Math Notes

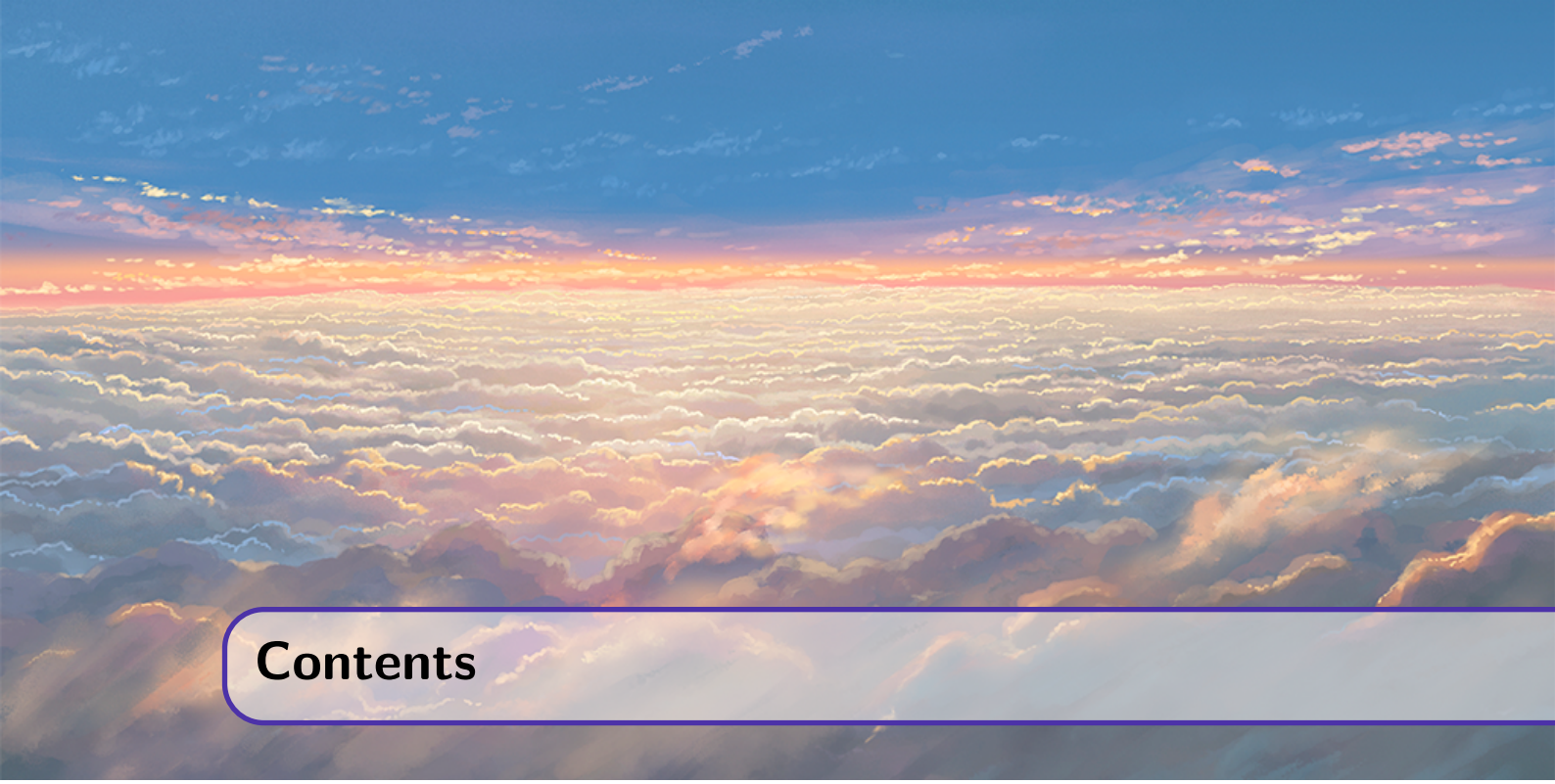
Linear Algebra and Optimisation

Written by Xia Wenxuan, 2021

https://github.com/gegeji

All the material in this document is taken from my textbook or video material, and some material uses optical character recognition (OCR) to aid input. It may contains typographical or content inaccuracies. This document is for my personal study only. I am not responsible for the accuracy of the content of the text.

Edited and Revised on December 29, 2021



Contents

IV Extensive Reading

Vectors

1. 对向量的介绍

1.1 Vector

Definition 1.1.1 — **Vector.** 一个有序的数字列表.

$$\begin{bmatrix} -1.1 \\ 0.0 \\ 3.6 \\ -7.2 \end{bmatrix} \text{或者} \quad \begin{pmatrix} -1.1 \\ 0.0 \\ 3.6 \\ -7.2 \end{pmatrix} \text{或者} \quad (-1.1, 0, 3.6, -7.2)$$

表中的数字是元素 (项、系数、分量). 元素的数量是向量的大小 (维数, 长度). 大小为 n 的向量称为n 维向量. 向量中的数字通常被称作标量.

用符号来表示向量, 比如 a , b, 一般用小写字母表示向量. 其它表示形式如 $\boldsymbol{g}, \vec{a}$ 等。

Definition 1.1.2 — **n 维向量 a 的第 i 元素.** n 维向量 a 的第 i 元素表示为 a_i.

有时 i 指的是向量列表中的第 i 个向量.

Definition 1.1.3 — $a = b$. 对于所有 i, 如果有 $a_i = b_i$, 则称两个相同大小的向量 a 和 b 是相等的, 可写成 $a = b$。

Definition 1.1.4 — **stacked vector.** 假设 b、c、d 是大小为 m、n、p 的向量

$$a = \begin{bmatrix} b \\ c \\ d \end{bmatrix}$$

$$a = (b_1, b_2, \ldots, b_m, c_1, c_2, \ldots, c_n, d_1, d_2, \ldots, d_p)$$

Definition 1.1.5 — **Subvectors.** Colon notation can be used to define subvectors (slices) of a vector.

subvectors. If a is a vector, then $a_{r:s}$ is the vector of size $s - r + 1$, with entries $a_r, \ldots, a_s$:

$$a_{r:s} = (a_r, \ldots, a_s)$$

The subscript $r : s$ is called the index range. Thus, in our example above, we have

$$b = a_{1:m}, \quad c = a_{(m+1):(m+n)}, \quad d = a_{(m+n+1):(m+n+p)}.$$

在本书中，下标从 1 开始。

Definition 1.1.6 — **零向量**. 所有项为 0 的 n 维向量表示为 0_n 或者 0.

Definition 1.1.7 — **全一向量**. 所有项为 1 的 n 维向量表示为 $\mathbf{1}_n$ 或者 1.

Definition 1.1.8 — **单位向量**. 当第 i 项为 1, 其余项为 0 时表示为 e_i

■ **Example 1.1**

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

■

Definition 1.1.9 — **稀疏向量**. 如果一个向量的许多项都是 0, 该向量为稀疏 (sparse) 的. 稀疏向量能在计算机上高效地存储和操作.

$\mathbf{nnz}(x)$ 是指向量 x 中非零的项数 (number of non-zeros), 有时用 ℓ_0 表示.

向量 $x = (x_1, x_2)$ 可以在二维中表示一个位置 (location) 或一个位移 (displacement)、图像 (列向量)、表示一句话中每个单词出现多少次、颜色等.

■ **Example 1.2 — Monochrome (black and white) image.** grayscale values of $M \times N$ pixels stored as MN-vector (e.g., row-wise)

Color image: $3MN$-vectors with R, G, B values of the MN pixels

Video: vector of size KMN represents K monochrome images of $M \times N$ pixels ■

■ **Example 1.3 — 时间序列.** 每个元素是一个样本 (sample). ■

■ **Example 1.4 — 自然文本进行 tokenization.** 对于自然文本进行 tokenization。

将 'rained' 约简为 'rain' 称为 *stemming*。另外会将非常常用的词，如 'a', 'the' 和极少出现的词去掉。这些词称为停用词。 ■

■ **Example 1.5 — Feature vectors.** contain values of variables or attributes that describe members of a set. ■

Vector elements can represent very different quantities, in different units.

It can contain categorical features (e.g., 0/1 for male/female)

Its ordering has no particular meaning.

1.2 Vector Space

Definition 1.2.1 — **向量空间** V. 设 V 是非空子集, P 是一数域, 向量空间 V 满足:

1. 向量加法: $V+V \to V$, 记作 $\forall x, y \in V$, 则 $x+y \in V$ (加法封闭)
2. 标量乘法: $F \times V \to V$, 记作 $\forall x \in V, \lambda \in P$, 则 $\lambda x \in V$ (乘法封闭)

Theorem 1.2.1 上述两个运算满足下列八条规则 $(\forall x, y, z \in V, \lambda, \mu \in P)$

1. $x+y=y+x$ (交换律)
2. $x+(y+z)=(x+y)+z$ (结合律)
3. V 存在一个零元素, 记作 0, $x+0=x$
4. 存在 x 的负元素, 记作 $-x$, 满足 $x+(-x)=0$
5. $\forall x \in V$, 都有 $1x=x, 1 \in P$
6. $\lambda(\mu x)=(\lambda\mu)x$
7. $(\lambda+\mu)x=\lambda x+\mu x$
8. $\lambda(x+y)=\lambda x+\lambda y$

Corollary 1.2.2 向量空间也称为线性空间.

Corollary 1.2.3 如果 $x, y \in \mathbb{R}^2$, 则 $x+y \in \mathbb{R}^2, \lambda x \in \mathbb{R}^2 (\lambda \in \mathbb{R})$.

Definition 1.2.2 — **数域**. 数的非空集合 P, 且其中任意两个数的和、差、积、商 (除数不为零) 仍属于该集合, 则称数集 P 为一个数域.

- **Example 1.6** 有理数 $\mathbb{Q}$ ∎
- **Example 1.7** 实数 $\mathbb{R}$
  $x, y \in \mathbb{R}, x=1, y=2\ x+y \in \mathbb{R}, x \times y \in \mathbb{R}$ ∎
- **Example 1.8** 复数 $\mathbb{C}$ ∎

1.3 向量运算

Definition 1.3.1 — **向量加法**. n 维向量 a 和 b 可以相加, 求和形式表示为 $a+b$.

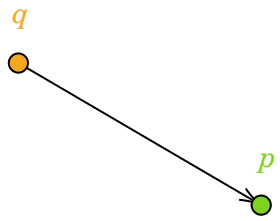
Theorem 1.3.1 设向量 a, b, c 是向量空间 V 的元素, 即 $a, b, c \in V$。向量加法满足以下性质:

1. 交换律: $a+b=b+a$
2. 结合律: $(a+b)+c=a+(b+c)$ (因此可写成 $a+b+c$)
3. $a+0=0+a=a$
4. $a-a=0$

Corollary 1.3.2 — **向量位移相加**. 如果二维向量 a 和 b 都表示位移, 则它们的位移之和为 $a+b$。

▪ **Example 1.9** 点 q 到点 p 的位移是 $p - q$.

Figure 1.1: The translation from q to p



▪

Definition 1.3.2 — **标量与向量的乘法.**

$$\beta a = \begin{bmatrix} \beta a_1 \\ \vdots \\ \beta a_n \end{bmatrix}$$

Theorem 1.3.3 标量 β, γ 与向量 a、b 进行乘法, 有如下性质:

1. 结合律: $(\beta\gamma)a = \beta(\gamma a)$
2. 左分配律: $(\beta + \gamma)a = \beta a + \gamma a$
3. 右分配律: $\beta(a + b) = \beta a + \beta b$

Definition 1.3.3 — **线性组合.** 对于向量 $a_1, \ldots, a_m$ 和标量 $\beta_1, \ldots, \beta_m$,

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

是向量的线性组合. $\beta_1, \ldots, \beta_m$ 是该向量的系数.

Theorem 1.3.4 对于任何向量 $b \in \mathbb{R}^n$, 满足

$$b = b_1 e_1 + \cdots + b_n e_n, b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

1.4 内积

Definition 1.4.1 — **内积.** 在数域 $\mathbb{R}$ 上的向量空间 V, 定义函数 $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$, 满足:

1. $\langle a, a \rangle \geq 0, \forall a \in V$, 当且仅当 $a = 0$ 时 $\langle a, a \rangle = 0$.
2. $\langle \alpha a + \beta b, c \rangle = \alpha \langle a, c \rangle + \beta \langle b, c \rangle, \forall \alpha, \beta \in \mathbb{R}$, 且 $a, b, c \in V$.
3. $\langle a, b \rangle = \langle b, a \rangle, \forall a, b \in V$.

则称函数 $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$ 是内积.

▪ **Example 1.10 — the inner product of two n-vectors** a, b. 在向量空间 $\mathbb{R}^n$ 上, 计算两个向量对应项相乘之后求和函数

$$\langle a, b \rangle = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n = a^T b$$

where $a = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \in \mathbb{R}^n$. ∎

Proof. 对于性质 1

$$\langle a, a \rangle = a^T a = a_1 a_1 + a_2 a_2 + \cdots + a_n a_n = \sum_{i=1}^{n} a_i^2 \geq 0,$$

$\langle a, a \rangle = 0$则$a = 0$.
对于性质 2

$$\begin{aligned} \langle \alpha a + \beta b, c \rangle &= (\alpha a + \beta b)^T c \\ &= (\alpha a_1 + \beta b_1)\, c_1 + (\alpha a_2 + \beta b_2)\, c_2 + \cdots + (\alpha a_n + \beta b_n)\, c_n \\ &= \alpha \sum_{i=1}^{n} a_i c_i + \beta \sum_{i=1}^{n} b_i c_i \\ &= \alpha \langle a, c \rangle + \beta \langle b, c \rangle \end{aligned}$$

对于性质 3

$$\langle a, b \rangle = a^T b = b^T a = \langle b, a \rangle$$

■

Theorem 1.4.1 内积的性质：交换律、结合律、分配律.
交换律 (commutative)：$a^T b = b^T a$
结合律 (associative with scalar multiplication)：$(\gamma a)^T b = \gamma \left(a^T b\right)$
分配律 (distributive with vector addition)：$(a + b)^T c = a^T c + b^T c$

1.4.1 常用的内积等式

Corollary 1.4.2 — 选出第 i 项.

$$e_i^T a = a_i$$

Corollary 1.4.3 — 向量每一项之和.

$$\mathbf{1}^T a = a_1 + \cdots + a_n$$

Corollary 1.4.4 — 向量每一项的平方和.

$$a^T a = a_1^2 + \cdots + a_n^2$$

Corollary 1.4.5 — 向量元素的平均值.

$$(\frac{\mathbf{1}}{n})^T a = \frac{a_1 + \cdots + a_n}{n}$$

Corollary 1.4.6 — Selective sum. Let b be a vector all of whose entries are either 0 or 1. Then

$$b^T a$$

is the sum of the elements in a for which $b_i = 1$.

Corollary 1.4.7 — Differencing.

$$\left(e_i - e_j\right)^T a = a_i - a_j$$

Definition 1.4.2 — The sum of block vectors. If the vectors a and b are block vectors, and the corresponding blocks have the same sizes (in which case we say they *conform*), then

$$a^T b = \begin{bmatrix} a_1 \\ \vdots \\ a_k \end{bmatrix}^T \begin{bmatrix} b_1 \\ \vdots \\ b_k \end{bmatrix} = a_1^T b_1 + \cdots + a_k^T b_k$$

1.5 Examples for Inner Product

内积用途很广.

■ **Example 1.11 — 计算同时出现的项目数.**

$$a = (0,1,1,1,1,1,1), \quad b = (1,0,1,0,1,0,0)$$

Here we have $a^T b = 2$, which is the number of objects in both A and B (i.e., objects 3 and 5). ■

■ **Example 1.12 — Weights, features, and score.** When the vector f represents a set of *features* of an object, and w is a vector of the same size (often called a *weight vector*), the inner product $w^T f$ is the sum of the feature values, scaled (or weighted) by the weights, and is sometimes called a *score*.

Inner product $w^T f = w_1 f_1 + w_2 f_2 + \cdots + w_n f_n$ is total score.

Some linear combinations of the vectors $a_1, \ldots, a_m$ have special names. For example, the linear combination with $\beta_1 = \cdots = \beta_m = 1$, given by $a_1 + \cdots + a_m$, is the *sum* of the vectors, and the linear combination with $\beta_1 = \cdots = \beta_m = 1/m$, given by $(1/m)(a_1 + \cdots + a_m)$, is the *average* of the vectors. When the coefficients sum to one, i.e., $\beta_1 + \cdots + \beta_m = 1$, the linear combination is called an *affine combination*. When the coefficients in an affine combination are nonnegative, it is called a *convex combination*, a *mixture*, or a *weighted average*. The coefficients in an affine or convex combination are sometimes given as *percentages*, which add up to 100%. ■

■ **Example 1.13 — 多项式.**

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

Let t be a number, $z = \left(1, t, t^2, \ldots, t^{n-1}\right)$ be the n-vector of powers of t. Then

$$c^T z = p(t)$$

■

1.6 Cauchy-Schwartz Inequality

Theorem 1.6.1 — Cauchy-Schwartz Inequality. 设 $\langle\cdot,\cdot\rangle$ 是向量空间 V 上的内积, $\forall x, y \in V$, 则有

$$|\langle x, y\rangle|^2 \leq \langle x, x\rangle\langle y, y\rangle$$

当 $x = -\lambda y$ 时，有 $|\langle x, y\rangle|^2 = \langle x, x\rangle\langle y, y\rangle$。

Proof. 令 $\lambda \in \mathbb{R}$, 则有

$$0 \leq \langle x+\lambda y, x+\lambda y\rangle = \langle x, x\rangle + \lambda\langle y, x\rangle + \lambda\langle x, y\rangle + \lambda^2\langle y, y\rangle = \langle x, x\rangle + 2\lambda\langle y, x\rangle + \lambda^2\langle y, y\rangle$$

则

$$\lambda^2\langle y, y\rangle + 2\lambda\langle y, x\rangle + \langle x, x\rangle \geq 0, \forall\lambda \in \mathbb{R}$$

所以

$$\nabla = (2\langle y, x\rangle)^2 - 4\langle y, y\rangle\langle x, x\rangle \leq 0$$
$$|\langle x, y\rangle|^2 \leq \langle x, x\rangle\langle y, y\rangle$$

当 $|\langle x, y\rangle|^2 = \langle x, x\rangle\langle y, y\rangle$ 时, 有

$$\langle x, x\rangle^2 + 2\lambda\langle y, x\rangle + \lambda^2\langle y, y\rangle = 0$$

也即

$$\langle x+\lambda y, x+\lambda y\rangle = 0$$

因此 $x + \lambda y = 0$, 即 $x = -\lambda y$。 ■

Theorem 1.6.2 — Cauchy-Schwarz 不等式的矩阵元素形式.

$$\left(\sum_{i=1}^{n} u_i v_i\right)^2 \leq \left(\sum_{i=1}^{n} u_i^2\right)\left(\sum_{i=1}^{n} v_i^2\right)$$

Proof. The Cauchy-Schwarz inequality can be proved using only ideas from elementary algebra in this case. Consider the following quadratic polynomial in x

$$0 \leq (u_1 x + v_1)^2 + \cdots + (u_n x + v_n)^2 = \left(\sum_i u_i^2\right)x^2 + 2\left(\sum_i u_i v_i\right)x + \sum_i v_i^2$$

Since it is nonnegative, it has at most one real root for x. Hence its discriminant is less than or equal to zero. That is,

$$\Delta = \left(\sum_i u_i v_i\right)^2 - \left(\sum_i u_i^2\right)\left(\sum_i v_i^2\right) \leq 0$$

which yields the Cauchy-Schwarz inequality. ■

Corollary 1.6.3 — Cauchy-Schwarz 不等式的等价形式. 由 Cauchy-Schwarz 不等式

$$|\langle x, y\rangle|^2 \le \langle x, x\rangle\langle y, y\rangle$$

可以推得

$$|\langle a, b\rangle| \le \|a\|_2\|b\|_2$$
$$\langle a, b\rangle \ge -\|a\|_2\|b\|_2$$

1.7 浮点运算

计算机以浮点格式存储 (实) 数值. 存储 n 维的向量需要 $8n$ 字节进行存储。

基本的算术运算 (加法, 乘法等) 被称为浮点运算 (flop).

