Gradient descent and accelerated methods - heavy ball method. Nesterov's accelerated method. Features of nonsmooth optimization. Subgradient method. Proximal gradient method. Newton's method and quasi-Newton's methods

Daniil Merkulov

Applied Math for Data Science. Sberuniversity.

Gradient Descent

Exact line search aka steepest descent

$$
\alpha_k = \text{arg}\min_{\alpha \in \mathbb{R}^+} f(x_{k+1}) = \text{arg}\min_{\alpha \in \mathbb{R}^+} f(x_k - \alpha \nabla f(x_k))
$$

More theoretical than practical approach. It also allows you to analyze the convergence, but often exact line search can be difficult if the function calculation takes too long or costs a lot. An interesting theoretical property of this method is that each following iteration is orthogonal to the previous one:

$$
\alpha_k = \text{arg}\min_{\alpha \in \mathbb{R}^+} f(x_k - \alpha \nabla f(x_k))
$$

Optimality conditions:

$$
\nabla f(x_{k+1})^\top \nabla f(x_k) = 0
$$

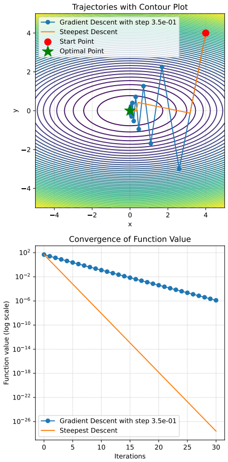


Figure 1: Steepest Descent

Open In Colab ♣

Strongly convex quadratics

Coordinate shift

Consider the following quadratic optimization problem:

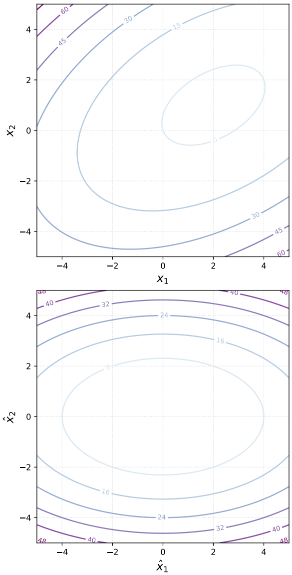
$$\min_{x \in \mathbb{R}^d} f(x) = \min_{x \in \mathbb{R}^d} \frac{1}{2} x^\top A x - b^\top x + c, \text{ where } A \in \mathbb{S}^d_{++}.$$

Coordinate shift

Consider the following quadratic optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x) = \min_{x \in \mathbb{R}^d} \frac{1}{2} x^\top A x - b^\top x + c, \text{ where } A \in \mathbb{S}^d_{++}.$$

- Firstly, without loss of generality we can set $c = 0$, which will or affect optimization process.



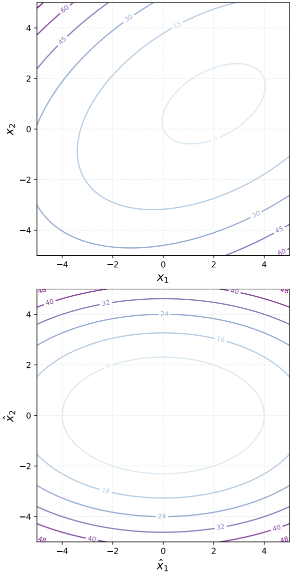
Coordinate shift

Consider the following quadratic optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x) = \min_{x \in \mathbb{R}^d} \frac{1}{2} x^\top A x - b^\top x + c, \text{ where } A \in \mathbb{S}^d_{++}.$$

- Firstly, without loss of generality we can set $c = 0$, which will or affect optimization process.
- Secondly, we have a spectral decomposition of the matrix A:

$$A = Q \Lambda Q^T$$



Coordinate shift

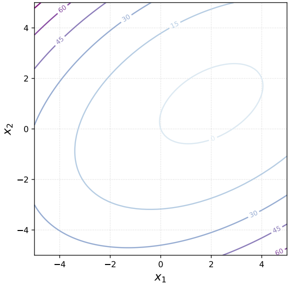
Consider the following quadratic optimization problem:

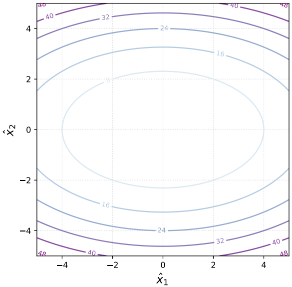
$$\min_{x \in \mathbb{R}^d} f(x) = \min_{x \in \mathbb{R}^d} \frac{1}{2} x^\top A x - b^\top x + c, \text{ where } A \in \mathbb{S}^d_{++}.$$

- Firstly, without loss of generality we can set $c = 0$, which will or affect optimization process.
- Secondly, we have a spectral decomposition of the matrix A:

$$A = Q \Lambda Q^T$$

- Let's show, that we can switch coordinates to make an analysis a little bit easier. Let $\hat{x} = Q^T(x - x^*)$, where x^* is the minimum point of initial function, defined by $Ax^* = b$. At the same time $x = Q\hat{x} + x^*$.





Coordinate shift

Consider the following quadratic optimization problem:

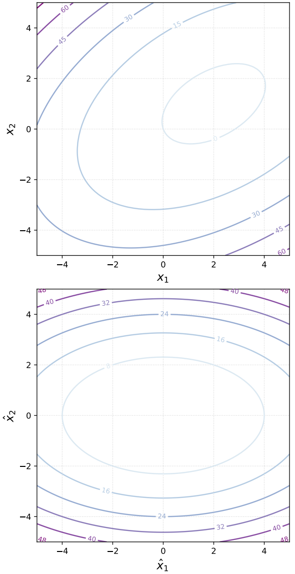
$$
\min_{x \in \mathbb{R}^d} f(x) = \min_{x \in \mathbb{R}^d} \frac{1}{2} x^\top A x - b^\top x + c, \text{ where } A \in \mathbb{S}^d_{++}.
$$

- Firstly, without loss of generality we can set $c = 0$, which will or affect optimization process.
- Secondly, we have a spectral decomposition of the matrix A:

$$
A = Q \Lambda Q^T
$$

- Let's show, that we can switch coordinates to make an analysis a little bit easier. Let $\hat{x} = Q^T(x - x^*)$, where x^* is the minimum point of initial function, defined by $Ax^* = b$. At the same time $x = Q\hat{x} + x^*$.

$$
f(\hat{x}) = \frac{1}{2}(Q\hat{x} + x^*)^\top A (Q\hat{x} + x^*) - b^\top (Q\hat{x} + x^*)
$$



Coordinate shift

Consider the following quadratic optimization problem:

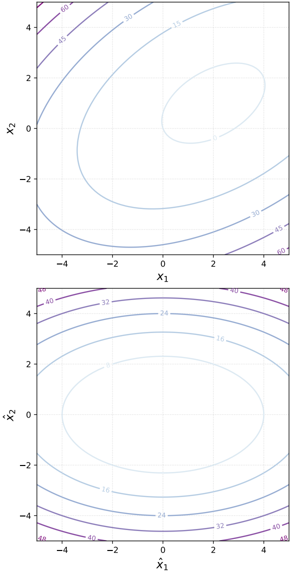
$$\min_{x \in \mathbb{R}^d} f(x) = \min_{x \in \mathbb{R}^d} \frac{1}{2} x^\top A x - b^\top x + c, \text{ where } A \in \mathbb{S}^d_{++}.$$

- Firstly, without loss of generality we can set $c = 0$, which will or affect optimization process.
- Secondly, we have a spectral decomposition of the matrix A:

$$A = Q \Lambda Q^T$$

- Let's show, that we can switch coordinates to make an analysis a little bit easier. Let $\hat{x} = Q^T(x - x^*)$, where x^* is the minimum point of initial function, defined by $Ax^* = b$. At the same time $x = Q\hat{x} + x^*$.

$$\begin{aligned} f(\hat{x}) &= \frac{1}{2}(Q\hat{x} + x^*)^\top A (Q\hat{x} + x^*) - b^\top (Q\hat{x} + x^*) \\ &= \frac{1}{2} \hat{x}^T Q^T A Q \hat{x} + (x^*)^T A Q \hat{x} + \frac{1}{2}(x^*)^T A (x^*)^T - b^T Q \hat{x} - b^T x^* \end{aligned}$$



Coordinate shift

Consider the following quadratic optimization problem:

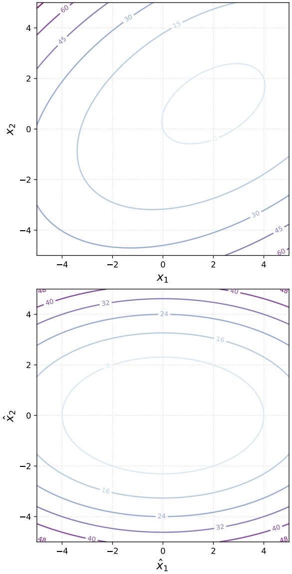
$$\min_{x \in \mathbb{R}^d} f(x) = \min_{x \in \mathbb{R}^d} \frac{1}{2} x^\top A x - b^\top x + c, \text{ where } A \in \mathbb{S}^d_{++}.$$

- Firstly, without loss of generality we can set $c = 0$, which will or affect optimization process.
- Secondly, we have a spectral decomposition of the matrix A:

$$A = Q \Lambda Q^T$$

- Let's show, that we can switch coordinates to make an analysis a little bit easier. Let $\hat{x} = Q^T(x - x^*)$, where x^* is the minimum point of initial function, defined by $Ax^* = b$. At the same time $x = Q\hat{x} + x^*$.

$$\begin{aligned} f(\hat{x}) &= \frac{1}{2}(Q\hat{x} + x^*)^\top A (Q\hat{x} + x^*) - b^\top (Q\hat{x} + x^*) \\ &= \frac{1}{2}\hat{x}^T Q^T A Q \hat{x} + (x^*)^T A Q \hat{x} + \frac{1}{2}(x^*)^T A (x^*)^T - b^T Q \hat{x} - b^T x^* \\ &= \frac{1}{2}\hat{x}^T \Lambda \hat{x} \end{aligned}$$



Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T \Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k)$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T\Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T\Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k \\
&= (I - \alpha^k \Lambda) x^k
\end{aligned}
$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T \Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k \\
&= (I - \alpha^k \Lambda) x^k \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)}) x^k_{(i)} \text{ For } i\text{-th coordinate}
\end{aligned}
$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T\Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k \\
&= (I - \alpha^k \Lambda) x^k \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)}) x^k_{(i)} \text{ For } i\text{-th coordinate} \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}
\end{aligned}
$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T\Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k \\
&= (I - \alpha^k \Lambda) x^k \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)}) x^k_{(i)} \text{ For } i\text{-th coordinate} \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}
\end{aligned}
$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:

$$
\rho(\alpha) = \max_i |1 - \alpha \lambda_{(i)}| < 1
$$

Remember, that $\lambda_{\text{min}} = \mu > 0, \lambda_{\text{max}} = L \geq \mu$.

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T\Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k \\
&= (I - \alpha^k \Lambda) x^k \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)}) x^k_{(i)} \text{ For } i\text{-th coordinate} \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}
\end{aligned}
$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:

$$
\rho(\alpha) = \max_i |1 - \alpha\lambda_{(i)}| < 1
$$

Remember, that $\lambda_{\text{min}} = \mu > 0, \lambda_{\text{max}} = L \geq \mu$.

$$
|1 - \alpha\mu| < 1
$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T\Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k \\
&= (I - \alpha^k \Lambda) x^k \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)}) x^k_{(i)} \text{ For } i\text{-th coordinate} \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}
\end{aligned}
$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:

$$
\rho(\alpha) = \max_i |1 - \alpha \lambda_{(i)}| < 1
$$

Remember, that $\lambda_{\text{min}} = \mu > 0, \lambda_{\text{max}} = L \geq \mu$.

$$
\begin{aligned}
&|1 - \alpha\mu| < 1 \\
&-1 < 1 - \alpha\mu < 1
\end{aligned}
$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T\Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k \\
&= (I - \alpha^k \Lambda) x^k \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)}) x^k_{(i)} \text{ For } i\text{-th coordinate} \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}
\end{aligned}
$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:

$$
\rho(\alpha) = \max_i |1 - \alpha \lambda_{(i)}| < 1
$$

Remember, that $\lambda_{\text{min}} = \mu > 0, \lambda_{\text{max}} = L \geq \mu$.

$$
\begin{aligned}
|1 - \alpha \mu| &< 1 \\
-1 < 1 &- \alpha \mu < 1 \\
\alpha < \frac{2}{\mu} \quad\quad & \alpha \mu > 0
\end{aligned}
$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T\Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k \\
&= (I - \alpha^k \Lambda) x^k \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)}) x^k_{(i)} \text{ For } i\text{-th coordinate} \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}
\end{aligned}
$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:

$$
\rho(\alpha) = \max_i |1 - \alpha\lambda_{(i)}| < 1
$$

Remember, that $\lambda_{\text{min}} = \mu > 0, \lambda_{\text{max}} = L \geq \mu$.

$$
\begin{aligned}
|1 - \alpha\mu| &< 1 \\
-1 < 1 - \alpha\mu &< 1 \\
\alpha < \frac{2}{\mu} \qquad & \alpha\mu > 0
\end{aligned}
$$

$$
|1 - \alpha L| < 1
$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T\Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k \\
&= (I - \alpha^k \Lambda) x^k \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)}) x^k_{(i)} \text{ For } i\text{-th coordinate} \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}
\end{aligned}
$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:

$$
\rho(\alpha) = \max_i |1 - \alpha \lambda_{(i)}| < 1
$$

Remember, that $\lambda_{\text{min}} = \mu > 0, \lambda_{\text{max}} = L \geq \mu$.

$$
\begin{aligned}
|1 - \alpha\mu| &< 1 & \qquad |1 - \alpha L| &< 1 \\
-1 < 1 - \alpha\mu &< 1 & \qquad -1 < 1 - \alpha L &< 1 \\
\alpha < \frac{2}{\mu} \qquad \alpha\mu &> 0 & &
\end{aligned}
$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T\Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k \\
&= (I - \alpha^k \Lambda) x^k \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)}) x^k_{(i)} \text{ For } i\text{-th coordinate} \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}
\end{aligned}
$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:

$$
\rho(\alpha) = \max_i |1 - \alpha\lambda_{(i)}| < 1
$$

Remember, that $\lambda_{\text{min}} = \mu > 0, \lambda_{\text{max}} = L \geq \mu$.

$$
\begin{aligned}
&|1 - \alpha\mu| < 1 && |1 - \alpha L| < 1 \\
&-1 < 1 - \alpha\mu < 1 && -1 < 1 - \alpha L < 1 \\
&\alpha < \frac{2}{\mu} \quad \alpha\mu > 0 && \alpha < \frac{2}{L} \quad \alpha L > 0
\end{aligned}
$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T\Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k \\
&= (I - \alpha^k \Lambda) x^k \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)}) x^k_{(i)} \text{ For } i\text{-th coordinate} \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}
\end{aligned}
$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:

$$
\rho(\alpha) = \max_i |1 - \alpha \lambda_{(i)}| < 1
$$

Remember, that $\lambda_{\text{min}} = \mu > 0, \lambda_{\text{max}} = L \geq \mu$.

$$
\begin{aligned}
&|1 - \alpha\mu| < 1 \\
&-1 < 1 - \alpha\mu < 1 \\
&\alpha < \frac{2}{\mu} \quad\quad \alpha\mu > 0
\end{aligned}
\qquad\qquad
\begin{aligned}
&|1 - \alpha L| < 1 \\
&-1 < 1 - \alpha L < 1 \\
&\alpha < \frac{2}{L} \quad\quad \alpha L > 0
\end{aligned}
$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T\Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k \\
&= (I - \alpha^k \Lambda) x^k \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)}) x^k_{(i)} \text{ For } i\text{-th coordinate} \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}
\end{aligned}
$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:

$$
\rho(\alpha) = \max_i |1 - \alpha \lambda_{(i)}| < 1
$$

Remember, that $\lambda_{\text{min}} = \mu > 0, \lambda_{\text{max}} = L \geq \mu$.

$$
\begin{aligned}
|1 - \alpha \mu| &< 1 & |1 - \alpha L| &< 1 \\
-1 < 1 - \alpha \mu &< 1 & -1 < 1 - \alpha L &< 1 \\
\alpha < \frac{2}{\mu} \quad & \quad \alpha \mu > 0 & \alpha < \frac{2}{L} \quad & \quad \alpha L > 0
\end{aligned}
$$

$\alpha < \frac{2}{L}$ is needed for convergence.

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T\Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k \\
&= (I - \alpha^k \Lambda) x^k \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)}) x^k_{(i)} \text{ For } i\text{-th coordinate} \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}
\end{aligned}
$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:

$$
\rho(\alpha) = \max_i |1 - \alpha\lambda_{(i)}| < 1
$$

Remember, that $\lambda_{\text{min}} = \mu > 0, \lambda_{\text{max}} = L \geq \mu$.

$$
\begin{aligned}
|1 - \alpha\mu| &< 1 \\
-1 < 1 - \alpha\mu &< 1 \\
\alpha < \frac{2}{\mu} \quad & \quad \alpha\mu > 0
\end{aligned}
\qquad\qquad
\begin{aligned}
|1 - \alpha L| &< 1 \\
-1 < 1 - \alpha L &< 1 \\
\alpha < \frac{2}{L} \quad & \quad \alpha L > 0
\end{aligned}
$$

$\alpha < \frac{2}{L}$ is needed for convergence.

Now we would like to tune α to choose the best (lowest) convergence rate

$$
\rho^* = \min_\alpha \rho(\alpha)
$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T\Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k \\
&= (I - \alpha^k \Lambda) x^k \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)}) x^k_{(i)} \text{ For } i\text{-th coordinate} \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}
\end{aligned}
$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:

$$
\rho(\alpha) = \max_i |1 - \alpha \lambda_{(i)}| < 1
$$

Remember, that $\lambda_{\text{min}} = \mu > 0, \lambda_{\text{max}} = L \geq \mu$.

$$
\begin{aligned}
|1 - \alpha \mu| &< 1 \\
-1 < 1 &- \alpha \mu < 1 \\
\alpha < \frac{2}{\mu} \quad & \quad \alpha \mu > 0
\end{aligned}
\qquad
\begin{aligned}
|1 - \alpha L| &< 1 \\
-1 < 1 &- \alpha L < 1 \\
\alpha < \frac{2}{L} \quad & \quad \alpha L > 0
\end{aligned}
$$

$\alpha < \frac{2}{L}$ is needed for convergence.

Now we would like to tune α to choose the best (lowest) convergence rate

$$
\rho^* = \min_{\alpha} \rho(\alpha) = \min_{\alpha} \max_i |1 - \alpha \lambda_{(i)}|
$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T\Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k \\
&= (I - \alpha^k \Lambda) x^k \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)}) x^k_{(i)} \text{ For } i\text{-th coordinate} \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}
\end{aligned}
$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:

$$
\rho(\alpha) = \max_i |1 - \alpha\lambda_{(i)}| < 1
$$

Remember, that $\lambda_{\text{min}} = \mu > 0, \lambda_{\text{max}} = L \geq \mu$.

$$
\begin{aligned}
|1 - \alpha\mu| &< 1 \\
-1 < 1 - \alpha\mu &< 1 \\
\alpha < \frac{2}{\mu} \quad\quad & \alpha\mu > 0
\end{aligned}
\qquad
\begin{aligned}
|1 - \alpha L| &< 1 \\
-1 < 1 - \alpha L &< 1 \\
\alpha < \frac{2}{L} \quad\quad & \alpha L > 0
\end{aligned}
$$

$\alpha < \frac{2}{L}$ is needed for convergence.

Now we would like to tune α to choose the best (lowest) convergence rate

$$
\begin{aligned}
\rho^* &= \min_\alpha \rho(\alpha) = \min_\alpha \max_i |1 - \alpha\lambda_{(i)}| \\
&= \min_\alpha \left\{ |1 - \alpha\mu|, |1 - \alpha L| \right\}
\end{aligned}
$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T\Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k \\
&= (I - \alpha^k \Lambda) x^k \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)}) x^k_{(i)} \text{ For } i\text{-th coordinate} \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}
\end{aligned}
$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:

$$
\rho(\alpha) = \max_i |1 - \alpha\lambda_{(i)}| < 1
$$

Remember, that $\lambda_{\text{min}} = \mu > 0, \lambda_{\text{max}} = L \geq \mu$.

$$
\begin{aligned}
&|1 - \alpha\mu| < 1 && |1 - \alpha L| < 1 \\
&-1 < 1 - \alpha\mu < 1 && -1 < 1 - \alpha L < 1 \\
&\alpha < \frac{2}{\mu} \quad \alpha\mu > 0 && \alpha < \frac{2}{L} \quad \alpha L > 0
\end{aligned}
$$

$\alpha < \frac{2}{L}$ is needed for convergence.

Now we would like to tune α to choose the best (lowest) convergence rate

$$
\begin{aligned}
\rho^* &= \min_\alpha \rho(\alpha) = \min_\alpha \max_i |1 - \alpha\lambda_{(i)}| \\
&= \min_\alpha \left\{ |1 - \alpha\mu|, |1 - \alpha L| \right\} \\
\alpha^* &: \quad 1 - \alpha^*\mu = \alpha^* L - 1
\end{aligned}
$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T\Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k \\
&= (I - \alpha^k \Lambda) x^k \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)}) x^k_{(i)} \text{ For } i\text{-th coordinate} \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}
\end{aligned}
$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:

$$
\rho(\alpha) = \max_i |1 - \alpha \lambda_{(i)}| < 1
$$

Remember, that $\lambda_{\text{min}} = \mu > 0, \lambda_{\text{max}} = L \geq \mu$.

$$
\begin{aligned}
&|1 - \alpha \mu| < 1 && |1 - \alpha L| < 1 \\
&-1 < 1 - \alpha \mu < 1 && -1 < 1 - \alpha L < 1 \\
&\alpha < \frac{2}{\mu} \quad \alpha \mu > 0 && \alpha < \frac{2}{L} \quad \alpha L > 0
\end{aligned}
$$

$\alpha < \frac{2}{L}$ is needed for convergence.

