0)})$$

where $\boldsymbol{g}$ is the gradient and $\boldsymbol{H}$ is the Hessian at $\boldsymbol{x}^{(0)}$. If we set $\boldsymbol{x} = \boldsymbol{x}^{(0)} - \epsilon\, \boldsymbol{g}$

$$f(\boldsymbol{x}^{(0)} - \epsilon \boldsymbol{g}) \approx f(\boldsymbol{x}^{(0)}) - \epsilon \boldsymbol{g}^\top \boldsymbol{g} + \frac{1}{2}\epsilon^2 \boldsymbol{g}^\top \boldsymbol{H} \boldsymbol{g}.$$

if $\boldsymbol{g}^T \boldsymbol{H} \boldsymbol{g}$ is positive

$$\epsilon^* = \frac{\boldsymbol{g}^\top \boldsymbol{g}}{\boldsymbol{g}^\top \boldsymbol{H} \boldsymbol{g}}$$

poor conditioned Hessian leads to a poor gradient descent!!!

2. Numerical Computation

❖ If there is no analytical solution, approximate/estimate it via iteratively numerical process

2.3 Gradient-Based Optimization

2.3.1 Beyond the Gradient: Jacobian and Hessian Matrices

Newton's method

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}^{(0)}) + (\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \nabla_{\boldsymbol{x}} f(\boldsymbol{x}^{(0)}) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(0)})^\top \boldsymbol{H}(f)(\boldsymbol{x}^{(0)})(\boldsymbol{x} - \boldsymbol{x}^{(0)})$$

solve for the critical point of this function

$$\boldsymbol{x}^* = \boldsymbol{x}^{(0)} - \boldsymbol{H}(f)(\boldsymbol{x}^{(0)})^{-1} \nabla_{\boldsymbol{x}} f(\boldsymbol{x}^{(0)})$$

Extension:

Gauss-Newton method
Conjugate-direction method
Quasi-Newton method

[2] A. Antoniou, W.-S. Lu, Practical optimization: algorithms and engineering applications, Springer, 2007.

2. Numerical Computation

❖ If there is no analytical solution, approximate/estimate it via iteratively numerical process

2.4 Constrained Optimization

$$\text{minimize } f(\mathbf{x})$$

$$\text{subject to: } \quad a_i(\mathbf{x}) = 0 \quad \text{for } i = 1, 2, \ldots, p$$
$$c_j(\mathbf{x}) \geq 0 \quad \text{for } j = 1, 2, \ldots, q$$

feasible region $\mathcal{R} = \{\mathbf{x} : \ a_i(\mathbf{x}) = 0 \text{ for } i = 1, 2, \ldots, p, \ c_j(\mathbf{x}) \geq 0 \text{ for } j = 1, 2, \ldots, q\}$

Karush-Kuhn-Tucker (KKT) approach, or generalized Lagrangian approach

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\alpha}) = f(\boldsymbol{x}) + \sum_i \lambda_i a_i(\boldsymbol{x}) + \sum_j \alpha_j c_j(\boldsymbol{x})$$

First-order sufficient/necessary conditions for a minimum
Second-order sufficient/necessary conditions for a minimum

[2] A. Antoniou, W.-S. Lu, Practical optimization: algorithms and engineering applications, Springer, 2007.