Optimization Techniques for Big Data Analysis

Chapter 3. Review of Fundamentals of Convex Optimization

Master of Science in Signal Theory and Communications
Dpto. de Señales, Sistemas y Radiocomunicaciones
E.T.S. Ingenieros de Telecomunicación
Universidad Politécnica de Madrid

2023

Convex sets

A set C is **convex** if the line segment between any two points of C lies in C, i.e., if for any $\mathbf{x}, \mathbf{y} \in C$ and any λ with $0 \leq \lambda \leq 1$, we have

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

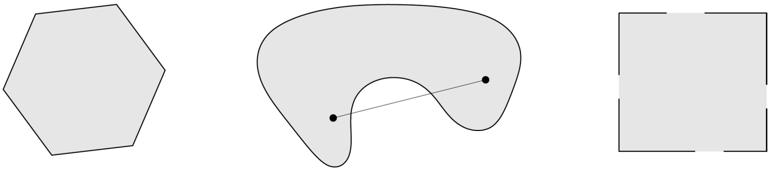


*Figure 2.2 from S. Boyd, L. Vandenberghe

left Convex

Middle Not Convex, since line segment not in set

Right Not convex, since some, but not all boundary points are contained in the set

Convex functions

A function $f : \mathbb{R}^d \to \mathbb{R}$ is **convex** if (i) $\mathbf{dom}(f)$ is a convex set and (ii) for all $\mathbf{x}, \mathbf{y} \in \mathbf{dom}(f)$, λ with $0 \leq \lambda \leq 1$, we have

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

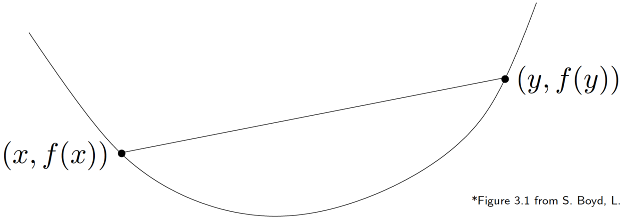
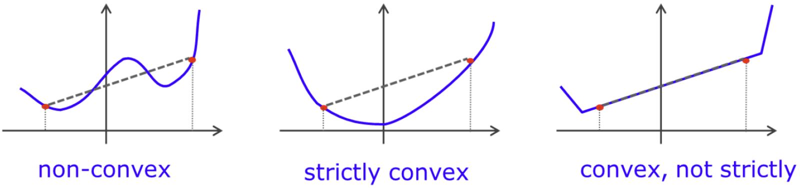


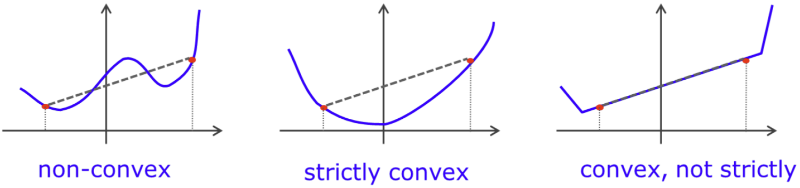
*Figure 3.1 from S. Boyd, L. Vandenberghe

Geometrically: The line segment between $(\mathbf{x}, f(\mathbf{x}))$ and $(\mathbf{y}, f(\mathbf{y}))$ lies above the graph of f (called epigraph)

Convex functions



Convex functions



Let's think about the following functions, are they convex?

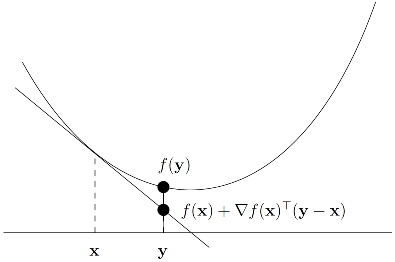
- $f(x) = \exp(x)$
- $f(x) = \log(x)$
- $f(x) = \sin(x)$
- $f(x) = \max\{x, 0\}$
- $f(x) = \log(1 + \exp(-x))$
- $f(x) = \sqrt{x}$

Convex functions

If the function $f(\cdot)$ is differentiable, the Jensen inequality can also be expressed in an alternative way:

$$f\left(\mathbf{y}\right) \geq f\left(\mathbf{x}\right) + \nabla f\left(\mathbf{x}\right)^{T}\left(\mathbf{y}-\mathbf{x}\right) \qquad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}$$

which establishes that the graph of f is above all its tangent hyperplanes.

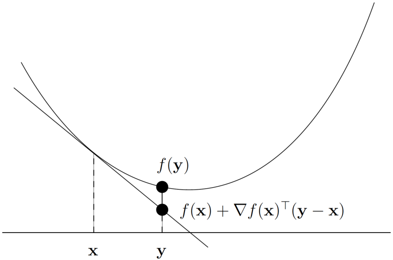


Convex functions

If the function $f(\cdot)$ is differentiable, the Jensen inequality can also be expressed in an alternative way:

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T (\mathbf{y} - \mathbf{x}) \qquad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}$$

which establishes that the graph of f is above all its tangent hyperplanes.



Besides, if $f(\cdot)$ is twice differentiable, convexity implies

$$\nabla^2 f(\mathbf{x}) \succeq 0, \ \forall \mathbf{x} \in \mathbb{R}^d$$

Why is convexity important?

Why is convexity important?

For a Convex function "*All local minima are global minima*"

Why is convexity important?

For a Convex function "*All local minima are global minima*"

Let's think about an optimization problem:

$$\arg\min_{\mathbf{x}} f(\mathbf{x})$$
$$\text{s.t. } \mathbf{x} \in \mathcal{B}$$

Why is convexity important?

For a Convex function "*All local minima are global minima*"

Let's think about an optimization problem:

$$\arg\min_{\mathbf{x}} f(\mathbf{x})$$
$$\text{s.t. } \mathbf{x} \in \mathcal{B}$$

From the first-order Taylor's expansion, we know that the rule:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \nabla f(\mathbf{x}_k),\ k = 1, 2, ...$$

is a descent algorithm for small values of η.

Why is convexity important?

For a Convex function "*All local minima are global minima*"

Let's think about an optimization problem:

$$\arg\min_{\mathbf{x}} f(\mathbf{x})$$
$$\text{s.t. } \mathbf{x} \in \mathcal{B}$$

From the first-order Taylor's expansion, we know that the rule:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \nabla f(\mathbf{x}_k),\ k = 1, 2, ...$$

is a descent algorithm for small values of η. So, Gradient descent can find one minimizer if $f(\cdot)$ is convex.

Smooth optimization

Assuming a Lipschitz (L is the Lipschitz constant) continuous function:

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq L \|\mathbf{x} - \mathbf{y}\|_2$$

That basically means that it can not change very quickly.

Smooth optimization

Assuming a Lipschitz (L is the Lipschitz constant) continuous function:

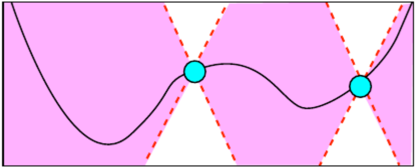
$$|f(\mathbf{x}) - f(\mathbf{y})| \leq L \|\mathbf{x} - \mathbf{y}\|_2$$

That basically means that it can not change very quickly.

A differentiable function is Lipschitz on a convex domain iff:

$$\|\nabla f(\mathbf{x})\|_\infty = \max_j |\nabla f(x_j)| \leq \infty$$

where x_j represents any component of variable $\mathbf{x}$ and $\max_j |\nabla f(x_j)| = L$.