Now we would like to tune α to choose the best (lowest) convergence rate

$$
\begin{aligned}
\rho^* &= \min_\alpha \rho(\alpha) = \min_\alpha \max_i |1 - \alpha \lambda_{(i)}| \\
&= \min_\alpha \left\{ |1 - \alpha \mu|, |1 - \alpha L| \right\} \\
\alpha^* &: \quad 1 - \alpha^* \mu = \alpha^* L - 1 \\
\alpha^* &= \frac{2}{\mu + L}
\end{aligned}
$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T\Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k \\
&= (I - \alpha^k \Lambda) x^k \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)}) x^k_{(i)} \text{ For } i\text{-th coordinate} \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}
\end{aligned}
$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:

$$
\rho(\alpha) = \max_i |1 - \alpha \lambda_{(i)}| < 1
$$

Remember, that $\lambda_{\text{min}} = \mu > 0, \lambda_{\text{max}} = L \geq \mu$.

$$
\begin{aligned}
&|1 - \alpha\mu| < 1 && |1 - \alpha L| < 1 \\
&-1 < 1 - \alpha\mu < 1 && -1 < 1 - \alpha L < 1 \\
&\alpha < \frac{2}{\mu} \quad \alpha\mu > 0 && \alpha < \frac{2}{L} \quad \alpha L > 0
\end{aligned}
$$

$\alpha < \frac{2}{L}$ is needed for convergence.

Now we would like to tune α to choose the best (lowest) convergence rate

$$
\begin{aligned}
\rho^* &= \min_\alpha \rho(\alpha) = \min_\alpha \max_i |1 - \alpha \lambda_{(i)}| \\
&= \min_\alpha \left\{ |1 - \alpha\mu|, |1 - \alpha L| \right\} \\
\alpha^* &: \quad 1 - \alpha^* \mu = \alpha^* L - 1 \\
\alpha^* &= \frac{2}{\mu + L} \quad \rho^* = \frac{L - \mu}{L + \mu}
\end{aligned}
$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T\Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k \\
&= (I - \alpha^k \Lambda) x^k \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)}) x^k_{(i)} \text{ For } i\text{-th coordinate} \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}
\end{aligned}
$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:

$$
\rho(\alpha) = \max_i |1 - \alpha \lambda_{(i)}| < 1
$$

Remember, that $\lambda_{\text{min}} = \mu > 0, \lambda_{\text{max}} = L \geq \mu$.

$$
\begin{aligned}
|1 - \alpha\mu| &< 1 & |1 - \alpha L| &< 1 \\
-1 < 1 - \alpha\mu &< 1 & -1 < 1 - \alpha L &< 1 \\
\alpha < \frac{2}{\mu} \quad & \alpha\mu > 0 & \alpha < \frac{2}{L} \quad & \alpha L > 0
\end{aligned}
$$

$\alpha < \frac{2}{L}$ is needed for convergence.

Now we would like to tune α to choose the best (lowest) convergence rate

$$
\begin{aligned}
\rho^* &= \min_\alpha \rho(\alpha) = \min_\alpha \max_i |1 - \alpha \lambda_{(i)}| \\
&= \min_\alpha \left\{ |1 - \alpha\mu|, |1 - \alpha L| \right\} \\
\alpha^* &: \quad 1 - \alpha^*\mu = \alpha^* L - 1 \\
\alpha^* &= \frac{2}{\mu + L} \quad \rho^* = \frac{L - \mu}{L + \mu} \\
x^{k+1} &= \left(\frac{L - \mu}{L + \mu} \right)^k x^0
\end{aligned}
$$

Convergence analysis

Now we can work with the function $f(x) = \frac{1}{2}x^T\Lambda x$ with $x^* = 0$ without loss of generality (drop the hat from the $\hat{x}$)

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k \Lambda x^k \\
&= (I - \alpha^k \Lambda) x^k \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)}) x^k_{(i)} \text{ For } i\text{-th coordinate} \\
x^{k+1}_{(i)} &= (1 - \alpha^k \lambda_{(i)})^k x^0_{(i)}
\end{aligned}
$$

Let's use constant stepsize $\alpha^k = \alpha$. Convergence condition:

$$
\rho(\alpha) = \max_i |1 - \alpha\lambda_{(i)}| < 1
$$

Remember, that $\lambda_{\text{min}} = \mu > 0, \lambda_{\text{max}} = L \geq \mu$.

$$
\begin{aligned}
&|1 - \alpha\mu| < 1 && |1 - \alpha L| < 1 \\
&-1 < 1 - \alpha\mu < 1 && -1 < 1 - \alpha L < 1 \\
&\alpha < \frac{2}{\mu} \quad \alpha\mu > 0 && \alpha < \frac{2}{L} \quad \alpha L > 0
\end{aligned}
$$

$\alpha < \frac{2}{L}$ is needed for convergence.

Now we would like to tune α to choose the best (lowest) convergence rate

$$
\begin{aligned}
\rho^* &= \min_\alpha \rho(\alpha) = \min_\alpha \max_i |1 - \alpha\lambda_{(i)}| \\
&= \min_\alpha \{|1 - \alpha\mu|, |1 - \alpha L|\} \\
\alpha^* &: \quad 1 - \alpha^*\mu = \alpha^* L - 1 \\
\alpha^* &= \frac{2}{\mu + L} \quad \rho^* = \frac{L - \mu}{L + \mu}
\end{aligned}
$$

$$
x^{k+1} = \left(\frac{L - \mu}{L + \mu}\right)^k x^0 \quad f(x^{k+1}) = \left(\frac{L - \mu}{L + \mu}\right)^{2k} f(x^0)
$$

Convergence analysis

So, we have a linear convergence in the domain with rate $\frac{\kappa-1}{\kappa+1} = 1 - \frac{2}{\kappa+1}$, where $\kappa = \frac{L}{\mu}$ is sometimes called *condition number* of the quadratic problem.

κ	ρ	Iterations to decrease domain gap 10 times	Iterations to decrease function gap 10 times
1.1	0.05	1	1
2	0.33	3	2
5	0.67	6	3
10	0.82	12	6
50	0.96	58	29
100	0.98	116	58
500	0.996	576	288
1000	0.998	1152	576

Polyak-Lojasiewicz smooth case

Polyak-Lojasiewicz condition. Linear convergence of gradient descent without convexity

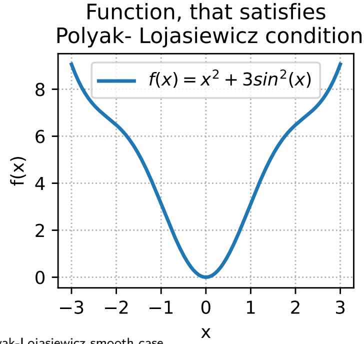
PL inequality holds if the following condition is satisfied for some $\mu > 0$,

$$\|\nabla f(x)\|^2 \geq 2\mu(f(x) - f^*) \quad \forall x$$

It is interesting, that the Gradient Descent algorithm might converge linearly even without convexity.

The following functions satisfy the PL condition but are not convex. Link to the code

$$f(x) = x^2 + 3\sin^2(x)$$



Polyak-Lojasiewicz condition. Linear convergence of gradient descent without convexity

PL inequality holds if the following condition is satisfied for some $\mu > 0$,

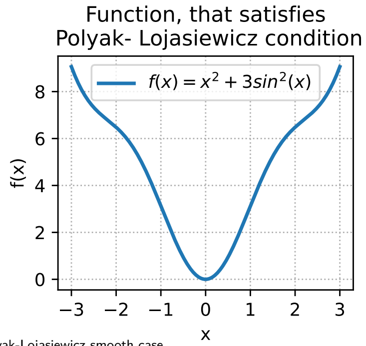
$$\|\nabla f(x)\|^2 \geq 2\mu(f(x) - f^*) \quad \forall x$$

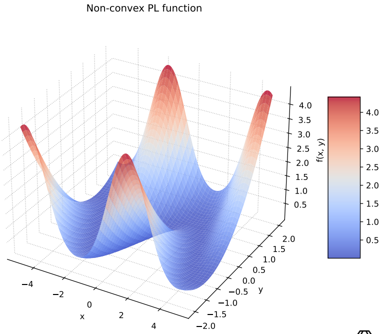
It is interesting, that the Gradient Descent algorithm might converge linearly even without convexity.

The following functions satisfy the PL condition but are not convex. Link to the code

$$f(x) = x^2 + 3\sin^2(x)$$

$$f(x, y) = \frac{(y - \sin x)^2}{2}$$





Convergence analysis

Theorem

Consider the Problem

$$f(x) \to \min_{x \in \mathbb{R}^d}$$

and assume that f is μ-Polyak-Lojasiewicz and L-smooth, for some $L \geq \mu > 0$.
Consider $(x^k)_{k \in \mathbb{N}}$ a sequence generated by the gradient descent constant stepsize algorithm, with a stepsize satisfying $0 < \alpha \leq \frac{1}{L}$. Then:

$$f(x^k) - f^* \leq (1 - \alpha\mu)^k (f(x^0) - f^*).$$

Example: linear least squares

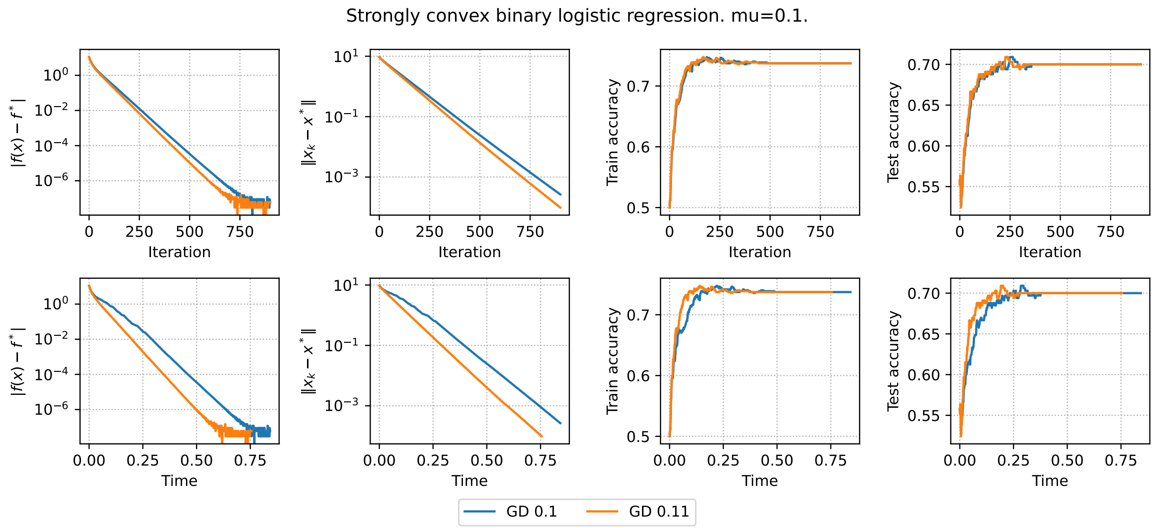


Figure 4: Convergence both in domain and in function value for regularized quadratics

Smooth convex case

Smooth convex case

Theorem

Consider the Problem

$$f(x) \to \min_{x \in \mathbb{R}^d}$$

and assume that f is convex and L-smooth, for some $L > 0$.
Let $(x^k)_{k \in \mathbb{N}}$ be the sequence of iterates generated by the gradient descent constant stepsize algorithm, with a stepsize satisfying $0 < \alpha \leq \frac{1}{L}$. Then, for all $x^* \in \operatorname{argmin} f$, for all $k \in \mathbb{N}$ we have that

$$f(x^k) - f^* \leq \frac{\|x^0 - x^*\|^2}{2\alpha k}.$$

Example: linear least squares

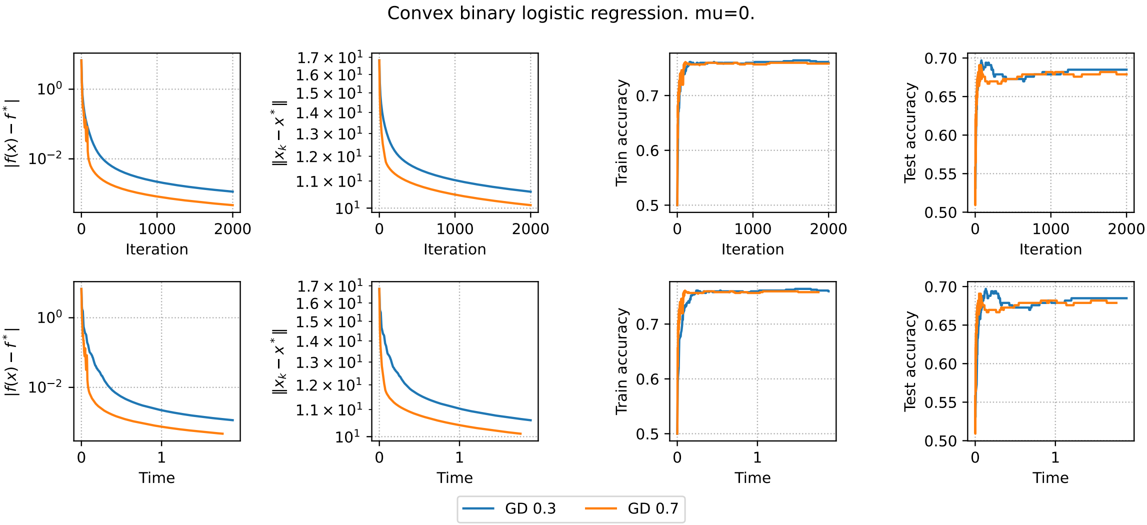


Figure 5: Convergence in function value for convex (but not strongly convex) quadratics

Lower bounds

How optimal is $\mathcal{O}\left(\frac{1}{k}\right)$?

- Is it somehow possible to understand, that the obtained convergence is the fastest possible with this class of problem and this class of algorithms?

How optimal is $\mathcal{O}\left(\frac{1}{k}\right)$?

- Is it somehow possible to understand, that the obtained convergence is the fastest possible with this class of problem and this class of algorithms?
- The iteration of gradient descent:

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) \\
&= x^{k-1} - \alpha^{k-1} \nabla f(x^{k-1}) - \alpha^k \nabla f(x^k) \\
&\;\;\vdots \\
&= x^0 - \sum_{i=0}^{k} \alpha^{k-i} \nabla f(x^{k-i})
\end{aligned}
$$

How optimal is $\mathcal{O}\left(\frac{1}{k}\right)$?

- Is it somehow possible to understand, that the obtained convergence is the fastest possible with this class of problem and this class of algorithms?
- The iteration of gradient descent:

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) \\
&= x^{k-1} - \alpha^{k-1} \nabla f(x^{k-1}) - \alpha^k \nabla f(x^k) \\
&\;\vdots \\
&= x^0 - \sum_{i=0}^{k} \alpha^{k-i} \nabla f(x^{k-i})
\end{aligned}
$$

- Consider a family of first-order methods, where

$$
x^{k+1} \in x^0 + \text{span}\left\{\nabla f(x^0), \nabla f(x^1), \dots, \nabla f(x^k)\right\} \tag{1}
$$

Smooth convex case

Theorem

There exists a function f that is L-smooth and convex such that any method 2 satisfies

$$\min_{i \in [1,k]} f(x^i) - f^* \geq \frac{3L\|x^0 - x^*\|_2^2}{32(1+k)^2}$$

Smooth convex case

Theorem

There exists a function f that is L-smooth and convex such that any method 2 satisfies

$$\min_{i \in [1,k]} f(x^i) - f^* \geq \frac{3L\|x^0 - x^*\|_2^2}{32(1+k)^2}$$

- No matter what gradient method you provide, there is always a function f that, when you apply your gradient method on minimizing such f, the convergence rate is lower bounded as $\mathcal{O}\left(\frac{1}{k^2}\right)$.

Smooth convex case

Theorem

There exists a function f that is L-smooth and convex such that any method 2 satisfies

$$\min_{i \in [1,k]} f(x^i) - f^* \geq \frac{3L\|x^0 - x^*\|_2^2}{32(1+k)^2}$$

- No matter what gradient method you provide, there is always a function f that, when you apply your gradient method on minimizing such f, the convergence rate is lower bounded as $\mathcal{O}\left(\frac{1}{k^2}\right)$.
- The key to the proof is to explicitly build a special function f.

Recap

Recap

Gradient Descent: $$\min_{x \in \mathbb{R}^n} f(x) \qquad x^{k+1} = x^k - \alpha^k \nabla f(x^k)$$

convex (non-smooth)	smooth (non-convex)	smooth & convex	smooth & strongly convex (or PL)
$f(x^k) - f^* \sim \mathcal{O}\left(\dfrac{1}{\sqrt{k}}\right)$	$\\|\nabla f(x^k)\\|^2 \sim \mathcal{O}\left(\dfrac{1}{k}\right)$	$f(x^k) - f^* \sim \mathcal{O}\left(\dfrac{1}{k}\right)$	$\\|x^k - x^*\\|^2 \sim \mathcal{O}\left(\left(1 - \dfrac{\mu}{L}\right)^k\right)$
$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon^2}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\kappa \log \dfrac{1}{\varepsilon}\right)$

Recap

Gradient Descent: $\min_{x \in \mathbb{R}^n} f(x)$ $x^{k+1} = x^k - \alpha^k \nabla f(x^k)$

convex (non-smooth)	smooth (non-convex)	smooth & convex	smooth & strongly convex (or PL)
$f(x^k) - f^* \sim \mathcal{O}\left(\dfrac{1}{\sqrt{k}}\right)$	$\lVert\nabla f(x^k)\rVert^2 \sim \mathcal{O}\left(\dfrac{1}{k}\right)$	$f(x^k) - f^* \sim \mathcal{O}\left(\dfrac{1}{k}\right)$	$\lVert x^k - x^*\rVert^2 \sim \mathcal{O}\left(\left(1 - \dfrac{\mu}{L}\right)^k\right)$
$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon^2}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\kappa \log \dfrac{1}{\varepsilon}\right)$

For smooth strongly convex we have:

$$
f(x^k) - f^* \leq \left(1 - \frac{\mu}{L}\right)^k (f(x^0) - f^*).
$$

Note also, that for any x

$$
1 - x \leq e^{-x}
$$

Recap

Gradient Descent: $\min_{x \in \mathbb{R}^n} f(x) \qquad x^{k+1} = x^k - \alpha^k \nabla f(x^k)$

convex (non-smooth)	smooth (non-convex)	smooth & convex	smooth & strongly convex (or PL)
$f(x^k) - f^* \sim \mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$\|\nabla f(x^k)\|^2 \sim \mathcal{O}\left(\frac{1}{k}\right)$	$f(x^k) - f^* \sim \mathcal{O}\left(\frac{1}{k}\right)$	$\|x^k - x^*\|^2 \sim \mathcal{O}\left(\left(1 - \frac{\mu}{L}\right)^k\right)$
$k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\kappa \log \frac{1}{\varepsilon}\right)$

For smooth strongly convex we have:

$$f(x^k) - f^* \leq \left(1 - \frac{\mu}{L}\right)^k (f(x^0) - f^*).$$

Note also, that for any x

$$1 - x \leq e^{-x}$$

Finally we have

$$\begin{aligned}
\varepsilon = f(x^{k_\varepsilon}) - f^* &\leq \left(1 - \frac{\mu}{L}\right)^{k_\varepsilon} (f(x^0) - f^*) \\
&\leq \exp\left(-k_\varepsilon \frac{\mu}{L}\right) (f(x^0) - f^*) \\
k_\varepsilon &\geq \kappa \log \frac{f(x^0) - f^*}{\varepsilon} = \mathcal{O}\left(\kappa \log \frac{1}{\varepsilon}\right)
\end{aligned}$$

Recap

Gradient Descent: $$\min_{x \in \mathbb{R}^n} f(x) \qquad x^{k+1} = x^k - \alpha^k \nabla f(x^k)$$

convex (non-smooth)	smooth (non-convex)	smooth & convex	smooth & strongly convex (or PL)
$f(x^k) - f^* \sim \mathcal{O}\left(\dfrac{1}{\sqrt{k}}\right)$	$\\|\nabla f(x^k)\\|^2 \sim \mathcal{O}\left(\dfrac{1}{k}\right)$	$f(x^k) - f^* \sim \mathcal{O}\left(\dfrac{1}{k}\right)$	$\\|x^k - x^*\\|^2 \sim \mathcal{O}\left(\left(1 - \dfrac{\mu}{L}\right)^k\right)$
$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon^2}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\kappa \log \dfrac{1}{\varepsilon}\right)$

For smooth strongly convex we have:

$$f(x^k) - f^* \leq \left(1 - \frac{\mu}{L}\right)^k (f(x^0) - f^*).$$

Note also, that for any x

$$1 - x \leq e^{-x}$$

Finally we have

$$\begin{aligned}
\varepsilon = f(x^{k_\varepsilon}) - f^* &\leq \left(1 - \frac{\mu}{L}\right)^{k_\varepsilon} (f(x^0) - f^*) \\
&\leq \exp\left(-k_\varepsilon \frac{\mu}{L}\right) (f(x^0) - f^*) \\
k_\varepsilon \geq \kappa \log \frac{f(x^0) - f^*}{\varepsilon} &= \mathcal{O}\left(\kappa \log \frac{1}{\varepsilon}\right)
\end{aligned}$$

Question: Can we do faster, than this using the first-order information?

