音響システム特論 第2回
Advanced Topics of Acoustic Systems Day 2

小山 翔一 / Shoichi Koyama

東京大学 大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo

Oct. 12, 2021

講義概要 / Summary

- 目的 / Goals
  - 音響信号を対象としたモデリングや最適化の方法論を学ぶことで，信号処理における発展的な数理的手法を，応用を通して理解することを目指す。
  - Gain understanding of advanced mathematical techniques in signal processing through applications by learning modeling and optimization methods for acoustics signals.
- 概要 / Summary
  - 前半は，音響信号を対象とした統計的推定法やアレイ信号処理，逆問題に関して講義する。具体的には，指向性制御，音源位置推定，空間音響，音場計測・制御などである。後半は，より発展的な内容に関して，主にゲスト講師による講義を行う。
  - The first half is lectures about statistical estimation, array signal processing, and inverse problems for acoustic signals. More specifically, beamforming, source localization, spatial audio, and sound field analysis and control. The second half is mainly talks by guest lecturers about more advanced topics.

講義概要 / Summary

- 火曜 2 限 オンライン講義 / Tuesday 2nd period, Online lecture
- キーワード / Keywords
  - 音響信号処理, 統計的信号処理, アレイ信号処理, スパースモデリング, 逆問題, 指向性制御, 音源位置推定, 空間音響, 音場計測, 音場制御
  - Acoustic signal processing, Statistical signal processing, Array signal processing, Sparse modeling, Inverse problems, Beamforming, Source localization, Spatial audio, Sound field analysis, Sound field control
- 講義は日本語で行いますが，スライドはできるだけ英語にします。 / Lectures are in Japanese, but slides are in English as much as possible.

講義日程 / Schedule

10/5	逆問題と統計的推定法 / Inverse Problem and Statistical Estimation
10/12	信号処理における最適化法 / Optimization in Signal Processing
10/19	休講 / Canceled
10/26	スパースモデリング / Sparse Modeling
11/2	アレイ信号処理 / Array Signal Processing
11/9	波動場のモデリング / Wavefield Modeling
11/16	空間音響の基礎 / Basics of Spatial Audio
11/30	[Guest] 高道慎之介先生 / Prof. Shinnosuke Takamichi
12/7	[Guest] 伊藤信貴先生 (新領域) / Prof. Nobutaka Ito (GSFS)
12/14	休講 / Canceled
12/21	[Guest] 井本桂右先生 (同志社大) / Prof. Keisuke Imoto (Doshisha)
1/4	音場の分析と合成 / Sound Field Analysis and Synthesis
1/11	[Guest] 猿渡洋先生 / Prof. Hiroshi Saruwatari

講義資料と成績評価 / Materials and Grades

- 講義資料 / Course materials
  - Downloadable at ITC-LMS or `http://www.sh01.org/ja/teaching/`
- 成績評価 / Grading system
  - レポート課題 (2 回) / Reporting assignments (twice)

Table of Contents

Optimization in signal processing

- Recall typical optimization problem appears in signal processing
- Model parameters $\boldsymbol{x}$ are estimated from observation values $\boldsymbol{y}$ as

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \mathcal{D}(\mathcal{A}(\boldsymbol{x}), \boldsymbol{y}) + \mathcal{R}_{\boldsymbol{\theta}}(\boldsymbol{x})$$

  Here, $\mathcal{A}$ is forward measurement operator, $\mathcal{D}$ is data fidelity term, and $\mathcal{R}_{\boldsymbol{\theta}}$ is regularization term with parameters $\boldsymbol{\theta}$.
- Closed-form solution is preferable, but the world is not that simple, unfortunately...
  - Cost function is nonlinear, nonconvex, or indifferentiable
  - Online algorithm is necessary for realtime application

Optimization in signal processing

- When there is no closed-form solution, iterative method is necessary.
  - Optimization problems appear in signal processing are *continuous optimization* in most cases.
  - *Discrete optimization* rarely appears with some exceptions (e.g., sensor placement problem)

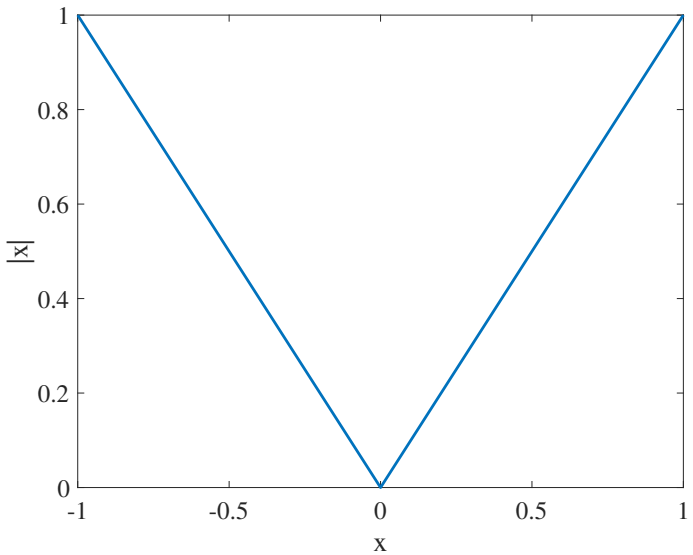
Developing efficient optimization algorithm is one of important research topics in signal processing

Examples

- ℓ_1-regularization problem
  - Least-squares error + Sparsity-promoting regularization

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}}\ \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2+\lambda\|\boldsymbol{x}\|_1$$

  Here, $\boldsymbol{x}\in\mathbb{R}^N$, $\boldsymbol{y}\in\mathbb{R}^M$, $\boldsymbol{A}\in\mathbb{R}^{M\times N}$, and $N>M$.
  - Equivalent to MAP estimation with Laplace distribution for prior
  - Cost function is convex, but indifferentiable.



Examples

- Phase retrieval problem
  - Optimization problem is generally formulated as

$$\underset{\boldsymbol{x}\in\mathbb{C}^N}{\text{minimize}} \, \| \boldsymbol{y} - |\boldsymbol{A}\boldsymbol{x}| \|_2^2$$

Here, $|\cdot|$ represents element-wise absolute value, $\boldsymbol{A} \in \mathbb{C}^{M\times N}$ is STFT matrix, $\boldsymbol{y} \in \mathbb{R}_+^M$ is given vectorized magnitude spectrogram, and $\boldsymbol{x} \in \mathbb{C}^N$ is time-domain signal to be recovered.

  - Cost function is nonconvex and indifferentiable.

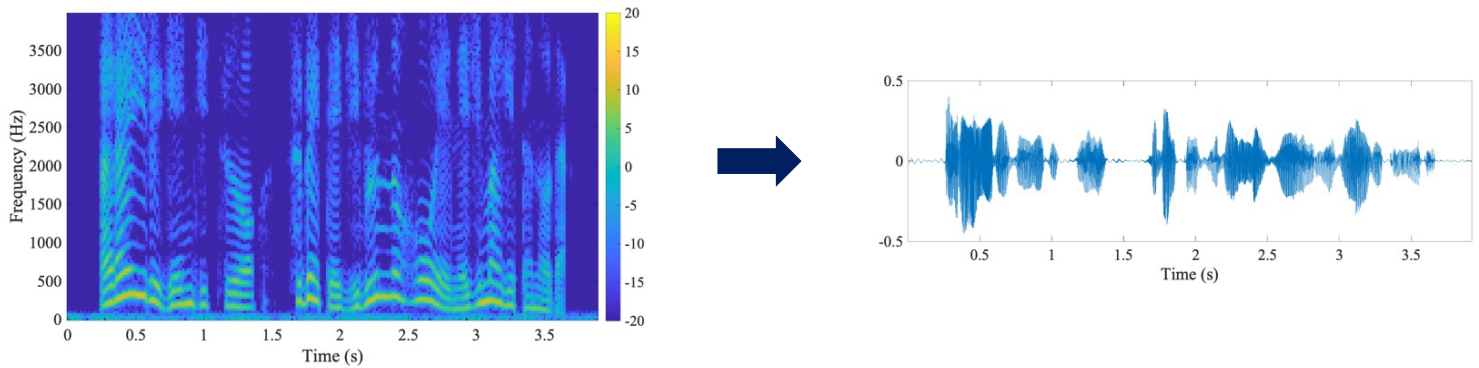


Table of Contents

Gradient method

- Gradient method (勾配法) is basic algorithm using gradient of objective function for unconstrained optimization problem

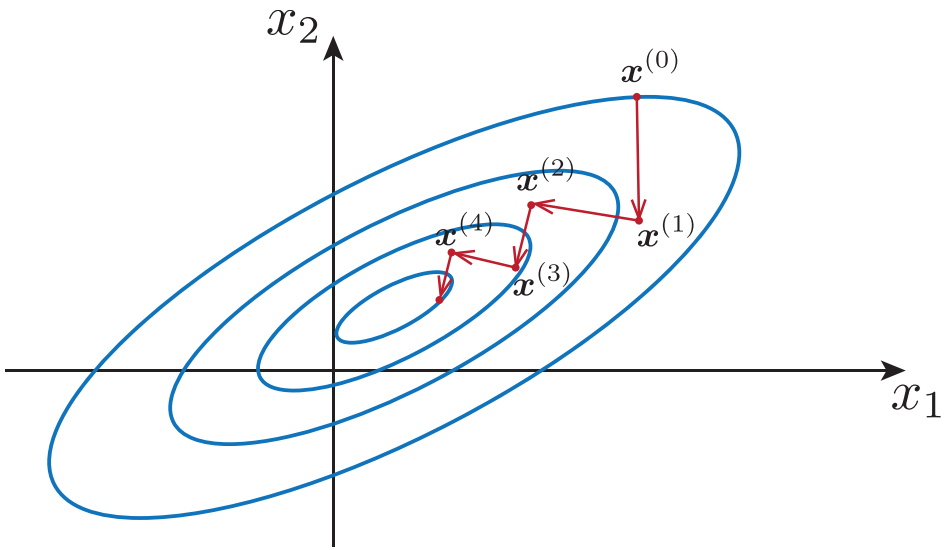
$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}}\, f(\boldsymbol{x})$$

  $f\colon \mathbb{R}^N \to \mathbb{R}$ is assumed to be differentiable.
- Steepest descent method (最急降下法) is representative first-order method, which iteratively update $\boldsymbol{x}$ in direction to $-\nabla f(\boldsymbol{x}^{(i)})$ (i is iteration index)
- First-order methods, e.g., steepest descent and conjugate gradient methods, are usually preferred to second-order methods, e.g., Newton's method, in signal processing because of computational efficiency.