Smooth optimization

In the literature, functions whose derivatives are Lipschitz continuous are also known as $L-$smooth functions ($L > 0$), i.e.:

$$\left\|\nabla f\left(\mathbf{x}\right) - \nabla f\left(\mathbf{y}\right)\right\|_2 \leq L \left\|\mathbf{x} - \mathbf{y}\right\|_2$$

This added condition is very remarkable because the Hessian is upper bounded as follows:

$$\nabla^2 f\left(\mathbf{x}\right) \preceq L\mathbf{I} \qquad \forall \mathbf{x} \in \mathbb{R}^d$$

Smooth optimization

In the literature, functions whose derivatives are Lipschitz continuous are also known as $L-$smooth functions ($L > 0$), i.e.:

$$\left\|\nabla f\left(\mathbf{x}\right) - \nabla f\left(\mathbf{y}\right)\right\|_2 \leq L \left\|\mathbf{x} - \mathbf{y}\right\|_2$$

This added condition is very remarkable because the Hessian is upper bounded as follows:

$$\nabla^2 f\left(\mathbf{x}\right) \preceq L\mathbf{I} \qquad \forall \mathbf{x} \in \mathbb{R}^d$$

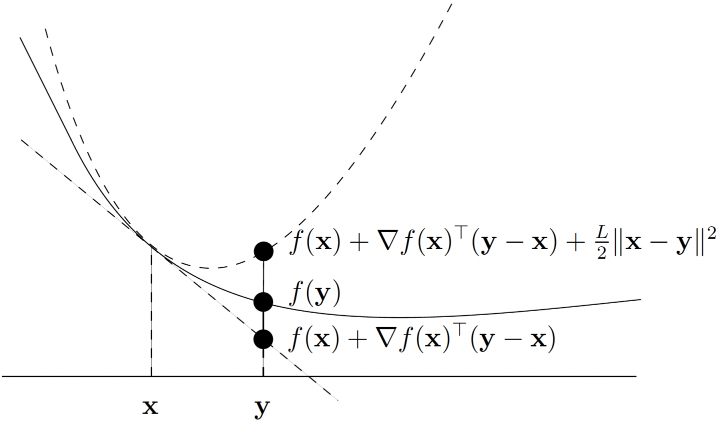
If we approximate the original function by a second-order Taylor series:

$$f\left(\mathbf{y}\right) \approx f\left(\mathbf{x}\right) + \nabla f\left(\mathbf{x}\right)^T \left(\mathbf{y} - \mathbf{x}\right) + \frac{1}{2}\left(\mathbf{y} - \mathbf{x}\right)^T \nabla^2 f\left(\mathbf{x}\right)\left(\mathbf{y} - \mathbf{x}\right)$$

we have the following quadratic upper bound:

$$f\left(\mathbf{y}\right) \leq f\left(x\right) + \nabla f\left(\mathbf{x}\right)^T \left(\mathbf{y} - \mathbf{x}\right) + \frac{L}{2}\left\|\mathbf{y} - \mathbf{x}\right\|_2^2$$

Smooth optimization



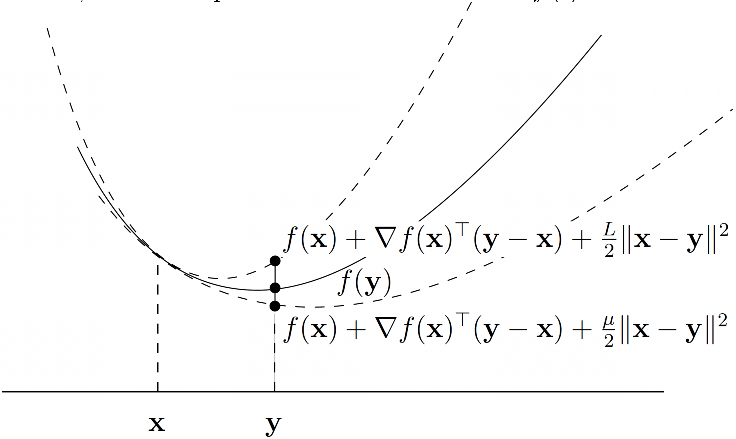
Meaning: we can set a bound on the function rate of variation. It defines the maximum speed of convergence of an iterative algorithm (step-size bound).

Strong convexity

Another important concept is related to **strong** convexity. We define a $\mu-$strong convex ($\mu > 0$) if this inequality fulfills:

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla^T f(\mathbf{x})(\mathbf{y}-\mathbf{x}) + \frac{\mu}{2}\|\mathbf{y}-\mathbf{x}\|_2^2$$

Therefore, this is a quadratic lower bound of $f(\cdot)$.



Strong convexity

For a strongly convex function, "*There exists a unique local minimum which is also global.*"

Strong convexity

For a strongly convex function, "*There exists a unique local minimum which is also global.*"

Besides, regarding the second-order condition, we now have

$$\nabla^2 f(\mathbf{x}) \succeq \mu \mathbf{I} \qquad \forall \mathbf{x} \in \mathbb{R}$$

Strong convexity

For a strongly convex function, "*There exists a unique local minimum which is also global.*"

Besides, regarding the second-order condition, we now have

$$\nabla^2 f(\mathbf{x}) \succeq \mu \mathbf{I} \qquad \forall \mathbf{x} \in \mathbb{R}$$

Why is it relevant?

Strong convexity

For a strongly convex function, "*There exists a unique local minimum which is also global.*"

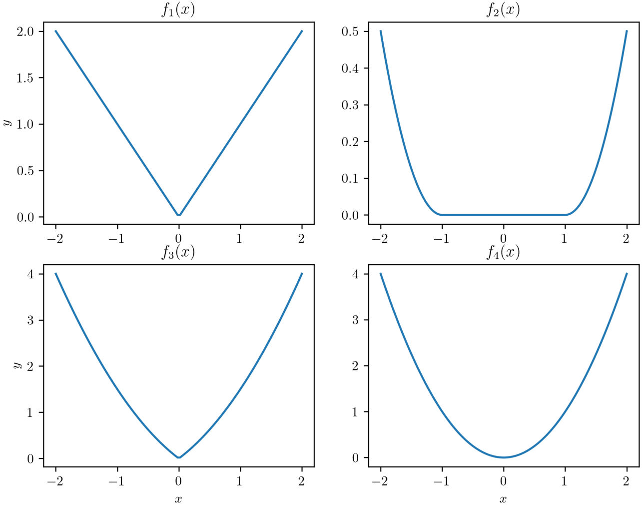
Besides, regarding the second-order condition, we now have

$$\nabla^2 f(\mathbf{x}) \succeq \mu \mathbf{I} \qquad \forall \mathbf{x} \in \mathbb{R}$$

Why is it relevant?

1. Provides "self-tuning" property to the gradient regarding gradient descent algorithm.
2. Guarantee of faster convergence
3. Guarantee of the existence of a single minimum

Classify the following functions



Example 3.1

Calculate L and μ for the Ridge regressor (norm-2 regularizer)

$$\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \left(\frac{1}{n} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 \right)$$

Example 3.1

Calculate L and μ for the Ridge regressor (norm-2 regularizer)

$$\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \left(\frac{1}{n}\left\|\mathbf{X}\mathbf{w}-\mathbf{y}\right\|_2^2+\frac{\lambda}{2}\left\|\mathbf{w}\right\|_2^2\right)$$

we have: $\nabla^2 f\left(\mathbf{w}\right)=\frac{2}{n}\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I}$

Example 3.1

Calculate L and μ for the Ridge regressor (norm-2 regularizer)

$$\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \left(\frac{1}{n} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 \right)$$

we have: $\nabla^2 f(\mathbf{w}) = \frac{2}{n}\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}$

Since the Hessian matrix $\mathbf{H} = \frac{2}{n}\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}$ is symmetric, the eigendecomposition is $\mathbf{H} = \mathbf{U}(\Sigma + \lambda\mathbf{I})\mathbf{U}^T$

Example 3.1

Calculate L and μ for the Ridge regressor (norm-2 regularizer)

$$\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \left(\frac{1}{n}\|\mathbf{X}\mathbf{w}-\mathbf{y}\|_2^2+\frac{\lambda}{2}\|\mathbf{w}\|_2^2\right)$$

we have: $\nabla^2 f(\mathbf{w}) = \frac{2}{n}\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I}$

Since the Hessian matrix $\mathbf{H}=\frac{2}{n}\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I}$ is symmetric, the eigendecomposition is $\mathbf{H}=\mathbf{U}(\Sigma+\lambda\mathbf{I})\mathbf{U}^T$

Considering that $\mathbf{X}$ is not full rank (Why?), what can be said about $\mathbf{H}$ for:

