Sparsity Methods for Systems and Control

Algorithms for Convex Optimization

Masaaki Nagahara[1]

[1]The University of Kitakyushu
nagahara@ieee.org

Table of Contents

Table of Contents

ℓ^1 optimization by CVX

ℓ^1 Optimization

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \|x\|_1 \quad \text{subject to} \ \ \Phi x = y.$$

The MATLAB CVX code

```
cvx_begin
    variable x(n)
    minimize( norm(x, 1) )
    subject to
        y == Phi * x
cvx_end
```

- Useful for a small or middle scale problems
- Not that useful for
  - large-scale problems like image processing
  - real-time applications for control system
- You need to build an efficient algorithm by yourself for your specific problem.

ℓ^1 optimization by CVX

ℓ^1 Optimization

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \|x\|_1 \quad \text{subject to} \quad \Phi x = y.$$

The MATLAB CVX code

```
cvx_begin
    variable x(n)
    minimize( norm(x, 1) )
    subject to
        y == Phi * x
cvx_end
```

- Useful for a small or middle scale problems
- Not that useful for
    - large-scale problems like image processing
    - real-time applications for control systems
- You need to build an efficient algorithm by yourself for your specific problem.

ℓ^1 optimization by CVX

ℓ^1 Optimization

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \|x\|_1 \quad \text{subject to} \quad \Phi x = y.$$

The MATLAB CVX code

```
cvx_begin
    variable x(n)
    minimize( norm(x, 1) )
    subject to
        y == Phi * x
cvx_end
```

- Useful for a small or middle scale problems
- Not that useful for
  - large-scale problems like image processing
  - real-time applications for control system
- You need to build an efficient algorithm by yourself for your specific problem.

ℓ^1 optimization by CVX

ℓ^1 Optimization

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \|x\|_1 \quad \text{subject to} \ \Phi x = y.$$

The MATLAB CVX code

```
cvx_begin
    variable x(n)
    minimize( norm(x, 1) )
    subject to
        y == Phi * x
cvx_end
```

- Useful for a small or middle scale problems
- Not that useful for
  - large-scale problems like image processing
  - real-time applications for control system
- You need to build an efficient algorithm by yourself for your specific problem.

ℓ^1 optimization by CVX

ℓ^1 Optimization

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \|x\|_1 \quad \text{subject to} \ \Phi x = y.$$

The MATLAB CVX code

```
cvx_begin
    variable x(n)
    minimize( norm(x, 1) )
    subject to
        y == Phi * x
cvx_end
```

- Useful for a small or middle scale problems
- Not that useful for
  - large-scale problems like image processing
  - real-time applications for control system
- You need to build an efficient algorithm by yourself for your specific problem.

ℓ^1 optimization by CVX

ℓ^1 Optimization

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \ \|x\|_1 \quad \text{subject to} \ \ \Phi x = y.$$

The MATLAB CVX code

```
cvx_begin
    variable x(n)
    minimize( norm(x, 1) )
    subject to
        y == Phi * x
cvx_end
```

- Useful for a small or middle scale problems
- Not that useful for
  - large-scale problems like image processing
  - real-time applications for control system
- You need to build an efficient algorithm by yourself for your specific problem.

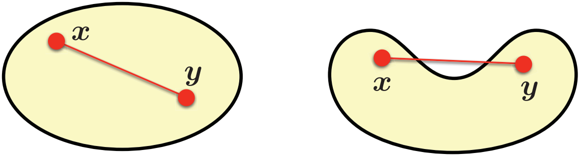
Convex set

Let C be a subset of $\mathbb{R}^n$. C is said to be a convex set if the following inclusion

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

holds for any vectors $\boldsymbol{x}, \boldsymbol{y} \in C$ and for any real number $t \in [0, 1]$.

- Convex and non-convex sets



Effective domain

- The effective domain of a function $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is defined by

$$\mathrm{dom}(f) \triangleq \{\boldsymbol{x} \in \mathbb{R}^n : f(\boldsymbol{x}) < \infty\}.$$

- Indicator function

$$f(\boldsymbol{x}) = \begin{cases} 0, & \text{if } \|\boldsymbol{x}\|_2 \le 1, \\ \infty, & \text{if } \|\boldsymbol{x}\|_2 > 1. \end{cases}$$

- The effective domain of the indicator function is

$$\mathrm{dom}(f) = \{\boldsymbol{x} \in \mathbb{R}^n : \|\boldsymbol{x}\|_2 \le 1\}.$$

Effective domain

- The effective domain of a function $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is defined by

$$\mathrm{dom}(f) \triangleq \{\boldsymbol{x} \in \mathbb{R}^n : f(\boldsymbol{x}) < \infty\}.$$

- Indicator function

$$f(\boldsymbol{x}) = \begin{cases} 0, & \text{if } \|\boldsymbol{x}\|_2 \leq 1, \\ \infty, & \text{if } \|\boldsymbol{x}\|_2 > 1. \end{cases}$$

- The effective domain of the indicator function is

$$\mathrm{dom}(f) = \{\boldsymbol{x} \in \mathbb{R}^n : \|\boldsymbol{x}\|_2 \leq 1\}.$$

Effective domain

- The effective domain of a function $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is defined by

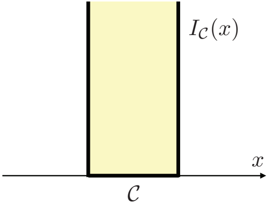
$$\mathrm{dom}(f) \triangleq \{\boldsymbol{x} \in \mathbb{R}^n : f(\boldsymbol{x}) < \infty\}.$$

- Indicator function

$$f(\boldsymbol{x}) = \begin{cases} 0, & \text{if } \|\boldsymbol{x}\|_2 \leq 1, \\ \infty, & \text{if } \|\boldsymbol{x}\|_2 > 1. \end{cases}$$

- The effective domain of the indicator function is

$$\mathrm{dom}(f) = \{\boldsymbol{x} \in \mathbb{R}^n : \|\boldsymbol{x}\|_2 \leq 1\}.$$



Convex function

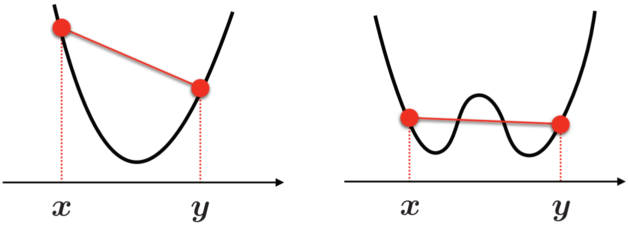
Convex function

Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be a proper function. The function f is said to be a convex function if the following inequality

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

holds for any vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathrm{dom}(f)$ and for any real number $t \in [0,1]$.

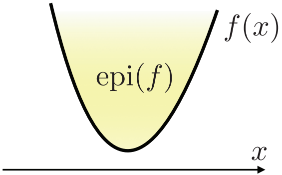
- Convex and non-convex functions



Epigraph

- The epigraph epi(f) of function f is defined by

$$\mathrm{epi}(f) \triangleq \left\{ (\boldsymbol{x}, t) \in \mathbb{R}^n \times \mathbb{R} : \boldsymbol{x} \in \mathrm{dom}(f),\, f(\boldsymbol{x}) \leq t \right\}.$$



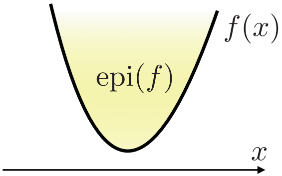
- Proper, convex, and closed function

function f	epigraph epi(f)
convex	convex set
closed	closed set
proper	non-empty

Epigraph

- The epigraph $\mathrm{epi}(f)$ of function f is defined by

$$\mathrm{epi}(f) \triangleq \left\{(\boldsymbol{x}, t) \in \mathbb{R}^n \times \mathbb{R} : \boldsymbol{x} \in \mathrm{dom}(f), f(\boldsymbol{x}) \leq t\right\}.$$



- Proper, convex, and closed function

function f	epigraph $\mathrm{epi}(f)$
convex	convex set
closed	closed set
proper	non-empty

Convex optimization problem

Convex optimization problem

Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be a proper, closed, and convex function, and $C \subset \mathbb{R}^n$ be a closed convex set. Then, a convex optimization problem is a problem to find a vector $\boldsymbol{x}^* \in \mathbb{R}^n$ that minimizes the function $f(\boldsymbol{x})$ over the set $C \subset \mathbb{R}^n$. The problem is briefly written as

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \;\; f(\boldsymbol{x}) \;\; \text{subject to} \;\; \boldsymbol{x} \in C.$$

- The function $f(x)$ is called a cost function or an objective function.
- The set C is called a constraint set or a feasible set.
- The entries of C is called feasible solutions.
- The inclusion $x \in C$ is called a constraint

Convex optimization problem

Convex optimization problem

Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be a proper, closed, and convex function, and $C \subset \mathbb{R}^n$ be a closed convex set. Then, a convex optimization problem is a problem to find a vector $\boldsymbol{x}^* \in \mathbb{R}^n$ that minimizes the function $f(\boldsymbol{x})$ over the set $C \subset \mathbb{R}^n$. The problem is briefly written as

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \ f(\boldsymbol{x}) \ \text{subject to} \ \boldsymbol{x} \in C.$$

- The function $f(\boldsymbol{x})$ is called a cost function or an objective function.
- The set C is called a constraint set or a feasible set.
- The entries of C is called feasible solutions.
- The inclusion $\boldsymbol{x} \in C$ is called a constraint

Convex optimization problem

Convex optimization problem

Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be a proper, closed, and convex function, and $C \subset \mathbb{R}^n$ be a closed convex set. Then, a convex optimization problem is a problem to find a vector $\boldsymbol{x}^* \in \mathbb{R}^n$ that minimizes the function $f(\boldsymbol{x})$ over the set $C \subset \mathbb{R}^n$. The problem is briefly written as

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \;\; f(\boldsymbol{x}) \;\; \text{subject to} \;\; \boldsymbol{x} \in C.$$

- The function $f(\boldsymbol{x})$ is called a cost function or an objective function.
- The set C is called a constraint set or a feasible set.
- The entries of C is called feasible solutions.
- The inclusion $\boldsymbol{x} \in C$ is called a constraint

Convex optimization problem

Convex optimization problem

Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be a proper, closed, and convex function, and $C \subset \mathbb{R}^n$ be a closed convex set. Then, a convex optimization problem is a problem to find a vector $\boldsymbol{x}^* \in \mathbb{R}^n$ that minimizes the function $f(\boldsymbol{x})$ over the set $C \subset \mathbb{R}^n$. The problem is briefly written as

$$\underset{\boldsymbol{x} \in \mathbb{R}^n}{\text{minimize}} \;\; f(\boldsymbol{x}) \;\; \text{subject to} \;\; \boldsymbol{x} \in C.$$

- The function $f(\boldsymbol{x})$ is called a cost function or an objective function.
- The set C is called a constraint set or a feasible set.
- The entries of C is called feasible solutions.
- The inclusion $\boldsymbol{x} \in C$ is called a constraint

Convex optimization problem

Convex optimization problem

Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be a proper, closed, and convex function, and $C \subset \mathbb{R}^n$ be a closed convex set. Then, a convex optimization problem is a problem to find a vector $\boldsymbol{x}^* \in \mathbb{R}^n$ that minimizes the function $f(\boldsymbol{x})$ over the set $C \subset \mathbb{R}^n$. The problem is briefly written as

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \;\; f(\boldsymbol{x}) \;\; \text{subject to} \;\; \boldsymbol{x} \in C.$$

- The function $f(\boldsymbol{x})$ is called a cost function or an objective function.
- The set C is called a constraint set or a feasible set.
- The entries of C is called feasible solutions.
- The inclusion $\boldsymbol{x} \in C$ is called a constraint

Notation

- Minimum value:

$$\min_{x \in C} f(x).$$

- Minimizer (set):

$$\arg\min_{x \in C} f(x) \triangleq \left\{ x^* \in C : f(x^*) \leq f(x), \ \ \forall x \in C \cap \mathrm{dom}(f) \right\}.$$

-

Notation

- Minimum value:

$$\min_{x \in C} f(x).$$

- Minimizer (set):

$$\arg\min_{x \in C} f(x) \triangleq \left\{ x^* \in C : f(x^*) \leq f(x), \ \forall x \in C \cap \mathrm{dom}(f) \right\}.$$

Notation

- Minimum value:

$$\min_{x \in C} f(x).$$

- Minimizer (set):

$$\arg\min_{x \in C} f(x) \triangleq \left\{ x^* \in C : f(x^*) \leq f(x), \ \forall x \in C \cap \mathrm{dom}(f) \right\}.$$

-

$$\underset{\boldsymbol{x} \in \mathbb{R}^n}{\text{minimize}} \ \underbrace{f(\boldsymbol{x})}_{\text{cost function}} \ \text{subject to} \ \underbrace{\boldsymbol{x} \in C}_{\text{constraint}}$$

$\min_{\boldsymbol{x} \in C} f(\boldsymbol{x})$ minimum value

$\arg\min_{\boldsymbol{x} \in C} f(\boldsymbol{x})$ minimizer (set)

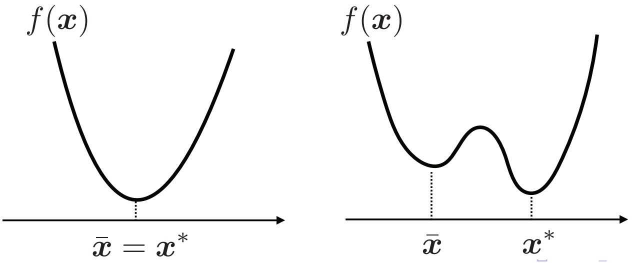
Global/local minimizers

- Local minimizer: there exists an open set $\mathcal{B}$ that contains a feasible solution $\bar{x} \in C \cap \mathrm{dom}(f)$ such that

$$f(x) \geq f(\bar{x}), \quad \forall x \in \mathcal{B} \cap C.$$

- Global minimizer: a feasible solution $x^* \in C$ that satisfies

$$f(x) \geq f(x^*), \quad \forall x \in C.$$



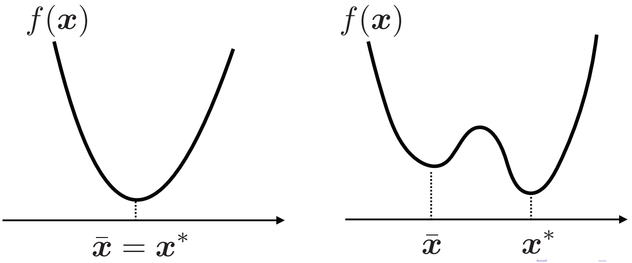
Global/local minimizers

- Local minimizer: there exists an open set $\mathcal{B}$ that contains a feasible solution $\bar{x} \in C \cap \mathrm{dom}(f)$ such that

$$f(x) \geq f(\bar{x}), \quad \forall x \in \mathcal{B} \cap C.$$

- Global minimizer: a feasible solution $x^* \in C$ that satisfies

$$f(x) \geq f(x^*), \quad \forall x \in C.$$



Global/local minimizers

Theorem

For the convex optimization problem,

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \ f(x) \ \text{ subject to } \ x \in C.$$

any local minimizer is (if it exists) a global minimizer, and the set of global minimizers is a convex set.