Gradient method

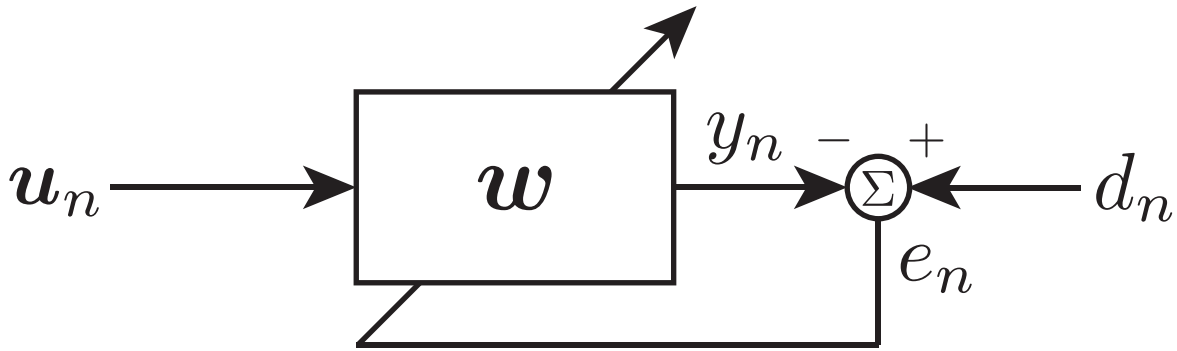
Algorithm Steepest descent method

1: Set initial value $\boldsymbol{x}^{(0)}$ and $i \leftarrow 0$
2: **while** stopping condition is not satisfied **do**
3: Compute descent direction $\boldsymbol{d}^{(i)} \leftarrow -\nabla f(\boldsymbol{x}^{(i)})$
4: Line search of step size $\mu \geq 0$
5: Update $\boldsymbol{x}^{(i+1)} \leftarrow \boldsymbol{x}^{(i)} + \mu \boldsymbol{d}^{(i)}$
6: $i \leftarrow i + 1$
7: **end while**



Adaptive filter

- Adaptive filter (適応フィルタ) is linear filter whose transfer function is adjusted based on observed data by adaptive process
- Gradient method can be applied to obtain adaptive filtering algorithm
- Application examples of adaptive filter are
  - Active noise control (ANC: 能動騒音制御)
  - Acoustic echo cancellation (AEC: 音響エコー除去)



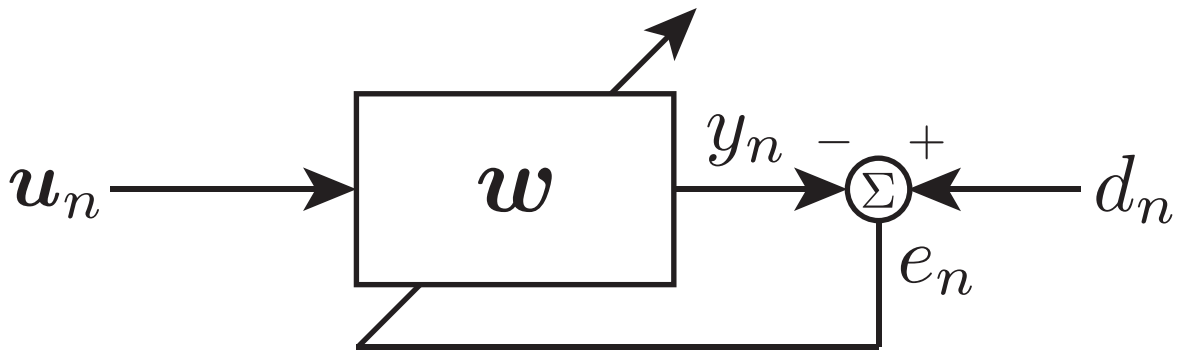
Adaptive filter

- Recall Wiener filter to obtain filter that extract desired signal
- Input time-series signal $\boldsymbol{u}_n$ and filter coefficients $\boldsymbol{w}$ are

$$\boldsymbol{u}_n = [u_n, u_{n-1}, \ldots, u_{n-K+1}]^\mathsf{T}$$
$$\boldsymbol{w} = [w_1, w_2, \ldots, w_K]^\mathsf{T}$$

  $\mathbb{E}[u_n] = 0$ is assumed.
- Then, output signal y_n is convolution of $\boldsymbol{u}_k$ and $\boldsymbol{w}$ as

$$y_n = \boldsymbol{w}^\mathsf{T}\boldsymbol{u}_n = \sum_{j=1}^{K} w_j u_{n-j+1}$$



Adaptive filter

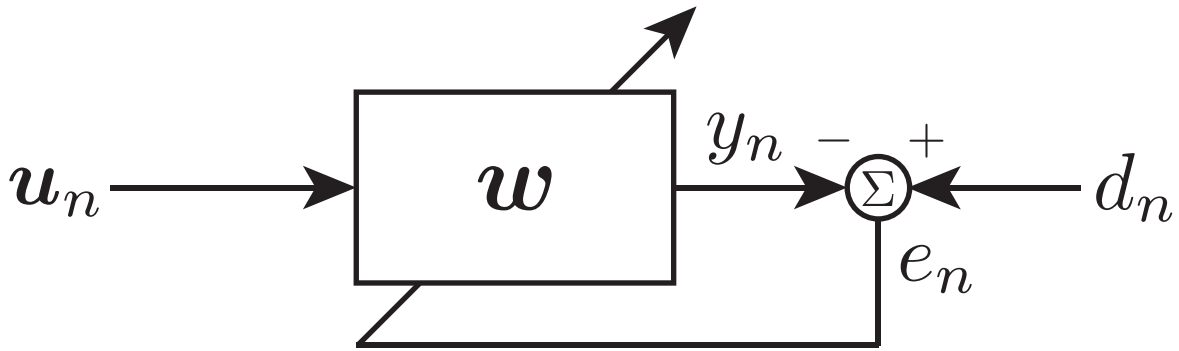
- Wiener filter is obtained by MMSE estimate of $e_n = d_n - y_n$

$$\underset{\boldsymbol{w} \in \mathbb{R}^K}{\text{minimize}}\, \mathcal{J} := \mathbb{E}[|e_n|^2]$$

- Closed-form solution of $\boldsymbol{w}$ is obtained as

$$\hat{\boldsymbol{w}} = \boldsymbol{R}_u^{-1} \boldsymbol{r}_{ud}$$

where $\boldsymbol{R}_u = \mathbb{E}[\boldsymbol{u}_n \boldsymbol{u}_n^{\mathsf{T}}]$ and $\boldsymbol{r}_{ud} = \mathbb{E}[\boldsymbol{u}_n d_n]$.



Steepest descent method for adaptive filter

- Online algorithm for adaptively update filter $\boldsymbol{w}$ can be obtained based steepest descent method
- Gradient of the cost function $\mathcal{J}$ at $(n-1)$th iteration, $\boldsymbol{w}_{n-1}$, is obtained as

$$\nabla \mathcal{J} = \left. \frac{\partial \mathcal{J}}{\partial \boldsymbol{w}} \right|_{\boldsymbol{w}=\boldsymbol{w}_{n-1}} = -2\boldsymbol{r}_{ud} + 2\boldsymbol{R}_u \boldsymbol{w}_{n-1}$$

- Filter $\boldsymbol{w}$ at nth iteration is obtained by updating $\boldsymbol{w}_{n-1}$ to direction of $-\nabla \mathcal{J}$ as

$$\begin{aligned} \boldsymbol{w}_n &= \boldsymbol{w}_{n-1} - \eta \nabla \mathcal{J} \\ &= \boldsymbol{w}_{n-1} + \mu(\boldsymbol{r}_{ud} - \boldsymbol{R}_u \boldsymbol{w}_{n-1}) \end{aligned}$$

where μ $(= 2\eta)$ is step-size parameter.

Steepest descent method for adaptive filter

- Update formula is reformulated as

$$\begin{aligned}\boldsymbol{w}_n &= \boldsymbol{w}_{n-1} + \mu(\mathbb{E}[\boldsymbol{u}_n d_n] - \mathbb{E}[\boldsymbol{u}_n \boldsymbol{u}_n^{\mathsf{T}}]\boldsymbol{w}_{n-1}) \\ &= \boldsymbol{w}_{n-1} + \mu\mathbb{E}[\boldsymbol{u}_n(d_n - \boldsymbol{u}_n^{\mathsf{T}}\boldsymbol{w}_{n-1})] \\ &= \boldsymbol{w}_{n-1} + \mu\mathbb{E}[\boldsymbol{u}_n \xi_n]\end{aligned}$$

  Here, *prior estimation error* ξ_n is defined as

$$\xi_n = d_n - \boldsymbol{u}_n^{\mathsf{T}}\boldsymbol{w}_{n-1}$$

- Newton's method is also applicable by calculating Hessian matrix $\nabla^2\mathcal{J}$, but computationally expensive.

