Overview

1. Nonnegative Least Squares
2. Solving NNLS by Projected Gradient Descent
3. Solving NNLS by Accelerated Projected Gradient Descent
4. Solving NNLS by Accelerated Projected Gradient Descent with restart
5. Other variants of Accelerated Projected Gradient Descent
6. Summary
Nonnegative Least Squares (NNLS): given \( A \in \mathbb{R}^{m \times n} \), \( b \in \mathbb{R}^m \), find \( x \in \mathbb{R}^n_+ \) by solving

\[
\begin{aligned}
(\mathcal{P}) : \ \arg\min_{x \geq 0} f(x) := \frac{1}{2} \| Ax - b \|^2_2.
\end{aligned}
\]

A constrained optimization problem: \( x \) has to be nonnegative.

- \( x \) is the coefficient of columns of \( A \)
- \( x_{\text{NNLS}} \) tells the contributions of each columns \( a_i \) towards \( b \)
- \( x_{\text{LS}} \) is less interpretable as coefficient \( x_{\text{LS}} \) can has mixed signs, leading to mutual elimination
Equivalent constrained QP formulation of NNLS

Expand the function \( \frac{1}{2} \|Ax - b\|_2^2 \):

\[
\begin{align*}
    f(x) &= \frac{1}{2} (Ax - b)^\top (Ax - b) \\
    &= \frac{1}{2} \left( x^\top A^\top Ax - x^\top A^\top b - b^\top Ax + b^\top b \right) \\
    &= \frac{1}{2} \left( x^\top A^\top Ax - 2b^\top Ax + \|b\|_2^2 \right) \\
    &= \frac{1}{2} x^\top A^\top Ax - b^\top Ax + \frac{1}{2} \|b\|_2^2.
\end{align*}
\]

Let \( Q = A^\top A \), \( p = (b^\top A)^\top = A^\top b \) and \( c = \frac{1}{2} \|b\|_2^2 \), NNLS becomes a constrained quadratic programming (QP) problem

\[
\min_{x \geq 0} \frac{1}{2} x^\top Qx - p^\top x + c.
\]

In the following, we ignore the constant \( c \).
NNLS(NNQP) is a convex problem

\[
\min_{x \in \mathbb{R}^n_+} \frac{1}{2} x^\top Q x - p^\top x, \quad Q = A^\top A, \quad p = A^\top b.
\]

- matrix \( Q = A^\top A \) is always positive-semidefinite and symmetric
- If \( A \) is full rank then \( Q \) is positive-definite
- NNLS(NNQP) is a convex optimization problem
  - the function convex : it is quadratic
  - the constraint set is convex : it is the nonnegative orthant
Solving NNLS by pseudo inverse and projection

The simplest (but wrong) way to solve NNLS is to modify the solution obtained from the corresponding ordinary least squares: if $A^\top A$ is invertible, set the gradient $\nabla f(x) = A^\top A x - A^\top b$ zero gives

$$x_{LS} = (A^\top A)^{-1} A^\top b.$$ 

Now we have a two-step method to solve the NNLS

1. $y = (A^\top A)^{-1} A^\top b$ (solution of ordinary least squares)
2. $x = P_{\mathbb{R}^n_+}(y) = \max(y, 0)$ (projection onto nonnegative orthant)
   where $P_{\mathbb{R}^n_+}$ is the projection operator.

In fact, this method can produce a wrong solution. For example, if all components in $x_{LS}$ are negative, then this method basically output a zero vector, while the true $x$ which is non-zero may exists.
For \( f(x) = \frac{1}{2} x^\top Q x - p^\top x \), the gradient and the projection are

\[
\nabla f = Q x - p, \quad P_{\mathbb{R}^n_+}(x) = [x]_+ := \max(x, 0).
\]

The Projected Gradient Descent (PGD) algorithm for solving NNLS is:

**Algorithm 1: PGD for NNLS**

- **Result:** A solution \( x \) that approximately solves \((P)\)
- **Initialization** Set \( x_0 \in \mathbb{R}^n_+ \), \( p = A^\top b \), \( Q = A^\top A \), \( k = 1 \)
- **while** stopping condition is not met **do**
  - \( x_k = [x_{k-1} - t_k(Qx_{k-1} - p)]_+ \)
  - \( k = k + 1 \)
- **end**

where stepsize \( t_k \) can be set as \( \frac{1}{L} \), where \( L \) is the Lipschitz constant of \( \nabla f(x) \). The next slide shows \( L = \|Q\|_2 \).
Fact 1. For all matrix $A$ and all vector $x$, we have $\|Ax\|_2 \leq \|A\|_2 \|x\|_2$.

Remark: this is operator norm inequality, which is an immediate consequence of the definition of operator norm.