- $\lambda = 0$
- $\lambda = 1$

Example 3.1

We know that $\mathbf{H} = \sum_{j=0}^{d} \lambda_j \mathbf{u}_j \mathbf{u}_j^T$. $\mathbf{H}$ is known to be positive semidefinite definite $\mathbf{H} \succeq 0$ implying that all eigenvalues are non negative $\lambda_j \geq 0 \;\; \forall j$ and matrices $\mathbf{u}_j \mathbf{u}_j^T \succ 0$.Therefore

$$\begin{cases} \sum_{j=0}^{d} \lambda_j \mathbf{u}_j \mathbf{u}_j^T \leq \sum_{j=0}^{d} \lambda_{\max} \mathbf{u}_j \mathbf{u}_j^T = \lambda_{\max} \sum_{j=0}^{d} \mathbf{u}_j \mathbf{u}_j^T = \lambda_{\max} \mathbf{U}\mathbf{U}^T = \lambda_{\max} \mathbf{I} \\ \sum_{j=0}^{d} \lambda_j \mathbf{u}_j \mathbf{u}_j^T \geq \sum_{j=0}^{d} \lambda_{\min} \mathbf{u}_j \mathbf{u}_j^T = \lambda_{\min} \sum_{j=0}^{d} \mathbf{u}_j \mathbf{u}_j^T = \lambda_{\min} \mathbf{U}\mathbf{U}^T = \lambda_{\min} \mathbf{I} \end{cases}$$

Example 3.1

We know that $\mathbf{H} = \sum_{j=0}^{d} \lambda_j \mathbf{u}_j \mathbf{u}_j^T$. $\mathbf{H}$ is known to be positive semidefinite definite $\mathbf{H} \succeq 0$ implying that all eigenvalues are non negative $\lambda_j \geq 0 \;\; \forall j$ and matrices $\mathbf{u}_j \mathbf{u}_j^T \succ 0$.Therefore

$$\begin{cases} \sum_{j=0}^{d} \lambda_j \mathbf{u}_j \mathbf{u}_j^T \leq \sum_{j=0}^{d} \lambda_{\max} \mathbf{u}_j \mathbf{u}_j^T = \lambda_{\max} \sum_{j=0}^{d} \mathbf{u}_j \mathbf{u}_j^T = \lambda_{\max} \mathbf{U}\mathbf{U}^T = \lambda_{\max} \mathbf{I} \\ \sum_{j=0}^{d} \lambda_j \mathbf{u}_j \mathbf{u}_j^T \geq \sum_{j=0}^{d} \lambda_{\min} \mathbf{u}_j \mathbf{u}_j^T = \lambda_{\min} \sum_{j=0}^{d} \mathbf{u}_j \mathbf{u}_j^T = \lambda_{\min} \mathbf{U}\mathbf{U}^T = \lambda_{\min} \mathbf{I} \end{cases}$$

Finally, we have

$$\lambda_{\min}\left(\mathbf{H}\right)\mathbf{I} \preceq \nabla^2 f\left(\mathbf{w}\right) \preceq \lambda_{\max}\left(\mathbf{H}\right)\mathbf{I} \qquad \forall \mathbf{w} \in \mathbb{R}^{d+1}$$

The quotient $\lambda_{\max} \,/\, \lambda_{\min} = L/\mu$ is known as the condition number and affects the convergence rate.

Smoothness/ strong convexity

$$\mu \mathbf{I} \preceq \nabla^2 f(\mathbf{w}) \preceq L\mathbf{I}$$

Smoothness/ strong convexity

$$\mu \mathbf{I} \preceq \nabla^2 f\left(\mathbf{w}\right) \preceq L\mathbf{I}$$

1. **Smoothness** (L) establishes the maximum step size that guarantees convergence in iterative processes $(n < \frac{1}{L})$.

Smoothness/ strong convexity

$$\mu\mathbf{I} \preceq \nabla^2 f\left(\mathbf{w}\right) \preceq L\mathbf{I}$$

1. **Smoothness** (L) establishes the maximum step size that guarantees convergence in iterative processes $(n < \frac{1}{L})$.
2. **Strong convexity** determines that the optimization process has a single solution and guarantees that the convergence rate is acceptable because there are no flat regions ("self-tunning").

Smoothness/ strong convexity

$$\mu \mathbf{I} \preceq \nabla^2 f(\mathbf{w}) \preceq L\mathbf{I}$$

1. **Smoothness** (L) establishes the maximum step size that guarantees convergence in iterative processes ($n < \frac{1}{L}$).
2. **Strong convexity** determines that the optimization process has a single solution and guarantees that the convergence rate is acceptable because there are no flat regions ("self-tunning").
3. If $\kappa = L/\mu \approx 1$, the problem is well conditioned, and the convergence rate using gradient methods is typically very competitive.

Smoothness/ strong convexity

$$\mu \mathbf{I} \preceq \nabla^2 f(\mathbf{w}) \preceq L\mathbf{I}$$

1. **Smoothness** (L) establishes the maximum step size that guarantees convergence in iterative processes $(n < \frac{1}{L})$.
2. **Strong convexity** determines that the optimization process has a single solution and guarantees that the convergence rate is acceptable because there are no flat regions ("self-tunning").
3. If $\kappa = L/\mu \approx 1$, the problem is well conditioned, and the convergence rate using gradient methods is typically very competitive.

What does data normalization have to do with these characteristics?

Example 3.2

Review the corresponding notebook.

Example 3.2

Review the corresponding notebook.

- In the code, you can see that we generate a random matrix with 5 rows and 7 columns. Matrix $\mathbf{A}^T\mathbf{A}$ has dimension 7×7 but the rank is 5 because $\mathbf{A}$ has dimension 5×7.

Example 3.2

Review the corresponding notebook.

- In the code, you can see that we generate a random matrix with 5 rows and 7 columns. Matrix $\mathbf{A}^T\mathbf{A}$ has dimension 7×7 but the rank is 5 because $\mathbf{A}$ has dimension 5×7.
- For this analysis, you need to calculate the eigenvalues and check that the rank is related to the number of nonzeros eigenvalues. In this case, you can notice that the last two are negligible.

Example 3.2

Review the corresponding notebook.

- In the code, you can see that we generate a random matrix with 5 rows and 7 columns. Matrix $\mathbf{A}^T\mathbf{A}$ has dimension 7×7 but the rank is 5 because $\mathbf{A}$ has dimension 5×7.
- For this analysis, you need to calculate the eigenvalues and check that the rank is related to the number of nonzeros eigenvalues. In this case, you can notice that the last two are negligible.
- However, if we add a full rank matrix as the (scaled) identity matrix, the combination is full rank (7), and the system is strongly convex because you have a determined system of equations with a single solution.

Rates of convergence: Nomenclature

Convergence is a fundamental property to assess the quality of an optimization algorithm, but as a first step, we need to clarify some terminology.

Rates of convergence: Nomenclature

Convergence is a fundamental property to assess the quality of an optimization algorithm, but as a first step, we need to clarify some terminology.

- When talking about how fast a positive sequence $\{\zeta_k\} = \{f(x_k) - f(x^*)\}$ of scalars is decreasing to zero, we have different types of convergence rates toward the optimum value $f(x^*)$.

Rates of convergence: Nomenclature

Convergence is a fundamental property to assess the quality of an optimization algorithm, but as a first step, we need to clarify some terminology.

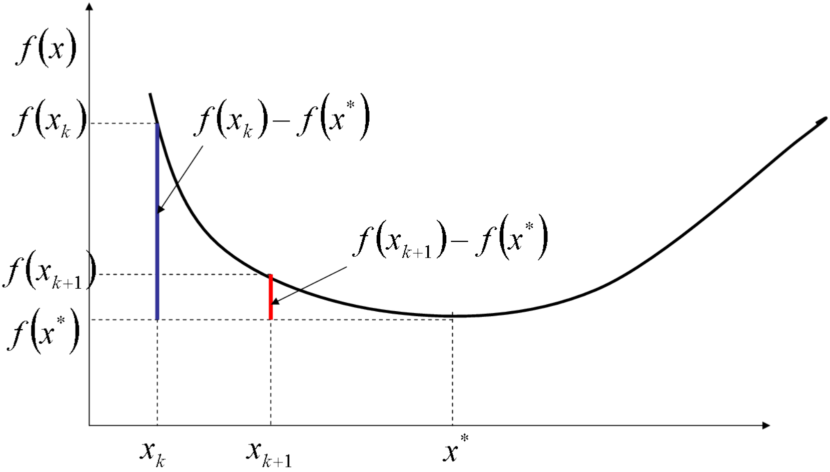
- When talking about how fast a positive sequence $\{\zeta_k\} = \{f(x_k) - f(x^*)\}$ of scalars is decreasing to zero, we have different types of convergence rates toward the optimum value $f(x^*)$.
- Bounds are expressed in terms of the order of magnitude $\mathcal{O}(\zeta_k)$ of the rate.

