

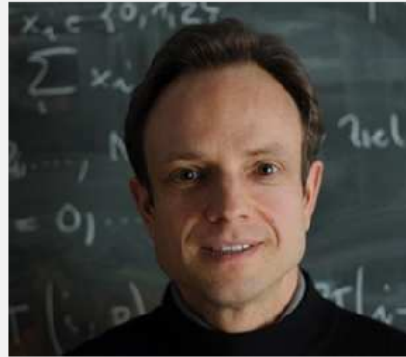
Course Video

Section 3.4.3: Solving General Least-Squares Problems by SVD

Prof. R. Hiptmair, SAM, ETH Zurich

Date: September 29, 2020

(C) Seminar für Angewandte Mathematik, ETH Zürich

**Dependencies.** Section 3.1.1, Section 3.1.3, Section 3.4.1, Section 1.2.1, Section 3.4.2

Duration: 24 minutes

III. Direct Methods for Linear Least Squares Problems

3.4. Singular Value Decomposition (SVD)

3.4.3. Solving General Least-Squares Problems by SVD(General linear) **least squares problem:**

$$r := \text{rank}(A) \leq \min\{m, n\}$$

given: $A \in \mathbb{R}^{m,n}$, $m, n \in \mathbb{N}$, $b \in \mathbb{R}^m$, $\rightarrow$ not necessarily full rankfind: $x^+ \in \mathbb{R}^n$ such that

$$x^+ \in \text{lsq}(A, b) \Leftrightarrow \begin{aligned} & \text{(i)} \quad \|Ax^+ - b\|_2 = \min\{\|Ay - b\|_2 : y \in \mathbb{R}^n\}, \\ & \text{(ii)} \quad \|x^+\|_2 \text{ is minimal under the condition (i).} \end{aligned} \quad (3.1.3.7)$$

$$x^+ = A^+ b, \quad A^+ \triangleq \text{pseudo-inverse}$$

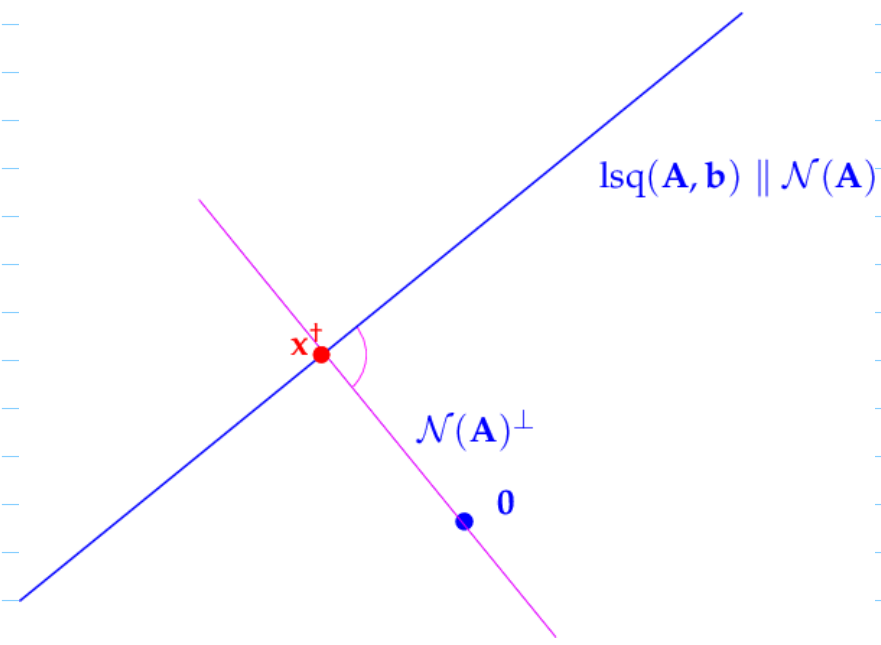
Tool : economical SVD

② $A \in \mathbb{R}^{m,n}$, full SVD

$$A \in \mathbb{R}^{m,n} = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}$$

Diagram illustrating the SVD decomposition of matrix A . The matrix A is shown as a rectangle. It is equal to the product of three matrices: U (partitioned into U_1 and U_2), Σ (partitioned into Σ_r and zeros), and V^T (partitioned into V_1^T and V_2^T). The matrix V is defined as $V = [V_1 \ V_2]$.

(3.4.3.1)



$x^+ \in \mathcal{N}(A)^\perp$

can be retrieved from SVD

Lemma 3.4.1.13. SVD and rank of a matrix $\rightarrow$

If, for some $1 \leq r \leq p := \min\{m, n\}$, the singular values of $A \in \mathbb{K}^{m,n}$ satisfy $\sigma_1 \geq \dots \geq \sigma_r > \sigma_{r+1} = \dots = \sigma_p = 0$, then

- $\text{rank}(A) = r$ (no. of non-zero singular values),
- $\mathcal{N}(A) = \text{Span}\{(V)_{:,r+1}, \dots, (V)_{:,n}\}$,
- $\mathcal{R}(A) = \text{Span}\{(U)_{:,1}, \dots, (U)_{:,r}\}$.