NLMS algorithm

- Update formula of steepest descent method still include expectation, which is difficult to calculate in practice.
- In *least-mean-square (LMS) algorithm*, the expectation is replaced with instantaneous value.

$$\boldsymbol{w}_n = \boldsymbol{w}_{n-1} + \mu \boldsymbol{u}_n \xi_n$$

- In normalized least-mean-square (NLMS) algorithm, gradient is normalized by power of input.

$$\boldsymbol{w}_n = \boldsymbol{w}_{n-1} + \frac{\mu}{\|\boldsymbol{u}_n\|^2} \boldsymbol{u}_n \xi_n$$

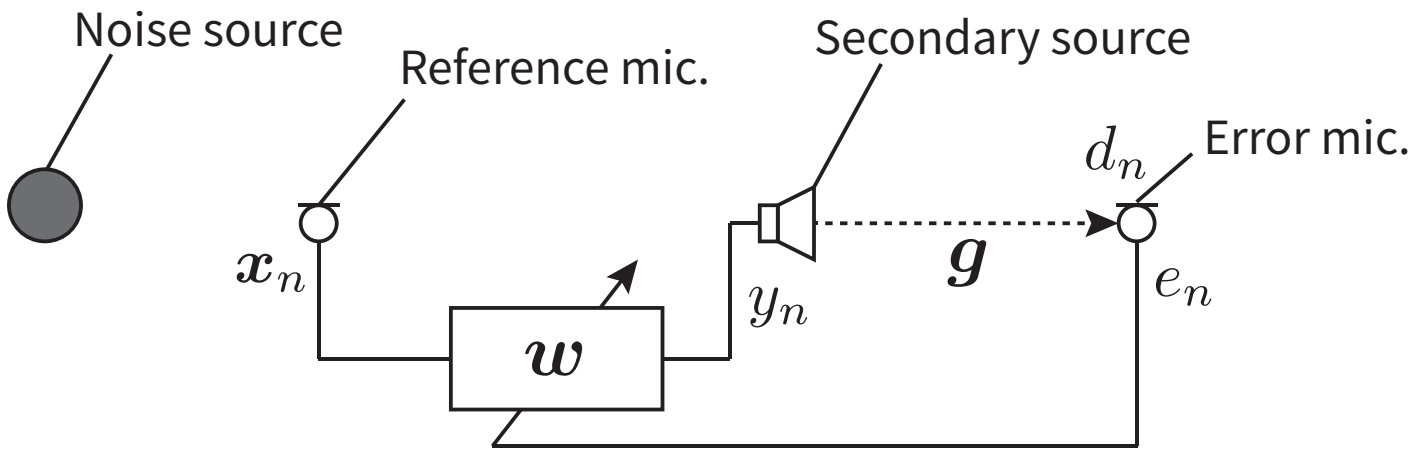
  Convergence of NLMS algorithm is guaranteed by setting μ as

$$0 < \mu < 2$$

- There are a lot of variants, e.g., *affine projection* and *recursive least squares* algorithms.

Adaptive filter in ANC

- Adaptive filter is used to obtain optimal filter $\boldsymbol{w}$ to reduce power of error microphone $\mathbb{E}[|e_n|^2]$ by secondary loudspeaker output



How to handle complex variables...?

- In acoustic signal processing, many optimization problems are formulated in *time-frequency domain*, which means cost function becomes function of complex variables.
- Let us consider derivative of single complex variable function $f(z)$ where $z = x + \mathrm{j}y$ and $x, y \in \mathbb{R}$.
- Function $f(z)$ can be represented as

$$f(z) = u(x, y) + \mathrm{j}v(x, y)$$

where $u, v \colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ are real and imaginary parts of $f(z)$.

How to handle complex variables...?

- Simple strategy is to treat f as function of two independent variables x and y:

$$\frac{\partial}{\partial x} f(x,y) = \frac{\partial u}{\partial x} + \mathrm{j}\frac{\partial v}{\partial x}, \quad \frac{\partial}{\partial y} f(x,y) = \frac{\partial u}{\partial y} + \mathrm{j}\frac{\partial v}{\partial y}$$

  However, this procedure is sometimes complicated...

- In Wirtinger derivative, $f(z)$ is considered as function of z and z^*:

$$f(z,z^*) = u\left(\frac{z+z^*}{2}, \frac{z-z^*}{2}\right) + \mathrm{j}v\left(\frac{z+z^*}{2}, \frac{z-z^*}{2}\right)$$

$$\frac{\partial}{\partial z} f(z,z^*) = \frac{1}{2}\left(\frac{\partial f(z)}{\partial x} - \mathrm{j}\frac{\partial f(z)}{\partial y}\right)$$

$$\frac{\partial}{\partial z^*} f(z,z^*) = \frac{1}{2}\left(\frac{\partial f(z)}{\partial x} + \mathrm{j}\frac{\partial f(z)}{\partial y}\right)$$

How to handle complex variables...?

- $\partial f / \partial z^*$ is generally used in optimization problems:

$$\frac{\partial}{\partial z^*} f(z, z^*) = \frac{1}{2} \left\{ \frac{\partial u}{\partial x} - \frac{\partial v}{\partial y} + \mathrm{j} \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) \right\}$$

- Example:

$$\frac{\partial |z|^2}{\partial z^*} = \frac{\partial z z^*}{\partial z^*} = z$$

Complex matrix differentiation

- Can be extended to vectors and matrices.
- Inner product

$$\frac{\partial}{\partial \boldsymbol{z}^*} \boldsymbol{z}^{\mathsf{H}} \boldsymbol{a} = \boldsymbol{a}, \quad \frac{\partial}{\partial \boldsymbol{z}^*} \boldsymbol{a}^{\mathsf{H}} \boldsymbol{z} = \boldsymbol{0}$$

- Quadratic form

$$\frac{\partial}{\partial \boldsymbol{z}^*} \boldsymbol{z}^{\mathsf{H}} \boldsymbol{A} \boldsymbol{z} = \boldsymbol{A} \boldsymbol{z}$$

- Trace

$$\frac{\partial}{\partial \boldsymbol{A}^*} \mathrm{trace}(\boldsymbol{A} \boldsymbol{B}) = \boldsymbol{B}^{\mathsf{T}}, \quad \frac{\partial}{\partial \boldsymbol{A}^*} \mathrm{trace}(\boldsymbol{A} \boldsymbol{B} \boldsymbol{A}^{\mathsf{H}}) = \boldsymbol{A}^* \boldsymbol{B}^{\mathsf{T}}$$

Table of Contents

Why convex optimization?

- Convex optimization is the methods for minimizing *convex function* over *convex sets*, for example,

$$\begin{aligned} &\underset{\boldsymbol{x}}{\text{minimize}} \;\; f(\boldsymbol{x}) \\ &\text{subject to} \;\; g_i(\boldsymbol{x}) \leq 0 \;\; (i = 1, \ldots, P) \end{aligned}$$

  where $f, g_1, \ldots, g_P$ are convex functions.
- In particular, first-order methods in convex optimization have been attracted attention in signal processing and machine learning since about 2005.
- Three reasons:
  - Applicable to optimization problem including indifferentiable (but convex) cost function (e.g., ℓ_1-regularization problem)
  - Suitable for large-scale problems because of low time and space complexities
  - Global optimality is guaranteed by established algorithms

Preliminaries

- Set C is convex if for any $\boldsymbol{x}, \boldsymbol{y} \in C$ and any $\lambda \in [0, 1]$,