Rates of convergence: Nomenclature

Convergence is a fundamental property to assess the quality of an optimization algorithm, but as a first step, we need to clarify some terminology.

- When talking about how fast a positive sequence $\{\zeta_k\} = \{f(x_k) - f(x^*)\}$ of scalars is decreasing to zero, we have different types of convergence rates toward the optimum value $f(x^*)$.
- Bounds are expressed in terms of the order of magnitude $\mathcal{O}(\zeta_k)$ of the rate.
- It is also interesting to calculate the number of iterations requested in order to achieve a certain accuracy ϵ. That calculation just required to solve $\zeta_k < \varepsilon \to k < \lceil \zeta_k^{-1}(\varepsilon) \rceil$ where notation $\lceil\rceil$ refers to the next integer.

Rates of convergence



Rates of convergence

Types of convergence:

Sublinear : $\zeta_k \to 0$, but $\zeta_{k+1}/\zeta_k \to 1$.

- Example: $\zeta_k \leq C/k^q$, for $q > 0$ and $C > 0$
- To achieve an ε error $k \sim \mathcal{O}\left(1/\varepsilon^{1/q}\right)$.

Rates of convergence

Types of convergence:

Sublinear : $\zeta_k \to 0$, but $\zeta_{k+1}/\zeta_k \to 1$.

- Example: $\zeta_k \leq C/k^q$, for $q > 0$ and $C > 0$
- To achieve an ε error $k \sim \mathcal{O}\left(1/\varepsilon^{1/q}\right)$.

Linear : $\zeta_{k+1}/\zeta_k \leq r$ for some $r \in (0, 1)$.

- Example: $\zeta_k \leq Cq^k$, for $q \in (0, 1)$ and $C > 0$
- To achieve an ε error, $k \sim \mathcal{O}\left(\log(1/\varepsilon)\right)$.

Rates of convergence

Types of convergence:

Sublinear : $\zeta_k \to 0$, but $\zeta_{k+1}/\zeta_k \to 1$.

- Example: $\zeta_k \leq C/k^q$, for $q > 0$ and $C > 0$
- To achieve an ε error $k \sim \mathcal{O}\left(1/\varepsilon^{1/q}\right)$.

Linear : $\zeta_{k+1}/\zeta_k \leq r$ for some $r \in (0, 1)$.

- Example: $\zeta_k \leq Cq^k$, for $q \in (0, 1)$ and $C > 0$
- To achieve an ε error, $k \sim \mathcal{O}\left(\log(1/\varepsilon)\right)$.

Superlinear : $\zeta_{k+1}/\zeta_k \to 0$.

- Example: $\zeta_k \leq Cq^{k^2}$, for $q \in (0, 1)$ and $C > 0$
- To achieve an ε error, $k \sim \mathcal{O}\left(\sqrt{\log(1/\varepsilon)}\right)$.

Rates of convergence

Types of convergence:

Sublinear : $\zeta_k \to 0$, but $\zeta_{k+1}/\zeta_k \to 1$.

- Example: $\zeta_k \leq C/k^q$, for $q > 0$ and $C > 0$
- To achieve an ε error $k \sim \mathcal{O}\left(1/\varepsilon^{1/q}\right)$.

Linear : $\zeta_{k+1}/\zeta_k \leq r$ for some $r \in (0,1)$.

- Example: $\zeta_k \leq Cq^k$, for $q \in (0,1)$ and $C > 0$
- To achieve an ε error, $k \sim \mathcal{O}\left(\log(1/\varepsilon)\right)$.

Superlinear : $\zeta_{k+1}/\zeta_k \to 0$.

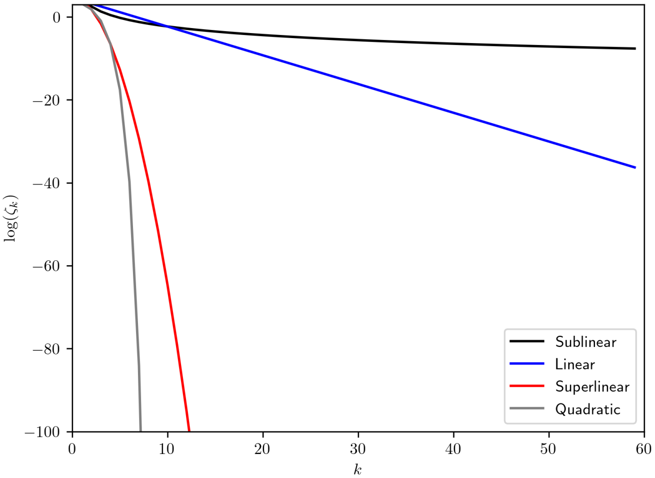
- Example: $\zeta_k \leq Cq^{k^2}$, for $q \in (0,1)$ and $C > 0$
- To achieve an ε error, $k \sim \mathcal{O}\left(\sqrt{\log(1/\varepsilon)}\right)$.

Quadratic : $\zeta_{k+1} \leq \zeta_k^2$

- Example: $\zeta_k \leq Cq^{2^k}$, for $q \in (0,1)$ and $C > 0$
- To achieve an ε error, $k \sim \mathcal{O}\left(\log(\log(1/\varepsilon))\right)$.

SSR

Rates of convergence



Example 3.3

Calculate the minimum number of iterations in order to reach an accuracy $\varepsilon = 10^{-5}$ in the following cases:

Example 3.3

Calculate the minimum number of iterations in order to reach an accuracy $\varepsilon = 10^{-5}$ in the following cases:

1. $\zeta_k = \frac{1}{k^2}$. In this case,
   $\frac{1}{k^2} \leq \varepsilon \rightarrow k \geq \sqrt{\frac{1}{\varepsilon}} = 316.28 \rightarrow k = 317.$

Example 3.3

Calculate the minimum number of iterations in order to reach an accuracy $\varepsilon = 10^{-5}$ in the following cases:

1. $\zeta_k = \frac{1}{k^2}$. In this case,
   $\frac{1}{k^2} \leq \varepsilon \to k \geq \sqrt{\frac{1}{\varepsilon}} = 316.28 \to k = 317.$
2. $\zeta_k = e^{-2k}$. In this case,
   $e^{-2k} \leq \varepsilon \to k \geq \frac{1}{2} \ln \frac{1}{\varepsilon} = 5.36 \to k = 6.$

Example 3.3

Calculate the minimum number of iterations in order to reach an accuracy $\varepsilon = 10^{-5}$ in the following cases:

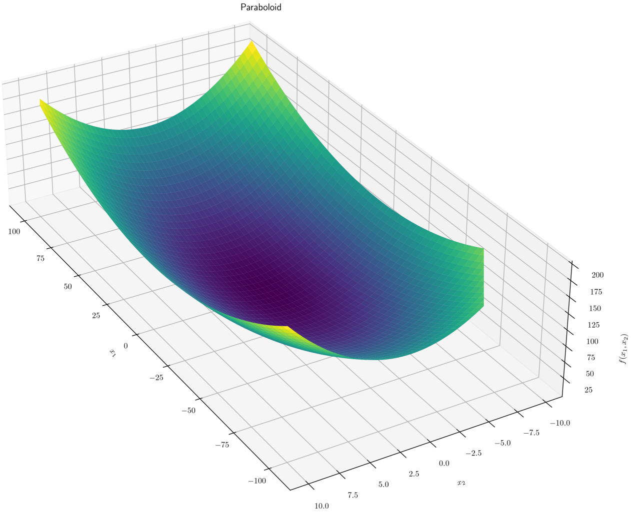
1. $\zeta_k = \frac{1}{k^2}$. In this case,
   $\frac{1}{k^2} \leq \varepsilon \rightarrow k \geq \sqrt{\frac{1}{\varepsilon}} = 316.28 \rightarrow k = 317.$
2. $\zeta_k = e^{-2k}$. In this case,
   $e^{-2k} \leq \varepsilon \rightarrow k \geq \frac{1}{2} \ln \frac{1}{\varepsilon} = 5.36 \rightarrow k = 6.$
3. $\zeta_k = e^{-2k^2}$. In this case,
   $e^{-2k^2} \leq \varepsilon \rightarrow k \geq \sqrt{\frac{\ln \frac{1}{\varepsilon}}{2}} = 2.4 \rightarrow k = 3.$

