Solving Nonnegative Least Squares by multiplicative update

Andersen Ang

Mathématique et recherche opérationnelle
UMONS, Belgium

manshun.ang@umons.ac.be Homepage: angms.science

First draft : November 30, 2018
Last update : February 19, 2020

Nonnegative Least Squares

Non**n**egative **L**east **S**quares (NNLS) : given $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, find $\mathbf{x} \in \mathbb{R}^n_+$ by solving

$$(\mathcal{P}) \ : \ \underset{\mathbf{x} \geq 0}{\operatorname{argmin}} \ f(\mathbf{x}) := \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2.$$

Let $\mathbf{Q} = \mathbf{A}^\top \mathbf{A}$, $\mathbf{p} = \mathbf{A}^\top \mathbf{b}$ and $c = \frac{1}{2}\|\mathbf{b}\|_2^2$, problem $(\mathcal{P})$ can be written as a constrained quadratic programming (QP) problem

$$\underset{\mathbf{x} \geq 0}{\operatorname{argmin}} \ f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \mathbf{p}^\top \mathbf{x} + c.$$

Due to argmin, we can ignore the constant c.

$\mathbf{x} \geq 0$ means $\mathbf{x} \in \mathbb{R}^n_+$.

Solving NNLS by Projected Gradient Descent

For $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \mathbf{p}^\top \mathbf{x}$, the gradient and the projection are

$$\nabla f = \mathbf{Q}\mathbf{x} - \mathbf{p}, \qquad \mathcal{P}_{\mathbb{R}^n_+}(\mathbf{x}) = [\mathbf{x}]_+ := \max(\mathbf{x}, 0).$$

Projected Gradient Descent (PGD) algorithm for solving NNLS is to iterate

$$\mathbf{x}_{k+1} = [\mathbf{x}_k - \alpha_k(\mathbf{Q}\mathbf{x}_k - \mathbf{p})]_+,$$

where $\alpha_k \geq 0$ is the stepsize parameter.

As we can see f is $\|\mathbf{Q}\|_2-$smooth[1], by the theory of gradient descent, we can let the stepsize α_k as $\frac{1}{\|\mathbf{Q}\|_2}$ for all k. But suppose we do not use $\frac{1}{\|\mathbf{Q}\|_2}$ as the stepsize, we pick the stepsize α_k such that the update point always stays within the feasible region, and thus projection is not required, which we get

$$\mathbf{x}^+ = \mathbf{x} - \alpha_k(\mathbf{Q}\mathbf{x} - \mathbf{p}).$$

For solving NNLS with projection, see this document.

[1] The Lipschitz constant of $\nabla f(\mathbf{x})$ is $\|\mathbf{Q}\|_2 = \sigma_{\max}(\mathbf{Q})$.

Multiplicative update

Using the componentwise stepsize $\alpha_k^{[i]} = \frac{x^{[i]}}{[\mathbf{Qx}]^{[i]}}$, where $v^{[i]}$ denotes the i-th component of the vector $\mathbf{v}$, we have

$$\begin{aligned} x_{k+1}^{[i]} &= x^{[i]} - \alpha_k^{[i]}([\mathbf{Qx}]^{[i]} - p^{[i]}) \\ &= \frac{[\mathbf{Qx}]^{[i]}}{[\mathbf{Qx}]^{[i]}} x^{[i]} - \frac{x^{[i]}}{[\mathbf{Qx}]^{[i]}}([\mathbf{Qx}]^{[i]} - p^{[i]}) \\ &= \frac{p^{[i]} x^{[i]}}{[\mathbf{Qx}]^{[i]}} \end{aligned}$$

In vector form,

$$\mathbf{x}_{k+1} = \mathbf{x}_{k+1} \star \mathbf{p} \diamond \mathbf{Qx},$$

where the multiplication $\star$ and the division $\diamond$ are elementwise. That is, they are the `.*` and `./` in `MATLAB`.

Note that as $\mathbf{p}$, $\mathbf{Q}$ and $\mathbf{x}_0$ are all nonnegative, so the iteration produce a nonnegative output.

Solving NNLS by MU algorithm

$$(\mathcal{P}) \ : \ \underset{\mathbf{x}\geq 0}{\operatorname{argmin}} \ f(\mathbf{x}) := \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2.$$

Algorithm 1: MU for NNLS

Result: A solution $\mathbf{x}$ that approximately solves $(\mathcal{P})$
Initialization Set $\mathbf{x}_0 \in \mathbb{R}_+^n$, $\mathbf{p} = \mathbf{A}^\top \mathbf{b}$, $\mathbf{Q} = \mathbf{A}^\top \mathbf{A}$, $k = 1$
while *stopping condition is not met* **do**
  $\mathbf{x}_{k+1} = \mathbf{x}_k \star \mathbf{p} \diamond \mathbf{Q}\mathbf{x}_k$
  $k = k + 1$
end

About convergence : it can be proved that, the objective function $f(\mathbf{x})$ is non-increasing under MU iteration.

Relationship between MU stepsize and PGD step size

For $x^{[i]}$, the corresponding stepsize used in MU is $\frac{x^{[i]}}{[\mathbf{Qx}]^{[i]}}$, while the stepsize used in PGD for is $\frac{1}{\|\mathbf{Q}\|_2}$.

