$M^T M v = W A^T A W^{-1} v$

routine to form $W^{-1}v$

$y = W^{-1}v$ $\quad Ay \rightarrow A^T(Ay) = z$

apply forward WT routine on z

$M = \underbrace{A}_{m \times n} \underbrace{W^{-1}}_{n \times n}$. Need to compute

$Mx = AW^{-1}x = A(W^{-1}x)$

$M^T y = (AW^{-1})^T y = W^{-T} A^T y = W A^T y$

since $W^{-1} = W^T$ for orthogonal transforms

QR factorization

$A = QR$ (unpivoted QR)

$AP = QR$ (column pivoted QR)

P is a permutation matrix of the columns of A $\Rightarrow A = QRP^T$ (P is orthogonal matrix)

$\Rightarrow A(:, I) = QR$

equivalently, Permutation matrix can be expressed as an index vector.

Application to least squares

$$\min \|Ax-b\|_2 \Leftrightarrow \min \|Ax-b\|_2^2$$

$$\bar{x} = \arg\min_x \left\{ \|Ax-b\|_2^2 \right\} \Rightarrow 2A^T(A\bar{x}-b) = 0$$

$$\Rightarrow A^TA\bar{x} = A^Tb$$

$$R^TQ^TQR\bar{x} = R^TQ^Tb$$

$$\Rightarrow R^TR\bar{x} = R^TQ^Tb$$

$$\Rightarrow R\bar{x} = Q^Tb \quad \text{(as long as } R \text{ nonsingular)}$$

Then $\bar{x}$ can be obtained by back substitution.

$$A = Q \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & & \vdots \\ 0 & 0 & & r_{nn} \end{pmatrix}$$

pivoting done so that $|r_{11}| > |r_{22}| > \dots > |r_{nn}|$

Linear and nonlinear data fitting

(I) linear data fitting

$(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$

$y = mx + b$ (linear model)

$$\left.\begin{aligned} y_1 &= mx_1 + b \\ &\dots \\ y_n &= mx_n + b \end{aligned}\right\}$$ except most likely fit will not be exact

$$\begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \begin{pmatrix} b \\ m \end{pmatrix} \approx \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

$$A \quad x \quad \simeq \quad b \Rightarrow \quad \bar{x} = \underset{x}{\operatorname{argmin}} \{\|Ax - b\|\}$$

$$\Rightarrow \bar{x} = \underset{x}{\operatorname{argmin}} \{\|Ax - b\|^2\} \Rightarrow \underline{A^T A \bar{x} = A^T b}$$

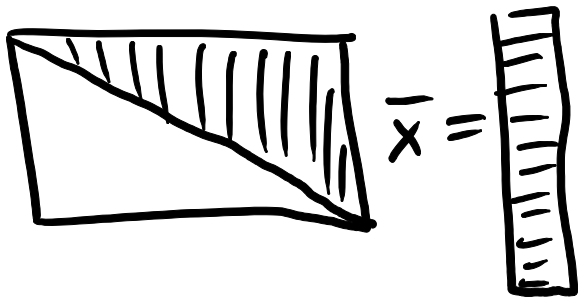
$\Rightarrow$ can solve directly or iteratively

direct solves:

using QR factorization

$A = QR$ (expensive) $\sim \Theta(mn^2)$

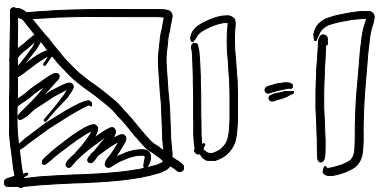
$\Rightarrow R^T \underbrace{Q^T Q}_{=I} R\bar{x} = R^T Q^T b$

$\Rightarrow R\bar{x} = Q^T b$ 

proceed by back substitution

advantage for multiply rhs $(y_1, \ldots, y_n)$ once QR of A has been computed, easy.

using LU factorization

$A^T A = LU$ (expensive) 

$LU\bar{x} = A^T b = c$

$\Rightarrow$ solve $Ly = c$ (L lower triangular use forward substitution)

Then set $\underline{y = U\bar{x}}$ (solve for $\bar{x}$ by back substitution)

again, advantage for multiple y value sets.

alternative: $\bar{x} = CG(A^TA, A^Tb)$ but this is done for each new b vector.