The effect of strong-convexity: convergence rates

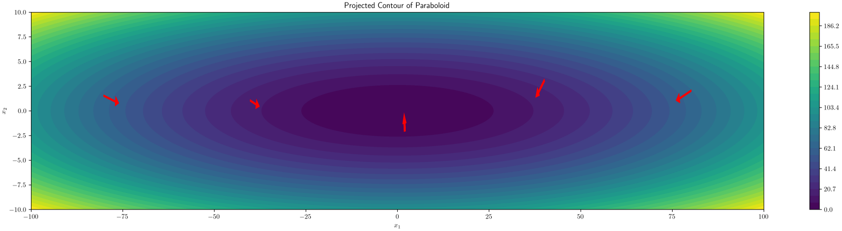
Some theoretical results [1]:

Gradient Alg.	Bound (Upper / Lower)	Rate
Conv $\eta = \frac{1}{L}$	$f(x_k) - f(x^*) \leq \frac{2L}{k+4} \left\|x_0 - x^*\right\|_2^2$	$\mathcal{O}(1/k)$
Conv $\eta = \frac{1}{L}$	$f(x_k) - f(x^*) \geq \frac{3L}{32(k+1)^2} \left\|x_0 - x^*\right\|_2^2$	$\mathcal{O}\left(1/k^2\right)$
Str. conv $\eta = \frac{2}{\mu+L}$	$f(x_k) - f(x^*) \leq \frac{L}{2} \left(\frac{\kappa-1}{\kappa+1}\right)^{2k} \left\|x_0 - x^*\right\|_2^2$	$\mathcal{O}\left(q^k\right)$
Str. conv $\eta = \frac{2}{\mu+L}$	$f(x_k) - f(x^*) \geq \frac{\mu}{2} \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2k} \left\|x_0 - x^*\right\|_2$	$\mathcal{O}\left(q^k\right)$

The effect of strong-convexity: condition number



The effect of strong-convexity: condition number



The effect of strong-convexity: example

Let us consider the Ridge problem

$$\arg\min_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w}) = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \left(\frac{1}{2}\left\|\mathbf{X}\mathbf{w}-\mathbf{y}\right\|_2^2 + \frac{\lambda}{2}\left\|\mathbf{w}\right\|_2^2\right)$$

where $\mathbf{X} \in \mathbb{R}^{n\times d}$ with i.i.d. zero mean unit variance Gaussian entries. The optimum $\mathbf{w}^* \sim \mathcal{N}(\mathbf{0},\mathbf{I}) \to \mathbf{y} = \mathbf{X}\mathbf{w}^*$.

The effect of strong-convexity: example

Let us consider the Ridge problem

$$\arg\min_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w}) = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \left(\frac{1}{2} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 \right)$$

where $\mathbf{X} \in \mathbb{R}^{n\times d}$ with i.i.d. zero mean unit variance Gaussian entries. The optimum $\mathbf{w}^* \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \rightarrow \mathbf{y} = \mathbf{X}\mathbf{w}^*$.

- Example 3.4: Run gradient descend for $\lambda = 0$
- Example 3.5: Run gradient descend for $\lambda = 0.04L$

The effect of strong-convexity: example

Let us consider the Ridge problem

$$\arg\min_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w}) = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \left(\frac{1}{2} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 \right)$$

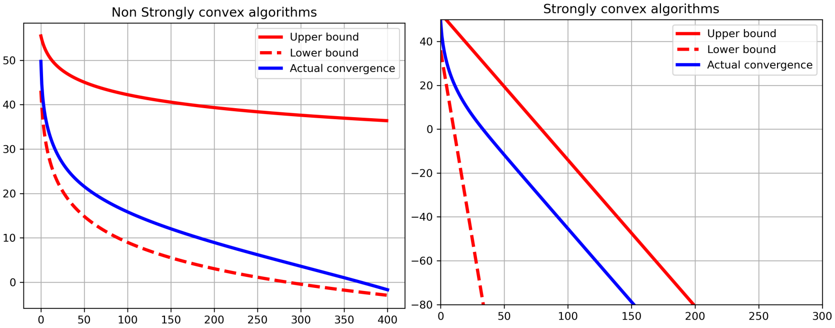
where $\mathbf{X} \in \mathbb{R}^{n\times d}$ with i.i.d. zero mean unit variance Gaussian entries. The optimum $\mathbf{w}^* \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \to \mathbf{y} = \mathbf{X}\mathbf{w}^*$.

- Example 3.4: Run gradient descend for $\lambda = 0$
- Example 3.5: Run gradient descend for $\lambda = 0.04L$

Plot the evolution of $f(\mathbf{w}) - f(\mathbf{w}*)$ in both cases and compare it with the theoretical results, assuming $\mathbf{w}_0 = 0$. Use $n = 400$ and $d = 500$.

Note: The function `bounds` contains the values for the theoretical bounds.

The effect of strong-convexity: results



Is the gradient method an optimal first-order method?

Is the gradient method an optimal first-order method?

- In the previous table, we have seen that there is a gap between the upper bounds of gradient-like methods and the best achievable performance (lower bound).

Is the gradient method an optimal first-order method?

- In the previous table, we have seen that there is a gap between the upper bounds of gradient-like methods and the best achievable performance (lower bound).
- We have also seen that the gradient does not always point to the optimum.

Is the gradient method an optimal first-order method?

- In the previous table, we have seen that there is a gap between the upper bounds of gradient-like methods and the best achievable performance (lower bound).
- We have also seen that the gradient does not always point to the optimum.
- There is a straightforward alternative update rule named accelerated method, or momentum method, that approaches the optimality.

Is the gradient method an optimal first-order method?

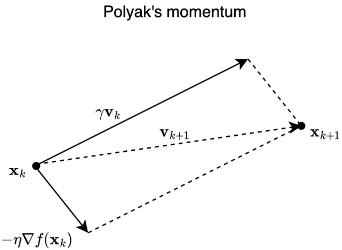
- In the previous table, we have seen that there is a gap between the upper bounds of gradient-like methods and the best achievable performance (lower bound).
- We have also seen that the gradient does not always point to the optimum.
- There is a straightforward alternative update rule named accelerated method, or momentum method, that approaches the optimality.
- The alternative update rule looks as follows:

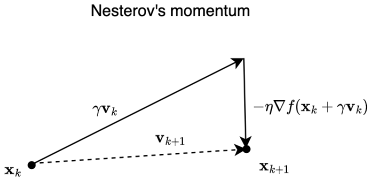
$$\begin{aligned} \mathbf{v}_{k+1} &= \gamma \mathbf{v}_k - \eta \nabla f(\mathbf{z}_k) \\ \mathbf{x}_{k+1} &= \mathbf{x}_k + \mathbf{v}_{k+1} \end{aligned}$$

where:

Accelerated gradient descend

- Polyak's momentum: $\mathbf{z}_k = \mathbf{x}_k$
- Nesterov's momentum: $\mathbf{z}_k = \mathbf{x}_k + \gamma \mathbf{v}_k$





Accelerated gradient descend

The key idea in accelerated methods is the addition of a **momentum term**, whereby the next iterate $\mathbf{x}_{k+1}$ depends not only on the gradient and previous point $\mathbf{x}_k$ but also on the point previous to that, $\mathbf{x}_{k-1}$.

If parameters L, μ are known, we use $\eta = \frac{1}{L}$ and $\gamma = \frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}$

If L, μ are not known, η can be determined as a constant step-size or applying a line search procedure, and $\gamma_k = \frac{k-2}{k+1}$ (making it dependent on the iteration).

