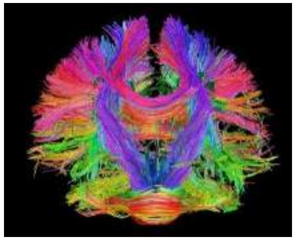



CS-570
Statistical Signal Processing

Lecture 2: Review of basic concepts

Spring Semester 2019

Grigorios Tsagkatakis

Today's Objectives

- Review of linear algebra

Disclaimer: Material used:

- Deep Learning, Ian Goodfellow, Yoshua Bengio and Aaron Courville
- Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares, Stephen Boyd ,Lieven Vandenberghe

  http://vmls-book.stanford.edu/

Vectors

- a *vector* is an ordered list of numbers
- written as

$$\begin{bmatrix} -1.1 \\ 0.0 \\ 3.6 \\ -7.2 \end{bmatrix} \quad \text{or} \quad \begin{pmatrix} -1.1 \\ 0.0 \\ 3.6 \\ -7.2 \end{pmatrix}$$

  or $(-1.1, 0, 3.6, -7.2)$

- numbers in the list are the *elements* (*entries*, *coefficients*, *components*)
- number of elements is the *size* (*dimension*, *length*) of the vector
- vector above has dimension 4; its third entry is 3.6
- vector of size n is called an n-*vector*
- numbers are called *scalars*

Zeros, ones and unit vectors

- n-vector with all entries 0 is denoted 0_n or just 0
- n-vector with all entries 1 is denoted $\mathbf{1}_n$ or just $\mathbf{1}$
- a *unit vector* has one entry 1 and all others 0
- denoted e_i where i is entry that is 1
- unit vectors of length 3:

$$e_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \qquad e_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \qquad e_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Sparsity

- a vector is *sparse* if many of its entries are 0
- can be stored and manipulated efficiently on a computer
- $\mathbf{nnz}(x)$ is number of entries that are nonzero
- examples: zero vectors, unit vectors

Linear combinations

- for vectors $a_1, \ldots, a_m$ and scalars $\beta_1, \ldots, \beta_m$,

$$\beta_1 a_1 + \cdots + \beta_m a_m$$

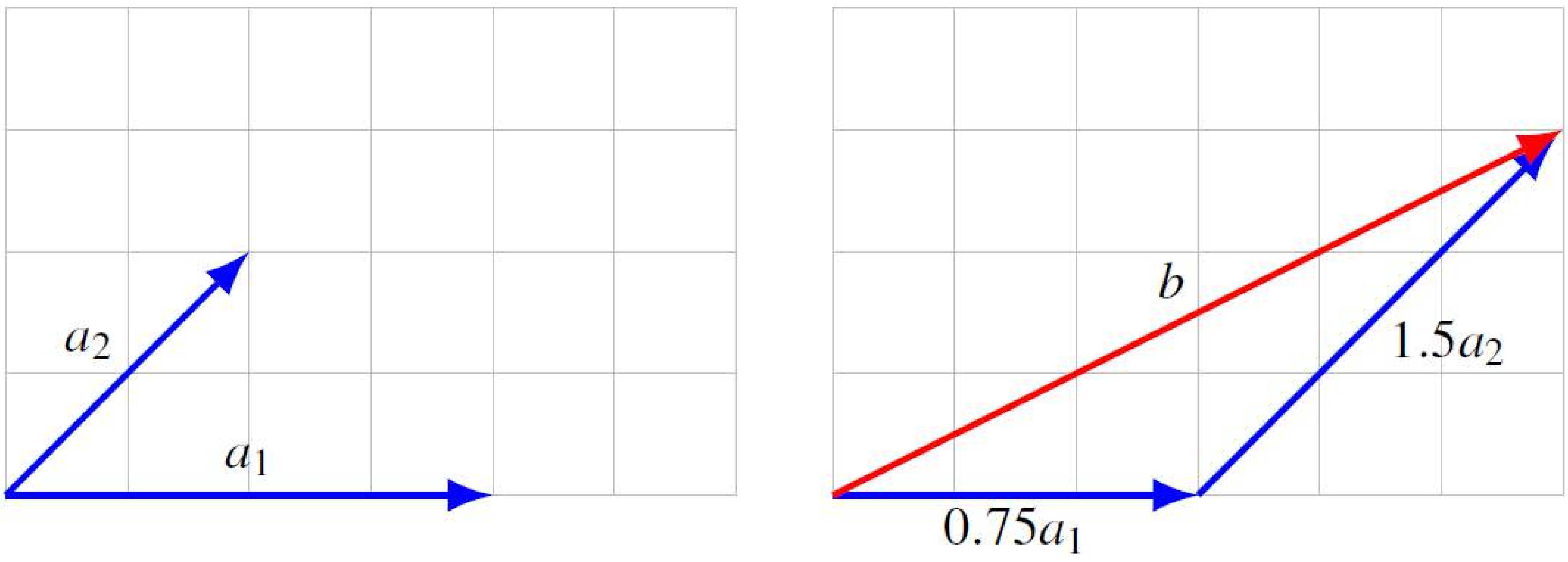
  is a *linear combination* of the vectors

- $\beta_1, \ldots, \beta_m$ are the *coefficients*
- a *very* important concept
- a simple identity: for any n-vector b,

$$b = b_1 e_1 + \cdots + b_n e_n$$

Example

two vectors a_1 and a_2, and linear combination $b = 0.75a_1 + 1.5a_2$



Flop counts

- computers store (real) numbers in *floating-point format*
- basic arithmetic operations (addition, multiplication, ...) are called *floating point operations* or flops
- complexity of an algorithm or operation: total number of flops needed, as function of the input dimension(s)
- this can be *very grossly approximated*
- crude approximation of time to execute: computer speed/flops
- current computers are around 1Gflop/sec (10^9 flops/sec)
- but this can vary by factor of 100