- For convex optimization problems, you just need to find a local minimizer, which is consequently a global minimizer.

Global/local minimizers

Theorem

For the convex optimization problem,

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ f(x) \ \text{subject to} \ x \in C.$$

any local minimizer is (if it exists) a global minimizer, and the set of global minimizers is a convex set.

- For convex optimization problems, you just need to find a local minimizer, which is consequently a global minimizer.

Strictly and strongly convex functions

- Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be a proper function.
- The function f is said to be a strictly convex function if for any $x, y \in \mathrm{dom}(f) \subset \mathbb{R}^n$ with $x \neq y$ and any $t \in (0, 1)$,

$$f\big(tx + (1-t)y\big) < tf(x) + (1-t)f(y)$$

- The function f is said to be a strongly convex function if there exists $\beta > 0$ such that for any $x, y \in \mathrm{dom}(f) \subset \mathbb{R}^n$ and any $t \in [0, 1]$,

$$f\big(tx + (1-t)y\big) \leq tf(x) + (1-t)f(y) - t(1-t)\frac{\beta}{2}\|x - y\|_2^2$$

The constant β is called a *modulus*.

Strictly and strongly convex functions

- Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be a proper function.
- The function f is said to be a strictly convex function if for any $\boldsymbol{x}, \boldsymbol{y} \in \mathrm{dom}(f) \subset \mathbb{R}^n$ with $\boldsymbol{x} \neq \boldsymbol{y}$ and any $t \in (0, 1)$,

$$f\big(t\boldsymbol{x} + (1-t)\boldsymbol{y}\big) < t f(\boldsymbol{x}) + (1-t) f(\boldsymbol{y})$$

- The function f is said to be a strongly convex function if there exists $\beta > 0$ such that for any $\boldsymbol{x}, \boldsymbol{y} \in \mathrm{dom}(f) \subset \mathbb{R}^n$ and any $t \in [0, 1]$,

$$f\big(t\boldsymbol{x} + (1-t)\boldsymbol{y}\big) \leq t f(\boldsymbol{x}) + (1-t) f(\boldsymbol{y}) - t(1-t)\frac{\beta}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2$$

The constant β is called a *modulus*.

Strictly and strongly convex functions

- Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be a proper function.
- The function f is said to be a strictly convex function if for any $\boldsymbol{x}, \boldsymbol{y} \in \mathrm{dom}(f) \subset \mathbb{R}^n$ with $\boldsymbol{x} \neq \boldsymbol{y}$ and any $t \in (0, 1)$,

$$f\big(t\boldsymbol{x} + (1-t)\boldsymbol{y}\big) < t f(\boldsymbol{x}) + (1-t) f(\boldsymbol{y})$$

- The function f is said to be a strongly convex function if there exists $\beta > 0$ such that for any $\boldsymbol{x}, \boldsymbol{y} \in \mathrm{dom}(f) \subset \mathbb{R}^n$ and any $t \in [0, 1]$,

$$f\big(t\boldsymbol{x} + (1-t)\boldsymbol{y}\big) \leq t f(\boldsymbol{x}) + (1-t) f(\boldsymbol{y}) - t(1-t)\frac{\beta}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2$$

The constant β is called a *modulus*.

Strongly convex functions

Theorem

Assume $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is a proper, closed, and strongly convex function with modulus $\beta > 0$. Then f has the unique minimizer $\boldsymbol{x}^* \in \mathrm{dom}(f)$. That is, for all $\boldsymbol{x} \in \mathrm{dom}(f)$ such that $\boldsymbol{x} \neq \boldsymbol{x}^*$,

$$f(\boldsymbol{x}) > f(\boldsymbol{x}^*).$$

Moreover, for any $\boldsymbol{x} \in \mathrm{dom}(f)$, we have

$$f(\boldsymbol{x}) \geq f(\boldsymbol{x}^*) + \frac{\beta}{2}\|\boldsymbol{x} - \boldsymbol{x}^*\|_2^2.$$

- This is an important property of strongly convex functions.
- This is used to define the proximal operator (see next Section).

Strongly convex functions

Theorem

Assume $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is a proper, closed, and strongly convex function with modulus $\beta > 0$. Then f has the unique minimizer $\boldsymbol{x}^* \in \mathrm{dom}(f)$. That is, for all $\boldsymbol{x} \in \mathrm{dom}(f)$ such that $\boldsymbol{x} \neq \boldsymbol{x}^*$,

$$f(\boldsymbol{x}) > f(\boldsymbol{x}^*).$$

Moreover, for any $\boldsymbol{x} \in \mathrm{dom}(f)$, we have

$$f(\boldsymbol{x}) \geq f(\boldsymbol{x}^*) + \frac{\beta}{2}\|\boldsymbol{x} - \boldsymbol{x}^*\|_2^2.$$

- This is an important property of strongly convex functions.
- This is used to define the proximal operator (see next Section).

Strongly convex functions

Theorem

Assume $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is a proper, closed, and strongly convex function with modulus $\beta > 0$. Then f has the unique minimizer $\boldsymbol{x}^* \in \mathrm{dom}(f)$. That is, for all $\boldsymbol{x} \in \mathrm{dom}(f)$ such that $\boldsymbol{x} \neq \boldsymbol{x}^*$,

$$f(\boldsymbol{x}) > f(\boldsymbol{x}^*).$$

Moreover, for any $\boldsymbol{x} \in \mathrm{dom}(f)$, we have

$$f(\boldsymbol{x}) \geq f(\boldsymbol{x}^*) + \frac{\beta}{2}\|\boldsymbol{x} - \boldsymbol{x}^*\|_2^2.$$

- This is an important property of strongly convex functions.
- This is used to define the proximal operator (see next Section).

Table of Contents

Proximal operator

Proximal operator

Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be a proper, closed, and convex function. The proximal operator $\mathrm{prox}_{\gamma f}$ with parameter $\gamma > 0$ is defined by

$$\mathrm{prox}_{\gamma f}(v) \triangleq \underset{x \in \mathrm{dom}(f)}{\arg\min} \left\{ f(x) + \frac{1}{2\gamma} \|x - v\|_2^2 \right\}.$$

- $\gamma = \infty$: Minimizer of $f(z)$:

$$\mathrm{prox}_{\gamma f}(v) = \underset{x \in \mathrm{dom}(f)}{\arg\min} f(x)$$

- $\gamma = 0$: Projection onto $\mathrm{dom}(f)$:

$$\mathrm{prox}_{\gamma f}(v) = \underset{x \in \mathrm{dom}(f)}{\arg\min} \frac{1}{2\gamma} \|x - v\|_2^2$$

- $\gamma \in (0, \infty)$: a mixture of those.

Proximal operator

Proximal operator

Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be a proper, closed, and convex function. The proximal operator $\mathrm{prox}_{\gamma f}$ with parameter $\gamma > 0$ is defined by

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

- $\gamma = \infty$: Minimizer of $f(z)$:

$$\mathrm{prox}_{\gamma f}(\boldsymbol{v}) = \underset{\boldsymbol{x} \in \mathrm{dom}(f)}{\arg\min}\, f(\boldsymbol{x})$$

- $\gamma = 0$: Projection onto $\mathrm{dom}(f)$:

$$\mathrm{prox}_{\gamma f}(\boldsymbol{v}) = \underset{\boldsymbol{x} \in \mathrm{dom}(f)}{\arg\min}\, \frac{1}{2\gamma} \|\boldsymbol{x} - \boldsymbol{v}\|_2^2$$

- $\gamma \in (0, \infty)$: a mixture of those.

Proximal operator

Proximal operator

Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be a proper, closed, and convex function. The proximal operator $\mathrm{prox}_{\gamma f}$ with parameter $\gamma > 0$ is defined by

$$\mathrm{prox}_{\gamma f}(v) \triangleq \mathop{\arg\min}_{x \in \mathrm{dom}(f)} \left\{ f(x) + \frac{1}{2\gamma}\|x - v\|_2^2 \right\}.$$

- $\gamma = \infty$: Minimizer of $f(z)$:

$$\mathrm{prox}_{\gamma f}(v) = \mathop{\arg\min}_{x \in \mathrm{dom}(f)} f(x)$$

- $\gamma = 0$: Projection onto $\mathrm{dom}(f)$:

$$\mathrm{prox}_{\gamma f}(v) = \mathop{\arg\min}_{x \in \mathrm{dom}(f)} \frac{1}{2\gamma}\|x - v\|_2^2$$

- $\gamma \in (0, \infty)$: a mixture of those.

Proximal operator

Proximal operator

Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be a proper, closed, and convex function. The proximal operator $\mathrm{prox}_{\gamma f}$ with parameter $\gamma > 0$ is defined by

$$\mathrm{prox}_{\gamma f}(\boldsymbol{v}) \triangleq \operatorname*{arg\,min}_{\boldsymbol{x} \in \mathrm{dom}(f)} \left\{ f(\boldsymbol{x}) + \frac{1}{2\gamma} \|\boldsymbol{x} - \boldsymbol{v}\|_2^2 \right\}.$$

- $\gamma = \infty$: Minimizer of $f(z)$:

$$\mathrm{prox}_{\gamma f}(\boldsymbol{v}) = \operatorname*{arg\,min}_{\boldsymbol{x} \in \mathrm{dom}(f)} f(\boldsymbol{x})$$

- $\gamma = 0$: Projection onto $\mathrm{dom}(f)$:

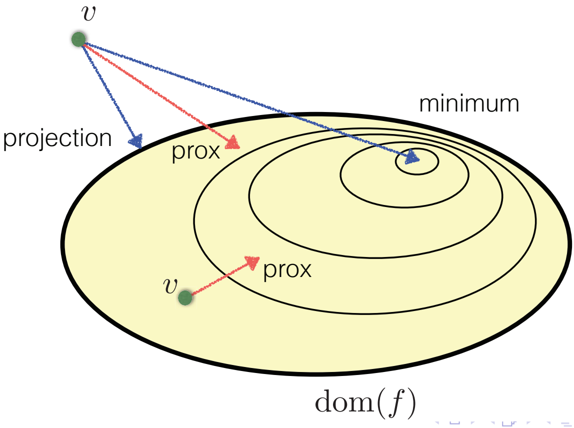
$$\mathrm{prox}_{\gamma f}(\boldsymbol{v}) = \operatorname*{arg\,min}_{\boldsymbol{x} \in \mathrm{dom}(f)} \frac{1}{2\gamma} \|\boldsymbol{x} - \boldsymbol{v}\|_2^2$$

- $\gamma \in (0, \infty)$: a mixture of those.

Proximal operator

Proximal operator

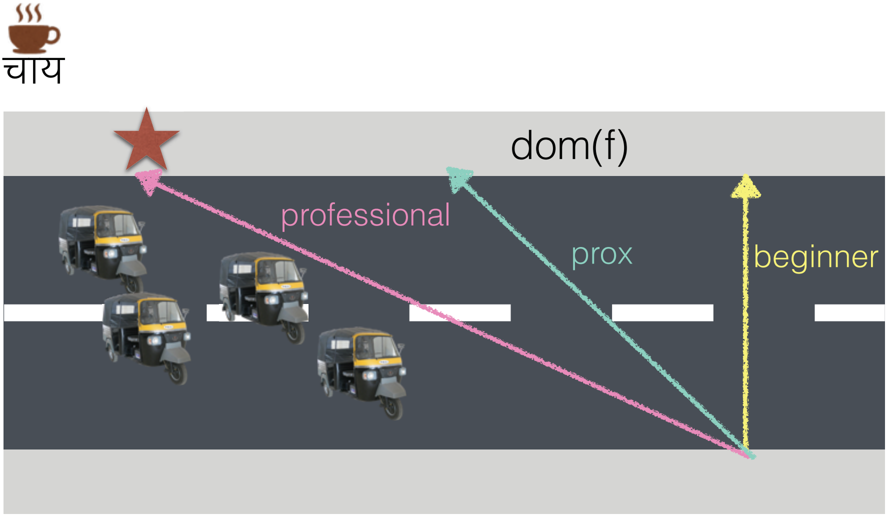
$$\mathrm{prox}_{\gamma f}(v) \triangleq \underset{x \in \mathrm{dom}(f)}{\arg\min} \left\{ f(x) + \frac{1}{2\gamma} \|x - v\|_2^2 \right\}.$$



Proximal operator

The "crossing the street" problem.

The "crossing the street" problem.



Proximal algorithm

Proximal algorithm

Initialization: give an initial vector $\boldsymbol{x}[0]$ and positive numbers $\gamma_0, \gamma_1, \gamma_2, \ldots$

Iteration: for $k = 0, 1, 2, \ldots,$ do

$$x[k+1] = \mathrm{prox}_{\gamma_k f}(x[k]) = \underset{x \in \mathrm{dom}(f)}{\arg\min} \Big\{ f(x) + \frac{1}{2\gamma_k} \|x - x[k]\|_2^2 \Big\}.$$

- The algorithm minimizes the strongly convex function

$$g_k(x) \triangleq f(x) + \frac{1}{2\gamma_k} \|x - x[k]\|_2^2$$

at each step k, which is an approximation of f that may not be strongly convex.

Proximal algorithm

Proximal algorithm

Initialization: give an initial vector $\boldsymbol{x}[0]$ and positive numbers $\gamma_0, \gamma_1, \gamma_2, \dots$
Iteration: for $k = 0, 1, 2, \dots,$ do

$$x[k+1] = \mathrm{prox}_{\gamma_k f}(x[k]) = \underset{x \in \mathrm{dom}(f)}{\arg\min} \Big\{ f(x) + \frac{1}{2\gamma_k} \|x - x[k]\|_2^2 \Big\}.$$

- The algorithm minimizes the strongly convex function

$$g_k(x) \triangleq f(x) + \frac{1}{2\gamma_k} \|x - x[k]\|_2^2$$

at each step k, which is an approximation of f that may not be strongly convex.

Proximal algorithm

Initialization: give an initial vector $\boldsymbol{x}[0]$ and positive numbers $\gamma_0, \gamma_1, \gamma_2, \ldots$

Iteration: for $k = 0, 1, 2, \ldots,$ do

$$\boldsymbol{x}[k+1] = \mathrm{prox}_{\gamma_k f}(\boldsymbol{x}[k]) = \mathop{\arg\min}_{\boldsymbol{x}\in\mathrm{dom}(f)} \left\{ f(\boldsymbol{x}) + \frac{1}{2\gamma_k}\|\boldsymbol{x} - \boldsymbol{x}[k]\|_2^2 \right\}.$$

- The algorithm minimizes the strongly convex function

$$g_k(\boldsymbol{x}) \triangleq f(\boldsymbol{x}) + \frac{1}{2\gamma_k}\|\boldsymbol{x} - \boldsymbol{x}[k]\|_2^2$$

at each step k, which is an approximation of f that may not be strongly convex.