$$\lambda \boldsymbol{x} + (1 - \lambda)\boldsymbol{y} \in C$$

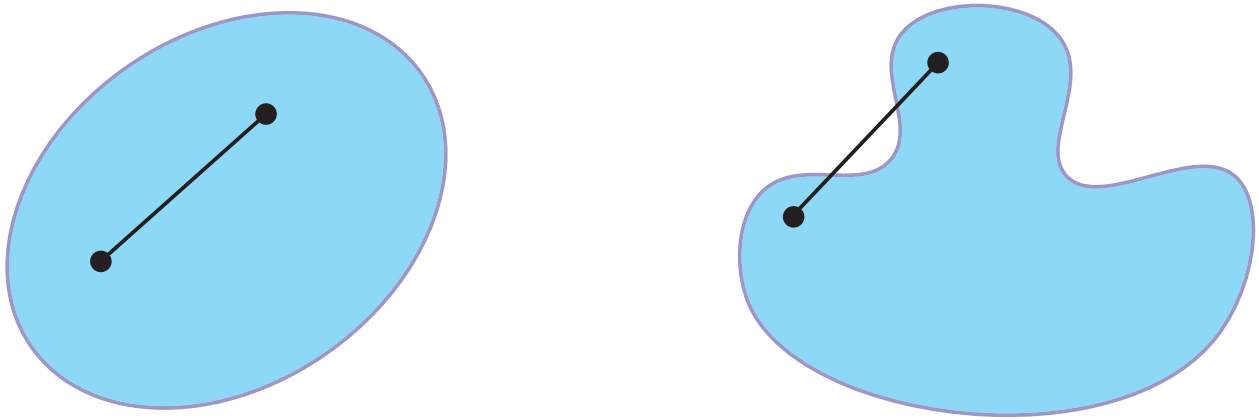


Figure: Left: Convex set, Right: Nonconvex set

Preliminaries

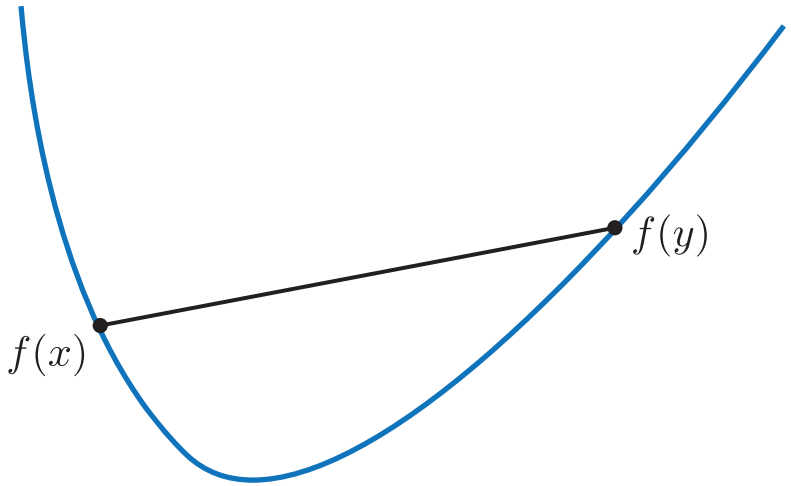
- Function $f : \mathbb{R}^N \to \mathbb{R} \cup \{+\infty\}$ is convex if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathrm{dom}(f)$ and $\lambda \in [0, 1]$,

$$f(\lambda \boldsymbol{x} + (1 - \lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1 - \lambda) f(\boldsymbol{y})$$

Here, $\mathrm{dom}(f)$ is *effective domain* (実効定義域) of f defined as

$$\mathrm{dom}(f) \coloneqq \{\boldsymbol{x} \in \mathbb{R}^N \mid f(\boldsymbol{x}) < \infty\}$$

i.e., set of points for which f takes finite value.

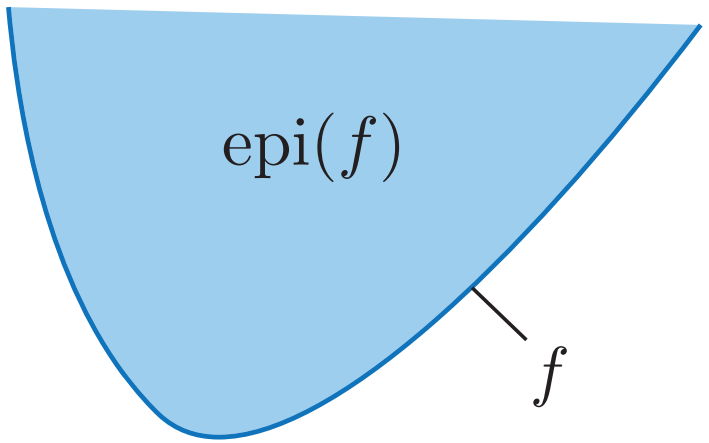


Preliminaries

- *Epigraph* (エピグラフ) of f is defined as

$$\mathrm{epi}(f) \coloneqq \left\{(\boldsymbol{x}, t) \in \mathbb{R}^N \times \mathbb{R} \mid f(\boldsymbol{x}) \leq t\right\}$$

- Epigraph of convex function is convex set.



Preliminaries

- Suppose convex function $f\colon \mathbb{R}^N \to \mathbb{R} \cup \{+\infty\}$
- f is proper, if its effective domain

$$\mathrm{dom}(f) \coloneqq \{\boldsymbol{x} \in \mathbb{R}^N \mid f(\boldsymbol{x}) < \infty\}$$

  is not empty.
- f is closed if its epigraph

$$\mathrm{epi}(f) \coloneqq \left\{(\boldsymbol{x}, t) \in \mathbb{R}^N \times \mathbb{R} \mid f(\boldsymbol{x}) \leq t\right\}$$

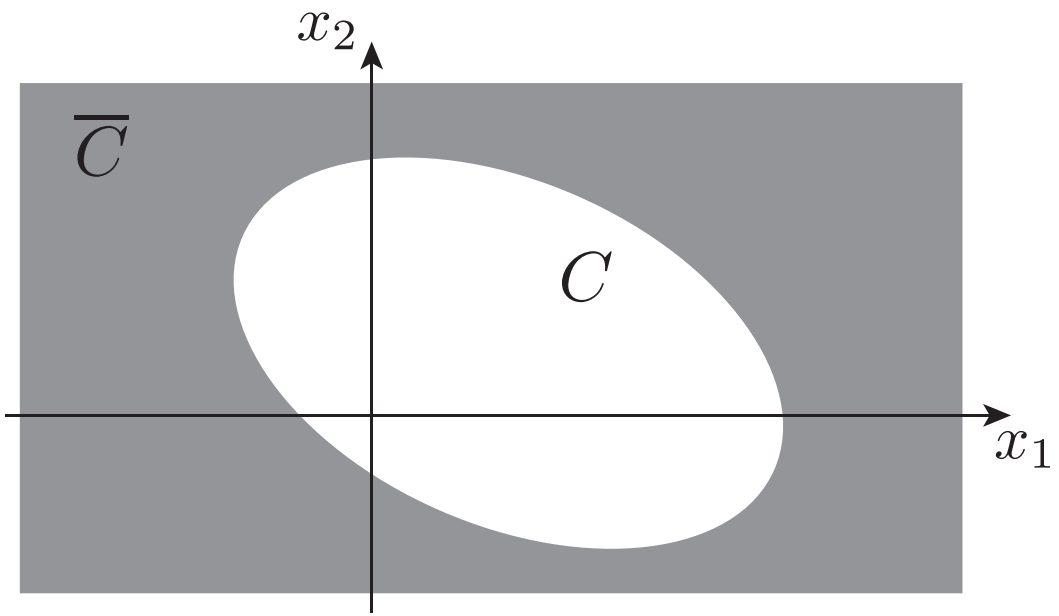
  is closed set, which is equivalent to f is lower semicontinuous.
- Let $\Gamma_0(\mathbb{R}^N)$ be class of proper lower semicontinuous convex functions on $\mathbb{R}^N$.

Preliminaries

- For example, indicator function (指示関数) ι_C defined with closed nonempty convex set C as

$$\iota_C(\boldsymbol{x}) := \begin{cases} 0, & \boldsymbol{x} \in C \\ \infty, & \text{otherwise} \end{cases}$$

belongs to $\Gamma_0(\mathbb{R}^N)$.



Proximal operator

- Single step of steepest descent method for f from $\boldsymbol{v}$ to $\boldsymbol{x}$ with step size γ can be rewritten as

$$\begin{aligned}\boldsymbol{x} &= \boldsymbol{v} - \gamma \nabla f(\boldsymbol{v}) \\ &= \arg\min_{\boldsymbol{x}} \left\{ f(\boldsymbol{v}) + \langle \nabla f(\boldsymbol{v}), \boldsymbol{x} - \boldsymbol{v} \rangle + \frac{1}{2\gamma} \|\boldsymbol{x} - \boldsymbol{v}\|_2^2 \right\}\end{aligned}$$

- Minimization problem of
  - first-order approximation of f at $\boldsymbol{v}$: $f(\boldsymbol{v}) + \langle \nabla f(\boldsymbol{v}), \boldsymbol{x} - \boldsymbol{v} \rangle$
  - proximity term with weight $1/2\gamma$: $\frac{1}{2\gamma} \|\boldsymbol{x} - \boldsymbol{v}\|_2^2$

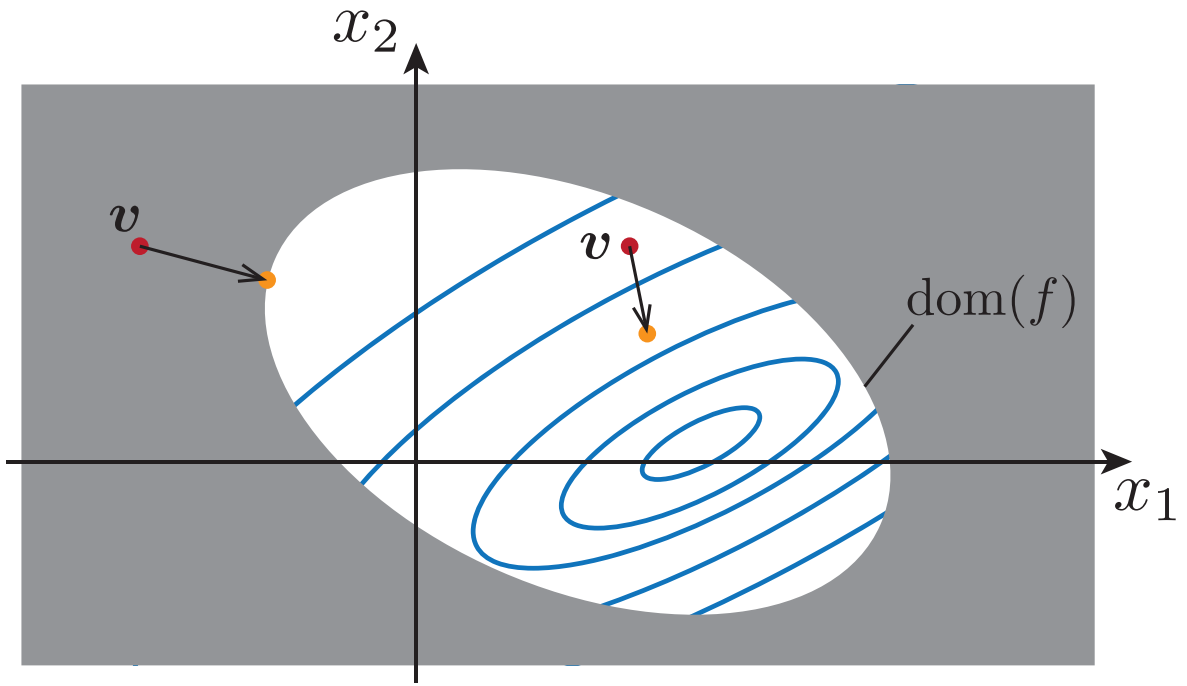
Proximal operator

- Proximal operator/mapping (近接作用素/写像) is powerful tool in convex optimization, which can be regarded as generalization of gradient step.
- Proximal operator $\mathrm{prox}_{\gamma f} \colon \mathbb{R}^N \to \mathbb{R}^N$ for $f \in \Gamma_0(\mathbb{R}^N)$ with parameter $\gamma > 0$ is defined as

$$\mathrm{prox}_{\gamma f}(\boldsymbol{v}) = \underset{\boldsymbol{x} \in \mathbb{R}^N}{\arg\min} \left\{ f(\boldsymbol{x}) + \frac{1}{2\gamma} \|\boldsymbol{x} - \boldsymbol{v}\|_2^2 \right\}$$