Recap

Gradient Descent: $\min_{x \in \mathbb{R}^n} f(x)$ $\qquad x^{k+1} = x^k - \alpha^k \nabla f(x^k)$

convex (non-smooth)	smooth (non-convex)	smooth & convex	smooth & strongly convex (or PL)
$f(x^k) - f^* \sim \mathcal{O}\left(\dfrac{1}{\sqrt{k}}\right)$	$\\|\nabla f(x^k)\\|^2 \sim \mathcal{O}\left(\dfrac{1}{k}\right)$	$f(x^k) - f^* \sim \mathcal{O}\left(\dfrac{1}{k}\right)$	$\\|x^k - x^*\\|^2 \sim \mathcal{O}\left(\left(1 - \dfrac{\mu}{L}\right)^k\right)$
$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon^2}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\kappa \log \dfrac{1}{\varepsilon}\right)$

For smooth strongly convex we have:

$$
f(x^k) - f^* \leq \left(1 - \frac{\mu}{L}\right)^k (f(x^0) - f^*).
$$

Note also, that for any x

$$
1 - x \leq e^{-x}
$$

Finally we have

$$
\begin{aligned}
\varepsilon = f(x^{k_\varepsilon}) - f^* &\leq \left(1 - \frac{\mu}{L}\right)^{k_\varepsilon} (f(x^0) - f^*) \\
&\leq \exp\left(-k_\varepsilon \frac{\mu}{L}\right) (f(x^0) - f^*) \\
k_\varepsilon \geq \kappa \log \frac{f(x^0) - f^*}{\varepsilon} &= \mathcal{O}\left(\kappa \log \frac{1}{\varepsilon}\right)
\end{aligned}
$$

Question: Can we do faster, than this using the first-order information? **Yes, we can.**

Lower bounds

Lower bounds

convex (non-smooth)	smooth (non-convex)[1]	smooth & convex[2]	smooth & strongly convex (or PL)
$\mathcal{O}\left(\dfrac{1}{\sqrt{k}}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon^2}\right)$	$\mathcal{O}\left(\dfrac{1}{k^2}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\sqrt{\varepsilon}}\right)$	$\mathcal{O}\left(\dfrac{1}{k^2}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\sqrt{\varepsilon}}\right)$	$\mathcal{O}\left(\left(1 - \sqrt{\dfrac{\mu}{L}}\right)^k\right)$ $k_\varepsilon \sim \mathcal{O}\left(\sqrt{\kappa} \log \dfrac{1}{\varepsilon}\right)$

[1] Carmon, Duchi, Hinder, Sidford, 2017
[2] Nemirovski, Yudin, 1979

Lower bounds

The iteration of gradient descent:

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) \\
&= x^{k-1} - \alpha^{k-1} \nabla f(x^{k-1}) - \alpha^k \nabla f(x^k) \\
&\vdots \\
&= x^0 - \sum_{i=0}^{k} \alpha^{k-i} \nabla f(x^{k-i})
\end{aligned}
$$

Lower bounds

The iteration of gradient descent:

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) \\
&= x^{k-1} - \alpha^{k-1} \nabla f(x^{k-1}) - \alpha^k \nabla f(x^k) \\
&\vdots \\
&= x^0 - \sum_{i=0}^{k} \alpha^{k-i} \nabla f(x^{k-i})
\end{aligned}
$$

Consider a family of first-order methods, where

$$
x^{k+1} \in x^0 + \text{span}\left\{\nabla f(x^0), \nabla f(x^1), \ldots, \nabla f(x^k)\right\} \quad (2)
$$

Lower bounds

The iteration of gradient descent:

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) \\
&= x^{k-1} - \alpha^{k-1} \nabla f(x^{k-1}) - \alpha^k \nabla f(x^k) \\
&\;\vdots \\
&= x^0 - \sum_{i=0}^{k} \alpha^{k-i} \nabla f(x^{k-i})
\end{aligned}
$$

Consider a family of first-order methods, where

$$
x^{k+1} \in x^0 + \text{span}\left\{\nabla f(x^0), \nabla f(x^1), \ldots, \nabla f(x^k)\right\} \quad (2)
$$

Non-smooth convex case

There exists a function f that is M-Lipschitz and convex such that any first-order method of the form 2 satisfies

$$
\min_{i \in [1,k]} f(x^i) - f^* \geq \frac{M \|x^0 - x^*\|_2}{2(1 + \sqrt{k})}
$$

Lower bounds

The iteration of gradient descent:

$$
\begin{aligned}
x^{k+1} &= x^k - \alpha^k \nabla f(x^k) \\
&= x^{k-1} - \alpha^{k-1} \nabla f(x^{k-1}) - \alpha^k \nabla f(x^k) \\
&\ \vdots \\
&= x^0 - \sum_{i=0}^{k} \alpha^{k-i} \nabla f(x^{k-i})
\end{aligned}
$$

Consider a family of first-order methods, where

$$
x^{k+1} \in x^0 + \text{span}\left\{\nabla f(x^0), \nabla f(x^1), \ldots, \nabla f(x^k)\right\} \quad (2)
$$

Non-smooth convex case

There exists a function f that is M-Lipschitz and convex such that any first-order method of the form 2 satisfies

$$
\min_{i \in [1,k]} f(x^i) - f^* \geq \frac{M\|x^0 - x^*\|_2}{2(1+\sqrt{k})}
$$

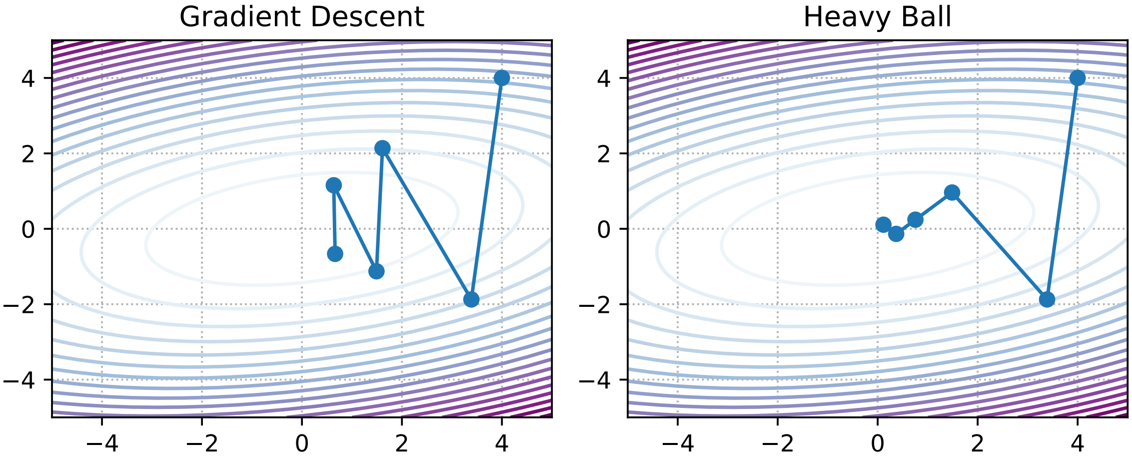
Smooth and convex case

There exists a function f that is L-smooth and convex such that any first-order method of the form 2 satisfies

$$
\min_{i \in [1,k]} f(x^i) - f^* \geq \frac{3L\|x^0 - x^*\|_2^2}{32(1+k)^2}
$$

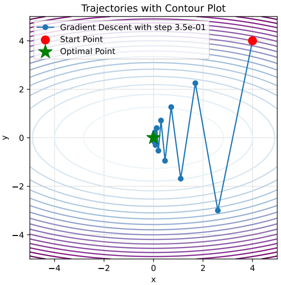
Strongly convex quadratic problem

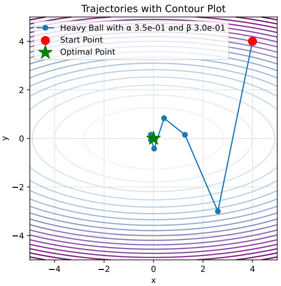
Oscillations and acceleration



Heavy ball

Polyak Heavy ball method





Let's introduce the idea of momentum, proposed by Polyak in 1964. Recall that the momentum update is

$$x^{k+1} = x^k - \alpha \nabla f(x^k) + \beta(x^k - x_{k-1}).$$

optimal hyperparameters for strongly convex quadratics:

$$\alpha^*, \beta^* = \arg\min_{\alpha,\beta} \max_{\lambda \in [\mu, L]} \rho(M) \quad \alpha^* = \frac{4}{(\sqrt{L} + \sqrt{\mu})^2}; \quad \beta^* = \left(\frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}\right)^2.$$

Heavy Ball quadratics convergence

Theorem

Assume that f is quadratic μ-strongly convex L-smooth quadratics, then Heavy Ball method with parameters

$$x\alpha = \frac{4}{(\sqrt{L} + \sqrt{\mu})^2}, \beta = \frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}$$

converges linearly:

$$\|x_k - x^*\|_2 \leq \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right) \|x_0 - x^*\|$$

Heavy Ball Global Convergence [3]

Theorem

Assume that f is smooth and convex and that

$$\beta \in [0,1), \quad \alpha \in \left(0, \frac{2(1-\beta)}{L}\right).$$

Then, the sequence $\{x_k\}$ generated by Heavy-ball iteration satisfies

$$f(\overline{x}_T) - f^\star \leq \begin{cases} \frac{\|x_0 - x^\star\|^2}{2(T+1)}\left(\frac{L\beta}{1-\beta} + \frac{1-\beta}{\alpha}\right), & \text{if } \alpha \in \left(0, \frac{1-\beta}{L}\right], \\ \frac{\|x_0 - x^\star\|^2}{2(T+1)(2(1-\beta)-\alpha L)}\left(L\beta + \frac{(1-\beta)^2}{\alpha}\right), & \text{if } \alpha \in \left[\frac{1-\beta}{L}, \frac{2(1-\beta)}{L}\right), \end{cases}$$

where $\overline{x}_T$ is the Cesaro average of the iterates, i.e.,

$$\overline{x}_T = \frac{1}{T+1}\sum_{k=0}^{T} x_k.$$

[3] Global convergence of the Heavy-ball method for convex optimization, Euhanna Ghadimi et.al.

Heavy Ball Global Convergence [4]

Theorem

Assume that f is smooth and strongly convex and that

$$\alpha \in (0, \frac{2}{L}), \quad 0 \leq \beta < \frac{1}{2}\left(\frac{\mu\alpha}{2} + \sqrt{\frac{\mu^2\alpha^2}{4} + 4(1 - \frac{\alpha L}{2})}\right).$$

where $\alpha_0 \in (0, 1/L]$. Then, the sequence $\{x_k\}$ generated by Heavy-ball iteration converges linearly to a unique optimizer $x^\star$. In particular,

$$f(x_k) - f^\star \leq q^k(f(x_0) - f^\star),$$

where $q \in [0, 1)$.

[4] Global convergence of the Heavy-ball method for convex optimization, Euhanna Ghadimi et.al.

Heavy ball method summary

- Ensures accelerated convergence for strongly convex quadratic problems

Heavy ball method summary

- Ensures accelerated convergence for strongly convex quadratic problems
- Local accelerated convergence was proved in the original paper.

Heavy ball method summary

- Ensures accelerated convergence for strongly convex quadratic problems
- Local accelerated convergence was proved in the original paper.
- Recently was proved, that there is no global accelerated convergence for the method.

Heavy ball method summary

- Ensures accelerated convergence for strongly convex quadratic problems
- Local accelerated convergence was proved in the original paper.
- Recently was proved, that there is no global accelerated convergence for the method.
- Method was not extremely popular until the ML boom

Heavy ball method summary

* Ensures accelerated convergence for strongly convex quadratic problems
* Local accelerated convergence was proved in the original paper.
* Recently was proved, that there is no global accelerated convergence for the method.
* Method was not extremely popular until the ML boom
* Nowadays, it is de-facto standard for practical acceleration of gradient methods, even for the non-convex problems (neural network training)

Nesterov accelerated gradient

The concept of Nesterov Accelerated Gradient method

$$x_{k+1} = x_k - \alpha \nabla f(x_k)$$

$$x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta (x_k - x_{k-1})$$

$$\begin{cases} y_{k+1} = x_k + \beta (x_k - x_{k-1}) \\ x_{k+1} = y_{k+1} - \alpha \nabla f(y_{k+1}) \end{cases}$$

The concept of Nesterov Accelerated Gradient method

$$x_{k+1} = x_k - \alpha \nabla f(x_k) \qquad x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta (x_k - x_{k-1}) \qquad \begin{cases} y_{k+1} = x_k + \beta (x_k - x_{k-1}) \\ x_{k+1} = y_{k+1} - \alpha \nabla f(y_{k+1}) \end{cases}$$

Let's define the following notation

$$x^+ = x - \alpha \nabla f(x) \qquad \text{Gradient step}$$
$$d_k = \beta_k (x_k - x_{k-1}) \qquad \text{Momentum term}$$

Then we can write down:

$$x_{k+1} = x_k^+ \qquad \text{Gradient Descent}$$
$$x_{k+1} = x_k^+ + d_k \qquad \text{Heavy Ball}$$
$$x_{k+1} = (x_k + d_k)^+ \qquad \text{Nesterov accelerated gradient}$$

NAG convergence for quadratics

General case convergence

Theorem

Let $f : \mathbb{R}^n \to \mathbb{R}$ is convex and L-smooth. The Nesterov Accelerated Gradient Descent (NAG) algorithm is designed to solve the minimization problem starting with an initial point $x_0 = y_0 \in \mathbb{R}^n$ and $\lambda_0 = 0$. The algorithm iterates the following steps:

$$
\begin{aligned}
&\textbf{Gradient update:} && y_{k+1} = x_k - \frac{1}{L}\nabla f(x_k) \\
&\textbf{Extrapolation:} && x_{k+1} = (1-\gamma_k)y_{k+1} + \gamma_k y_k \\
&\textbf{Extrapolation weight:} && \lambda_{k+1} = \frac{1+\sqrt{1+4\lambda_k^2}}{2} \\
&\textbf{Extrapolation weight:} && \gamma_k = \frac{1-\lambda_k}{\lambda_{k+1}}
\end{aligned}
$$

The sequences $\{f(y_k)\}_{k\in\mathbb{N}}$ produced by the algorithm will converge to the optimal value f^* at the rate of $\mathcal{O}\left(\frac{1}{k^2}\right)$, specifically:

$$
f(y_k) - f^* \leq \frac{2L\|x_0 - x^*\|^2}{k^2}
$$

General case convergence

Theorem

Let $f : \mathbb{R}^n \to \mathbb{R}$ is μ-strongly convex and L-smooth. The Nesterov Accelerated Gradient Descent (NAG) algorithm is designed to solve the minimization problem starting with an initial point $x_0 = y_0 \in \mathbb{R}^n$ and $\lambda_0 = 0$. The algorithm iterates the following steps:

Gradient update: $$y_{k+1} = x_k - \frac{1}{L}\nabla f(x_k)$$

Extrapolation: $$x_{k+1} = (1 - \gamma_k)y_{k+1} + \gamma_k y_k$$

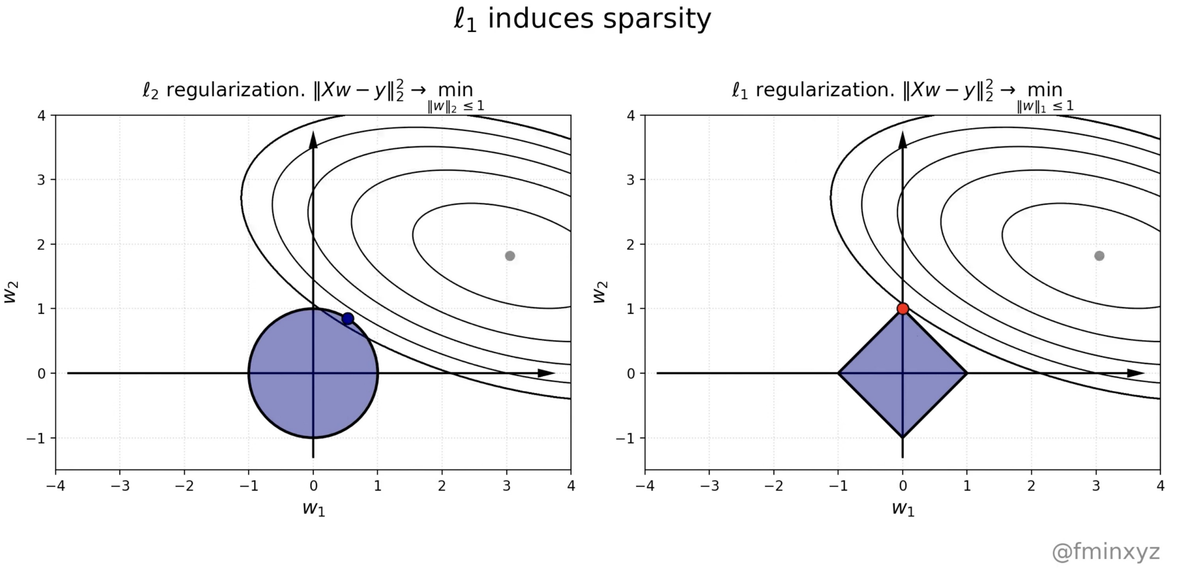
Extrapolation weight: $$\gamma_k = \frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}$$

The sequences $\{f(y_k)\}_{k \in \mathbb{N}}$ produced by the algorithm will converge to the optimal value f^* linearly:

$$f(y_k) - f^* \leq \frac{\mu + L}{2}\|x_0 - x^*\|_2^2 \exp\left(-\frac{k}{\sqrt{\kappa}}\right)$$

Non-smooth problems

ℓ_1-regularized linear least squares



Norms are not smooth

$$\min_{x \in \mathbb{R}^n} f(x),$$

A classical convex optimization problem is considered. We assume that $f(x)$ is a convex function, but now we do not require smoothness.

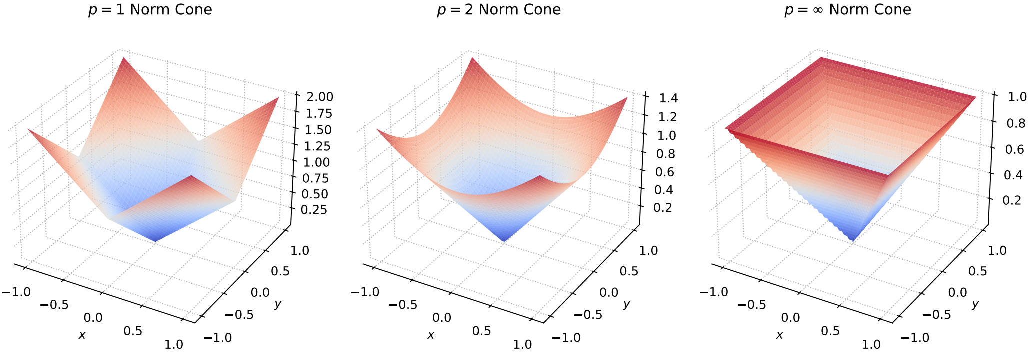


Figure 6: Norm cones for different p - norms are non-smooth

Wolfe's example

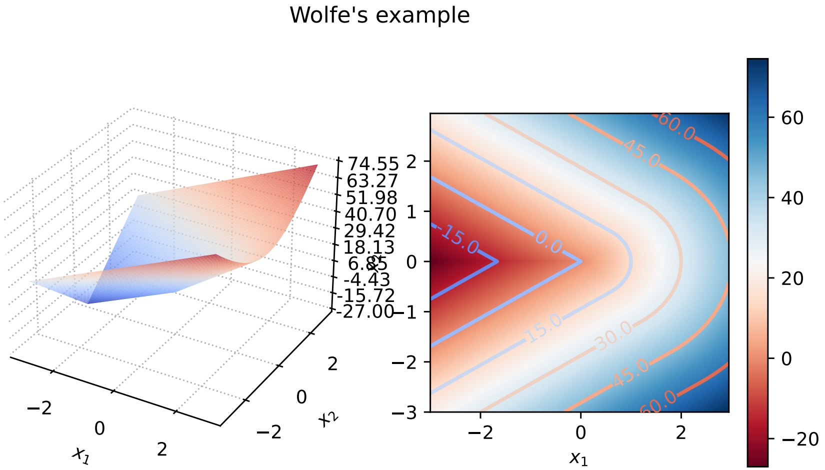


Figure 7: Wolfe's example. Open in Colab

Subgradient calculus

Convex function linear lower bound

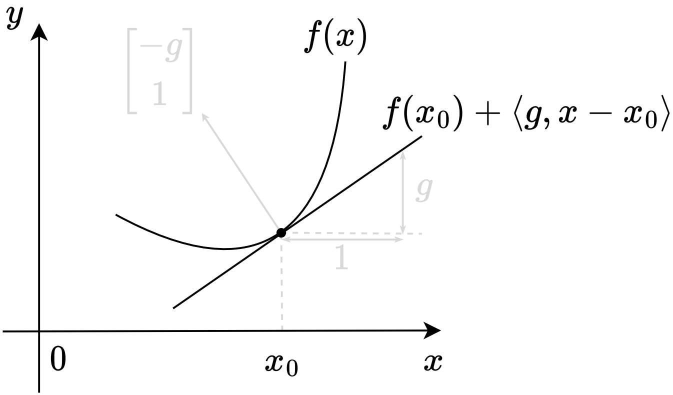


Figure 8: Taylor linear approximation serves as a global lower bound for a convex function

An important property of a continuous convex function $f(x)$ is that at any chosen point x_0 for all $x \in \text{dom } f$ the inequality holds:

$$f(x) \geq f(x_0) + \langle g, x - x_0 \rangle$$

Convex function linear lower bound

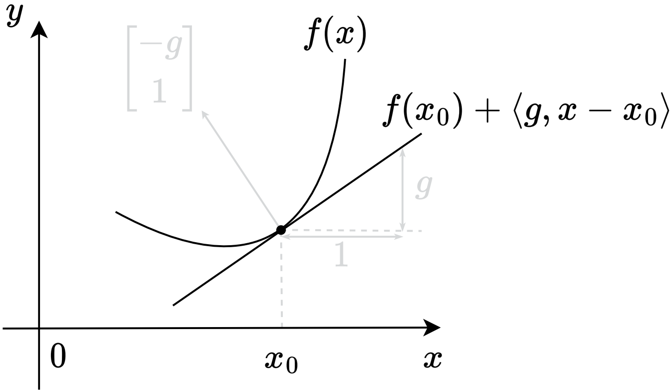


Figure 8: Taylor linear approximation serves as a global lower bound for a convex function

An important property of a continuous convex function $f(x)$ is that at any chosen point x_0 for all $x \in \text{dom } f$ the inequality holds:

$$f(x) \geq f(x_0) + \langle g, x - x_0 \rangle$$

for some vector g, i.e., the tangent to the graph of the function is the *global* estimate from below for the function.

- If $f(x)$ is differentiable, then $g = \nabla f(x_0)$

Convex function linear lower bound

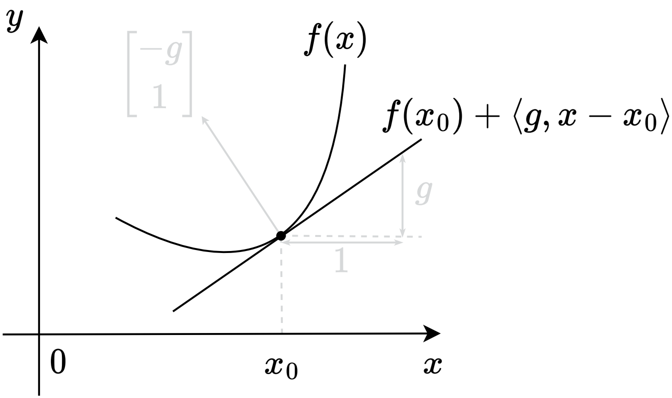


Figure 8: Taylor linear approximation serves as a global lower bound for a convex function

An important property of a continuous convex function $f(x)$ is that at any chosen point x_0 for all $x \in \text{dom } f$ the inequality holds:

$$f(x) \geq f(x_0) + \langle g, x - x_0 \rangle$$

for some vector g, i.e., the tangent to the graph of the function is the *global* estimate from below for the function.

