Lecture slides for

Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares

Developed by Stephen Boyd   Lieven Vandenberghe
Modified by John Duchi

10. Matrix multiplication

Outline

Matrix multiplication

Composition of linear functions

Matrix powers

QR factorization

Householder transformations

Matrix multiplication

- can multiply $m \times p$ matrix A and $p \times n$ matrix B to get $C = AB$:

$$C_{ij} = \sum_{k=1}^{p} A_{ik}B_{kj} = A_{i1}B_{1j} + \cdots + A_{ip}B_{pj}$$

  for $i = 1, \ldots, m$, $j = 1, \ldots, n$

- to get C_{ij}: move along ith row of A, jth column of B
- example:

$$\begin{bmatrix} -1.5 & 3 & 2 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} -1 & -1 \\ 0 & -2 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 3.5 & -4.5 \\ -1 & 1 \end{bmatrix}$$

Special cases of matrix multiplication

- scalar-vector product (with scalar on right!) $x\alpha$
- inner product $a^T b$
- matrix-vector multiplication Ax
- *outer product* of m-vector a and n-vector b

$$ab^T = \begin{bmatrix} a_1b_1 & a_1b_2 & \cdots & a_1b_n \\ a_2b_1 & a_2b_2 & \cdots & a_2b_n \\ \vdots & \vdots & & \vdots \\ a_mb_1 & a_mb_2 & \cdots & a_mb_n \end{bmatrix}$$

Properties

- $(AB)C = A(BC)$, so both can be written ABC
- $A(B + C) = AB + AC$
- $(AB)^T = B^T A^T$
- $AI = A$ and $IA = A$
- $AB = BA$ *does not hold in general*

Block matrices

block matrices can be multiplied using the same formula, *e.g.*,

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE+BG & AF+BH \\ CE+DG & CF+DH \end{bmatrix}$$

(provided the products all make sense)

Column interpretation

- denote columns of B by b_i:
$$B = \begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix}$$
- then we have
$$\begin{aligned} AB &= A \begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix} \\ &= \begin{bmatrix} Ab_1 & Ab_2 & \cdots & Ab_n \end{bmatrix} \end{aligned}$$
- so AB is 'batch' multiply of A times columns of B

Multiple sets of linear equations

- given k systems of linear equations, with same $m \times n$ coefficient matrix

$$Ax_i = b_i, \quad i = 1, \ldots, k$$

- write in compact matrix form as $AX = B$
- $X = [x_1 \ \cdots \ x_k]$, $B = [b_1 \ \cdots \ b_k]$

Inner product interpretation

- with a_i^T the rows of A, b_j the columns of B, we have

$$AB = \begin{bmatrix} a_1^T b_1 & a_1^T b_2 & \cdots & a_1^T b_n \\ a_2^T b_1 & a_2^T b_2 & \cdots & a_2^T b_n \\ \vdots & \vdots & & \vdots \\ a_m^T b_1 & a_m^T b_2 & \cdots & a_m^T b_n \end{bmatrix}$$

- so matrix product is all inner products of rows of A and columns of B, arranged in a matrix

Gram matrix

- let A be an $m \times n$ matrix with columns $a_1, \ldots, a_n$
- the *Gram matrix* of A is

$$G = A^T A = \begin{bmatrix} a_1^T a_1 & a_1^T a_2 & \cdots & a_1^T a_n \\ a_2^T a_1 & a_2^T a_2 & \cdots & a_2^T a_n \\ \vdots & \vdots & \ddots & \vdots \\ a_n^T a_1 & a_n^T a_2 & \cdots & a_n^T a_n \end{bmatrix}$$

- Gram matrix gives all inner products of columns of A
- example: $G = A^T A = I$ means columns of A are orthonormal

Complexity

- to compute $C_{ij} = (AB)_{ij}$ is inner product of p-vectors
- so total required flops is $(mn)(2p) = 2mnp$ flops
- multiplying two 1000×1000 matrices requires 2 billion flops
- ...and can be done in well under a second on current computers

Outline

Matrix multiplication

Composition of linear functions

Matrix powers

QR factorization

Householder transformations

Composition of linear functions

- A is an $m \times p$ matrix, B is $p \times n$
- define $f : \mathbf{R}^p \to \mathbf{R}^m$ and $g : \mathbf{R}^n \to \mathbf{R}^p$ as

$$f(u) = Au, \qquad g(v) = Bv$$

- f and g are linear functions
- *composition* of f and g is $h : \mathbf{R}^n \to \mathbf{R}^m$ with $h(x) = f(g(x))$
- we have

$$h(x) = f(g(x)) = A(Bx) = (AB)x$$

- composition of linear functions is linear
- associated matrix is product of matrices of the functions