- $\mathrm{prox}_{\gamma f}(\boldsymbol{v})$ is a point that compromises between minimizing f and being near to $\boldsymbol{v}$ (*proximal point*), and γ is relative weight.

Proximal operator



- Under some assumptions, proximal operator is approximated as

$$\mathrm{prox}_{\gamma f}(\boldsymbol{v}) \approx \boldsymbol{v} - \gamma \nabla f(\boldsymbol{v})$$

  when γ is small and f is differentiable.
- One of the benefits of using proximal operator is f can be *nonsmooth*.

Proximal point algorithm

- Proximal point/minimization algorithm (近接点/最小化アルゴリズム) is the simplest algorithm based on proximal operator.
- Just repeatedly applying $\mathrm{prox}_{\gamma f}$ as

Algorithm Proximal point algorithm

1: Set initial value $\boldsymbol{x}^{(0)}$ and $i \leftarrow 0$
2: **while** stopping condition is not satisfied **do**
3: $\quad \boldsymbol{x}^{(i+1)} \leftarrow \mathrm{prox}_{\gamma^{(i)} f}(\boldsymbol{x}^{(i)})$
4: $\quad i \leftarrow i + 1$
5: **end while**

Proximal point algorithm

- Proximal point algorithm is equivalent to steepest descent method for smooth approximation function (more precisely, *Moreau envelope*) of f with step size $\gamma^{(i)}$.
- Useful when proximal operator can be quickly evaluated.

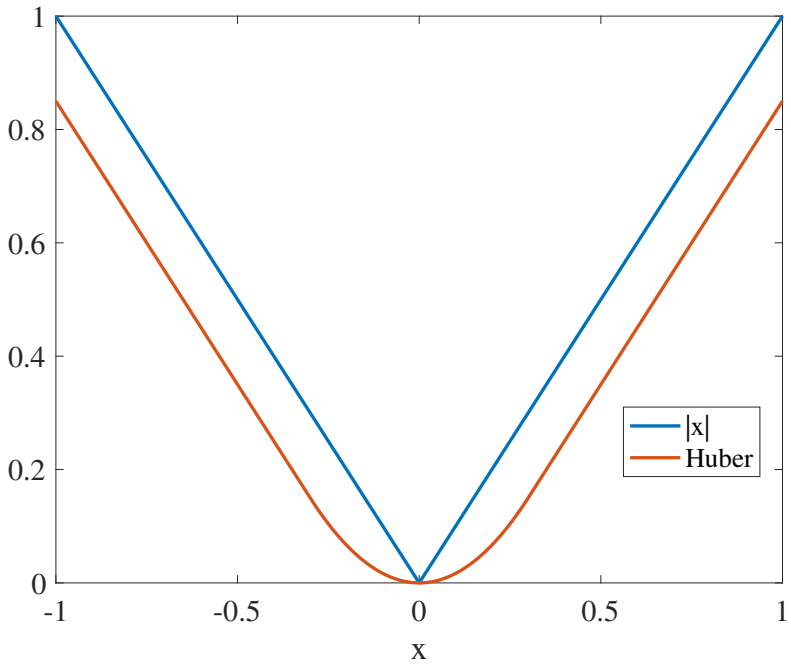


Figure: Moreau envelope of $|x|$ becomes Huber function.

How to evaluate proximal operator?

- Obviously, computability of proximal operator depends on f.
- Fortunately, proximal operators for various f useful in applications can be efficiently evaluated (*proximable*)
- Exhaustive list of proximal operators can be found at

  `http://proximity-operator.net/`

  (Sometimes, proximal-operator-like operator is formally defined for $f \notin \Gamma_0(\mathbb{R}^N)$.)

Examples of proximal operator

- Indicator function for closed nonempty convex set C

$$\iota_C(\boldsymbol{x}) \coloneqq \begin{cases} 0, & \boldsymbol{x} \in C \\ \infty, & \text{otherwise} \end{cases}$$

- Proximal operator of $\iota_C \in \Gamma_0(\mathbb{R}^N)$ is

$$\begin{aligned} \operatorname{prox}_{\iota_C}(\boldsymbol{v}) &= \operatorname*{arg\,min}_{\boldsymbol{x} \in \mathbb{R}^N} \left\{ \iota_C(\boldsymbol{x}) + \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{v}\|_2^2 \right\} \\ &= \operatorname*{arg\,min}_{\boldsymbol{x} \in C} \|\boldsymbol{x} - \boldsymbol{v}\|_2^2 \\ &\coloneqq P_C(\boldsymbol{v}) \end{aligned}$$

$\operatorname{prox}_{\iota_C}$ is projection operator onto C, i.e., P_C.

Examples of proximal operator

- ℓ_1-norm of $\boldsymbol{x} \in \mathbb{R}^N$ defined as

$$f(\boldsymbol{x}) = \|\boldsymbol{x}\|_1 = \sum_{n=1}^{N} |x_n|$$

- Proximal operator of ℓ_1-norm is derived as

$$\begin{aligned}\mathrm{prox}_{\gamma\ell_1}(\boldsymbol{v}) &= \underset{\boldsymbol{x}\in\mathbb{R}^N}{\arg\min} \left\{ \|\boldsymbol{x}\|_1 + \frac{1}{2\gamma}\|\boldsymbol{x}-\boldsymbol{v}\|_2^2 \right\} \\ &= \underset{\boldsymbol{x}\in\mathbb{R}^N}{\arg\min} \sum_{n=1}^{N} \left\{ |x_n| + \frac{1}{2\gamma}(x_n - v_n)^2 \right\}\end{aligned}$$

  x_n, v_n are nth element of $\boldsymbol{x}, \boldsymbol{v}$, respectively.

- Simplified to minimization problem of each element.

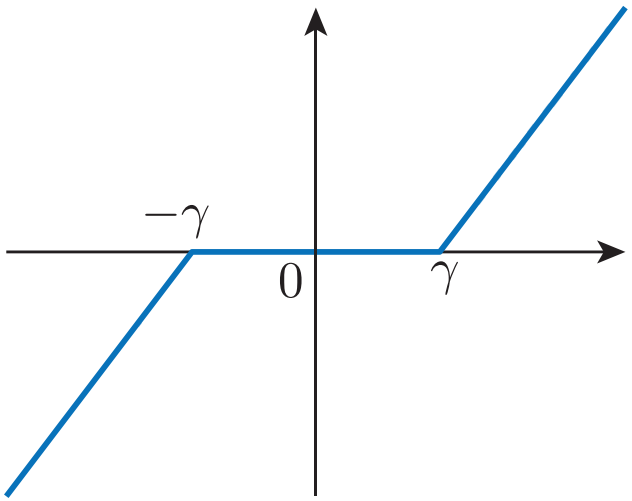
Examples of proximal operator

- Each element of $\mathrm{prox}_{\gamma \ell_1}(\boldsymbol{v})$ is written as

$$\left[\mathrm{prox}_{\gamma \ell_1}(\boldsymbol{v})\right]_n = \begin{cases} v_n - \gamma, & v_n \geq \gamma \\ 0, & -\gamma < v_n < \gamma \\ v_n + \gamma, & v_n \leq -\gamma \end{cases} \\ := \mathcal{T}_\gamma(v_n)$$

  $\mathcal{T}_\gamma$ is referred to as soft thresholding operator.

- For simplicity, $\mathrm{prox}_{\gamma \ell_1}(\boldsymbol{v}) = \mathcal{T}_\gamma(\boldsymbol{v})$.



Examples of proximal operator

- Quadratic function defined with positive definite symmetric matrix $\boldsymbol{Q} \in \mathbb{R}^{N \times N}$ and vector $\boldsymbol{y} \in \mathbb{R}^N$

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\mathsf{T}\boldsymbol{Q}\boldsymbol{x} - \boldsymbol{y}^\mathsf{T}\boldsymbol{x}$$

  Note that this function is differentiable.