As $[\mathbf{Qx}]^{[i]} = \mathbf{q}_i^\top \mathbf{x}$, where $\mathbf{q}_i$ is the i-th row of $\mathbf{Q}$, we have

$$[\mathbf{Qx}]^{[i]} = \mathbf{q}_i^\top \mathbf{x} \overset{\text{cs}}{\leq} \|\mathbf{q}_i\|_2 \|\mathbf{x}\|_2 = \|\mathbf{Q}\mathbf{e}_i\|_2 \|\mathbf{x}\|_2 \overset{\text{oni}}{\leq} \|\mathbf{Q}\|_2 \|\mathbf{x}\|_2,$$

where cs = Cauchy-Schwarz inequality, oni = operator norm inequality, $\mathbf{e}_i = [0 \cdots 010 \cdots 0]^\top$ for 1 at the i-th position and $\|\mathbf{e}_i\|_2 = 1$.

We have

$$\alpha_{\text{MU}} = \frac{x^{[i]}}{[\mathbf{Qx}]^{[i]}} \geq \frac{x^{[i]}}{\|\mathbf{Q}\|_2 \|\mathbf{x}\|_2} = \frac{1}{\|\mathbf{Q}\|_2} \frac{x^{[i]}}{\|\mathbf{x}\|_2} = \alpha_{\text{PGD}} \underbrace{\frac{x^{[i]}}{\|\mathbf{x}\|_2}}_{\leq 1}.$$

i.e. the stepsize α_{MU} is greater than a fraction of α_{PGD}, and this depends on the ratio $\frac{x^{[i]}}{\|\mathbf{x}\|_2}$ (how significant is $x^{[i]}$ in whole $\mathbf{x}$) and the ratio $\frac{\|\mathbf{Q}\|_2}{\|\mathbf{q}_i\|_2}$.

Drawback of MU : slow

Projected gradient descent without acceleration is already much faster than MU. Not to mention the accelerated version.

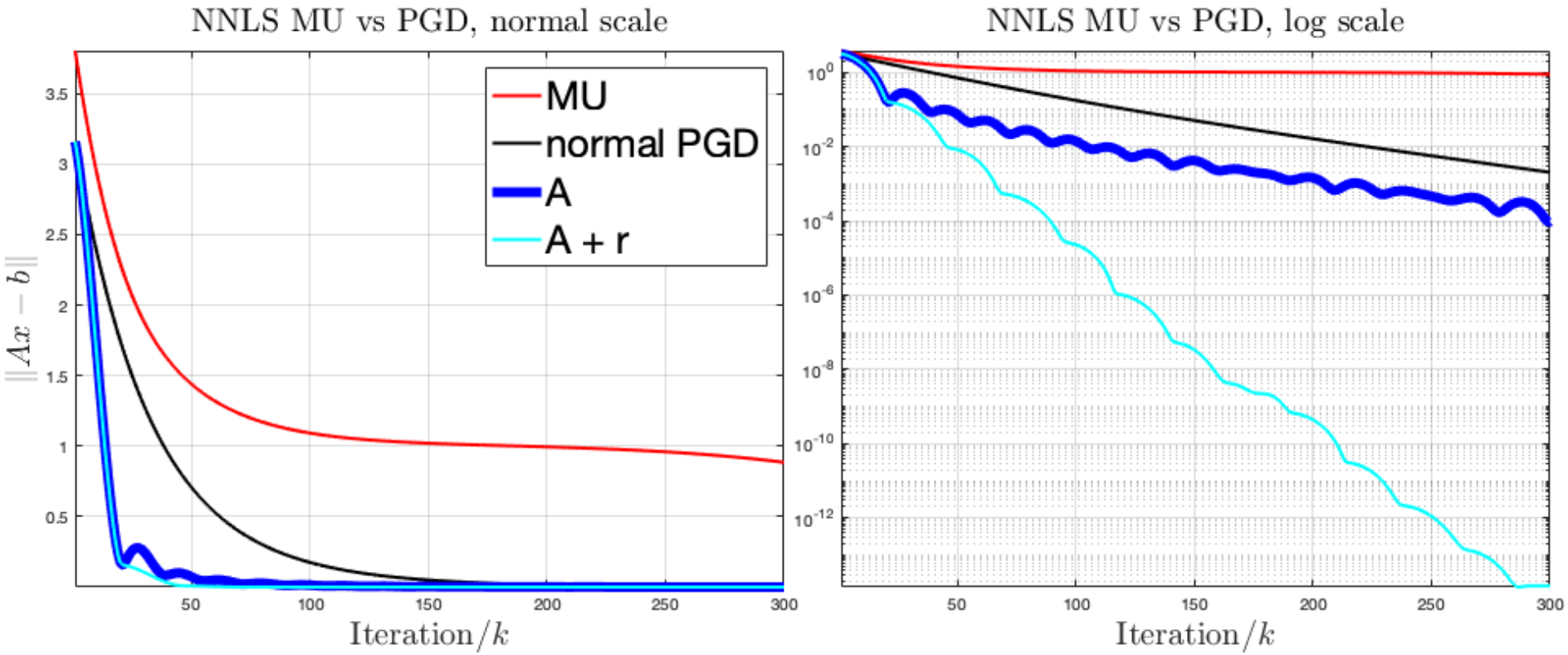

Figure: An illustrative example $(m, n) = 100, 10$. MATLAB code (click me)

Last page - summary

- MU iteration for NNLS

$$\mathbf{x}_{k+1} = \mathbf{x}_{k+1} \star \mathbf{p} \diamond \mathbf{Q}\mathbf{x},$$

  where the multiplication $\star$ and the division $\diamond$ are elementwise. That is, they are the `.*` and `./` in `MATLAB`.

- Compared with projected gradient descent, MU is very slow.

End of document