6. Matrices

Outline

Matrices

Matrix-vector multiplication

Examples

Matrices

- a *matrix* is a rectangular array of numbers, *e.g.*,

$$\begin{bmatrix} 0 & 1 & -2.3 & 0.1 \\ 1.3 & 4 & -0.1 & 0 \\ 4.1 & -1 & 0 & 1.7 \end{bmatrix}$$

- its *size* is given by (row dimension) $\times$ (column dimension)
  e.g., matrix above is 3×4

- *elements* also called *entries* or *coefficients*

- B_{ij} is i,j element of matrix B

- i is the *row index*, j is the *column index*; indexes start at 1

- two matrices are *equal* (denoted with $=$) if they are the same size and corresponding entries are equal

Matrix shapes

an $m \times n$ matrix A is

- *tall* if $m > n$
- *wide* if $m < n$
- *square* if $m = n$

Column and row vectors

- we consider an $n \times 1$ matrix to be an n-vector
- we consider a 1×1 matrix to be a number
- a $1 \times n$ matrix is called a *row vector*, *e.g.*,

$$\begin{bmatrix} 1.2 & -0.3 & 1.4 & 2.6 \end{bmatrix}$$

  which is *not* the same as the (column) vector

$$\begin{bmatrix} 1.2 \\ -0.3 \\ 1.4 \\ 2.6 \end{bmatrix}$$

Columns and rows of a matrix

- suppose A is an $m \times n$ matrix with entries A_{ij} for $i = 1, \ldots, m$, $j = 1, \ldots, n$
- its jth *column* is (the m-vector)

$$\left[\begin{array}{c} A_{1j} \\ \vdots \\ A_{mj} \end{array}\right]$$

- its ith *row* is (the n-row-vector)

$$\left[\begin{array}{ccc} A_{i1} & \cdots & A_{in} \end{array}\right]$$

- *slice* of matrix: $A_{p:q,r:s}$ is the $(q-p+1) \times (s-r+1)$ matrix

$$A_{p:q,r:s} = \left[\begin{array}{cccc} A_{pr} & A_{p,r+1} & \cdots & A_{ps} \\ A_{p+1,r} & A_{p+1,r+1} & \cdots & A_{p+1,s} \\ \vdots & \vdots & & \vdots \\ A_{qr} & A_{q,r+1} & \cdots & A_{qs} \end{array}\right]$$

Block matrices

- we can form *block matrices*, whose entries are matrices, such as

$$A = \begin{bmatrix} B & C \\ D & E \end{bmatrix}$$

  where B, C, D, and E are matrices (called *submatrices* or *blocks* of A)
- matrices in each block row must have same height (row dimension)
- matrices in each block column must have same width (column dimension)
- example: if

$$B = \begin{bmatrix} 0 & 2 & 3 \end{bmatrix}, \quad C = \begin{bmatrix} -1 \end{bmatrix}, \quad D = \begin{bmatrix} 2 & 2 & 1 \\ 1 & 3 & 5 \end{bmatrix}, \quad E = \begin{bmatrix} 4 \\ 4 \end{bmatrix}$$

  then

$$\begin{bmatrix} B & C \\ D & E \end{bmatrix} = \begin{bmatrix} 0 & 2 & 3 & -1 \\ 2 & 2 & 1 & 4 \\ 1 & 3 & 5 & 4 \end{bmatrix}$$

Column and row representation of matrix

- A is an $m \times n$ matrix
- can express as block matrix with its (m-vector) columns $a_1, \ldots, a_n$

$$A = \left[\begin{array}{cccc} a_1 & a_2 & \cdots & a_n \end{array} \right]$$

- or as block matrix with its (n-row-vector) rows $b_1, \ldots, b_m$

$$A = \left[\begin{array}{c} b_1 \\ b_2 \\ \vdots \\ b_m \end{array} \right]$$

Examples

- *image:* X_{ij} is i,j pixel value in a monochrome image
- *rainfall data:* A_{ij} is rainfall at location i on day j
- *multiple asset returns:* R_{ij} is return of asset j in period i
- *contingency table:* A_{ij} is number of objects with first attribute i and second attribute j
- *feature matrix:* X_{ij} is value of feature i for entity j

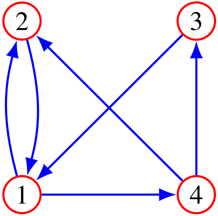
in each of these, what do the rows and columns mean?

Graph or relation

- a *relation* is a set of pairs of *objects*, labeled $1, \ldots, n$, such as

$$\mathcal{R} = \{(1,2),\ (1,3),\ (2,1),\ (2,4),\ (3,4),\ (4,1)\}$$

- same as *directed graph*



- can be represented as $n \times n$ matrix with $A_{ij} = 1$ if $(i,j) \in \mathcal{R}$

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

Special matrices

- $m \times n$ *zero matrix* has all entries zero, written as $0_{m\times n}$ or just 0
- *identity matrix* is square matrix with $I_{ii} = 1$ and $I_{ij} = 0$ for $i \neq j$, *e.g.*,

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- *sparse matrix:* most entries are zero
  - examples: 0 and I
  - can be stored and manipulated efficiently
  - $\mathbf{nnz}(A)$ is number of nonzero entries

Diagonal and triangular matrices

- *diagonal matrix*: square matrix with $A_{ij} = 0$ when $i \neq j$
- $\mathbf{diag}(a_1, \ldots, a_n)$ denotes the diagonal matrix with $A_{ii} = a_i$ for $i = 1, \ldots, n$
- example:

$$\mathbf{diag}(0.2, -3, 1.2) = \begin{bmatrix} 0.2 & 0 & 0 \\ 0 & -3 & 0 \\ 0 & 0 & 1.2 \end{bmatrix}$$

- *lower triangular matrix*: $A_{ij} = 0$ for $i < j$
- *upper triangular matrix*: $A_{ij} = 0$ for $i > j$
- examples:

$$\begin{bmatrix} 1 & -1 & 0.7 \\ 0 & 1.2 & -1.1 \\ 0 & 0 & 3.2 \end{bmatrix} \text{ (upper triangular)}, \qquad \begin{bmatrix} -0.6 & 0 \\ -0.3 & 3.5 \end{bmatrix} \text{ (lower triangular)}$$

Transpose

- the *transpose* of an $m \times n$ matrix A is denoted A^T, and defined by

$$(A^T)_{ij} = A_{ji}, \quad i = 1, \ldots, n, \quad j = 1, \ldots, m$$

- for example,

$$\begin{bmatrix} 0 & 4 \\ 7 & 0 \\ 3 & 1 \end{bmatrix}^T = \begin{bmatrix} 0 & 7 & 3 \\ 4 & 0 & 1 \end{bmatrix}$$