- Proximal operator of quadratic function is

$$\begin{aligned}\mathrm{prox}_{\gamma f}(\boldsymbol{v}) &= \underset{\boldsymbol{x}\in\mathbb{R}^N}{\arg\min} \left\{ \frac{1}{2}\boldsymbol{x}^\mathsf{T}\boldsymbol{Q}\boldsymbol{x} - \boldsymbol{y}^\mathsf{T}\boldsymbol{x} + \frac{1}{2\gamma}(\boldsymbol{x}-\boldsymbol{v})^\mathsf{T}(\boldsymbol{x}-\boldsymbol{v}) \right\} \\ &= \left(\boldsymbol{Q} + \frac{1}{\gamma}\boldsymbol{I}\right)^{-1}\left(\boldsymbol{y} + \frac{1}{\gamma}\boldsymbol{v}\right)\end{aligned}$$

  Simply derived by evaluating gradient $= 0$.

- Applicable to stably compute $\boldsymbol{Q}^{-1}\boldsymbol{y}$.

Proximal splitting algorithms

- Generally, proximal point algorithm is not directly applicable to cost functions used in practical applications.
- Many cost functions in signal processing are written as

$$f(\boldsymbol{x}) = f_1(\boldsymbol{x}) + \cdots + f_K(\boldsymbol{x})$$

- Algorithms separately handling each (possibly proximable) cost function $f_1, \ldots, f_K$ are collectively called proximal splitting algorithms (近接分離アルゴリズム).
- Several convergence-guranteed proximal splitting algorithms are introduced.

Proximal gradient method

- Suppose the following optimization problem:

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}}\, f(\boldsymbol{x}) + g(\boldsymbol{x})$$

  where $f, g \in \Gamma_0(\mathbb{R}^N)$, and f is differentiable.
- Proximal gradient / forward-backward splitting method (近接勾配法) for solving this problem:

Algorithm Proximal gradient method

1: Set initial value $\boldsymbol{x}^{(0)}$, $\gamma > 0$, and $i \leftarrow 0$
2: **while** stopping condition is not satisfied **do**
3: $\quad \boldsymbol{x}^{(i+1)} \leftarrow \mathrm{prox}_{\gamma g}(\boldsymbol{x}^{(i)} - \gamma \nabla f(\boldsymbol{x}^{(i)}))$
4: $\quad i \leftarrow i + 1$
5: **end while**

Douglas–Rachford splitting

- Suppose the following optimization problem:

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}}\, f(\boldsymbol{x}) + g(\boldsymbol{x})$$

  where $f, g \in \Gamma_0(\mathbb{R}^N)$.

- Douglas–Rachford splitting for solving this problem:

Algorithm Douglas–Rachford splitting

1: Set initial value $\boldsymbol{z}^{(0)}$, $\gamma > 0$, and $i \leftarrow 0$
2: **while** stopping condition is not satisfied **do**
3: $\quad \boldsymbol{x}^{(i+1)} \leftarrow \mathrm{prox}_{\gamma f}(\boldsymbol{z}^{(i)})$
4: $\quad \boldsymbol{z}^{(i+1)} \leftarrow \boldsymbol{z}^{(i)} + \mathrm{prox}_{\gamma g}(2\boldsymbol{x}^{(i+1)} - \boldsymbol{z}^{(i)}) - \boldsymbol{x}^{(i+1)}$
5: $\quad i \leftarrow i + 1$
6: **end while**

Alternating direction method of multipliers

- Suppose the following optimization problem:

$$\underset{\boldsymbol{x}\in\mathbb{R}^N,\boldsymbol{z}\in\mathbb{R}^M}{\text{minimize}}\ f(\boldsymbol{x})+g(\boldsymbol{z})\ \text{ subject to }\ \boldsymbol{z}=\boldsymbol{A}\boldsymbol{x}$$

  where $f\in\Gamma_0(\mathbb{R}^N)$, $g\in\Gamma_0(\mathbb{R}^M)$, and $\boldsymbol{A}\in\mathbb{R}^{M\times N}$.
- Alternating direction method of multipliers (ADMM: 交互方向乗数法) for solving this problem:

Algorithm ADMM

1: Set initial value $\boldsymbol{z}^{(0)}$, $\boldsymbol{\theta}^{(0)}$, $\gamma>0$, and $i\leftarrow 0$
2: **while** stopping condition is not satisfied **do**
3: $\quad\boldsymbol{x}^{(i+1)}\leftarrow\underset{\boldsymbol{x}\in\mathbb{R}^N}{\arg\min}\left\{f(\boldsymbol{x})+\frac{1}{2\gamma}\left\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{z}^{(i)}+\boldsymbol{\theta}^{(i)}\right\|_2^2\right\}$
4: $\quad\boldsymbol{z}^{(i+1)}\leftarrow\mathrm{prox}_{\gamma g}(\boldsymbol{A}\boldsymbol{x}^{(i+1)}+\boldsymbol{\theta}^{(i)})$
5: $\quad\boldsymbol{\theta}^{(i+1)}\leftarrow\boldsymbol{\theta}^{(i)}+\boldsymbol{A}\boldsymbol{x}^{(i+1)}-\boldsymbol{z}^{(i+1)}$
6: $\quad i\leftarrow i+1$
7: **end while**

Primal-dual splitting method

- Suppose the following optimization problem:

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}}\, f(\boldsymbol{x}) + g(\boldsymbol{x}) + h(\boldsymbol{A}\boldsymbol{x})$$

  where $f, g \in \Gamma_0(\mathbb{R}^N)$, $h \in \Gamma_0(\mathbb{R}^M)$, $\boldsymbol{A} \in \mathbb{R}^{M\times N}$, and f is differentiable.
- Primal-dual splitting method (主-双対近接分離法) for solving this problem:

Algorithm Primal-dual splitting method

Set initial value $\boldsymbol{x}^{(0)}$, $\boldsymbol{z}^{(0)}$, $\gamma_1, \gamma_2 > 0$, and $i \leftarrow 0$
while stopping condition is not satisfied **do**
  $\boldsymbol{x}^{(i+1)} \leftarrow \mathrm{prox}_{\gamma_1 g}(\boldsymbol{x}^{(i)} - \gamma_1(\nabla f(\boldsymbol{x}^{(i)}) + \boldsymbol{A}^\mathsf{T}\boldsymbol{z}^{(i)}))$
  $\boldsymbol{z}^{(i+1)} \leftarrow \mathrm{prox}_{\gamma_2 h^*}(\boldsymbol{z}^{(i)} + \gamma_2(2\boldsymbol{x}^{(i+1)} - \boldsymbol{x}^{(i)}))$
  $i \leftarrow i + 1$
end while

Proximal splitting in signal processing

- Typical optimization problem appears in signal processing

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \mathcal{D}(\mathcal{A}(\boldsymbol{x}), \boldsymbol{y}) + \mathcal{R}_{\boldsymbol{\theta}}(\boldsymbol{x})$$

- Proximal splitting algorithms are useful when $\mathcal{D}$ and $\mathcal{R}_{\boldsymbol{\theta}}$ are convex functions, and $\mathcal{A}$ is linear operator.

  $\Rightarrow$ Look up the list of proximal operators!

- Convexity is too strong constraint?

  $\Rightarrow$ Effectiveness of proximal splitting algorithms are empirically shown in some nonconvex optimization problems.

Table of Contents

Majorization-minimization algorithm

- Majorization-minimization (MM) algorithm (補助関数法 or 上界最小化アルゴリズム) is a general framework to develop optimization algorithms rather than specific algorithm.
- Many algorithms can be interpreted as special case of MM algorithm:
  - Steepest descent method, Newton's method, EM algorithm, proximal gradient method, multiplicative update rule in NMF, etc.
- Applicable to *nonconvex* optimization problem with guarantee of *monotonic non-increase of cost function*.
  - Surrogate/auxiliary function (代理関数/補助関数) of cost function is constructed
  - Parameter of surrogate function and parameter to be optimized are alternately updated

Majorization-minimization algorithm

- Objective function $\mathcal{L}(\boldsymbol{x})$ and its surrogate function $\mathcal{L}^+(\boldsymbol{x}, \boldsymbol{\xi})$ with additional variable $\boldsymbol{\xi}$.
- Surrogate function $\mathcal{L}^+(\boldsymbol{x}, \boldsymbol{\xi})$ is upper bound function of $\mathcal{L}(\boldsymbol{x})$, and is tangent to $\mathcal{L}(\boldsymbol{x})$ at $\boldsymbol{\xi} = \boldsymbol{x}$.

$$\mathcal{L}(\boldsymbol{x}) \leq \mathcal{L}^+(\boldsymbol{x}, \boldsymbol{\xi})$$
$$\mathcal{L}(\boldsymbol{x}) = \mathcal{L}^+(\boldsymbol{x}, \boldsymbol{x})$$