- If $f(x)$ is differentiable, then $g = \nabla f(x_0)$
- Not all continuous convex functions are differentiable.

Convex function linear lower bound

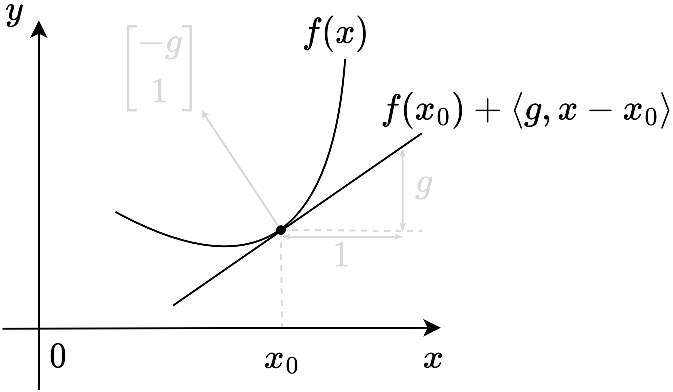


Figure 8: Taylor linear approximation serves as a global lower bound for a convex function

An important property of a continuous convex function $f(x)$ is that at any chosen point x_0 for all $x \in \text{dom } f$ the inequality holds:

$$f(x) \geq f(x_0) + \langle g, x - x_0 \rangle$$

for some vector g, i.e., the tangent to the graph of the function is the *global* estimate from below for the function.

- If $f(x)$ is differentiable, then $g = \nabla f(x_0)$
- Not all continuous convex functions are differentiable.

Convex function linear lower bound

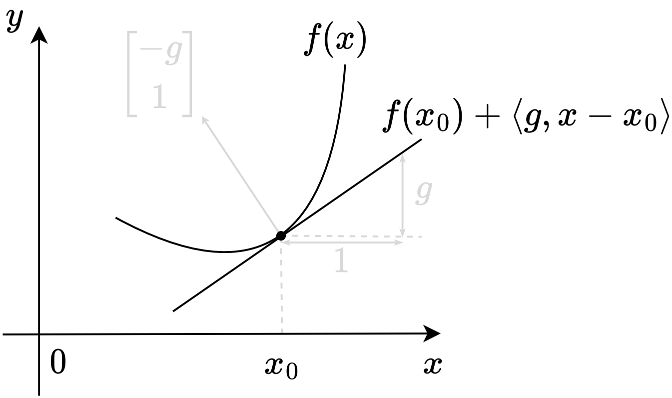


Figure 8: Taylor linear approximation serves as a global lower bound for a convex function

An important property of a continuous convex function $f(x)$ is that at any chosen point x_0 for all $x \in \text{dom } f$ the inequality holds:

$$f(x) \geq f(x_0) + \langle g, x - x_0 \rangle$$

for some vector g, i.e., the tangent to the graph of the function is the *global* estimate from below for the function.

- If $f(x)$ is differentiable, then $g = \nabla f(x_0)$
- Not all continuous convex functions are differentiable.

We wouldn't want to lose such a nice property.

Subgradient and subdifferential

A vector g is called the **subgradient** of a function $f(x) : S \to \mathbb{R}$ at a point x_0 if $\forall x \in S$:

$$f(x) \geq f(x_0) + \langle g, x - x_0 \rangle$$

Subgradient and subdifferential

A vector g is called the **subgradient** of a function $f(x) : S \to \mathbb{R}$ at a point x_0 if $\forall x \in S$:

$$
f(x) \geq f(x_0) + \langle g, x - x_0 \rangle
$$

The set of all subgradients of a function $f(x)$ at a point x_0 is called the **subdifferential** of f at x_0 and is denoted by $\partial f(x_0)$.

Subgradient and subdifferential

A vector g is called the **subgradient** of a function $f(x) : S \to \mathbb{R}$ at a point x_0 if $\forall x \in S$:

$$f(x) \geq f(x_0) + \langle g, x - x_0 \rangle$$

The set of all subgradients of a function $f(x)$ at a point x_0 is called the **subdifferential** of f at x_0 and is denoted by $\partial f(x_0)$.

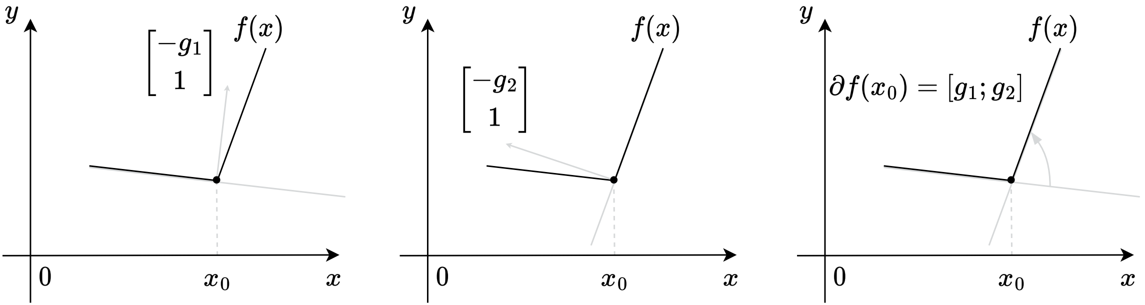


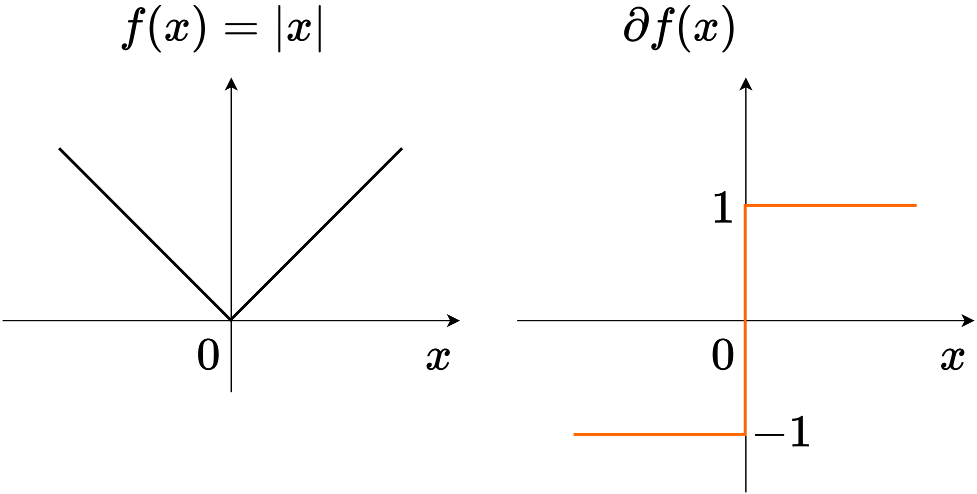
Figure 9: Subdifferential is a set of all possible subgradients

Subgradient and subdifferential

Find $\partial f(x)$, if $f(x) = |x|$

Subgradient and subdifferential

Find $\partial f(x)$, if $f(x) = |x|$



Subgradient Method

Algorithm

A vector g is called the **subgradient** of the function $f(x) : S \to \mathbb{R}$ at the point x_0 if $\forall x \in S$:

$$f(x) \geq f(x_0) + \langle g, x - x_0 \rangle$$

Algorithm

A vector g is called the **subgradient** of the function $f(x) : S \to \mathbb{R}$ at the point x_0 if $\forall x \in S$:

$$f(x) \geq f(x_0) + \langle g, x - x_0 \rangle$$

The idea is very simple: let's replace the gradient $\nabla f(x_k)$ in the gradient descent algorithm with a subgradient g_k at point x_k:

$$x_{k+1} = x_k - \alpha_k g_k,$$

where g_k is an arbitrary subgradient of the function $f(x)$ at the point x_k, $g_k \in \partial f(x_k)$

Convergence results

Theorem

Let f be a convex G-Lipschitz function. For a fixed step size $\alpha = \frac{\|x_0 - x^*\|_2}{G}\sqrt{\frac{1}{K}}$, subgradient method satisfies

$$f(\overline{x}) - f^* \leq \frac{G\|x_0 - x^*\|_2}{\sqrt{K}} \qquad \overline{x} = \frac{1}{K}\sum_{k=0}^{K-1} x_i$$

- $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$ is slow, but already hits the lower bound ($\mathcal{O}\left(\frac{1}{T}\right)$ in the strongly convex case).

Convergence results

Theorem

Let f be a convex G-Lipschitz function. For a fixed step size $\alpha = \frac{\|x_0 - x^*\|_2}{G}\sqrt{\frac{1}{K}}$, subgradient method satisfies

$$f(\overline{x}) - f^* \leq \frac{G\|x_0 - x^*\|_2}{\sqrt{K}} \qquad \overline{x} = \frac{1}{K}\sum_{k=0}^{K-1} x_i$$

- $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$ is slow, but already hits the lower bound ($\mathcal{O}\left(\frac{1}{T}\right)$ in the strongly convex case).
- Proved result requires pre-defined step size strategy, which is not practical (usually one cas just use several diminishes strategies).

Convergence results

Theorem

Let f be a convex G-Lipschitz function. For a fixed step size $\alpha = \frac{\|x_0 - x^*\|_2}{G}\sqrt{\frac{1}{K}}$, subgradient method satisfies

$$f(\overline{x}) - f^* \leq \frac{G\|x_0 - x^*\|_2}{\sqrt{K}} \qquad \overline{x} = \frac{1}{K}\sum_{k=0}^{K-1} x_i$$

- $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$ is slow, but already hits the lower bound ($\mathcal{O}\left(\frac{1}{T}\right)$ in the strongly convex case).
- Proved result requires pre-defined step size strategy, which is not practical (usually one cas just use several diminishes strategies).
- There is no monotonic decrease of objective.

Convergence results

Theorem

Let f be a convex G-Lipschitz function. For a fixed step size $\alpha = \dfrac{\|x_0 - x^*\|_2}{G}\sqrt{\dfrac{1}{K}}$, subgradient method satisfies

$$
f(\overline{x}) - f^* \leq \frac{G\|x_0 - x^*\|_2}{\sqrt{K}} \qquad \overline{x} = \frac{1}{K}\sum_{k=0}^{K-1} x_i
$$

- $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$ is slow, but already hits the lower bound ($\mathcal{O}\left(\frac{1}{T}\right)$ in the strongly convex case).
- Proved result requires pre-defined step size strategy, which is not practical (usually one cas just use several diminishes strategies).
- There is no monotonic decrease of objective.
- Convergence is slower, than for the gradient descent (smooth case). However, if we will go deeply for the problem structure, we can improve convergence (proximal gradient method).

Convergence results

Theorem

Let f be a convex G-Lipschitz function and $f_k^{\text{best}} = \min_{i=1,\ldots,k} f(x^i)$. For a fixed step size α, subgradient method satisfies

$$\lim_{k\to\infty} f_k^{\text{best}} \leq f^* + \frac{G^2\alpha}{2}$$

Theorem

Let f be a convex G-Lipschitz function and $f_k^{\text{best}} = \min_{i=1,\ldots,k} f(x^i)$. For a diminishing step size α_k (square summable but not summable. Important here that step sizes go to zero, but not too fast), subgradient method satisfies

$$\lim_{k\to\infty} f_k^{\text{best}} \leq f^*$$

Applications

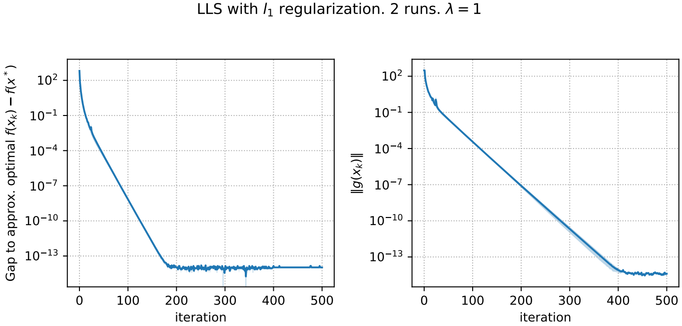
Linear Least Squares with l_1-regularization

$$\min_{x \in \mathbb{R}^n} \frac{1}{2}\|Ax - b\|_2^2 + \lambda\|x\|_1$$

Algorithm will be written as:

$$x_{k+1} = x_k - \alpha_k \left(A^\top(Ax_k - b) + \lambda \text{sign}(x_k)\right)$$

where signum function is taken element-wise.



Regularized logistic regression

Given $(x_i, y_i) \in \mathbb{R}^p \times \{0, 1\}$ for $i = 1, \ldots, n$, the logistic regression function is defined as:

$$f(\theta) = \sum_{i=1}^{n} \left(-y_i x_i^T \theta + \log(1 + \exp(x_i^T \theta)) \right)$$

This is a smooth and convex function with its gradient given by:

$$\nabla f(\theta) = \sum_{i=1}^{n} (y_i - s_i(\theta))\, x_i$$

where $s_i(\theta) = \frac{\exp(x_i^T \theta)}{1+\exp(x_i^T \theta)}$, for $i = 1, \ldots, n$. Consider the regularized problem:

$$f(\theta) + \lambda r(\theta) \to \min_{\theta}$$

where $r(\theta) = \|\theta\|_2^2$ for the ridge penalty, or $r(\theta) = \|\theta\|_1$ for the lasso penalty.

Support Vector Machines

Let $D = \{(x_i, y_i) \mid x_i \in \mathbb{R}^n, y_i \in \{\pm 1\}\}$

We need to find $\theta \in \mathbb{R}^n$ and $b \in \mathbb{R}$ such that

$$\min_{\theta \in \mathbb{R}^n, b \in \mathbb{R}} \frac{1}{2} \|\theta\|_2^2 + C \sum_{i=1}^{m} \text{max}[0, 1 - y_i(\theta^\top x_i + b)]$$

Subgradient method

Subgradient Method: $\min_{x \in \mathbb{R}^n} f(x)$ $\quad x_{k+1} = x_k - \alpha_k g_k, \quad g_k \in \partial f(x_k)$

Subgradient method

Subgradient Method: $\min_{x \in \mathbb{R}^n} f(x)$ $\quad x_{k+1} = x_k - \alpha_k g_k, \quad g_k \in \partial f(x_k)$

convex (non-smooth)	strongly convex (non-smooth)
$f(x_k) - f^* \sim \mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$	$f(x_k) - f^* \sim \mathcal{O}\left(\frac{1}{k}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon}\right)$

Subgradient method

Subgradient Method: $\min_{x \in \mathbb{R}^n} f(x)$ $\quad x_{k+1} = x_k - \alpha_k g_k, \quad g_k \in \partial f(x_k)$

convex (non-smooth)	strongly convex (non-smooth)
$f(x_k) - f^* \sim \mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$	$f(x_k) - f^* \sim \mathcal{O}\left(\frac{1}{k}\right)$
$k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$	$k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon}\right)$

Theorem

Assume that f is G-Lipschitz and convex, then Subgradient method converges as:

$$f(\overline{x}) - f^* \leq \frac{GR}{\sqrt{k}},$$

where

- $\alpha = \frac{R}{G\sqrt{k}}$

Subgradient method

Subgradient Method: $\min_{x \in \mathbb{R}^n} f(x)$ $\quad x_{k+1} = x_k - \alpha_k g_k, \quad g_k \in \partial f(x_k)$

convex (non-smooth)	strongly convex (non-smooth)
$f(x_k) - f^* \sim \mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$	$f(x_k) - f^* \sim \mathcal{O}\left(\frac{1}{k}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon}\right)$

Theorem

Assume that f is G-Lipschitz and convex, then Subgradient method converges as:

$$f(\overline{x}) - f^* \leq \frac{GR}{\sqrt{k}},$$

where

- $\alpha = \frac{R}{G\sqrt{k}}$
- $R = \|x_0 - x^*\|$

Subgradient method

Subgradient Method: $\min_{x \in \mathbb{R}^n} f(x)$ $\qquad x_{k+1} = x_k - \alpha_k g_k, \quad g_k \in \partial f(x_k)$

convex (non-smooth)	strongly convex (non-smooth)
$f(x_k) - f^* \sim \mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$	$f(x_k) - f^* \sim \mathcal{O}\left(\frac{1}{k}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\frac{1}{\varepsilon}\right)$

Theorem

Assume that f is G-Lipschitz and convex, then Subgradient method converges as:

$$f(\overline{x}) - f^* \leq \frac{GR}{\sqrt{k}},$$

where

- $\alpha = \frac{R}{G\sqrt{k}}$
- $R = \|x_0 - x^*\|$
- $\overline{x} = \frac{1}{k}\sum_{i=0}^{k-1} x_i$

Non-smooth convex optimization lower bounds

convex (non-smooth)	strongly convex (non-smooth)
$f(x_k) - f^* \sim \mathcal{O}\left(\dfrac{1}{\sqrt{k}}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon^2}\right)$	$f(x_k) - f^* \sim \mathcal{O}\left(\dfrac{1}{k}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon}\right)$

Non-smooth convex optimization lower bounds

convex (non-smooth)	strongly convex (non-smooth)
$f(x_k) - f^* \sim \mathcal{O}\left(\dfrac{1}{\sqrt{k}}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon^2}\right)$	$f(x_k) - f^* \sim \mathcal{O}\left(\dfrac{1}{k}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon}\right)$

- Subgradient method is optimal for the problems above.

Non-smooth convex optimization lower bounds

convex (non-smooth)	strongly convex (non-smooth)
$f(x_k) - f^* \sim \mathcal{O}\left(\dfrac{1}{\sqrt{k}}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon^2}\right)$	$f(x_k) - f^* \sim \mathcal{O}\left(\dfrac{1}{k}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon}\right)$

- Subgradient method is optimal for the problems above.
- One can use Mirror Descent (a generalization of the subgradient method to a possiby non-Euclidian distance) with the same convergence rate to better fit the geometry of the problem.

Non-smooth convex optimization lower bounds

convex (non-smooth)	strongly convex (non-smooth)
$f(x_k) - f^* \sim \mathcal{O}\left(\dfrac{1}{\sqrt{k}}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon^2}\right)$	$f(x_k) - f^* \sim \mathcal{O}\left(\dfrac{1}{k}\right)$ $k_\varepsilon \sim \mathcal{O}\left(\dfrac{1}{\varepsilon}\right)$

- Subgradient method is optimal for the problems above.
- One can use Mirror Descent (a generalization of the subgradient method to a possiby non-Euclidian distance) with the same convergence rate to better fit the geometry of the problem.
- However, we can achieve standard gradient descent rate $\mathcal{O}\left(\frac{1}{k}\right)$ (and even accelerated version $\mathcal{O}\left(\frac{1}{k^2}\right)$) if we will exploit the structure of the problem.

Proximal operator

Proximal mapping intuition

Consider Gradient Flow ODE:

$$\frac{dx}{dt} = -\nabla f(x)$$

Explicit Euler discretization:

Proximal mapping intuition

Consider Gradient Flow ODE:

$$\frac{dx}{dt} = -\nabla f(x)$$

Explicit Euler discretization:

$$\frac{x_{k+1} - x_k}{\alpha} = -\nabla f(x_k)$$

Leads to ordinary Gradient Descent method

Proximal mapping intuition

Consider Gradient Flow ODE:

$$\frac{dx}{dt} = -\nabla f(x)$$

Explicit Euler discretization:

$$\frac{x_{k+1} - x_k}{\alpha} = -\nabla f(x_k)$$

Leads to ordinary Gradient Descent method

Implicit Euler discretization:

$$\frac{x_{k+1} - x_k}{\alpha} = -\nabla f(x_{k+1})$$

Proximal mapping intuition

Consider Gradient Flow ODE:

$$\frac{dx}{dt} = -\nabla f(x)$$

Explicit Euler discretization:

$$\frac{x_{k+1} - x_k}{\alpha} = -\nabla f(x_k)$$

Leads to ordinary Gradient Descent method

Implicit Euler discretization:

$$\frac{x_{k+1} - x_k}{\alpha} = -\nabla f(x_{k+1})$$

$$\frac{x_{k+1} - x_k}{\alpha} + \nabla f(x_{k+1}) = 0$$

Proximal mapping intuition

Consider Gradient Flow ODE:

$$\frac{dx}{dt} = -\nabla f(x)$$

Explicit Euler discretization:

$$\frac{x_{k+1} - x_k}{\alpha} = -\nabla f(x_k)$$

Leads to ordinary Gradient Descent method

Implicit Euler discretization:

$$\frac{x_{k+1} - x_k}{\alpha} = -\nabla f(x_{k+1})$$

$$\frac{x_{k+1} - x_k}{\alpha} + \nabla f(x_{k+1}) = 0$$

$$\left.\frac{x - x_k}{\alpha} + \nabla f(x)\right|_{x = x_{k+1}} = 0$$

Proximal mapping intuition

Consider Gradient Flow ODE:

$$\frac{dx}{dt} = -\nabla f(x)$$

Explicit Euler discretization:

$$\frac{x_{k+1} - x_k}{\alpha} = -\nabla f(x_k)$$

Leads to ordinary Gradient Descent method

Implicit Euler discretization:

$$\begin{aligned}
\frac{x_{k+1} - x_k}{\alpha} &= -\nabla f(x_{k+1}) \\
\frac{x_{k+1} - x_k}{\alpha} + \nabla f(x_{k+1}) &= 0 \\
\left. \frac{x - x_k}{\alpha} + \nabla f(x) \right|_{x = x_{k+1}} &= 0 \\
\left. \nabla \left[\frac{1}{2\alpha} \|x - x_k\|_2^2 + f(x) \right] \right|_{x = x_{k+1}} &= 0
\end{aligned}$$

