

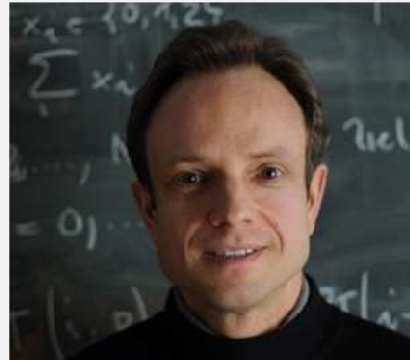
Course Video

Section 3.4.4.1: Norm-Constrained Extrema of Quadratic Forms

Prof. R. Hiptmair, SAM, ETH Zurich

Date: September 29, 2020

(C) Seminar für Angewandte Mathematik, ETH Zürich



Dependencies. Section 3.4.1, Section 3.4.2

Duration: 21 minutes

III. Direct Methods for Linear Least Squares Problems

3.4. Singular Value Decomposition (SVD)

3.4.4. SVD-Based Optimization and Approximation

3.4.4.1. Norm-Constrained Extrema of Quadratic Forms

given $A \in \mathbb{K}^{m,n}$, $m \geq n$, find $x \in \mathbb{K}^n$, $\|x\|_2 = 1$, $\|Ax\|_2^2 \rightarrow \max$, $\hookrightarrow$ Eco-SVD: $A = U \Sigma V^T$

$$\|Ax\|_2 = \|\cancel{U} \Sigma V^T x\|_2 \underset{\substack{\uparrow \\ \text{cols. are orthonormal: "Pythagoras" }}}}{=} \|\Sigma V^T x\|_2 \rightarrow \max$$

$$y = V^T x \Rightarrow \|Ax\|_2^2 = \sum_{\ell=1}^n \sigma_\ell^2 y_\ell^2 \rightarrow \max$$

orthogonal $\Rightarrow$ under constraint $\sum_{\ell=1}^n y_\ell^2 = 1$

② Solution $\underline{x} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \underline{e}_1 \Rightarrow \underline{x} = V \underline{e}_1 = (V)_{:,1}$
 Maximal value: $\|A \underline{x}\|_2 = \sigma_1$
 Euclidean matrix norm $\|A\|_2 = \max_{\|\underline{x}\|=1} \|A \underline{x}\| = \sigma_1$
 $\hookrightarrow$ can be computed via SVD



Solution for $\|A \underline{x}\|_2 \rightarrow \min, \|\underline{x}\|_2 = 1$
 $\underline{x} = (V)_{:,n}, \|A \underline{x}\|_2 = \sigma_n$

Ex. 3.4.4.5: Fitting of hyperplanes

$\hookrightarrow$ in Hesse normal form

$\mathcal{H} := \{ \underline{x} \in \mathbb{R}^d : c + \underline{n}^T \underline{x} = 0 \}$ for $\|\underline{n}\|_2 = 1, c \in \mathbb{R}$

Goal: given the point coordinate vectors $\underline{y}_1, \dots, \underline{y}_m \in \mathbb{R}^d, m > d$, find $\mathcal{H} \leftrightarrow \{c \in \mathbb{R}, \underline{n} \in \mathbb{R}^d, \|\underline{n}\|_2 = 1\}$, such that

$$\sum_{j=1}^m \text{dist}(\mathcal{H}, \underline{y}_j)^2 = \sum_{j=1}^m |c + \underline{n}^T \underline{y}_j|^2 \rightarrow \min. \quad (3.4.4.8)$$

$$(3.4.4.8) \Leftrightarrow \left\| \begin{bmatrix} 1 & y_{1,1} & \dots & y_{1,d} \\ 1 & y_{2,1} & \dots & y_{2,d} \\ \vdots & \vdots & & \vdots \\ 1 & y_{m,1} & \dots & y_{m,d} \end{bmatrix} \begin{bmatrix} c \\ n_1 \\ \vdots \\ n_d \end{bmatrix} \right\|_2 \rightarrow \min \text{ under the constraint } \|\underline{n}\|_2 = 1.$$