Definition 1.7.1 — Floating point operation (flop). the unit of complexity when comparing vector and matrix algorithms.

1 flop = one basic arithmetic operation $(+, -, *, /, \sqrt{}, \ldots)$ in $\mathbf{R}$ or $\mathbf{C}$.

Comments: this is a very simplified model of complexity of algorithms

- we don't distinguish between the different types of arithmetic operations
- we don't distinguish between real and complex arithmetic
- we ignore integer operations (indexing, loop counters, ...)
- we ignore cost of memory access

算法或操作的时间复杂度：作为输入维数的函数所需要的浮点运算总数. 算法复杂度通常以非常粗略地近似估算. （程序）执行时间的粗略估计：计算机速度/flops，目前的计算机大约是 1Gflops/秒 (10^9flops/秒)。

Definition 1.7.2 — Operation count (flop count). Total number of operations in an algorithm. In linear algebra, it is typically a polynomial of the dimensions in the problem

Theorem 1.7.1 — 通过浮点运算次数大致预测程序的运行时间. a crude predictor of run time of the algorithm.

$$\text{run time} \approx \frac{\text{number of operations (flops)}}{\text{computer speed (flops per second)}}$$

Definition 1.7.3 — Dominant term, Order. Dominant term: the highest-order term in the flop count

$$\frac{1}{3}n^3 + 100n^2 + 10n + 5 \approx \frac{1}{3}n^3$$

Order: the power in the dominant term

$$\frac{1}{3}n^3 + 10n^2 + 100 = \text{ order } n^3$$

Corollary 1.7.2 假设有 n 维向量 x 和 y
- $x+y$ 需要 n 次加法, 所以时间复杂度为 (n)flops.
- $x^T y$ 需要 n 次乘法和 $n-1$ 次加法, 所以时间复杂度为 $(2n-1)$flops.
- 对于 $x^T y$, 通常将其时间复杂度简化为 $2n$, 甚至为 n.
- 当 x 或 y 是稀疏的时候, 算法的实际运算时间会比理论时间更少.

1.8 Complex numbers and Vectors

Definition 1.8.1 — Complex Numbers. $x = \alpha + \mathrm{j}\beta$ with α, β real scalars
$\mathrm{j} = \sqrt{-1}$ (more common notation is i or j), α is the *real* part of x, denoted $\operatorname{Re} x$, β is the *imaginary* part, denoted $\operatorname{Im} x$

Definition 1.8.2 — set of complex numbers. set of complex numbers is denoted $\mathbf{C}$

Definition 1.8.3 — Modulus. modulus (absolute value, magnitude):

$$|x| = \sqrt{(\operatorname{Re} x)^2 + (\operatorname{Im} x)^2}$$

Definition 1.8.4 — conjugate.

$$\bar{x} = \operatorname{Re} x - \mathrm{j} \operatorname{Im} x$$

Theorem 1.8.1

$$\operatorname{Re} x = \frac{x + \bar{x}}{2}$$

Theorem 1.8.2

$$\operatorname{Im} x = \frac{x - \bar{x}}{2\mathrm{j}}$$

Theorem 1.8.3

$$|x|^2 = \bar{x}x$$

1.8.1 Polar representation

Theorem 1.8.4 nonzero complex number $x = \operatorname{Re} x + \mathrm{j} \operatorname{Im} x$ can be written as

$$x = |x|(\cos\theta + \mathrm{j}\sin\theta) = |x|e^{\mathrm{j}\theta}$$

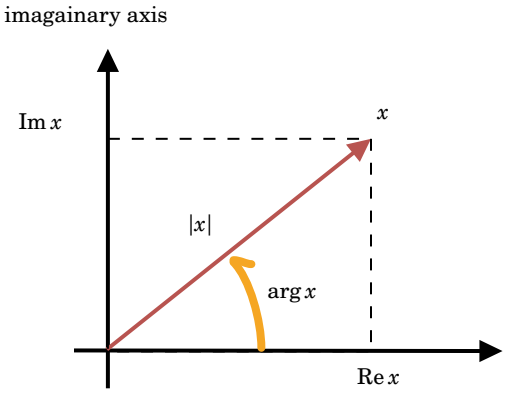
$\theta \in [0, 2\pi)$ is the *argument* (*phase angle*) of x (notation: $\arg x$). $e^{\mathrm{j}\theta}$ is complex exponential.

Theorem 1.8.5 $e^{\mathrm{j}\theta} = \cos\theta + \mathrm{j}\sin\theta$

1.8.2 Complex vector

Definition 1.8.5 — vector with complex elements. $a = \alpha + \mathrm{j}\beta$ with α, β real vectors

Figure 1.2: Complex Numbers



Definition 1.8.6 — **real and imaginary part, conjugate of complex numbers.** real and imaginary part, conjugate are defined componentwise:

$$\begin{aligned} \operatorname{Re} a &= (\operatorname{Re} a_1, \operatorname{Re} a_2, \ldots, \operatorname{Re} a_n) \\ \operatorname{Im} a &= (\operatorname{Im} a_1, \operatorname{Im} a_2, \ldots, \operatorname{Im} a_n) \\ \bar{a} &= \operatorname{Re} a - \mathrm{j} \operatorname{Im} a \end{aligned}$$

Definition 1.8.7 — **set of complex n-vectors.**

$$\mathbf{C}^n$$

Definition 1.8.8

$$a + b = \begin{bmatrix} a_1 + b_1 \\ a_2 + b_2 \\ \vdots \\ a_n + b_n \end{bmatrix}, \quad \gamma a = \begin{bmatrix} \gamma a_1 \\ \gamma a_2 \\ \vdots \\ \gamma a_n \end{bmatrix}, \quad a \circ b = \begin{bmatrix} a_1 b_1 \\ a_2 b_2 \\ \vdots \\ a_n b_n \end{bmatrix}$$

Definition 1.8.9 — **Complex inner product.** the inner product of complex n-vectors a, b is defined as

$$b^H a = \bar{b}_1 a_1 + \bar{b}_2 a_2 + \cdots + \bar{b}_n a_n$$

other notation: $\langle a, b \rangle$, $(a \mid b)$, ...
for real vectors, reduces to real inner product $b^T a$.
for complex n-vectors a, b, c and complex scalars γ

Theorem 1.8.6 — $a^H a$ **的性质**. $a^H a$ 有如下性质:
$a^H a \geq 0$: follows from

$$\begin{aligned} a^H a &= \bar{a}_1 a_1 + \bar{a}_2 a_2 + \cdots + \bar{a}_n a_n \\ &= |a_1|^2 + |a_2|^2 + \cdots + |a_n|^2 \end{aligned}$$

$$a^H a = 0 \text{ only if } a = 0$$

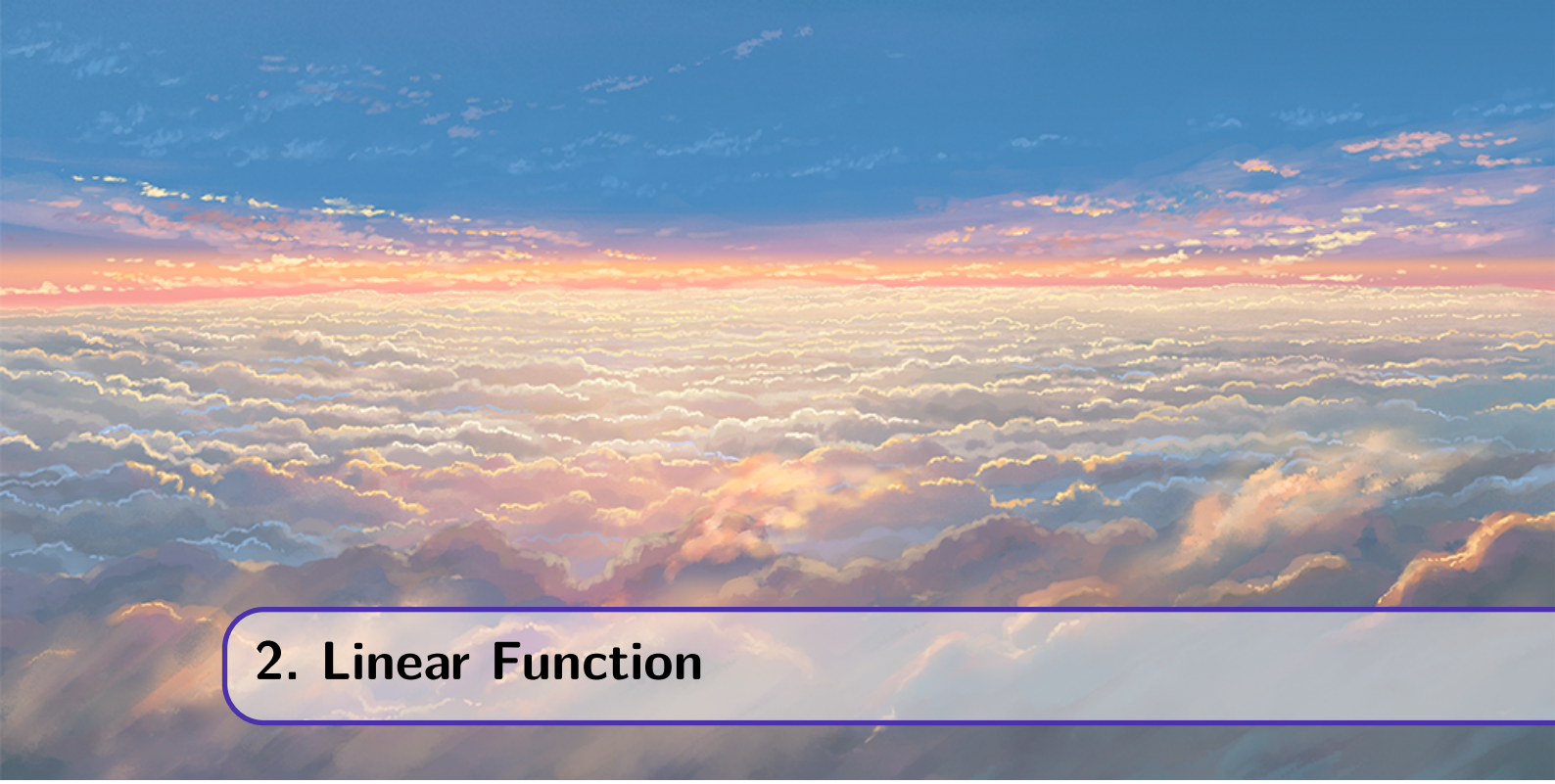
$$b^H a = \overline{a^H b}$$

$$(\gamma b)^H a = \bar{\gamma}\left(b^H a\right)$$

$$(b+c)^H a = b^H a + c^H a$$

$$b^H (a+c) = b^H a + b^H c$$

$$b^H (\gamma a) = \gamma\left(b^H a\right)$$



2. Linear Function

2.1 Linear Function

Definition 2.1.1 — Linear Function. f 是一个将 n 维向量映射成数的函数.

$$f : \mathbb{R}^n \to \mathbb{R}$$

线性函数 f 满足以下两个性质 ($k \in \mathbb{R}, x, y \in \mathbb{R}^n$) :

- 齐次性 (homogeneity): $f(kx) = kf(x)$
- 叠加性 (Additivity): $f(x+y) = f(x) + f(y)$

▪ **Example 2.1** 求平均值: $f(x) = \frac{1}{n}\sum_{i=1}^{n} x_i$ 为线性函数. ▪

▪ **Example 2.2** 求最大值: $f(x) = \max\{x_1, x_2, \ldots, x_n\}$ 并不是线性函数. ▪

Proof. 令 $x = (1, -1), y = (-1, 1), \alpha = 0.5, \beta = 0.5$, 有

$$f(\alpha x + \beta y) = 0 \neq \alpha f(x) + \beta f(y) = 1$$

但是

$$\begin{aligned} f(x+y) &= \max\{x_1 + y_1, x_2 + y_2, \ldots, x_n + y_n\} \\ &\leq \max\{x_1, x_2, \ldots, x_n\} + \max\{y_1, y_2, \ldots, y_n\} \\ &\leq f(x) + f(y) \end{aligned}$$

不满足线性函数的定义 ■

Theorem 2.1.1 a function $f : \mathbf{R}^n \to \mathbf{R}$ is linear if the superposition property (叠加原理)

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

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

Corollary 2.1.2 设 $\alpha_{1,\ldots,}\alpha_m \in \mathbb{R}, u_1, \ldots, u_m \in \mathbb{R}^n$, 则线性函数 f 满足

$$\begin{aligned} f\left(\alpha_1 u_1 + \alpha_2 u_2 + \ldots + \alpha_m u_m\right) &= f\left(\alpha_1 u_1\right) + f\left(\alpha_2 u_2 + \ldots + \alpha_m u_m\right) \\ &= \alpha_1 f\left(u_1\right) + f\left(\alpha_2 u_2 + \ldots + \alpha_m u_m\right) \\ &= \alpha_1 f\left(u_1\right) + \alpha_2 f\left(u_2\right) + \ldots + \alpha_m f\left(u_m\right) \end{aligned}$$

Definition 2.1.2 — 内积函数 (inner product function). 对于 n 维向量 a, 满足以下形式的函数 $f : \mathbf{R}^n \to \mathbf{R}$ 被称为内积函数

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

上述 $f(x)$ 可以看作是每项 x_i 的加权之和.

- **Example 2.3** $f(x) = \frac{1}{3}(x_1 + x_2 + x_3)$ is linear: $f(x) = a^T x$ with $a = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$ ∎
- **Example 2.4** $f(x) = -x_1$ is linear: $f(x) = a^T x$ with $a = (-1, 0, 0)$ ∎
- **Example 2.5** $f(x) = \max\{x_1, x_2, x_3\}$ is not linear: superposition does not hold for

$$x = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad y = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \alpha = -1, \quad \beta = 1$$

we have $f(x) = 1, f(y) = 0$,

$$f(\alpha x + \beta y) = 0 \neq \alpha f(x) + \beta f(y) = -1$$

∎

Corollary 2.1.3 所有的内积函数都是线性的.

$$a^T(\alpha x + \beta y) = \alpha\left(a^T x\right) + \beta\left(a^T y\right)$$

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

Proof.

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

■

Corollary 2.1.4 所有的线性函数都是内积函数.

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

Proof. 假设 $f : \mathbb{R}^n \to \mathbb{R}$ 是线性函数, 那么可用 $f(x) = a^T x$ 来表示, a 为常量.

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

■

2.2 Affine Function

Definition 2.2.1 — **仿射函数 (affine function).** 其一般形式为 $f(x) = a^T x + \mathrm{b}$, 其中 $a \in \mathbb{R}^n$, $b \in \mathbb{R}$ 为标量.

Theorem 2.2.1 函数 $f : \mathbb{R}^n \to \mathbb{R}$ 为仿射函数需要满足
$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y), \alpha + \beta = 1, \alpha, \beta \in \mathbb{R}, x, y \in \mathbb{R}^n$

Corollary 2.2.2 If f is affine, then

$$f\left(\alpha_1 u_1 + \alpha_2 u_2 + \cdots + \alpha_m u_m\right) = \alpha_1 f\left(u_1\right) + \alpha_2 f\left(u_2\right) + \cdots + \alpha_m f\left(u_m\right)$$

for all n-vectors $u_1, \ldots, u_m$ and all scalars $\alpha_1, \ldots, \alpha_m$ with

$$\alpha_1 + \alpha_2 + \cdots + \alpha_m = 1$$

Definition 2.2.2 for fixed $a \in \mathbf{R}^n, b \in \mathbf{R}$, define a function $f : \mathbf{R}^n \to \mathbf{R}$ by

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

i.e., an inner-product function plus a constant (offset)

Theorem 2.2.3 any function of this type is affine
if $\alpha + \beta = 1$ then

$$a^T(\alpha x + \beta y) + b = \alpha\left(a^T x + b\right) + \beta\left(a^T y + b\right)$$

Theorem 2.2.4 every affine function can be written as $f(x) = a^T x + b$ with:

$$\begin{aligned} a &= \left(f\left(e_1\right) - f(0), f\left(e_2\right) - f(0), \ldots, f\left(e_n\right) - f(0)\right) \\ b &= f(0) \end{aligned}$$

2.3 泰勒展开

Definition 2.3.1 — **函数 f 第 i 个分量的一阶偏导数.** 假设 $f : \mathbb{R}^n \to \mathbb{R}$, 函数 f 在 z 点可微

$$\begin{aligned} \frac{\partial f}{\partial z_i}(z) &= \lim_{t \to 0} \frac{f\left(z_1, \cdots, z_{i-1}, z_i + t, z_{i+1}, \cdots, z_n\right) - f(z)}{t} \\ &= \lim_{t \to 0} \frac{f\left(z + t e_i\right) - f(z)}{t} \end{aligned}$$

Definition 2.3.2 — **f 在点 z 的梯度.**

$$\nabla f(z) = \begin{bmatrix} \frac{\partial f}{\partial z_1}(z) \\ \vdots \\ \frac{\partial f}{\partial z_n}(z) \end{bmatrix}$$

Definition 2.3.3 — **Taylor's Approximation.** 假设 $f : \mathbb{R}^n \to \mathbb{R}$, 函数 f 在 z 点充分光滑, 即处处可导.