Complexity of vector addition, inner product

- $x + y$ needs n additions, so: n flops
- $x^T y$ needs n multiplications, $n - 1$ additions so: $2n - 1$ flops
- we simplify this to $2n$ (or even n) flops for $x^T y$
- and much less when x or y is sparse

Superposition and linear functions

- $f : \mathbf{R}^n \to \mathbf{R}$ means f is a function mapping n-vectors to numbers
- f satisfies the *superposition property* if

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

  holds for all numbers α, β, and all n-vectors x, y

- be sure to parse this very carefully!
- a function that satisfies superposition is called *linear*

The inner product function

- with a an n-vector, the function

$$f(x) = a^T x = a_1 x_1 + a_2 x_2 + \cdots + a_n x_n$$

  is the *inner product function*

- $f(x)$ is a weighted sum of the entries of x

- the inner product function is linear:

$$\begin{aligned} f(\alpha x + \beta y) &= a^T(\alpha x + \beta y) \\ &= a^T(\alpha x) + a^T(\beta y) \\ &= \alpha(a^T x) + \beta(a^T y) \\ &= \alpha f(x) + \beta f(y) \end{aligned}$$

. . .and all linear functions are inner products

- suppose $f : \mathbf{R}^n \to \mathbf{R}$ is linear
- then it can be expressed as $f(x) = a^T x$ for some a
- specifically: $a_i = f(e_i)$
- follows from

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

Affine functions

- a function that is linear plus a constant is called *affine*
- general form is $f(x) = a^T x + b$, with a an n-vector and b a scalar
- a function $f : \mathbf{R}^n \to \mathbf{R}$ is affine if and only if

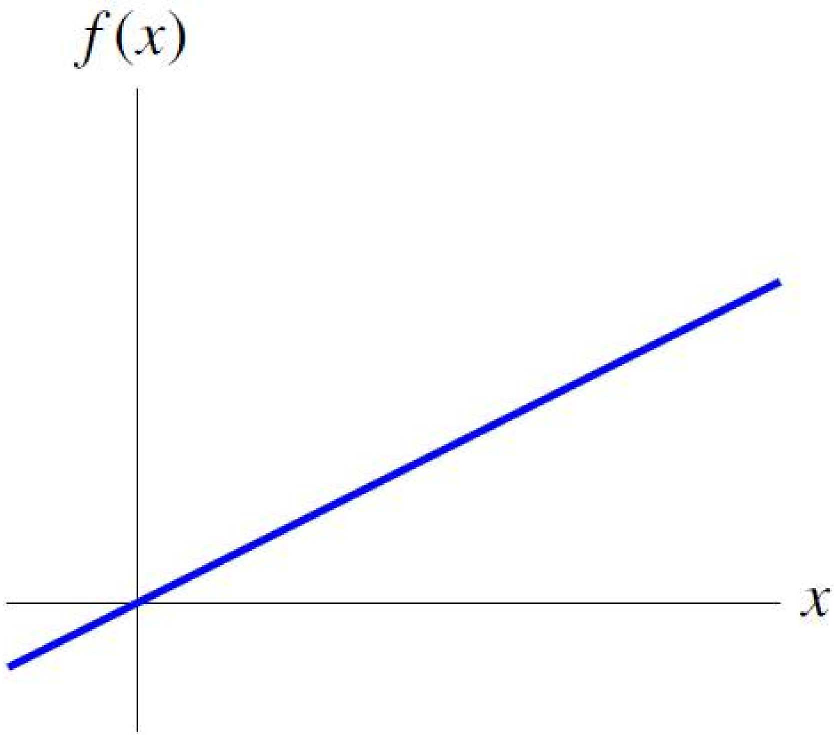
$$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y)$$

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

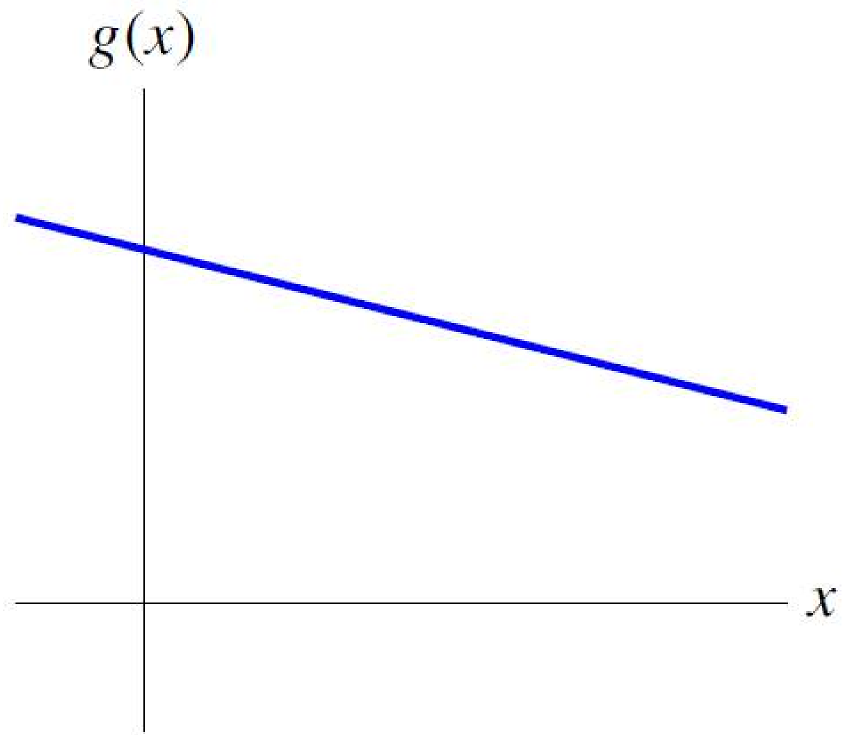
- sometimes (ignorant) people refer to affine functions as linear