Second difference matrix

- D_n is $(n-1) \times n$ difference matrix:

$$D_n x = (x_2 - x_1, \ldots, x_n - x_{n-1})$$

- D_{n-1} is $(n-2) \times (n-1)$ difference matrix:

$$D_{n-1} y = (y_2 - y_1, \ldots, y_{n-1} - y_{n-2})$$

- $\Delta = D_{n-1} D_n$ is $(n-2) \times n$ second difference matrix:

$$\Delta x = (x_1 - 2x_2 + x_3, x_2 - 2x_3 + x_4, \ldots, x_{n-2} - 2x_{n-1} + x_n)$$

- for $n = 5$, $\Delta = D_{n-1} D_n$ is

$$\begin{bmatrix} 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Outline

Matrix multiplication

Composition of linear functions

Matrix powers

QR factorization

Householder transformations

Matrix powers

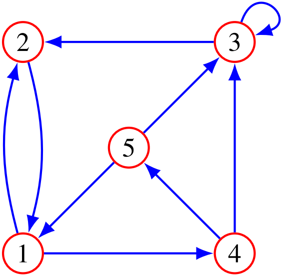
- for A square, A^2 means AA, and same for higher powers
- with convention $A^0 = I$ we have $A^k A^l = A^{k+l}$
- negative powers later; fractional powers in other courses

Directed graph

- $n \times n$ matrix A is adjacency matrix of directed graph:

$$A_{ij} = \begin{cases} 1 & \text{there is a edge from vertex } j \text{ to vertex } i \\ 0 & \text{otherwise} \end{cases}$$

- example:



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

Paths in directed graph

- square of adjacency matrix:

$$(A^2)_{ij} = \sum_{k=1}^{n} A_{ik}A_{kj}$$

- $(A^2)_{ij}$ is number of paths of length 2 from j to i
- for the example,

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

  e.g., there are two paths from 4 to 3 (via 3 and 5)
- more generally, $(A^\ell)_{ij}$ = number of paths of length ℓ from j to i

Outline

Matrix multiplication

Composition of linear functions

Matrix powers

QR factorization

Householder transformations

Gram–Schmidt in matrix notation

- run Gram–Schmidt on columns $a_1, \ldots, a_k$ of $n \times k$ matrix A
- if columns are linearly independent, get orthonormal $q_1, \ldots, q_k$
- define $n \times k$ matrix Q with columns $q_1, \ldots, q_k$
- $Q^T Q = I$
- from Gram–Schmidt algorithm

$$
\begin{aligned}
a_i &= (q_1^T a_i) q_1 + \cdots + (q_{i-1}^T a_i) q_{i-1} + \|\tilde{q}_i\| q_i \\
&= R_{1i} q_1 + \cdots + R_{ii} q_i
\end{aligned}
$$

  with $R_{ij} = q_i^T a_j$ for $i < j$ and $R_{ii} = \|\tilde{q}_i\|$
- defining $R_{ij} = 0$ for $i > j$ we have $A = QR$
- R is upper triangular, with positive diagonal entries

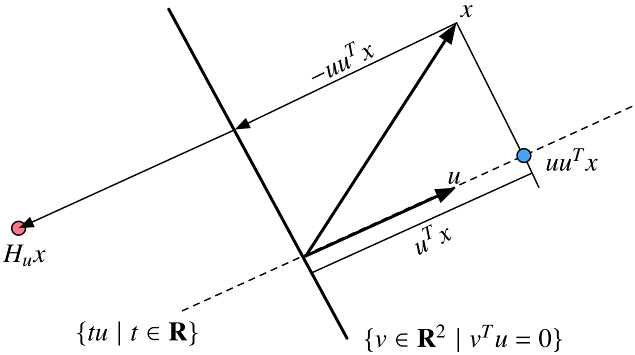
QR factorization

- $A = QR$ is called *QR factorization* of A
- factors satisfy $Q^T Q = I$, R upper triangular with positive diagonal entries
- can be computed using Gram–Schmidt algorithm (or some variations)
- has a *huge* number of uses, which we'll see soon

Outline

Householder transforms for QR factorization

- matrix multiplication to *compute* QR factorization
- Householder transform: for unit vector u, $H_u = I - 2uu^T$



