MTh 4101/510 Inverse Problems and DA

iterative regularization methods

Solution to large-scale (high-dimensional) inverse problems must avoid matrix factorization such as SVD and rely only on matrix-vector products.

→ the number of iterations plays the role of the regularization parameter.

Consider the model problem

$$Ax = b$$

Given an estimate $x^{(k)}$ to the solution we construct $x^{(k+1)}$ in terms of $x^{(k)}$

→ Landweber iteration

→ Algebraic reconstruction technique (ART)

→ Conjugate gradient methods.

Landweber iteration (1950s). Given $x^{(0)}$, (typically $x^{(0)}=0$)

$$x^{(k+1)} = x^{(k)} + \omega A^T (b - A x^{(k)}), \quad k = 0, 1, 2, \ldots$$

where $0 < \omega < 2 \|A^T A\|_2^{-1} = \frac{2}{\sigma_1^2}$

May be interpreted as a steepest descent iteration to minimize

$$f(x) = \frac{1}{2} \|Ax - b\|^2$$

Notice that $\nabla f(x) = A^T (Ax - b)$

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

The k-iterate may be also expressed as

$$x^{(k)} = \sum_{i=1}^{n} f_i^{(k)} \frac{u_i^T \cdot b}{\sigma_i} v_i \quad , \quad A = U S V^T$$

where $f_i^{(k)} = 1 - (1 - \omega \sigma_i^2)^k, \quad i = 1:n$

Then $f_i^{(k)} \approx 1$ for large σ_i

$f_i^{(k)} \approx k \omega \sigma_i^2$ for $\omega \sigma_i^2 \ll 1$

and $f_i^{(k)} \longrightarrow 1$ as $k \to \infty$.

ART (algebraic reconstruction technique)

(Kaczmarz's method)

Let $a_i^T = A(i,:)$ denote the vector of row i of A. The k^{th} iteration consists on succesive projections onto the hyperplanes $H_i = \{x : a_i^T \cdot x = b_i\}$, $i = 1:n$

Notice that for a generic $\tilde{x}$, the projection onto H_i is

$$\hat{x} = P_{H_i}\tilde{x} = \tilde{x} + \frac{b_i - a_i^T \cdot \tilde{x}}{\|a_i\|^2} a_i$$

Algorithm to produce $x^{(k+1)}$:

$z = x^{(k)}$

for $i = 1:n$

$$z = z + \frac{b_i - a_i^T \cdot z}{\|a_i\|^2} a_i$$

end

$x^{(k+1)} = z$

Conjugate gradient methods

Given matrix $A \in \mathbb{R}^{n \times n}$ symmetric and positive definite, consider linear system

$$(*) \quad Ax = b$$

Solution to $(*)$ also is the solution to the optimization problem

$$(**) \quad \min_{x \in \mathbb{R}^n} \left\{ \frac{1}{2} x^T A x - b^T \cdot x \right\} \longrightarrow \text{denote } \phi(x)$$

Conjugate gradient (CG) methods provide an iterative approach to approximate the solution x^* to $(*)$ / $(**)$ using A-conjugate directions

Definition The vectors $\{p_0, \dots, p_k\}$ are A-conjugate if $p_i \neq 0$, $i = 0:k$ and

$$p_i^T A p_j = 0, \quad \forall\, i \neq j$$

Property A-conjugate vectors are linearly independent.

Notice $c_0 p_0 + \dots + c_k p_k = 0 \;/\; p_i^T A \cdot \Rightarrow c_i = 0$

Assume that $p_0, \dots, p_{n-1}$ are A-conjugate vectors. (thus form a basis to $\mathbb{R}^n$).

Given an initial guess $x_0 \in \mathbb{R}^n$, define

$$x_{k+1} = x_k + \alpha_k p_k, \qquad k = 0, 1, \dots, n-1$$

where $\alpha_k \in \mathbb{R}$ (positive or negative!) is obtained as solution to

$$\min_{\alpha \in \mathbb{R}} \phi(x_k + \alpha p_k) = \frac{1}{2}(x_k + \alpha p_k)^T A (x_k + \alpha p_k) - b^T (x_k + \alpha p_k)$$

<u>Notation</u> For $x \in \mathbb{R}^n$, $\nabla \phi(x) = Ax - b = \underline{r(x)}$

$r(x)$ is called residual.

$$\varphi(\alpha) = \phi(x_k + \alpha p_k) = \frac{1}{2}(p_k^T A p_k)\alpha^2 + \alpha(p_k^T A x_k - b^T p_k) + \phi(x_k)$$

Minimizer

$$\boxed{\alpha_k = -\frac{p_k^T r_k}{p_k^T A p_k}}$$

Theorem Let $A \in \mathbb{R}^{n\times n}$ symmetric and positive definite matrix and $p_0, \dots, p_{n-1}$ A-conjugate vectors in $\mathbb{R}^n$.

Then for any $x_0 \in \mathbb{R}^n$, the sequence

$$x_{k+1} = x_k + \alpha_k p_k, \quad \alpha_k = -\frac{p_k^T r_k}{p_k^T A p_k}$$

converges to the solution x^* of the problem (*)/(**) in <u>at most</u> n steps (iterations).

Proof Since $\{p_0, \dots, p_{n-1}\}$ are linearly independent, they form a basis to $\mathbb{R}^n$. Then

$$x^* - x_0 = c_0 p_0 + \cdots + c_{n-1} p_{n-1}$$

where the coefficients c_k are obtained as

$$c_k = \frac{p_k^T A(x^* - x_0)}{p_k^T A p_k} = \frac{p_k^T (b - A x_0)}{p_k^T A p_k} = -\frac{p_k^T r_0}{p_k^T A p_k}$$

Notice that $x_k = x_0 + \alpha_0 p_0 + \cdots + \alpha_{k-1} p_{k-1}$

implies $p_k^T A(x_k - x_0) = 0 \Rightarrow p_k^T r_k = p_k^T r_0$

Thus $\underline{c_k = \alpha_k}$

Conjugate Gradient Algorithm (practical version)

x_0 initial guess

$r_0 = A x_0 - b$

$p_0 = -r_0$

$k = 0$

while $r_k \neq 0$

next iterate:

$$\alpha_k = \frac{r_k^T \cdot r_k}{p_k^T \cdot A p_k}$$

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

$$r_{k+1} = r_k + \alpha_k A p_k$$

next direction:

$$\beta_{k+1} = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$$

$$p_{k+1} = -r_{k+1} + \beta_{k+1} p_k$$

$k = k+1$

end

For implementation work with 3 pairs of vectors

(x, x_{new})

(r, r_{new})

(p, p_{new})

Remarks: CG algorithm generates at once with x_{k+1} the next direction p_{k+1}

Cost per iteration: $A p_k \rightarrow O(n^2)$, $p_k^T \cdot (A p_k) \rightarrow n$

$r_{k+1}^T \cdot r_k \rightarrow n$

→ No need for explicit matrix A, just the ability to evaluate matrix·vector product Ap