Linear versus affine functions

f is linear



g is affine, not linear



First-order Taylor approximation

- suppose $f : \mathbf{R}^n \to \mathbf{R}$
- *first-order Taylor approximation* of f, near point z:

$$\hat{f}(x) = f(z) + \frac{\partial f}{\partial x_1}(z)(x_1 - z_1) + \cdots + \frac{\partial f}{\partial x_n}(z)(x_n - z_n)$$

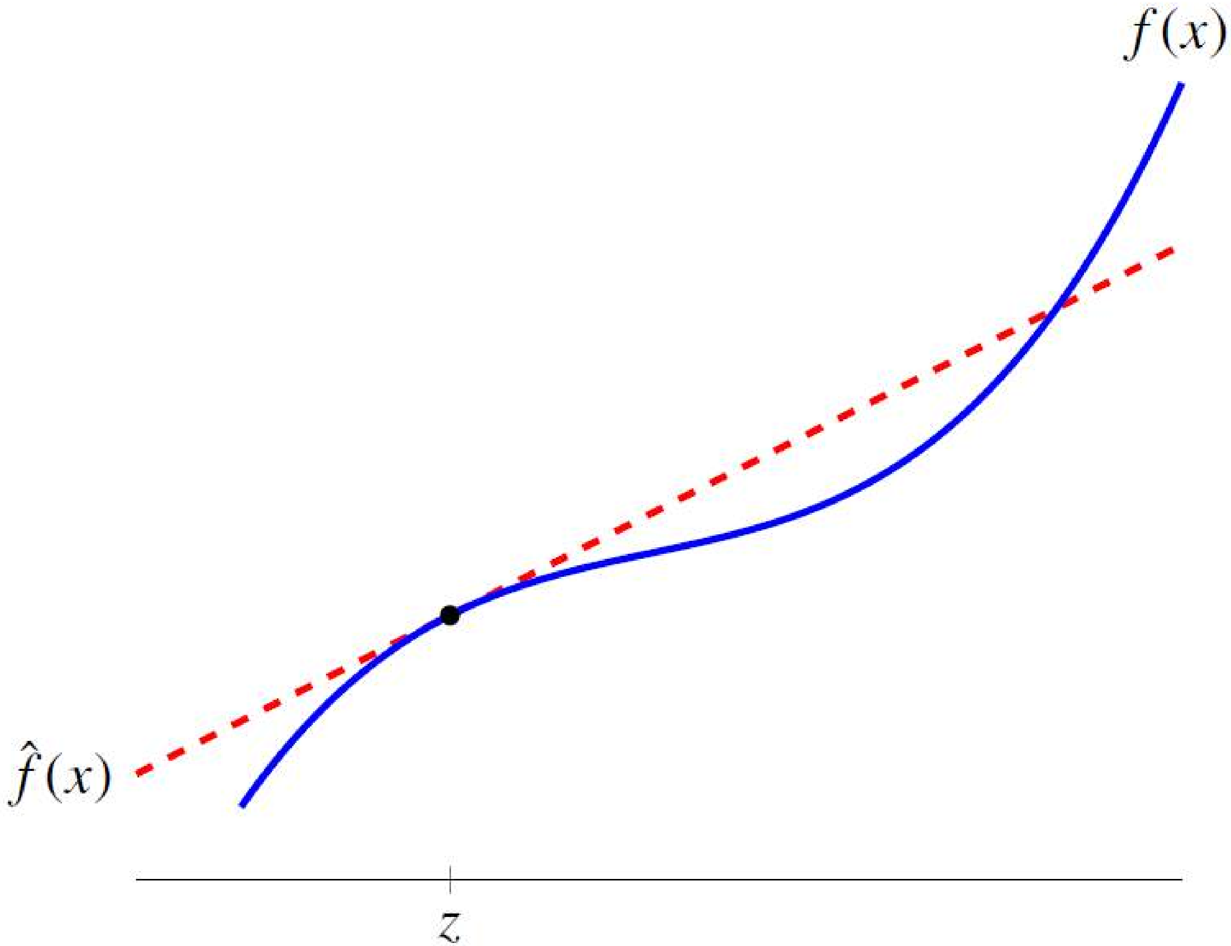
- $\hat{f}(x)$ is *very* close to $f(x)$ when x_i are all near z_i
- $\hat{f}$ is an affine function of x
- can write using inner product as

$$\hat{f}(x) = f(z) + \nabla f(z)^T (x - z)$$

where n-vector $\nabla f(z)$ is the *gradient* of f at z,

$$\nabla f(z) = \left(\frac{\partial f}{\partial x_1}(z), \ldots, \frac{\partial f}{\partial x_n}(z) \right)$$

Example



Regression model

- *regression model* is (the affine function of x)

$$\hat{y} = x^T \beta + v$$

- x is a feature vector; its elements x_i are called *regressors*
- n-vector β is the *weight vector*
- scalar v is the *offset*
- scalar $\hat{y}$ is the *prediction*
  (of some actual outcome or *dependent variable*, denoted y)