$f(x)$ 在 z 附近的泰勒展开是

$$f(x) = f(z) + \frac{\partial f}{\partial x_1}(z)\,(x_1 - z_1) + \frac{\partial f}{\partial x_2}(z)\,(x_2 - z_2) + \cdots + \frac{\partial f}{\partial x_n}(z)\,(x_n - z_n) + \frac{1}{2!}\sum_{j=1}^{n}\sum_{i=1}^{n}\frac{\partial f^2}{\partial x_i \partial x_j}(z)\,(x_i - z_i)\left(x_j - z_j\right) + \cdots$$

Definition 2.3.4 — **一阶泰勒公式.** 假设 $f : \mathbb{R}^n \to \mathbb{R}$, 函数 f 在 z 点可导

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

当 x 非常接近 z 时, $\hat{f}(x)$ 也非常接近 $f(z)$. $\hat{f}(x)$ 是关于 x 的一个仿射函数.

Corollary 2.3.1 — **一阶泰勒公式的内积形式.**

$$\hat{f}(x) = f(z) + \nabla f(z)^T (x - z) \quad \left(\nabla f(z) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(z) \\ \vdots \\ \frac{\partial f}{\partial x_n}(z) \end{bmatrix}\right)$$

the n-vector $\nabla f(z)$ is called the gradient of f at z.

一维时, $\hat{f}(x) = f(z) + f'(z)(x - z)$.

■ **Example 2.6**

$$f(x) = x_1 - 3x_2 + e^{2x_1 + x_2 - 1}$$

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(x) \\ \frac{\partial f}{\partial x_2}(x) \end{bmatrix} = \begin{bmatrix} 1 + 2e^{2x_1 + x_2 - 1} \\ -3 + e^{2x_1 + x_2 - 1} \end{bmatrix}$$

函数 f 在 o 点的一阶泰勒公式为:

$$\hat{f}(x) = f(0) + \nabla f(0)^T (x - 0) = e^{-1} + \left(1 + 2e^{-1}\right) x_1 + \left(-3 + e^{-1}\right) x_2$$

■

2.4 高阶泰勒公式

泰勒公式利用多项式在一点附近逼近函数.

■ **Example 2.7**

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots + (-1)^{k-1}\frac{x^{2k-1}}{(2k-1)!} + \frac{\sin\left[\xi + (2k+1)\frac{\pi}{2}\right]}{(2k+1)!}x^{2k+1}$$

一次逼近: $\sin x \approx x$
三次逼近: $\sin x \approx x - \frac{x^3}{3!}$ ■

Proof.

$$f(x) = P_n(x) + R_n(x)$$

$P_n(x) = a_0 + a_1(x-x_0) + a_2(x-x_0)^2 + \cdots + a_n(x-x_0)^n$
$R_n(x) = o(x-x_0)^n$
$f(x) \approx P_n(x)$
$\therefore P_n(x_0) = f(x_0), P_n'(x_0) = f'(x_0), P_n''(x_0) = f''(x_0), \cdots, P_n^{(n)}(x_0) = f^{(n)}(x_0)$
要求 $P_n(x_0) = f(x_0) \Rightarrow a_0 = f(x_0)$

$$P_n'(x) = a_1 + 2a_2(x-x_0) + \cdots + na_n(x-x_0)^{n-1} \Rightarrow a_1 = f'(x_0)$$

依此类推. $a_n = \frac{f^{(n)}(x_0)}{n!}$ ■

Corollary 2.4.1 — n 阶泰勒多项式.

$$P_n(x) = f(x_0) + f'(x_0)(x-x_0) + \frac{f''(x_0)}{2!}(x-x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x-x_0)^n$$

where $a_n = \frac{f^{(n)}(x_0)}{n!}$

Corollary 2.4.2 — 对于高阶余项的公式. 带拉格朗日余项的泰勒公式

$$f(x) = f(x_0) + f'(x_0)(x-x_0) + \frac{f''(x_0)}{2!}(x-x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x-x_0)^n + \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)^{n+1}$$

$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)^{n+1}$ (ξ 在 x_0 与 x 之间)

Corollary 2.4.3 — 麦克劳林 (Maclaurin) 公式. 在零点展开麦克劳林 (Maclaurin) 公式

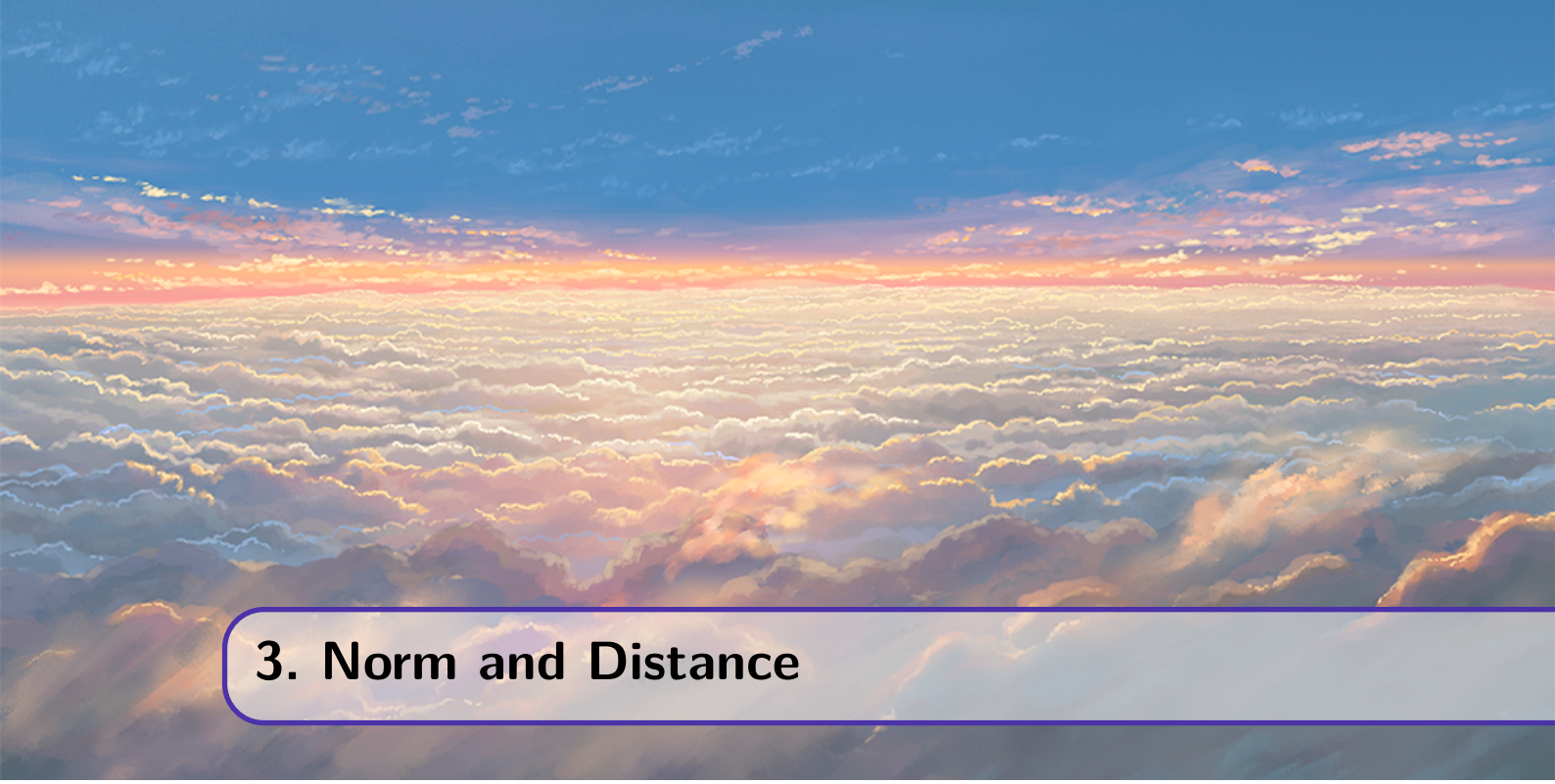
$$f(x) = f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \cdots + \frac{f^{(n)}(0)}{n!}x^n + \frac{f^{(n+1)}(\theta x)}{(n+1)!}x^{n+1} \qquad (0 < \theta < 1)$$

2.5 Regression Model

Definition 2.5.1 — Regression Model. 回归模型 (regression model) 为关于 x 的仿射函数

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

x 是特征向量 *(feature vector)*, 它的元素 x_i 称为回归元 *(regressors)*. n 维向量 β 是权重向量 *(weight vector)*. 标量 v 是偏移量 *(offset)*. 标量 $\hat{y}$ 是预测值 *(prediction)*. 表示某个实际结果或因变量, 用 y 表示.



3. Norm and Distance

3.1 Vector Norm

Definition 3.1.1 — Vector Norm. 在向量空间中存在一个函数 $\|\cdot\|:\mathbb{R}^n\to\mathbb{R}$, 且满足以下条件

- 齐次性: $\|\alpha x\|=|\alpha|\|x\|,\quad \alpha\in\mathbb{R}$ 且 $x\in\mathbb{R}^n$;
- 三角不等式: $\|x+y\|\le\|x\|+\|y\|,\quad x,y\in\mathbb{R}^n$;
- 非负性: $\|x\|\ge 0, \boldsymbol{x}\in\mathbb{R}^n$ 且 $\|\boldsymbol{x}\|=0\Leftrightarrow\boldsymbol{x}=0$;

则称 $\|\cdot\|$ 为向量范数.

3.1.1 ℓ_1-范数

▪ **Example 3.1 — ℓ_1-范数（曼哈顿范数, Manhattan norm）.**

$$\|x\|_1=|x_1|+|x_2|+\ldots+|x_n|\quad x,y\in\mathbb{R}^n,\alpha\in\mathbb{R}$$

▪

Proof.

$$\|\alpha x\|_1=|\alpha x_1|+|\alpha x_2|+\cdots+|\alpha x_n|=|\alpha|\|x\|_1\ge 0$$

$$\|x+y\|_1=|x_1+y_1|+\cdots+|x_n+y_n|\le|x_1|+|y_1|+\cdots+|x_n|+|y_n|=\|x\|_1+\|y\|_1$$

■

3.1.2 ℓ_2-范数

▪ **Example 3.2 — ℓ_2-范数（欧几里得范数, Euclidean norm）.**

$$\|x\|_2=\sqrt{\left(x_1^2+x_2^2+\cdots+x_n^2\right)}=\sqrt{x^Tx}=(\langle x,x\rangle)^{\frac{1}{2}}$$

▪

Proof.

$$\|\alpha x\|_2 = (\langle \alpha x, \alpha x\rangle)^{\frac{1}{2}} = |\alpha|(\langle x, x\rangle)^{\frac{1}{2}} = |\alpha|\|x\|_2$$

$$\begin{aligned}\|x+y\|_2^2 &= \langle x+y, x+y\rangle = \langle x,x\rangle + \langle x,y\rangle + \langle y,x\rangle + \langle y,y\rangle \\ &= \|x\|_2^2 + 2\langle x,y\rangle + \|y\|_2^2 \le \|x\|_2^2 + 2\|x\|_2\|y\|_2 + \|y\|_2^2 \\ &= (\|x\|_2 + \|y\|_2)^2\end{aligned}$$

对于

$$\|x+y\|_2 \le \|x\|_2 + \|y\|_2$$

(for vectors a, b of equal size)

$$\begin{aligned}\|a+b\|^2 &= (a+b)^T(a+b) \\ &= a^Ta + b^Ta + a^Tb + b^Tb \\ &= \|a\|^2 + 2a^Tb + \|b\|^2 \\ &\le \|a\|^2 + 2\|a\|\|b\| + \|b\|^2 \quad \text{(by Cauchy-Schwarz)} \\ &= (\|a\| + \|b\|)^2\end{aligned}$$

taking square-roots gives the triangle inequality
triangle inequality is an equality if and only if $a^Tb = \|a\|\|b\|$
Note from line 3 that $\|a+b\|^2 = \|a\|^2 + \|b\|^2$ if $a^Tb = 0$. ■

Corollary 3.1.1 — ℓ_2-Norm of block vector. If a, b are vectors,

$$\left\| \begin{bmatrix} a \\ b \end{bmatrix} \right\|_2 = \sqrt{\|a\|^2 + \|b\|^2}$$

Cauchy–Schwarz inequality

Theorem 3.1.2 $\left|a^Tb\right| \le \|a\|\|b\|$ for all $a, b \in \mathbf{R}^n$

Corollary 3.1.3 moreover, equality $\left|a^Tb\right| = \|a\|\|b\|$ holds if:

- $a = 0$ or $b = 0$; in this case $a^Tb = 0 = \|a\|\|b\|$
- $a \neq 0$ and $b \neq 0$, and $b = \gamma a$ for some $\gamma > 0$; in this case

$$0 < a^Tb = \gamma\|a\|^2 = \|a\|\|b\|$$

- $a \neq 0$ and $b \neq 0$, and $b = -\gamma a$ for some $\gamma > 0$; in this case

$$0 > a^Tb = -\gamma\|a\|^2 = -\|a\|\|b\|$$

Proof. 1. trivial if $a = 0$ or $b = 0$
2. assume $\|a\| = \|b\| = 1$; we show that $-1 \le a^Tb \le 1$

$$\begin{aligned}0 &\le \|a-b\|^2 \\ &= (a-b)^T(a-b) \\ &= \|a\|^2 - 2a^Tb + \|b\|^2 \\ &= 2\left(1 - a^Tb\right) \\ &\text{with equality only if } a = b\end{aligned}$$

$$\begin{aligned}0 &\le \|a+b\|^2 \\ &= (a+b)^T(a+b) \\ &= \|a\|^2 + 2a^Tb + \|b\|^2 \\ &= 2\left(1 + a^Tb\right) \\ &\text{with equality only if } a = -b\end{aligned}$$

3. for general nonzero a, b, apply case 2 to the unit-norm vectors

$$\frac{1}{\|a\|}a, \quad \frac{1}{\|b\|}b$$

■

Corollary 3.1.4 — 用 ℓ_2 范数表示的柯西—施瓦茨不等式.

$$|\langle x, y\rangle|^2 \le \langle x, x\rangle\langle y, y\rangle = \|x\|_2^2\|y\|_2^2$$

3.1.3 ℓ_∞-范数

Definition 3.1.2 — ℓ_∞-范数.

$$\|x\|_\infty = \max_{1\le i\le n} |x_i|, x \in \mathbb{R}^n$$

Proof.

$$\begin{aligned}\max_{1\le i\le n} |x_i| &\le (|x_1|^p + \cdots + |x_i|^p + \cdots + |x_n|^p)^{1/p} \\ &\le \left(n \max_{1\le i\le n} |x_i|^p\right)^{1/p} \\ &= n^{1/p} \max_{1\le i\le n} |x_i| \\ &\to \max_{1\le i\le n} |x_i| \quad (p \to \infty)\end{aligned}$$

■

3.1.4 ℓ_p-范数

Definition 3.1.3 — ℓ_p-范数.

$$\|x\|_p = \left(x_1^p + x_2^p + \cdots + x_n^p\right)^{\frac{1}{p}}, \quad x \in \mathbb{R}^n, p \ge 1$$

ℓ_1 范数 $\|x\|_1$,ℓ_2-范数 $\|x\|_2$, ℓ_∞-范数是 ℓ_p-范数的特例.

证明可以使用以下两条不等式

Theorem 3.1.5 — Minkowshi Inequality.

$$\left(\sum_{i=1}^n |x_i + y_i|^p\right)^{\frac{1}{p}} \le \left(\sum_{i=1}^n |x_i|^p\right)^{\frac{1}{p}} + \left(\sum_{i=1}^n |y_i|^p\right)^{\frac{1}{p}}, p \ge 1, x, y \in \mathbb{R}^n$$

Theorem 3.1.6 — Hölder Inequality.

$$\sum_{i=1}^n |x_i y_i| \le \left(\sum_{i=1}^n |x_i|^p\right)^{1/p} \left(\sum_{i=1}^n |y_i|^q\right)^{1/q}, \frac{1}{p} + \frac{1}{q} = 1, 1 < p, q < \infty$$

3.2 Root Mean Square Value (RMS)

Definition 3.2.1 — 向量 x 的均方值 (mean-square value). 向量 $x \in \mathbb{R}^n$ 的均方值 (mean-square value)

$$\frac{x_1^2 + x_2^2 + \cdots + x_n^2}{n} = \frac{\|x\|_2^2}{n}$$

Definition 3.2.2 — n 维向量 x 的均方根 (root-mean-square value, RMS).

$$\mathrm{rms}(x) = \sqrt{\frac{x_1^2 + x_2^2 + \cdots + x_n^2}{n}} = \frac{\|x\|_2}{\sqrt{n}}$$

$\mathrm{rms}(x)$ 给出了 $|x_i|$ 的 “典型" (typical) 值. 例如, $\mathrm{rms}(\mathbf{1}) = 1$ (与 n 无关). 均方根 (RMS) 值对于比较不同长度的向量大小是比较有用的.

Theorem 3.2.1

$$|\operatorname{avg}(x)| \le \mathrm{rms}(x)$$

3.3 Chebyshev's Inequality

Theorem 3.3.1 — Chebyshev's Inequality.

$$P(|X - \mu| \ge \varepsilon) \le \frac{\sigma^2}{\varepsilon^2}$$

$$P(|X - \mu| < \varepsilon) \ge 1 - \frac{\sigma^2}{\varepsilon^2}$$

Theorem 3.3.2 — Chebyshev's Inequality. 假设 k 为向量 x 分量满足条件 $|x_i| \ge a$ 的个数, 即 $x_i^2 \ge a^2$ 的个数.

则满足 $|x_i| \ge a$ 的 x_i 数量不会超过 $\frac{\|x\|_2^2}{a^2}$。

Proof. 假设 k 为向量 x 分量满足条件 $|x_i| \ge a$ 的个数, 即 $x_i^2 \ge a^2$ 的个数。

将 a^2 移项, 可得到 $k \le \frac{\|x\|_2^2}{a^2}$

因此

$$\|x\|_2^2 = x_1^2 + x_2^2 + \cdots + x_n^2 \ge ka^2$$

■

Corollary 3.3.3 — Chebyshev's Inequality Using RMS.

$$\mathrm{rms}(x) = \sqrt{\frac{x_1^2 + x_2^2 + \cdots + x_n^2}{n}} = \frac{\|x\|_2}{\sqrt{n}}$$

$|x_i| \ge a$ 的项数占整体的比例不会超过 $\left(\frac{\mathrm{rms}(x)}{a}\right)^2$, 即 $\frac{k}{n} \le \left(\frac{\mathrm{rms}(x)}{a}\right)^2$

3.4 Distance

Definition 3.4.1 — Euclidean distance. n 维向量 a 和 b 之间的欧氏距离

$$\mathrm{dist}(a,b) = \|a-b\|_2$$

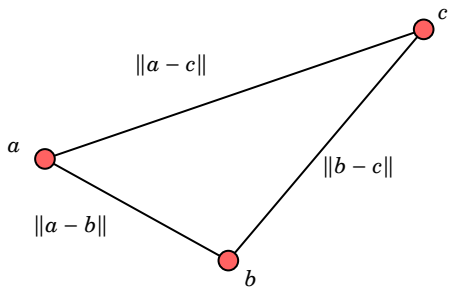
Theorem 3.4.1 $\|a-b\| \geq 0$ for all a, b and $\|a-b\| = 0$ only if $a = b$

Definition 3.4.2 — RMS deviation. $\mathrm{rms}(a-b)$ 是 a 和 b 之间的均方根偏差.

