Linear algebra with applications

Ivan Markovsky

- Why is linear algebra useful?
- Numerical methods for matrix computations
- Applications of linear algebra in:
  - System and control theory (realization and identification)
  - Signal processing (array and image precessing)
  - Machine learning (data compression and information retrieval)

Topics

1. Review of linear algebra
2. Numerical linear algebra: matrix factorizations
3. Some optimization problems with analytic solution
4. Applications (in estimation, control, and signal processing

References

- G. Strang, Linear algebra and its applications
- N. Trefethen & Bau, Numerical linear algebra
- G. Golub & Van Loan, Matrix computations
- S. Boyd, EE263: Linear dynamical systems (online)

Journals

- SIAM Journal on Matrix Analysis and Applications
- Linear Algebra and Its Applications
- Journal of Computational and Applied Mathematics
- Numerical Linear Algebra with Applications
- NA digest mailing list

Some net resources

- NA digest mailing list
- Netlib repository
- Decision tree for optimization software

Numerical analysis

Common misconception: numerical analysis is about

analysis of rounding errors

Indeed

- perturbation analysis
- stability, cost, and convergence analysis

are important topics in numerical computing, however,

The core of numerical analysis is

development of algorithms

Numerical linear algebra software

Should you write numerical linear algebra code?

No, whenever possible use existing (high quality) libraries for linear algebra computation instead of writing your own implementation.

In special cases (*e.g.*, structured matrix, real-time implementation) writing linear algebra code or modifying existing one is needed.

This course teaches

1. how to use linear algebra and optimization (What is available?)
2. basic understanding of numerical methods (How does it work?)
3. pitfalls in using the methods (When it does not work?)

Vectors and matrices

Data structure point of view: 1D and 2D arrays of numbers, a table

Convention: vectors are denoted by lowercase letters (*e.g.*, v) and matrices are denoted by uppercase letters (*e.g.*, M)

Use indexes to retrieving data from a vector/matrix:

- v_i (or $v(i)$) — the ith element of v
- M_{ij} (or $M(i,j)$) — the element in the ith row and the jth column

Dimensions of a vector/matrix: # of rows and # of columns

Slices: $v_{i_1:i_2}$ (or $v(i_1{:}i_2)$) and $M_{i_1:i_2,j_1:j_2}$ (or $M(i_1{:}i_2,j_1{:}j_2)$)

Example: diet design

Nutrients/food ingredients table:

		food ingredients		
		x_1	$\cdots$	x_n
nutrients	y_1	a_{11}	$\cdots$	a_{1n}
	$\vdots$	$\vdots$		$\vdots$
	y_m	a_{m1}	$\cdots$	a_{mn}

$A = [a_{ij}] \in \mathbb{R}^{m \times n}$, $\quad a_{ij}$ — amount of nutrient i in 1 unit of food j

Diet recipe: a vector $x \in \mathbb{R}^n$, with x_j being the amount of ingredient j

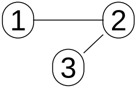
$y = Ax$ — nutrients amounts resulting of recipe x

Diet design:

find a diet x that achieves desired nutrients amounts y_{spec}
find the cheapest diet that achieves y_{spec}
find an x achieving the closest nutrients amounts to y_{spec}

Example: adjacency matrix of a graph

Graph $\mathscr{G}$ with n nodes and undirected edges, *e.g.*,

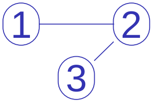


The adjacency matrix $A \in \{0,1\}^{n\times n}$ of $\mathscr{G}$ has 1 in the (i,j)th entry, if the ith node is connected to the jth node by an edge, and 0 otherwise, *e.g.*,

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

A completely describes $\mathscr{G}$ and vice verse, $A \leftrightarrow \mathscr{G}$. Moreover, questions related to properties of $\mathscr{G}$ can be answered by matrix operations on A.

For example, the entries of $A^2 = AA$ have interpretation in terms of $\mathscr{G}$.



$$A^2 = AA = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

- $a_{i_1 k} a_{k i_2} = 1$ when there is an edge between i_1 and k and an edge between k and i_2, *i.e.*, a path with length 2 between i_1 and i_2
- $(A^2)_{i_1 i_2} = \sum_{k=1}^{n} a_{i_1 k} a_{k i_2} =$ # of paths with length 2 between i_1 and i_2
- $(A^k)_{i_1 i_2} =$ # of paths with length k between i_1 and i_2