Example

- y is selling price of house in $\$1000$ (in some location, over some period)
- regressor is

$$x = (\text{house area, \# bedrooms})$$

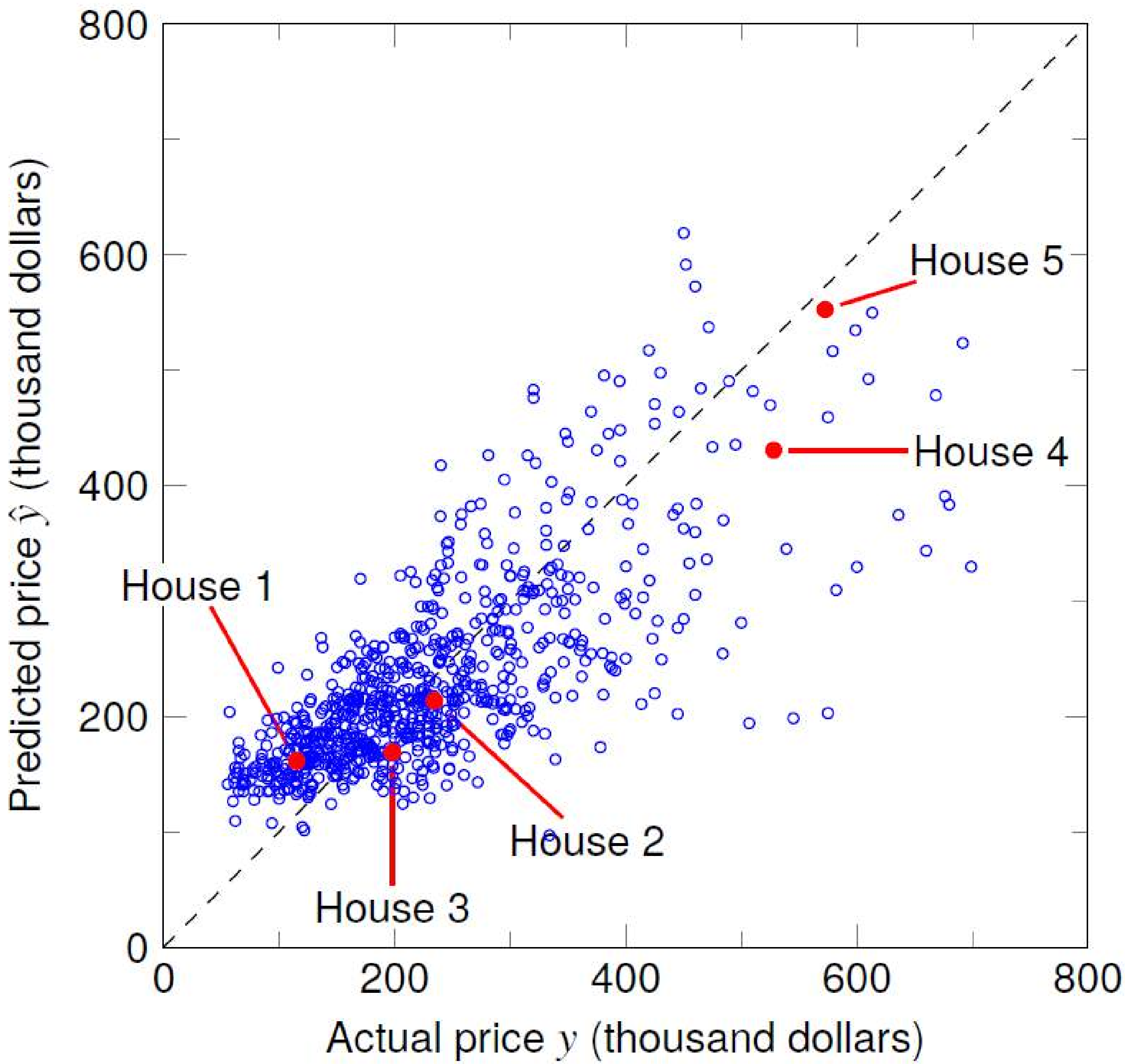
  (house area in 1000 sq.ft.)

- regression model weight vector and offset are

$$\beta = (148.73, -18.85), \qquad v = 54.40$$

- we'll see later how to guess β and v from sales data

Example



800
600
400
200
0
0
200
400
600
800
Predicted price $\hat{y}$ (thousand dollars)
Actual price y (thousand dollars)
House 1
House 2
House 3
House 4
House 5

Example

House	x_1 (area)	x_2 (beds)	y (price)	$\hat{y}$ (prediction)
1	0.846	1	115.00	161.37
2	1.324	2	234.50	213.61
3	1.150	3	198.00	168.88
4	3.037	4	528.00	430.67
5	3.984	5	572.50	552.66

Linear dependence

- set of n-vectors $\{a_1,\ldots,a_k\}$ (with $k \geq 1$) is *linearly dependent* if

$$\beta_1 a_1 + \cdots + \beta_k a_k = 0$$

  holds for some $\beta_1,\ldots,\beta_k$, that are not all zero
- equivalent to: at least one a_i is a linear combination of the others
- we say '$a_1,\ldots,a_k$ are linearly dependent'

- $\{a_1\}$ is linearly dependent only if $a_1 = 0$
- $\{a_1,a_2\}$ is linearly dependent only if one a_i is a multiple of the other
- for more than two vectors, there is no simple to state condition

Example

- the vectors

$$a_1 = \begin{bmatrix} 0.2 \\ -7 \\ 8.6 \end{bmatrix}, \qquad a_2 = \begin{bmatrix} -0.1 \\ 2 \\ -1 \end{bmatrix}, \qquad a_3 = \begin{bmatrix} 0 \\ -1 \\ 2.2 \end{bmatrix}$$

  are linearly dependent, since $a_1 + 2a_2 - 3a_3 = 0$

- can express any of them as linear combination of the other two, *e.g.*,

$$a_2 = (-1/2)a_1 + (3/2)a_3$$

Linear independence

- set of n-vectors $\{a_1, \ldots, a_k\}$ (with $k \geq 1$) is *linearly independent* if it is not linearly dependent, *i.e.*,

$$\beta_1 a_1 + \cdots + \beta_k a_k = 0$$

  holds only when $\beta_1 = \cdots = \beta_k = 0$

- we say '$a_1, \ldots, a_k$ are linearly independent'
- equivalent to: no a_i is a linear combination of the others
- example: the unit n-vectors $e_1, \ldots, e_n$ are linearly independent