- transpose converts column to row vectors (and vice versa)
- $(A^T)^T = A$

Addition, subtraction, and scalar multiplication

- (just like vectors) we can add or subtract matrices of the same size:

$$(A+B)_{ij} = A_{ij} + B_{ij}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n$$

  (subtraction is similar)

- scalar multiplication:

$$(\alpha A)_{ij} = \alpha A_{ij}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n$$

- many obvious properties, *e.g.*,

$$A + B = B + A, \qquad \alpha(A+B) = \alpha A + \alpha B, \qquad (A+B)^T = A^T + B^T$$

Matrix norm

- for $m \times n$ matrix A, we define

$$\|A\| = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij}^2 \right)^{1/2}$$

- agrees with vector norm when $n = 1$
- satisfies norm properties:

$$\|\alpha A\| = |\alpha| \|A\|$$

$$\|A + B\| \leq \|A\| + \|B\|$$

$$\|A\| \geq 0$$

$$\|A\| = 0 \text{ only if } A = 0$$

- distance between two matrices: $\|A - B\|$
- (there are other matrix norms, which we won't use)

Outline

Matrices

Matrix-vector multiplication

Examples

Matrix-vector product

- *matrix-vector product* of $m \times n$ matrix A, n-vector x, denoted $y = Ax$, with

$$y_i = A_{i1}x_1 + \cdots + A_{in}x_n, \quad i = 1, \ldots, m$$

- for example,

$$\begin{bmatrix} 0 & 2 & -1 \\ -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 3 \\ -4 \end{bmatrix}$$

Row interpretation

- $y = Ax$ can be expressed as

$$y_i = b_i^T x, \quad i = 1, \ldots, m$$

  where $b_1^T, \ldots, b_m^T$ are rows of A

- so $y = Ax$ is a 'batch' inner product of all rows of A with x
- example: $A\mathbf{1}$ is vector of row sums of matrix A

Column interpretation

- $y = Ax$ can be expressed as

$$y = x_1 a_1 + x_2 a_2 + \cdots + x_n a_n$$

  where $a_1, \ldots, a_n$ are columns of A

- so $y = Ax$ is linear combination of columns of A, with coefficients $x_1, \ldots, x_n$
- important example: $Ae_j = a_j$
- columns of A are linearly independent if $Ax = 0$ implies $x = 0$

Outline

Matrices

Matrix-vector multiplication

Examples

General examples

- $0x = 0$, *i.e.*, multiplying by zero matrix gives zero
- $Ix = x$, *i.e.*, multiplying by identity matrix does nothing
- inner product $a^T b$ is matrix-vector product of $1 \times n$ matrix a^T and n-vector b
- $\tilde{x} = Ax$ is de-meaned version of x, with

$$A = \begin{bmatrix} 1 - 1/n & -1/n & \cdots & -1/n \\ -1/n & 1 - 1/n & \cdots & -1/n \\ \vdots & & \ddots & \vdots \\ -1/n & -1/n & \cdots & 1 - 1/n \end{bmatrix}$$

Difference matrix

- $(n-1) \times n$ *difference matrix* is

$$D = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 & 0 \\ & & \ddots & \ddots & & & \\ & & & \ddots & \ddots & & \\ 0 & 0 & 0 & \cdots & -1 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & -1 & 1 \end{bmatrix}$$

  $y = Dx$ is $(n-1)$-vector of differences of consecutive entries of x:

$$Dx = \begin{bmatrix} x_2 - x_1 \\ x_3 - x_2 \\ \vdots \\ x_n - x_{n-1} \end{bmatrix}$$

- *Dirichlet energy:* $\|Dx\|^2$ is measure of wiggliness for x a time series

Return matrix – portfolio vector

- R is $T \times n$ matrix of asset returns
- R_{ij} is return of asset j in period i (say, in percentage)
- n-vector w gives portfolio (investments in the assets)
- T-vector Rw is time series of the portfolio return
- $\mathbf{avg}(Rw)$ is the portfolio (mean) return, $\mathbf{std}(Rw)$ is its risk

Feature matrix – weight vector

- $X = [x_1 \ \cdots \ x_N]$ is $n \times N$ *feature matrix*
- column x_j is feature n-vector for object or example j
- X_{ij} is value of feature i for example j
- n-vector w is weight vector
- $s = X^T w$ is vector of scores for each example; $s_j = x_j^T w$

Input – output matrix

- A is $m \times n$ matrix
- $y = Ax$
- n-vector x is *input* or *action*
- m-vector y is *output* or *result*
- A_{ij} is the factor by which y_i depends on x_j
- A_{ij} is the *gain* from input j to output i
- *e.g.*, if A is lower triangular, then y_i only depends on $x_1, \ldots, x_i$

Complexity

- $m \times n$ matrix stored A as $m \times n$ array of numbers
  (for sparse A, store only $\mathbf{nnz}(A)$ nonzero values)
- matrix addition, scalar-matrix multiplication cost mn flops
- matrix-vector multiplication costs $m(2n-1) \approx 2mn$ flops
  (for sparse A, around $2\mathbf{nnz}(A)$ flops)

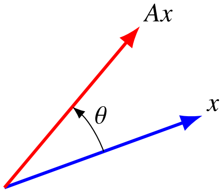
7. Matrix examples

Outline

Geometric transformations

- many geometric transformations and mappings of 2-D and 3-D vectors can be represented via matrix multiplication $y = Ax$
- for example, rotation by θ:

$$y = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} x$$



(to get the entries, look at Ae_1 and Ae_2)

Outline

Geometric transformations

Selectors

Incidence matrix

Convolution

Selectors

- an $m \times n$ *selector matrix*: each row is a unit vector (transposed)

$$A = \begin{bmatrix} e_{k_1}^T \\ \vdots \\ e_{k_m}^T \end{bmatrix}$$

- multiplying by A selects entries of x:

$$Ax = (x_{k_1}, x_{k_2}, \ldots, x_{k_m})$$

- example: the $m \times 2m$ matrix

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

  'down-samples' by 2: if x is a $2m$-vector then $y = Ax = (x_1, x_3, \ldots, x_{2m-1})$

- other examples: image cropping, permutation, ...