Proximal algorithm

Initialization: give an initial vector $\boldsymbol{x}[0]$ and positive numbers $\gamma_0, \gamma_1, \gamma_2, \ldots$
Iteration: for $k = 0, 1, 2, \ldots,$ do

$$\boldsymbol{x}[k+1] = \mathrm{prox}_{\gamma_k f}(\boldsymbol{x}[k]) = \arg\min_{\boldsymbol{x} \in \mathrm{dom}(f)} \left\{ f(\boldsymbol{x}) + \frac{1}{2\gamma_k} \|\boldsymbol{x} - \boldsymbol{x}[k]\|_2^2 \right\}.$$

- The algorithm minimizes the strongly convex function

$$g_k(\boldsymbol{x}) \triangleq f(\boldsymbol{x}) + \frac{1}{2\gamma_k} \|\boldsymbol{x} - \boldsymbol{x}[k]\|_2^2$$

  at each step k, which is an approximation of f that may not be strongly convex.

Convergence theorem of proximal algorithm

Theorem

Suppose that the parameter sequence $\{\gamma_k\}$ satisfies $\gamma_k > 0$ for all k and

$$\sum_{k=0}^{\infty} \gamma_k = \infty.$$

Then, the vector sequence $\{x[k]\}$ generated by the proximal algorithm

$$x[k+1] = \mathrm{prox}_{\gamma_k f}(x[k])$$

converges to one of the minimizers of f for any initial vector $x[0]$.

Convergence theorem of proximal algorithm

Theorem

Suppose that the parameter sequence $\{\gamma_k\}$ satisfies $\gamma_k > 0$ for all k and

$$\sum_{k=0}^{\infty} \gamma_k = \infty.$$

Then, the vector sequence $\{\boldsymbol{x}[k]\}$ generated by the proximal algorithm

$$\boldsymbol{x}[k+1] = \mathrm{prox}_{\gamma_k f}(\boldsymbol{x}[k])$$

converges to one of the minimizers of f for any initial vector $\boldsymbol{x}[0]$.

Proximable functions

- Proximal operator

$$\mathrm{prox}_{\gamma f}(v) \triangleq \mathop{\arg\min}_{x \in \mathrm{dom}(f)} \left\{ f(x) + \frac{1}{2\gamma} \|x - v\|_2^2 \right\}.$$

  needs to be obtained in a closed form to derive an efficient algorithm.
- A proximable function is a function that has a closed-form proximal operator.
- The following functions are proximable:
  - quadratic functions (including the ℓ^2 norm)
  - indicator functions
  - the ℓ^1 norm

Proximable functions

- Proximal operator

$$\mathrm{prox}_{\gamma f}(v) \triangleq \underset{x \in \mathrm{dom}(f)}{\arg\min} \left\{ f(x) + \frac{1}{2\gamma} \|x - v\|_2^2 \right\}.$$

needs to be obtained in a closed form to derive an efficient algorithm.

- A proximable function is a function that has a closed-form proximal operator.
- The following functions are proximable:
    - quadratic functions (including the ℓ^2 norm)
    - indicator functions
    - the ℓ^1 norm

Proximable functions

- Proximal operator

$$\mathrm{prox}_{\gamma f}(v) \triangleq \mathop{\arg\min}_{x \in \mathrm{dom}(f)} \left\{ f(x) + \frac{1}{2\gamma} \|x - v\|_2^2 \right\}.$$

  needs to be obtained in a closed form to derive an efficient algorithm.
- A proximable function is a function that has a closed-form proximal operator.
- The following functions are proximable:
  - quadratic functions (including the ℓ^2 norm)
  - indicator functions
  - the ℓ^1 norm

Proximable functions

- Proximal operator

$$\mathrm{prox}_{\gamma f}(v) \triangleq \arg\min_{x \in \mathrm{dom}(f)} \left\{ f(x) + \frac{1}{2\gamma} \|x - v\|_2^2 \right\}.$$

needs to be obtained in a closed form to derive an efficient algorithm.

- A proximable function is a function that has a closed-form proximal operator.
- The following functions are proximable:
  - quadratic functions (including the ℓ^2 norm)
  - indicator functions
  - the ℓ^1 norm

Proximable functions

- Proximal operator

$$\mathrm{prox}_{\gamma f}(v) \triangleq \mathop{\arg\min}_{x \in \mathrm{dom}(f)} \left\{ f(x) + \frac{1}{2\gamma} \|x - v\|_2^2 \right\}.$$

  needs to be obtained in a closed form to derive an efficient algorithm.
- A proximable function is a function that has a closed-form proximal operator.
- The following functions are proximable:
  - quadratic functions (including the ℓ^2 norm)
  - indicator functions
  - the ℓ^1 norm

Proximable functions

- Proximal operator

$$\operatorname{prox}_{\gamma f}(v) \triangleq \underset{x \in \operatorname{dom}(f)}{\arg\min} \left\{ f(x) + \frac{1}{2\gamma} \|x - v\|_2^2 \right\}.$$

  needs to be obtained in a closed form to derive an efficient algorithm.
- A proximable function is a function that has a closed-form proximal operator.
- The following functions are proximable:
  - quadratic functions (including the ℓ^2 norm)
  - indicator functions
  - the ℓ^1 norm

Quadratic function

- The quadratic function

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top \Phi \boldsymbol{x} - \boldsymbol{y}^\top \boldsymbol{x},$$

  where Φ is a positive-definite matrix.

- The proximal operator is given by

$$\begin{aligned} \mathrm{prox}_{\gamma f}(\boldsymbol{v}) &= \arg\min_{\boldsymbol{x}\in\mathbb{R}^n} \left\{ \frac{1}{2}\boldsymbol{x}^\top \Phi \boldsymbol{x} - \boldsymbol{y}^\top \boldsymbol{x} + \frac{1}{2\gamma}(\boldsymbol{x}-\boldsymbol{v})^\top(\boldsymbol{x}-\boldsymbol{v}) \right\} \\ &= \left(\Phi + \frac{1}{\gamma}I\right)^{-1}\left(\boldsymbol{y} + \frac{1}{\gamma}\boldsymbol{v}\right). \end{aligned}$$

Quadratic function

- The quadratic function

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top \Phi \boldsymbol{x} - \boldsymbol{y}^\top \boldsymbol{x},$$

  where Φ is a positive-definite matrix.
- The proximal operator is given by

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

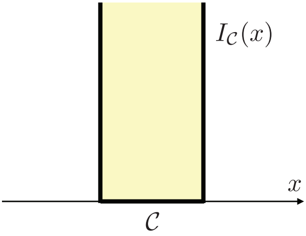
Indicator function

Indicator function

For a subset C in $\mathbb{R}^n$, the indicator function is defined by

$$I_C(\boldsymbol{x}) = \begin{cases} 0, & \boldsymbol{x} \in C \\ \infty, & \boldsymbol{x} \notin C \end{cases}$$

- C: non-empty, closed, and convex $\Rightarrow I_C(\boldsymbol{x})$: a proper, closed, and convex function
- Draw the epigraph of I_C.



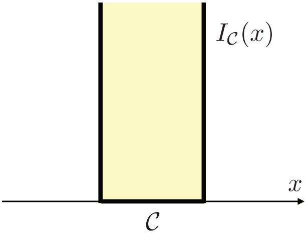
Indicator function

Indicator function

For a subset C in $\mathbb{R}^n$, the indicator function is defined by

$$I_C(\boldsymbol{x}) = \begin{cases} 0, & \boldsymbol{x} \in C \\ \infty, & \boldsymbol{x} \notin C \end{cases}$$

- C: non-empty, closed, and convex $\Rightarrow$ $I_C(\boldsymbol{x})$: a proper, closed, and convex function
- Draw the epigraph of I_C.



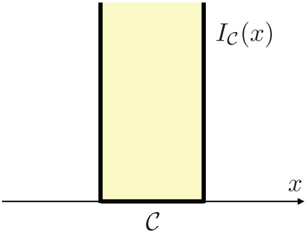
Indicator function

Indicator function

For a subset C in $\mathbb{R}^n$, the indicator function is defined by

$$I_C(\boldsymbol{x}) = \begin{cases} 0, & \boldsymbol{x} \in C \\ \infty, & \boldsymbol{x} \notin C \end{cases}$$

- C: non-empty, closed, and convex $\Rightarrow I_C(\boldsymbol{x})$: a proper, closed, and convex function
- Draw the epigraph of I_C.



Indicator function

- The proximal operator of $I_C(\boldsymbol{x})$ is given by

$$
\begin{aligned}
\operatorname{prox}_{\gamma I_C}(\boldsymbol{v}) &= \underset{\boldsymbol{x}\in\mathbb{R}^n}{\arg\min} \left\{ I_C(\boldsymbol{x}) + \frac{1}{2\gamma}\|\boldsymbol{x}-\boldsymbol{v}\|_2^2 \right\} \\
&= \underset{\boldsymbol{x}\in C}{\arg\min}\, \|\boldsymbol{x}-\boldsymbol{v}\|_2^2 \\
&= \Pi_C(\boldsymbol{v}).
\end{aligned}
$$

where Π_C is the projection operator onto the nonempty, closed, and convex set C.

Indicator function

- The proximal operator of $I_C(\boldsymbol{x})$ is given by

$$\begin{aligned}\operatorname{prox}_{\gamma I_C}(\boldsymbol{v}) &= \underset{\boldsymbol{x}\in\mathbb{R}^n}{\arg\min}\ \left\{ I_C(\boldsymbol{x}) + \frac{1}{2\gamma}\|\boldsymbol{x}-\boldsymbol{v}\|_2^2 \right\} \\ &= \underset{\boldsymbol{x}\in C}{\arg\min}\ \|\boldsymbol{x}-\boldsymbol{v}\|_2^2 \\ &= \Pi_C(\boldsymbol{v}).\end{aligned}$$

where Π_C is the projection operator onto the nonempty, closed, and convex set C.

Indicator function

- The proximal operator of $I_C(\boldsymbol{x})$ is given by

$$
\begin{aligned}
\mathrm{prox}_{\gamma I_C}(\boldsymbol{v}) &= \operatorname*{arg\,min}_{\boldsymbol{x}\in\mathbb{R}^n} \left\{ I_C(\boldsymbol{x}) + \frac{1}{2\gamma}\|\boldsymbol{x}-\boldsymbol{v}\|_2^2 \right\} \\
&= \operatorname*{arg\,min}_{\boldsymbol{x}\in C} \|\boldsymbol{x}-\boldsymbol{v}\|_2^2 \\
&= \Pi_C(\boldsymbol{v}).
\end{aligned}
$$

where Π_C is the projection operator onto the nonempty, closed, and convex set C.

ℓ^1 norm

- The proximal operator of the ℓ^1 norm $\|\boldsymbol{x}\|_1$ has a closed form

$$\mathrm{prox}_{\gamma\|\cdot\|_1}(\boldsymbol{v}) = S_\gamma(\boldsymbol{v}),$$

where $S_\gamma : \mathbb{R}^n \to \mathbb{R}^n$ is the soft-thresholding operator defined by

$$[S_\gamma(\boldsymbol{v})]_i = \begin{cases} v_i - \gamma, & v_i \geq \gamma, \\ 0, & -\gamma < v_i < \gamma, \\ v_i + \gamma, & v_i \leq -\gamma. \end{cases}$$

ℓ^1 norm

- The proximal operator of the ℓ^1 norm $\|\boldsymbol{x}\|_1$ has a closed form

$$\mathrm{prox}_{\gamma\|\cdot\|_1}(\boldsymbol{v}) = S_\gamma(\boldsymbol{v}),$$

where $S_\gamma : \mathbb{R}^n \to \mathbb{R}^n$ is the soft-thresholding operator defined by

$$[S_\gamma(\boldsymbol{v})]_i = \begin{cases} v_i - \gamma, & v_i \geq \gamma, \\ 0, & -\gamma < v_i < \gamma, \\ v_i + \gamma, & v_i \leq -\gamma. \end{cases}$$

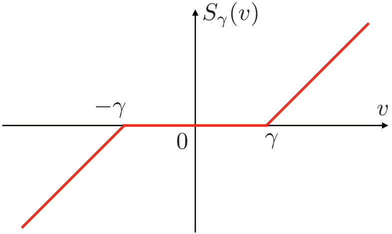


Table of Contents

ℓ^1 optimization

ℓ^1 optimization

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \|x\|_1 \quad \text{subject to} \quad \Phi x = y,$$

- $\Phi \in \mathbb{R}^{m \times n}$ and $y \in \mathbb{R}^m$ are given
- $m < n$
- Φ has full row rank, that is, $\text{rank}(\Phi) = m$.

ℓ^1 optimization

ℓ^1 optimization

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \|x\|_1 \quad \text{subject to} \ \ \Phi x = y,$$

- $\Phi \in \mathbb{R}^{m \times n}$ and $y \in \mathbb{R}^m$ are given
- $m < n$
- Φ has full row rank, that is, $\text{rank}(\Phi) = m$.

ℓ^1 optimization

ℓ^1 optimization

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \; \|x\|_1 \quad \text{subject to} \quad \Phi x = y,$$

- $\Phi \in \mathbb{R}^{m \times n}$ and $y \in \mathbb{R}^m$ are given
- $m < n$
- Φ has full row rank, that is, $\text{rank}(\Phi) = m$.

ℓ^1 optimization

ℓ^1 optimization

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \ \|\boldsymbol{x}\|_1 \quad \text{subject to} \ \Phi\boldsymbol{x} = \boldsymbol{y},$$

- $\Phi \in \mathbb{R}^{m\times n}$ and $\boldsymbol{y} \in \mathbb{R}^m$ are given
- $m < n$
- Φ has full row rank, that is, $\text{rank}(\Phi) = m$.