Linear combinations of linearly independent vectors

- suppose x is linear combination of linearly independent vectors $a_1, \ldots, a_k$:

$$x = \beta_1 a_1 + \cdots + \beta_k a_k$$

- the coefficients $\beta_1, \ldots, \beta_k$ are *unique*, *i.e.*, if

$$x = \gamma_1 a_1 + \cdots + \gamma_k a_k$$

  then $\beta_i = \gamma_i$ for $i = 1, \ldots, k$

- this means that (in principle) we can deduce the coefficients from x
- to see why, note that

$$(\beta_1 - \gamma_1) a_1 + \cdots + (\beta_k - \gamma_k) a_k = 0$$

  and so (by linear independence) $\beta_1 - \gamma_1 = \cdots = \beta_k - \gamma_k = 0$

Independence-dimension inequality

- *a linearly independent set of n-vectors can have at most n elements*
- put another way: *any set of $n+1$ or more n-vectors is linearly dependent*

Basis

- a set of n linearly independent n-vectors $a_1, \ldots, a_n$ is called a *basis*
- any n-vector b can be expressed as a linear combination of them:

$$b = \beta_1 a_1 + \cdots + \beta_n a_n$$

  for some $\beta_1, \ldots, \beta_n$

- and these coefficients are unique
- formula above is called *expansion of b in the $a_1, \ldots, a_n$ basis*
- example: $e_1, \ldots, e_n$ is a basis, expansion of b is

$$b = b_1 e_1 + \cdots + b_n e_n$$

Orthonormal vectors

- set of n-vectors $a_1, \ldots, a_k$ are *(mutually) orthogonal* if $a_i \perp a_j$ for $i \neq j$
- they are *normalized* if $\|a_i\| = 1$ for $i = 1, \ldots, k$
- they are *orthonormal* if both hold
- can be expressed using inner products as

$$a_i^T a_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

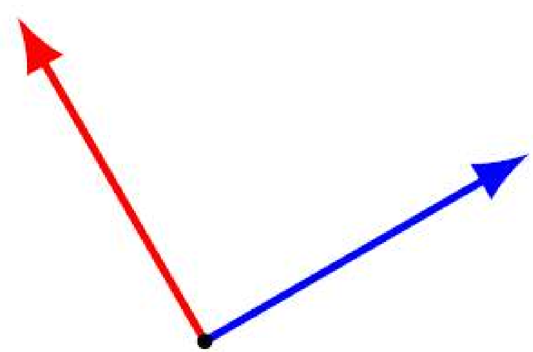
- orthonormal sets of vectors are linearly independent
- by independence-dimension inequality, must have $k \leq n$
- when $k = n$, $a_1, \ldots, a_n$ are an *orthonormal basis*

Examples of orthonormal bases

- standard unit n-vectors $e_1, \ldots, e_n$
- the 3-vectors

$$\begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}, \quad \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \quad \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}$$

- the 2-vectors shown below



Orthonormal expansion

- if $a_1, \ldots, a_n$ is an orthonormal basis, we have for any n-vector x

$$x = (a_1^T x)a_1 + \cdots + (a_n^T x)a_n$$

- called *orthonormal expansion of* x (in the orthonormal basis)
- to verify formula, take inner product of both sides with a_i

Orthogonal sets

Let V be a vector space with an inner product.

Definition. Nonzero vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k \in V$ form an **orthogonal set** if they are orthogonal to each other: $\langle \mathbf{v}_i, \mathbf{v}_j \rangle = 0$ for $i \neq j$.

If, in addition, all vectors are of unit norm, $\|\mathbf{v}_i\| = 1$, then $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ is called an **orthonormal set**.

Theorem Any orthogonal set is linearly independent.

Orthogonal projection

Let V be an inner product space.

Let $\mathbf{x}, \mathbf{v} \in V$, $\mathbf{v} \neq \mathbf{0}$. Then $\mathbf{p} = \frac{\langle \mathbf{x}, \mathbf{v} \rangle}{\langle \mathbf{v}, \mathbf{v} \rangle} \mathbf{v}$ is the **orthogonal projection** of the vector $\mathbf{x}$ onto the vector $\mathbf{v}$. That is, the remainder $\mathbf{o} = \mathbf{x} - \mathbf{p}$ is orthogonal to $\mathbf{v}$.

If $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ is an orthogonal set of vectors then

$$\mathbf{p} = \frac{\langle \mathbf{x}, \mathbf{v}_1 \rangle}{\langle \mathbf{v}_1, \mathbf{v}_1 \rangle} \mathbf{v}_1 + \frac{\langle \mathbf{x}, \mathbf{v}_2 \rangle}{\langle \mathbf{v}_2, \mathbf{v}_2 \rangle} \mathbf{v}_2 + \cdots + \frac{\langle \mathbf{x}, \mathbf{v}_n \rangle}{\langle \mathbf{v}_n, \mathbf{v}_n \rangle} \mathbf{v}_n$$

is the **orthogonal projection** of the vector $\mathbf{x}$ onto the subspace spanned by $\mathbf{v}_1, \ldots, \mathbf{v}_n$. That is, the remainder $\mathbf{o} = \mathbf{x} - \mathbf{p}$ is orthogonal to $\mathbf{v}_1, \ldots, \mathbf{v}_n$.