Outline

Geometric transformations

Selectors

Incidence matrix

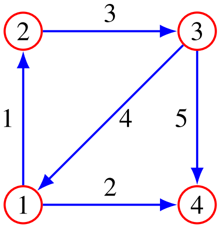
Convolution

Incidence matrix

- graph with n vertices or nodes, m (directed) edges or links
- incidence matrix is $n \times m$ matrix

$$A_{ij} = \begin{cases} 1 & \text{edge } j \text{ points to node } i \\ -1 & \text{edge } j \text{ points from node } i \\ 0 & \text{otherwise} \end{cases}$$

- example with $n = 4$, $m = 5$:



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

Flow conservation

- m-vector x gives flows (of something) along the edges
- examples: heat, money, power, mass, people, ...
- $x_j > 0$ means flow follows edge direction
- Ax is n-vector that gives the total or net flows
- $(Ax)_i$ is the net flow into node i
- $Ax = 0$ is *flow conservation*; x is called a *circulation*

Potentials and Dirichlet energy

- suppose v is an n-vector, called a *potential*
- v_i is potential value at node i
- $u = A^T v$ is an m-vector of *potential differences* across the m edges
- $u_j = v_l - v_k$, where edge j goes from k to node l
- *Dirichlet energy* is $\mathcal{D}(v) = \|A^T v\|^2$,

  $$\mathcal{D}(v) = \sum_{\text{edges } (k,l)} (v_l - v_k)^2$$

  (sum of squares of potential differences across the edges)
- $\mathcal{D}(v)$ is small when potential values of neighboring nodes are similar

Outline

Geometric transformations

Selectors

Incidence matrix

Convolution

Convolution

- for n-vector a, m-vector b, the *convolution* $c = a * b$ is the $(n+m-1)$-vector

$$c_k = \sum_{i+j=k+1} a_i b_j, \quad k = 1, \ldots, n+m-1$$

- for example with $n = 4$, $m = 3$, we have

$$\begin{aligned}
c_1 &= a_1 b_1 \\
c_2 &= a_1 b_2 + a_2 b_1 \\
c_3 &= a_1 b_3 + a_2 b_2 + a_3 b_1 \\
c_4 &= a_2 b_3 + a_3 b_2 + a_4 b_1 \\
c_5 &= a_3 b_3 + a_4 b_2 \\
c_6 &= a_4 b_3
\end{aligned}$$

- example: $(1, 0, -1) * (2, 1, -1) = (2, 1, -3, -1, 1)$

Polynomial multiplication

- a and b are coefficients of two polynomials:

$$p(x) = a_1 + a_2 x + \cdots + a_n x^{n-1}, \qquad q(x) = b_1 + b_2 x + \cdots + b_m x^{m-1}$$

- convolution $c = a * b$ gives the coefficients of the product $p(x)q(x)$:

$$p(x)q(x) = c_1 + c_2 x + \cdots + c_{n+m-1} x^{n+m-2}$$

- this gives simple proofs of many properties of convolution; for example,

$$a * b = b * a$$
$$(a * b) * c = a * (b * c)$$
$$a * b = 0 \text{ only if } a = 0 \text{ or } b = 0$$

Toeplitz matrices

- can express $c = a * b$ using matrix-vector multiplication as $c = T(b)a$, with

$$T(b) = \begin{bmatrix} b_1 & 0 & 0 & 0 \\ b_2 & b_1 & 0 & 0 \\ b_3 & b_2 & b_1 & 0 \\ 0 & b_3 & b_2 & b_1 \\ 0 & 0 & b_3 & b_2 \\ 0 & 0 & 0 & b_3 \end{bmatrix}$$

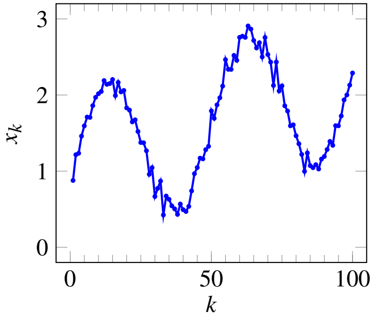
- $T(b)$ is a Toeplitz matrix (values on diagonals are equal)

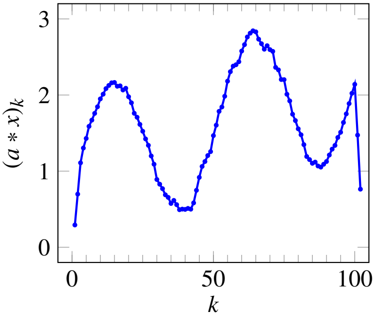
Moving average of time series

- n-vector x represents a time series
- convolution $y = a * x$ with $a = (1/3, 1/3, 1/3)$ is 3-period *moving average*:

$$y_k = \frac{1}{3}(x_k + x_{k-1} + x_{k-2}), \quad k = 1, 2, \ldots, n+2$$

(with x_k interpreted as zero for $k < 1$ and $k > n$)





Input-output convolution system

- m-vector u represents a time series *input*
- $m+n-1$ vector y represents a time series *output*
- $y = h * u$ is a *convolution model*
- n-vector h is called the *system impulse response*
- we have

$$y_i = \sum_{j=1}^{n} u_{i-j+1} h_j$$

(interpreting u_k as zero for $k < n$ or $k > n$)

- interpretation: y_i, output at time i is a linear combination of $u_i, \ldots, u_{i-n+1}$
- h_3 is the factor by which current output depends on what the input was 2 time steps before

8. Linear equations

Outline

Linear functions

Linear function models

Linear equations

Balancing chemical equations

Superposition

- $f : \mathbf{R}^n \to \mathbf{R}^m$ means f is a function that maps n-vectors to m-vectors
- we write $f(x) = (f_1(x), \ldots, f_m(x))$ to emphasize components of $f(x)$
- we write $f(x) = f(x_1, \ldots, x_n)$ to emphasize components of x
- f satisfies *superposition* if for all x, y, α, β

$$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y)$$

  (this innocent looking equation says a lot ...)
- such an f is called *linear*

Matrix-vector product function

- with A an $m \times n$ matrix, define f as $f(x) = Ax$
- f is linear:

$$\begin{aligned} f(\alpha x + \beta y) &= A(\alpha x + \beta y) \\ &= A(\alpha x) + A(\beta y) \\ &= \alpha(Ax) + \beta(Ay) \\ &= \alpha f(x) + \beta f(y) \end{aligned}$$

- converse is true: if $f : \mathbf{R}^n \to \mathbf{R}^m$ is linear, then

$$\begin{aligned} f(x) &= f(x_1 e_1 + x_2 e_2 + \cdots + x_n e_n) \\ &= x_1 f(e_1) + x_2 f(e_2) + \cdots + x_n f(e_n) \\ &= Ax \end{aligned}$$

with $A = \begin{bmatrix} f(e_1) & f(e_2) & \cdots & f(e_n) \end{bmatrix}$

Examples

- reversal: $f(x) = (x_n, x_{n-1}, \ldots, x_1)$

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

- running sum: $f(x) = (x_1, x_1 + x_2, x_1 + x_2 + x_3, \ldots, x_1 + x_2 + \cdots + x_n)$