ℓ^1 optimization

ℓ^1 optimization

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \|x\|_1 \quad \text{subject to} \quad \Phi x = y,$$

- Constraint set

$$C \triangleq \{x \in \mathbb{R}^n : \Phi x = y\}.$$

- Indicator function

$$I_C(x) = \begin{cases} 0, & \text{if } \Phi x = y, \\ \infty, & \text{if } \Phi x \neq y. \end{cases}$$

- Equivalent unconstrained optimization

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \|x\|_1 + I_C(x).$$

ℓ^1 optimization

ℓ^1 optimization

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \|x\|_1 \quad \text{subject to} \quad \Phi x = y,$$

- Constraint set

$$C \triangleq \{x \in \mathbb{R}^n : \Phi x = y\}.$$

- Indicator function

$$I_C(x) = \begin{cases} 0, & \text{if } \Phi x = y, \\ \infty, & \text{if } \Phi x \neq y. \end{cases}$$

- Equivalent unconstrained optimization

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \|x\|_1 + I_C(x).$$

ℓ^1 optimization

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \|x\|_1 \quad \text{subject to} \quad \Phi x = y,$$

- Constraint set

$$C \triangleq \{x \in \mathbb{R}^n : \Phi x = y\}.$$

- Indicator function

$$I_C(x) = \begin{cases} 0, & \text{if } \Phi x = y, \\ \infty, & \text{if } \Phi x \neq y. \end{cases}$$

- Equivalent unconstrained optimization

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \|x\|_1 + I_C(x).$$

Splitting method

- Equivalent unconstrained optimization

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}}\ \|\boldsymbol{x}\|_1 + I_C(\boldsymbol{x}).$$

- $\|x\|_1 + I_C(x)$ is proper, closed, and convex but not proximable.
  - The proximal algorithm cannot be directly applied.
  - Both functions,

$$f_1(x) \triangleq \|x\|_1, \quad f_2(x) \triangleq I_C(x)$$

  are proximable.
- We split the cost function as $f = f_1 + f_2$.

Splitting method

- Equivalent unconstrained optimization

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \|x\|_1 + I_C(x).$$

- $\|x\|_1 + I_C(x)$ is proper, closed, and convex but not proximable.
  - The proximal algorithm cannot be directly applied.
  - Both functions,

$$f_1(x) \triangleq \|x\|_1, \quad f_2(x) \triangleq I_C(x)$$

  are proximable

- We split the cost function as $f = f_1 + f_2$.

Splitting method

- Equivalent unconstrained optimization

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \|x\|_1 + I_C(x).$$

- $\|x\|_1 + I_C(x)$ is proper, closed, and convex but not proximable.
  - The proximal algorithm cannot be directly applied.
  - Both functions,

$$f_1(x) \triangleq \|x\|_1, \quad f_2(x) \triangleq I_C(x)$$

    are proximable

- We split the cost function as $f = f_1 + f_2$.

Splitting method

- Equivalent unconstrained optimization

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \ \|\boldsymbol{x}\|_1 + I_C(\boldsymbol{x}).$$

- $\|\boldsymbol{x}\|_1 + I_C(\boldsymbol{x})$ is proper, closed, and convex but not proximable.
  - The proximal algorithm cannot be directly applied.
  - Both functions,

$$f_1(\boldsymbol{x}) \triangleq \|\boldsymbol{x}\|_1, \quad f_2(\boldsymbol{x}) \triangleq I_C(\boldsymbol{x})$$

    are proximable

- We split the cost function as $f = f_1 + f_2$.

Splitting method

- Equivalent unconstrained optimization

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \ \|\boldsymbol{x}\|_1 + I_C(\boldsymbol{x}).$$

- $\|\boldsymbol{x}\|_1 + I_C(\boldsymbol{x})$ is proper, closed, and convex but not proximable.
  - The proximal algorithm cannot be directly applied.
  - Both functions,

$$f_1(\boldsymbol{x}) \triangleq \|\boldsymbol{x}\|_1, \quad f_2(\boldsymbol{x}) \triangleq I_C(\boldsymbol{x})$$

    are proximable
- We split the cost function as $f = f_1 + f_2$.

Douglas-Rachford splitting algorithm

- General optimization problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \; f_1(x) + f_2(x),$$

- f_1 and f_2 are proper, closed, and convex functions.
- f_1 and f_2 are proximable.

Douglas-Rachford splitting algorithm

Initialization: give an initial vector $z[0]$ and a parameter $\gamma > 0$
Iteration: for $k = 0, 1, 2, \ldots$ do

$$x[k+1] = \operatorname{prox}_{\gamma f_1}(z[k])$$

$$z[k+1] = z[k] + \operatorname{prox}_{\gamma f_2}(2x[k+1] - z[k]) - x[k+1]$$

Douglas-Rachford splitting algorithm

- General optimization problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \;\; f_1(x) + f_2(x),$$

- f_1 and f_2 are proper, closed, and convex functions.
- f_1 and f_2 are proximable.

Douglas-Rachford splitting algorithm

Initialization: give an initial vector $z[0]$ and a parameter $\gamma > 0$
Iteration: for $k = 0, 1, 2, \ldots$ do

$$x[k+1] = \text{prox}_{\gamma f_1}(z[k])$$

$$z[k+1] = z[k] + \text{prox}_{\gamma f_2}(2x[k+1] - z[k]) - x[k+1]$$

Douglas-Rachford splitting algorithm

- General optimization problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \;\; f_1(x) + f_2(x),$$

- f_1 and f_2 are proper, closed, and convex functions.
- f_1 and f_2 are proximable.

Douglas-Rachford splitting algorithm

Initialization: give an initial vector $z[0]$ and a parameter $\gamma > 0$
Iteration: for $k = 0, 1, 2, \ldots$ do

$$x[k+1] = \mathrm{prox}_{\gamma f_1}(z[k])$$

$$z[k+1] = z[k] + \mathrm{prox}_{\gamma f_2}(2x[k+1] - z[k]) - x[k+1]$$

Douglas-Rachford splitting algorithm

- General optimization problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ f_1(x) + f_2(x),$$

- f_1 and f_2 are proper, closed, and convex functions.
- f_1 and f_2 are proximable.

Douglas-Rachford splitting algorithm

Initialization: give an initial vector $z[0]$ and a parameter $\gamma > 0$
Iteration: for $k = 0, 1, 2, \ldots$ do

$$\begin{aligned} x[k+1] &= \mathrm{prox}_{\gamma f_1}(z[k]) \\ z[k+1] &= z[k] + \mathrm{prox}_{\gamma f_2}(2x[k+1] - z[k]) - x[k+1] \end{aligned}$$

Douglas-Rachford splitting for ℓ^1 optimization

- ℓ^1 optimization: $\text{minimize}_{\boldsymbol{x}\in\mathbb{R}^n} \;\; \|\boldsymbol{x}\|_1 + I_C(\boldsymbol{x})$
  - $C = \{\boldsymbol{x} \in \mathbb{R}^n : \Phi\boldsymbol{x} = \boldsymbol{y}\}$.
- $f_1(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$ and $f_2(\boldsymbol{x}) = I_C(\boldsymbol{x})$ are proximable.

$$\begin{aligned} \text{prox}_{\gamma f_1}(\boldsymbol{v}) &= S_\gamma(\boldsymbol{v}), \\ \text{prox}_{\gamma f_2}(\boldsymbol{v}) &= \Pi_C(\boldsymbol{v}) = \boldsymbol{v} + \Phi^\top(\Phi\Phi^\top)^{-1}(\boldsymbol{y} - \Phi\boldsymbol{v}). \end{aligned}$$

- Now we have got the algorithm!

Douglas-Rachford splitting algorithm for ℓ^1 optimization

Initialization: give an initial vector $\boldsymbol{z}[0]$ and a parameter $\gamma > 0$
Iteration: for $k = 0, 1, 2, \ldots$ do

$$\begin{aligned} \boldsymbol{x}[k+1] &= S_\gamma(\boldsymbol{z}[k]) \\ \boldsymbol{z}[k+1] &= \boldsymbol{z}[k] + \Pi_C(2\boldsymbol{x}[k+1] - \boldsymbol{z}[k]) - \boldsymbol{x}[k+1] \end{aligned}$$

Douglas-Rachford splitting for ℓ^1 optimization

- ℓ^1 optimization: $\text{minimize}_{x \in \mathbb{R}^n} \ \|x\|_1 + I_C(x)$
  - $C = \{x \in \mathbb{R}^n : \Phi x = y\}$.
- $f_1(x) = \|x\|_1$ and $f_2(x) = I_C(x)$ are proximable.

$$
\begin{aligned}
\mathrm{prox}_{\gamma f_1}(v) &= S_\gamma(v), \\
\mathrm{prox}_{\gamma f_2}(v) &= \Pi_C(v) = v + \Phi^\top(\Phi\Phi^\top)^{-1}(y - \Phi v).
\end{aligned}
$$

- Now we have got the algorithm!

Douglas-Rachford splitting algorithm for ℓ^1 optimization

Initialization: give an initial vector $z[0]$ and a parameter $\gamma > 0$

Iteration: for $k = 0, 1, 2, \ldots$ do

$$
\begin{aligned}
x[k+1] &= S_\gamma(z[k]) \\
z[k+1] &= z[k] + \Pi_C(2x[k+1] - z[k]) - x[k+1]
\end{aligned}
$$

Douglas-Rachford splitting for ℓ^1 optimization

- ℓ^1 optimization: $\text{minimize}_{x \in \mathbb{R}^n} \ \|x\|_1 + I_C(x)$
  - $C = \{x \in \mathbb{R}^n : \Phi x = y\}$.
- $f_1(x) = \|x\|_1$ and $f_2(x) = I_C(x)$ are proximable.

$$\begin{aligned} \text{prox}_{\gamma f_1}(v) &= S_\gamma(v), \\ \text{prox}_{\gamma f_2}(v) &= \Pi_C(v) = v + \Phi^\top(\Phi\Phi^\top)^{-1}(y - \Phi v). \end{aligned}$$

- Now we have got the algorithm!

Douglas-Rachford splitting algorithm for ℓ^1 optimization

Initialization: give an initial vector $z[0]$ and a parameter $\gamma > 0$

Iteration: for $k = 0, 1, 2, \ldots$ do

$$\begin{aligned} x[k+1] &= S_\gamma(z[k]) \\ z[k+1] &= z[k] + \Pi_C(2x[k+1] - z[k]) - x[k+1] \end{aligned}$$

Douglas-Rachford splitting for ℓ^1 optimization

- ℓ^1 optimization: $\text{minimize}_{x \in \mathbb{R}^n} \;\; \|x\|_1 + I_C(x)$
  - $C = \{x \in \mathbb{R}^n : \Phi x = y\}$.
- $f_1(x) = \|x\|_1$ and $f_2(x) = I_C(x)$ are proximable.

$$\mathrm{prox}_{\gamma f_1}(v) = S_\gamma(v),$$
$$\mathrm{prox}_{\gamma f_2}(v) = \Pi_C(v) = v + \Phi^\top(\Phi\Phi^\top)^{-1}(y - \Phi v).$$

- Now we have got the algorithm!

Douglas-Rachford splitting algorithm for ℓ^1 optimization

Initialization: give an initial vector $z[0]$ and a parameter $\gamma > 0$
Iteration: for $k = 0, 1, 2, \ldots$ do

$$x[k+1] = S_\gamma(z[k])$$
$$z[k+1] = z[k] + \Pi_C(2x[k+1] - z[k]) - x[k+1]$$

Douglas-Rachford splitting for ℓ^1 optimization

- ℓ^1 optimization: $\text{minimize}_{x \in \mathbb{R}^n} \;\; \|x\|_1 + I_C(x)$
  - $C = \{x \in \mathbb{R}^n : \Phi x = y\}$.
- $f_1(x) = \|x\|_1$ and $f_2(x) = I_C(x)$ are proximable.

$$
\begin{aligned}
\mathrm{prox}_{\gamma f_1}(v) &= S_\gamma(v),\\
\mathrm{prox}_{\gamma f_2}(v) &= \Pi_C(v) = v + \Phi^\top(\Phi\Phi^\top)^{-1}(y - \Phi v).
\end{aligned}
$$

- Now we have got the algorithm!

Douglas-Rachford splitting algorithm for ℓ^1 optimization

Initialization: give an initial vector $z[0]$ and a parameter $\gamma > 0$
Iteration: for $k = 0, 1, 2, \ldots$ do

$$
\begin{aligned}
x[k+1] &= S_\gamma(z[k])\\
z[k+1] &= z[k] + \Pi_C(2x[k+1] - z[k]) - x[k+1]
\end{aligned}
$$

Douglas-Rachford splitting for ℓ^1 optimization

- ℓ^1 optimization: $\text{minimize}_{x \in \mathbb{R}^n} \ \|x\|_1 + I_C(x)$
  - $C = \{x \in \mathbb{R}^n : \Phi x = y\}$.
- $f_1(x) = \|x\|_1$ and $f_2(x) = I_C(x)$ are proximable.

$$\begin{aligned}
\text{prox}_{\gamma f_1}(v) &= S_\gamma(v), \\
\text{prox}_{\gamma f_2}(v) &= \Pi_C(v) = v + \Phi^\top(\Phi\Phi^\top)^{-1}(y - \Phi v).
\end{aligned}$$

- Now we have got the algorithm!

Douglas-Rachford splitting algorithm for ℓ^1 optimization

Initialization: give an initial vector $z[0]$ and a parameter $\gamma > 0$
Iteration: for $k = 0, 1, 2, \ldots$ do

$$\begin{aligned}
x[k+1] &= S_\gamma(z[k]) \\
z[k+1] &= z[k] + \Pi_C(2x[k+1] - z[k]) - x[k+1]
\end{aligned}$$

Douglas-Rachford splitting algorithm

Douglas-Rachford splitting algorithm for ℓ^1 optimization

Initialization: give an initial vector $z[0]$ and a parameter $\gamma > 0$

Iteration: for $k = 0, 1, 2, \ldots$ do

$$
\begin{aligned}
x[k+1] &= S_\gamma(z[k]) \\
z[k+1] &= z[k] + \Pi_C(2x[k+1] - z[k]) - x[k+1]
\end{aligned}
$$

- Douglas-Rachford algorithm only requires
  - simple continuous mapping of the soft-thresholding function S_γ
  - linear computation of matrix-vector multiplication and vector addition
- Much faster and easier to implement than the standard interior-point method
  - The standard interior-point method requires to solve linear equations at each step

Douglas-Rachford splitting algorithm

Douglas-Rachford splitting algorithm for ℓ^1 optimization

Initialization: give an initial vector $z[0]$ and a parameter $\gamma > 0$
Iteration: for $k = 0, 1, 2, \ldots$ do

$$
\begin{aligned}
x[k+1] &= S_\gamma(z[k]) \\
z[k+1] &= z[k] + \Pi_C(2x[k+1] - z[k]) - x[k+1]
\end{aligned}
$$

- Douglas-Rachford algorithm only requires
    - simple continuous mapping of the soft-thresholding function S_γ
    - linear computation of matrix-vector multiplication and vector addition
- Much faster and easier to implement than the standard interior-point method
    - The standard interior-point method requires to solve linear equations at each step.