Gram–Schmidt (orthogonalization) algorithm

Let V be a vector space with an inner product.
Suppose $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ is a basis for V. Let

$$\mathbf{v}_1 = \mathbf{x}_1,$$

$$\mathbf{v}_2 = \mathbf{x}_2 - \frac{\langle \mathbf{x}_2, \mathbf{v}_1 \rangle}{\langle \mathbf{v}_1, \mathbf{v}_1 \rangle}\mathbf{v}_1,$$

$$\mathbf{v}_3 = \mathbf{x}_3 - \frac{\langle \mathbf{x}_3, \mathbf{v}_1 \rangle}{\langle \mathbf{v}_1, \mathbf{v}_1 \rangle}\mathbf{v}_1 - \frac{\langle \mathbf{x}_3, \mathbf{v}_2 \rangle}{\langle \mathbf{v}_2, \mathbf{v}_2 \rangle}\mathbf{v}_2,$$

. . . .

$$\mathbf{v}_n = \mathbf{x}_n - \frac{\langle \mathbf{x}_n, \mathbf{v}_1 \rangle}{\langle \mathbf{v}_1, \mathbf{v}_1 \rangle}\mathbf{v}_1 - \cdots - \frac{\langle \mathbf{x}_n, \mathbf{v}_{n-1} \rangle}{\langle \mathbf{v}_{n-1}, \mathbf{v}_{n-1} \rangle}\mathbf{v}_{n-1}.$$

Then $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ is an orthogonal basis for V.

Gram–Schmidt (orthogonalization) algorithm

Any basis		Orthogonal basis
$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$	$\longrightarrow$	$\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$

Properties of the Gram-Schmidt process:

- $\mathbf{v}_k = \mathbf{x}_k - (\alpha_1 \mathbf{x}_1 + \cdots + \alpha_{k-1} \mathbf{x}_{k-1})$, $1 \leq k \leq n$;
- the span of $\mathbf{v}_1, \ldots, \mathbf{v}_k$ is the same as the span of $\mathbf{x}_1, \ldots, \mathbf{x}_k$;
- $\mathbf{v}_k$ is orthogonal to $\mathbf{x}_1, \ldots, \mathbf{x}_{k-1}$;
- $\mathbf{v}_k = \mathbf{x}_k - \mathbf{p}_k$, where $\mathbf{p}_k$ is the orthogonal projection of the vector $\mathbf{x}_k$ on the subspace spanned by $\mathbf{x}_1, \ldots, \mathbf{x}_{k-1}$;
- $\|\mathbf{v}_k\|$ is the distance from $\mathbf{x}_k$ to the subspace spanned by $\mathbf{x}_1, \ldots, \mathbf{x}_{k-1}$.

Example

Using the Gram-Schmidt process, we orthogonalize the basis $\mathbf{x}_1 = (1, 2, 2)$, $\mathbf{x}_2 = (-1, 0, 2)$, $\mathbf{x}_3 = (0, 0, 1)$:

$$\mathbf{v}_1 = \mathbf{x}_1 = (1, 2, 2),$$

$$\mathbf{v}_2 = \mathbf{x}_2 - \frac{\langle \mathbf{x}_2, \mathbf{v}_1 \rangle}{\langle \mathbf{v}_1, \mathbf{v}_1 \rangle}\mathbf{v}_1 = (-1, 0, 2) - \frac{3}{9}(1, 2, 2)$$

$$= (-4/3, -2/3, 4/3),$$

$$\mathbf{v}_3 = \mathbf{x}_3 - \frac{\langle \mathbf{x}_3, \mathbf{v}_1 \rangle}{\langle \mathbf{v}_1, \mathbf{v}_1 \rangle}\mathbf{v}_1 - \frac{\langle \mathbf{x}_3, \mathbf{v}_2 \rangle}{\langle \mathbf{v}_2, \mathbf{v}_2 \rangle}\mathbf{v}_2$$

$$= (0, 0, 1) - \frac{2}{9}(1, 2, 2) - \frac{4/3}{4}(-4/3, -2/3, 4/3)$$

$$= (2/9, -2/9, 1/9).$$

Example

Now $\mathbf{v}_1 = (1, 2, 2)$, $\mathbf{v}_2 = (-4/3, -2/3, 4/3)$, $\mathbf{v}_3 = (2/9, -2/9, 1/9)$ is an orthogonal basis for $\mathbb{R}^3$

$$\langle \mathbf{v}_1, \mathbf{v}_1 \rangle = 9 \implies \|\mathbf{v}_1\| = 3$$
$$\langle \mathbf{v}_2, \mathbf{v}_2 \rangle = 4 \implies \|\mathbf{v}_2\| = 2$$
$$\langle \mathbf{v}_3, \mathbf{v}_3 \rangle = 1/9 \implies \|\mathbf{v}_3\| = 1/3$$

$$\mathbf{w}_1 = \mathbf{v}_1/\|\mathbf{v}_1\| = (1/3, 2/3, 2/3) = \tfrac{1}{3}(1, 2, 2),$$
$$\mathbf{w}_2 = \mathbf{v}_2/\|\mathbf{v}_2\| = (-2/3, -1/3, 2/3) = \tfrac{1}{3}(-2, -1, 2),$$
$$\mathbf{w}_3 = \mathbf{v}_3/\|\mathbf{v}_3\| = (2/3, -2/3, 1/3) = \tfrac{1}{3}(2, -2, 1).$$

Matrix-vector product function

- *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,

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

- matrix-vector multiplication costs $m(2n-1) \approx 2mn$ flops
(for sparse A, around $2\mathbf{nnz}(A)$ flops)