Theorem 3.4.2 — Trianglar Inequality.

$$\|a-c\|_2 = \|(a-b)+(b-c)\|_2 \leq \|a-b\|_2 + \|b-c\|_2 \text{ for all } a, b, c$$

Figure 3.1: 三角形边长



Theorem 3.4.3 RMS deviation between n-vectors a and b is $\mathrm{rms}(a-b) = \frac{\|a-b\|}{\sqrt{n}}$.

3.4.1 Feature Distance and Nearest Neighbor

Definition 3.4.3 — Feature Distance. 如果 x 和 y 分别为两个实体的特征向量, 那么它们的特征距离 (feature distance) 为 $\|x-y\|_2$

Definition 3.4.4 — Nearest Neighbor. 给定向量 x, 一个组向量 $z_1, \ldots, z_m$, 当 $\hat{q}_j$ 满足:

$$\|x - z_j\|_2 \leq \|x - z_i\|_2, \quad i = 1, \ldots, m$$

则称 z_j 是 x 的最近邻 (nearest neighbor)

3.5 Standard Derivation

Definition 3.5.1 — 算术平均值. 对于 n 维向量 x

$$\mathrm{avg}(x) = \frac{\mathbf{1}^T x}{n}$$

Definition 3.5.2 — De-meaned Vector.

$$\tilde{x} = x - \text{avg}(x)\mathbf{1} = \begin{bmatrix} x_1 - \mathbf{avg}(x) \\ x_2 - \mathbf{avg}(x) \\ \vdots \\ x_n - \mathbf{avg}(x) \end{bmatrix} = \begin{bmatrix} x_1 - \frac{\mathbf{1}^T x}{n} \\ x_2 - \frac{\mathbf{1}^T x}{n} \\ \vdots \\ x_n - \frac{\mathbf{1}^T x}{n} \end{bmatrix}$$

因此 $\text{avg}(\tilde{x}) = 0$

Definition 3.5.3 — x 的标准差.

$$\text{std}(x) = \text{rms}(\tilde{x}) = \frac{\left\| x - (1^T x/n)\, 1 \right\|_2}{\sqrt{n}}$$

$\text{std}(x)$ 表示数据元素的变化程度. 对于常数 a, 当且仅当 $x = a\mathbf{1}$ 时, $\text{std}(x) = 0$.

Corollary 3.5.1 the de-meaned vector in standard units is

$$\frac{1}{\text{std}(a)}(a - \text{avg}(a)\mathbf{1})$$

Theorem 3.5.2

$$\text{rms}(x)^2 = \text{avg}(x)^2 + \text{std}(x)^2$$

Proof.

$$\begin{aligned} \text{std}(x)^2 &= \frac{\|\mathbf{avg}(x)\mathbf{1}\|^2}{n} \\ &= \frac{1}{n}\left(x - \frac{\mathbf{1}^T x}{n}\mathbf{1}\right)^T \left(x - \frac{\mathbf{1}^T x}{n}\mathbf{1}\right) \\ &= \frac{1}{n}\left(x^T x - \frac{(\mathbf{1}^T x)^2}{n} - \frac{(\mathbf{1}^T x)^2}{n} + \left(\frac{\mathbf{1}^T x}{n}\right)^2 n\right) \\ &= \frac{1}{n}\left(x^T x - \frac{(\mathbf{1}^T x)^2}{n}\right) \\ &= \text{rms}(x)^2 - \text{avg}(x)^2 \end{aligned}$$

■

3.6 Angle

Definition 3.6.1 — 两个非零向量 a 和 b 之间的角 (angle).

$$\angle(a, b) = \arccos\left(\frac{a^T b}{\|a\|_2 \|b\|_2}\right)$$

$\angle(a, b)$ 的取值范围为 $[0, \pi]$, 且满足

$$a^T b = \|a\|_2 \|b\|_2 \cos(\angle(a, b))$$

在二维和三维向量之中, 这里的角与普通角度 (ordinary angle) 是一致的.

- $\theta = \frac{\pi}{2} = 90$: a 和 b 为正交, 写作 $a \perp b (a^T b = 0)$.

- $\theta = 0$：a 和 b 为同向 (aligned or parallel) 的 ($a^T b = \|a\|\|b\|$).
- $\theta = \pi = 180$：a 和 b 为反向的 (anti-aligned or opposed)($a^T b = -\|a\|\|b\|$).
- $\theta < \frac{\pi}{2} = 90$：a 和 b 成锐角 (acute angle)($a^T b > 0$).
- $\theta > \frac{\pi}{2} = 90$：a 和 b 成钝角 (obtuse angle)($a^T b < 0$).

Theorem 3.6.1 Cauchy-Schwarz inequality guarantees that

$$-1 \leq \frac{a^T b}{\|a\|\|b\|} \leq 1$$

Definition 3.6.2 — **球面的距离.**

$$\mathrm{R}\angle(a, b)$$

3.6.1 相关系数

给定向量 a 和 b, 其去均值向量为：

$$\tilde{a} = a - \mathrm{avg}(a)1, \tilde{b} = b - \mathrm{avg}(b)1$$

Definition 3.6.3 — a **和** b **的相关系数.**

$$\rho = \frac{\tilde{a}^T \tilde{b}}{\|\tilde{a}\|_2 \|\tilde{b}\|_2} = \cos \angle(\tilde{a}, \tilde{b})$$

where $\tilde{a} \neq 0$, $\tilde{b} \neq 0$.

It is only defined when a and b are not constant ($\tilde{a} \neq 0$ and $\tilde{b} \neq 0$), and is a number between -1 and 1. ρ_{ab} is the cosine of the angle between the de-meaned vectors.

Theorem 3.6.2 ρ_{ab} is the average product of the deviations from the mean in standard units

$$\rho_{ab} = \frac{1}{n} \sum_{i=1}^{n} \frac{(a_i - \mathbf{avg}(a))}{\mathrm{std}(a)} \frac{(b_i - \mathbf{avg}(b))}{\mathrm{std}(b)}$$

■ **Example 3.3** 高度相关的向量：
- 邻近地区的降雨时间序列.
- 类型密切相关文档的单词计数向量.
- 同行业中类似公司的日收益.

比较不相关的向量：
- 无关的向量.
- 音频信号 (比如, 在多轨录音中的不同轨).

负相关的向量：
- 深圳与墨尔本的每天气温变化

■

3.7 Regression Line

- scatter plot shows two n-vectors a, b as n points (a_k, b_k)
  - straight line shows affine function $f(x) = c_1 + c_2 x$ with

$$f(a_k) \approx b_k, \quad k = 1, \ldots, n$$

Problem 3.1 use coefficients c_1, c_2 that minimize $J = \frac{1}{n}\sum_{k=1}^{n}\left(f\left(a_k\right) - b_k\right)^2$

J is a quadratic function of c_1 and c_2 :

$$\begin{aligned} J &= \frac{1}{n}\sum_{k=1}^{n}\left(c_1 + c_2 a_k - b_k\right)^2 \\ &= \frac{nc_1^2 + 2n\,\mathrm{avg}(a)c_1c_2 + \|a\|^2c_2^2 - 2n\,\mathrm{avg}(b)c_1 - 2a^Tbc_2 + \|b\|^2}{n} \end{aligned}$$

■

to minimize J, set derivatives with respect to c_1, c_2 to zero:

$$c_1 + \mathrm{avg}(a)c_2 = \mathrm{avg}(b), \quad n\,\mathrm{avg}(a)c_1 + \|a\|^2c_2 = a^Tb$$

Theorem 3.7.1 solution is

$$c_2 = \frac{a^Tb - n\,\mathrm{avg}(a)\,\mathrm{avg}(b)}{\|a\|^2 - n\,\mathrm{avg}(a)^2}, \quad c_1 = \mathrm{avg}(b) - \mathrm{avg}(a)c_2$$

Corollary 3.7.2 slope c_2 can be written in terms of correlation coefficient of a and b :

$$c_2 = \frac{(a - \mathrm{avg}(a)\mathbf{1})^T(b - \mathrm{avg}(b)\mathbf{1})}{\|a - \mathrm{avg}(a)\mathbf{1}\|^2} = \rho_{ab}\frac{\mathrm{std}(b)}{\mathrm{std}(a)}$$

hence,

Corollary 3.7.3 expression for regression line can be written as

$$f(x) = \mathrm{avg}(b) + \frac{\rho_{ab}\,\mathrm{std}(b)}{\mathrm{std}(a)}(x - \mathrm{avg}(a))$$

Corollary 3.7.4 correlation coefficient ρ_{ab} is the slope after converting to standard units:

$$\frac{f(x) - \mathbf{avg}(b)}{\mathrm{std}(b)} = \rho_{ab}\frac{x - \mathbf{avg}(a)}{\mathrm{std}(a)}$$

3.8 Norm for Complex Numbers

Definition 3.8.1 norm of vector $a \in \mathbf{C}^n$:

$$\begin{aligned} \|a\| &= \sqrt{|a_1|^2 + |a_2|^2 + \cdots + |a_n|^2} \\ &= \sqrt{a^Ha} \end{aligned}$$

Theorem 3.8.1 — positive definite. $\|a\| \geq 0$ for all a, $\|a\| = 0$ only if $a = 0$

Theorem 3.8.2 — homogeneous. $\|\beta a\| = |\beta|\|a\|$ for all vectors a, complex scalars β

Theorem 3.8.3 — triangle inequality. $\|a+b\| \leq \|a\|+\|b\|$ for all vectors a, b of equal size

Theorem 3.8.4 — Cauchy–Schwarz inequality for complex vectors. $\left|a^H b\right| \leq \|a\|\|b\|$ for all $a, b \in \mathbf{C}^n$

moreover, equality $\left|a^H b\right| = \|a\|\|b\|$ holds if:

- $a = 0$ or $b = 0$
- $a \neq 0$ and $b \neq 0$, and $b = \gamma a$ for some (complex) scalar γ

We say a and b are orthogonal if $a^H b = 0$.
We will not need definition of angle, correlation coefficient, . . . in $\mathbf{C}^n$.

4. 优化问题初步

4.1 优化问题引入

Problem 4.1 假设 N 个样本向量 $x_1, \ldots, x_N \in \mathbb{R}^n$, 需要找到中心向量（centroid）$z$ 满足

$$\min_{z \in \mathbf{R}^n} \sum_{i=1}^{N} \|x_i - z\|_2^2$$

■

Definition 4.1.1 — 高阶无穷小记号 o. 设 x, y 是同一变化过程中的无穷小, 即 $x \to 0, y \to 0$, 如果它们极限

$$\lim_{x \to 0, y \to 0} \frac{y}{x} = 0$$

则称 y 是 x 的高阶无穷小, 记作 $y = o(x)$.

Corollary 4.1.1

$$\lim_{x \to 0, y \to 0} \frac{y}{Cx} = \frac{1}{C} \lim_{x \to 0, y \to 0} \frac{y}{x} = 0$$

也即则称 y 是 Cx 的高阶无穷小, 记作 $y = o(Cx)$.

Theorem 4.1.2 — 优化求解的必要条件. 假设函数 f 在 $\hat{x}$ 可微, 则有

$$\hat{x} = \arg\min_{x \in \mathbb{R}^n} f(x) \Rightarrow \nabla f(\hat{x}) = 0$$

Proof. 假设函数 f 在 $\hat{x}$ 一阶泰勒展开, 有

$$f(x) = f(\hat{x}) + \langle \nabla f(\hat{x}), x - \hat{x} \rangle + o\left(\|x - \hat{x}\|_2\right)$$

假设 $\delta f(\hat{x}) \neq 0$, 则令 $\tilde{x} = \hat{x} - t\nabla f(\hat{x}), t > 0$, 可得

$$f(\tilde{x}) = f(\hat{x}) - t\|\nabla f(\hat{x})\|_2^2 + o\left(t\|\nabla f(\hat{x})\|_2\right)$$

当 $t \to 0$ 则 $t\|\nabla f(\hat{x})\|_2 \to 0$, 高阶无穷小 $o'\left(t\|\nabla f(\hat{x})\|_2\right) \to 0$
当 t 足够小时, 存在 $t\|\nabla f(\hat{x})\|_2 \geq o\left(t\|\nabla f(\hat{x})\|_2\right)$, 即

$$-t\|\nabla f(\hat{x})\|_2^2 + o\left(t\|\nabla f(\hat{x})\|_2\right) \leq 0$$

$$f(\tilde{x}) = f(\hat{x}) - t\|\nabla f(\hat{x})\|_2^2 + o\left(t\|\nabla f(\hat{x})\|_2\right) \leq f(\hat{x})$$

与 $\hat{x} = \arg\min_{\mathbf{R}^n} f(x)$ 矛盾.
$\nabla f(\hat{x}) = 0$, 是最优问题解的必要条件. 通常 $\nabla f(\hat{x}) = 0 \Leftrightarrow \hat{x} = \arg\min_{\mathbf{R}^n} f(x)$. ■

▪ **Example 4.1**

$$f(x) = -x^2, \quad x \in \mathbb{R}, \hat{x} = \operatorname{argmin}_{\mathbb{R}} f(x)$$

$\nabla f(\hat{x}) = 0$, 则有 $-2\hat{x} = 0$, 即 $\hat{x} = 0$

$$f(\hat{x}) = 0 \geq f(x), \quad x \in \mathbf{R}$$

▪

4.2 Convex Set

Definition 4.2.1 — **凸集**. $\forall x, y \in \Omega, \alpha \in \mathbb{R}, 0 \leq \alpha \leq 1$ 有

$$\alpha x + (1-\alpha)y \in \Omega$$

则定义域 $\Omega \in \mathbb{R}^n$ 称为凸的 (Convex) 集合
(域内两点连线之间都属于这个域)

Definition 4.2.2 — **凸函数**. 设函数 $f(x)$ 定义于称为凸的定义域 $\Omega \in \mathbb{R}^n$ 满足

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y), \forall x, y \in \Omega, \alpha \in \mathbf{R}, 0 \leq \alpha \leq 1$$

称其为凸函数.

4.2.1 Examples for Convex Functions

▪ **Example 4.2**

$$f(x) = x^2, x \in \mathbf{R}$$

$$\begin{aligned} f(\alpha x + (1-\alpha)y) &= (\alpha x + (1-\alpha)y)^2 \\ &= \alpha^2 x^2 + 2\alpha(1-\alpha)xy + (1-\alpha)^2 y^2 \\ &= \alpha x^2 + (1-\alpha)y^2 + \left(\alpha^2 - \alpha\right)x^2 + \left(\alpha^2 - \alpha\right)y^2 + 2\alpha(1-\alpha)xy \\ &= \alpha x^2 + (1-\alpha)y^2 - \alpha(1-\alpha)(x-y)^2 \\ &\leq \alpha x^2 + (1-\alpha)y^2 = \alpha f(x) + (1-\alpha)f(y) \end{aligned}$$

▪

■ **Example 4.3** $f(x) = \|x\|$, 其中 $\|\cdot\|$ 表示 $\mathbb{R}^n$ 上的向量范数, $x \in \mathbb{R}^n$. ■

Proof.

$$\|\alpha x + (1-\alpha)y\| \le \|\alpha x\| + \|(1-\alpha)y\| = |\alpha|\|x\| + |1-\alpha|\|y\|$$

■

■ **Example 4.4**

$$f(x) = \|x\|_2^2, x \in \mathbb{R}^n$$

■

4.2.2 凸函数的充要条件

Theorem 4.2.1 — 可微函数 f 是凸函数的充要条件.

$$f(y) \ge f(x) + \langle \nabla f(x), y - x \rangle, \quad \forall x, y$$

Proof. 首先, 证明一维情况 $f : \mathbb{R} \to \mathbb{R}, \alpha \in [0,1]$.
$\Rightarrow$ 充分条件: $f(\alpha x + (1-\alpha)y) = f(x + (1-\alpha)(y-x)) \le \alpha f(x) + (1-\alpha)f(y)$, 有

$$f(y) \ge f(x) + \frac{f(x + (1-\alpha)(y-x)) - f(x)}{(1-\alpha)(y-x)}(y-x)$$

令 $\alpha \to 1^-$, 则有 $f(y) \ge f(x) + f'(x)(y-x)$.
$\Leftarrow$ 必要条件: 令 $y \ne x, z = \alpha x + (1-\alpha)y$ 则有

$$f(x) \ge f(z) + f'(z)(x-z), f(y) \ge f(z) + f'(z)(y-z)$$

可得

$$\begin{aligned} \alpha f(x) + (1-\alpha)f(y) &\ge f(z) + \alpha f'(z)(x-z) + (1-\alpha)f'(z)(y-z) \\ &= f(z) + f'(z)(\alpha x + (1-\alpha)y - z) \\ &= f(z) \end{aligned}$$

证明 n 维情况 $f : \mathbb{R}^n \to \mathbb{R}$.
$\Rightarrow$ 充分条件: 令 $g(t) = f(tx + (1-t)y), t \in \mathbb{R}$, 则 $g'(t) = \langle \nabla f(tx + (1-t)y), x - y \rangle$ 由于 f 是凸函数, 证明 $g(t)$ 也是凸函数; 并可得 $g(0) \ge g(1) + g'(1)(-1)$, 得证.
$\Leftarrow$ 必要条件: 与一维类似. (将 f' 改为 $\nabla f(z)^T$) ■

Theorem 4.2.2 如果可微函数 f 是凸函数, 则有

$$\hat{x} = \arg\min_{x \in \mathbb{R}^n} f(x) \Leftrightarrow \nabla f(\hat{x}) = 0$$

Proof. 已证 $\hat{x} = \arg\min_{x \in \mathbb{R}^n} f(x) \Rightarrow$ 可得 $\nabla f(\hat{x}) = 0$
只需证 $\nabla f(\hat{x}) = 0 \Rightarrow \hat{x} = \arg\min_{x \in \mathbb{R}^n} f(x)$.
由于函数 f 是可微凸的, 则有 $\forall x \in \mathbb{R}^n$,

$$\begin{aligned} f(x) &\ge f(\hat{x}) + \langle \nabla f(\hat{x}), x - \hat{x} \rangle \\ &= f(\hat{x}) + \langle 0, x - \hat{x} \rangle \ge f(\hat{x}) \end{aligned}$$

可得 $f(x) \ge f(\hat{x}), \hat{x} = \arg\min_{x \in \mathbb{R}^n} f(x)$. ■

4.3 向量偏导

Definition 4.3.1 — 向量对向量的导数.