Convergence rates. Overview

The upper bounds of accelerated methods are much closer to the optimum bounds:

Gradient Alg.	Upper Bound	Rate
Acc. Conv. $\eta = \frac{1}{L}$ $\gamma_k = g\left(\gamma_{k-1}\right)$	$f\left(x_k\right) - f\left(x^*\right) \leq \frac{4L}{(k+2)^2} \left\|x_0 - x^*\right\|_2^2$	$\mathcal{O}\left(1/k^2\right)$
Conv $\eta = \frac{1}{L}$	$f\left(x_k\right) - f\left(x^*\right) \leq \frac{2L}{k+4} \left\|x_0 - x^*\right\|_2^2$	$\mathcal{O}\left(1/k\right)$
Acc. Str.conv. $\eta = \frac{1}{L}$ $\gamma = \frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}$	$f\left(x_k\right) - f\left(x^*\right) \leq L\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}}\right)^k \left\|x_0 - x^*\right\|_2^2$	$\mathcal{O}\left(c^k\right)$
Str. conv $\eta = \frac{2}{\mu+L}$	$f\left(x_k\right) - f\left(x^*\right) \leq \frac{L}{2}\left(\frac{\kappa-1}{\kappa+1}\right)^{2k} \left\|x_0 - x^*\right\|_2^2$	$\mathcal{O}\left(q^k\right)$

Notice that in general $c = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}} < q = \left(\frac{\kappa-1}{\kappa+1}\right)^2$

Example 3.6

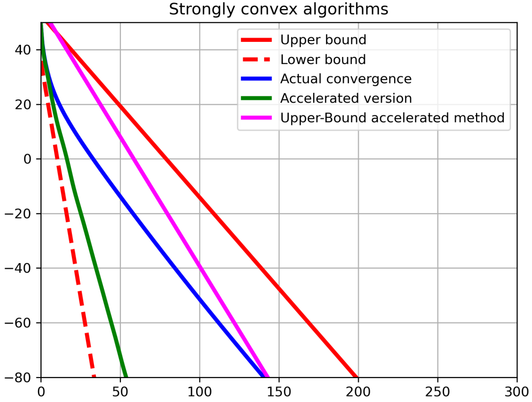
Let us consider the Ridge problem again

$$\arg\min_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w}) = \arg\min_{\mathbf{w}\in\mathbb{R}^d} \left(\frac{1}{2} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 \right)$$

Plot the evolution of $f(\mathbf{w}) - f(\mathbf{w}*)$ adding the curves corresponding to the upper bound of the accelerated method given and the implementation of the accelerated algorithm. Use the following parameters:

$$\eta = \frac{1}{L} \quad \gamma = \frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}$$

Example 3.6: result



You can notice that the simulated result is very close to the optimum performance.

Non smooth functions. Subgradient methods

Let us recall that if the function is continuous and convex, then $f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla^T f(\mathbf{x})(\mathbf{y}-\mathbf{x})$.

If the function is non-smooth, we use a different concept: the subdifferential denoted as $\partial f(\mathbf{x})$ is a set of vectors such that $f(\mathbf{y}) \geq f(\mathbf{x}) + \partial^T f(\mathbf{x})(\mathbf{y}-\mathbf{x})$.

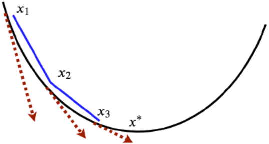
Non smooth functions. Subgradient methods

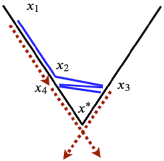
Let us recall that if the function is continuous and convex, then $f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla^T f(\mathbf{x})(\mathbf{y} - \mathbf{x})$.

If the function is non-smooth, we use a different concept: the subdifferential denoted as $\partial f(\mathbf{x})$ is a set of vectors such that $f(\mathbf{y}) \geq f(\mathbf{x}) + \partial^T f(\mathbf{x})(\mathbf{y} - \mathbf{x})$.

Consequences:

- $\mathbf{x}^*$ is a minimizer if and only if 0 is a subgradient of f at $\mathbf{x}^*$.
- We have lost the “self-tuning” property of the gradient.
- The subgradient method is not a descent method!





Non smooth functions. LASSO

However, if a decreasing step is used, the following update converges:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta_k \mathbf{g}_k$$

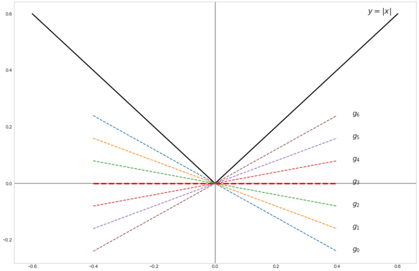
where η_k is the step size and $\mathbf{g} \in \partial f\left(\mathbf{x}\right)$. In fact, the analysis of convergence reveals that optimum $\eta_k \sim \frac{1}{\sqrt{k+1}}$.

Non smooth functions. LASSO

However, if a decreasing step is used, the following update converges:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta_k \mathbf{g}_k$$

where η_k is the step size and $\mathbf{g} \in \partial f(\mathbf{x})$. In fact, the analysis of convergence reveals that optimum $\eta_k \sim \frac{1}{\sqrt{k+1}}$.



LASSO: The subgradient of the absolute value is:

$$\partial |x| = \begin{cases} 1 & x > 0 \\ -1 & x < 0 \\ [-1, 1] & x = 0 \end{cases}$$

Take a look at some loss functions in ML

	Function	gradient or subgrad.
Hinge loss	$\max\left\{0, 1 - y\mathbf{x}^T\mathbf{w}\right\} \qquad y \in \{\pm 1\}$	$\begin{array}{ll} -y\mathbf{x} & \text{if} \quad y\mathbf{x}^T\mathbf{w} < 1 \\ 0 & \text{otherwise} \end{array}$
Logistic loss	$\ln\left(1 + \exp\left(-y\mathbf{x}^T\mathbf{w}\right)\right) \ y \in \{\pm 1\}$	$-y\left(\frac{1}{1+\exp(y\mathbf{x}^T\mathbf{w})}\right)\mathbf{x}$
Square loss	$\frac{1}{2}\left(\mathbf{x}^T\mathbf{w} - y\right)^2$	$\left(\mathbf{x}^T\mathbf{w} - y\right)\mathbf{x}$
L2 reg.	$\frac{1}{2}\left\Vert\mathbf{w}\right\Vert_2^2$	$\mathbf{w}$
L1 reg.	$\left\Vert\mathbf{w}\right\Vert_1$	$\operatorname{sgn}(\mathbf{w})$

Non smooth functions. Convergence rates

Regarding convergence rates, it can be shown that:

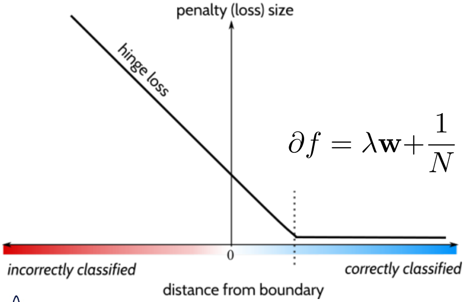
1. If $f(x)$ is only known to be convex, using the subgradient descent method, the convergence rate is $\mathcal{O}\left(1/\sqrt{k}\right)$, i.e., to achieve ε accuracy, we need $\mathcal{O}\left(1/\varepsilon^2\right)$ iterations.
2. If $f(x)$ is known to be strongly convex, using the subgradient descent method, the convergence rate is $\mathcal{O}(1/k)$, i.e., to achieve ε accuracy, we need $\mathcal{O}(1/\varepsilon)$ iterations.

Note that subgradient methods are not very competitive methods either in terms of convergence rate or in terms of convergence level.