Douglas-Rachford splitting algorithm

Douglas-Rachford splitting algorithm for ℓ^1 optimization

Initialization: give an initial vector $z[0]$ and a parameter $\gamma > 0$

Iteration: for $k = 0, 1, 2, \ldots$ do

$$\begin{aligned} x[k+1] &= S_\gamma(z[k]) \\ z[k+1] &= z[k] + \Pi_C(2x[k+1] - z[k]) - x[k+1] \end{aligned}$$

- Douglas-Rachford algorithm only requires
  - simple continuous mapping of the soft-thresholding function S_γ
  - linear computation of matrix-vector multiplication and vector addition
- Much faster and easier to implement than the standard interior-point method
  - The standard interior-point method requires to solve linear equations at each step

Douglas-Rachford splitting algorithm

Douglas-Rachford splitting algorithm for ℓ^1 optimization

Initialization: give an initial vector $z[0]$ and a parameter $\gamma > 0$

Iteration: for $k = 0, 1, 2, \ldots$ do

$$\begin{aligned} x[k+1] &= S_\gamma(z[k]) \\ z[k+1] &= z[k] + \Pi_C(2x[k+1] - z[k]) - x[k+1] \end{aligned}$$

- Douglas-Rachford algorithm only requires
  - simple continuous mapping of the soft-thresholding function S_γ
  - linear computation of matrix-vector multiplication and vector addition
- Much faster and easier to implement than the standard interior-point method
  - The standard interior-point method requires to solve linear equations at each step.

Douglas-Rachford splitting algorithm

Douglas-Rachford splitting algorithm for ℓ^1 optimization

Initialization: give an initial vector $z[0]$ and a parameter $\gamma > 0$
Iteration: for $k = 0, 1, 2, \ldots$ do

$$x[k+1] = S_\gamma(z[k])$$
$$z[k+1] = z[k] + \Pi_C(2x[k+1] - z[k]) - x[k+1]$$

- Douglas-Rachford algorithm only requires
  - simple continuous mapping of the soft-thresholding function S_γ
  - linear computation of matrix-vector multiplication and vector addition
- Much faster and easier to implement than the standard interior-point method
  - The standard interior-point method requires to solve linear equations at each step.

Douglas-Rachford splitting algorithm

Douglas-Rachford splitting algorithm for ℓ^1 optimization

Initialization: give an initial vector $z[0]$ and a parameter $\gamma > 0$
Iteration: for $k = 0, 1, 2, \ldots$ do

$$
\begin{aligned}
x[k+1] &= S_\gamma(z[k]) \\
z[k+1] &= z[k] + \Pi_C(2x[k+1] - z[k]) - x[k+1]
\end{aligned}
$$

- Douglas-Rachford algorithm only requires
  - simple continuous mapping of the soft-thresholding function S_γ
  - linear computation of matrix-vector multiplication and vector addition
- Much faster and easier to implement than the standard interior-point method
  - The standard interior-point method requires to solve linear equations at each step.

Table of Contents

ℓ^1 regularization (LASSO)

ℓ^1 regularization (LASSO)

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ \frac{1}{2}\|\Phi x - y\|_2^2 + \lambda\|x\|_1.$$

- Sum of two convex functions $f = f_1 + f_2$:

$$f_1(x) = \frac{1}{2}\|\Phi x - y\|_2^2, \quad f_2(x) = \lambda\|x\|_1$$

- f_1 and f_2 are both proximable $\rightarrow$ Douglas-Rachford splitting
- f_1 is also differentiable
- A yet faster algorithm exists for this type of optimization problem.

ℓ^1 regularization (LASSO)

ℓ^1 regularization (LASSO)

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \frac{1}{2}\|\Phi x - y\|_2^2 + \lambda \|x\|_1.$$

- Sum of two convex functions $f = f_1 + f_2$:

$$f_1(x) = \frac{1}{2}\|\Phi x - y\|_2^2, \quad f_2(x) = \lambda \|x\|_1$$

- f_1 and f_2 are both proximable $\longrightarrow$ Douglas-Rachford splitting
- f_1 is also differentiable
- A yet faster algorithm exists for this type of optimization problem.

ℓ^1 regularization (LASSO)

ℓ^1 regularization (LASSO)

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \frac{1}{2}\|\Phi x - y\|_2^2 + \lambda \|x\|_1.$$

- Sum of two convex functions $f = f_1 + f_2$:

$$f_1(x) = \frac{1}{2}\|\Phi x - y\|_2^2, \quad f_2(x) = \lambda \|x\|_1$$

- f_1 and f_2 are both proximable $\longrightarrow$ Douglas-Rachford splitting
- f_1 is also differentiable
- A yet faster algorithm exists for this type of optimization problem.

ℓ^1 regularization (LASSO)

ℓ^1 regularization (LASSO)

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \frac{1}{2}\|\Phi x - y\|_2^2 + \lambda \|x\|_1.$$

- Sum of two convex functions $f = f_1 + f_2$:

$$f_1(x) = \frac{1}{2}\|\Phi x - y\|_2^2, \quad f_2(x) = \lambda \|x\|_1$$

- f_1 and f_2 are both proximable $\longrightarrow$ Douglas-Rachford splitting
- f_1 is also differentiable
- A yet faster algorithm exists for this type of optimization problem.

ℓ^1 regularization (LASSO)

ℓ^1 regularization (LASSO)

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \frac{1}{2}\|\Phi x - y\|_2^2 + \lambda \|x\|_1.$$

- Sum of two convex functions $f = f_1 + f_2$:

$$f_1(x) = \frac{1}{2}\|\Phi x - y\|_2^2, \quad f_2(x) = \lambda \|x\|_1$$

- f_1 and f_2 are both proximable $\longrightarrow$ Douglas-Rachford splitting
- f_1 is also differentiable
- A yet faster algorithm exists for this type of optimization problem.

Proximal gradient algorithm

- Optimization problem:

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \ f_1(x) + f_2(x),$$

- f_1 is differentiable and convex, satisfying $\mathrm{dom}(f_1) = \mathbb{R}^n$
- f_2 is a proper, closed, and convex function.

Proximal gradient algorithm

Initialization: give an initial vector $x[0]$ and a real number $\gamma > 0$
Iteration: for $k = 0, 1, 2, \ldots$ do

$$x[k+1] = \mathrm{prox}_{\gamma f_2}\big(x[k] - \gamma \nabla f_1(x[k])\big).$$

Proximal gradient algorithm

- Optimization problem:

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}}\ f_1(\boldsymbol{x}) + f_2(\boldsymbol{x}),$$

- f_1 is differentiable and convex, satisfying $\mathrm{dom}(f_1) = \mathbb{R}^n$
- f_2 is a proper, closed, and convex function.

Proximal gradient algorithm

Initialization: give an initial vector $\boldsymbol{x}[0]$ and a real number $\gamma > 0$
Iteration: for $k = 0, 1, 2, \ldots$ do

$$\boldsymbol{x}[k+1] = \mathrm{prox}_{\gamma f_2}\big(\boldsymbol{x}[k] - \gamma \nabla f_1(\boldsymbol{x}[k])\big).$$

Proximal gradient algorithm

- Optimization problem:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \; f_1(x) + f_2(x),$$

- f_1 is differentiable and convex, satisfying $\mathrm{dom}(f_1) = \mathbb{R}^n$
- f_2 is a proper, closed, and convex function.

Proximal gradient algorithm

Initialization: give an initial vector $x[0]$ and a real number $\gamma > 0$

Iteration: for $k = 0, 1, 2, \ldots$ do

$$x[k+1] = \mathrm{prox}_{\gamma f_2}\big(x[k] - \gamma \nabla f_1(x[k])\big).$$

Proximal gradient algorithm

- Optimization problem:

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \;\; f_1(\boldsymbol{x}) + f_2(\boldsymbol{x}),$$

- f_1 is differentiable and convex, satisfying $\text{dom}(f_1) = \mathbb{R}^n$
- f_2 is a proper, closed, and convex function.

Proximal gradient algorithm

Initialization: give an initial vector $\boldsymbol{x}[0]$ and a real number $\gamma > 0$
Iteration: for $k = 0, 1, 2, \ldots$ do

$$\boldsymbol{x}[k+1] = \text{prox}_{\gamma f_2}\big(\boldsymbol{x}[k] - \gamma \nabla f_1(\boldsymbol{x}[k])\big).$$

Geometrical interpretation

- Proximal gradient algorithm

$$x[k+1] = \underbrace{\mathrm{prox}_{\gamma f_2}\bigl(x[k] - \gamma \nabla f_1(x[k])\bigr)}_{\triangleq \phi(x[k])}.$$

- The function $\phi(x)$ is rewritten as

$$\begin{aligned}\phi(x) &= \arg\min_{z\in\mathbb{R}^n}\Bigl\{f_2(z) + \frac{1}{2\gamma}\bigl\|z - (x - \gamma\nabla f_1(x))\bigr\|_2^2\Bigr\}\\ &= \arg\min_{z\in\mathbb{R}^n}\Bigl\{\underbrace{f_1(x) + \nabla f_1(x)^\top (z - x)}_{\triangleq \tilde{f}_1(z;x)} + f_2(z) + \frac{1}{2\gamma}\|z - x\|_2^2\Bigr\}.\end{aligned}$$

Geometrical interpretation

- Proximal gradient algorithm

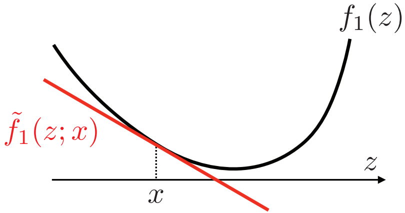
$$x[k+1] = \underbrace{\mathrm{prox}_{\gamma f_2}\bigl(x[k] - \gamma \nabla f_1(x[k])\bigr)}_{\triangleq \phi(x[k])}.$$

- The function $\phi(x)$ is rewritten as

$$\begin{aligned}\phi(x) &= \arg\min_{z \in \mathbb{R}^n} \Bigl\{ f_2(z) + \frac{1}{2\gamma} \bigl\| z - \bigl(x - \gamma \nabla f_1(x)\bigr) \bigr\|_2^2 \Bigr\} \\ &= \arg\min_{z \in \mathbb{R}^n} \Bigl\{ \underbrace{f_1(x) + \nabla f_1(x)^\top (z - x)}_{\triangleq \tilde{f}_1(z;x)} + f_2(z) + \frac{1}{2\gamma} \|z - x\|_2^2 \Bigr\}.\end{aligned}$$

Geometrical interpretation

- The function $\tilde{f}_1(\boldsymbol{z};\boldsymbol{x})$ is a linear approximation of $f_1(\boldsymbol{z})$ around the point $\boldsymbol{x} \in \mathbb{R}^n$.



- The iteration becomes

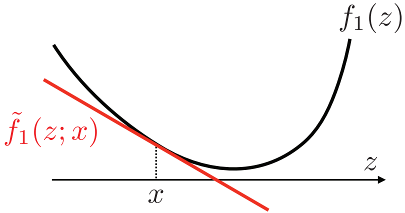
$$\begin{aligned}\boldsymbol{x}[k+1] &= \arg\min_{\boldsymbol{z}\in\mathbb{R}^n}\left\{\tilde{f}_1(\boldsymbol{z};\boldsymbol{x}) + f_2(\boldsymbol{z}) + \frac{1}{2\gamma}\|\boldsymbol{z}-\boldsymbol{x}\|_2^2\right\}\\ &= \mathrm{prox}_{\gamma\tilde{f}}(\boldsymbol{x}[k])\end{aligned}$$

where $\tilde{f}(\boldsymbol{z}) = \tilde{f}_1(\boldsymbol{z};\boldsymbol{x}) + f_2(\boldsymbol{z})$.

- This is a proximal algorithm for finding the minimizer of $\tilde{f}$.

Geometrical interpretation

- The function $\tilde{f}_1(\boldsymbol{z};\boldsymbol{x})$ is a linear approximation of $f_1(\boldsymbol{z})$ around the point $\boldsymbol{x} \in \mathbb{R}^n$.



- The iteration becomes

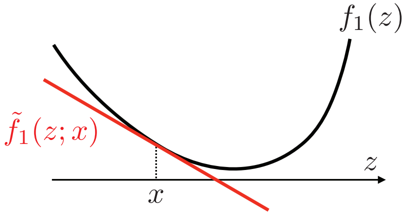
$$
\begin{aligned}
\boldsymbol{x}[k+1] &= \underset{\boldsymbol{z}\in\mathbb{R}^n}{\arg\min}\left\{\tilde{f}_1(\boldsymbol{z};\boldsymbol{x}) + f_2(\boldsymbol{z}) + \frac{1}{2\gamma}\|\boldsymbol{z}-\boldsymbol{x}\|_2^2\right\} \\
&= \mathrm{prox}_{\gamma\tilde{f}}(\boldsymbol{x}[k])
\end{aligned}
$$

where $\tilde{f}(\boldsymbol{z}) = \tilde{f}_1(\boldsymbol{z};\boldsymbol{x}) + f_2(\boldsymbol{z})$.

- This is a proximal algorithm for finding the minimizer of $\tilde{f}$.

Geometrical interpretation

- The function $\tilde{f}_1(\boldsymbol{z};\boldsymbol{x})$ is a linear approximation of $f_1(\boldsymbol{z})$ around the point $\boldsymbol{x} \in \mathbb{R}^n$.



- The iteration becomes

$$\begin{aligned}\boldsymbol{x}[k+1] &= \underset{\boldsymbol{z}\in\mathbb{R}^n}{\arg\min}\left\{\tilde{f}_1(\boldsymbol{z};\boldsymbol{x}) + f_2(\boldsymbol{z}) + \frac{1}{2\gamma}\|\boldsymbol{z}-\boldsymbol{x}\|_2^2\right\} \\ &= \operatorname{prox}_{\gamma\tilde{f}}(\boldsymbol{x}[k])\end{aligned}$$

where $\tilde{f}(\boldsymbol{z}) = \tilde{f}_1(\boldsymbol{z};\boldsymbol{x}) + f_2(\boldsymbol{z})$.

- This is a proximal algorithm for finding the minimizer of $\tilde{f}$.

Convergence analysis

Theorem

Assume the gradient ∇f_1 is Lipschitz continuous over $\mathbb{R}^n$ with Lipschitz constant L. Assume also that the step size γ satisfies

$$\gamma \leq \frac{1}{L}.$$

Then the sequence $\{x[k]\}$ generated by the proximal gradient algorithm converges to an optimal solution x^* at the rate of $O(1/k)$.

- The gradient ∇f_1 is Lipschitz continuous over $\mathbb{R}^n$ with Lipschitz constant L if

$$\|\nabla f_1(x) - \nabla f_1(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y \in \mathbb{R}^n.$$

Convergence analysis

Theorem

Assume the gradient ∇f_1 is Lipschitz continuous over $\mathbb{R}^n$ with Lipschitz constant L. Assume also that the step size γ satisfies

$$\gamma \leq \frac{1}{L}.$$

Then the sequence $\{x[k]\}$ generated by the proximal gradient algorithm converges to an optimal solution x^* at the rate of $O(1/k)$.