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, z = \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix}$$

$$\nabla f(z) = \begin{bmatrix} \frac{\partial f(z)}{\partial z_1} \\ \vdots \\ \frac{\partial f(z)}{\partial z_n} \end{bmatrix}$$

■ **Example 4.5**

$$f(z) = x^T z + z^T z = \sum_{i=1}^{n} \left\{ x_i z_i + z_i^2 \right\}$$

它的导数是

$$\nabla f(z) = \begin{bmatrix} \frac{\partial f(z)}{\partial z_1} \\ \vdots \\ \frac{\partial f(z)}{\partial z_n} \end{bmatrix} = \begin{bmatrix} x_1 + 2z_1 \\ \vdots \\ x_n + 2z_n \end{bmatrix} = x + 2z$$

■

问题4.1中已知目标函数是凸函数.（见4.2, 4.3, 4.2.1）
则可以求解

Proof.

$$f(z) = \sum_{i=1}^{N} \|x_i - z\|_2^2 = \sum_{i=1}^{N} \langle x_i - z, x_i - z \rangle = \sum_{i=1}^{N} \left\{ x_i^T x_i - 2x_i^T z + z_i^T z \right\}$$

利用等价条件4.2.1

$$\nabla f(z) = \sum_{i=1}^{N} \{-2x_i + 2z\} = 0$$

(求导 4.3.1)

$$z = \frac{1}{N} \sum_{i=1}^{N} x_i$$

■

另解:

Proof. $J(x_0) = \sum_{i=1}^{n} \|x_0 - x_i\|^2$ ， 其中 $m = \frac{1}{n} \sum_{i=1}^{n} x_i$

$$
\begin{aligned}
J(x_0) &= \sum_{i=1}^{n} \|(x_0 - m) - (x_i - m)\|^2 \\
&= \sum_{i=1}^{n} \|x_0 - m\|^2 - 2\sum_{i=1}^{n} (x_0 - m)^T (x_i - m) + \sum_{i=1}^{n} \|x_i - m\|^2 \\
&= \sum_{i=1}^{n} \|x_0 - m\|^2 - 2(x_0 - m)^T \sum_{i=1}^{n} (x_i - m) + \sum_{i=1}^{n} \|x_i - m\|^2
\end{aligned}
$$

因为

$$
\sum_{i=1}^{n} (x_i - m) = \sum_{i=1}^{n} x_i - nm = \sum_{i=1}^{n} x_i - n \cdot \frac{1}{n} \sum_{i=1}^{n} x_i = 0
$$

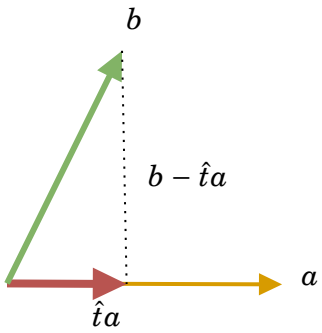
所以有

$$
J(x_0) = \sum_{i=1}^{n} \|x_0 - m\|^2 + \sum_{i=1}^{n} \|x_i - m\|^2
$$

即当 x_0 等于均值时，有最小均方. ■

4.4 标量优化问题：投影问题

Figure 4.1: Projection onto a line



Problem 4.2 假设 $a, b \in \mathbb{R}^n, a \neq 0, t \in \mathbb{R}$, 当 t 多大时, ta 到 b 之间的距离最小

$$
\hat{t} = \min_t \|ta - b\|_2^2
$$

■

t 是标量，ta 是向量.

定义

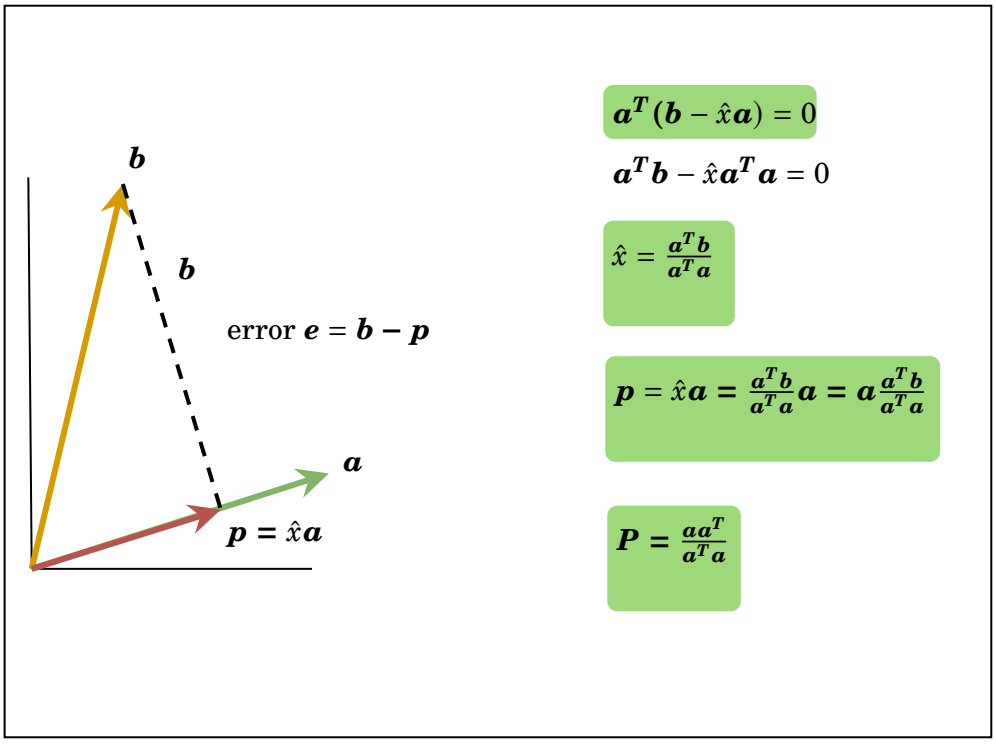
$$
f(t) = \|ta - b\|_2^2 = \langle ta - b, ta - b \rangle = t^2 a^T a - 2ta^T b + b^T b
$$

It is a quadratic function of t with positive leading coefficient $a^T a$. 可以验证 $f(t)$ 满足凸函数的定义。

$$
\nabla f(t) = 2ta^T a - 2a^T b = 0
$$

$$\hat{t} = \frac{a^T b}{a^T a} = \frac{a^T b}{\|a\|_2^2}$$

Figure 4.2: Projection onto a line from the prospective of linear algebra



4.5 Clustering

将物理或抽象对象的集合分成由类似特征组成的多个类的过程称为聚类 *(clustering)*.

目标：分成 k 个集合, 尽量使得同一个集合中的向量彼此接近.

Notation 4.1. 给定 N 个 n 维向量 $x_1, \ldots, x_N \in \mathbb{R}^n$

- 标签 $c_i \in \{1, 2, \cdots, k\}$ 表示向量 x_i 所属类别, 例如 $c_i = 2$ 表示 x_i 属于第 **2** 类.
- 对于 $j = 1, \ldots, k$, $G_j = \{i : c_i = j\}$ 表示属于第 j 类的向量 x_i 的下标集合.
- 向量 $z_j, j = 1, \ldots, k$, 表示同属于 j 类的向量 $x_i, i \in G_j$ 的聚类中心.

聚类目标是找到向量 x_i 的“标签 c_i ”和“聚类中心 z_j ”

Problem 4.3

$$\min_{z_j} \sum_{i \in G_j} \|x_i - z_j\|_2^2, j = 1, \cdots, k$$

$$c_i = \underset{j=\{1,\cdots,k\}}{\operatorname{argmin}} \|x_i - z_j\|_2^2, i = 1, 2, \ldots, N$$

■

k-means 算法是将 N 向量 $x_i \in \mathbb{R}^n$ 划分成 k 类的迭代聚类算法.

Proof. 更新聚类标签 c_i:

$$\|x_i - z_j\|_2^2 = \operatorname{argmin} \left\{ \|x_i - z_1\|_2^2, \|x_i - z_2\|_2^2, \cdots, \|x_i - z_k\|_2^2 \right\}$$

Algorithm 1: k-means Algorithm

1 在 N 个点中随机选取 k 个点, 分别作为聚类中心 z_j
2 更新聚类标签 c_i : 计算每个点 x_i 到 k 个聚类中心 z_j 的距离, 并将其分配到最近的聚类中心 z_j 所在的聚类中 $c_i = j$
3 更新聚类中心 z_j : 重新计算每个聚类现在的质心, 并以其作作为新的聚类中心, 根据更新标签 c_i, 更新属于第 j 类下标集合 $G_j = \{i : c_i = j\}$, 重新计算 c_i 类的聚类中心 z_j
4 重复步骤 2、3, 直到所有聚类中心不再变化

更新聚类中心 z_j:

$$\nabla f_j\left(z_j\right) = \sum_{i \in G_j} 2\left(x_i - z_j\right) = 0$$

$$z_j = \frac{1}{\left|G_j\right|} \sum_{i \in G_j} x_i$$

$\left|G_j\right|$ 表示集合 G_j 中元素的数目. ■

在每一次迭代中目标函数 J 都会下降, 直到聚类中心 $z_1, \cdots, z_k$ 和划分聚类标签集合 $G_1, \cdots, G_k$ 不再变化.

但是 k-means 算法依赖于初始随机生成的聚类中心, 只可得到目标函数 J 的局部局部最优.

解决方案: 使用不同的 (随机的) 初始聚类中心运行 k-means 算法若干次, 取目标函数 J 值最小的一次作为最终的聚类结果.

5. Linear Independence

5.1 线性相关、线性无关

Definition 5.1.1 — 线性相关 (linearly dependent). 定义：对于向量 $a_1, \dots a_m \in \mathbb{R}^n$, 如果存在不全为零的数 $\beta_1, \dots \beta_m \in \mathbb{R}$, 使得

$$\beta_1 a_1 + \cdots + \beta_m a_m = 0$$

则称向量 $a_1, \dots a_m$ 是线性相关 (linearly dependent).

Corollary 5.1.1 线性相关等价于至少有一个向量 a_i 是其它向量的线性组合.
equivalently, at least one vector a_i is a linear combination of the other vectors:

$$a_i = -\frac{x_1}{x_i}a_1 - \cdots - \frac{x_{i-1}}{x_i}a_{i-1} - \frac{x_{i+1}}{x_i}a_{i+1} - \cdots - \frac{x_n}{x_i}a_n$$

if $x_i \neq 0$.

Corollary 5.1.2 the vector o can be written as a nontrivial linear combination of $a_1, \dots, a_n$.

Corollary 5.1.3 向量集 $\{a_1\}$ 是线性相关的, 当且仅当 $a_1 = 0$.

Corollary 5.1.4 向量集 $\{a_1, a_2\}$ 是线性相关的, 当且仅当其中一个 $a_1 = \beta a_2, \beta \neq 0$.

Definition 5.1.2 — 线性独立 (linearly independent). 如果 n 维向量集 $\{a_1, \dots, a_m\}$ 不是

线性相关的, 即线性独立 (linearly dependent), 也称线性无关, 即:

$$\beta_1 a_1 + \cdots + \beta_m a_m = 0$$

当且仅当 $\beta_1 = \cdots = \beta_m = 0$, 上述等式成立.

线性无关等价于不存在一个向量 a_i 是其它向量的线性组合.

Corollary 5.1.5 注: 一个 n 维向量集最多有 n 个线性无关的向量, 也就是说如果 n 维向量集有 $n+1$ 个向量, 那它们必线性相关

- **Example 5.1** n 维单位向量 $e_1, \ldots, e_n$ 是线性独立的. ∎
- **Example 5.2**

$$a_1 = \begin{bmatrix} 1 \\ -2 \\ 0 \end{bmatrix}, \quad a_2 = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}, \quad a_3 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

$$\beta_1 a_1 + \beta_2 a_2 + \beta_3 a_3 = \begin{bmatrix} \beta_1 - \beta_2 \\ -2\beta_1 + \beta_3 \\ \beta_2 + \beta_3 \end{bmatrix} = 0$$

$$\beta_1 = \beta_2 = \beta_3 = 0$$

∎

Corollary 5.1.6

$$A = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix}$$

has linearly independent columns if

$$Ax = 0 \quad \Longrightarrow \quad x = 0$$

Theorem 5.1.7 假设 x 是线性无关向量 $a_1, \ldots, a_k$ 的线性组合:

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

则其系数 $\beta_1, \ldots \beta_k$ 是唯一的, 即如果有:

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

则对于 $i = 1, \ldots k$, 有 $\beta_i = \gamma_i$.

Proof. 系数是唯一的原因:

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

由于向量 $a_1, \ldots, a_k$ 线性无关, 有 $\beta_1 - \gamma_1 = \beta_k - \gamma_k = 0$. ∎

5.2 Basis

Definition 5.2.1 — **基 (Basis).** n 个线性独立的 n 维向量 $a_1, \ldots, a_n$ 的集合

Definition 5.2.2 — **向量 b 在基底 $a_1, \ldots, a_n$ 下的分解.** 任何一个 n 维向量 b 都可以用它们的线性组合来表示

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

Proof. 同一向量的系数是唯一的. ■

▪ **Example 5.3** $e_1, \ldots, e_n$ 是一组基, 那么 b 在此基底下的分解为

$$b = b_1 e_1 + \cdots + b_n e_n, b = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} \in \mathbb{R}^n$$

▪

5.3 标准正交向量

Definition 5.3.1 — **Orthogonal Vectors.** 在 n 维向量集 $a_1, \ldots, a_k$ 中, 如果对于 $i \neq j$, 都有 $a_i \perp a_j$, 则称它们相互正交 (orthogonal).

Definition 5.3.2 — **Orthonormal Vectors.** 如果 n 维向量集 $a_1, \ldots, a_k$ 相互正交, 且每个向量的模长都为单位长度 1 , 即对于 $i = 1, \ldots k$, 有 $\|a_i\|_2^2 = 1$, 则称它们是标准正交 (orthonormal) 的.

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

Corollary 5.3.1 标准正交的向量集是线性无关的.

Theorem 5.3.2 if n vectors $a_1, a_2, \ldots, a_n$ of length m are linearly independent, then

$$n \leq m$$

(根据线性无关的性质, 必有向量集向量个数 $k \leq n$.)

Proof. The proof is by induction on the dimension n.

First consider a linearly independent collection $a_1, \ldots, a_k$ of 1-vectors. We must have $a_1 \neq 0$. This means that every element a_i of the collection can be expressed as a multiple $a_i = (a_i/a_1)\, a_1$ of the first element a_1. This contradicts linear independence unless $k = 1$.

Next suppose $n \geq 2$ and the independence-dimension inequality holds for dimension $n - 1$.

Let $a_1, \ldots, a_k$ be a linearly independent list of n-vectors. We need to show that $k \leq n$. We partition the vectors as

$$a_i = \begin{bmatrix} b_i \\ \alpha_i \end{bmatrix}, \quad i = 1, \ldots, k$$

where b_i is an $(n-1)$-vector and α_i is a scalar.

First suppose that $\alpha_1 = \cdots = \alpha_k = 0$. Then the vectors $b_1, \ldots, b_k$ are linearly independent: $\sum_{i=1}^k \beta_i b_i = 0$ holds if and only if $\sum_{i=1}^k \beta_i a_i = 0$, which is only possible

for $\beta_1 = \cdots = \beta_k = 0$ because the vectors a_i are linearly independent. The vectors $b_1, \ldots, b_k$ therefore form a linearly independent collection of $(n-1)$ -vectors. By the induction hypothesis we have $k \leq n-1$, so certainly $k \leq n$.

Next suppose that the scalars α_i are not all zero. Assume $\alpha_j \neq 0$. We define a collection of $k-1$ vectors c_i of length $n-1$ as follows:

$$c_i = \begin{cases} b_i - \frac{\alpha_i}{\alpha_j} b_j, & i = 1, \ldots, j-1 \\ b_{i+1} - \frac{\alpha_{i+1}}{\alpha_j} b_j, & i = j, \ldots, k-1 \end{cases}$$

These $k-1$ vectors are linearly independent: If $\sum_{i=1}^{k-1} \beta_i c_i = 0$ then

$$\sum_{i=1}^{j-1} \beta_i \begin{bmatrix} b_i \\ \alpha_i \end{bmatrix} + \gamma \begin{bmatrix} b_j \\ \alpha_j \end{bmatrix} + \sum_{i=j+1}^{k} \beta_{i-1} \begin{bmatrix} b_i \\ \alpha_i \end{bmatrix} = 0 \tag{5.1}$$

with

$$\gamma = -\frac{1}{\alpha_j} \left(\sum_{i=1}^{j-1} \beta_i \alpha_i + \sum_{i=j+1}^{k} \beta_{i-1} \alpha_i \right)$$

Since the vectors $a_i = (b_i, \alpha_i)$ are linearly independent, the eq. (5.1) only holds when all the coefficients β_i and γ are all zero. This in turns implies that the vectors $c_1, \ldots, c_{k-1}$ are linearly independent. By the induction hypothesis $k-1 \leq n-1$ so we have established that $k \leq n$ ■

Corollary 5.3.3 If an $m \times n$ matrix has linearly independent columns then $m \geq n$.

Corollary 5.3.4 If an $m \times n$ matrix has linearly independent rows then $m \leq n$.

Definition 5.3.3 — **n 维向量的一个标准正交基.** 当 $k = n$ 时, $a_1, \ldots, a_n$ 是 n 维向量的一个标准正交基.

Definition 5.3.4 — **x 在标准正交基下的标准正交分解.** 如果 $a_1, \ldots, a_n$ 是一个标准正交基, 对于任意维向量 x;

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

则称其为 x 在标准正交基下的标准正交分解.

这个分解可以用于计算不同标准正交基下的系数.

Proof. 由于正交向量的性质

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

所以

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

■

Algorithm 2: Gram-Schmidt Algorithm

Input: n 维向量 $a_1, \ldots, a_k$
Output: 若这些向量线性无关，返回标准正交基 $q_1, \ldots, q_k$；若线性相关时判断 a_j 是 $a_1, \ldots, a_{j-1}$ 的线性组合

1 $q_1 = a_1 / \|a_1\|_2$
2 **while** $i = 2, \cdots, k$ **do**
3 正交化: $\widetilde{q}_i = a_i - \left(q_1^T a_i\right) q_1 - \cdots - \left(q_{i-1}^T a_i\right) q_{i-1}$
4 检验线性相关：如果 $\widetilde{q}_i = 0$, 提前退出迭代
5 单位化: $q_i = \widetilde{q}_i / \|\widetilde{q}_i\|_2$
6 **end**

5.4 Gram-Schmidt Algorithm

如果步骤 2 中未提前结束迭代, 那么 $a_1, \ldots, a_k$ 是线性独立的, 而且 $q_1, \ldots, q_k$ 是标准正交基.