Proximal mapping intuition

Consider Gradient Flow ODE:

$$\frac{dx}{dt} = -\nabla f(x)$$

Explicit Euler discretization:

$$\frac{x_{k+1} - x_k}{\alpha} = -\nabla f(x_k)$$

Leads to ordinary Gradient Descent method

Implicit Euler discretization:

$$\frac{x_{k+1} - x_k}{\alpha} = -\nabla f(x_{k+1})$$

$$\frac{x_{k+1} - x_k}{\alpha} + \nabla f(x_{k+1}) = 0$$

$$\left.\frac{x - x_k}{\alpha} + \nabla f(x)\right|_{x=x_{k+1}} = 0$$

$$\left.\nabla \left[\frac{1}{2\alpha}\|x - x_k\|_2^2 + f(x)\right]\right|_{x=x_{k+1}} = 0$$

$$x_{k+1} = \text{arg}\min_{x \in \mathbb{R}^n} \left[f(x) + \frac{1}{2\alpha}\|x - x_k\|_2^2\right]$$

Proximal mapping intuition

Consider Gradient Flow ODE:

$$\frac{dx}{dt} = -\nabla f(x)$$

Explicit Euler discretization:

$$\frac{x_{k+1} - x_k}{\alpha} = -\nabla f(x_k)$$

Leads to ordinary Gradient Descent method

Implicit Euler discretization:

$$\frac{x_{k+1} - x_k}{\alpha} = -\nabla f(x_{k+1})$$

$$\frac{x_{k+1} - x_k}{\alpha} + \nabla f(x_{k+1}) = 0$$

$$\left.\frac{x - x_k}{\alpha} + \nabla f(x)\right|_{x = x_{k+1}} = 0$$

$$\left.\nabla \left[\frac{1}{2\alpha} \|x - x_k\|_2^2 + f(x) \right]\right|_{x = x_{k+1}} = 0$$

$$x_{k+1} = \text{arg}\min_{x \in \mathbb{R}^n} \left[f(x) + \frac{1}{2\alpha} \|x - x_k\|_2^2 \right]$$

Proximal mapping intuition

Consider Gradient Flow ODE:

$$\frac{dx}{dt} = -\nabla f(x)$$

Explicit Euler discretization:

$$\frac{x_{k+1} - x_k}{\alpha} = -\nabla f(x_k)$$

Leads to ordinary Gradient Descent method

Implicit Euler discretization:

$$\begin{aligned}
\frac{x_{k+1} - x_k}{\alpha} &= -\nabla f(x_{k+1}) \\
\frac{x_{k+1} - x_k}{\alpha} + \nabla f(x_{k+1}) &= 0 \\
\left.\frac{x - x_k}{\alpha} + \nabla f(x)\right|_{x = x_{k+1}} &= 0 \\
\left.\nabla \left[\frac{1}{2\alpha} \|x - x_k\|_2^2 + f(x) \right]\right|_{x = x_{k+1}} &= 0 \\
x_{k+1} = \text{arg}\min_{x \in \mathbb{R}^n} \left[f(x) + \frac{1}{2\alpha} \|x - x_k\|_2^2 \right]
\end{aligned}$$

! Proximal operator

$$\text{prox}_{f,\alpha}(x_k) = \text{arg}\min_{x \in \mathbb{R}^n} \left[f(x) + \frac{1}{2\alpha} \|x - x_k\|_2^2 \right]$$

Proximal operator visualization

$$\text{Prox}_f(x) = \underset{x'}{\text{argmin}}\ \frac{1}{2}\|x - x'\|^2 + f(x')$$

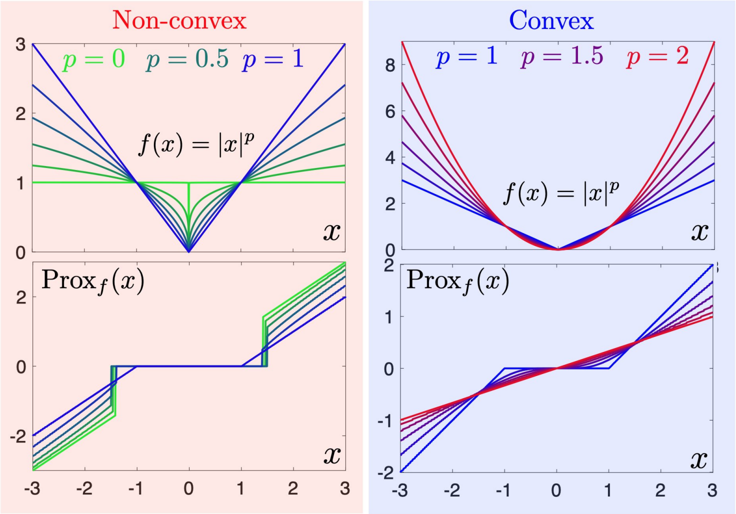


Figure 12: Source

Proximal mapping intuition

- **GD from proximal method.** Back to the discretization:

Proximal mapping intuition

- **GD from proximal method.** Back to the discretization:

$$x_{k+1} + \alpha \nabla f(x_{k+1}) = x_k$$

Proximal mapping intuition

- **GD from proximal method.** Back to the discretization:

$$
\begin{aligned}
x_{k+1} + \alpha \nabla f(x_{k+1}) &= x_k \\
(I + \alpha \nabla f)(x_{k+1}) &= x_k
\end{aligned}
$$

Proximal mapping intuition

- **GD from proximal method.** Back to the discretization:

$$
\begin{aligned}
x_{k+1} + \alpha \nabla f(x_{k+1}) &= x_k \\
(I + \alpha \nabla f)(x_{k+1}) &= x_k \\
x_{k+1} = (I + \alpha \nabla f)^{-1} x_k &\overset{\alpha \to 0}{\approx} (I - \alpha \nabla f) x_k
\end{aligned}
$$

Proximal mapping intuition

- **GD from proximal method.** Back to the discretization:

$$
\begin{aligned}
x_{k+1} + \alpha\nabla f(x_{k+1}) &= x_k \\
(I + \alpha\nabla f)(x_{k+1}) &= x_k \\
x_{k+1} = (I + \alpha\nabla f)^{-1}x_k &\overset{\alpha\to 0}{\approx} (I - \alpha\nabla f)x_k
\end{aligned}
$$

Proximal mapping intuition

- **GD from proximal method.** Back to the discretization:

$$
\begin{aligned}
x_{k+1} + \alpha \nabla f(x_{k+1}) &= x_k \\
(I + \alpha \nabla f)(x_{k+1}) &= x_k \\
x_{k+1} = (I + \alpha \nabla f)^{-1} x_k &\overset{\alpha \to 0}{\approx} (I - \alpha \nabla f) x_k
\end{aligned}
$$

  Thus, we have a usual gradient descent with $\alpha \to 0$: $x_{k+1} = x_k - \alpha \nabla f(x_k)$

- **Newton from proximal method.** Now let's consider proximal mapping of a second order Taylor approximation of the function $f^{II}_{x_k}(x)$:

Proximal mapping intuition

- **GD from proximal method.** Back to the discretization:

$$
\begin{aligned}
x_{k+1} + \alpha \nabla f(x_{k+1}) &= x_k \\
(I + \alpha \nabla f)(x_{k+1}) &= x_k \\
x_{k+1} = (I + \alpha \nabla f)^{-1} x_k &\overset{\alpha \to 0}{\approx} (I - \alpha \nabla f) x_k
\end{aligned}
$$

  Thus, we have a usual gradient descent with $\alpha \to 0$: $x_{k+1} = x_k - \alpha \nabla f(x_k)$

- **Newton from proximal method.** Now let's consider proximal mapping of a second order Taylor approximation of the function $f^{II}_{x_k}(x)$:

$$
x_{k+1} = \text{prox}_{f^{II}_{x_k}, \alpha}(x_k) = \text{arg}\min_{x \in \mathbb{R}^n} \left[f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2} \langle \nabla^2 f(x_k)(x - x_k), x - x_k \rangle + \frac{1}{2\alpha} \|x - x_k\|_2^2 \right]
$$

Proximal mapping intuition

- **GD from proximal method.** Back to the discretization:

$$
\begin{aligned}
x_{k+1} + \alpha \nabla f(x_{k+1}) &= x_k \\
(I + \alpha \nabla f)(x_{k+1}) &= x_k \\
x_{k+1} = (I + \alpha \nabla f)^{-1} x_k &\overset{\alpha \to 0}{\approx} (I - \alpha \nabla f) x_k
\end{aligned}
$$

  Thus, we have a usual gradient descent with $\alpha \to 0$: $x_{k+1} = x_k - \alpha \nabla f(x_k)$

- **Newton from proximal method.** Now let's consider proximal mapping of a second order Taylor approximation of the function $f^{II}_{x_k}(x)$:

$$
\begin{aligned}
x_{k+1} = \text{prox}_{f^{II}_{x_k}, \alpha}(x_k) &= \text{arg}\min_{x \in \mathbb{R}^n} \left[f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2} \langle \nabla^2 f(x_k)(x - x_k), x - x_k \rangle + \frac{1}{2\alpha} \|x - x_k\|_2^2 \right] \\
& \nabla f(x_k) + \nabla^2 f(x_k)(x - x_k) + \frac{1}{\alpha}(x - x_k) \Big|_{x = x_{k+1}} = 0
\end{aligned}
$$

Proximal mapping intuition

- **GD from proximal method.** Back to the discretization:

$$
\begin{aligned}
x_{k+1} + \alpha \nabla f(x_{k+1}) &= x_k \\
(I + \alpha \nabla f)(x_{k+1}) &= x_k \\
x_{k+1} = (I + \alpha \nabla f)^{-1} x_k &\overset{\alpha \to 0}{\approx} (I - \alpha \nabla f) x_k
\end{aligned}
$$

  Thus, we have a usual gradient descent with $\alpha \to 0$: $x_{k+1} = x_k - \alpha \nabla f(x_k)$

- **Newton from proximal method.** Now let's consider proximal mapping of a second order Taylor approximation of the function $f^{II}_{x_k}(x)$:

$$
\begin{aligned}
x_{k+1} = \text{prox}_{f^{II}_{x_k}, \alpha}(x_k) = \text{arg}\min_{x \in \mathbb{R}^n} \left[f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2} \langle \nabla^2 f(x_k)(x - x_k), x - x_k \rangle + \frac{1}{2\alpha} \|x - x_k\|_2^2 \right] \\
\left. \nabla f(x_k) + \nabla^2 f(x_k)(x - x_k) + \frac{1}{\alpha}(x - x_k) \right|_{x = x_{k+1}} = 0 \\
x_{k+1} = x_k - \left[\nabla^2 f(x_k) + \frac{1}{\alpha} I \right]^{-1} \nabla f(x_k)
\end{aligned}
$$

From projections to proximity

Let $\mathbb{I}_S$ be the indicator function for closed, convex S. Recall orthogonal projection $\pi_S(y)$

From projections to proximity

Let $\mathbb{I}_S$ be the indicator function for closed, convex S. Recall orthogonal projection $\pi_S(y)$

$$\pi_S(y) := \arg\min_{x \in S} \frac{1}{2}\|x - y\|_2^2.$$

From projections to proximity

Let $\mathbb{I}_S$ be the indicator function for closed, convex S. Recall orthogonal projection $\pi_S(y)$

$$\pi_S(y) := \arg\min_{x \in S} \frac{1}{2}\|x - y\|_2^2.$$

With the following notation of indicator function

$$\mathbb{I}_S(x) = \begin{cases} 0, & x \in S, \\ \infty, & x \notin S, \end{cases}$$

From projections to proximity

Let $\mathbb{I}_S$ be the indicator function for closed, convex S. Recall orthogonal projection $\pi_S(y)$

$$\pi_S(y) := \arg\min_{x \in S} \frac{1}{2}\|x - y\|_2^2.$$

With the following notation of indicator function

$$\mathbb{I}_S(x) = \begin{cases} 0, & x \in S, \\ \infty, & x \notin S, \end{cases}$$

Rewrite orthogonal projection $\pi_S(y)$ as

$$\pi_S(y) := \arg\min_{x \in \mathbb{R}^n} \frac{1}{2}\|x - y\|^2 + \mathbb{I}_S(x).$$

From projections to proximity

Let $\mathbb{I}_S$ be the indicator function for closed, convex S. Recall orthogonal projection $\pi_S(y)$

$$\pi_S(y) := \arg\min_{x \in S} \frac{1}{2}\|x - y\|_2^2.$$

With the following notation of indicator function

$$\mathbb{I}_S(x) = \begin{cases} 0, & x \in S, \\ \infty, & x \notin S, \end{cases}$$

Rewrite orthogonal projection $\pi_S(y)$ as

$$\pi_S(y) := \arg \min_{x \in \mathbb{R}^n} \frac{1}{2}\|x - y\|^2 + \mathbb{I}_S(x).$$

Proximity: Replace $\mathbb{I}_S$ by some convex function!

$$\text{prox}_r(y) = \text{prox}_{r,1}(y) := \arg\min \frac{1}{2}\|x - y\|^2 + r(x)$$

Composite optimization

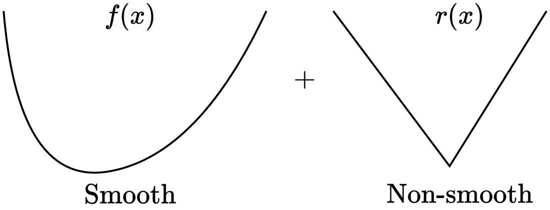
Regularized / Composite Objectives

Many nonsmooth problems take the form

$$\min_{x \in \mathbb{R}^n} \varphi(x) = f(x) + r(x)$$

- **Lasso, L1-LS, compressed sensing**

$$f(x) = \frac{1}{2}\|Ax - b\|_2^2, r(x) = \lambda\|x\|_1$$



Regularized / Composite Objectives

Many nonsmooth problems take the form

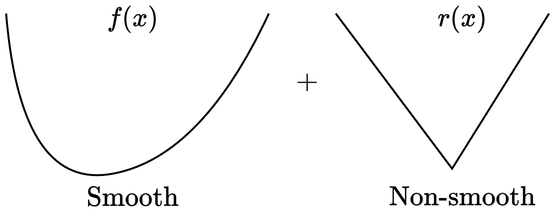
$$\min_{x \in \mathbb{R}^n} \varphi(x) = f(x) + r(x)$$

- **Lasso, L1-LS, compressed sensing**

$$f(x) = \frac{1}{2}\|Ax - b\|_2^2, r(x) = \lambda\|x\|_1$$

- **L1-Logistic regression, sparse LR**

$$f(x) = -y \log h(x) - (1-y)\log(1-h(x)), r(x) = \lambda\|x\|_1$$



Proximal mapping intuition

Optimality conditions:

$$0 \in \nabla f(x^*) + \partial r(x^*)$$

Proximal mapping intuition

Optimality conditions:

$$
\begin{aligned}
0 &\in \nabla f(x^*) + \partial r(x^*) \\
0 &\in \alpha \nabla f(x^*) + \alpha \partial r(x^*)
\end{aligned}
$$

Proximal mapping intuition

Optimality conditions:

$$
\begin{aligned}
0 &\in \nabla f(x^*) + \partial r(x^*) \\
0 &\in \alpha \nabla f(x^*) + \alpha \partial r(x^*) \\
x^* &\in \alpha \nabla f(x^*) + (I + \alpha \partial r)(x^*)
\end{aligned}
$$

Proximal mapping intuition

Optimality conditions:

$$
\begin{aligned}
0 &\in \nabla f(x^*) + \partial r(x^*) \\
0 &\in \alpha \nabla f(x^*) + \alpha \partial r(x^*) \\
x^* &\in \alpha \nabla f(x^*) + (I + \alpha \partial r)(x^*) \\
x^* - \alpha \nabla f(x^*) &\in (I + \alpha \partial r)(x^*)
\end{aligned}
$$

Proximal mapping intuition

Optimality conditions:

$$
\begin{aligned}
0 &\in \nabla f(x^*) + \partial r(x^*) \\
0 &\in \alpha \nabla f(x^*) + \alpha \partial r(x^*) \\
x^* &\in \alpha \nabla f(x^*) + (I + \alpha \partial r)(x^*) \\
x^* - \alpha \nabla f(x^*) &\in (I + \alpha \partial r)(x^*) \\
x^* &= (I + \alpha \partial r)^{-1}(x^* - \alpha \nabla f(x^*))
\end{aligned}
$$

Proximal mapping intuition

Optimality conditions:

$$
\begin{aligned}
0 &\in \nabla f(x^*) + \partial r(x^*) \\
0 &\in \alpha \nabla f(x^*) + \alpha \partial r(x^*) \\
x^* &\in \alpha \nabla f(x^*) + (I + \alpha \partial r)(x^*) \\
x^* - \alpha \nabla f(x^*) &\in (I + \alpha \partial r)(x^*) \\
x^* &= (I + \alpha \partial r)^{-1}(x^* - \alpha \nabla f(x^*)) \\
x^* &= \text{prox}_{r,\alpha}(x^* - \alpha \nabla f(x^*))
\end{aligned}
$$

Proximal mapping intuition

Optimality conditions:

$$
\begin{aligned}
0 &\in \nabla f(x^*) + \partial r(x^*) \\
0 &\in \alpha \nabla f(x^*) + \alpha \partial r(x^*) \\
x^* &\in \alpha \nabla f(x^*) + (I + \alpha \partial r)(x^*) \\
x^* - \alpha \nabla f(x^*) &\in (I + \alpha \partial r)(x^*) \\
x^* &= (I + \alpha \partial r)^{-1}(x^* - \alpha \nabla f(x^*)) \\
x^* &= \text{prox}_{r,\alpha}(x^* - \alpha \nabla f(x^*))
\end{aligned}
$$

Proximal mapping intuition

Optimality conditions:

$$
\begin{aligned}
0 &\in \nabla f(x^*) + \partial r(x^*) \\
0 &\in \alpha \nabla f(x^*) + \alpha \partial r(x^*) \\
x^* &\in \alpha \nabla f(x^*) + (I + \alpha \partial r)(x^*) \\
x^* - \alpha \nabla f(x^*) &\in (I + \alpha \partial r)(x^*) \\
x^* &= (I + \alpha \partial r)^{-1}(x^* - \alpha \nabla f(x^*)) \\
x^* &= \text{prox}_{r,\alpha}(x^* - \alpha \nabla f(x^*))
\end{aligned}
$$

Which leads to the proximal gradient method:

$$
x_{k+1} = \text{prox}_{r,\alpha}(x_k - \alpha \nabla f(x_k))
$$

And this method converges at a rate of $\mathcal{O}(\frac{1}{k})$!

Proximal mapping intuition

Optimality conditions:

$$
\begin{aligned}
0 &\in \nabla f(x^*) + \partial r(x^*) \\
0 &\in \alpha \nabla f(x^*) + \alpha \partial r(x^*) \\
x^* &\in \alpha \nabla f(x^*) + (I + \alpha \partial r)(x^*) \\
x^* - \alpha \nabla f(x^*) &\in (I + \alpha \partial r)(x^*) \\
x^* &= (I + \alpha \partial r)^{-1}(x^* - \alpha \nabla f(x^*)) \\
x^* &= \text{prox}_{r,\alpha}(x^* - \alpha \nabla f(x^*))
\end{aligned}
$$

Which leads to the proximal gradient method:

$$
x_{k+1} = \text{prox}_{r,\alpha}(x_k - \alpha \nabla f(x_k))
$$

And this method converges at a rate of $\mathcal{O}(\frac{1}{k})$!

Another form of proximal operator

$$
\text{prox}_{f,\alpha}(x_k) = \text{prox}_{\alpha f}(x_k) = \text{arg}\min_{x \in \mathbb{R}^n} \left[\alpha f(x) + \frac{1}{2}\|x - x_k\|_2^2 \right] \qquad \text{prox}_{f}(x_k) = \text{arg}\min_{x \in \mathbb{R}^n} \left[f(x) + \frac{1}{2}\|x - x_k\|_2^2 \right]
$$

Proximal operators examples

- $r(x) = \lambda \|x\|_1$, $\lambda > 0$

$$[\text{prox}_r(x)]_i = [|x_i| - \lambda]_+ \cdot \text{sign}(x_i),$$

which is also known as soft-thresholding operator.

Proximal operators examples

- $r(x) = \lambda \|x\|_1$, $\lambda > 0$

$$[\text{prox}_r(x)]_i = [|x_i| - \lambda]_+ \cdot \text{sign}(x_i),$$

  which is also known as soft-thresholding operator.
- $r(x) = \frac{\lambda}{2}\|x\|_2^2$, $\lambda > 0$

$$\text{prox}_r(x) = \frac{x}{1+\lambda}.$$

Proximal operators examples

- $r(x) = \lambda \|x\|_1$, $\lambda > 0$

$$
[\text{prox}_r(x)]_i = \left[|x_i| - \lambda\right]_+ \cdot \text{sign}(x_i),
$$

  which is also known as soft-thresholding operator.
- $r(x) = \frac{\lambda}{2}\|x\|_2^2$, $\lambda > 0$

$$
\text{prox}_r(x) = \frac{x}{1+\lambda}.
$$

- $r(x) = \mathbb{I}_S(x)$.

$$
\text{prox}_r(x_k - \alpha \nabla f(x_k)) = \text{proj}_r(x_k - \alpha \nabla f(x_k))
$$

Proximal Gradient Method. Convex case

Convergence

Theorem

Consider the proximal gradient method

$$x_{k+1} = \text{prox}_{\alpha r}\left(x_k - \alpha \nabla f(x_k)\right)$$

For the criterion $\varphi(x) = f(x) + r(x)$, we assume:

- f is convex, differentiable, $\text{dom}(f) = \mathbb{R}^n$, and ∇f is Lipschitz continuous with constant $L > 0$.
- r is convex, and $\text{prox}_{\alpha r}(x_k) = \text{arg}\min\limits_{x \in \mathbb{R}^n} \left[\alpha r(x) + \frac{1}{2}\|x - x_k\|_2^2\right]$ can be evaluated.

Proximal gradient descent with fixed step size $\alpha = 1/L$ satisfies

$$\varphi(x_k) - \varphi^* \leq \frac{L\|x_0 - x^*\|^2}{2k},$$

Proximal gradient descent has a convergence rate of $O(1/k)$ or $O(1/\varepsilon)$. This matches the gradient descent rate! (But remember the proximal operation cost)

Proximal Gradient Method. Strongly convex case

Convergence

Theorem

Consider the proximal gradient method

$$x_{k+1} = \text{prox}_{\alpha r}\left(x_k - \alpha \nabla f(x_k)\right)$$

For the criterion $\varphi(x) = f(x) + r(x)$, we assume:

- f is μ-strongly convex, differentiable, $\text{dom}(f) = \mathbb{R}^n$, and ∇f is Lipschitz continuous with constant $L > 0$.
- r is convex, and $\text{prox}_{\alpha r}(x_k) = \arg\min\limits_{x \in \mathbb{R}^n} \left[\alpha r(x) + \frac{1}{2}\|x - x_k\|_2^2\right]$ can be evaluated.