Lemma 1. $f(x) = \frac{1}{2} \|Ax - b\|_2^2$ is $L$-smooth with $L = \|A^\top A\|_2$.

i.e. $\|\nabla f(x_1) - \nabla f(x_2)\|_2 \leq L \|x_1 - x_2\|_2$ and $L = \|Q\|_2 = \|A^\top A\|_2$.

Proof (Direct proof).

\[
\|\nabla f(x_1) - \nabla f(x_2)\|_2 = \|(A^\top Ax_1 - A^\top b) - (A^\top Ax_2 - A^\top b)\|_2 \\
= \|A^\top Ax_1 - A^\top Ax_2\|_2 \\
= \|A^\top A(x_1 - x_2)\|_2 \\
\leq \|A^\top A\|_2 \|x_1 - x_2\|_2 \quad \Box
\]
With $L = \|A^\top A\|_2$, step size $t = \frac{1}{L} = \frac{1}{\|A^\top A\|_2}$, the PGD algorithm for solving NNLS becomes:

**Algorithm 2:** PGD (constant step size) for NNLS

**Result:** A solution $x$ that approximately solves $(\mathcal{P})$

**Initialization** Set $x_0 \in \mathbb{R}^n_+$, $p = A^\top b$, $Q = A^\top A$, $t = \frac{1}{\|Q\|_2}$, $k = 1$

**while** stopping condition is not met **do**

$$x_k = [x_{k-1} - t(Qx_{k-1} - p)]_+$$

$$k = k + 1$$

**end**

From the theory of gradient descent, PGD converges at rate $O\left(\frac{1}{k}\right)$, where $k$ is the iteration number.
Rewrite the update in compact form

\[ x_k = \left[ (I_n - tQ)x_{k-1} + tp \right]_+ \]

Fix constants can be pre-computed outside the loop, we have

**Algorithm 3: PGD (constant step size) for NNLS (compact form)**

**Result:** A solution \( x \) that approximately solves (\( \mathcal{P} \))

**Initialization** Set \( x_0 \in \mathbb{R}^n_+ \), \( \Theta_1 = I_n - \frac{A^\top A}{\|A^\top A\|_2} \), \( \theta_2 = \frac{A^\top b}{\|A^\top A\|_2} \), \( k = 1 \)

**while stopping condition is not met** do

\[ x_{k+1} = \left[ \Theta_1 x_k + \theta_2 \right]_+ \]

\[ k = k + 1 \]

end
Nesterov’s Acceleration

With Nesterov’s acceleration, the accelerated PGD (APGD, with constant step size) algorithm is

Algorithm 4: APGD for NNLS

Result: A solution $x$ that approximately solves $(P)$

Initialization Set $y_0 = x_0 \in \mathbb{R}_+^n$, $\Theta_1 = I_n - \frac{A^T A}{\|A^T A\|_2}$, $\theta_2 = \frac{A^T b}{\|A^T A\|_2}$, $k = 1$, set $\alpha_0 \in (0 \ 1)$

while stopping condition is not met do
  $x_k = [\Theta_1 y_{k-1} + \theta_2]_+$ (projected gradient step)
  $\alpha_k = \frac{1}{2} \left( \sqrt{\alpha_{k-1}^4 + 4\alpha_{k-1}^2 - \alpha_{k-1}^2} \right)$, $\beta_k = \frac{\alpha_{k-1}(1-\alpha_{k-1})}{\alpha_{k-1}^2 + \alpha_k}$
  $y_k = x_k + \beta_k(x_k - x_{k-1})$ (extrapolation)
  $k = k + 1$
end

The items in blue are the modifications from Nesterov’s acceleration.
Recall, PGD is a *monotone*\(^1\) method: for all \(k\), \(f(x_{k+1}) \leq f(x_k)\). However, Nesterov’s accelerated method is not monotone: in some iterations, the objective value actually increases.

An illustrative example \((m, n) = 100, 5\).

\(^1\)In Nesterov’s wording, relaxation sequence.
Accelerated Projected Gradient Descent with restarts

To make the scheme monotone, we can apply \textit{adaptive restarts}: if error increases, we switch to gradient descent, and reset all parameters.

\textbf{Algorithm 5: APGD for NNLS}

\textbf{Result:} A solution $x$ that approximately solves $(P)$

\textbf{Initialization} Set $y_0 = x_0 \in \mathbb{R}^n$, $\Theta_1 = I_n - \frac{A^\top A}{\|A^\top A\|_2}$, $\theta_2 = \frac{A^\top b}{\|A^\top A\|_2}$, $k = 1$, set $\alpha_0 \in (0, 1)$

\textbf{while stopping condition is not met do}

\hspace{1em} $x_k = [\Theta_1 y_{k-1} + \theta_2]_+$ (projected gradient step)

\hspace{1em} $\alpha_k = \frac{1}{2} \left( \sqrt{\alpha_{k-1}^4 + 4\alpha_{k-1}^2 \alpha_k^2} - \alpha_{k-1}^2 \right)$, $\beta_k = \frac{\alpha_{k-1}(1-\alpha_{k-1})}{\alpha_{k-1}^2 + \alpha_k}$