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

Affine functions

- function $f : \mathbf{R}^n \to \mathbf{R}^m$ is *affine* if it is a linear function plus a constant, *i.e.*,

$$f(x) = Ax + b$$

- same as:

$$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y)$$

  holds for all x, y, and α, β with $\alpha + \beta = 1$

- can recover A and b from f using

$$\begin{aligned} A &= \left[\begin{array}{cccc} f(e_1) - f(0) & f(e_2) - f(0) & \cdots & f(e_n) - f(0) \end{array} \right] \\ b &= f(0) \end{aligned}$$

- affine functions sometimes (incorrectly) called linear

Outline

Linear functions

Linear function models

Linear equations

Balancing chemical equations

Linear and affine functions models

- in many applications, relations between n-vectors and m vectors are *approximated* as linear or affine
- sometimes the approximation is excellent, and holds over large ranges of the variables (*e.g.*, electromagnetics)
- sometimes the approximation is reasonably good over smaller ranges (*e.g.*, aircraft dynamics)
- in other cases it is quite approximate, but still useful (*e.g.*, econometric models)

Price elasticity of demand

- n goods or services
- prices given by n-vector p, demand given as n-vector d
- $\delta_i^{\text{price}} = (p_i^{\text{new}} - p_i)/p_i$ is fractional changes in prices
- $\delta_i^{\text{dem}} = (d_i^{\text{new}} - d_i)/d_i$ is fractional change in demands
- *price-demand elasticity model*: $\delta^{\text{dem}} = E\delta^{\text{price}}$

- what do the following mean?

$$E_{11} = -0.3, \qquad E_{12} = +0.1, \qquad E_{23} = -0.05$$

Taylor series approximation

- suppose $f : \mathbf{R}^n \to \mathbf{R}^m$ is differentiable
- first order Taylor approximation $\hat{f}$ of f near z:

$$\begin{aligned}\hat{f}_i(x) &= f_i(z) + \frac{\partial f_i}{\partial x_1}(z)(x_1 - z_1) + \cdots + \frac{\partial f_i}{\partial x_n}(z)(x_n - z_n) \\ &= f_i(z) + \nabla f_i(z)^T(x - z)\end{aligned}$$

- in compact notation: $\hat{f}(x) = f(z) + Df(z)(x - z)$
- $Df(z)$ is the $m \times n$ *derivative* or *Jacobian* matrix of f at z

$$Df(z)_{ij} = \frac{\partial f_i}{\partial x_j}(z), \quad i = 1, \ldots, m, \quad j = 1, \ldots, n$$

- $\hat{f}(x)$ is a very good approximation of $f(x)$ for x near z
- $\hat{f}(x)$ is an affine function of x

Regression model

- regression model: $\hat{y} = x^T \beta + v$
  - x is n-vector of features or regressors
  - β is n-vector of model parameters; v is offset parameter
  - (scalar) $\hat{y}$ is our prediction of y
- now suppose we have N *examples* or *samples* $x^{(1)}, \ldots, x^{(N)}$, and associated responses $y^{(1)}, \ldots, y^{(N)}$
- associated predictions are $\hat{y}^{(i)} = (x^{(i)})^T \beta + v$
- write as $\hat{y}^{\mathrm{d}} = X^T \beta + v\mathbf{1}$
  - X is feature matrix with columns $x^{(1)}, \ldots, x^{(N)}$
  - y^{d} is N-vector of responses $(y^{(1)}, \ldots, y^{(N)})$
  - $\hat{y}^{\mathrm{d}}$ is N-vector of predictions $(\hat{y}^{(1)}, \ldots, \hat{y}^{(N)})$
- *prediction error* (vector) is $y^{\mathrm{d}} - \hat{y}^{\mathrm{d}} = y^{\mathrm{d}} - X^T \beta - v\mathbf{1}$

Outline

Linear functions

Linear function models

Linear equations

Balancing chemical equations

Systems of linear equations

- set (or *system*) of m linear equations in n variables $x_1, \ldots, x_n$:

$$\begin{array}{rcl} A_{11}x_1 + A_{12}x_2 + \cdots + A_{1n}x_n & = & b_1 \\ A_{21}x_1 + A_{22}x_2 + \cdots + A_{2n}x_n & = & b_2 \\ & \vdots & \\ A_{m1}x_1 + A_{m2}x_2 + \cdots + A_{mn}x_n & = & b_m \end{array}$$

- n-vector x is called the variable or unknowns
- A_{ij} are the *coefficients*; A is the coefficient matrix
- b is called the *right-hand side*
- can express very compactly as $Ax = b$

Systems of linear equations

- systems of linear equations classified as
  - under-determined if $m < n$ (A wide)
  - square if $m = n$ (A square)
  - over-determined if $m > n$ (A tall)
- x is called a *solution* if $Ax = b$
- depending on A and b, there can be
  - no solution
  - one solution
  - many solutions
- we'll see how to solve linear equations later

Outline

Linear functions

Linear function models

Linear equations

Balancing chemical equations

Chemical equations

- a chemical reaction involves p reactants, q products (molecules)
- expressed as

$$a_1 R_1 + \cdots + a_p R_p \longrightarrow b_1 P_1 + \cdots + b_q P_q$$

- $R_1, \ldots, R_p$ are reactants
- $P_1, \ldots, P_q$ are products
- $a_1, \ldots, a_p, b_1, \ldots, b_q$ are positive coefficients
- coefficients usually integers, but can be scaled
  - *e.g.*, multiplying all coefficients by $1/2$ doesn't change the reaction

Example: electrolysis of water

$$2H_2O \longrightarrow 2H_2 + O_2$$

- one reactant: water (H_2O)
- two products: hydrogen (H_2) and oxygen (O_2)
- reaction consumes 2 water molecules and produces 2 hydrogen molecules and 1 oxygen molecule

Balancing equations

- each molecule (reactant/product) contains specific numbers of (types of) atoms, given in its formula
  - *e.g.*, $\mathrm{H_2O}$ contains two H and one O
- *conservation of mass*: total number of each type of atom in a chemical equation must *balance*
- for each atom, total number on LHS must equal total on RHS
- *e.g.*, electrolysis reaction is balanced:
  - 4 units of H on LHS and RHS
  - 2 units of O on LHS and RHS
- finding (nonzero) coefficients to achieve balance is called *balancing* equations