Proximal gradient descent with fixed step size $\alpha \leq 1/L$ satisfies

$$\|x_{k+1} - x^*\|_2^2 \leq (1 - \alpha\mu)^k \|x_0 - x^*\|_2^2$$

This is exactly gradient descent convergence rate. Note, that the original problem is even non-smooth!

Accelerated Proximal Method

Accelerated Proximal Method

Let $x_0 = y_0 \in \text{dom}(r)$. For $k \geq 1$:

$$x_k = \text{prox}_{\alpha_k h}(y_{k-1} - \alpha_k \nabla f(y_{k-1}))$$
$$y_k = x_k + \frac{k-1}{k+2}(x_k - x_{k-1})$$

Achieves

$$\varphi(x_k) - \varphi^* \leq \frac{2L\|x_0 - x^*\|^2}{k^2}.$$

Accelerated Proximal Method

Accelerated Proximal Method

Let $x_0 = y_0 \in \text{dom}(r)$. For $k \geq 1$:

$$
\begin{aligned}
x_k &= \text{prox}_{\alpha_k h}(y_{k-1} - \alpha_k \nabla f(y_{k-1})) \\
y_k &= x_k + \frac{k-1}{k+2}(x_k - x_{k-1})
\end{aligned}
$$

Achieves

$$
\varphi(x_k) - \varphi^* \leq \frac{2L\|x_0 - x^*\|^2}{k^2}.
$$

Framework due to: Nesterov (1983, 2004); also Beck, Teboulle (2009). Simplified analysis: Tseng (2008).

- Uses extra “memory” for interpolation

Accelerated Proximal Method

Accelerated Proximal Method

Let $x_0 = y_0 \in \text{dom}(r)$. For $k \geq 1$:

$$x_k = \text{prox}_{\alpha_k h}(y_{k-1} - \alpha_k \nabla f(y_{k-1}))$$
$$y_k = x_k + \frac{k-1}{k+2}(x_k - x_{k-1})$$

Achieves

$$\varphi(x_k) - \varphi^* \leq \frac{2L\|x_0 - x^*\|^2}{k^2}.$$

Framework due to: Nesterov (1983, 2004); also Beck, Teboulle (2009). Simplified analysis: Tseng (2008).

- Uses extra “memory” for interpolation
- Same computational cost as ordinary prox-grad

Accelerated Proximal Method

Accelerated Proximal Method

Let $x_0 = y_0 \in \text{dom}(r)$. For $k \geq 1$:

$$
\begin{aligned}
x_k &= \text{prox}_{\alpha_k h}(y_{k-1} - \alpha_k \nabla f(y_{k-1})) \\
y_k &= x_k + \frac{k-1}{k+2}(x_k - x_{k-1})
\end{aligned}
$$

Achieves

$$
\varphi(x_k) - \varphi^* \leq \frac{2L\|x_0 - x^*\|^2}{k^2}.
$$

Framework due to: Nesterov (1983, 2004); also Beck, Teboulle (2009). Simplified analysis: Tseng (2008).

- Uses extra "memory" for interpolation
- Same computational cost as ordinary prox-grad
- Convergence rate theoretically optimal

Example: ISTA

Iterative Shrinkage-Thresholding Algorithm (ISTA)

ISTA is a popular method for solving optimization problems involving L1 regularization, such as Lasso. It combines gradient descent with a shrinkage operator to handle the non-smooth L1 penalty effectively.

- **Algorithm**:

Example: ISTA

Iterative Shrinkage-Thresholding Algorithm (ISTA)

ISTA is a popular method for solving optimization problems involving L1 regularization, such as Lasso. It combines gradient descent with a shrinkage operator to handle the non-smooth L1 penalty effectively.

- **Algorithm**:
  - Given x_0, for $k \geq 0$, repeat:

$$x_{k+1} = \text{prox}_{\lambda\alpha\|\cdot\|_1}\left(x_k - \alpha\nabla f(x_k)\right),$$

where $\text{prox}_{\lambda\alpha\|\cdot\|_1}(v)$ applies soft thresholding to each component of v.

Example: ISTA

Iterative Shrinkage-Thresholding Algorithm (ISTA)

ISTA is a popular method for solving optimization problems involving L1 regularization, such as Lasso. It combines gradient descent with a shrinkage operator to handle the non-smooth L1 penalty effectively.

- **Algorithm**:
  - Given x_0, for $k \geq 0$, repeat:

$$x_{k+1} = \text{prox}_{\lambda\alpha\|\cdot\|_1}\left(x_k - \alpha\nabla f(x_k)\right),$$

    where $\text{prox}_{\lambda\alpha\|\cdot\|_1}(v)$ applies soft thresholding to each component of v.
- **Convergence**:

Example: ISTA

Iterative Shrinkage-Thresholding Algorithm (ISTA)

ISTA is a popular method for solving optimization problems involving L1 regularization, such as Lasso. It combines gradient descent with a shrinkage operator to handle the non-smooth L1 penalty effectively.

- **Algorithm**:
  - Given x_0, for $k \geq 0$, repeat:

$$x_{k+1} = \text{prox}_{\lambda\alpha\|\cdot\|_1}\left(x_k - \alpha \nabla f(x_k)\right),$$

  where $\text{prox}_{\lambda\alpha\|\cdot\|_1}(v)$ applies soft thresholding to each component of v.
- **Convergence**:
  - Converges at a rate of $O(1/k)$ for suitable step size α.

Example: ISTA

Iterative Shrinkage-Thresholding Algorithm (ISTA)

ISTA is a popular method for solving optimization problems involving L1 regularization, such as Lasso. It combines gradient descent with a shrinkage operator to handle the non-smooth L1 penalty effectively.

- **Algorithm**:
  - Given x_0, for $k \geq 0$, repeat:
  $$x_{k+1} = \text{prox}_{\lambda\alpha\|\cdot\|_1}\left(x_k - \alpha \nabla f(x_k)\right),$$
  where $\text{prox}_{\lambda\alpha\|\cdot\|_1}(v)$ applies soft thresholding to each component of v.
- **Convergence**:
  - Converges at a rate of $O(1/k)$ for suitable step size α.
- **Application**:

Example: ISTA

Iterative Shrinkage-Thresholding Algorithm (ISTA)

ISTA is a popular method for solving optimization problems involving L1 regularization, such as Lasso. It combines gradient descent with a shrinkage operator to handle the non-smooth L1 penalty effectively.

- **Algorithm**:
  - Given x_0, for $k \geq 0$, repeat:

$$x_{k+1} = \text{prox}_{\lambda\alpha\|\cdot\|_1}\left(x_k - \alpha\nabla f(x_k)\right),$$

  where $\text{prox}_{\lambda\alpha\|\cdot\|_1}(v)$ applies soft thresholding to each component of v.
- **Convergence**:
  - Converges at a rate of $O(1/k)$ for suitable step size α.
- **Application**:
  - Efficient for sparse signal recovery, image processing, and compressed sensing.

Example: FISTA

Fast Iterative Shrinkage-Thresholding Algorithm (FISTA)

FISTA improves upon ISTA's convergence rate by incorporating a momentum term, inspired by Nesterov's accelerated gradient method.

- **Algorithm**:

Example: FISTA

Fast Iterative Shrinkage-Thresholding Algorithm (FISTA)

FISTA improves upon ISTA's convergence rate by incorporating a momentum term, inspired by Nesterov's accelerated gradient method.

- **Algorithm**:
  - Initialize $x_0 = y_0$, $t_0 = 1$.

Example: FISTA

Fast Iterative Shrinkage-Thresholding Algorithm (FISTA)

FISTA improves upon ISTA's convergence rate by incorporating a momentum term, inspired by Nesterov's accelerated gradient method.

- **Algorithm**:
  - Initialize $x_0 = y_0$, $t_0 = 1$.
  - For $k \geq 1$, update:

$$
\begin{aligned}
x_k &= \text{prox}_{\lambda\alpha\|\cdot\|_1}\left(y_{k-1} - \alpha\nabla f(y_{k-1})\right), \\
t_k &= \frac{1 + \sqrt{1 + 4t_{k-1}^2}}{2}, \\
y_k &= x_k + \frac{t_{k-1} - 1}{t_k}(x_k - x_{k-1}).
\end{aligned}
$$

Example: FISTA

Fast Iterative Shrinkage-Thresholding Algorithm (FISTA)

FISTA improves upon ISTA's convergence rate by incorporating a momentum term, inspired by Nesterov's accelerated gradient method.

- **Algorithm**:
  - Initialize $x_0 = y_0$, $t_0 = 1$.
  - For $k \geq 1$, update:

$$
\begin{aligned}
x_k &= \text{prox}_{\lambda\alpha\|\cdot\|_1}\left(y_{k-1} - \alpha \nabla f(y_{k-1})\right), \\
t_k &= \frac{1 + \sqrt{1 + 4t_{k-1}^2}}{2}, \\
y_k &= x_k + \frac{t_{k-1} - 1}{t_k}(x_k - x_{k-1}).
\end{aligned}
$$

- **Convergence**:

Example: FISTA

Fast Iterative Shrinkage-Thresholding Algorithm (FISTA)

FISTA improves upon ISTA's convergence rate by incorporating a momentum term, inspired by Nesterov's accelerated gradient method.

- **Algorithm**:
  - Initialize $x_0 = y_0$, $t_0 = 1$.
  - For $k \geq 1$, update:

$$
\begin{aligned}
x_k &= \text{prox}_{\lambda\alpha\|\cdot\|_1}\left(y_{k-1} - \alpha\nabla f(y_{k-1})\right), \\
t_k &= \frac{1 + \sqrt{1 + 4t_{k-1}^2}}{2}, \\
y_k &= x_k + \frac{t_{k-1} - 1}{t_k}(x_k - x_{k-1}).
\end{aligned}
$$

- **Convergence**:
  - Improves the convergence rate to $O(1/k^2)$.

Example: FISTA

Fast Iterative Shrinkage-Thresholding Algorithm (FISTA)

FISTA improves upon ISTA's convergence rate by incorporating a momentum term, inspired by Nesterov's accelerated gradient method.

- **Algorithm**:
  - Initialize $x_0 = y_0$, $t_0 = 1$.
  - For $k \geq 1$, update:

$$
\begin{aligned}
x_k &= \text{prox}_{\lambda\alpha\|\cdot\|_1}\left(y_{k-1} - \alpha \nabla f(y_{k-1})\right), \\
t_k &= \frac{1 + \sqrt{1 + 4t_{k-1}^2}}{2}, \\
y_k &= x_k + \frac{t_{k-1} - 1}{t_k}(x_k - x_{k-1}).
\end{aligned}
$$

- **Convergence**:
  - Improves the convergence rate to $O(1/k^2)$.
- **Application**:

Example: FISTA

Fast Iterative Shrinkage-Thresholding Algorithm (FISTA)

FISTA improves upon ISTA's convergence rate by incorporating a momentum term, inspired by Nesterov's accelerated gradient method.

- **Algorithm**:
  - Initialize $x_0 = y_0$, $t_0 = 1$.
  - For $k \geq 1$, update:

$$
\begin{aligned}
x_k &= \text{prox}_{\lambda\alpha\|\cdot\|_1}\left(y_{k-1} - \alpha\nabla f(y_{k-1})\right), \\
t_k &= \frac{1 + \sqrt{1 + 4t_{k-1}^2}}{2}, \\
y_k &= x_k + \frac{t_{k-1} - 1}{t_k}(x_k - x_{k-1}).
\end{aligned}
$$

- **Convergence**:
  - Improves the convergence rate to $O(1/k^2)$.
- **Application**:
  - Especially useful for large-scale problems in machine learning and signal processing where the L1 penalty induces sparsity.

Example: Matrix Completion

Solving the Matrix Completion Problem

Matrix completion problems seek to fill in the missing entries of a partially observed matrix under certain assumptions, typically low-rank. This can be formulated as a minimization problem involving the nuclear norm (sum of singular values), which promotes low-rank solutions.

- **Problem Formulation**:

$$\min_X \frac{1}{2}\|P_\Omega(X) - P_\Omega(M)\|_F^2 + \lambda\|X\|_*,$$

  where P_Ω projects onto the observed set Ω, and $\|\cdot\|_*$ denotes the nuclear norm.

Example: Matrix Completion

Solving the Matrix Completion Problem

Matrix completion problems seek to fill in the missing entries of a partially observed matrix under certain assumptions, typically low-rank. This can be formulated as a minimization problem involving the nuclear norm (sum of singular values), which promotes low-rank solutions.

- **Problem Formulation**:

$$\min_X \frac{1}{2}\|P_\Omega(X) - P_\Omega(M)\|_F^2 + \lambda\|X\|_*,$$

  where P_Ω projects onto the observed set Ω, and $\|\cdot\|_*$ denotes the nuclear norm.

- **Proximal Operator**:

Example: Matrix Completion

Solving the Matrix Completion Problem

Matrix completion problems seek to fill in the missing entries of a partially observed matrix under certain assumptions, typically low-rank. This can be formulated as a minimization problem involving the nuclear norm (sum of singular values), which promotes low-rank solutions.

- **Problem Formulation**:
$$\min_X \frac{1}{2}\|P_\Omega(X) - P_\Omega(M)\|_F^2 + \lambda\|X\|_*,$$
where P_Ω projects onto the observed set Ω, and $\|\cdot\|_*$ denotes the nuclear norm.
- **Proximal Operator**:
    - The proximal operator for the nuclear norm involves singular value decomposition (SVD) and soft-thresholding of the singular values.

Example: Matrix Completion

Solving the Matrix Completion Problem

Matrix completion problems seek to fill in the missing entries of a partially observed matrix under certain assumptions, typically low-rank. This can be formulated as a minimization problem involving the nuclear norm (sum of singular values), which promotes low-rank solutions.

- **Problem Formulation**:

$$\min_X \frac{1}{2}\|P_\Omega(X) - P_\Omega(M)\|_F^2 + \lambda\|X\|_*,$$

  where P_Ω projects onto the observed set Ω, and $\|\cdot\|_*$ denotes the nuclear norm.
- **Proximal Operator**:
  - The proximal operator for the nuclear norm involves singular value decomposition (SVD) and soft-thresholding of the singular values.
- **Algorithm**:

Example: Matrix Completion

Solving the Matrix Completion Problem

Matrix completion problems seek to fill in the missing entries of a partially observed matrix under certain assumptions, typically low-rank. This can be formulated as a minimization problem involving the nuclear norm (sum of singular values), which promotes low-rank solutions.

- **Problem Formulation**:
$$\min_X \frac{1}{2}\|P_\Omega(X) - P_\Omega(M)\|_F^2 + \lambda\|X\|_*,$$
where P_Ω projects onto the observed set Ω, and $\|\cdot\|_*$ denotes the nuclear norm.
- **Proximal Operator**:
  - The proximal operator for the nuclear norm involves singular value decomposition (SVD) and soft-thresholding of the singular values.
- **Algorithm**:
  - Similar proximal gradient or accelerated proximal gradient methods can be applied, where the main computational effort lies in performing partial SVDs.

Example: Matrix Completion

Solving the Matrix Completion Problem

Matrix completion problems seek to fill in the missing entries of a partially observed matrix under certain assumptions, typically low-rank. This can be formulated as a minimization problem involving the nuclear norm (sum of singular values), which promotes low-rank solutions.

- **Problem Formulation**:
$$\min_X \frac{1}{2}\|P_\Omega(X) - P_\Omega(M)\|_F^2 + \lambda\|X\|_*,$$
where P_Ω projects onto the observed set Ω, and $\|\cdot\|_*$ denotes the nuclear norm.
- **Proximal Operator**:
  - The proximal operator for the nuclear norm involves singular value decomposition (SVD) and soft-thresholding of the singular values.
- **Algorithm**:
  - Similar proximal gradient or accelerated proximal gradient methods can be applied, where the main computational effort lies in performing partial SVDs.
- **Application**:

Example: Matrix Completion

Solving the Matrix Completion Problem

Matrix completion problems seek to fill in the missing entries of a partially observed matrix under certain assumptions, typically low-rank. This can be formulated as a minimization problem involving the nuclear norm (sum of singular values), which promotes low-rank solutions.

- **Problem Formulation**:

$$\min_X \frac{1}{2}\|P_\Omega(X) - P_\Omega(M)\|_F^2 + \lambda\|X\|_*,$$

  where P_Ω projects onto the observed set Ω, and $\|\cdot\|_*$ denotes the nuclear norm.
- **Proximal Operator**:
  - The proximal operator for the nuclear norm involves singular value decomposition (SVD) and soft-thresholding of the singular values.
- **Algorithm**:
  - Similar proximal gradient or accelerated proximal gradient methods can be applied, where the main computational effort lies in performing partial SVDs.
- **Application**:
  - Widely used in recommender systems, image recovery, and other domains where data is naturally matrix-formed but partially observed.

Summary

- If we exploit the structure of the problem, we may beat the lower bounds for the unstructured problem.

Summary

- If we exploit the structure of the problem, we may beat the lower bounds for the unstructured problem.
- Proximal gradient method for a composite problem with an L-smooth convex function f and a convex proximal friendly function r has the same convergence as the gradient descent method for the function f. The smoothness/non-smoothness properties of r do not affect convergence.

Summary

- If we exploit the structure of the problem, we may beat the lower bounds for the unstructured problem.
- Proximal gradient method for a composite problem with an L-smooth convex function f and a convex proximal friendly function r has the same convergence as the gradient descent method for the function f. The smoothness/non-smoothness properties of r do not affect convergence.
- It seems that by putting $f = 0$, any nonsmooth problem can be solved using such a method. Question: is this true?

Summary

- If we exploit the structure of the problem, we may beat the lower bounds for the unstructured problem.
- Proximal gradient method for a composite problem with an L-smooth convex function f and a convex proximal friendly function r has the same convergence as the gradient descent method for the function f. The smoothness/non-smoothness properties of r do not affect convergence.
- It seems that by putting $f = 0$, any nonsmooth problem can be solved using such a method. Question: is this true?

Summary

- If we exploit the structure of the problem, we may beat the lower bounds for the unstructured problem.
- Proximal gradient method for a composite problem with an L-smooth convex function f and a convex proximal friendly function r has the same convergence as the gradient descent method for the function f. The smoothness/non-smoothness properties of r do not affect convergence.
- It seems that by putting $f = 0$, any nonsmooth problem can be solved using such a method. Question: is this true?

  If we allow the proximal operator to be inexact (numerically), then it is true that we can solve any nonsmooth optimization problem. But this is not better from the point of view of theory than solving the problem by subgradient descent, because some auxiliary method (for example, the same subgradient descent) is used to solve the proximal subproblem.
- Proximal method is a general modern framework for many numerical methods. Further development includes accelerated, stochastic, primal-dual modifications and etc.

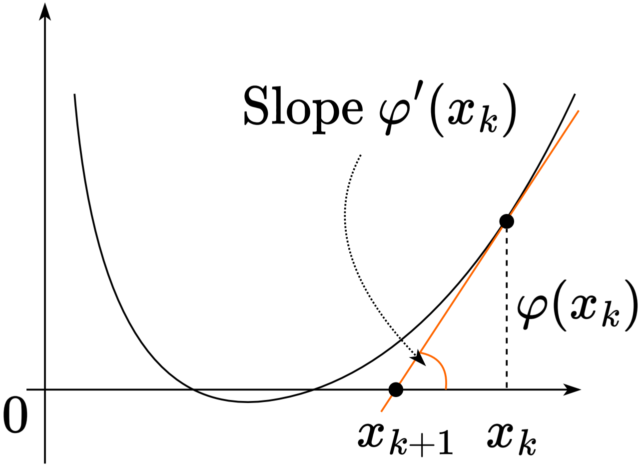
Summary

- If we exploit the structure of the problem, we may beat the lower bounds for the unstructured problem.
- Proximal gradient method for a composite problem with an L-smooth convex function f and a convex proximal friendly function r has the same convergence as the gradient descent method for the function f. The smoothness/non-smoothness properties of r do not affect convergence.
- It seems that by putting $f = 0$, any nonsmooth problem can be solved using such a method. Question: is this true?

  If we allow the proximal operator to be inexact (numerically), then it is true that we can solve any nonsmooth optimization problem. But this is not better from the point of view of theory than solving the problem by subgradient descent, because some auxiliary method (for example, the same subgradient descent) is used to solve the proximal subproblem.
- Proximal method is a general modern framework for many numerical methods. Further development includes accelerated, stochastic, primal-dual modifications and etc.
- Further reading: Proximal operator splitting, Douglas-Rachford splitting, Best approximation problem, Three operator splitting.

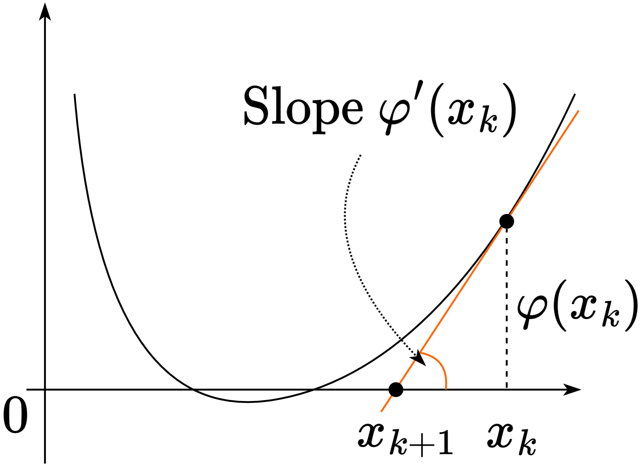
Newton method

Idea of Newton method of root finding



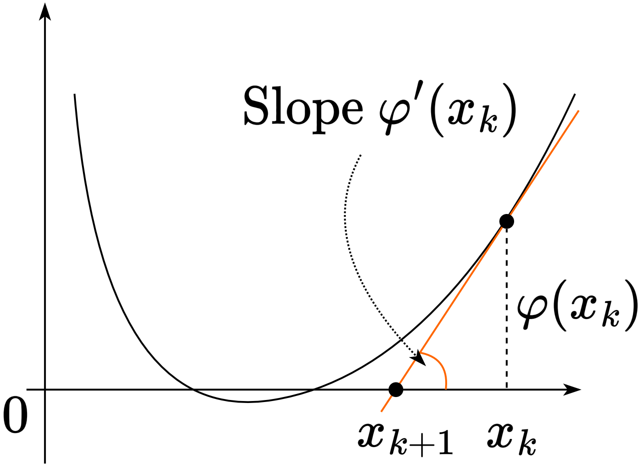
Consider the function $\varphi(x) : \mathbb{R} \to \mathbb{R}$.