- The gradient ∇f_1 is Lipschitz continuous over $\mathbb{R}^n$ with Lipschitz constant L if

$$\|\nabla f_1(x) - \nabla f_1(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y \in \mathbb{R}^n.$$

Convergence analysis

Theorem

Assume the gradient ∇f_1 is Lipschitz continuous over $\mathbb{R}^n$ with Lipschitz constant L. Assume also that the step size γ satisfies

$$\gamma \leq \frac{1}{L}.$$

Then the sequence $\{x[k]\}$ generated by the proximal gradient algorithm converges to an optimal solution x^* at the rate of $O(1/k)$.

- The gradient ∇f_1 is Lipschitz continuous over $\mathbb{R}^n$ with Lipschitz constant L if

$$\|\nabla f_1(x) - \nabla f_1(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y \in \mathbb{R}^n.$$

Convergence analysis

Theorem

Assume the gradient ∇f_1 is Lipschitz continuous over $\mathbb{R}^n$ with Lipschitz constant L. Assume also that the step size γ satisfies

$$\gamma \leq \frac{1}{L}.$$

Then the sequence $\{x[k]\}$ generated by the proximal gradient algorithm converges to an optimal solution $\boldsymbol{x}^*$ at the rate of $O(1/k)$.

- The gradient ∇f_1 is Lipschitz continuous over $\mathbb{R}^n$ with Lipschitz constant L if

$$\|\nabla f_1(\boldsymbol{x}) - \nabla f_1(\boldsymbol{y})\|_2 \leq L\|\boldsymbol{x} - \boldsymbol{y}\|_2, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n.$$

ℓ^1 regularization (LASSO)

- ℓ^1 regularization problem (LASSO)

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

- Sum of two convex functions $f = f_1 + f_2$:

$$f_1(\boldsymbol{x}) = \frac{1}{2}\|\Phi\boldsymbol{x}-\boldsymbol{y}\|_2^2, \quad f_2(\boldsymbol{x}) = \lambda\|\boldsymbol{x}\|_1$$

with

$$\nabla f_1(\boldsymbol{x}) = \Phi^\top(\Phi\boldsymbol{x}-\boldsymbol{y}), \quad \mathrm{prox}_{\gamma f_2}(\boldsymbol{v}) = S_{\gamma\lambda}(\boldsymbol{v}).$$

ISTA (Iterative Shrinkage Thresholding Algorithm)

Initialization: give an initial vector $\boldsymbol{x}[0]$ and parameter $\gamma > 0$
Iteration: for $k = 0, 1, 2, \ldots$ do

$$\boldsymbol{x}[k+1] = S_{\gamma\lambda}\big(\boldsymbol{x}[k]-\gamma\Phi^\top(\Phi\boldsymbol{x}[k]-\boldsymbol{y})\big).$$

ℓ^1 regularization (LASSO)

- ℓ^1 regularization problem (LASSO)

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

- Sum of two convex functions $f = f_1 + f_2$:

$$f_1(\boldsymbol{x}) = \frac{1}{2}\|\Phi\boldsymbol{x}-\boldsymbol{y}\|_2^2, \quad f_2(\boldsymbol{x}) = \lambda\|\boldsymbol{x}\|_1$$

with

$$\nabla f_1(x) = \Phi^\top(\Phi x - y), \quad \mathrm{prox}_{\gamma f_2}(v) = S_{\gamma\lambda}(v).$$

ISTA (Iterative Shrinkage Thresholding Algorithm)

Initialization: give an initial vector $x[0]$ and parameter $\gamma > 0$

Iteration: for $k = 0, 1, 2, \ldots$ do

$$x[k+1] = S_{\gamma\lambda}\big(x[k] - \gamma\Phi^\top(\Phi x[k] - y)\big).$$

ℓ^1 regularization (LASSO)

- ℓ^1 regularization problem (LASSO)

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \frac{1}{2}\|\Phi x - y\|_2^2 + \lambda \|x\|_1.$$

- Sum of two convex functions $f = f_1 + f_2$:

$$f_1(x) = \frac{1}{2}\|\Phi x - y\|_2^2, \quad f_2(x) = \lambda \|x\|_1$$

with

$$\nabla f_1(x) = \Phi^\top(\Phi x - y), \quad \mathrm{prox}_{\gamma f_2}(v) = S_{\gamma\lambda}(v).$$

ISTA (Iterative Shrinkage Thresholding Algorithm)

Initialization: give an initial vector $x[0]$ and parameter $\gamma > 0$
Iteration: for $k = 0, 1, 2, \ldots$ do

$$x[k+1] = S_{\gamma\lambda}\big(x[k] - \gamma\Phi^\top(\Phi x[k] - y)\big).$$

ℓ^1 regularization (LASSO)

- ℓ^1 regularization problem (LASSO)

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \frac{1}{2}\|\Phi x - y\|_2^2 + \lambda \|x\|_1.$$

- Sum of two convex functions $f = f_1 + f_2$:

$$f_1(x) = \frac{1}{2}\|\Phi x - y\|_2^2, \quad f_2(x) = \lambda \|x\|_1$$

with

$$\nabla f_1(x) = \Phi^\top(\Phi x - y), \quad \text{prox}_{\gamma f_2}(v) = S_{\gamma\lambda}(v).$$

ISTA (Iterative Shrinkage Thresholding Algorithm)

Initialization: give an initial vector $x[0]$ and parameter $\gamma > 0$
Iteration: for $k = 0, 1, 2, \ldots$ do

$$x[k+1] = S_{\gamma\lambda}\big(x[k] - \gamma \Phi^\top(\Phi x[k] - y)\big).$$

Convergence of ISTA

- Gradient

$$\nabla f_1(x) = \Phi^\top(\Phi x - y).$$

- Lipschitz constant L of ∇f_1:

$$\|\nabla f_1(x_1) - \nabla f_1(x_2)\|_2 = \|\Phi^\top\Phi(x_1 - x_2)\|_2 \leq \lambda_{\max}(\Phi^\top\Phi)\|x_1 - x_2\|_2$$

where $\lambda_{\max}(\Phi^\top\Phi)$ is the largest eigenvalue of $\Phi^\top\Phi$.

- We have $L = \lambda_{\max}(\Phi^\top\Phi) = \sigma_{\max}(\Phi)^2 = \|\Phi\|^2$, where $\sigma_{\max}(\Phi)$ is the largest singular value of Φ and $\|\Phi\|$ is a matrix norm.
- A sufficient condition for convergence:

$$\gamma \leq \frac{1}{L} = \frac{1}{\|\Phi\|^2}$$

- The convergence rate is $O(1/k)$.

Convergence of ISTA

- Gradient
$$\nabla f_1(\boldsymbol{x}) = \Phi^\top(\Phi\boldsymbol{x} - \boldsymbol{y}).$$
- Lipschitz constant L of ∇f_1:
$$\|\nabla f_1(\boldsymbol{x}_1) - \nabla f_1(\boldsymbol{x}_2)\|_2 = \|\Phi^\top\Phi(\boldsymbol{x}_1 - \boldsymbol{x}_2)\|_2 \leq \lambda_{\max}(\Phi^\top\Phi)\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2$$
where $\lambda_{\max}(\Phi^\top\Phi)$ is the largest eigenvalue of $\Phi^\top\Phi$.
- We have $L = \lambda_{\max}(\Phi^\top\Phi) = \sigma_{\max}(\Phi)^2 = \|\Phi\|^2$, where $\sigma_{\max}(\Phi)$ is the largest singular value of Φ and $\|\Phi\|$ is a matrix norm.
- A sufficient condition for convergence:
$$\gamma \leq \frac{1}{L} = \frac{1}{\|\Phi\|^2}$$
- The convergence rate is $O(1/k)$.

Convergence of ISTA

- Gradient
$$\nabla f_1(\boldsymbol{x}) = \Phi^\top(\Phi\boldsymbol{x} - \boldsymbol{y}).$$
- Lipschitz constant L of ∇f_1:
$$\|\nabla f_1(\boldsymbol{x}_1) - \nabla f_1(\boldsymbol{x}_2)\|_2 = \|\Phi^\top\Phi(\boldsymbol{x}_1 - \boldsymbol{x}_2)\|_2 \leq \lambda_{\max}(\Phi^\top\Phi)\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2$$
where $\lambda_{\max}(\Phi^\top\Phi)$ is the largest eigenvalue of $\Phi^\top\Phi$.
- We have $L = \lambda_{\max}(\Phi^\top\Phi) = \sigma_{\max}(\Phi)^2 = \|\Phi\|^2$, where $\sigma_{\max}(\Phi)$ is the largest singular value of Φ and $\|\Phi\|$ is a matrix norm.
- A sufficient condition for convergence:
$$\gamma \leq \frac{1}{L} = \frac{1}{\|\Phi\|^2}$$
- The convergence rate is $O(1/k)$.

Convergence of ISTA

- Gradient

$$\nabla f_1(\boldsymbol{x}) = \Phi^\top(\Phi\boldsymbol{x} - \boldsymbol{y}).$$

- Lipschitz constant L of ∇f_1:

$$\|\nabla f_1(\boldsymbol{x}_1) - \nabla f_1(\boldsymbol{x}_2)\|_2 = \|\Phi^\top\Phi(\boldsymbol{x}_1 - \boldsymbol{x}_2)\|_2 \leq \lambda_{\max}(\Phi^\top\Phi)\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2$$

  where $\lambda_{\max}(\Phi^\top\Phi)$ is the largest eigenvalue of $\Phi^\top\Phi$.
- We have $L = \lambda_{\max}(\Phi^\top\Phi) = \sigma_{\max}(\Phi)^2 = \|\Phi\|^2$, where $\sigma_{\max}(\Phi)$ is the largest singular value of Φ and $\|\Phi\|$ is a matrix norm.
- A sufficient condition for convergence:

$$\gamma \leq \frac{1}{L} = \frac{1}{\|\Phi\|^2}$$

- The convergence rate is $O(1/k)$.

Convergence of ISTA

- Gradient
$$\nabla f_1(\boldsymbol{x}) = \Phi^\top(\Phi\boldsymbol{x} - \boldsymbol{y}).$$
- Lipschitz constant L of ∇f_1:
$$\|\nabla f_1(\boldsymbol{x}_1) - \nabla f_1(\boldsymbol{x}_2)\|_2 = \|\Phi^\top\Phi(\boldsymbol{x}_1 - \boldsymbol{x}_2)\|_2 \leq \lambda_{\max}(\Phi^\top\Phi)\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2$$
where $\lambda_{\max}(\Phi^\top\Phi)$ is the largest eigenvalue of $\Phi^\top\Phi$.
- We have $L = \lambda_{\max}(\Phi^\top\Phi) = \sigma_{\max}(\Phi)^2 = \|\Phi\|^2$, where $\sigma_{\max}(\Phi)$ is the largest singular value of Φ and $\|\Phi\|$ is a matrix norm.
- A sufficient condition for convergence:
$$\gamma \leq \frac{1}{L} = \frac{1}{\|\Phi\|^2}$$
- The convergence rate is $O(1/k)$.

Fast ISTA (FISTA)

- Accelerated algorithm of ISTA = FISTA (Fast ISTA)
- The convergence rate is $O(1/k^2)$.

FISTA

Initialization: give initial vectors $x[0]$, $z[0]$, initial number $t[0]$, and parameter $\gamma > 0$
Iteration: for $k = 0, 1, 2, \ldots$ do

$$x[k+1] = S_{\gamma\lambda}\left(z[k] - \gamma\Phi^\top(\Phi z[k] - y)\right),$$

$$t[k+1] = \frac{1 + \sqrt{1 + 4t[k]^2}}{2},$$

$$z[k+1] = x[k+1] + \frac{t[k] - 1}{t[k+1]}(x[k+1] - x[k]).$$

Fast ISTA (FISTA)

- Accelerated algorithm of ISTA = FISTA (Fast ISTA)
- The convergence rate is $O(1/k^2)$.

FISTA

Initialization: give initial vectors $x[0]$, $z[0]$, initial number $t[0]$, and parameter $\gamma > 0$
Iteration: for $k = 0, 1, 2, \ldots$ do

$$x[k+1] = S_{\gamma\lambda}\left(z[k] - \gamma\Phi^\top(\Phi z[k] - y)\right),$$

$$t[k+1] = \frac{1 + \sqrt{1 + 4t[k]^2}}{2},$$

$$z[k+1] = x[k+1] + \frac{t[k] - 1}{t[k+1]}(x[k+1] - x[k]).$$

Fast ISTA (FISTA)

- Accelerated algorithm of ISTA = FISTA (Fast ISTA)
- The convergence rate is $O(1/k^2)$.

FISTA

Initialization: give initial vectors $\boldsymbol{x}[0]$, $\boldsymbol{z}[0]$, initial number $t[0]$, and parameter $\gamma > 0$

Iteration: for $k = 0, 1, 2, \ldots$ do

$$
\begin{aligned}
\boldsymbol{x}[k+1] &= S_{\gamma\lambda}\left(\boldsymbol{z}[k] - \gamma\Phi^\top(\Phi\boldsymbol{z}[k] - \boldsymbol{y})\right),\\
t[k+1] &= \frac{1 + \sqrt{1 + 4t[k]^2}}{2},\\
\boldsymbol{z}[k+1] &= \boldsymbol{x}[k+1] + \frac{t[k]-1}{t[k+1]}(\boldsymbol{x}[k+1] - \boldsymbol{x}[k]).
\end{aligned}
$$

Table of Contents

Generalized LASSO and ADMM

- A generalized regularization problem:

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}}\ \frac{1}{2}\|\Phi\boldsymbol{x}-\boldsymbol{y}\|_2^2+\lambda\|\Psi\boldsymbol{x}\|_1,$$

- Ψ is a matrix.
- We call this the generalized LASSO.
- If $\Psi = I$, then this is LASSO.
- $\|\Psi \boldsymbol{x}\|_1$ is not proximable in general.
    - No closed-form expression for the proximal operator.
- We need yet another algorithm.

Generalized LASSO and ADMM

- A generalized regularization problem:

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \ \frac{1}{2}\|\Phi\boldsymbol{x}-\boldsymbol{y}\|_2^2 + \lambda\|\Psi\boldsymbol{x}\|_1,$$

- Ψ is a matrix.
- We call this the generalized LASSO.
- If $\Psi = I$, then this is LASSO.
- $\|\Psi x\|_1$ is not proximable in general.
  - No closed-form expression for its proximal operator.
- We need yet another algorithm.

Generalized LASSO and ADMM

- A generalized regularization problem:

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \ \frac{1}{2}\|\Phi\boldsymbol{x}-\boldsymbol{y}\|_2^2+\lambda\|\Psi\boldsymbol{x}\|_1,$$

- Ψ is a matrix.
- We call this the generalized LASSO.
- If $\Psi = I$, then this is LASSO.
- $\|\Psi\boldsymbol{x}\|_1$ is not proximable in general.
    - No closed-form expression for its proximal operator.
- We need yet another algorithm.