Case study 3.1. SVM

`Calculation_subgrad_svm`: Recall how the hinge function looks like.

$$\underset{\mathbf{w}\in\mathbb{R}^{d+1}}{\arg\min}\left(\frac{1}{n}\sum_{i=1}^{n}\max\left(1-y_i\left(\mathbf{w}^T\mathbf{x}_i\right),0\right)^p+\frac{\lambda}{2}\|\mathbf{w}\|_2^2\right)$$



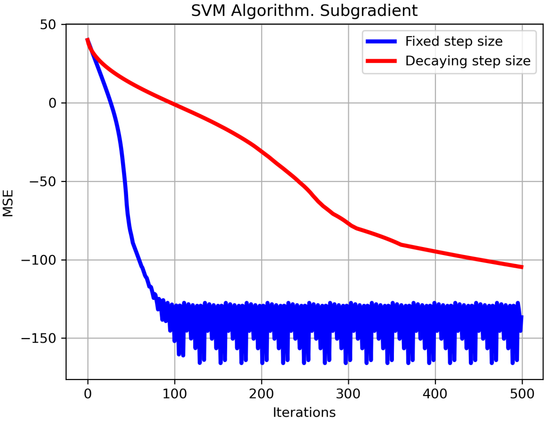
$$\partial f=\lambda\mathbf{w}+\frac{1}{N}\sum_{i=1}^{n}\begin{cases}0 & if\ y_i\mathbf{w}^T\mathbf{x}_i>1\\ -y_i\mathbf{x}_i & if\ y_i\mathbf{w}^T\mathbf{x}_i<1\end{cases}$$

Case study 3.1: Results

Plot the evolution of $f(\mathbf{w}) - f(\mathbf{w}^*)$ for the subgradient method for constant learning rate and for $\eta_k = \frac{1}{\sqrt{k+1}}$.

Case study 3.1: Results

Plot the evolution of $f(\mathbf{w}) - f(\mathbf{w}^*)$ for the subgradient method for constant learning rate and for $\eta_k = \frac{1}{\sqrt{k+1}}$.



Proximal operator

Can we do better for non-smooth convex functions?

Proximal operator

Can we do better for non-smooth convex functions?

The essential reason for the slow convergence of those functions is because there are plenty of subgradients that are large near and even at the solution

Proximal operator

Can we do better for non-smooth convex functions?

The essential reason for the slow convergence of those functions is because there are plenty of subgradients that are large near and even at the solution

The proximal operator solves this problem by adding a smooth regularization term:

$$\mathrm{Prox}_{\eta_k f}(\mathbf{z}) = \arg\min_{\mathbf{x}\in\mathbb{R}^d} \left(f(\mathbf{x}) + \frac{1}{2\eta_k}\|\mathbf{x}-\mathbf{z}\|_2^2 \right)$$

Note that for a convex function $f(\cdot)$, $\mathrm{Prox}_{\eta_k f}(\cdot)$ is strictly convex for $1/2\eta_k > 0$.

Example 3.7

$f(\mathbf{x}) = \lambda \|\mathbf{x}\|_1 \rightarrow \mathrm{Prox}_{\eta f}(\mathbf{z}) = \arg\min_{\mathbf{x}} \left(\frac{1}{2\eta} \|\mathbf{x} - \mathbf{z}\|_2^2 + \lambda \|\mathbf{x}\|_1 \right)$ which is separable in indexes. So, we can optimize separately obtaining for the j-coordinate:

$$\mathrm{Prox}_{\eta f}(z_j) = \arg\min_{x} \left(\frac{1}{2\eta} (x - z_j)^2 + \lambda |x| \right)$$

If we take derivatives we have: $0 \in \frac{1}{\eta}(x - z_j) + \lambda \partial |x|$, so we have $x_j = z_j - \lambda\eta \,\mathrm{sgn}(x_j)$ whose solution is:

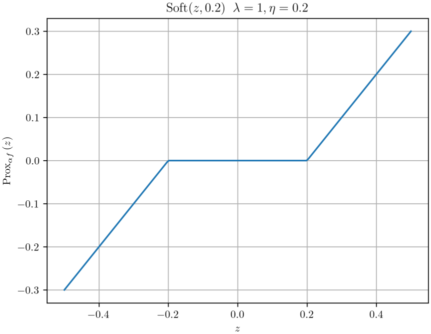
$$\begin{array}{cc} x_j = z_j - \lambda\eta & z_j > \lambda\eta \\ x_j = z_j + \lambda\eta & z_j < -\lambda\eta \\ x_j = 0 & |z_j| \leq \lambda\eta \end{array}$$

Example 3.7

This is usually expressed in a compact way as
$x_j = \mathrm{Soft}\,(a, b) = (a - b)^+ - (-a - b)^+$ where in this case $a = z_j$ and $b = \lambda\eta$ and $(\bullet)^+ = \max\{0, \bullet\}$. Therefore, we have:

$$\mathrm{Prox}_{\eta f}\,(z_j) = \mathrm{Soft}\,(z_j, \lambda\eta)$$

This function is known as *Soft Thresholding* operator and is shown in the next figure.



Example 3.8

Calculate

$$\mathrm{Prox}_{\eta f}\left(\mathbf{z}\right) = \arg\min_{\mathbf{x}} \left(\frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c + \frac{1}{2\eta}\left\|\mathbf{x}-\mathbf{z}\right\|_2^2 \right)$$

Example 3.8

Calculate
$\text{Prox}_{\eta f}(\mathbf{z}) = \arg\min_{\mathbf{x}} \left(\frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c + \frac{1}{2\eta}\|\mathbf{x}-\mathbf{z}\|_2^2 \right)$

The Proximal operator of a Quadratic problem is defined as:

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c$$

Taking derivatives: $\mathbf{A}\mathbf{x} + \mathbf{b} + \frac{1}{\eta}(\mathbf{x}-\mathbf{z}) = 0$ we get
$\mathbf{x} = \left(\mathbf{A} + \frac{1}{\eta}\mathbf{I}\right)^{-1}\left(\frac{1}{\eta}\mathbf{z} - \mathbf{b}\right)$. So, we have:

$$\text{Prox}_{\eta f}(\mathbf{z}) = \left(\mathbf{A} + \frac{1}{\eta}\mathbf{I}\right)^{-1}\left(\frac{1}{\eta}\mathbf{z} - \mathbf{b}\right)$$

Proximal gradient

$$\mathbf{x}_{k+1} = \mathrm{Prox}_{\eta f}\left(\mathbf{x}_k\right) = \arg\min_{\mathbf{x}\in\mathbb{R}^d} \left(f(\mathbf{x}) + \frac{1}{2\eta}\|\mathbf{x}-\mathbf{x}_k\|_2^2 \right)$$

If $f(\cdot)$ is differentiable, this is equivalent to gradient descent.

Proximal gradient

$$\mathbf{x}_{k+1} = \mathrm{Prox}_{\eta f}\left(\mathbf{x}_k\right) = \arg\min_{\mathbf{x}\in\mathbb{R}^d} \left(f(\mathbf{x}) + \frac{1}{2\eta}\|\mathbf{x} - \mathbf{x}_k\|_2^2 \right)$$

If $f(\cdot)$ is differentiable, this is equivalent to gradient descent.

Composite functions: Consider an objective function broken into two parts:

$$f(\mathbf{x}) = g(\mathbf{x}) + h(\mathbf{x})$$

where both g and h are convex, but g is smooth and h is a non-smooth function with an easy-to-evaluate Prox.

Proximal gradient

$$\mathbf{x}_{k+1} = \text{Prox}_{\eta f}\left(\mathbf{x}_k\right) = \arg\min_{\mathbf{x}\in\mathbb{R}^d}\left(f(\mathbf{x}) + \frac{1}{2\eta}\|\mathbf{x} - \mathbf{x}_k\|_2^2\right)$$

If $f(\cdot)$ is differentiable, this is equivalent to gradient descent.

Composite functions: Consider an objective function broken into two parts:

$$f(\mathbf{x}) = g(\mathbf{x}) + h(\mathbf{x})$$

where both g and h are convex, but g is smooth and h is a non-smooth function with an easy-to-evaluate Prox.