Properties of Householder transforms

- symmetric and orthogonal:

$$H_u^T = (I - 2uu^T)^T = I - 2(uu^T)^T = I - 2uu^T$$

  and

$$H_u^T H_u = H_u^2 = I - 4uu^T uu^T + 4uu^T uu^T uu^T = I$$

- product of any number is orthogonal: for unit norm vectors $u_1, u_2, \ldots, u_k$

$$Q = H_{u_1} H_{u_2} \cdots H_{u_k}$$

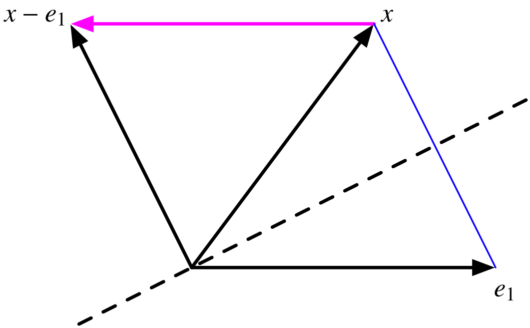
  satisfies

$$\begin{aligned} Q^T Q &= H_{u_k} H_{u_{k-1}} \cdots H_{u_2} H_{u_1} H_{u_1} H_{u_2} \cdots H_{u_k} \\ &= H_{u_k} H_{u_{k-1}} \cdots H_{u_2} H_{u_2} \cdots H_{u_{k-1}} H_{u_k} = \cdots = I. \end{aligned}$$

Annihilation with Householders

- for unit-norm n-vector $x \in \mathbf{R}^n$, find u so that

$$H_u x = e_1$$



- take $u = (x - e_1)/\|x - e_1\|$, verify $H_u x = e_1$

Annihilating below diagonals

- start with $n \times k$ matrix A with columns $a_1, a_2, \ldots, a_k$
- annihilate below diagonal for first column: if $a_1 \neq 0$, set $v = a_1/\|a_1\|$

$$H_u = I - 2uu^T \quad \text{for} \quad u = \frac{v - e_1}{\|v - e_1\|}$$

- then

$$H_u A = \begin{bmatrix} H_u a_1 & H_u a_2 & \cdots & H_u a_k \end{bmatrix} = \begin{bmatrix} \|a_1\| & * \\ 0 & \tilde{A}_1 \end{bmatrix}$$

- recurse: for block matrix with R_i an $i \times i$ upper triangular matrix

$$\begin{bmatrix} R_i & * \\ 0 & \tilde{A}_{i-1} \end{bmatrix}$$

then for $\tilde{A}_{i-1} = [\tilde{a}_1 \ \cdots \ \tilde{a}_{k-i}]$, choosing unit $\tilde{u}$ as above for $\tilde{a}_1$:

$$\begin{bmatrix} I & 0 \\ 0 & H_{\tilde{u}} \end{bmatrix} \begin{bmatrix} R_i & * \\ 0 & \tilde{A}_{i-1} \end{bmatrix} = \begin{bmatrix} R_i & * \\ 0 & \begin{pmatrix} \|\tilde{a}_1\| & * \\ 0 & \tilde{A}_i \end{pmatrix} \end{bmatrix}$$

Householder algorithm for QR factorization

---

given $n \times k$ matrix A with columns $a_1, \ldots, a_k$
initialize $\tilde{A}_0 = A$
for $i = 0, \ldots, \min\{k, n-2\}$

1. *find first column* $\tilde{a} \in \mathbf{R}^{n-i}$ of $\tilde{A}_i$
2. *test for linear dependence*: if $\tilde{a} = 0$, recurse on $\tilde{A}_{i+1}$
   (the lower right $(n-i-1) \times (n-i-1)$ block of $\tilde{A}_i$)
3. *normalize:* $v = \tilde{a}/\|\tilde{a}\|$
4. *find projector:* $u = (v - e_1)/\|v - e_1\| \in \mathbf{R}^{n-i}$
5. *construct* $(n-i) \times (n-i)$ Householder matrix $H_i = I - 2uu^T$
6. *apply* H_i to obtain

$$\begin{bmatrix} r_{ii} & * \\ 0 & \tilde{A}_i \end{bmatrix} = H_i \tilde{A}_{i-1}$$

- final output: $A = QR$, where for $m = \min\{n-2, k\}$

$$Q = H_0 \begin{bmatrix} 1 & 0 \\ 0 & H_1 \end{bmatrix} \begin{bmatrix} I_2 & 0 \\ 0 & H_2 \end{bmatrix} \cdots \begin{bmatrix} I_m & 0 \\ 0 & H_m \end{bmatrix} \quad \text{and} \quad R = Q^T A$$