Idea of Newton method of root finding



Consider the function $\varphi(x) : \mathbb{R} \to \mathbb{R}$.
The whole idea came from building a linear approximation at the point x_k and find its root, which will be the new iteration point:

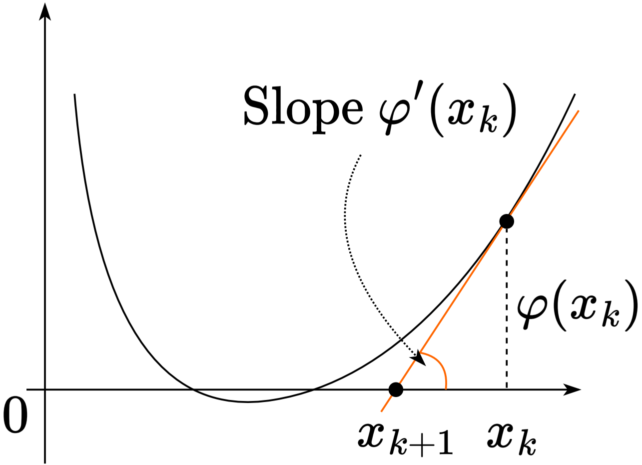
Idea of Newton method of root finding



Consider the function $\varphi(x) : \mathbb{R} \to \mathbb{R}$.
The whole idea came from building a linear approximation at the point x_k and find its root, which will be the new iteration point:

$$\varphi'(x_k) = \frac{\varphi(x_k)}{x_{k+1} - x_k}$$

Idea of Newton method of root finding

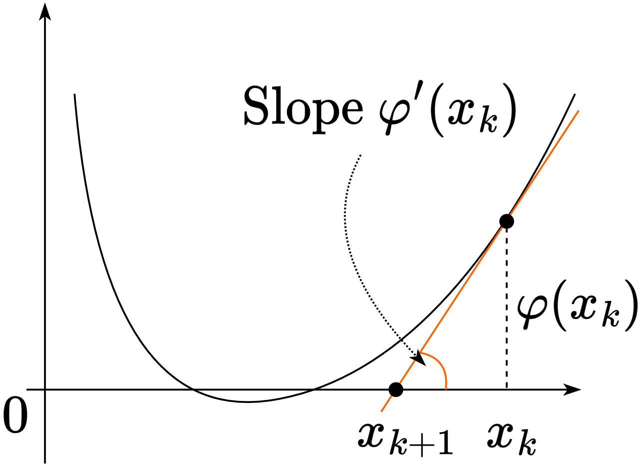


Consider the function $\varphi(x) : \mathbb{R} \to \mathbb{R}$.
The whole idea came from building a linear approximation at the point x_k and find its root, which will be the new iteration point:

$$\varphi'(x_k) = \frac{\varphi(x_k)}{x_{k+1} - x_k}$$

We get an iterative scheme:

Idea of Newton method of root finding



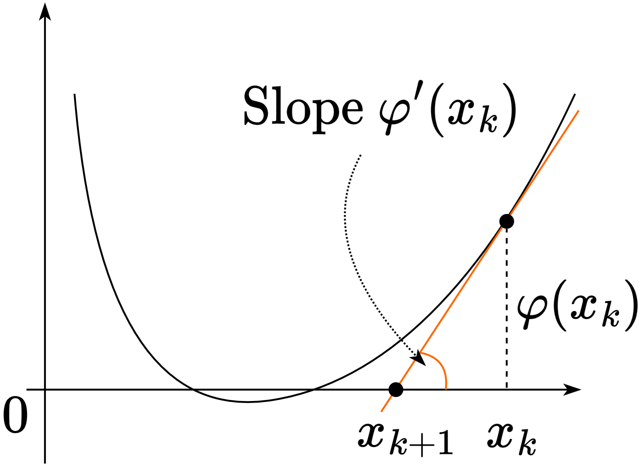
Consider the function $\varphi(x) : \mathbb{R} \to \mathbb{R}$.
The whole idea came from building a linear approximation at the point x_k and find its root, which will be the new iteration point:

$$\varphi'(x_k) = \frac{\varphi(x_k)}{x_{k+1} - x_k}$$

We get an iterative scheme:

$$x_{k+1} = x_k - \frac{\varphi(x_k)}{\varphi'(x_k)}.$$

Idea of Newton method of root finding



Consider the function $\varphi(x) : \mathbb{R} \to \mathbb{R}$.
The whole idea came from building a linear approximation at the point x_k and find its root, which will be the new iteration point:

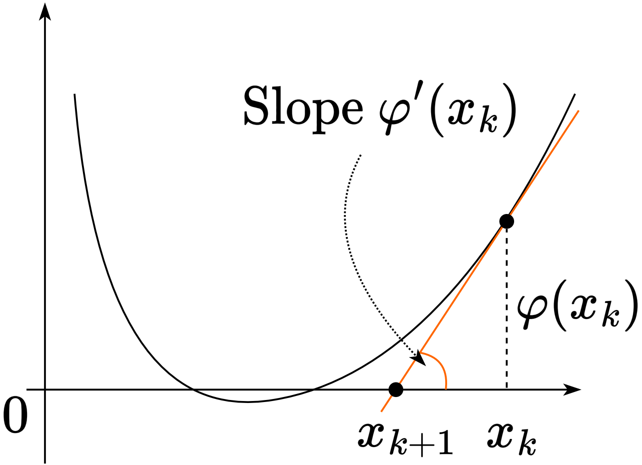
$$\varphi'(x_k) = \frac{\varphi(x_k)}{x_{k+1} - x_k}$$

We get an iterative scheme:

$$x_{k+1} = x_k - \frac{\varphi(x_k)}{\varphi'(x_k)}.$$

Which will become a Newton optimization method in case $f'(x) = \varphi(x)$[a]:

Idea of Newton method of root finding



Consider the function $\varphi(x) : \mathbb{R} \to \mathbb{R}$.
The whole idea came from building a linear approximation at the point x_k and find its root, which will be the new iteration point:

$$\varphi'(x_k) = \frac{\varphi(x_k)}{x_{k+1} - x_k}$$

We get an iterative scheme:

$$x_{k+1} = x_k - \frac{\varphi(x_k)}{\varphi'(x_k)}.$$

Which will become a Newton optimization method in case $f'(x) = \varphi(x)$[a]:

$$x_{k+1} = x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$$

[a] Literally we aim to solve the problem of finding stationary points $\nabla f(x) = 0$

Newton method as a local quadratic Taylor approximation minimizer

Let us now have the function $f(x)$ and a certain point x_k. Let us consider the quadratic approximation of this function near x_k:

Newton method as a local quadratic Taylor approximation minimizer

Let us now have the function $f(x)$ and a certain point x_k. Let us consider the quadratic approximation of this function near x_k:

$$f^{II}_{x_k}(x) = f(x_k) + \langle \nabla f(x_k), x - x_k\rangle + \frac{1}{2} \langle \nabla^2 f(x_k)(x - x_k), x - x_k\rangle.$$

Newton method as a local quadratic Taylor approximation minimizer

Let us now have the function $f(x)$ and a certain point x_k. Let us consider the quadratic approximation of this function near x_k:

$$f^{II}_{x_k}(x) = f(x_k) + \langle \nabla f(x_k), x - x_k\rangle + \frac{1}{2} \langle \nabla^2 f(x_k)(x-x_k), x-x_k \rangle.$$

The idea of the method is to find the point x_{k+1}, that minimizes the function $f^{II}_{x_k}(x)$, i.e. $\nabla f^{II}_{x_k}(x_{k+1}) = 0$.

Newton method as a local quadratic Taylor approximation minimizer

Let us now have the function $f(x)$ and a certain point x_k. Let us consider the quadratic approximation of this function near x_k:

$$f^{II}_{x_k}(x) = f(x_k) + \langle \nabla f(x_k), x - x_k\rangle + \frac{1}{2} \langle \nabla^2 f(x_k)(x-x_k), x-x_k \rangle.$$

The idea of the method is to find the point x_{k+1}, that minimizes the function $f^{II}_{x_k}(x)$, i.e. $\nabla f^{II}_{x_k}(x_{k+1}) = 0$.

$$\nabla f^{II}_{x_k}(x_{k+1}) = \nabla f(x_k) + \nabla^2 f(x_k)(x_{k+1} - x_k) = 0$$

Newton method as a local quadratic Taylor approximation minimizer

Let us now have the function $f(x)$ and a certain point x_k. Let us consider the quadratic approximation of this function near x_k:

$$f^{II}_{x_k}(x) = f(x_k) + \langle \nabla f(x_k), x - x_k\rangle + \frac{1}{2} \langle \nabla^2 f(x_k)(x-x_k), x-x_k \rangle.$$

The idea of the method is to find the point x_{k+1}, that minimizes the function $f^{II}_{x_k}(x)$, i.e. $\nabla f^{II}_{x_k}(x_{k+1}) = 0$.

$$\begin{aligned}
\nabla f^{II}_{x_k}(x_{k+1}) &= \nabla f(x_k) + \nabla^2 f(x_k)(x_{k+1} - x_k) = 0 \\
\nabla^2 f(x_k)(x_{k+1} - x_k) &= -\nabla f(x_k)
\end{aligned}$$

Newton method as a local quadratic Taylor approximation minimizer

Let us now have the function $f(x)$ and a certain point x_k. Let us consider the quadratic approximation of this function near x_k:

$$f^{II}_{x_k}(x) = f(x_k) + \langle \nabla f(x_k), x - x_k\rangle + \frac{1}{2} \langle \nabla^2 f(x_k)(x-x_k), x-x_k \rangle.$$

The idea of the method is to find the point x_{k+1}, that minimizes the function $f^{II}_{x_k}(x)$, i.e. $\nabla f^{II}_{x_k}(x_{k+1}) = 0$.

$$\begin{aligned}
\nabla f^{II}_{x_k}(x_{k+1}) &= \nabla f(x_k) + \nabla^2 f(x_k)(x_{k+1} - x_k) = 0 \\
\nabla^2 f(x_k)(x_{k+1} - x_k) &= -\nabla f(x_k) \\
\left[\nabla^2 f(x_k)\right]^{-1} \nabla^2 f(x_k)(x_{k+1} - x_k) &= -\left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)
\end{aligned}$$

Newton method as a local quadratic Taylor approximation minimizer

Let us now have the function $f(x)$ and a certain point x_k. Let us consider the quadratic approximation of this function near x_k:

$$f^{II}_{x_k}(x) = f(x_k) + \langle \nabla f(x_k), x - x_k\rangle + \frac{1}{2} \langle \nabla^2 f(x_k)(x-x_k), x-x_k \rangle.$$

The idea of the method is to find the point x_{k+1}, that minimizes the function $f^{II}_{x_k}(x)$, i.e. $\nabla f^{II}_{x_k}(x_{k+1}) = 0$.

$$\begin{aligned}
\nabla f^{II}_{x_k}(x_{k+1}) &= \nabla f(x_k) + \nabla^2 f(x_k)(x_{k+1} - x_k) = 0 \\
\nabla^2 f(x_k)(x_{k+1} - x_k) &= -\nabla f(x_k) \\
\left[\nabla^2 f(x_k)\right]^{-1} \nabla^2 f(x_k)(x_{k+1} - x_k) &= -\left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k) \\
x_{k+1} &= x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k).
\end{aligned}$$

Newton method as a local quadratic Taylor approximation minimizer

Let us now have the function $f(x)$ and a certain point x_k. Let us consider the quadratic approximation of this function near x_k:

$$f^{II}_{x_k}(x) = f(x_k) + \langle \nabla f(x_k), x - x_k\rangle + \frac{1}{2} \langle \nabla^2 f(x_k)(x-x_k), x-x_k\rangle.$$

The idea of the method is to find the point x_{k+1}, that minimizes the function $f^{II}_{x_k}(x)$, i.e. $\nabla f^{II}_{x_k}(x_{k+1}) = 0$.

$$\begin{aligned}
\nabla f^{II}_{x_k}(x_{k+1}) &= \nabla f(x_k) + \nabla^2 f(x_k)(x_{k+1} - x_k) = 0 \\
\nabla^2 f(x_k)(x_{k+1} - x_k) &= -\nabla f(x_k) \\
\left[\nabla^2 f(x_k)\right]^{-1} \nabla^2 f(x_k)(x_{k+1} - x_k) &= -\left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k) \\
x_{k+1} &= x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k).
\end{aligned}$$

Newton method as a local quadratic Taylor approximation minimizer

Let us now have the function $f(x)$ and a certain point x_k. Let us consider the quadratic approximation of this function near x_k:

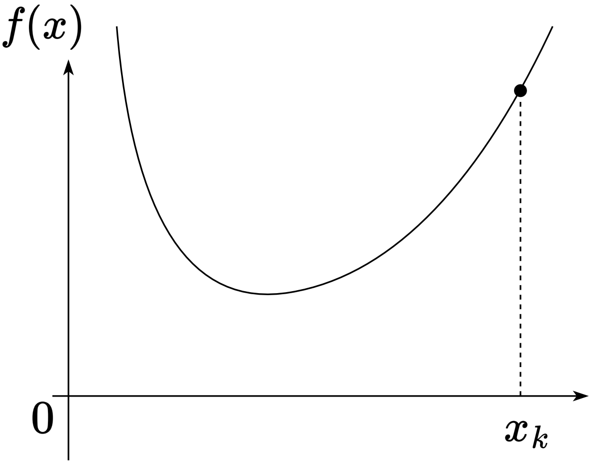
$$f^{II}_{x_k}(x) = f(x_k) + \langle \nabla f(x_k), x - x_k\rangle + \frac{1}{2} \langle \nabla^2 f(x_k)(x-x_k), x-x_k \rangle.$$

The idea of the method is to find the point x_{k+1}, that minimizes the function $f^{II}_{x_k}(x)$, i.e. $\nabla f^{II}_{x_k}(x_{k+1}) = 0$.

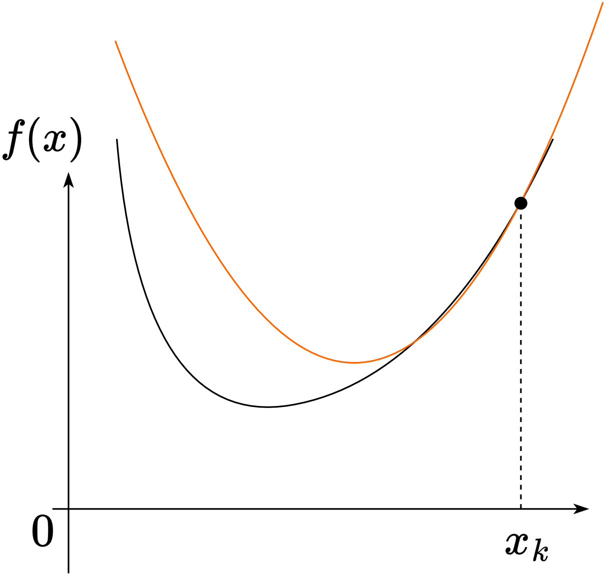
$$\begin{aligned}
\nabla f^{II}_{x_k}(x_{k+1}) &= \nabla f(x_k) + \nabla^2 f(x_k)(x_{k+1} - x_k) = 0 \\
\nabla^2 f(x_k)(x_{k+1} - x_k) &= -\nabla f(x_k) \\
\left[\nabla^2 f(x_k)\right]^{-1} \nabla^2 f(x_k)(x_{k+1} - x_k) &= -\left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k) \\
x_{k+1} &= x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k).
\end{aligned}$$

Let us immediately note the limitations related to the necessity of the Hessian's non-degeneracy (for the method to exist), as well as its positive definiteness (for the convergence guarantee).

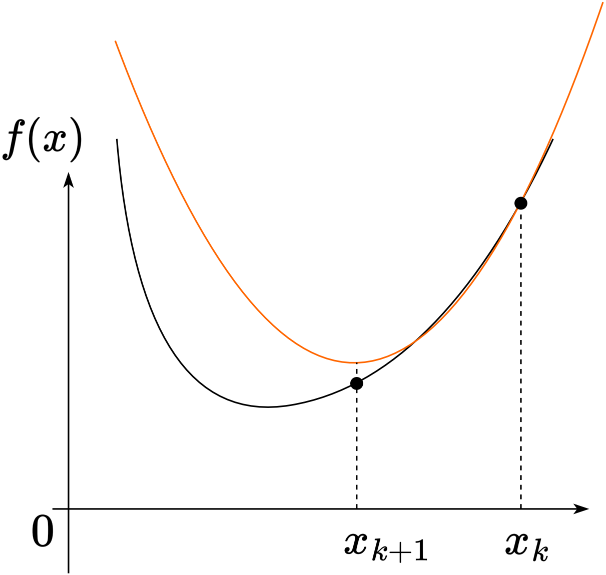
Newton method as a local quadratic Taylor approximation minimizer



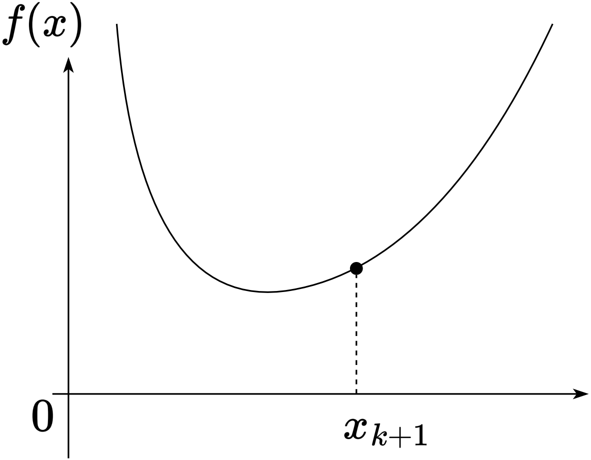
Newton method as a local quadratic Taylor approximation minimizer



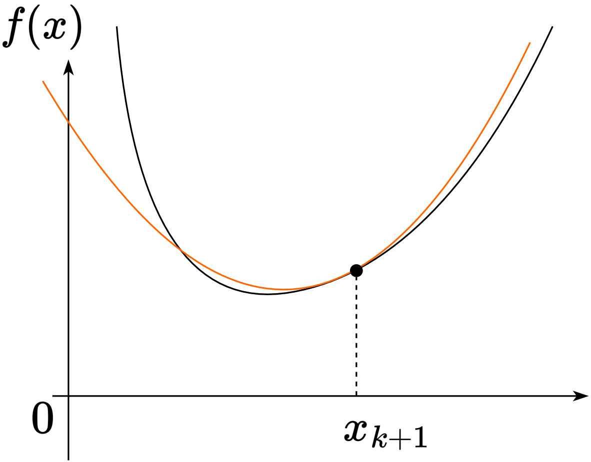
Newton method as a local quadratic Taylor approximation minimizer



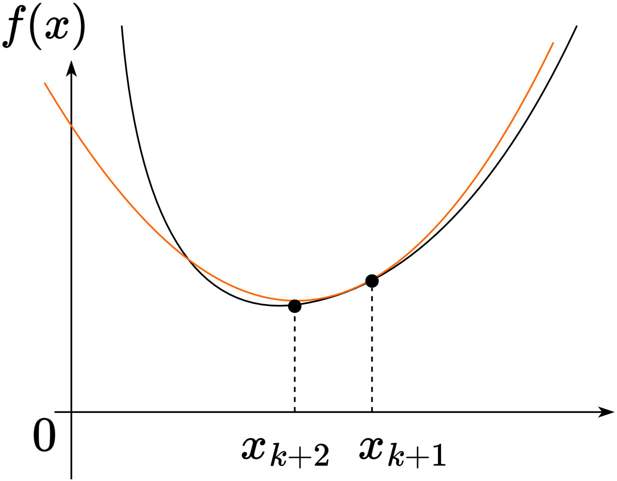
Newton method as a local quadratic Taylor approximation minimizer



Newton method as a local quadratic Taylor approximation minimizer



Newton method as a local quadratic Taylor approximation minimizer



Convergence

Theorem

Let $f(x)$ be a strongly convex twice continuously differentiable function at $\mathbb{R}^n$, for the second derivative of which inequalities are executed: $\mu I_n \preceq \nabla^2 f(x) \preceq L I_n$. Then Newton's method with a constant step locally converges to solving the problem with superlinear speed. If, in addition, Hessian is M-Lipschitz continuous, then this method converges locally to x^* at a quadratic rate.

We have an important result: Newton's method for the function with Lipschitz positive-definite Hessian converges **quadratically** near $(\|x_0 - x^*\| < \frac{2\mu}{3M})$ to the solution.