$\triangleright [V \in \mathbb{R}^{n,n} \text{ orthogonal}] :$

$$\mathcal{N}(A)^\perp = \text{Span}\{v_1, \dots, v_r\} = \mathcal{R}(V_1) = \{V_1 y, y \in \mathbb{R}^r\}$$

$$x^+ = V_1 y^+ \text{ for some } y^+ \in \mathbb{R}^r \quad (1)$$

Recall: $\text{lsq}(A, b) = \{x \in \mathbb{R}^n : A^T A x = A^T b\} \quad (2)$

Theorem 3.1.2.1. Obtaining least squares solutions by solving normal equations

The vector $x \in \mathbb{R}^n$ is a least squares solution ($\rightarrow$ Def. 3.1.1.1) of the linear system of equations $Ax = b$, $A \in \mathbb{R}^{m,n}$, $b \in \mathbb{R}^m$, if and only if it solves the **normal equations (NEQ)**

$$A^T A x = A^T b \quad (3.1.2.2)$$

(1) & (2)

$$A^T A V_1 y^+ = A^T b$$

Trick:

$$V_1^T A^T A V_1 y^+ = V_1^T A^T b$$

Plug in SVD: $A = U \Sigma V^T$

③

$$V_1^T V \underbrace{\Sigma^T U^T U \Sigma}_{=I} V_1^T x^+ = V_1^T V \Sigma^T U^T b$$

$$\underbrace{V_1^T [V_1, V_2]}_{[I \ 0]} \underbrace{\left[\begin{array}{c|c} \Sigma_r & 0 \\ \hline 0 & 0 \end{array} \right]}_{\left[\begin{array}{c|c} \Sigma_r^{-1} & 0 \\ \hline 0 & 0 \end{array} \right]} \underbrace{\left[\begin{array}{c} V_1^T \\ V_2^T \end{array} \right]}_{\left[\begin{array}{c} \tilde{b} \\ 0 \end{array} \right]} V_1 x^+ = V_1^T [V_1, V_2] \left[\begin{array}{c|c} \Sigma_r & 0 \\ \hline 0 & 0 \end{array} \right] \left[\begin{array}{c} U_1^T \\ U_2^T \end{array} \right] b$$

▷

$$\Sigma_r x^+ = \Sigma_r U_1^T b$$

$$\Sigma_r \in \mathbb{R}^{r,r} \text{ regular} \Rightarrow x^+ = \Sigma_r^{-1} U_1^T b$$

economical SVD sufficient

$$x^+ = V_1 \Sigma_r^{-1} U_1^T b$$

C++-code 3.4.3.13: Computing generalized solution of $Ax = b$ via SVD

```

1 #include <Eigen/SVD>
2
3
4 VectorXd lsqsvd(const MatrixXd &A, const VectorXd &b) {
5     // Compute economical SVD, compare Code 3.4.2.1
6     Eigen::JacobiSVD<MatrixXd> svd(A, Eigen::ComputeThinU | Eigen::ComputeThinV);
7     VectorXd sv = svd.singularValues();
8     unsigned int r = svd.rank(); // Numerical rank, default tolerance
9     MatrixXd U = svd.matrixU(), V = svd.matrixV();
10    //  $x^+ = V_1 \Sigma_r^{-1} U_1^T b$ , see
11    // (3.4.3.5)
12    return V.leftCols(r) * (sv.head(r).cwiseInverse().asDiagonal() *
13                           (U.leftCols(r).adjoint() * b));
14 }

```

just a scaling

C++-code 3.4.3.14: Computing generalized solution of $Ax = b$ via SVD

```

2 VectorXd lsqsvd_eigen(const MatrixXd &A, const VectorXd &b) {
3     Eigen::JacobiSVD<MatrixXd> svd(A, Eigen::ComputeThinU | Eigen::ComputeThinV);
4     return svd.solve(b);
5 }

```

$$x^+ = V_1 \Sigma_r^{-1} U_1^T b = A^+ b$$

Theorem 3.4.3.16. Pseudoinverse and SVD

If $A \in \mathbb{K}^{m,n}$ has the SVD decomposition $A = U \Sigma V^H$ partitioned as in (3.4.3.1), then its Moore-Penrose pseudoinverse ($\rightarrow$ Thm. 3.1.3.6) is given by $A^+ = V_1 \Sigma_r^{-1} U_1^H$.

(4)

Review questions

Please answer review questions 3.4.3.17

5

7