如果在第 j 次迭代中提前结束, 说明 a_j 是 $a_1, \ldots, a_{j-1}$ 的线性组合, 因此 $a_1, \ldots, a_k$ 是线性相关的.

Theorem 5.4.1 $q_1, \ldots, q_{i-1}, q_i$ 是标准正交的.

Proof. 假设第 $i-1$ 次迭代成立, 即: $q_r \perp q_s, \forall r, s < i$.

正交化步骤保证有以下关系成立

$$\widetilde{q}_i = a_i - \left(q_1^T a_i\right) q_1 - \cdots - \left(q_{i-1}^T a_i\right) q_{i-1}$$

等式两边同时乘以 $q_j^T, j = 1, \ldots, i-1$

$$\begin{aligned} q_j^T \tilde{q}_i &= q_j^T a_i - \left(q_1^T a_i\right)\left(q_j^T q_1\right) - \cdots - \left(q_{i-1}^T a_i\right)\left(q_j^T q_{i-1}\right) \\ &= q_j^T a_i - q_j^T a_i \\ &= 0 \end{aligned}$$

$\because q_j^T q_r = 0, j \neq r, q_j^T q_j = 1$
$\therefore \widetilde{q}_i \perp q_1, \ldots, \widetilde{q}_i \perp q_{i-1}$.
单位化步骤保证了 $q_i = \widetilde{q}_i / \|\widetilde{q}_i\|_2$, 即 $q_1, \ldots, q_i$ 是标准正交. ■

5.4.1 The Analysis of Gram-Schmidt Algorithm

假设 Gram-Schmidt 正交法未在第 i 次迭代提前终止:

Corollary 5.4.2 a_i 是 $q_1, \ldots, q_i$ 的一个线性组合.

$$a_i = \|\tilde{q}_i\|_2 q_i + \left(q_1^T a_i\right) q_1 + \cdots + \left(q_{i-1}^T a_i\right) q_{i-1}$$

Proof.

$$\widetilde{q}_i = a_i - \left(q_1^T a_i\right) q_1 - \cdots - \left(q_{i-1}^T a_i\right) q_{i-1}$$

$$a_i = \widetilde{q}_i + \left(q_1^T a_i\right) q_1 + \cdots + \left(q_{i-1}^T a_i\right) q_{i-1}$$

Algorithm 3: Gram-Schmidt Algorithm for Three Vectors

Input: Three independent vectors $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$

Output: Three orthonormal vectors $\boldsymbol{q}_1 = \boldsymbol{A}/\|\boldsymbol{A}\|, \boldsymbol{q}_2 = \boldsymbol{B}/\|\boldsymbol{B}\|, \boldsymbol{q}_3 = \boldsymbol{C}/\|\boldsymbol{C}\|$.

1 Choose $\boldsymbol{A} = \boldsymbol{a}$

2
$$\boldsymbol{B} = \boldsymbol{b} - \frac{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{b}}{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}}\boldsymbol{A}$$

3
$$\boldsymbol{C} = \boldsymbol{c} - \frac{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{c}}{\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}}\boldsymbol{A} - \frac{\boldsymbol{B}^{\mathrm{T}}\boldsymbol{c}}{\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B}}\boldsymbol{B}$$

4 单位化

注意有性质: $q_i = \widetilde{q}_i/\|\widetilde{q}_i\|_2$.

$$a_i = \|\tilde{q}_i\|_2\, q_i + \left(q_1^T a_i\right) q_1 + \cdots + \left(q_{i-1}^T a_i\right) q_{i-1}$$

■

则有

Corollary 5.4.3
$$q_i = \frac{a_i - \left(q_1^T a_i\right) q_1 - \cdots - \left(q_{i-1}^T a_i\right) q_{i-1}}{\|\tilde{q}_i\|_2}$$

Corollary 5.4.4 q_i 是 $a_1, \ldots, a_i$ 的一个线性组合.

Proof. 归纳假设, 每个 q_{i-1} 都是 $a_1, \ldots, a_{i-1}$ 的线性组合:

$$\begin{aligned} q_2 &= \frac{a_2 - \left(q_1^T a_2\right) q_1}{\|\tilde{q}_2\|_2} \\ &= \frac{a_2 - \left(q_1^T a_2\right) \frac{a_1}{\|a_1\|_2}}{\|\tilde{q}_2\|_2} \end{aligned}$$

$$q_3 = \frac{a_3 - \left(q_1^T a_3\right) q_1 - \left(q_2^T a_3\right) q_2}{\|\tilde{q}_3\|_2}$$

通过对 i 的归纳证明, 可得 q_i 是 $a_1, \ldots, a_i$ 的线性组合. ■

假设 Schmidt 正交法在第 j 次迭代提前终止:

Corollary 5.4.5 a_j 是 $q_1, \ldots, q_{j-1}$ 的一个线性组合.

$$a_j = \left(q_1^T a_j\right) q_1 + \cdots + \left(q_{j-1}^T a_j\right) q_{j-1}$$

Proof.

$$\widetilde{q}_i = a_i - \left(q_1^T a_i\right) q_1 - \cdots - \left(q_{i-1}^T a_i\right) q_{i-1}$$

$$0 = a_i - \left(q_1^T a_i\right) q_1 - \cdots - \left(q_{i-1}^T a_i\right) q_{i-1}$$

$$a_i = \|\tilde{q}_i\|_2 \, q_i + \left(q_1^T a_i\right) q_1 + \cdots + \left(q_{i-1}^T a_i\right) q_{i-1}$$

■

Corollary 5.4.6 a_j 是 $a_1, \ldots, a_{j-1}$ 的线性组合.

Proof. 每一个 $q_1, \ldots, q_{j-1}$ 都是 $a_1, \ldots, a_{j-1}$ 的线性组合.
因此 a_j 是 $a_1, \ldots, a_{j-1}$ 的线性组合. ■

II Matrices

6. Matrices

6.1 Matrices

Definition 6.1.1 — **矩阵.** 矩阵是一个由数字构成的矩阵数组.

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

上述矩阵大小 *(size)* 为 3×4, 矩阵的每一个元素 *(element)* 又称为系数 *(coefficient)*;

Notation 6.1. 设 B_{ij} 表示矩阵 B 中第 i 行第 j 列的元素
实数域中大小为 $m \times n$ 的矩阵集合写为 $\mathbb{R}^{m \times n}$
复数域中大小为 $m \times n$ 的矩阵集合写为 $\mathbb{C}^{m \times n}$

Definition 6.1.2 — **标量.** 不区分一个 1×1 矩阵和一个标量.

Definition 6.1.3 — **向量.** 不区分一个 $n \times 1$ 矩阵和一个向量.

Definition 6.1.4 — **行向量, 列向量.** 一个 $1 \times n$ 矩阵被称为一个行向量.
一个 $n \times 1$ 矩阵杯称为一个列向量.

Definition 6.1.5 — **高形, 宽形和方形矩阵.** 一个大小为 $m \times n$ 的矩阵为:

- 高的, 如果 $m > n$
- 宽的, 如果 $m < n$
- 方的, 如果 $m = n$

Definition 6.1.6 — **分块矩阵.** 分块矩阵的每一个元素都是一个矩阵.

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

其中 B, C, D, E 都是矩阵 (被称为矩阵 A 的子矩阵).

分块矩阵位于同一行的子矩阵行维度必须相等, 位于同一列的子矩阵列维度必须相等.

Definition 6.1.7 — 矩阵的列向量表示. 矩阵 $A \in \mathbb{R}^{m\times n}$, 可通过其列向量 (m-vector) 进行表示, 假设其列向量为 $a_1, \ldots, a_n \in \mathbb{R}^m$, 则有

$$A = \begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix}$$

Definition 6.1.8 — 矩阵的行向量表示. 矩阵 $A \in \mathbb{R}^{m\times n}$ 通过其行向量 $b_1, \ldots, b_m$ 进行表示

$$A = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}, b_i^T \in \mathbb{R}^n, i = 1, \cdots, m$$

6.2 矩阵运算

Definition 6.2.1 — 矩阵数乘. 设矩阵 $A \in \mathbb{R}^{m\times n}$

$$\beta A = \begin{bmatrix} \beta A_{11} & \beta A_{12} & \cdots & \beta A_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ \beta A_{m1} & \beta A_{m2} & \cdots & \beta A_{mn} \end{bmatrix}, \beta \in \mathbb{R}$$

(A and β can be real or complex.)

Definition 6.2.2 — 矩阵加法. 矩阵 $A, B \in \mathbb{R}^{m\times n}$ 的和为

$$A + B = \begin{bmatrix} A_{11} + B_{11} & A_{12} + B_{12} & \cdots & A_{1n} + B_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1} + B_{m1} & A_{m2} + B_{m2} & \cdots & A_{mn} + B_{mn} \end{bmatrix}$$

Definition 6.2.3 — Transpose. 矩阵 A 的转置表示为 A^T
若 $A \in \mathbb{R}^{m\times n}$, 则 $A^T \in \mathbb{R}^{n\times m}$, 其被定义为: $\left(A^T\right)_{ij} = A_{ji}, i = 1, \ldots, n; j = 1, \ldots, m$

转置将原矩阵的行向量转化为列向量.

Corollary 6.2.1 — 转置的性质. 有如下性质:

- $\left(A^T\right)^T = A$
- 对称矩阵满足 $A^T = A$
- A may be complex, but transpose of a complex matrix is rarely needed
- $(\beta A)^T = \beta A^T, (A + B)^T = A^T + B^T$

Definition 6.2.4 — 共轭转置. 矩阵 A 的共轭转置表示为 A^{H}

若 A $\in \mathbb{C}^{m\times n}$, 则 $A^H \in \mathbb{C}^{n\times m}$, 其被定义为: $\left(A^H\right)_{ij} = \bar{A}_{ji}, i = 1,\ldots,n; j = 1,\ldots,m;$

设矩阵 $A \in \mathbb{C}^{m\times n}$, 则其共轭转置为一个 $n \times m$ 矩阵

$$A^H = \begin{bmatrix} \bar{A}_{11} & \bar{A}_{21} & \cdots & \bar{A}_{m1} \\ \bar{A}_{21} & \bar{A}_{22} & \cdots & \bar{A}_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{A}_{1n} & \bar{A}_{2n} & \cdots & \bar{A}_{mn} \end{bmatrix}$$

Corollary 6.2.2 — 共轭转置的性质. 有如下性质:

- $\left(A^H\right)^H = A$
- Hermitian 矩阵满足 $A = A^H$
- $(\beta A)^H = \beta A^H, (A+B)^H = A^H + B^H$

Definition 6.2.5 — 矩阵乘法. 设矩阵 $A \in \mathbb{R}^{m\times p}, B \in \mathbb{R}^{p\times n}$, 那么矩阵 A 与 B 的乘积, 记作 C = AB , 矩阵 $C \in \mathbb{R}^{m\times n}$ 的第 i 行第 j 列元素 C_{ij}

$$\mathrm{C}_{ij} = \sum_{k=1}^{p} A_{ik}B_{kj}$$

矩阵 A 的列大小必须等于 B 的行大小.

Corollary 6.2.3 — 矩阵乘法性质. 有如下性质:

- 结合律: $(AB)C = A(BC)$
- 分配律: $A(B+C) = AB + AC$
- $$(AB)^T = B^T A^T, \quad (AB)^{\mathrm{H}} = B^H A^H$$
- 一般情况下 $AB \neq BA$
- 对于方阵 A 有, $IA = AI = A$

Definition 6.2.6 — 分块矩阵乘法.

$$\begin{bmatrix} A & B \\ \mathrm{C} & D \end{bmatrix} \begin{bmatrix} W & Y \\ X & Z \end{bmatrix} = \begin{bmatrix} AW + BX & AY + BZ \\ CW + DX & CY + DZ \end{bmatrix}$$

Definition 6.2.7 — 矩阵-向量乘积 Ax. 矩阵 $A \in \mathbb{R}^{m\times n}$ 和一个向量 $x \in \mathbb{R}^n$ 的积为

$$Ax = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 + \cdots + A_{1n}x_n \\ A_{21}x_1 + A_{22}x_2 + \cdots + A_{2n}x_n \\ \vdots \\ A_{m1}x_1 + A_{m2}x_2 + \cdots + A_{mn}x_n \end{bmatrix}$$

Corollary 6.2.4 Ax 是矩阵 A 列向量的线性组合.

$$Ax = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = x_1 a_1 + \cdots + x_n a_n$$

可以引出 A 的 Row Picture 和 Column Picture 的概念。

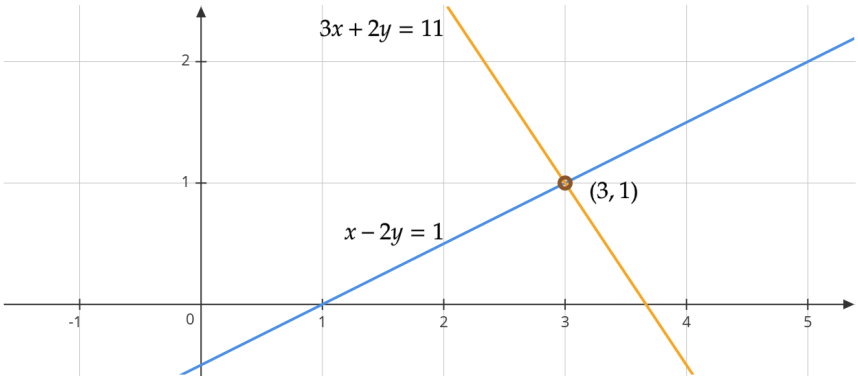
■ **Example 6.1**

$$\begin{cases} x - 2y = 1 \\ 3x + 2y = 11 \end{cases}$$

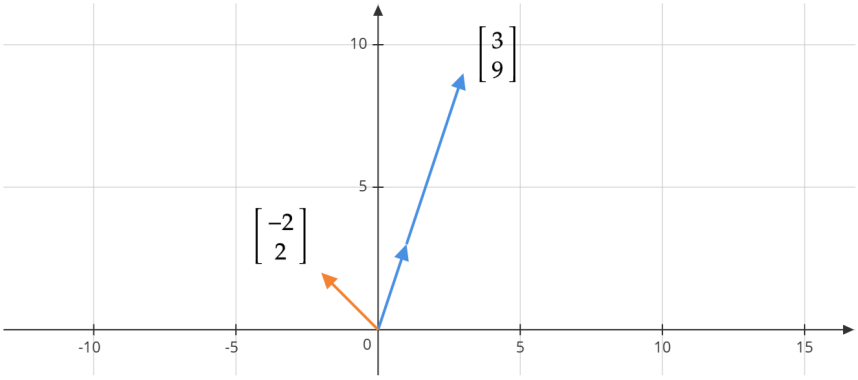
■

Figure 6.1: Row Picture and Column Picture for 6.1

(a) Row Picture



(b) Column Picture



$f : \mathbf{R}^n \to \mathbf{R}^m$ in terms of its effect on x.

signal processing/control interpretation: n inputs x_i, m outputs y_i. f is linear if we can represent its action on x as a product $f(x) = Ax$

Definition 6.2.8 — **矩阵-向量乘积函数** $f(x) = Ax$. 给定矩阵 $A \in \mathbb{R}^{m \times n}$，定义函数 $f : R^n \to R^m, f(x) = Ax$, 其中 $A = [f(e_1) \ldots f(e_n)]$.

Proof. 该函数为一个线性函数：

$$A(\alpha x + \beta y) = \alpha(Ax) + \beta(Ay)$$

任意一个线性函数都可以写成矩阵-向量乘积函数的形式

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

因此 $f(x) = Ax$, 其中 $A = [f\left(e_1\right) \ldots f\left(e_n\right)]$ ■

6.2.1 Matrix Power

Definition 6.2.9 — Matrix Power. It makes sense to multiply a square matrix A by itself to form AA. We refer to this matrix as A^2. Similarly, if k is a positive integer, then k copies of A multiplied together is denoted A^k. If k and l are positive integers, and A is square, then $A^k A^l = A^{k+l}$ and $\left(A^k\right)^l = A^{kl}$.

$$\left(A^{\ell+1}\right)_{ij} = \sum_{k=1}^{n} A_{ik} \left(A^{\ell}\right)_{kj}$$

By convention we take $A^0 = I$, which makes the formulas above hold for all nonnegative integer values of k and l.

▪ **Example 6.2 — Paths in a 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 6.3 — Linear dynamical system.**

$$x_{t+\ell} = A^{\ell} x_t$$

▪

6.2.2 矩阵乘法的算法复杂度

一般矩阵的乘法

$C = AB$ with A of size $m \times p$ and B of size $p \times n$

The product matrix C has size $m \times n$, so there are mn elements to compute.

The i, j element of C is the inner product of row i of A with column j of B. This is an inner product of vectors of length p and requires $2p - 1$ flops. Therefore the total is $mn(2p - 1)$ flops, which we approximate as $2mnp$ flops. $O(mnp)$

稀疏矩阵的乘法

Suppose that A is $m \times p$ and sparse, and B is $p \times n$, but not necessarily sparse.

The inner product of the i th row a_i^T of A with the j th column of B requires no more than $2 \operatorname{nnz}\left(a_i^T\right)$ flops.

Summing over $i = 1, \ldots, m$ and $j = 1, \ldots, n$ we get $2\mathbf{nnz}(A)n$ flops.

If B is sparse, the total number of flops is no more that $2\mathbf{nnz}(B)m$ flops.

Note that these formulas agree with the one given above, $2mnp$, when the sparse matrices have all entries nonzero.

三重矩阵相乘

$$D = ABC$$

with A of size $m \times n$, B of size $n \times p$, and C of size $p \times q$.

The matrix D can be computed in two ways, as $(AB)C$ and as $A(BC)$.

In the first method we start with AB($2mnp$ flops) and then form $D = (AB)C$($2mpq$ flops), for a total of $2mp(n+q)$ flops.

In the second method we compute the product BC (2npq flops) and then form $D = A(BC)$($2mnq$ flops), for a total of $2nq(m+p)$ flops.

You might guess that the total number of flops required is the same with the two methods, but it turns out it is not. The first method is less expensive when $2mp(n+q) < 2nq(m+p)$, i.e., when

$$\frac{1}{n} + \frac{1}{q} < \frac{1}{m} + \frac{1}{p}$$

■ **Example 6.4** As a more specific example, consider the product

$$ab^T c$$

where a, b, c are n vectors.