Reactant and product matrices

- consider reaction with m types of atoms, p reactants, q products
- $m \times p$ *reactant matrix* R is defined by

$$R_{ij} = \text{number of atoms of type } i \text{ in reactant } R_j,$$

  for $i = 1, \ldots, m$ and $j = 1, \ldots, p$

- with $a = (a_1, \ldots, a_p)$ (vector of reactant coefficients)

$$Ra = \text{(vector of) total numbers of atoms of each type in reactants}$$

- define product $m \times q$ matrix P in similar way
- m-vector Pb is total numbers of atoms of each type in products
- conservation of mass is $Ra = Pb$

Balancing equations via linear equations

- conservation of mass is

$$\begin{bmatrix} R & -P \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = 0$$

- simple solution is $a = b = 0$
- to find a nonzero solution, set any coefficient (say, a_1) to be 1
- balancing chemical equations can be expressed as solving a set of $m + 1$ linear equations in $p + q$ variables

$$\begin{bmatrix} R & -P \\ e_1^T & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = e_{m+1}$$

(we ignore here that a_i and b_i should be nonnegative integers)

Conservation of charge

- can extend to include charge, *e.g.*, $\mathrm{Cr_2O_7^{2-}}$ has charge -2
- *conservation of charge*: total charge on each side of reaction must balance
- we can simply treat charge as another type of atom to balance

Example

$$a_1 Cr_2O_7^{2-} + a_2 Fe^{2+} + a_3 H^+ \longrightarrow b_1 Cr^{3+} + b_2 Fe^{3+} + b_3 H_2O$$

- 5 atoms/charge: Cr, O, Fe, H, charge
- reactant and product matrix:

$$R = \begin{bmatrix} 2 & 0 & 0 \\ 7 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -2 & 2 & 1 \end{bmatrix}, \qquad P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \\ 3 & 3 & 0 \end{bmatrix}$$

- balancing equations (including $a_1 = 1$ constraint)

$$\begin{bmatrix} 2 & 0 & 0 & -1 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -2 \\ -2 & 2 & 1 & -3 & -3 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Balancing equations example

- solving the system yields

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 6 \\ 14 \\ 2 \\ 6 \\ 7 \end{bmatrix}$$

- the balanced equation is

$$\mathrm{Cr_2O_7^{2-}} + 6\mathrm{Fe^{2+}} + 14\mathrm{H^+} \longrightarrow 2\mathrm{Cr^{3+}} + 6\mathrm{Fe^{3+}} + 7\mathrm{H_2O}$$

9. Linear dynamical systems

Outline

Linear dynamical systems

Population dynamics

Epidemic dynamics

State sequence

- sequence of n-vectors $x_1, x_2, \ldots$
- t denotes time or period
- x_t is called *state* at time t; sequence is called *state trajectory*
- assuming t is current time,
  - x_t is current state
  - x_{t-1} is previous state
  - x_{t+1} is next state
- examples: x_t represents
  - age distribution in a population
  - economic output in n sectors
  - mechanical variables

Linear dynamics

- linear dynamical system:

$$x_{t+1} = A_t x_t, \quad t = 1, 2, \ldots$$

- A_t are $n \times n$ dynamics matrices
- $(A_t)_{ij}(x_t)_j$ is contribution to $(x_{t+1})_i$ from $(x_t)_j$
- system is called *time-invariant* if $A_t = A$ doesn't depend on time
- can simulate evolution of x_t using recursion $x_{t+1} = A_t x_t$

Variations

- linear dynamical system with input

$$x_{t+1} = A_t x_t + B_t u_t + c_t, \quad t = 1, 2, \ldots$$

  - u_t is an *input* m-vector
  - B_t is $n \times m$ *input matrix*
  - c_t is *offset*

- K-Markov model:

$$x_{t+1} = A_1 x_t + \cdots + A_K x_{t-K+1}, \quad t = K, K+1, \ldots$$

  - next state depends on current state and $K - 1$ previous states
  - also known as *auto-regresssive model*
  - for $K = 1$, this is the standard linear dynamical system $x_{t+1} = Ax_t$

Outline

Linear dynamical systems

Population dynamics

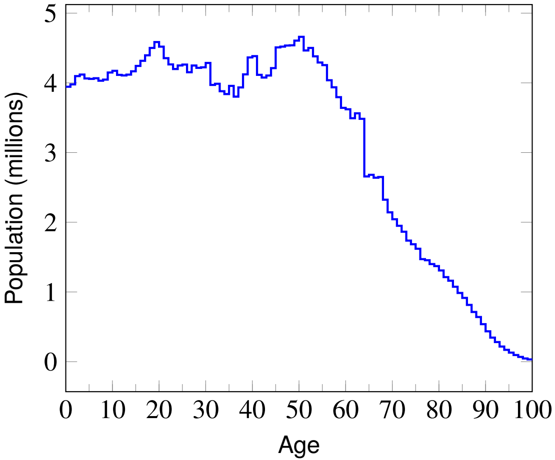
Epidemic dynamics

Population distribution

- $x_t \in \mathbf{R}^{100}$ gives population distribution in year $t = 1, \ldots, T$
- $(x_t)_i$ is the number of people with age $i - 1$ in year t (say, on January 1)
- total population in year t is $\mathbf{1}^T x_t$
- number of people age 70 or older in year t is $(0_{70}, \mathbf{1}_{30})^T x_t$

Population distribution of the U.S.

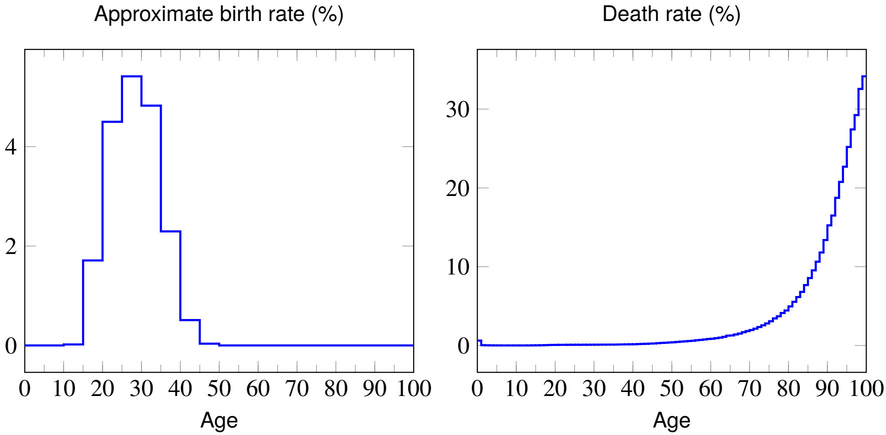
(from 2010 census)



Birth and death rates

- birth rate $b \in \mathbf{R}^{100}$, death (or mortality) rate $d \in \mathbf{R}^{100}$
- b_i is the number of births per person with age $i - 1$
- d_i is the portion of those aged $i - 1$ who will die this year (we'll take $d_{100} = 1$)
- b and d can vary with time, but we'll assume they are constant

Birth and death rates in the U.S.