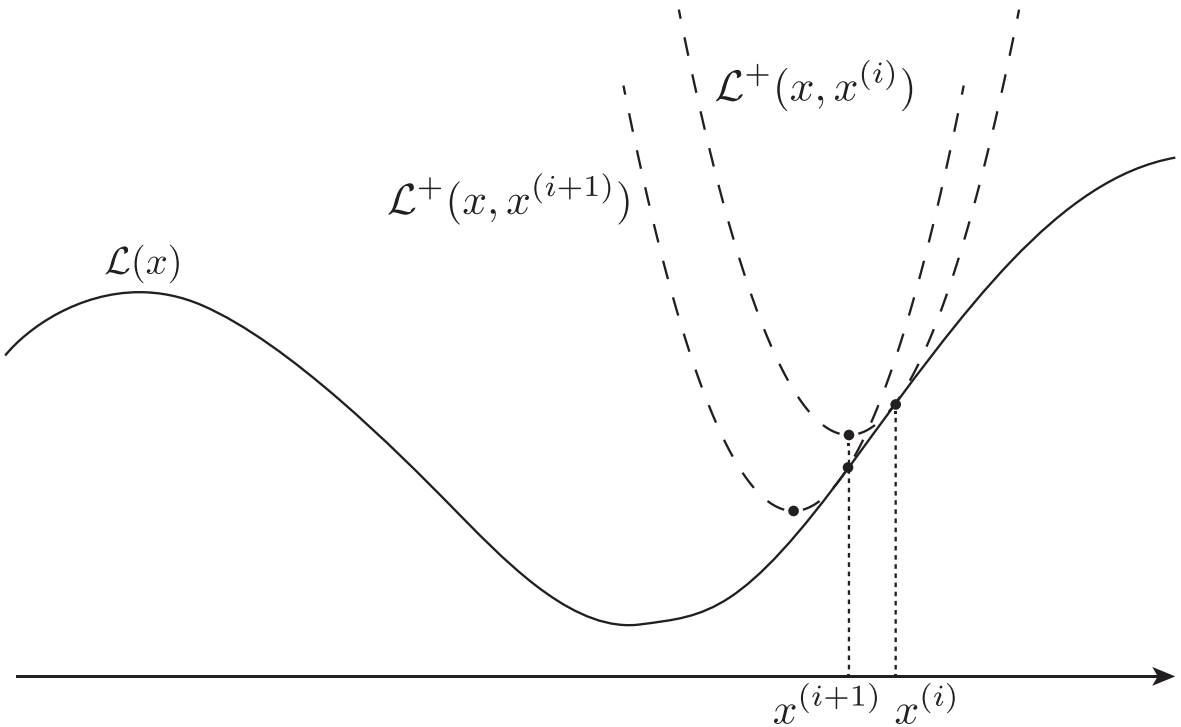
- Parameter to be optimized $\boldsymbol{x}$ is iteratively updated as

$$\boldsymbol{x}^{(i+1)} = \arg\min_{\boldsymbol{x}} \mathcal{L}^+(\boldsymbol{x}, \boldsymbol{x}^{(i)})$$

Majorization-minimization algorithm

Algorithm MM algorithm

1: Set initial value $\boldsymbol{x}^{(0)}$, and $i \leftarrow 0$
2: **while** stopping condition is not satisfied **do**
3: $\quad \boldsymbol{x}^{(i+1)} \leftarrow \arg\min_{\boldsymbol{x}} \mathcal{L}^{+}(\boldsymbol{x}, \boldsymbol{x}^{(i)})$
4: $\quad i \leftarrow i + 1$
5: **end while**



Majorization-minimization algorithm

- Monotonic non-increase of objective function is guaranteed as

$$\mathcal{L}(\boldsymbol{x}^{(i+1)}) \leq \mathcal{L}^{+}(\boldsymbol{x}^{(i+1)}, \boldsymbol{x}^{(i)}) \leq \mathcal{L}^{+}(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(i)}) = \mathcal{L}(\boldsymbol{x}^{(i)})$$

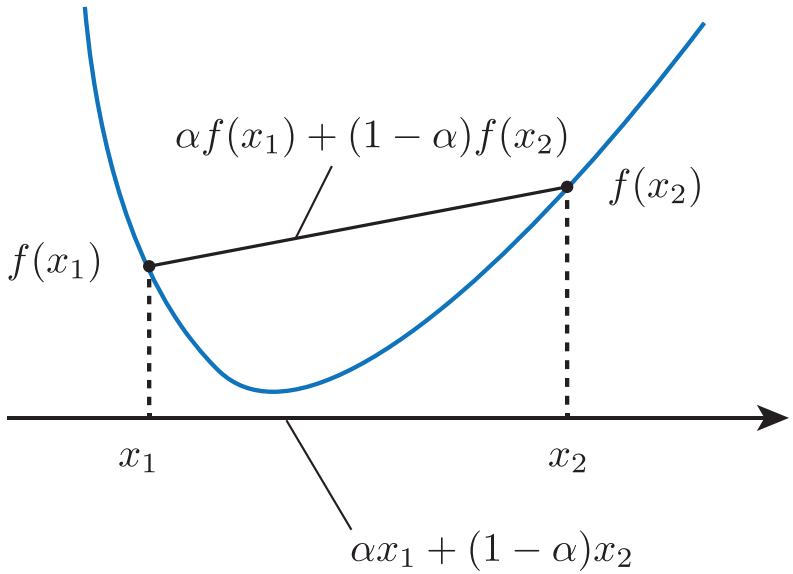
- Efficient update rule is obtained if minimization problem at each iteration can be simply solved, e.g., in closed form.
- $\mathcal{L}(\boldsymbol{x})$ can be nonconvex and/or indifferentiable.
- How to develop surrogate function is problem-dependent.

Tips on surrogate function design

- Jensen's inequality for convex function f

$$\sum_i \alpha_i f(\boldsymbol{x}_i) \geq f\left(\sum_i \alpha_i \boldsymbol{x}_i\right)$$

where $\sum_i \alpha_i = 1$ and $\alpha_i \geq 0 \ \ \forall i$. Equality holds if $\boldsymbol{x}_i$'s are equal.



Tips on surrogate function design

- Examples of Jensen's inequality
  - Negative log function

$$-\sum_i \alpha_i \ln x_i \geq -\ln\left(\sum_i \alpha_i x_i\right)$$

  - Inverse function

$$\sum_i \alpha_i \frac{1}{x_i} \geq \frac{1}{\sum_i \alpha_i x_i}$$

Again, $\sum_i \alpha_i = 1$ and $\alpha_i \geq 0 \ \ \forall i$. Equality holds if x_i's are equal.

Tips on surrogate function design

- Tangent line of concave function

$$f(\boldsymbol{\xi}) + \langle \nabla f(\boldsymbol{\xi}), \boldsymbol{x} - \boldsymbol{\xi} \rangle \geq f(\boldsymbol{x})$$

- Example:
  - Log function

$$\ln \xi + \frac{x}{\xi} - 1 \geq \ln x$$

    Equality holds for $x = \xi$.

- In general, several inequalities are combined to design surrogate function

Hyperparameter estimation

- Example of MM algorithm (actually, EM algorithm): hyperparameter estimation in linear discrete model
- MAP estimation in linear discrete model ($\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{n}$) with Gaussian likelihood $\mathcal{N}(\boldsymbol{n}|\boldsymbol{0}, \boldsymbol{\Lambda}^{-1})$ and prior $\mathcal{N}(\boldsymbol{x}|\boldsymbol{0}, \boldsymbol{\Lambda}_x^{-1})$

$$\hat{\boldsymbol{x}} = \left(\boldsymbol{\Lambda}_x + \boldsymbol{A}^{\mathsf{T}}\boldsymbol{\Lambda}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^{\mathsf{T}}\boldsymbol{\Lambda}\boldsymbol{y}$$

  requires to set hyperparameters, $\boldsymbol{\Lambda}_x$ and $\boldsymbol{\Lambda}$.
- Several techniques to determine hyperparameters (or regularization parameter) exist, such as (generalized) cross validation and L-curve method.
- A method to estimate $\boldsymbol{\Lambda}_x$ and $\boldsymbol{\Lambda}$ from observation data is introduced here.

Hyperparameter estimation

- Here, hyperparameters are denoted by one variable as $\boldsymbol{\theta} = \{\boldsymbol{\Lambda}_x, \boldsymbol{\Lambda}\}$ for notational simplicity.
- Let us consider ML estimate of $\boldsymbol{\theta}$:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{y}|\boldsymbol{\theta})$$

  which is referred to as *marginal likelihood* or *data evidence*.
- $p(\boldsymbol{y}|\boldsymbol{\theta})$ is represented by *marginalizing* $p(\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{\theta})$ w.r.t. $\boldsymbol{x}$ as

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = \int p(\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{\theta}) \mathrm{d}\boldsymbol{x} = \int p(\boldsymbol{y}|\boldsymbol{x}) p(\boldsymbol{x}|\boldsymbol{\theta}) \mathrm{d}\boldsymbol{x}$$

  However, the integral in right-hand side is hard to calculate.
- In EM algorithm, *average log likelihood* $\mathbb{E}[\ln p(\boldsymbol{y}, \boldsymbol{x}|\boldsymbol{\theta})]$ is used instead of *marginal log likelihood* $\ln p(\boldsymbol{y}|\boldsymbol{\theta})$.

Hyperparameter estimation

- Original cost function is marginal log likelihood:

$$\mathcal{L}(\boldsymbol{\theta}) = \ln p(\boldsymbol{y}|\boldsymbol{\theta}) = \int \ln p(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\theta}) \mathrm{d}\boldsymbol{x}$$

- By using posterior distribution with $\boldsymbol{\theta}^{(i)}$ ($\boldsymbol{\theta}$ at ith iteration), $\mathcal{L}(\boldsymbol{\theta})$ is written as

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}) &= \ln \int p(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\theta}) \mathrm{d}\boldsymbol{x} \\
&= \ln \int \frac{p(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\theta})}{p(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}^{(i)})} p(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}^{(i)}) \mathrm{d}\boldsymbol{x} \\
&= \ln \mathbb{E}\left[\frac{p(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\theta})}{p(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}^{(i)})}\right]
\end{aligned}$$

Hyperparameter estimation

- Based on Jensen's inequality, $\ln \mathbb{E}[f(\boldsymbol{x})] \geq \mathbb{E}[\ln f(\boldsymbol{x})]$, the following inequality holds:

$$
\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}) &= \ln \mathbb{E}\left[\frac{p(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\theta})}{p(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}^{(i)})}\right] \\
&\geq \mathbb{E}\left[\ln \frac{p(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\theta})}{p(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}^{(i)})}\right] \\
&= \int p(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}^{(i)}) \ln \frac{p(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\theta})}{p(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}^{(i)})} \mathrm{d}\boldsymbol{x} \\
&= \int p(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}^{(i)}) \ln p(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\theta}) \mathrm{d}\boldsymbol{x} \\
&\qquad - \int p(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}^{(i)}) \ln p(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}^{(i)}) \mathrm{d}\boldsymbol{x} \\
&:= \bar{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(i)})
\end{aligned}
$$

Equality holds for $\boldsymbol{\theta} = \boldsymbol{\theta}^{(i)}$.