If we first evaluate the outer product ab^T, the cost is n^2 flops, and we need to store n^2 values. We then multiply the vector c by this $n \times n$ matrix, which costs $2n^2$ flops. The total cost is $3n^2$ flops.

If we first evaluate the inner product $b^T c$, the cost is $2n$ flops, and we only need to store one number (the result). Multiplying the vector a by this number costs n flops, so the total cost is $3n$ flops. For n large, there is a dramatic difference between $3n$ and $3n^2$ flops.

(The storage requirements are also dramatically different for the two methods of evaluating $ab^T c$: 1 number versus n^2 numbers.) ■

6.2.3 矩阵向量乘积复杂度

矩阵 $A \in \mathbb{R}^{m \times n}$ 和向量 $x \in \mathbb{R}^n$ 的乘积 y = Ax, 需要 (2n − 1)m flops;

乘积 $y \in \mathbb{R}^m$, 每个元素需要做向量内积, 需要 $2n - 1$ flops;

当 n 足够大时, 复杂度近似于 2mn;

特殊情况:

- A 为对角矩阵：n flops
- A 为下三角矩阵：n^2 flops
- A 为稀疏矩阵时：flops << $2mn$

6.3 Special Matrices and Matrices in Different Applications

Definition 6.3.1 — Zero Matrix. 所有元素都为 0 的矩阵.

记作

$$0, 0_{m\times n}$$

Definition 6.3.2 — **单位矩阵.** 为方形矩阵, 其中对角线元素为 1, 其它元素为 0.
记作 I 或者 I_n.

Corollary 6.3.1 I_n 的每一列是一个单位向量, 例如

$$\mathrm{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} e_1 & e_2 & e_3 \end{bmatrix}$$

Definition 6.3.3 — **Symmetric Matrices.**

$$A_{ij} = A_{ji}, A^T = A$$

Definition 6.3.4 — **Hermitian Matrices.** $A_{ij} = \bar{A}_{ji}, A^H = A$ (共轭复数).

diagonal elements are real (since $A_{ii} = \bar{A}_{ii}$).

Definition 6.3.5 — **Diagonal Matrices.** 对角线上元素不全为 0, 其余元素全为 0.

对角矩阵用于膨胀 (dilation).

Definition 6.3.6 — **下三角矩阵.** 方形矩阵且当 $i < j$ 时 $A_{ij} = 0$.

Definition 6.3.7 — **上三角矩阵.** 方形矩阵且当 $i > j$ 时 $A_{ij} = 0$

通过 Gram-Schmidt 正交化算法可以化为上三角或者下三角矩阵。

Definition 6.3.8 — **Sparse Matrices.** a matrix is sparse if most (almost all) of its elements are zero

sparse matrix storage formats and algorithms exploit sparsity, efficiency depends on number of nonzeros and their positions. The positions of nonzeros are visualized in a 'spy plot'.

6.3.1 $f(x) = Ax$ 中的 A

引入上节 $f(x) = Ax$(矩阵-向量乘积函数) 的概念,

■ **Example 6.5** — **Permutation Matrices.** f 颠倒向量 x 中的元素的顺序, 一个线性函数 $f(x) = Ax$

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

是一个置换矩阵. ■

Proof.

$$Ax = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_3 \\ x_2 \\ x_1 \end{bmatrix}$$

■

- **Example 6.6** f 对向量 x 中的元素进行升序排序, 非线性; ▪
- **Example 6.7** f 将向量 x 中的元素替换成相应的绝对值, 非线性; ▪
- **Example 6.8 — 反转矩阵 (Reverser matrix).**

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

▪

Proof.

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

■

- **Example 6.9 — 循环移位矩阵 (Circular shift matrix).**

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

▪

Proof.

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

■

- **Example 6.10 — 旋转矩阵.**

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

Ax 将向量 x 进行旋转, 角度为 θ. ▪

▪ **Example 6.11 — Reflection Matrices.** Suppose that y is the vector obtained by reflecting x through the line that passes through the origin, inclined θ radians with respect to horizontal. Then

$$y = \begin{bmatrix} \cos(2\theta) & \sin(2\theta) \\ \sin(2\theta) & -\cos(2\theta) \end{bmatrix} x$$

▪

6.3.2 Selectors

Definition 6.3.9 — Selector matrices.

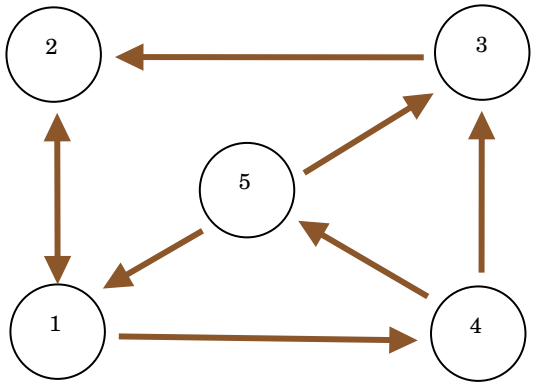
Definition 6.3.10 — Downsampling.

6.3.3 图论：节点弧关联矩阵

Definition 6.3.11 — 关联矩阵. 假设有向图 G 有 m 个顶点, n 条弧, 则关联矩阵 A 大小为 $m \times n$, 其中

$$A_{ij} = \begin{cases} 1 & \text{如果点}i\text{是弧}j\text{的终点} \\ -1 & \text{如果点}i\text{是弧}j\text{的起点} \\ 0 & \text{其它} \end{cases}$$

Figure 6.2: paths in directed graph



▪ **Example 6.12 — 路径矩阵.** matrix representation

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

$A_{ij} = 1$ indicates an edge $j \to i$

give a graph interpretation of $A^2 = AA, A^3 = AAA, \ldots$

$$A^2 = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 2 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad A^3 = \begin{bmatrix} 1 & 1 & 0 & 1 & 2 \\ 2 & 0 & 1 & 2 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

▪

6.3.4 Convolution

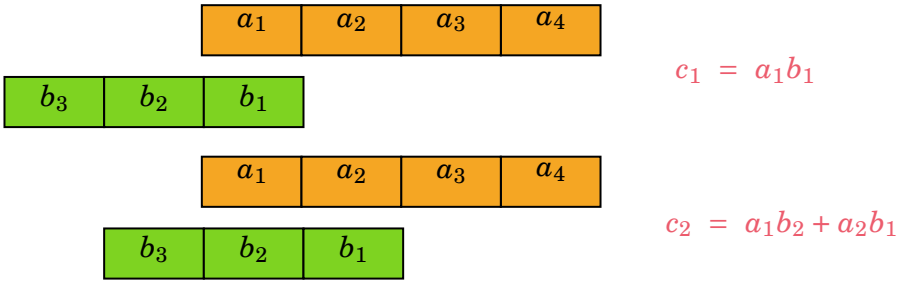
Definition 6.3.12 — 一维卷积. 向量 $a \in \mathbb{R}^n$ 和向量 $b \in \mathbb{R}^m$ 的卷积是一个 $(n+m-1)$ 维向量 $c \in \mathbb{R}^{m+n-1}$

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

记为 $c = a * b$

■ **Example 6.13** 设 $n = 4, m = 3$

Figure 6.3: An example of convolution



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

■

Corollary 6.3.2 假设向量 a 和 b 分别是以下多项式的系数

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

则 c = a* b 是多项式 $p(x)q(x)$ 的系数.

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

Corollary 6.3.3 — 卷积性质. 有如下性质:

- 对称性: $a * b = b * a$
- 结合律: $(a * b) * c = a * (b * c)$
- 如果 $a * b = 0$, 则 $a = 0$, 或者 $b = 0$

Corollary 6.3.4 如果固定 a 或 b, 则 $c = a * b$ 是一个线性函数

■ **Example 6.14 — Toeplitz Matrix.** 4 维向量 a 和 3 维向量 b , 则 $c = a * b$

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{bmatrix} = \begin{bmatrix} a_1 & 0 & 0 \\ a_2 & a_1 & 0 \\ a_3 & a_2 & a_1 \\ a_4 & a_3 & a_2 \\ 0 & a_4 & a_3 \\ 0 & 0 & a_4 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} b_1 & 0 & 0 & 0 \\ b_2 & b_1 & 0 & 0 \\ b_3 & b_2 & b_1 & 0 \\ 0 & b_3 & b_2 & b_1 \\ 0 & 0 & b_3 & b_2 \\ 0 & 0 & 0 & b_3 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}$$

■

■ **Example 6.15 — moving average of a time series.** n-vector x represents a time series. The *3-period moving average of the time series* is the time series

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

(with x_k interpreted as zero for $k < 1$ and $k > n$)
This can be expressed as a convolution $y = a * x$ with $a = (1/3, 1/3, 1/3)$. ■

6.3.5 多项式

Definition 6.3.13 — 多项式. 多项式 $p(t)$, 度为 $n-1$, 系数为 $x_1, x_2, \ldots, x_n$

$$p(t) = x_1 + x_2 t + x_3 t^2 + \cdots + x_n t^{n-1}$$

Definition 6.3.14 — Vandermonde Matrices. p(t) 在 m 个点中 $t_1, t_2, \ldots, t_m$ 的值为

$$\begin{bmatrix} p(t_1) \\ p(t_2) \\ \vdots \\ p(t_m) \end{bmatrix} = \begin{bmatrix} 1 & t_1 & \cdots & {t_1}^{n-1} \\ 1 & t_2 & \cdots & {t_2}^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & t_m & \cdots & {t_m}^{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = Ax$$

矩阵 A 被称为 *Vandermonde* 矩阵.

6.3.6 Fourier Transform

Definition 6.3.15 — Discrete Fourier Transform (DFT). DFT 将 n 维复向量 x 映射为 n 维复向量 y ($\mathbb{C}^n \to \mathbb{C}^n$)

$$y_k = \sum_{\ell=1}^{n} x_\ell e^{-i\frac{2\pi}{n}(k-1)(\ell-1)}, k = 1, \cdots, n$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \cdots & \omega^{-(n-1)} \\ 1 & \omega^{-2} & \omega^{-4} & \cdots & \omega^{-2(n-1)} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & \omega^{-(n-1)} & \omega^{-2(n-1)} & \cdots & \omega^{-(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$

其中 $\omega = e^{2\pi i/n}$.

DFT 矩阵 W 的第 k 行第 l 列的元素为 $W_{kl} = \omega^{-(k-1)(l-1)}$.

Definition 6.3.16 — Discrete Inverse Fourier Transform.

$$x_\ell = \frac{1}{n}\sum_{k=1}^{n} y_k e^{i\frac{2\pi}{n}(k-1)(\ell-1)}, \ell = 1, \cdots, n$$

6.3.7 Semi-Definite Matrices

Definition 6.3.17 — 半正定矩阵. 对称矩阵 $A \in \mathbb{R}^{n\times n}$ 称为半正定矩阵, 满足以下条件

$$x^T Ax \geq 0 \quad \forall x \in \mathbb{R}^n$$

Definition 6.3.18 — 正定矩阵. 对称矩阵 $A \in \mathbb{R}^{n\times n}$ 称为正定矩阵, 满足以下条件

$$x^T Ax > 0 \quad \forall x \neq 0$$

Definition 6.3.19 — 二次型. 如果对称矩阵 $A \in \mathbb{R}^{n\times n}$, 则 $x^T Ax$ 是二次型

Proof.

$$x^T Ax = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i A_{ij} x_j = \sum_{i=1}^{n} A_{ii}x_i^2 + 2\sum_{i>j} A_{ij}x_i x_j$$

■

▪ **Example 6.16**

$$A = \begin{bmatrix} 9 & 6 \\ 6 & a \end{bmatrix}$$

$$x^T Ax = 9x_1^2 + 12x_1x_2 + ax_2^2 = (3x_1 + 2x_2)^2 + (a-4)x_2^2$$

如果 $a > 4$, 矩阵 A 为正定矩阵:

$$x^T Ax > 0 \quad \forall x \neq 0$$

如果 $a = 4$, 矩阵 A 为半正定矩阵, 但不是正定矩阵:

$$x^T Ax \geq 0 \quad \forall x, \quad x^T Ax = 0 \quad \exists x = \begin{bmatrix} 2 \\ -3 \end{bmatrix}$$

如果 $a < 4$, 矩阵 A 不是半正定矩阵:

$$x^T Ax < 0 \quad \exists x = \begin{bmatrix} 2 \\ -3 \end{bmatrix}$$

▪

Corollary 6.3.5 正定矩阵 A 都是非奇异的.

Proof.

$$Ax = 0 \quad \Rightarrow \quad x^T Ax = 0 \quad \Rightarrow \quad x = 0$$

最后一步由正定性得到的.($x^T Ax > 0 \quad \forall x \neq 0$)

■

Theorem 6.3.6 — 正定矩阵对角元素性质. 正定矩阵 A 有正的对角元素.

$$A_{ii} = e_i^T A e_i > 0$$

Theorem 6.3.7 — 半正定矩阵对角元素性质. 每个半正定矩阵 A 都有非负的对角元素.

$$A_{ii} = e_i^T A e_i \geq 0$$

6.4 Gram 矩阵

Definition 6.4.1 — 实矩阵 A 的 Gram 矩阵.

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

Definition 6.4.2 — 复矩阵的 A 的 Gram 矩阵.

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

Theorem 6.4.1 每个 Gram 矩阵都是半正定的.

Proof.

$$x^T A x = x^T B^T B x = \|Bx\|_2^2 \geq 0, \forall x$$

■

Theorem 6.4.2 如果 Gram 矩阵是正定的, 则要满足

$$x^T A x = x^T B^T B x = \|Bx\|_2^2 > 0 (\forall x \neq 0)$$

Corollary 6.4.3 如果 Gram 矩阵是正定的, 则 B 的列向量是线性无关的.

Proof.

$$\|Bx\|_2^2 > 0 (\forall x \neq 0)$$

所以 $\forall x \neq 0, Bx \neq 0$.
注意和线性无关 5.1.2 的定义进行参照. ■

6.5 Affine functions and matrix-vector product

回想仿射函数、泰勒展开的定义。

for fixed $A \in \mathbf{R}^{m \times n}, b \in \mathbf{R}^m$, define a function $f : \mathbf{R}^n \to \mathbf{R}^m$ by

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

i.e., a matrix-vector product plus a constant

any function of this type is affine: if $\alpha + \beta = 1$ then

$$A(\alpha x + \beta y) + b = \alpha(Ax + b) + \beta(Ay + b)$$

every affine function can be written as $f(x) = Ax + b$ with:

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

and $b = f(0)$

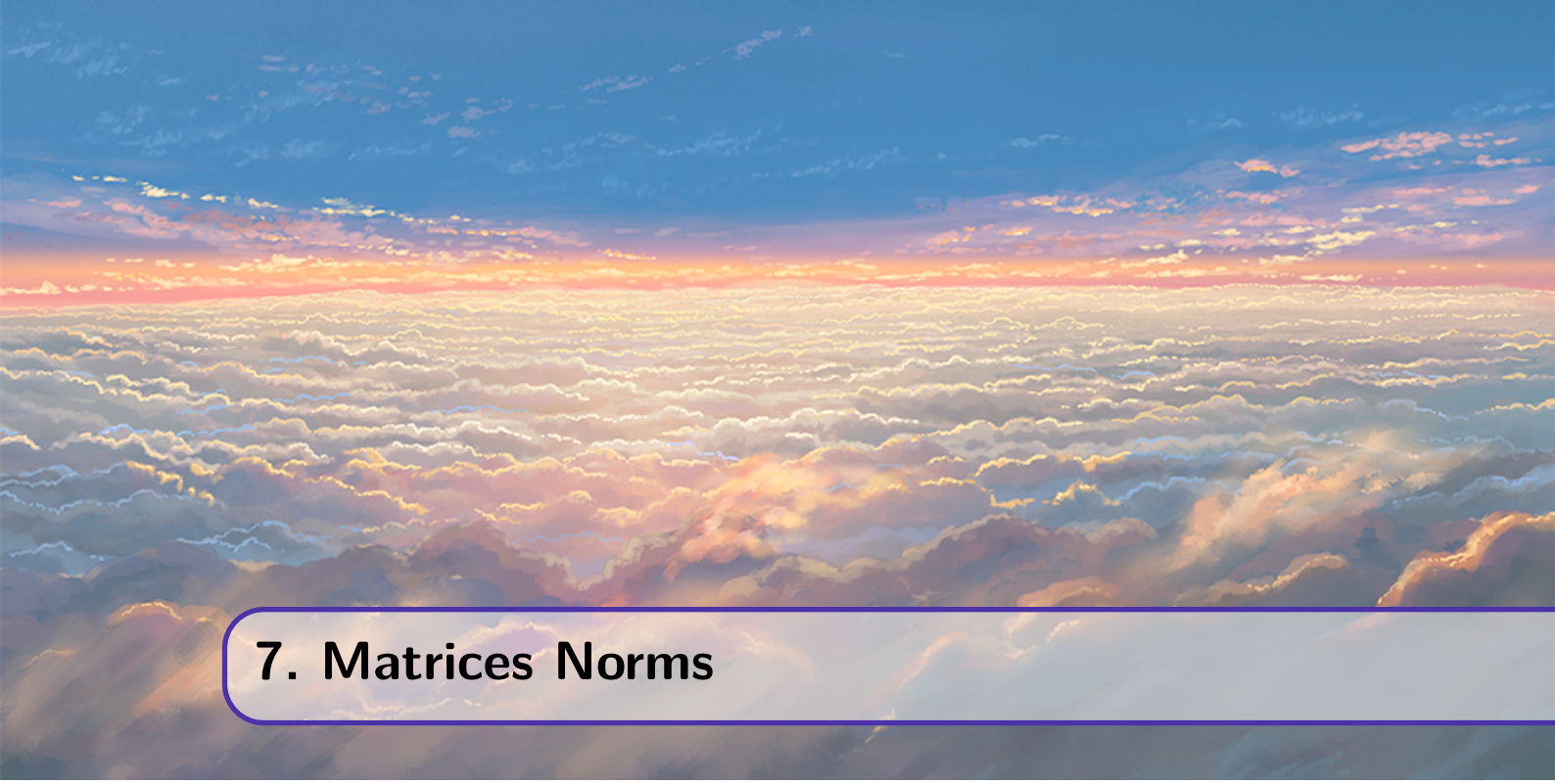
Theorem 6.5.1 first-order Taylor approximation of differentiable $f : \mathbf{R}^n \to \mathbf{R}^m$ around z :

$$\widehat{f}_i(x) = f_i(z) + \frac{\partial f_i}{\partial x_1}(z)\,(x_1 - z_1) + \cdots + \frac{\partial f_i}{\partial x_n}(z)\,(x_n - z_n), \quad i = 1, \ldots, m$$

in matrix-vector notation: $\widehat{f}(x) = f(z) + Df(z)(x - z)$ where

$$Df(z) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(z) & \frac{\partial f_1}{\partial x_2}(z) & \cdots & \frac{\partial f_1}{\partial x_n}(z) \\ \frac{\partial f_2}{\partial x_1}(z) & \frac{\partial f_2}{\partial x_2}(z) & \cdots & \frac{\partial f_2}{\partial x_n}(z) \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1}(z) & \frac{\partial f_m}{\partial x_2}(z) & \cdots & \frac{\partial f_m}{\partial x_n}(z) \end{bmatrix} = \begin{bmatrix} \nabla f_1(z)^T \\ \nabla f_2(z)^T \\ \vdots \\ \nabla f_m(z)^T \end{bmatrix}$$

$Df(z)$ is called the derivative matrix or Jacobian matrix of f at z, $\widehat{f}$ is a local affine approximation of f around z.