Dynamics

- let's find next year's population distribution x_{t+1} (ignoring immigration)
- number of 0-year-olds next year is total births this year:

$$(x_{t+1})_1 = b^T x_t$$

- number of i-year-olds next year is number of $(i-1)$-year-olds this year, minus those who die:

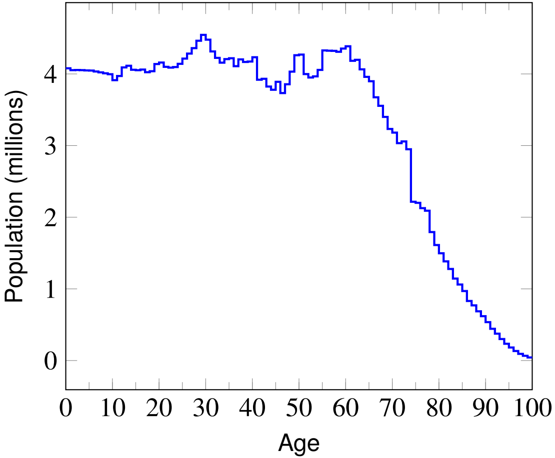
$$(x_{t+1})_{i+1} = (1-d_i)(x_t)_i, \quad i = 1, \ldots, 99$$

- $x_{t+1} = Ax_t$, where

$$A = \begin{bmatrix} b_1 & b_2 & \cdots & b_{99} & b_{100} \\ 1-d_1 & 0 & \cdots & 0 & 0 \\ 0 & 1-d_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1-d_{99} & 0 \end{bmatrix}$$

Predicting future population distributions

predicting U.S. 2020 distribution from 2010 (ignoring immigration)



Outline

Linear dynamical systems

Population dynamics

Epidemic dynamics

SIR model

- 4-vector x_t gives proportion of population in 4 infection states

Susceptible:	can acquire the disease the next day
Infected:	have the disease
Recovered:	had the disease, recovered, now immune
Deceased:	had the disease, and unfortunately died

- sometimes called *SIR model*
- *e.g.*, $x_t = (0.75, 0.10, 0.10, 0.05)$

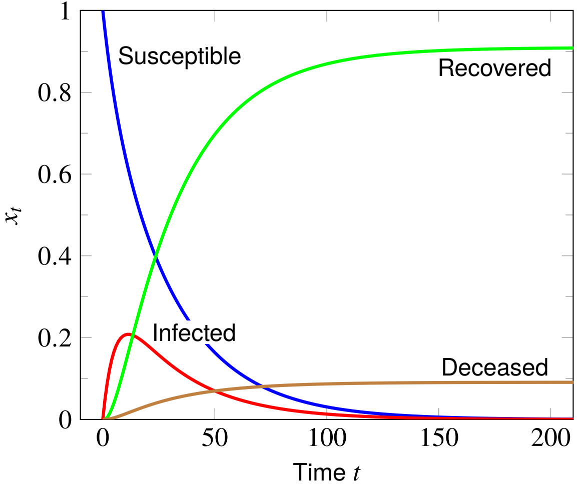
Epidemic dynamics

over each day,

- among susceptible population,
  - 5% acquires the disease
  - 95% remain susceptible
- among infected population,
  - 1% dies
  - 10% recovers with immunity
  - 4% recover without immunity (*i.e.*, become susceptible)
  - 85% remain infected
- 100% of immune and dead people remain in their state
- epidemic dynamics as linear dynamical system

$$x_{t+1} = \begin{bmatrix} 0.95 & 0.04 & 0 & 0 \\ 0.05 & 0.85 & 0 & 0 \\ 0 & 0.10 & 1 & 0 \\ 0 & 0.01 & 0 & 1 \end{bmatrix} x_t$$

Simulation from $x_1 = (1, 0, 0, 0)$



10. Matrix multiplication

Outline

Matrix multiplication

Composition of linear functions

Matrix powers

QR factorization

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

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

- 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

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:

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

- $\Delta = D_{n-1} D_n$ 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 powers

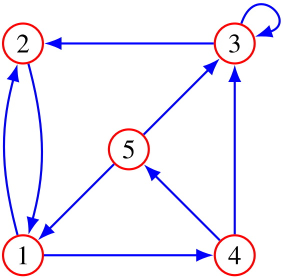
- for A square, A^2 means AA, and same for higher powers
- with convention $A^0 = I$ we have $A^k A^l = A^{k+l}$
- negative powers later; fractional powers in other courses

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

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

11. Matrix inverses

Outline

Left inverses

- a number x that satisfies $xa = 1$ is called the inverse of a
- inverse (*i.e.*, $1/a$) exists if and only if $a \neq 0$, and is unique
- a matrix X that satisfies $XA = I$ is called a *left inverse* of A
- if a left inverse exists we say that A is *left-invertible*
- example: the matrix

$$A = \begin{bmatrix} -3 & -4 \\ 4 & 6 \\ 1 & 1 \end{bmatrix}$$

  has two different left inverses:

$$B = \frac{1}{9}\begin{bmatrix} -11 & -10 & 16 \\ 7 & 8 & -11 \end{bmatrix}, \qquad C = \frac{1}{2}\begin{bmatrix} 0 & -1 & 6 \\ 0 & 1 & -4 \end{bmatrix}$$

Left inverse and column independence

- if A has a left inverse C then the columns of A are linearly independent
- to see this: if $Ax = 0$ and $CA = I$ then

$$0 = C0 = C(Ax) = (CA)x = Ix = x$$

- we'll see later the converse is also true, so

  a matrix is left-invertible if and only if its columns are linearly independent

- matrix generalization of

  a number is invertible if and only if it is nonzero

- so left-invertible matrices are tall or square

Solving linear equations with a left inverse

- suppose $Ax = b$, and A has a left inverse C
- then $Cb = C(Ax) = (CA)x = Ix = x$
- so multiplying the right-hand side by a left inverse yields the solution

Example

$$A = \begin{bmatrix} -3 & -4 \\ 4 & 6 \\ 1 & 1 \end{bmatrix}, \qquad b = \begin{bmatrix} 1 \\ -2 \\ 0 \end{bmatrix}$$