Examples

- 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 $x_1, \ldots, x_i$

Hadamard Product

- For two matrices, $\mathbf{A}$, $\mathbf{B}$, of the same dimension, $m \times n$ the Hadamard product, $\mathbf{A} \circ \mathbf{B}$, is a matrix, of the same dimension as the operands, with elements given by

$$(\mathbf{A} \circ \mathbf{B})_{i,j} = (\mathbf{A})_{i,j} \cdot (\mathbf{B})_{i,j}$$

  - For example the Hadamard product for a 3×3 matrix $\mathbf{A}$ with a 3×3 matrix $\mathbf{B}$ is:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \circ \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix} = \begin{bmatrix} A_{11}B_{11} & A_{12}B_{12} & A_{13}B_{13} \\ A_{21}B_{21} & A_{22}B_{22} & A_{23}B_{23} \\ A_{31}B_{31} & A_{32}B_{32} & A_{33}B_{33} \end{bmatrix}$$

Kronecker Product

- If $\mathbf{A}$ is an $m \times n$ matrix and $\mathbf{B}$ is a $p \times q$ matrix, then the Kronecker product $\mathbf{A} \otimes \mathbf{B}$ is the $mp \times nq$ block matrix:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} A_{11}\mathbf{B} & \cdots & A_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ A_{m1}\mathbf{B} & \cdots & A_{mn}\mathbf{B} \end{bmatrix}$$

  - For example, the Kronecker product for a 2×2 matrix $\mathbf{A}$ with a 2×3 matrix $\mathbf{B}$ is:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} A_{11}B_{11} & A_{11}B_{12} & A_{11}B_{13} & A_{12}B_{11} & A_{12}B_{12} & A_{12}B_{13} \\ A_{11}B_{21} & A_{11}B_{22} & A_{11}B_{23} & A_{12}B_{21} & A_{12}B_{22} & A_{12}B_{23} \\ A_{21}B_{11} & A_{21}B_{12} & A_{21}B_{13} & A_{22}B_{11} & A_{22}B_{12} & A_{22}B_{13} \\ A_{21}B_{21} & A_{21}B_{22} & A_{21}B_{23} & A_{22}B_{21} & A_{22}B_{22} & A_{22}B_{23} \end{bmatrix}$$

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

$$\begin{aligned} A &= \begin{bmatrix} f(e_1) - f(0) & f(e_2) - f(0) & \cdots & f(e_n) - f(0) \end{bmatrix} \\ b &= f(0) \end{aligned}$$

- affine functions sometimes (incorrectly) called linear

Systems of linear equations

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

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

- 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

Left inverse

- 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 linaerly 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

Right inverse

- 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

Generalized 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

Pseudo-inverse of a tall matrix

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

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

- it is a left inverse of A:

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

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

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

Pseudo-inverse of a 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:

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

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

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

Least squares problem

- suppose $m \times n$ matrix A is tall, so $Ax = b$ is over-determined
- for most choices of b, there is no x that satisfies $Ax = b$
- *residual* is $r = Ax - b$
- *least squares problem*: choose x to minimize $\|Ax - b\|^2$
- $\|Ax - b\|^2$ is the *objective function*
- $\hat{x}$ is a *solution* of least squares problem if

$$\|A\hat{x} - b\|^2 \leq \|Ax - b\|^2$$

  for any n-vector x
- idea: $\hat{x}$ makes residual as small as possible, if not 0
- also called *regression* (in data fitting context)

Least squares problem

- $\hat{x}$ called *least squares approximate solution* of $Ax = b$
- $\hat{x}$ is sometimes called 'solution of $Ax = b$ in the least squares sense'
  - this is very confusing
  - never say this
  - do not associate with people who say this

- $\hat{x}$ need not (and usually does not) satisfy $A\hat{x} = b$
- but if $\hat{x}$ does satisfy $A\hat{x} = b$, then it solves least squares problem

Least squares problem – column interpretation

- suppose $a_1, \ldots, a_n$ are columns of A
- then

$$\|Ax - b\|^2 = \|(x_1 a_1 + \cdots + x_n a_n) - b\|^2$$

- so least squares problem is to find a linear combination of columns of A that is closest to b
- if $\hat{x}$ is a solution of least squares problem, the m-vector

$$A\hat{x} = \hat{x}_1 a_1 + \cdots + \hat{x}_n a_n$$

is closest to b among all linear combinations of columns of A

Least squares problem – row interpretation

- suppose $\tilde{a}_1^T, \ldots, \tilde{a}_m^T$ are rows of A
- residual components are $r_i = \tilde{a}_i^T x - b_i$
- least squares objective is

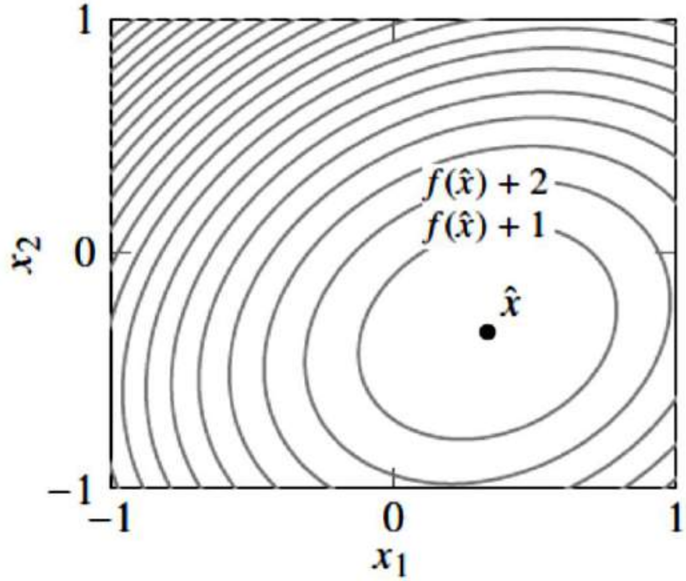
$$\|Ax - b\|^2 = (\tilde{a}_1^T x - b_1)^2 + \cdots + (\tilde{a}_m^T x - b_m)^2$$

  the sum of squares of the residuals

- so least squares minimizes sum of squares of residuals
  - solving $Ax = b$ is making all residuals zero
  - least squares attempts to make them all small