Generalized LASSO and ADMM

- A generalized regularization problem:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \frac{1}{2}\|\Phi x - y\|_2^2 + \lambda\|\Psi x\|_1,$$

- Ψ is a matrix.
- We call this the generalized LASSO.
- If $\Psi = I$, then this is LASSO.
- $\|\Psi x\|_1$ is not proximable in general.
  - No closed-form expression for its proximal operator.
- We need yet another algorithm.

Generalized LASSO and ADMM

- A generalized regularization problem:

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \ \frac{1}{2}\|\Phi\boldsymbol{x}-\boldsymbol{y}\|_2^2+\lambda\|\Psi\boldsymbol{x}\|_1,$$

- Ψ is a matrix.
- We call this the generalized LASSO.
- If $\Psi = I$, then this is LASSO.
- $\|\Psi\boldsymbol{x}\|_1$ is not proximable in general.
  - No closed-form expression for its proximal operator.
- We need yet another algorithm.

Generalized LASSO and ADMM

- A generalized regularization problem:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \frac{1}{2}\|\Phi x - y\|_2^2 + \lambda \|\Psi x\|_1,$$

- Ψ is a matrix.
- We call this the generalized LASSO.
- If $\Psi = I$, then this is LASSO.
- $\|\Psi x\|_1$ is not proximable in general.
  - No closed-form expression for its proximal operator.
- We need yet another algorithm.

Generalized LASSO and ADMM

- A generalized regularization problem:

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \ \frac{1}{2}\|\Phi\boldsymbol{x}-\boldsymbol{y}\|_2^2+\lambda\|\Psi\boldsymbol{x}\|_1,$$

- Ψ is a matrix.
- We call this the generalized LASSO.
- If $\Psi = I$, then this is LASSO.
- $\|\Psi\boldsymbol{x}\|_1$ is not proximable in general.
    - No closed-form expression for its proximal operator.
- We need yet another algorithm.

ADMM

- Optimization problem

$$\underset{x \in \mathbb{R}^n, z \in \mathbb{R}^p}{\text{minimize}} \; f_1(x) + f_2(z) \;\; \text{subject to} \;\; z = \Psi x,$$

- f_1, f_2: proper, closed, and convex
- $\Psi \in \mathbb{R}^{p \times n}$
- Alternating Direction Method of Multipliers (ADMM)

ADMM

Initialization: give initial vectors $z[0], v[0] \in \mathbb{R}^p$, and real number $\gamma > 0$

Iteration: for $k = 0, 1, 2, \ldots$ do

$$x[k+1] := \underset{x \in \mathbb{R}^n}{\arg\min} \Big\{ f_1(x) + \frac{1}{2\gamma} \big\| \Psi x - z[k] + v[k] \big\|^2 \Big\},$$

$$z[k+1] := \mathrm{prox}_{\gamma f_2} \big(\Psi x[k+1] + v[k] \big),$$

$$v[k+1] := v[k] + \Psi x[k+1] - z[k+1].$$

ADMM

- Optimization problem

$$\underset{x\in\mathbb{R}^n, z\in\mathbb{R}^p}{\text{minimize}} \; f_1(x) + f_2(z) \;\; \text{subject to} \;\; z = \Psi x,$$

- f_1, f_2: proper, closed, and convex
- $\Psi \in \mathbb{R}^{p\times n}$
- Alternating Direction Method of Multipliers (ADMM)

ADMM

Initialization: give initial vectors $z[0], v[0] \in \mathbb{R}^p$, and real number $\gamma > 0$

Iteration: for $k = 0, 1, 2, \ldots$ do

$$x[k+1] := \arg\min_{x\in\mathbb{R}^n} \Big\{ f_1(x) + \frac{1}{2\gamma} \big\| \Psi x - z[k] + v[k] \big\|^2 \Big\},$$

$$z[k+1] := \operatorname{prox}_{\gamma f_2}\big(\Psi x[k+1] + v[k]\big),$$

$$v[k+1] := v[k] + \Psi x[k+1] - z[k+1].$$

ADMM

- Optimization problem

$$\underset{x\in\mathbb{R}^n, z\in\mathbb{R}^p}{\text{minimize}} \; f_1(x) + f_2(z) \; \text{ subject to } \; z = \Psi x,$$

- f_1, f_2: proper, closed, and convex
- $\Psi \in \mathbb{R}^{p\times n}$
- Alternating Direction Method of Multipliers (ADMM)

ADMM

Initialization: give initial vectors $z[0], v[0] \in \mathbb{R}^p$, and real number $\gamma > 0$

Iteration: for $k = 0, 1, 2, \dots$ do

$$x[k+1] := \underset{x\in\mathbb{R}^n}{\arg\min}\Big\{f_1(x) + \frac{1}{2\gamma}\big\|\Psi x - z[k] + v[k]\big\|^2\Big\},$$

$$z[k+1] := \mathrm{prox}_{\gamma f_2}\big(\Psi x[k+1] + v[k]\big),$$

$$v[k+1] := v[k] + \Psi x[k+1] - z[k+1].$$

ADMM

- Optimization problem

$$\underset{x \in \mathbb{R}^n, z \in \mathbb{R}^p}{\text{minimize}} \ f_1(x) + f_2(z) \ \text{subject to} \ z = \Psi x,$$

- f_1, f_2: proper, closed, and convex
- $\Psi \in \mathbb{R}^{p \times n}$
- Alternating Direction Method of Multipliers (ADMM)

ADMM

Initialization: give initial vectors $z[0], v[0] \in \mathbb{R}^p$, and real number $\gamma > 0$

Iteration: for $k = 0, 1, 2, \ldots$ do

$$\begin{aligned} x[k+1] &:= \arg\min_{x \in \mathbb{R}^n} \left\{ f_1(x) + \frac{1}{2\gamma} \left\| \Psi x - z[k] + v[k] \right\|^2 \right\}, \\ z[k+1] &:= \operatorname{prox}_{\gamma f_2}\left(\Psi x[k+1] + v[k] \right), \\ v[k+1] &:= v[k] + \Psi x[k+1] - z[k+1]. \end{aligned}$$

ADMM

- Optimization problem

$$\underset{x \in \mathbb{R}^n, z \in \mathbb{R}^p}{\text{minimize}} \ f_1(x) + f_2(z) \ \text{ subject to } \ z = \Psi x,$$

- f_1, f_2: proper, closed, and convex
- $\Psi \in \mathbb{R}^{p \times n}$
- Alternating Direction Method of Multipliers (ADMM)

ADMM

Initialization: give initial vectors $z[0], v[0] \in \mathbb{R}^p$, and real number $\gamma > 0$

Iteration: for $k = 0, 1, 2, \ldots$ do

$$x[k+1] := \underset{x \in \mathbb{R}^n}{\arg\min} \left\{ f_1(x) + \frac{1}{2\gamma} \left\| \Psi x - z[k] + v[k] \right\|^2 \right\},$$

$$z[k+1] := \text{prox}_{\gamma f_2}\left(\Psi x[k+1] + v[k] \right),$$

$$v[k+1] := v[k] + \Psi x[k+1] - z[k+1].$$

Convergence of ADMM

- Assume f_1 and f_2 are proper, closed, and convex functions.
- Assume also that the Lagrangian

$$L(x, z, \lambda) = f_1(x) + f_2(z) + \lambda^\top(\Psi x - z).$$

has a saddle point, that is, there exist x^*, z^*, and λ^* such that

$$L(x^*, z^*, \lambda) \leq L(x^*, z^*, \lambda^*) \leq L(x, z, \lambda^*), \quad \forall x, z, \lambda.$$

- Then,

Convergence of ADMM

- Assume f_1 and f_2 are proper, closed, and convex functions.
- Assume also that the Lagrangian

$$L(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}) = f_1(\boldsymbol{x}) + f_2(\boldsymbol{z}) + \boldsymbol{\lambda}^\top(\Psi \boldsymbol{x} - \boldsymbol{z}).$$

has a saddle point, that is, there exist x^*, z^*, and λ^* such that

$$L(x^*, z^*, \lambda) \leq L(x^*, z^*, \lambda^*) \leq L(x, z, \lambda^*), \quad \forall x, z, \lambda.$$

- Then,
  - The residual

$$r[k] \triangleq \Psi x[k] - z[k] \longrightarrow 0 \quad \text{as} \quad k \longrightarrow \infty.$$

  - The objective value

Convergence of ADMM

- Assume f_1 and f_2 are proper, closed, and convex functions.
- Assume also that the Lagrangian

$$L(x, z, \lambda) = f_1(x) + f_2(z) + \lambda^\top(\Psi x - z).$$

has a saddle point, that is, there exist x^*, z^*, and λ^* such that

$$L(x^*, z^*, \lambda) \leq L(x^*, z^*, \lambda^*) \leq L(x, z, \lambda^*), \quad \forall x, z, \lambda.$$

- Then,
  - The residual

$$r[k] \triangleq \Psi x[k] - z[k] \to 0 \quad \text{as} \quad k \to \infty.$$

  - The objective value

$$f_1(x[k]) + f_2(z[k]) \to f^* \triangleq \inf_{\substack{x \in \mathbb{R}^n, z \in \mathbb{R}^p \\ \Psi x = z}} \{f_1(x) + f_2(z)\} \quad \text{as} \quad k \to \infty.$$

  - If $\Psi^\top \Psi$ is invertible, then the sequence

$$(x[k], z[k]) \to (x^*, z^*) \quad \text{as} \quad k \to \infty.$$

Convergence of ADMM

- Assume f_1 and f_2 are proper, closed, and convex functions.
- Assume also that the Lagrangian

$$L(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}) = f_1(\boldsymbol{x}) + f_2(\boldsymbol{z}) + \boldsymbol{\lambda}^\top(\Psi\boldsymbol{x} - \boldsymbol{z}).$$

  has a saddle point, that is, there exist $\boldsymbol{x}^*$, $\boldsymbol{z}^*$, and $\boldsymbol{\lambda}^*$ such that

$$L(\boldsymbol{x}^*, \boldsymbol{z}^*, \boldsymbol{\lambda}) \leq L(\boldsymbol{x}^*, \boldsymbol{z}^*, \boldsymbol{\lambda}^*) \leq L(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}^*), \quad \forall \boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}.$$

- Then,
  - The residual

$$\boldsymbol{r}[k] \triangleq \Psi\boldsymbol{x}[k] - \boldsymbol{z}[k] \to \boldsymbol{0} \quad \text{as} \ \ k \to \infty.$$

  - The objective value

$$f_1(\boldsymbol{x}[k]) + f_2(\boldsymbol{z}[k]) \to f^* \triangleq \inf_{\substack{\boldsymbol{x}\in\mathbb{R}^n, \boldsymbol{z}\in\mathbb{R}^p \\ \Psi\boldsymbol{x}=\boldsymbol{z}}} \left\{f_1(\boldsymbol{x}) + f_2(\boldsymbol{z})\right\} \quad \text{as} \ \ k \to \infty.$$

  - If $\Psi^\top\Psi$ is invertible, then the sequence

$$(\boldsymbol{x}[k], \boldsymbol{z}[k]) \to (\boldsymbol{x}^*, \boldsymbol{z}^*) \quad \text{as} \ \ k \to \infty.$$

Convergence of ADMM

- Assume f_1 and f_2 are proper, closed, and convex functions.
- Assume also that the Lagrangian

$$L(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}) = f_1(\boldsymbol{x}) + f_2(\boldsymbol{z}) + \boldsymbol{\lambda}^\top(\Psi\boldsymbol{x} - \boldsymbol{z}).$$

  has a saddle point, that is, there exist $\boldsymbol{x}^*$, $\boldsymbol{z}^*$, and $\boldsymbol{\lambda}^*$ such that

$$L(\boldsymbol{x}^*, \boldsymbol{z}^*, \boldsymbol{\lambda}) \leq L(\boldsymbol{x}^*, \boldsymbol{z}^*, \boldsymbol{\lambda}^*) \leq L(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}^*), \quad \forall \boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}.$$

- Then,
  - The residual

$$\boldsymbol{r}[k] \triangleq \Psi\boldsymbol{x}[k] - \boldsymbol{z}[k] \to \boldsymbol{0} \quad \text{as } k \to \infty.$$

  - The objective value

$$f_1(\boldsymbol{x}[k]) + f_2(\boldsymbol{z}[k]) \to f^* \triangleq \inf_{\substack{\boldsymbol{x}\in\mathbb{R}^n, \boldsymbol{z}\in\mathbb{R}^p \\ \Psi\boldsymbol{x}=\boldsymbol{z}}} \{f_1(\boldsymbol{x}) + f_2(\boldsymbol{z})\} \quad \text{as } k \to \infty.$$

  - If $\Psi^\top\Psi$ is invertible, then the sequence

$$(\boldsymbol{x}[k], \boldsymbol{z}[k]) \to (\boldsymbol{x}^*, \boldsymbol{z}^*) \quad \text{as } k \to \infty.$$

Convergence of ADMM

- Assume f_1 and f_2 are proper, closed, and convex functions.
- Assume also that the Lagrangian

$$L(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}) = f_1(\boldsymbol{x}) + f_2(\boldsymbol{z}) + \boldsymbol{\lambda}^\top(\Psi\boldsymbol{x} - \boldsymbol{z}).$$

  has a saddle point, that is, there exist $\boldsymbol{x}^*$, $\boldsymbol{z}^*$, and $\boldsymbol{\lambda}^*$ such that

$$L(\boldsymbol{x}^*, \boldsymbol{z}^*, \boldsymbol{\lambda}) \leq L(\boldsymbol{x}^*, \boldsymbol{z}^*, \boldsymbol{\lambda}^*) \leq L(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}^*), \quad \forall \boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}.$$

- Then,
  - The residual

$$\boldsymbol{r}[k] \triangleq \Psi\boldsymbol{x}[k] - \boldsymbol{z}[k] \to \mathbf{0} \quad \text{as } k \to \infty.$$

  - The objective value

$$f_1(\boldsymbol{x}[k]) + f_2(\boldsymbol{z}[k]) \to f^* \triangleq \inf_{\substack{\boldsymbol{x}\in\mathbb{R}^n, \boldsymbol{z}\in\mathbb{R}^p \\ \Psi\boldsymbol{x}=\boldsymbol{z}}} \left\{ f_1(\boldsymbol{x}) + f_2(\boldsymbol{z}) \right\} \quad \text{as } k \to \infty.$$

  - If $\Psi^\top\Psi$ is invertible, then the sequence

$$(\boldsymbol{x}[k], \boldsymbol{z}[k]) \to (\boldsymbol{x}^*, \boldsymbol{z}^*) \quad \text{as } k \to \infty.$$

ADMM for generalized LASSO

- Generalized LASSO

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \ \frac{1}{2}\|\Phi\boldsymbol{x}-\boldsymbol{y}\|_2^2+\lambda\|\Psi\boldsymbol{x}\|_1,$$