- over-determined equations $Ax = b$ have (unique) solution $x = (1, -1)$
- A has two different left inverses,

$$B = \frac{1}{9} \begin{bmatrix} -11 & -10 & 16 \\ 7 & 8 & -11 \end{bmatrix}, \qquad C = \frac{1}{2} \begin{bmatrix} 0 & -1 & 6 \\ 0 & 1 & -4 \end{bmatrix}$$

- multiplying the right-hand side with the left inverse B we get

$$Bb = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

- and also

$$Cb = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Right inverses

- a matrix X that satisfies $AX = I$ is a *right inverse* of A
- if a right inverse exists we say that A is *right-invertible*
- A is right-invertible if and only if A^T is left-invertible:

$$AX = I \iff (AX)^T = I \iff X^T A^T = I$$

- so we conclude

  A is right-invertible if and only if its rows are linearly independent

- right-invertible matrices are wide or square

Solving linear equations with a right inverse

- suppose A has a right inverse B
- consider the (square or underdetermined) equations $Ax = b$
- $x = Bb$ is a solution:

$$Ax = A(Bb) = (AB)b = Ib = b$$

- so $Ax = b$ has a solution for *any* b

Example

- same A, B, C in example above
- C^T and B^T are both right inverses of A^T
- under-determined equations $A^T x = (1,2)$ has (different) solutions

$$B^T(1,2) = (1/3, 2/3, -2/3), \qquad C^T(1,2) = (0, 1/2, -1)$$

(there are many other solutions as well)

Outline

Left and right inverses

Inverse

Solving linear equations

Examples

Pseudo-inverse

Inverse

- if A has a left and a right inverse, they are unique and equal (and we say that A is *invertible*)
- so A must be square
- to see this: if $AX = I$, $YA = I$

$$X = IX = (YA)X = Y(AX) = YI = Y$$

- we denote them by A^{-1}:

$$A^{-1}A = AA^{-1} = I$$

- inverse of inverse: $(A^{-1})^{-1} = A$

Solving square systems of linear equations

- suppose A is invertible
- for any b, $Ax = b$ has the unique solution

$$x = A^{-1}b$$

- matrix generalization of simple scalar equation $ax = b$ having solution $x = (1/a)b$ (for $a \neq 0$)
- simple-looking formula $x = A^{-1}b$ is basis for many applications

Invertible matrices

the following are equivalent for a square matrix A:

- A is invertible
- columns of A are linearly independent
- rows of A are linearly independent
- A has a left inverse
- A has a right inverse

if any of these hold, all others do

Examples

- $I^{-1} = I$
- if Q is orthogonal, *i.e.*, square with $Q^T Q = I$, then $Q^{-1} = Q^T$
- 2×2 matrix A is invertible if and only $A_{11}A_{22} \neq A_{12}A_{21}$

$$A^{-1} = \frac{1}{A_{11}A_{22} - A_{12}A_{21}} \begin{bmatrix} A_{22} & -A_{12} \\ -A_{21} & A_{11} \end{bmatrix}$$

  - you need to know this formula
  - there are similar but *much* more complicated formulas for larger matrices (and no, you do not need to know them)

Non-obvious example

$$A = \begin{bmatrix} 1 & -2 & 3 \\ 0 & 2 & 2 \\ -3 & -4 & -4 \end{bmatrix}$$

- A is invertible, with inverse

$$A^{-1} = \frac{1}{30} \begin{bmatrix} 0 & -20 & -10 \\ -6 & 5 & -2 \\ 6 & 10 & 2 \end{bmatrix}.$$

- verified by checking $AA^{-1} = I$ (or $A^{-1}A = I$)
- we'll soon see how to compute the inverse

Properties

- $(AB)^{-1} = B^{-1}A^{-1}$ (provided inverses exist)
- $(A^T)^{-1} = (A^{-1})^T$ (sometimes denoted A^{-T})
- negative matrix powers: $(A^{-1})^k$ is denoted A^{-k}
- with $A^0 = I$, identity $A^kA^l = A^{k+l}$ holds for any integers k, l

Triangular matrices

- lower triangular L with nonzero diagonal entries is invertible
- so see this, write $Lx = 0$ as

$$\begin{aligned} L_{11}x_1 \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad &= 0 \\ L_{21}x_1 + L_{22}x_2 \qquad\qquad\qquad\qquad\qquad\qquad &= 0 \\ &\vdots \\ L_{n1}x_1 + L_{n2}x_2 + \cdots + L_{n,n-1}x_{n-1} + L_{nn}x_n &= 0 \end{aligned}$$

  - from first equation, $x_1 = 0$ (since $L_{11} \neq 0$)
  - second equation reduces to $L_{22}x_2 = 0$, so $x_2 = 0$ (since $L_{22} \neq 0$)
  - and so on

  this shows columns of L are linearly independent, so L is invertible
- upper triangular R with nonzero diagonal entries is invertible

Inverse via QR factorization

- suppose A is square and invertible
- so its columns are linearly independent
- so Gram–Schmidt gives QR factorization
  - $A = QR$
  - Q is orthogonal: $Q^T Q = I$
  - R is upper triangular with positive diagonal entries, hence invertible
- so we have

$$A^{-1} = (QR)^{-1} = R^{-1}Q^{-1} = R^{-1}Q^T$$

Outline

Back substitution

- suppose R is upper triangular with nonzero diagonal entries
- write out $Rx = b$ as

$$\begin{aligned} R_{11}x_1 + R_{12}x_2 + \cdots + R_{1,n-1}x_{n-1} + R_{1n}x_n &= b_1 \\ &\vdots \\ R_{n-1,n-1}x_{n-1} + R_{n-1,n}x_n &= b_{n-1} \\ R_{nn}x_n &= b_n \end{aligned}$$

- from last equation we get $x_n = b_n/R_{nn}$
- from 2nd to last equation we get

$$x_{n-1} = (b_{n-1} - R_{n-1,n}x_n)/R_{n-1,n-1}$$

- continue to get $x_{n-2}, x_{n-3}, \ldots, x_1$

Back substitution

- called *back substitution* since we find the variables in reverse order, substituting the already known values of x_i
- computes $x = R^{-1}b$
- complexity:
  - first step requires 1 flop (division)
  - 2nd step needs 3 flops
  - ith step needs $2i - 1$ flops

  total is $1 + 3 + \cdots + (2n - 1) = n^2$ flops

Solving linear equations via QR factorization

- assuming A is invertible, let's solve $Ax = b$, *i.e.*, compute $x = A^{-1}b$
- with QR factorization $A = QR$, we have

$$A^{-1} = (QR)^{-1} = R^{-1}Q^T$$

- compute $x = R^{-1}(Q^T b)$ by back substitution

Solving linear equations via QR factorization

given an $n \times n$ invertible matrix A and an n-vector b

1. *QR factorization:* compute the QR factorization $A = QR$
2. compute $Q^T b$.
3. *Back substitution:* Solve the triangular equation $Rx = Q^T b$ using back substitution