New: c not subject to constraint

(Preliminary) idea: $c : (A \underline{x})_? = 0$

Trick: QR-decomposition of A

$$A := \begin{bmatrix} 1 & y_{1,1} & \dots & y_{1,d} \\ 1 & y_{2,1} & \dots & y_{2,d} \\ \vdots & \vdots & & \vdots \\ 1 & y_{m,1} & \dots & y_{m,d} \end{bmatrix} = \underset{\text{orthogonal}}{Q} R, \quad R := \begin{bmatrix} r_{11} & r_{12} & \dots & \dots & r_{1,d+1} \\ 0 & r_{22} & \dots & \dots & r_{2,d+1} \\ \vdots & & \ddots & & \vdots \\ 0 & \dots & & r_{d+1,d+1} & 0 \\ \vdots & & & & \vdots \\ 0 & \dots & & & 0 \end{bmatrix} \in \mathbb{R}^{m,d+1}.$$

$$\|A \underline{x}\|_2 = \|\cancel{Q} R \underline{x}\|_2 = \|R \underline{x}\| \rightarrow \min$$

$$\|A \underline{x}\|_2 \rightarrow \min \Leftrightarrow \|R \underline{x}\|_2 = \left\| \begin{bmatrix} r_{11} & r_{12} & \dots & \dots & r_{1,d+1} \\ 0 & r_{22} & \dots & \dots & r_{2,d+1} \\ \vdots & & \ddots & & \vdots \\ 0 & \dots & & r_{d+1,d+1} & 0 \\ \vdots & & & & \vdots \\ 0 & \dots & & & 0 \end{bmatrix} \begin{bmatrix} c \\ n_1 \\ \vdots \\ n_d \end{bmatrix} \right\|_2 \rightarrow \min. \quad (3.4.4.9)$$

$$(I) \quad \underline{n} \in \mathbb{R}^d, \|\underline{n}\| = 1 : \|\hat{A} \underline{n}\|_2 \rightarrow \min$$

"in standard form"

$$(II) \quad c := -\frac{1}{r_{11}} \sum_{\ell=1}^d r_{\ell 1} r_{1,\ell} \Rightarrow (A \underline{x})_1 = 0$$

③

C++-code 3.4.4.11: (Generalized) distance fitting of a hyperplane: solution of (3.4.4.12)

```

2 // Fitting of a hyperplane, returning Hesse normal form
3 void clsq(const MatrixXd& A, const unsigned dim,
4           double& c, VectorXd& n) {
5     unsigned p = A.cols(), m = A.rows();
6     if (p < dim + 1) { cerr << "not enough unknowns\n"; return; }
7     if (m < dim) { cerr << "not enough equations\n"; return; }
8
9     m = std::min(m, p);
10    // First step: orthogonal transformation, see Code 3.3.4.1
11    MatrixXd R = A.householderQr().matrixQR().template triangularView<Eigen::Upper>();
12    // compute matrix V from SVD composition of R, solve (3.4.4.10)
13    MatrixXd V = R.block(p - dim, p - dim, m + dim - p, dim)  $\stackrel{!}{=} \tilde{A}$ 
14                  .jacobiSvd(Eigen::ComputeFullV).matrixV();
15    n = V.col(dim - 1);
16
17    MatrixXd R_topleft = R.topLeftCorner(p - dim, p - dim);
18    c = -(R_topleft.template triangularView<Eigen::Upper>()
19          .solve(R.block(0, p - dim, p - dim, dim)) * n)(0);
20 }

```

Review questions :

Please look at Review Questions 3.4.4.13

$A \in \mathbb{R}^{m \times p}$: solves $\|A \begin{bmatrix} c \\ \underline{n} \end{bmatrix}\|_2 \rightarrow \min$
 $\underline{n} \in \mathbb{R}^d, \|\underline{n}\|_2 = 1$

4

5

7