Example

$$A = \begin{bmatrix} 2 & 0 \\ -1 & 1 \\ 0 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$



- $Ax = b$ has no solution
- least squares problem is to choose x to minimize

$$\|Ax - b\|^2 = (2x_1 - 1)^2 + (-x_1 + x_2)^2 + (2x_2 + 1)^2$$

- least squares approximate solution is $\hat{x} = (1/3, 1/3)$ (say, via calculus)
- $\|A\hat{x} - b\|^2 = 2/3$ is smallest posible value of $\|Ax - b\|^2$
- $A\hat{x} = (2/3, -2/3, -2/3)$ is linear combination of columns of A closest to b

Solution of least squares problem

- we make one assumption: A *has linearly independent columns*
- this implies that Gram matrix $A^T A$ is invertible
- unique solution of least squares problem is

$$\hat{x} = (A^T A)^{-1} A^T b = A^{\dagger} b$$

- cf. $x = A^{-1} b$, solution of square invertible system $Ax = b$

Matrix Calculus – The Gradient

- Let a function $f : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}$ takes as input a matrix A of size m × n and returns a real value.
- Then the **gradient** of **f:**

$$\nabla_A f(A) \in \mathbb{R}^{m \times n} = \begin{bmatrix} \frac{\partial f(A)}{\partial A_{11}} & \frac{\partial f(A)}{\partial A_{12}} & \cdots & \frac{\partial f(A)}{\partial A_{1n}} \\ \frac{\partial f(A)}{\partial A_{21}} & \frac{\partial f(A)}{\partial A_{22}} & \cdots & \frac{\partial f(A)}{\partial A_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f(A)}{\partial A_{m1}} & \frac{\partial f(A)}{\partial A_{m2}} & \cdots & \frac{\partial f(A)}{\partial A_{mn}} \end{bmatrix}$$

Matrix Calculus – The Gradient

- Every entry in the matrix is: $\nabla_A f(A))_{ij} = \frac{\partial f(A)}{\partial A_{ij}}.$

- The size of $\nabla_A f(A)$ is always the same as the size of A.

- So if A is just a vector x: $\nabla_x f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix}$

Exercise

- Example:

For $x \in \mathbb{R}^n$, let $f(x) = b^T x$ for some known vector $b \in \mathbb{R}^n$

$$f(x) = \begin{bmatrix} b_1 & b_2 & \dots & b_n \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

- Find:

$$\frac{\partial f(x)}{\partial x_k} = ?$$

$$\nabla_x f(x) = ?$$

Exercise

- Example:

For $x \in \mathbb{R}^n$, let $f(x) = b^T x$ for some known vector $b \in \mathbb{R}^n$

$$f(x) = \sum_{i=1}^{n} b_i x_i$$

$$\frac{\partial f(x)}{\partial x_k} = \frac{\partial}{\partial x_k} \sum_{i=1}^{n} b_i x_i = b_k.$$

- From this we can conclude that: $\nabla_x b^T x = b.$

Matrix Calculus – The Gradient

- Properties

  - $\nabla_x (f(x) + g(x)) = \nabla_x f(x) + \nabla_x g(x)$.
  - For $t \in \mathbb{R}$, $\nabla_x (t\, f(x)) = t \nabla_x f(x)$.

Matrix Calculus – The Hessian

- The Hessian matrix with respect to x, written $\nabla_x^2 f(x)$ or simply as H is the n × n matrix of partial derivatives

$$\nabla_x^2 f(x) \in \mathbb{R}^{n\times n} = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix}$$

Matrix Calculus – The Hessian

- Each entry can be written as: $\nabla_x^2 f(x))_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}$.

- The Hessian is always symmetric, $\frac{\partial^2 f(x)}{\partial x_i \partial x_j} = \frac{\partial^2 f(x)}{\partial x_j \partial x_i}$.

- This is known as Schwarz's theorem: The order of partial derivatives don't matter as long as the second derivative exists and is continuous.

Matrix Calculus – The Hessian

- Note that the hessian is not the gradient of whole gradient of a vector (this is not defined). It is actually the gradient of **every entry** of the gradient of the vector.

$$\nabla_x^2 f(x) \in \mathbb{R}^{n\times n} = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix}$$

Matrix Calculus – The Hessian

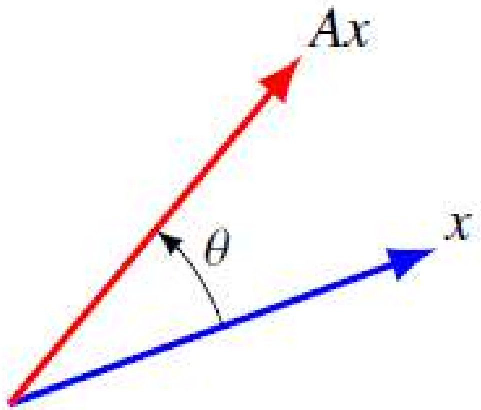
- Eg, the first column is the gradient of $\dfrac{\partial f(x)}{\partial x_1}$

$$\nabla_x^2 f(x) \in \mathbb{R}^{n\times n} = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix}$$

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)

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, ...

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