- The first update in ADMM:

$$\begin{aligned}\underset{\boldsymbol{x}\in\mathbb{R}^n}{\arg\min}&\left\{\frac{1}{2}\|\Phi\boldsymbol{x}-\boldsymbol{y}\|_2^2+\frac{1}{2\gamma}\|\Psi\boldsymbol{x}-\boldsymbol{z}[k]+\boldsymbol{v}[k]\|_2^2\right\}\\&=\left(\Phi^\top\Phi+\gamma^{-1}\Psi^\top\Psi\right)^{-1}\left(\Phi^\top\boldsymbol{y}+\gamma^{-1}\Psi^\top(\boldsymbol{z}[k]-\boldsymbol{v}[k])\right).\end{aligned}$$

- The proximal operator in the second update is the soft-thresholding operator:

$$\operatorname{prox}_{\gamma f_2}(\boldsymbol{v})=S_{\gamma\lambda}(\boldsymbol{v})$$

ADMM for generalized LASSO

- Generalized LASSO

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}}\ \frac{1}{2}\|\Phi\boldsymbol{x}-\boldsymbol{y}\|_2^2+\lambda\|\Psi\boldsymbol{x}\|_1,$$

- The first update in ADMM:

$$\begin{aligned}\underset{\boldsymbol{x}\in\mathbb{R}^n}{\arg\min}&\Big\{\frac{1}{2}\|\Phi\boldsymbol{x}-\boldsymbol{y}\|_2^2+\frac{1}{2\gamma}\|\Psi\boldsymbol{x}-\boldsymbol{z}[k]+\boldsymbol{v}[k]\|_2^2\Big\}\\&=\big(\Phi^\top\Phi+\gamma^{-1}\Psi^\top\Psi\big)^{-1}\big(\Phi^\top\boldsymbol{y}+\gamma^{-1}\Psi^\top(\boldsymbol{z}[k]-\boldsymbol{v}[k])\big).\end{aligned}$$

- The proximal operator in the second update is the soft-thresholding operator:

$$\text{prox}_{\gamma f_2}(\boldsymbol{v})=S_{\gamma\lambda}(\boldsymbol{v})$$

ADMM for generalized LASSO

- Generalized LASSO

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \frac{1}{2}\|\Phi x - y\|_2^2 + \lambda\|\Psi x\|_1,$$

- The first update in ADMM:

$$\begin{aligned}\underset{x \in \mathbb{R}^n}{\arg\min}\Big\{\frac{1}{2}\|\Phi x - y\|_2^2 + \frac{1}{2\gamma}\|\Psi x - z[k] + v[k]\|_2^2\Big\} \\ = \big(\Phi^\top\Phi + \gamma^{-1}\Psi^\top\Psi\big)^{-1}\big(\Phi^\top y + \gamma^{-1}\Psi^\top(z[k] - v[k])\big).\end{aligned}$$

- The proximal operator in the second update is the soft-thresholding operator:

$$\text{prox}_{\gamma f_2}(v) = S_{\gamma\lambda}(v)$$

ADMM for generalized LASSO

ADMM for generalized LASSO

Initialization: give initial vectors $\boldsymbol{z}[0], \boldsymbol{v}[0] \in \mathbb{R}^p$, and real number $\gamma > 0$
Iteration: for $k = 0, 1, 2, \ldots$ do

$$\begin{aligned}
\boldsymbol{x}[k+1] &= \left(\Phi^\top \Phi + \gamma^{-1}\Psi^\top \Psi\right)^{-1}\left(\Phi^\top \boldsymbol{y} + \gamma^{-1}\Psi^\top(\boldsymbol{z}[k] - \boldsymbol{v}[k])\right)\\
\boldsymbol{z}[k+1] &= S_{\gamma\lambda}\left(\Psi \boldsymbol{x}[k+1] + \boldsymbol{v}[k]\right)\\
\boldsymbol{v}[k+1] &= \boldsymbol{v}[k] + \Psi \boldsymbol{x}[k+1] - \boldsymbol{z}[k+1].
\end{aligned}$$

- The inverse matrix $(\Phi^\top \Phi + \gamma^{-1}\Psi^\top \Psi)^{-1}$ is computed offline.
- If the matrix $\Phi^\top \Phi + \gamma^{-1}\Psi^\top \Psi$ is a tridiagonal matrix, the linear equation

$$\left(\Phi^\top \Phi + \gamma^{-1}\Psi^\top \Psi\right) x = v$$

with unknown x can be solved in $O(n)$.

ADMM for generalized LASSO

Initialization: give initial vectors $\boldsymbol{z}[0], \boldsymbol{v}[0] \in \mathbb{R}^p$, and real number $\gamma > 0$

Iteration: for $k = 0, 1, 2, \ldots$ do

$$
\begin{aligned}
\boldsymbol{x}[k+1] &= \left(\Phi^\top \Phi + \gamma^{-1} \Psi^\top \Psi\right)^{-1} \left(\Phi^\top \boldsymbol{y} + \gamma^{-1} \Psi^\top (\boldsymbol{z}[k] - \boldsymbol{v}[k])\right) \\
\boldsymbol{z}[k+1] &= S_{\gamma\lambda}\left(\Psi \boldsymbol{x}[k+1] + \boldsymbol{v}[k]\right) \\
\boldsymbol{v}[k+1] &= \boldsymbol{v}[k] + \Psi \boldsymbol{x}[k+1] - \boldsymbol{z}[k+1].
\end{aligned}
$$

- The inverse matrix $(\Phi^\top \Phi + \gamma^{-1} \Psi^\top \Psi)^{-1}$ is computed offline.
- If the matrix $\Phi^\top \Phi + \gamma^{-1} \Psi^\top \Psi$ is a tridiagonal matrix, the linear equation

$$\left(\Phi^\top \Phi + \gamma^{-1} \Psi^\top \Psi\right) x = v$$

with unknown x can be solved in $O(n)$.

ADMM for generalized LASSO

ADMM for generalized LASSO

Initialization: give initial vectors $\boldsymbol{z}[0], \boldsymbol{v}[0] \in \mathbb{R}^p$, and real number $\gamma > 0$

Iteration: for $k = 0, 1, 2, \ldots$ do

$$\begin{aligned}
\boldsymbol{x}[k+1] &= \left(\Phi^\top \Phi + \gamma^{-1}\Psi^\top\Psi\right)^{-1}\left(\Phi^\top \boldsymbol{y} + \gamma^{-1}\Psi^\top(\boldsymbol{z}[k] - \boldsymbol{v}[k])\right)\\
\boldsymbol{z}[k+1] &= S_{\gamma\lambda}\left(\Psi\boldsymbol{x}[k+1] + \boldsymbol{v}[k]\right)\\
\boldsymbol{v}[k+1] &= \boldsymbol{v}[k] + \Psi\boldsymbol{x}[k+1] - \boldsymbol{z}[k+1].
\end{aligned}$$

- The inverse matrix $(\Phi^\top\Phi + \gamma^{-1}\Psi^\top\Psi)^{-1}$ is computed offline.
- If the matrix $\Phi^\top\Phi + \gamma^{-1}\Psi^\top\Psi$ is a tridiagonal matrix, the linear equation

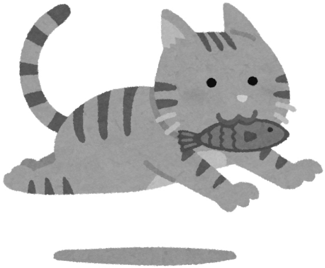
$$\left(\Phi^\top\Phi + \gamma^{-1}\Psi^\top\Psi\right)\boldsymbol{x} = \boldsymbol{v}$$

  with unknown $\boldsymbol{x}$ can be solved in $O(n)$.

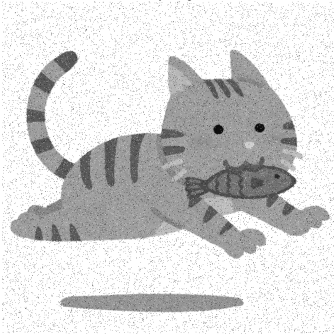
Application: Image denoising

- Remove noise from an image.
- Preserve edges at the same time.
  - Applying a low-pass filter does not work very well.

Original image



Noisy image

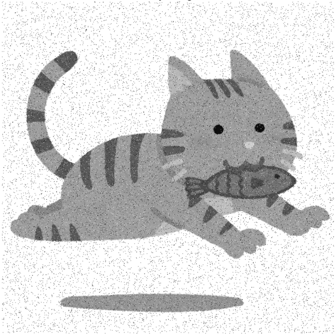


Application: Image denoising

- Remove noise from an image.
- Preserve edges at the same time.
    - Applying a low-pass filter does not work very well.

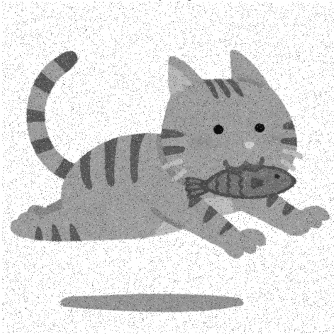
Original image

Noisy image



Application: Image denoising

- Remove noise from an image.
- Preserve edges at the same time.
  - Applying a low-pass filter does not work very well.



Total variation denoising

- $Y \in \mathbb{R}^{n \times m}$: a noisy image
- Pull out each column vector, say $y \in \mathbb{R}^n$, and solve the following optimization problem, one by one:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \|x - y\|_2^2 + \lambda \sum_{i=1}^{n} |x_{i+1} - x_i|.$$

- $\|x - y\|_2^2$: proximity to the data
- $\sum_{i=1}^{n} |x_{i+1} - x_i|$: total variation to preserve edges
- $\lambda > 0$: weight

Total variation denoising

- $\boldsymbol{Y} \in \mathbb{R}^{n \times m}$: a noisy image
- Pull out each column vector, say $\boldsymbol{y} \in \mathbb{R}^n$, and solve the following optimization problem, one by one:

$$\underset{\boldsymbol{x} \in \mathbb{R}^n}{\text{minimize}} \ \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \sum_{i=1}^{n} |x_{i+1} - x_i|.$$

- $\|x - y\|_2^2$: proximity to the data
- $\sum_{i=1}^{n} |x_{i+1} - x_i|$: total variation to preserve edges
- $\lambda > 0$: weight

Total variation denoising

- $Y \in \mathbb{R}^{n \times m}$: a noisy image
- Pull out each column vector, say $y \in \mathbb{R}^n$, and solve the following optimization problem, one by one:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \|x - y\|_2^2 + \lambda \sum_{i=1}^{n} |x_{i+1} - x_i|.$$

- $\|x - y\|_2^2$: proximity to the data
- $\sum_{i=1}^{n} |x_{i+1} - x_i|$: total variation to preserve edges
- $\lambda > 0$: weight

Total variation denoising

- $Y \in \mathbb{R}^{n \times m}$: a noisy image
- Pull out each column vector, say $y \in \mathbb{R}^n$, and solve the following optimization problem, one by one:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \|x - y\|_2^2 + \lambda \sum_{i=1}^{n} |x_{i+1} - x_i|.$$

- $\|x - y\|_2^2$: proximity to the data
- $\sum_{i=1}^{n} |x_{i+1} - x_i|$: total variation to preserve edges
- $\lambda > 0$: weight

Total variation denoising

- $\boldsymbol{Y} \in \mathbb{R}^{n \times m}$: a noisy image
- Pull out each column vector, say $\boldsymbol{y} \in \mathbb{R}^n$, and solve the following optimization problem, one by one:

$$\underset{\boldsymbol{x} \in \mathbb{R}^n}{\text{minimize}} \ \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \sum_{i=1}^{n} |x_{i+1} - x_i|.$$

- $\|\boldsymbol{x} - \boldsymbol{y}\|_2^2$: proximity to the data
- $\sum_{i=1}^{n} |x_{i+1} - x_i|$: total variation to preserve edges
- $\lambda > 0$: weight

ADMM for total variation denoising

- Total variation denoising:

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \ \|x-y\|_2^2 + \lambda \sum_{i=1}^{n} |x_{i+1}-x_i|.$$

- Define $\Phi = I$ and

$$\Psi = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & -1 & 1 \\ 0 & \dots & 0 & 0 & -1 \end{bmatrix}.$$

- Generalized LASSO

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \ \|\Phi x-y\|_2^2 + \lambda\|\Psi x\|_1,$$

- We can use ADMM!

ADMM for total variation denoising

- Total variation denoising:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \; \|x - y\|_2^2 + \lambda \sum_{i=1}^{n} |x_{i+1} - x_i|.$$

- Define $\Phi = I$ and

$$\Psi = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & -1 & 1 \\ 0 & \dots & 0 & 0 & -1 \end{bmatrix}.$$

- Generalized LASSO

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \; \|\Phi x - y\|_2^2 + \lambda \|\Psi x\|_1,$$

- We can use ADMM!

ADMM for total variation denoising

- Total variation denoising:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \;\; \|x - y\|_2^2 + \lambda \sum_{i=1}^{n} |x_{i+1} - x_i|.$$

- Define $\Phi = I$ and

$$\Psi = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & -1 & 1 \\ 0 & \dots & 0 & 0 & -1 \end{bmatrix}.$$

- Generalized LASSO

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \;\; \|\Phi x - y\|_2^2 + \lambda \|\Psi x\|_1,$$

- We can use ADMM!

ADMM for total variation denoising

- Total variation denoising:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \|x - y\|_2^2 + \lambda \sum_{i=1}^{n} |x_{i+1} - x_i|.$$

- Define $\Phi = I$ and

$$\Psi = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & -1 & 1 \\ 0 & \dots & 0 & 0 & -1 \end{bmatrix}.$$

- Generalized LASSO

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \|\Phi x - y\|_2^2 + \lambda \|\Psi x\|_1,$$

- We can use ADMM!

ADMM for total variation denoising

- The weight λ should be carefully chosen.
- $\lambda = 50$

ADMM for total variation denoising

- The weight λ should be carefully chosen.
- $\lambda = 100$

Restored image

ADMM for total variation denoising

- The weight λ should be carefully chosen.
- $\lambda = 200$

Summary

- In convex optimization, a local minimum is a global minimum.
- ℓ^1 optimization problems appeared in this course are convex optimization.
- Proximal operators are used to derive fast algorithms for convex optimization with non-differentiable ℓ^1 norm and constraints

Summary

- In convex optimization, a local minimum is a global minimum.
- ℓ^1 optimization problems appeared in this course are convex optimization.
- Proximal operators are used to derive fast algorithms for convex optimization with non-differentiable ℓ^1 norm and constraints

Summary

- In convex optimization, a local minimum is a global minimum.
- ℓ^1 optimization problems appeared in this course are convex optimization.
- Proximal operators are used to derive fast algorithms for convex optimization with non-differentiable ℓ^1 norm and constraints