The Proximal gradient, in this case, is equal to [2]:

$$\begin{aligned}\mathbf{x}_{k+1} &= \text{Prox}_{\eta h}\left(\mathbf{x}_k - \eta\nabla g(\mathbf{x}_k)\right)\\ &= \arg\min_{\mathbf{x}\in\mathbb{R}^d}\left(h(\mathbf{x}) + \frac{1}{2\eta}\|\mathbf{x} - \mathbf{x}_k + \eta\nabla g(\mathbf{x}_k)\|_2^2\right)\end{aligned}$$

Iterative Soft-thresholding algorithm (ISTA)

The LASSO case:

Iterative Soft-thresholding algorithm (ISTA)

The LASSO case:

$$\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \left(\frac{1}{n} \left\| \mathbf{X}\mathbf{w} - \mathbf{y} \right\|_2^2 + \lambda \left\| \mathbf{w} \right\|_1 \right)$$

Iterative Soft-thresholding algorithm (ISTA)

The LASSO case:

$$\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \left(\frac{1}{n} \left\| \mathbf{X}\mathbf{w} - \mathbf{y} \right\|_2^2 + \lambda \left\| \mathbf{w} \right\|_1 \right)$$

We take $g(\mathbf{w}) = \frac{1}{n} \left\| \mathbf{X}\mathbf{w} - \mathbf{y} \right\|_2^2$ and $h(\mathbf{w}) = \lambda \left\| \mathbf{w} \right\|_1$

Iterative Soft-thresholding algorithm (ISTA)

The LASSO case:

$$\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \left(\frac{1}{n}\left\|\mathbf{X}\mathbf{w}-\mathbf{y}\right\|_2^2 + \lambda\left\|\mathbf{w}\right\|_1\right)$$

We take $g(\mathbf{w}) = \frac{1}{n}\left\|\mathbf{X}\mathbf{w}-\mathbf{y}\right\|_2^2$ and $h(\mathbf{w}) = \lambda\left\|\mathbf{w}\right\|_1$

Defining the residual vector $r_k = (\mathbf{w}_k - \eta\nabla g(\mathbf{w}_k))$ with $\nabla g(\mathbf{w}_k) = \mathbf{X}^T(\mathbf{X}\mathbf{w}_k - \mathbf{y})$, the problem can be formulated component-wise:

$$w_{k+1,j} = \arg\min_{w_j}\left(\frac{1}{2\eta}(w_j - r_{k,j})^2 + \lambda|w_j|\right)$$

Iterative Soft-thresholding algorithm (ISTA)

The LASSO case:

$$\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \left(\frac{1}{n}\|\mathbf{X}\mathbf{w}-\mathbf{y}\|_2^2+\lambda\|\mathbf{w}\|_1\right)$$

We take $g(\mathbf{w})=\frac{1}{n}\|\mathbf{X}\mathbf{w}-\mathbf{y}\|_2^2$ and $h(\mathbf{w})=\lambda\|\mathbf{w}\|_1$

Defining the residual vector $r_k=(\mathbf{w}_k-\eta\nabla g(\mathbf{w}_k))$ with $\nabla g(\mathbf{w}_k)=\mathbf{X}^T(\mathbf{X}\mathbf{w}_k-\mathbf{y})$, the problem can be formulated component-wise:

$$\begin{aligned} w_{k+1,j} &= \arg\min_{w_j}\left(\frac{1}{2\eta}(w_j-r_{k,j})^2+\lambda|w_j|\right)\\ &= \mathrm{Soft}\left(r_{k,j,\lambda\eta}\right)\end{aligned}$$

Iterative Soft-thresholding algorithm (ISTA)

The LASSO case:

$$\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \left(\frac{1}{n} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_1 \right)$$

We take $g(\mathbf{w}) = \frac{1}{n} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$ and $h(\mathbf{w}) = \lambda \|\mathbf{w}\|_1$

Defining the residual vector $r_k = (\mathbf{w}_k - \eta \nabla g(\mathbf{w}_k))$ with $\nabla g(\mathbf{w}_k) = \mathbf{X}^T(\mathbf{X}\mathbf{w}_k - \mathbf{y})$, the problem can be formulated component-wise:

$$\begin{aligned} w_{k+1,j} &= \arg\min_{w_j} \left(\frac{1}{2\eta}(w_j - r_{k,j})^2 + \lambda |w_j| \right) \\ &= \mathrm{Soft}\left(r_{k,j,\lambda\eta}\right) \end{aligned}$$

Convergence rate increases to $\mathcal{O}\left(1/k\right)$!

Fast ISTA

The FISTA algorithm is essentially the same procedure, including the momentum term (there are several possible implementations):

$$\mathbf{w}_{k+1} = \arg\min_{\mathbf{w}} \left(\frac{1}{2\eta} \left\| \mathbf{w} - \left(\mathbf{v}_k - \eta_k \nabla g\left(\mathbf{v}_k\right) \right) \right\|_2^2 + \lambda \left\| \mathbf{w} \right\|_1 \right)$$

$$\mathbf{v}_{k+1} = \mathbf{w}_k + \frac{k-2}{k+1} \left(\mathbf{w}_{k+1} - \mathbf{w}_k \right)$$

Fast ISTA

The FISTA algorithm is essentially the same procedure, including the momentum term (there are several possible implementations):

$$\mathbf{w}_{k+1} = \arg\min_{\mathbf{w}} \left(\frac{1}{2\eta} \left\| \mathbf{w} - \left(\mathbf{v}_k - \eta_k \nabla g\left(\mathbf{v}_k\right)\right)\right\|_2^2 + \lambda \left\|\mathbf{w}\right\|_1 \right)$$
$$\mathbf{v}_{k+1} = \mathbf{w}_k + \frac{k-2}{k+1}\left(\mathbf{w}_{k+1} - \mathbf{w}_k\right)$$

Defining a new residual $r_k = \left(\mathbf{v}_k - \eta_k \nabla g\left(\mathbf{v}_k\right)\right)$, we reach the final compact expression also component-wise:

$$w_{k+1,j} = \mathrm{Soft}\left(r_{k,j}, \lambda\eta\right) \quad \forall j$$
$$v_{k+1,j} = w_{k+1,j} + \frac{k-2}{k+1}\left(w_{k+1,j} - w_{k,j}\right)$$

Fast ISTA

The FISTA algorithm is essentially the same procedure, including the momentum term (there are several possible implementations):

$$\mathbf{w}_{k+1} = \arg\min_{\mathbf{w}} \left(\frac{1}{2\eta} \left\| \mathbf{w} - \left(\mathbf{v}_k - \eta_k \nabla g \left(\mathbf{v}_k \right) \right) \right\|_2^2 + \lambda \left\| \mathbf{w} \right\|_1 \right)$$
$$\mathbf{v}_{k+1} = \mathbf{w}_k + \frac{k-2}{k+1} \left(\mathbf{w}_{k+1} - \mathbf{w}_k \right)$$

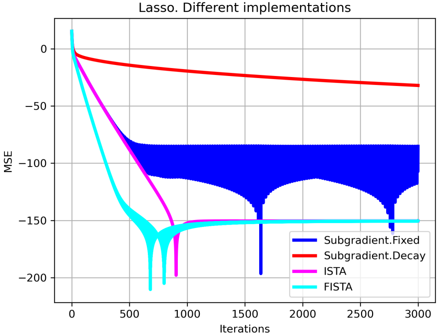
Defining a new residual $r_k = \left(\mathbf{v}_k - \eta_k \nabla g \left(\mathbf{v}_k \right) \right)$, we reach the final compact expression also component-wise:

$$w_{k+1,j} = \mathrm{Soft} \left(r_{k,j}, \lambda\eta \right) \quad \forall j$$
$$v_{k+1,j} = w_{k+1,j} + \frac{k-2}{k+1} \left(w_{k+1,j} - w_{k,j} \right)$$

Convergence rate increases to $\mathcal{O}\left(1/k^2\right)$!

Case study 3.2.

Implement ISTA and FISTA and compare with subgradient implementations. Complete the code provided in the notebook `case_study_3_2.ipynb`. Results should be similar to the following:



Before completing the code, take a look at the following functions included in the **utils** package:

- `ista_lasso`
- `fista_lasso`
- `prox_normL1`

Acknowledgments

I would like to acknowledge several sources I have used to create slides

- Martin Jaggi & Nicolas Flammarion's course at EPFL
  `https://github.com/epfml/OptML_course`
- Constantine Caramanis course at University of Texas
  `https://www.youtube.com/@constantine.caramanis`

Questions?

References

[1] Sébastien Bubeck et al. "Convex optimization: Algorithms and complexity". In: *Foundations and Trends® in Machine Learning* 8.3-4 (2015), pp. 231–357.

[2] M.A. Davenport, M.B. Egerstedt, and J. Romberg. *Proximal algorithms*. Tech. rep. Georgia Tech, 2021.

Thank You

Julián D. Arias-Londoño
julian.arias@upm.es