(II) non-linear fitting (generalized possibly)

Ex] $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$; $\vec{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$

$F(\vec{x}, t) = x_1 e^{-(t-x_2)^2/(2x_3^2)}$ ← may be linear or non-linear

some kind of Gaussian fit (t_i, y_i)

residual: $r_i(x) = y_i - F(\vec{x}, t_i)$ for each data point

$\vec{r}(x) = \begin{pmatrix} r_1(x) \\ \vdots \\ r_m(x) \end{pmatrix} \in \mathbb{R}^m$ for m data points

In general, $x \in \mathbb{R}^n$

non linear least squares problem:

$$\vec{x} = \underset{x}{\operatorname{argmin}} \left\{ \|r(x)\|^2 \right\} = \underset{x}{\operatorname{argmin}} \sum_{i=1}^{m} r_i^2(x)$$

Define the Jacobian of the vector function $r(x)$:

$$[J(x)]_{i,j} = \frac{\partial r_i(x)}{\partial x_j} = -\frac{\partial F(x, t_i)}{\partial x_j} \leftarrow$$

$$\vec{r}(x) = \begin{pmatrix} r_1(x) \\ \vdots \\ r_m(x) \end{pmatrix} \Rightarrow \nabla \vec{r}(x) = \begin{pmatrix} \partial r_i / \partial x_1 \\ \vdots \\ \partial r_i / \partial x_n \end{pmatrix}$$

$$\Rightarrow \boxed{[J(x)]_{i,:} = \nabla r_i(x)^T} = -\nabla F(x, t_i)^T$$

i-th row of Jacobian matrix is a row vector ($J(x) = J[r(x)]$)

Now $\vec{x} = \underset{x}{\operatorname{argmin}} \left\{ \|r(x)\|^2 \right\} = \operatorname{argmin} \{ g(x) \}$

has to satisfy $\nabla g(x) = 0$ optimality conditions!

Note: $\vec{x} = \underset{x}{\operatorname{argmin}} \left\{ \frac{1}{2} \|r(x)\|^2 \right\}$

so $g(x) = \frac{1}{2}\|r(x)\|^2$

Then $\nabla g(x) = \sum_{j=1}^{m} r_j(x)\,\nabla r_j(x)$

$$= \underline{J(x)^T r(x)}$$

The Hessian of g:

$$\nabla^2 g(x) = \sum \nabla r_j(x)\,\nabla r_j(x)^T + \overbrace{\sum r_j(x)\,\nabla^2 r_j(x)}^{R(x)}$$

$$\approx J(x)^T J(x)$$

optimality conditions:

$$\underline{\nabla g(x) = J(x)^T r(x) = 0}$$

second order min conditions:

$$\nabla^2 g(x) = J(x)^T J(x) + \underbrace{\sum_{j=1}^{m} r_j(x)\nabla^2 r_j(x)}_{R(x)}$$

has to be positive definite

To get model fit (vector $\vec{x}$) must

Solve non-linear system

$$\nabla g(x) = J(x)^T r(x) = 0$$

Newton's method

$$f(x_0 + \varepsilon) = f(x_0) + f'(x_0)\varepsilon + \frac{1}{2} f''(x_0)\varepsilon^2 + \dots$$

$$\Rightarrow f(x_0+\varepsilon) \approx f(x_0) + f'(x_0)\varepsilon$$

$$\Rightarrow x_1 = x_0 + \varepsilon = x_0 - \frac{f(x_0)}{f'(x_0)}$$

NM: for $f'(x)=0$

$$\left(x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \right) \Rightarrow x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$$

For $\nabla g(x)=0$ this takes the form:

$$x_{n+1} = x_n - \left(\nabla^2 g(x_n)\right)^{-1} \nabla g(x_n)$$

$$= x_n - \left[J(x_n)^T J(x_n) + R(x)\right]^{-1} J(x_n)^T r(x_n)$$

which is quite expensive

approximation (Gauss-Newton)

$$\Rightarrow x_{n+1} = x_n - \left[J(x_n)^T J(x_n)\right]^{-1} J(x_n)^T r(x_n)$$

There is a variety of NLS (non-linear least squares schemes). And there are analogues of regularization.

ID and CUR decompositions

related to QR decomp with pivoting

$$A = U\Sigma V^T \quad \text{(SVD)} \qquad \Sigma = \begin{pmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_r \end{pmatrix}$$

$r = \min(m,n)$

$$\hookrightarrow A_k \simeq U_k \Sigma_k V_k^T$$

truncated SVD

$\|A - A_k\|$ is minimal amongst $\|A - B\|$ where B has rank k in both Frobenius and spectral norms.

$$A = U\Sigma V^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T \simeq \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

① A_k (low rank SVD) is optimal approx to A of rank k.

② there are methods to compute this in $O(mnk)$.

A sparse

$\Rightarrow$ SVD $\Rightarrow$ $U\Sigma V^T \Rightarrow$ truncate $U_k \Sigma_k V_k^T$

(U_k: dense, V_k: dense)

makes sense as an approx. for small values k

① ID decomposition

Interpolative decomposition

$$A = CV^T = \underbrace{A(:, I)}_{\text{re-arrangement of columns of } A} V^T$$

This same vector I appears in the pivoted QR factorization

$$AP = QR \Rightarrow A(:, I) = QR$$

$$A = CV^T = A(:, I) V^T \approx \underbrace{A(:, I_k)}\, V_k^T$$

Related: CUR decomp.

$A \approx CUR$

$C = A(:, I_k)$ $\quad R = A(J_k, :)$

basic idea: apply pivoted QR to columns of $A \Rightarrow I_k$

apply pivoted QR to rows of $A \Rightarrow J_k$