- complexity $2n^3$ (step 1), $2n^2$ (step 2), n^2 (step 3)
- total is $2n^3 + 3n^2 \approx 2n^3$

Multiple right-hand sides

- let's solve $Ax_i = b_i$, $i = 1, \ldots, k$, with A invertible
- carry out QR factorization *once* ($2n^3$ flops)
- for $i = 1, \ldots, k$, solve $Rx_i = Q^T b_i$ via back substitution ($3kn^2$ flops)
- total is $2n^3 + 3kn^2$ flops
- if k is small compared to n, *same cost as solving one set of equations*

Outline

Left and right inverses

Inverse

Solving linear equations

Examples

Pseudo-inverse

Polynomial interpolation

- let's find coefficients of a cubic polynomial

$$p(x) = c_1 + c_2 x + c_3 x^2 + c_4 x^3$$

  that satisfies

$$p(-1.1) = b_1, \qquad p(-0.4) = b_2, \qquad p(0.1) = b_3, \qquad p(0.8) = b_4$$

- write as $Ac = b$, with

$$A = \begin{bmatrix} 1 & -1.1 & (-1.1)^2 & (-1.1)^3 \\ 1 & -0.4 & (-0.4)^2 & (-0.4)^3 \\ 1 & 0.1 & (0.1)^2 & (0.1)^3 \\ 1 & 0.8 & (0.8)^2 & (0.8)^3 \end{bmatrix}$$

Polynomial interpolation

- (unique) coefficients given by $c = A^{-1}b$, with

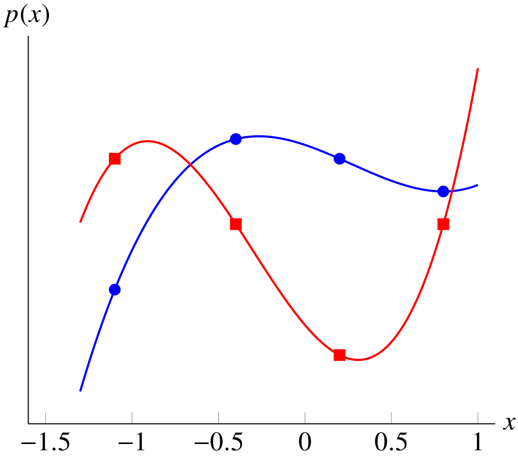
$$A^{-1} = \begin{bmatrix} -0.0370 & 0.3492 & 0.7521 & -0.0643 \\ 0.1388 & -1.8651 & 1.6239 & 0.1023 \\ 0.3470 & 0.1984 & -1.4957 & 0.9503 \\ -0.5784 & 1.9841 & -2.1368 & 0.7310 \end{bmatrix}$$

- so, *e.g.*, c_1 is not very sensitive to b_1 or b_4
- first column gives coefficients of polynomial that satisfies

$$p(-1.1) = 1, \qquad p(-0.4) = 0, \qquad p(0.1) = 0, \qquad p(0.8) = 0$$

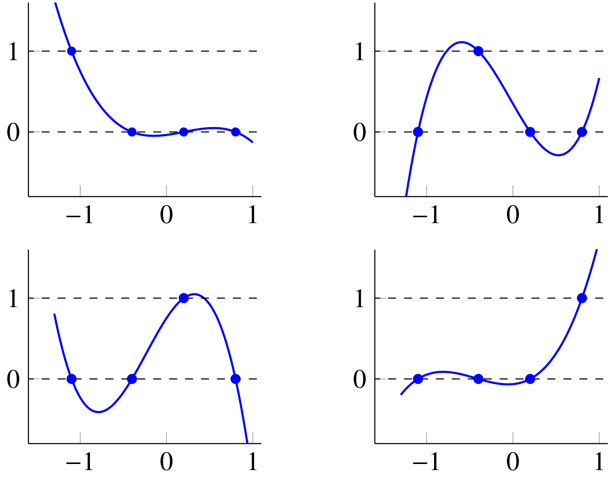
  called (first) *Lagrange polynomial*

Example



Lagrange polynomials

Lagrange polynomials associated with points -1.1, -0.4, 0.2, 0.8



Outline

Left and right inverses

Inverse

Solving linear equations

Examples

Pseudo-inverse

Invertibility of Gram matrix

- A has linearly independent columns if and only if A^TA is invertible
- to see this, we'll show that $Ax = 0 \Leftrightarrow A^TAx = 0$
- $\Rightarrow$: if $Ax = 0$ then $(A^TA)x = A^T(Ax) = A^T0 = 0$
- $\Leftarrow$: if $(A^TA)x = 0$ then

$$0 = x^T(A^TA)x = (Ax)^T(Ax) = \|Ax\|^2 = 0$$

so $Ax = 0$

Pseudo-inverse of tall matrix

- the *pseudo-inverse* of A with independent columns is

$$A^\dagger = (A^TA)^{-1}A^T$$

- it is a left inverse of A:

$$A^\dagger A = (A^TA)^{-1}A^TA = (A^TA)^{-1}(A^TA) = I$$

(we'll soon see that it's a very important left inverse of A)

- reduces to A^{-1} when A is square:

$$A^\dagger = (A^TA)^{-1}A^T = A^{-1}A^{-T}A^T = A^{-1}I = A^{-1}$$

Pseudo-inverse of wide matrix

- if A is wide, with linearly independent rows, AA^T is invertible
- pseudo-inverse is defined as

$$A^\dagger = A^T(AA^T)^{-1}$$

- $A^\dagger$ is a right inverse of A:

$$AA^\dagger = AA^T(AA^T)^{-1} = I$$

  (we'll see later it is an important right inverse)
- reduces to A^{-1} when A is square:

$$A^T(AA^T)^{-1} = A^T A^{-T} A^{-1} = A^{-1}$$

Pseudo-inverse via QR factorization

- suppose A has linearly independent columns, $A = QR$
- then $A^T A = (QR)^T (QR) = R^T Q^T Q R = R^T R$
- so

$$A^\dagger = (A^T A)^{-1} A^T = (R^T R)^{-1} (QR)^T = R^{-1} R^{-T} R^T Q^T = R^{-1} Q^T$$

- can compute $A^\dagger$ using back substitution on columns of Q^T
- for A with linearly independent rows, $A^\dagger = Q R^{-T}$