Affine Invariance of Newton's Method

An important property of Newton's method is **affine invariance**. Given a function f and a nonsingular matrix $A \in \mathbb{R}^{n \times n}$, let $x = Ay$, and define $g(y) = f(Ay)$. Note, that $\nabla g(y) = A^T \nabla f(x)$ and $\nabla^2 g(y) = A^T \nabla^2 f(x) A$. The Newton steps on g are expressed as:

$$y_{k+1} = y_k - \left(\nabla^2 g(y_k)\right)^{-1} \nabla g(y_k)$$

Affine Invariance of Newton's Method

An important property of Newton's method is **affine invariance**. Given a function f and a nonsingular matrix $A \in \mathbb{R}^{n \times n}$, let $x = Ay$, and define $g(y) = f(Ay)$. Note, that $\nabla g(y) = A^T \nabla f(x)$ and $\nabla^2 g(y) = A^T \nabla^2 f(x) A$. The Newton steps on g are expressed as:

$$
y_{k+1} = y_k - \left(\nabla^2 g(y_k)\right)^{-1} \nabla g(y_k)
$$

Expanding this, we get:

$$
y_{k+1} = y_k - \left(A^T \nabla^2 f(Ay_k) A\right)^{-1} A^T \nabla f(Ay_k)
$$

Affine Invariance of Newton's Method

An important property of Newton's method is **affine invariance**. Given a function f and a nonsingular matrix $A \in \mathbb{R}^{n \times n}$, let $x = Ay$, and define $g(y) = f(Ay)$. Note, that $\nabla g(y) = A^T \nabla f(x)$ and $\nabla^2 g(y) = A^T \nabla^2 f(x) A$. The Newton steps on g are expressed as:

$$y_{k+1} = y_k - \left(\nabla^2 g(y_k)\right)^{-1} \nabla g(y_k)$$

Expanding this, we get:

$$y_{k+1} = y_k - \left(A^T \nabla^2 f(Ay_k) A\right)^{-1} A^T \nabla f(Ay_k)$$

Using the property of matrix inverse $(AB)^{-1} = B^{-1} A^{-1}$, this simplifies to:

$$y_{k+1} = y_k - A^{-1} \left(\nabla^2 f(Ay_k)\right)^{-1} \nabla f(Ay_k)$$
$$Ay_{k+1} = Ay_k - \left(\nabla^2 f(Ay_k)\right)^{-1} \nabla f(Ay_k)$$

Affine Invariance of Newton's Method

An important property of Newton's method is **affine invariance**. Given a function f and a nonsingular matrix $A \in \mathbb{R}^{n \times n}$, let $x = Ay$, and define $g(y) = f(Ay)$. Note, that $\nabla g(y) = A^T \nabla f(x)$ and $\nabla^2 g(y) = A^T \nabla^2 f(x) A$. The Newton steps on g are expressed as:

$$y_{k+1} = y_k - \left(\nabla^2 g(y_k)\right)^{-1} \nabla g(y_k)$$

Expanding this, we get:

$$y_{k+1} = y_k - \left(A^T \nabla^2 f(Ay_k) A\right)^{-1} A^T \nabla f(Ay_k)$$

Using the property of matrix inverse $(AB)^{-1} = B^{-1} A^{-1}$, this simplifies to:

$$y_{k+1} = y_k - A^{-1} \left(\nabla^2 f(Ay_k)\right)^{-1} \nabla f(Ay_k)$$
$$Ay_{k+1} = Ay_k - \left(\nabla^2 f(Ay_k)\right)^{-1} \nabla f(Ay_k)$$

Thus, the update rule for x is:

$$x_{k+1} = x_k - \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)$$

Affine Invariance of Newton's Method

An important property of Newton's method is **affine invariance**. Given a function f and a nonsingular matrix $A \in \mathbb{R}^{n \times n}$, let $x = Ay$, and define $g(y) = f(Ay)$. Note, that $\nabla g(y) = A^T \nabla f(x)$ and $\nabla^2 g(y) = A^T \nabla^2 f(x) A$. The Newton steps on g are expressed as:

$$y_{k+1} = y_k - \left(\nabla^2 g(y_k)\right)^{-1} \nabla g(y_k)$$

Expanding this, we get:

$$y_{k+1} = y_k - \left(A^T \nabla^2 f(Ay_k) A\right)^{-1} A^T \nabla f(Ay_k)$$

Using the property of matrix inverse $(AB)^{-1} = B^{-1}A^{-1}$, this simplifies to:

$$y_{k+1} = y_k - A^{-1} \left(\nabla^2 f(Ay_k)\right)^{-1} \nabla f(Ay_k)$$

$$Ay_{k+1} = Ay_k - \left(\nabla^2 f(Ay_k)\right)^{-1} \nabla f(Ay_k)$$

Thus, the update rule for x is:

$$x_{k+1} = x_k - \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)$$

This shows that the progress made by Newton's method is independent of problem scaling. This property is not shared by the gradient descent method!

Summary

What's nice:

- quadratic convergence near the solution x^*

Summary

What's nice:

- quadratic convergence near the solution x^*
- affine invariance

Summary

What's nice:

- quadratic convergence near the solution x^*
- affine invariance
- the parameters have little effect on the convergence rate

Summary

What's nice:

- quadratic convergence near the solution x^*
- affine invariance
- the parameters have little effect on the convergence rate

Summary

What's nice:

- quadratic convergence near the solution x^*
- affine invariance
- the parameters have little effect on the convergence rate

What's not nice:

- it is necessary to store the (inverse) hessian on each iteration: $\mathcal{O}(n^2)$ memory

Summary

What's nice:

- quadratic convergence near the solution x^*
- affine invariance
- the parameters have little effect on the convergence rate

What's not nice:

- it is necessary to store the (inverse) hessian on each iteration: $\mathcal{O}(n^2)$ memory
- it is necessary to solve linear systems: $\mathcal{O}(n^3)$ operations

Summary

What's nice:

- quadratic convergence near the solution x^*
- affine invariance
- the parameters have little effect on the convergence rate

What's not nice:

- it is necessary to store the (inverse) hessian on each iteration: $\mathcal{O}(n^2)$ memory
- it is necessary to solve linear systems: $\mathcal{O}(n^3)$ operations
- the Hessian can be degenerate at x^*

Summary

What's nice:

- quadratic convergence near the solution x^*
- affine invariance
- the parameters have little effect on the convergence rate

What's not nice:

- it is necessary to store the (inverse) hessian on each iteration: $\mathcal{O}(n^2)$ memory
- it is necessary to solve linear systems: $\mathcal{O}(n^3)$ operations
- the Hessian can be degenerate at x^*
- the hessian may not be positively determined $\to$ direction $-(f''(x))^{-1}f'(x)$ may not be a descending direction

Newton method problems

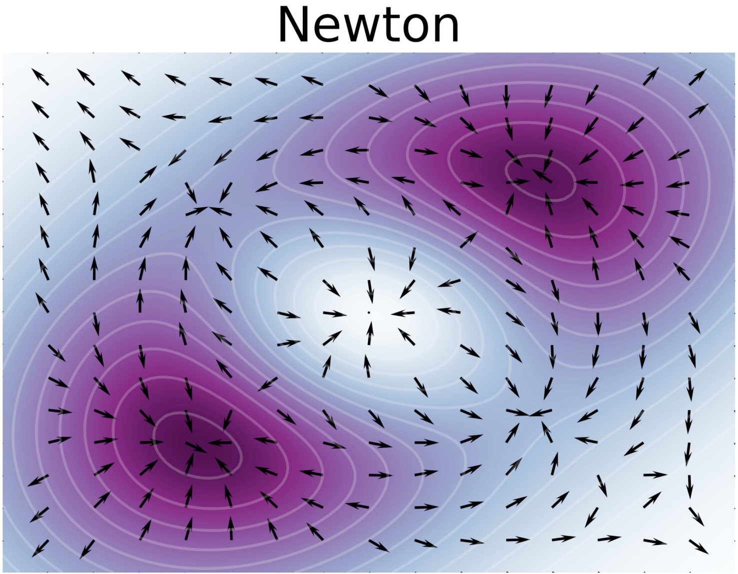


Figure 19: Animation

Newton method problems

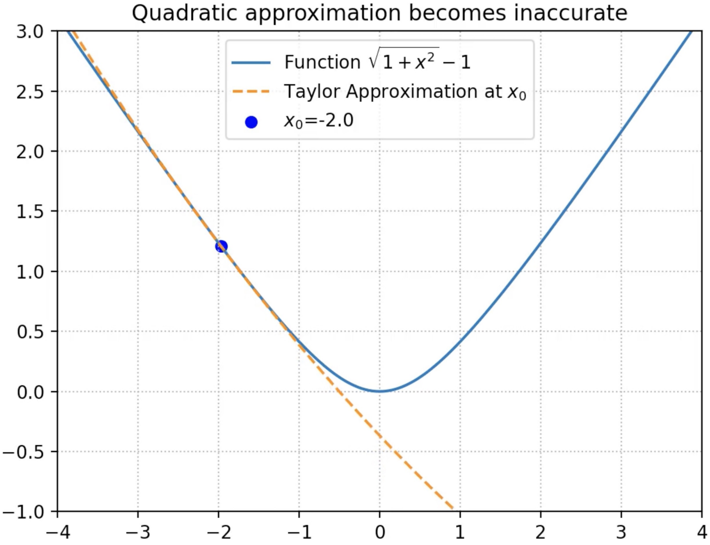


Figure 20: Animation

Quasi-Newton methods

Quasi-Newton methods intuition

For the classic task of unconditional optimization $f(x) \to \min_{x \in \mathbb{R}^n}$ the general scheme of iteration method is written as:

$$x_{k+1} = x_k + \alpha_k d_k$$

Quasi-Newton methods intuition

For the classic task of unconditional optimization $f(x) \to \min\limits_{x \in \mathbb{R}^n}$ the general scheme of iteration method is written as:

$$x_{k+1} = x_k + \alpha_k d_k$$

In the Newton method, the d_k direction (Newton's direction) is set by the linear system solution at each step:

$$B_k d_k = -\nabla f(x_k), \quad B_k = \nabla^2 f(x_k)$$

Quasi-Newton methods intuition

For the classic task of unconditional optimization $f(x) \to \min\limits_{x \in \mathbb{R}^n}$ the general scheme of iteration method is written as:

$$x_{k+1} = x_k + \alpha_k d_k$$

In the Newton method, the d_k direction (Newton's direction) is set by the linear system solution at each step:

$$B_k d_k = -\nabla f(x_k), \quad B_k = \nabla^2 f(x_k)$$

i.e. at each iteration it is necessary to **compute** hessian and gradient and **solve** linear system.

Quasi-Newton methods intuition

For the classic task of unconditional optimization $f(x) \to \min_{x \in \mathbb{R}^n}$ the general scheme of iteration method is written as:

$$x_{k+1} = x_k + \alpha_k d_k$$

In the Newton method, the d_k direction (Newton's direction) is set by the linear system solution at each step:

$$B_k d_k = -\nabla f(x_k), \quad B_k = \nabla^2 f(x_k)$$

i.e. at each iteration it is necessary to **compute** hessian and gradient and **solve** linear system.

Note here that if we take a single matrix of $B_k = I_n$ as B_k at each step, we will exactly get the gradient descent method.

The general scheme of quasi-Newton methods is based on the selection of the B_k matrix so that it tends in some sense at $k \to \infty$ to the truth value of the Hessian $\nabla^2 f(x_k)$.

Quasi-Newton Method Template

Let $x_0 \in \mathbb{R}^n$, $B_0 \succ 0$. For $k = 1, 2, 3, \ldots$, repeat:

1. Solve $B_k d_k = -\nabla f(x_k)$

Quasi-Newton Method Template

Let $x_0 \in \mathbb{R}^n$, $B_0 \succ 0$. For $k = 1, 2, 3, \ldots$, repeat:

1. Solve $B_k d_k = -\nabla f(x_k)$
2. Update $x_{k+1} = x_k + \alpha_k d_k$

Quasi-Newton Method Template

Let $x_0 \in \mathbb{R}^n$, $B_0 \succ 0$. For $k = 1, 2, 3, \ldots$, repeat:

1. Solve $B_k d_k = -\nabla f(x_k)$
2. Update $x_{k+1} = x_k + \alpha_k d_k$
3. Compute B_{k+1} from B_k

Quasi-Newton Method Template

Let $x_0 \in \mathbb{R}^n$, $B_0 \succ 0$. For $k = 1, 2, 3, \ldots$, repeat:

1. Solve $B_k d_k = -\nabla f(x_k)$
2. Update $x_{k+1} = x_k + \alpha_k d_k$
3. Compute B_{k+1} from B_k

Quasi-Newton Method Template

Let $x_0 \in \mathbb{R}^n$, $B_0 \succ 0$. For $k = 1, 2, 3, \ldots$, repeat:

1. Solve $B_k d_k = -\nabla f(x_k)$
2. Update $x_{k+1} = x_k + \alpha_k d_k$
3. Compute B_{k+1} from B_k

Different quasi-Newton methods implement Step 3 differently. As we will see, commonly we can compute $(B_{k+1})^{-1}$ from $(B_k)^{-1}$.

Quasi-Newton Method Template

Let $x_0 \in \mathbb{R}^n$, $B_0 \succ 0$. For $k = 1, 2, 3, \ldots$, repeat:

1. Solve $B_k d_k = -\nabla f(x_k)$
2. Update $x_{k+1} = x_k + \alpha_k d_k$
3. Compute B_{k+1} from B_k

Different quasi-Newton methods implement Step 3 differently. As we will see, commonly we can compute $(B_{k+1})^{-1}$ from $(B_k)^{-1}$.

Basic Idea: As B_k already contains information about the Hessian, use a suitable matrix update to form B_{k+1}.

Quasi-Newton Method Template

Let $x_0 \in \mathbb{R}^n$, $B_0 \succ 0$. For $k = 1, 2, 3, \dots$, repeat:

1. Solve $B_k d_k = -\nabla f(x_k)$
2. Update $x_{k+1} = x_k + \alpha_k d_k$
3. Compute B_{k+1} from B_k

Different quasi-Newton methods implement Step 3 differently. As we will see, commonly we can compute $(B_{k+1})^{-1}$ from $(B_k)^{-1}$.

Basic Idea: As B_k already contains information about the Hessian, use a suitable matrix update to form B_{k+1}.

Reasonable Requirement for B_{k+1} (motivated by the secant method):

$$\nabla f(x_{k+1}) - \nabla f(x_k) = B_{k+1}(x_{k+1} - x_k) = B_{k+1} d_k$$
$$\Delta y_k = B_{k+1} \Delta x_k$$

Quasi-Newton Method Template

Let $x_0 \in \mathbb{R}^n$, $B_0 \succ 0$. For $k = 1, 2, 3, \dots$, repeat:

1. Solve $B_k d_k = -\nabla f(x_k)$
2. Update $x_{k+1} = x_k + \alpha_k d_k$
3. Compute B_{k+1} from B_k

Different quasi-Newton methods implement Step 3 differently. As we will see, commonly we can compute $(B_{k+1})^{-1}$ from $(B_k)^{-1}$.

Basic Idea: As B_k already contains information about the Hessian, use a suitable matrix update to form B_{k+1}.

Reasonable Requirement for B_{k+1} (motivated by the secant method):

$$
\begin{aligned}
\nabla f(x_{k+1}) - \nabla f(x_k) = B_{k+1}(x_{k+1} - x_k) = B_{k+1} d_k \\
\Delta y_k = B_{k+1} \Delta x_k
\end{aligned}
$$

In addition to the secant equation, we want:

- B_{k+1} to be symmetric

Quasi-Newton Method Template

Let $x_0 \in \mathbb{R}^n$, $B_0 \succ 0$. For $k = 1, 2, 3, \dots$, repeat:

1. Solve $B_k d_k = -\nabla f(x_k)$
2. Update $x_{k+1} = x_k + \alpha_k d_k$
3. Compute B_{k+1} from B_k

Different quasi-Newton methods implement Step 3 differently. As we will see, commonly we can compute $(B_{k+1})^{-1}$ from $(B_k)^{-1}$.

Basic Idea: As B_k already contains information about the Hessian, use a suitable matrix update to form B_{k+1}.

Reasonable Requirement for B_{k+1} (motivated by the secant method):

$$
\begin{gathered}
\nabla f(x_{k+1}) - \nabla f(x_k) = B_{k+1}(x_{k+1} - x_k) = B_{k+1} d_k \\
\Delta y_k = B_{k+1} \Delta x_k
\end{gathered}
$$

In addition to the secant equation, we want:

- B_{k+1} to be symmetric
- B_{k+1} to be "close" to B_k

Quasi-Newton Method Template

Let $x_0 \in \mathbb{R}^n$, $B_0 \succ 0$. For $k = 1, 2, 3, \ldots$, repeat:

1. Solve $B_k d_k = -\nabla f(x_k)$
2. Update $x_{k+1} = x_k + \alpha_k d_k$
3. Compute B_{k+1} from B_k

Different quasi-Newton methods implement Step 3 differently. As we will see, commonly we can compute $(B_{k+1})^{-1}$ from $(B_k)^{-1}$.

Basic Idea: As B_k already contains information about the Hessian, use a suitable matrix update to form B_{k+1}.

Reasonable Requirement for B_{k+1} (motivated by the secant method):

$$
\nabla f(x_{k+1}) - \nabla f(x_k) = B_{k+1}(x_{k+1} - x_k) = B_{k+1} d_k
$$
$$
\Delta y_k = B_{k+1} \Delta x_k
$$

In addition to the secant equation, we want:

- B_{k+1} to be symmetric
- B_{k+1} to be "close" to B_k
- $B_k \succ 0 \Rightarrow B_{k+1} \succ 0$

Symmetric Rank-One Update

Let's try an update of the form:

$$B_{k+1} = B_k + auu^T$$

Symmetric Rank-One Update

Let's try an update of the form:

$$B_{k+1} = B_k + auu^T$$

The secant equation $B_{k+1}d_k = \Delta y_k$ yields:

$$(au^T d_k)u = \Delta y_k - B_k d_k$$

Symmetric Rank-One Update

Let's try an update of the form:

$$B_{k+1} = B_k + auu^T$$

The secant equation $B_{k+1}d_k = \Delta y_k$ yields:

$$(au^T d_k)u = \Delta y_k - B_k d_k$$

This only holds if u is a multiple of $\Delta y_k - B_k d_k$. Putting $u = \Delta y_k - B_k d_k$, we solve the above,

$$a = \frac{1}{(\Delta y_k - B_k d_k)^T d_k},$$

Symmetric Rank-One Update

Let's try an update of the form:

$$B_{k+1} = B_k + auu^T$$

The secant equation $B_{k+1}d_k = \Delta y_k$ yields:

$$(au^T d_k)u = \Delta y_k - B_k d_k$$

This only holds if u is a multiple of $\Delta y_k - B_k d_k$. Putting $u = \Delta y_k - B_k d_k$, we solve the above,

$$a = \frac{1}{(\Delta y_k - B_k d_k)^T d_k},$$

which leads to

$$B_{k+1} = B_k + \frac{(\Delta y_k - B_k d_k)(\Delta y_k - B_k d_k)^T}{(\Delta y_k - B_k d_k)^T d_k}$$

called the symmetric rank-one (SR1) update or Broyden method.

Symmetric Rank-One Update with inverse

How can we solve

$$B_{k+1}d_{k+1} = -\nabla f(x_{k+1}),$$

in order to take the next step? In addition to propagating B_k to B_{k+1}, let's propagate inverses, i.e., $C_k = B_k^{-1}$ to $C_{k+1} = (B_{k+1})^{-1}$.

Sherman-Morrison Formula:

The Sherman-Morrison formula states:

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1 + v^TA^{-1}u}$$

Thus, for the SR1 update, the inverse is also easily updated:

$$C_{k+1} = C_k + \frac{(d_k - C_k\Delta y_k)(d_k - C_k\Delta y_k)^T}{(d_k - C_k\Delta y_k)^T\Delta y_k}$$

In general, SR1 is simple and cheap, but it has a key shortcoming: it does not preserve positive definiteness.

Davidon-Fletcher-Powell Update

We could have pursued the same idea to update the inverse C:

$$C_{k+1} = C_k + auu^T + bvv^T.$$

Davidon-Fletcher-Powell Update

We could have pursued the same idea to update the inverse C:

$$C_{k+1} = C_k + auu^T + bvv^T.$$

Multiplying by Δy_k, using the secant equation $d_k = C_k \Delta y_k$, and solving for a, b, yields:

$$C_{k+1} = C_k - \frac{C_k \Delta y_k \Delta y_k^T C_k}{\Delta y_k^T C_k \Delta y_k} + \frac{d_k d_k^T}{\Delta y_k^T d_k}$$

Woodbury Formula Application

Woodbury then shows:

$$B_{k+1} = \left(I - \frac{\Delta y_k d_k^T}{\Delta y_k^T d_k} \right) B_k \left(I - \frac{d_k \Delta y_k^T}{\Delta y_k^T d_k} \right) + \frac{\Delta y_k \Delta y_k^T}{\Delta y_k^T d_k}$$

This is the Davidon-Fletcher-Powell (DFP) update. Also cheap: $O(n^2)$, preserves positive definiteness. Not as popular as BFGS.

Broyden-Fletcher-Goldfarb-Shanno update

Let's now try a rank-two update:

$$B_{k+1} = B_k + auu^T + bvv^T.$$

Broyden-Fletcher-Goldfarb-Shanno update

Let's now try a rank-two update:

$$B_{k+1} = B_k + auu^T + bvv^T.$$

The secant equation $\Delta y_k = B_{k+1} d_k$ yields:

$$\Delta y_k - B_k d_k = (au^T d_k)u + (bv^T d_k)v$$

Broyden-Fletcher-Goldfarb-Shanno update

Let's now try a rank-two update:

$$B_{k+1} = B_k + auu^T + bvv^T.$$

The secant equation $\Delta y_k = B_{k+1} d_k$ yields:

$$\Delta y_k - B_k d_k = (au^T d_k)u + (bv^T d_k)v$$

Putting $u = \Delta y_k$, $v = B_k d_k$, and solving for a, b we get:

$$B_{k+1} = B_k - \frac{B_k d_k d_k^T B_k}{d_k^T B_k d_k} + \frac{\Delta y_k \Delta y_k^T}{d_k^T \Delta y_k}$$

called the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update.

Broyden-Fletcher-Goldfarb-Shanno update with inverse

Woodbury Formula

The Woodbury formula, a generalization of the Sherman-Morrison formula, is given by:

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

Broyden-Fletcher-Goldfarb-Shanno update with inverse

Woodbury Formula

The Woodbury formula, a generalization of the Sherman-Morrison formula, is given by:

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

Applied to our case, we get a rank-two update on the inverse C:

$$C_{k+1} = C_k + \frac{(d_k - C_k\Delta y_k)d_k^T}{\Delta y_k^T d_k} + \frac{d_k(d_k - C_k\Delta y_k)^T}{\Delta y_k^T d_k} - \frac{(d_k - C_k\Delta y_k)^T\Delta y_k}{(\Delta y_k^T d_k)^2} d_k d_k^T$$

$$C_{k+1} = \left(I - \frac{d_k\Delta y_k^T}{\Delta y_k^T d_k}\right) C_k \left(I - \frac{\Delta y_k d_k^T}{\Delta y_k^T d_k}\right) + \frac{d_k d_k^T}{\Delta y_k^T d_k}$$

This formulation ensures that the BFGS update, while comprehensive, remains computationally efficient, requiring $O(n^2)$ operations. Importantly, BFGS update preserves positive definiteness. Recall this means $B_k \succ 0 \Rightarrow B_{k+1} \succ 0$. Equivalently, $C_k \succ 0 \Rightarrow C_{k+1} \succ 0$

Code

- Open In Colab

Code

- Open In Colab
- Comparison of quasi Newton methods

Code

- Open In Colab
- Comparison of quasi Newton methods
- Some practical notes about Newton method