Hyperparameter estimation

- First term is average log likelihood. Second term is entropy of $p(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}^{(i)})$, which does not depend on $\boldsymbol{\theta}$.

$$\begin{aligned}\bar{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(i)}) &= \int p(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}^{(i)}) \ln p(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\theta}) \mathrm{d}\boldsymbol{x} \\ &\quad - \int p(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}^{(i)}) \ln p(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}^{(i)}) \mathrm{d}\boldsymbol{x} \\ &= \Theta(\boldsymbol{\theta}|\boldsymbol{\theta}^{(i)}) + \mathcal{H}[p(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}^{(i)})]\end{aligned}$$

- By iteratively solving maximization problem of $\Theta(\boldsymbol{\theta}|\boldsymbol{\theta}^{(i)})$, monotonic non-decrease of $\mathcal{L}(\boldsymbol{\theta})$ is guaranteed.

$$\mathcal{L}(\boldsymbol{\theta}^{(i)}) = \bar{\mathcal{L}}(\boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(i)}) \leq \bar{\mathcal{L}}(\boldsymbol{\theta}^{(i+1)}, \boldsymbol{\theta}^{(i)}) = \mathcal{L}(\boldsymbol{\theta}^{(i+1)})$$

Hyperparameter estimation

- Specifically, average log likelihood is written as

$$\Theta(\boldsymbol{\Lambda}_x, \boldsymbol{\Lambda}) = \frac{1}{2}\ln|\boldsymbol{\Lambda}_x| - \frac{1}{2}\mathbb{E}[\boldsymbol{x}^\mathsf{T}\boldsymbol{\Lambda}_x\boldsymbol{x}] + \frac{1}{2}\ln|\boldsymbol{\Lambda}| - \frac{1}{2}\mathbb{E}[(\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x})^\mathsf{T}\boldsymbol{\Lambda}(\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x})]$$

  Maximization problem of $\Theta(\boldsymbol{\Lambda}_x, \boldsymbol{\Lambda})$ has closed-form solution.

- EM algorithm is summarized as follows:

E-step: Posterior average $\bar{\boldsymbol{x}}$ and precision matrix $\boldsymbol{\Gamma}$ are calculated as

$$\boldsymbol{\Gamma} = \boldsymbol{\Lambda}_x + \boldsymbol{A}^\mathsf{T}\boldsymbol{\Lambda}\boldsymbol{A}$$
$$\bar{\boldsymbol{x}} = \boldsymbol{\Gamma}^{-1}\boldsymbol{A}^\mathsf{T}\boldsymbol{\Lambda}\boldsymbol{y}$$

M-step: Update $\boldsymbol{\Lambda}_x$ and $\boldsymbol{\Lambda}$ as

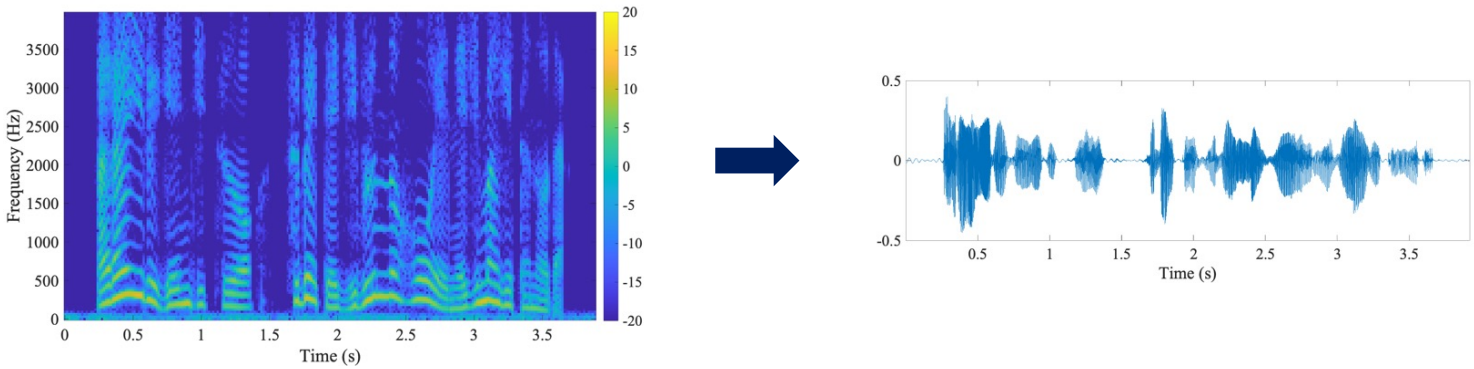
$$\hat{\boldsymbol{\Lambda}}_x^{-1} = \bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^\mathsf{T} + \boldsymbol{\Gamma}^{-1}$$
$$\hat{\boldsymbol{\Lambda}}^{-1} = (\boldsymbol{y}-\boldsymbol{A}\bar{\boldsymbol{x}})(\boldsymbol{y}-\boldsymbol{A}\bar{\boldsymbol{x}})^\mathsf{T} + \boldsymbol{A}\boldsymbol{\Gamma}^{-1}\boldsymbol{A}^\mathsf{T}$$

Phase retrieval

- Another example of MM algorithm: phase retrieval

$$\underset{\boldsymbol{x}\in\mathbb{C}^N}{\text{minimize}}\, \mathcal{L}(\boldsymbol{x}) := \|\boldsymbol{y} - |\boldsymbol{A}\boldsymbol{x}|\|_2^2$$

Here, $|\cdot|$ represents element-wise absolute value, $\boldsymbol{A} \in \mathbb{C}^{M\times N}$ is STFT matrix, $\boldsymbol{y} \in \mathbb{R}_+^M$ is given vectorized magnitude spectrogram, and $\boldsymbol{x} \in \mathbb{C}^N$ is time-domain signal to be recovered.



Phase retrieval

- By using $\boldsymbol{x}^{(i)} \in \mathbb{C}^N$, $\boldsymbol{v}^{(i)}$ is defined as

$$\boldsymbol{v}^{(i)} = \boldsymbol{y} \odot \exp\left(\mathrm{j} \arg(\boldsymbol{A}\boldsymbol{x}^{(i)})\right)$$

where $\arg(\cdot)$ is argument of complex variable. Then,

$$\| |\boldsymbol{A}\boldsymbol{x}| - \boldsymbol{y} \|^2 \leq \| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{v}^{(i)} \|^2$$

holds. Equality holds for

$$\boldsymbol{x} = \boldsymbol{x}^{(i)}$$

- Proof is omitted...

Phase retrieval

- Surrogate function for MM algorithm is constructed as

$$\mathcal{L}^{+}(\boldsymbol{x}, \boldsymbol{v}^{(i)}) := \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{v}^{(i)}\|^2$$

  which is simply minimized by

$$\arg\min_{\boldsymbol{x}} \mathcal{L}^{+}(\boldsymbol{x}, \boldsymbol{v}^{(i)}) = \left(\boldsymbol{A}^{\mathsf{H}}\boldsymbol{A}\right)^{-1} \boldsymbol{A}^{\mathsf{H}}\boldsymbol{v}^{(i)}$$

- MM algorithm for phase retrieval is as follows:
  1. Compute $\boldsymbol{v}^{(i)}$

$$\boldsymbol{v}^{(i)} = \boldsymbol{y} \odot \exp\left(\mathrm{j}\arg(\boldsymbol{A}\boldsymbol{x}^{(i)})\right)$$

  2. Update $\boldsymbol{x}$

$$\boldsymbol{x}^{(i+1)} = \left(\boldsymbol{A}^{\mathsf{H}}\boldsymbol{A}\right)^{-1} \boldsymbol{A}^{\mathsf{H}}\boldsymbol{v}^{(i)}$$

Phase retrieval

- MM algorithm for phase retrieval repeats
  1. Generation of time-frequency-domain signal with given magnitude and current phase
  2. Update estimate by inverse STFT
- Equivalent to Griffin–Lim algorithm.

References I

S. Haykin (2014)

Adaptive Filter Theory

Pearson Education Limited, Harlow.

S. Boyd and L. Vandenberghe (2004)

Convex Optimization

Cambridge University Press, Cambridge.

N. Parikh and S. Boyd (2013)

Proximal Algorithms

Foundations and Trends in Optimization, vol. 1, no. 3, pp. 123–231.

P. Combettes and J.-C. Pesquet (2011)

Proximal splitting methods in signal processing

Fixed-Point Algorithms for Inverse Problems in Science and Engineering, pp. 185–212.

References II

金森敬文, 鈴木大慈, 竹内一郎, 佐藤一誠 (2016)

機械学習のための連続最適化

講談社, Tokyo.

D. R. Hunter and K. Lange (2012)

A Tutorial on MM Algorithm

The American Statistician, vol. 58, no. 1, pp. 30–37.

Y. Sun, P. Babu, and D. P. Palomar (2017)

Majorization-Minimization Algorithms in Signal Processing, Communications, and Machine Learning

IEEE Trans. Signal Process., vol. 65, no. 3, pp. 794–816.