\hspace{1em} $y_k = x_k + \beta_k (x_k - x_{k-1})$ (extrapolation)

\hspace{1em} \textbf{if error increases do}

\hspace{2em} $x_{k+1} = [\Theta_1 x_k + \theta_2]_+$ (perform normal projected gradient step)

\hspace{2em} $y_{k+1} = x_{k+1}$ (restart)

\hspace{2em} $\alpha_k = \alpha_0$ (reset parameter)

\hspace{1em} \textbf{endif}

\hspace{1em} $k = k + 1$

\textbf{end}
Accelerated Projected Gradient Descent with restart

Figure: An illustrative example $(m, n) = 100, 10$. 

MATLAB code (click me)
The parameters

\[ \alpha_{k+1} = \frac{1}{2} \left( \sqrt{\alpha_k^4 + 4\alpha_k^2} - \alpha_k^2 \right), \quad \beta_k = \frac{\alpha_k (1 - \alpha_k)}{\alpha_k^2 + \alpha_{k+1}} \]

are so “complicated”. Is there a simpler one?

The answer is: Yes. Paul Tseng gave \( \beta_k = \frac{k - 1}{k + 2} \):

---

**Algorithm 6:** APGD for NNLS using Paul Tseng’s \( \beta \)

**Result:** A solution \( x \) that approximately solves (\( \mathcal{P} \))

**Initialization** Set \( y_0 = x_0 \in \mathbb{R}^n, \Theta_1 = I_n - \frac{A^\top A}{\|A^\top A\|_F}, \theta_2 = \frac{A^\top b}{\|A^\top A\|_F} \)

**while** stopping condition is not met **do**

\[
\begin{align*}
x_k &= \left[ \Theta_1 y_{k-1} + \theta_2 \right]_+ \text{ (projected gradient step)} \\
y_k &= x_k + \frac{k - 1}{k + 2} (x_k - x_{k-1}) \text{ (extrapolation)}
\end{align*}
\]

**end**

(Note that, for simplicity, the update of \( k \) is not shown in the algorithm)
APGD with constant $\beta$

Note that the function $f(x)$ in NNLS is smooth, and it is strongly convex if $A$ is full rank.

- **Strongly convex**: recall a function $f(x)$ is strongly convex iff $\nabla^2 f(x) - \mu I \geq 0$. As $\nabla^2 f(x) = Q = A^\top A$, we have
  $$Q - \mu I \geq 0.$$  
  Here, $\mu$ can be taken as $\lambda_{\text{min}}(Q) = \sigma_{\text{min}}(A)$.

- **$L$-Smooth**: as $f$ is twice differentiable, $f$ is $L$-smooth iff $\nabla^2 f(x) - LI \leq 0$. We have $L \leq \lambda_{\text{max}}(Q) = \sigma_{\text{max}}(A)$.

For smooth strongly convex function, the extrapolation parameter $\beta$ of Nesterov's acceleration can be set to

$$\beta_k = \beta = \frac{1 - \sqrt{Q}}{1 + \sqrt{Q}}$$

where $Q = \frac{L}{\mu}$ is the (optimization) condition number of the $f$.

Recall the (linear algebra) condition number of a matrix $A$ is $\kappa(A)$. 

Nesterov’s Acceleration with $\beta$

With constant $\beta$, we have the following

Algorithm 7: APGD for NNLS using fixed $\beta$

Result: A solution $x$ that approximately solves $(P)$

Initialization Set $y_0 = x_0 \in \mathbb{R}^n$, $\Theta_1 = I_n - \frac{A^\top A}{\|A^\top A\|_F}$, $\theta_2 = \frac{A^\top b}{\|A^\top A\|_F}$

Set $\beta = \frac{1 - \sqrt{\kappa}}{1 + \sqrt{\kappa}}$, where $\kappa = \frac{L}{\mu} = \frac{\lambda_{\text{max}}(Q)}{\lambda_{\text{min}}(Q)} = \frac{\sigma_{\text{max}}(A)}{\sigma_{\text{min}}(A)} = \frac{1}{\kappa(A)}$

while stopping condition is not met do
  $x_k = [\Theta_1 y_{k-1} + \theta_2]_+$ (projected gradient step)
  $y_k = x_k + \beta (x_k - x_{k-1})$ (extrapolation)
end

The items in blue are the modifications from the acceleration scheme with fixed $\beta$. 
Comparisons

Figure: An illustrative example \((m, n) = 100, 20\). MATLAB code (click me)
Summary:

- **NNLS problem** \( \min_{x \in \mathbb{R}^n_+} f(x) = \frac{1}{2} \| Ax - b \|_2^2 \)

- PGD algorithm for NNLS
- APGD algorithms for NNLS
- APGD algorithm with restart for NNLS

End of document