7. Matrices Norms

7.1 矩阵范数

Definition 7.1.1 — Matrix Norm. 向量空间中存在一个函数 $\|\cdot\|:\mathbb{R}^{m\times n}\to\mathbb{R}$ 且满足以下条件:

- 齐次性: $\|\alpha A\|=|\alpha|\|A\|, \alpha\in\mathbb{R}$ 且 $A\in\mathbb{R}^{m\times n}$;
- 三角不等式: $\|A+B\|\le\|A\|+\|B\|, A,B\in\mathbb{R}^{m\times n}$;
- 非负性: $\|A\|\ge 0, A\in\mathbb{R}^{m\times n}$ 且 $\|A\|=0\Leftrightarrow A=0$;

则称 $\|\cdot\|$ 为矩阵范数.

向量空间 $\mathbb{R}^{m\times n}$ 矩阵范数:

▪ **Example 7.1 — F-范数 (Frobenius norm).**

$$\|A\|_F=\left(\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ij}^2\right)^{\frac{1}{2}}$$

▪

Proof.

$$\|A\|_F\ge 0$$

$$\|\alpha A\|_F=|\alpha|\|A\|_F, \alpha\in\mathbb{R}$$

$$\begin{aligned}\|A+B\|_F=\left(\sum_{i=1}^{n}\sum_{j=1}^{n}(a_{ij}+b_{ij})^2\right)^{\frac{1}{2}}&\le\left(\sum_{i=1}^{n}\sum_{j=1}^{n}(a_{ij})^2\right)^{\frac{1}{2}}+\left(\sum_{i=1}^{n}\sum_{j=1}^{n}(b_{ij})^2\right)^{\frac{1}{2}}\\&=\|A\|_F+\|B\|_F\end{aligned}$$

■

Definition 7.1.2 — 从属于给定向量范数 $\|x\|_v$ 的矩阵范数. 设 $x \in \mathbb{R}^n, A \in \mathbb{R}^{m\times n}, \|\cdot\|_v$ 为一种向量范数. 则 $\frac{\|Ax\|_v}{\|x\|_v}$ 对所有 $x \neq 0$ 有最大值, 令

$$\|A\|_v = \max_{x\neq 0}\left\{\frac{\|Ax\|_v}{\|x\|_v}\right\} = \max_{x\neq 0}\left\{\left\|A\frac{x}{\|x\|_v}\right\|_v\right\} = \max_{\|y\|_v=1}\{\|Ay\|_v\}$$

即

$$\|A\|_v = \max_{x\neq 0}\left\{\frac{\|Ax\|_v}{\|x\|_v}\right\}$$

$\|A\|_v$ 称为从属于给定向量范数 $\|x\|_v$ 的矩阵范数, 简称为从属范数或算子范数.

Proof. 可以验证 $\|A\|_v$ 满足矩阵范数定义.

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

$$\|\alpha A\|_v = |\alpha|\|A\|_v, \alpha \in \mathbb{R}$$

$$\begin{aligned}\|A+B\|_v &= \max_{\|y\|_v=1}\|(A+B)y\|_v \\ &\leq \max_{\|y\|_v=1}\{\|Ay\|_v + \|By\|_v\} \\ &\leq \max_{\|y\|_v=1}\|Ay\|_v + \max_{\|y\|_v=1}\|By\|_v \\ &= \|A\|_v + \|B\|_v\end{aligned}$$

■

在本书中若未明确说明, $\|A\|$ 表示的是算子范数.

由定义 $\|A\|_v = \max_{x\neq 0}\left\{\frac{\|Ax\|_v}{\|x\|_v}\right\}$ 可得

Definition 7.1.3 — 向量范数和算子范数相容.

$$\frac{\|Ax\|_v}{\|x\|_v} \leq \|A\|_v \Rightarrow \|Ax\|_v \leq \|A\|_v\|x\|_v$$

称向量范数和算子范数相容.

Theorem 7.1.1 — 算子范数服从乘法范数相容性. 对于 $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{n\times p}$

$$\begin{aligned}\|AB\|_v &= \max_{x\neq 0}\left\{\frac{\|ABx\|_v}{\|x\|_v}\right\} \\ &\leq \max_{x\neq 0}\left\{\frac{\|A\|_v\|Bx\|_v}{\|x\|_v}\right\} \\ &\leq \|A\|_v \max_{x\neq 0}\left\{\frac{\|B\|_v\|x\|_v}{\|x\|_v}\right\} \\ &= \|A\|_v\|B\|_v\end{aligned}$$

算子范数服从乘法范数相容性.

根据向量的常用范数可以导出矩阵 $A \in \mathbb{R}^{m\times n}$ 的算子范数

Definition 7.1.4 — A 的列范数.

$$\|A\|_1 = \max_{x\neq 0}\left(\frac{\|Ax\|_1}{\|x\|_1}\right) = \max_{1\le j\le n}\sum_{i=1}^{m}|a_{ij}|$$

Definition 7.1.5 — A 的行范数.

$$\|A\|_\infty = \max_{x\neq 0}\left(\frac{\|Ax\|_\infty}{\|x\|_\infty}\right) = \max_{1\le i\le m}\sum_{j=1}^{n}|a_{ij}|$$

Definition 7.1.6 — A 的 2-范数.

$$\|A\|_2 = \max_{x\neq 0}\left(\frac{\|Ax\|_2}{\|x\|_2}\right) = \sqrt{\lambda_{\max}\left(A^TA\right)} \tag{7.1}$$

Proof. For any A Choose x to be the eigenvector of $A^{\mathrm{T}}A$ with largest eigenvalue $\lambda_{\max}$. The ratio in equation 7.1 is $\boldsymbol{x}^{\mathrm{T}}A^{\mathrm{T}}A\boldsymbol{x} = \boldsymbol{x}^{\mathrm{T}}\left(\lambda_{\max}\right)\boldsymbol{x}$ divided by $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}$. This is $\lambda_{\max}$.

No $\boldsymbol{x}$ can give a larger ratio. The symmetric matrix $A^{\mathrm{T}}A$ has eigenvalues $\lambda_1,\ldots,\lambda_n$ and orthonormal eigenvectors $q_1,\boldsymbol{q}_2,\ldots,\boldsymbol{q}_n$. Every $\boldsymbol{x}$ is a combination of those vectors. Try this combination in the ratio and remember that $\boldsymbol{q}_i^{\mathrm{T}}\boldsymbol{q}_j = 0$:

$$\frac{\boldsymbol{x}^{\mathrm{T}}A^{\mathrm{T}}A\boldsymbol{x}}{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}} = \frac{\left(c_1\boldsymbol{q}_1+\cdots+c_n\boldsymbol{q}_n\right)^{\mathrm{T}}\left(c_1\lambda_1\boldsymbol{q}_1+\cdots+c_n\lambda_n\boldsymbol{q}_n\right)}{\left(c_1\boldsymbol{q}_1+\cdots+c_n\boldsymbol{q}_n\right)^{\mathrm{T}}\left(c_1\boldsymbol{q}_1+\cdots+c_n\boldsymbol{q}_n\right)} = \frac{c_1^2\lambda_1+\cdots+c_n^2\lambda_n}{c_1^2+\cdots+c_n^2}$$

The maximum ratio $\lambda_{\max}$ is when all c 's are zero, except the one that multiplies $\lambda_{\max}$. ■

The ratio in equation 7.1 is the Rayleigh quotient for the symmetric matrix $A^{\mathrm{T}}A$. Its maximum is the largest eigenvalue $\lambda_{\max}\left(A^{\mathrm{T}}A\right)$. The minimum ratio is $\lambda_{\min}\left(A^{\mathrm{T}}A\right)$. If you substitute any vector $\boldsymbol{x}$ into the Rayleigh quotient $\boldsymbol{x}^{\mathrm{T}}A^{\mathrm{T}}A\boldsymbol{x}/\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}$, you are guaranteed to get a number between $\lambda_{\min}\left(A^{\mathrm{T}}A\right)$ and $\lambda_{\max}\left(A^{\mathrm{T}}A\right)$.

The norm $\|A\|$ equals the largest singular value $\sigma_{\max}$ of A. The singular values $\sigma_1,\ldots,\sigma_r$ are the square roots of the positive eigenvalues of $A^{\mathrm{T}}A$. So certainly $\sigma_{\max} = (\lambda_{\max})^{1/2}$. Since U and V are orthogonal in $A = U\Sigma V^{\mathrm{T}}$, the norm is $\|\boldsymbol{A}\| = \sigma_{\max}$.

■ **Example 7.2** 求矩阵 A 的各种常用范数

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

$$\|A\|_1 = \max_{1\le j\le n} \sum_{i=1}^{n} \left|a_{ij}\right| = \max_{1\le j\le n} \{2,5,2\} = 5$$

$$\|A\|_\infty = \max_{1\le i\le n} \sum_{j=1}^{n} \left|a_{ij}\right| = \max_{1\le i\le n} \{3,4,2\} = 4$$

由于 $\|A\|_2 = \sqrt{\lambda_{\max}\left(A^T A\right)}$, 因此先求 $A^T A$ 的特征值

$$A^T A = \begin{pmatrix} 1 & -1 & 0 \\ 2 & 2 & 1 \\ 0 & -1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 & 0 \\ -1 & 2 & -1 \\ 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 1 \\ 0 & 9 & -1 \\ 1 & -1 & 2 \end{pmatrix}$$

特征方程为

$$\det\left(\lambda I - A^T A\right) = \begin{vmatrix} \lambda - 2 & 0 & -1 \\ 0 & \lambda - 9 & 1 \\ -1 & 1 & \lambda - 2 \end{vmatrix} = 0$$

可得 $A^T A$ 的特征值

$$\lambda_1 = 9.1428, \lambda_2 = 2.9211, \lambda_3 = 0.9361$$

■

对于 $\|A\|_2$ 需要计算 $\lambda_{\max}\left(A^T A\right)$, 直接根据特征方程计算特征值的算法复杂度太高.

8. 适定问题

8.1 The Definition of Well-posed Problem

In 1923, the French mathematician Hadamard introduced the notion of well-posed (适定) problem:

- A solution for the problem exists;
- The solution is unique;
- Perturbations in the data should cause small perturbations in the solution.

One of these conditions is not satisfied, the problem is said to be ill-posed (病态) and demands a special consideration.

▪ **Example 8.1** 假设 A 是非奇异矩阵

$$Ax = b$$

如果将 b 为 $b + \Delta b$, 方程新的解 $x + \Delta x$, 则有:

$$A(x + \Delta x) = b + \Delta b$$

即

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

如果小的变化 Δb 导致小变化 Δx, 则称解是稳定的. 如果小的变化 Δb 导致大变化 Δx, 则称解不稳定的.

设

$$A = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 + 10^{-10} & 1 - 10^{-10} \end{bmatrix}, \quad A^{-1} = \begin{bmatrix} 1 - 10^{10} & 10^{10} \\ 1 + 10^{10} & -10^{10} \end{bmatrix}$$

若 $b = (1, 1)$, 方程 Ax 的解 $x = (1, 1)$. 如果将 b 改为 $b + \Delta b$, 那么 x 的变化量为

$$\Delta x = A^{-1}\Delta b = \begin{bmatrix} \Delta b_1 - 10^{10}\,(\Delta b_1 - \Delta b_2) \\ \Delta b_1 + 10^{10}\,(\Delta b_1 - \Delta b_2) \end{bmatrix}$$

很小变化 Δb 会导致非常大变化 Δx! 由矩阵 A 定义的问题, 称为适定问题或病态问题.

▪

8.2 绝对误差的界限

假设 A 是非奇异的, 并给出定义:

Notation 8.1.

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

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

$\|\Delta x\|$ 的上界为:

$$\|\Delta x\|_2 \le \left\|A^{-1}\right\|_2 \|\Delta b\|_2$$

矩阵范数 $\left\|A^{-1}\right\|_2$ 小时, 当 $\|\Delta b\|_2$ 变化很小, $\|\Delta x\|_2$ 也很小; $\left\|A^{-1}\right\|_2$ 大时, $\|\Delta x\|_2$ 可能很大, 即使 $\|\Delta b\|_2$ 很小.

8.3 相对误差的界限[1]

假设 $b \neq 0$; 因此 $x \neq 0$

$\|\Delta x\|_2/\|x\|_2$ 的上界为:

$$\|\Delta x\|_2 = \left\|A^{-1}\Delta b\right\|_2 \le \left\|A^{-1}\right\|_2 \|\Delta b\|_2 (\text{向量范数和算子范数相容})$$

$$\Rightarrow \frac{\|\Delta x\|_2}{\|x\|_2} \le \frac{\left\|A^{-1}\right\|_2 \|\Delta b\|_2}{\|x\|_2} = \frac{\|A\|_2 \left\|A^{-1}\right\|_2 \|\Delta b\|_2}{\|x\|_2\|A\|_2} \le \frac{\|A\|_2 \left\|A^{-1}\right\|_2 \|\Delta b\|_2}{\|b\|_2}$$

由 $\|b\|_2 = \|Ax\|_2 \le \|A\|_2\|x\|_2$, 可得

$$\frac{\|\Delta x\|_2}{\|x\|_2} \le \|A\|_2 \left\|A^{-1}\right\|_2 \frac{\|\Delta b\|_2}{\|b\|_2}$$

$\|A\|_2 \left\|A^{-1}\right\|_2$ 小，当 $\frac{\|\Delta b\|_2}{\|b\|_2}$ 相对变化很小时，$\frac{\|\Delta x\|_2}{\|x\|_2}$ 也变化很小;
$\|A\|_2 \left\|A^{-1}\right\|_2$ 大, $\frac{\|\Delta x\|_2}{\|x\|_2}$ 可远远大于 $\frac{\|\Delta b\|_2}{\|b\|_2}$.

Definition 8.3.1 — 非奇异矩阵 A 的条件数 (condition number) . 条件数定义

$$\kappa(A) = \|A\|_2 \left\|A^{-1}\right\|_2$$

Corollary 8.3.1 — 非奇异矩阵 A 的条件数 (condition number) 性质. 有如下性质:

- 对于所有 A, 有 $\kappa(A) \ge 1$;
- 如果 $\kappa(A)$ 比较小 (接近 1), x 的相对误差接近 b 的相对误差;
- 如果 $\kappa(A)$ 比较大 (超过 100) , x 的相对误差比 b 的相对误差大得多.

8.4 Cancellation

Instability in an algorithm is often (but not always) caused by an effect called cancellation. Cancellation occurs when two numbers are subtracted that are almost equal, and one of the numbers or both are subject to error (for example, due to rounding error in previous calculations). Suppose

$$\hat{x} = x + \Delta x, \quad \hat{y} = y + \Delta y$$

[1] Reference: MathWorks Blog.

are approximations of two numbers x, y, with absolute errors $|\Delta x|$ and $|\Delta y|$, respectively. The relative error in the difference $\hat{x} - \hat{y}$ is

$$\frac{|(\hat{x} - \hat{y}) - (x - y)|}{|x - y|} = \frac{|\Delta x - \Delta y|}{|x - y|} \leq \frac{|\Delta x| + |\Delta y|}{|x - y|}.$$

(The upper bound is achieved when Δx and Δy have opposite signs.) We see that if $x - y$ is small, then the relative error in $\hat{x} - \hat{y}$ can be very large, and much larger than the relative errors in $\hat{x}$ and $\hat{y}$. The result is that the relative errors $|\Delta x|/|x|$, $|\Delta y|/|y|$ are magnified enormously.

For example, suppose $x = 1, y = 1 + 10^{-5}$, and x and y have been calculated with an accuracy of about 10 significant digits, i.e., $|\Delta x|/|x| \approx 10^{-10}$ and $|\Delta y|/|y| \approx 10^{-10}$. The error in the result is

$$\frac{|(\hat{x} - \hat{y}) - (x - y)|}{|x - y|} \leq \frac{|\Delta x| + |\Delta y|}{|x - y|} \approx \frac{2 \cdot 10^{-10}}{|x - y|} = 2 \cdot 10^{-5}.$$

The result has only about 5 correct digits.

■ **Example 8.2** Example The most straightforward method for computing the two roots of the quadratic equation

$$ax^2 + bx + c = 0$$

(with $a \neq 0$) is to evaluate the expressions

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}, \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}.$$

This method is unstable if $b^2 \gg |4ac|$. If $b > 0$, there is a danger of cancellation in the expression for x_1; if $b < 0$, cancellation may occur in the expression for x_2. For example, suppose $a = c = 1, b = 10^5 + 10^{-5}$. The exact roots are given by

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} = -10^{-5}, \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} = -10^5,$$

We evaluate these expressions in MATLAB, rounding the square roots to 6 correct digits using the

```
chop
```

function:

```
>> a = 1; b = 1e5 + 1e-5; c = 1;
>> x1 = (-b + chop(sqrt(b^2 - 4*a*c), 6)) / (2*a)
ans =
-5.0000e-6
>> x2 = (-b - chop(sqrt(b^2 - 4*a*c), 6)) / (2*a)
ans =
-1.0000e+05
```

The relative error in x_1 is 50%, and is due to cancellation. We can formulate an algorithm that is more stable if $b^2 \gg |4ac|$ as follows. First suppose $b > 0$, so we have

cancellation in the expression for x_1. In this case we can calculate x_2 accurately. The expression for x_1 can be reformulated as

$$\begin{aligned} x_1 &= \frac{\left(-b+\sqrt{b^2-4ac}\right)\left(-b-\sqrt{b^2-4ac}\right)}{(2a)\left(-b-\sqrt{b^2-4ac}\right)} \\ &= \frac{b^2-b^2+4ac}{(2a)\left(-b-\sqrt{b^2-4ac}\right)} \\ &= \frac{c}{ax_2} \end{aligned}$$

Similarly, if $b > 0$, we can use the expression $x_2 = c/(ax_1)$ to compute x_2, given x_1. The modified algorithm that avoids cancellation is therefore: - if $b \leq 0$, calculate

$$x_1 = \frac{-b+\sqrt{b^2-4ac}}{2a}, \quad x_2 = \frac{c}{ax_1}$$

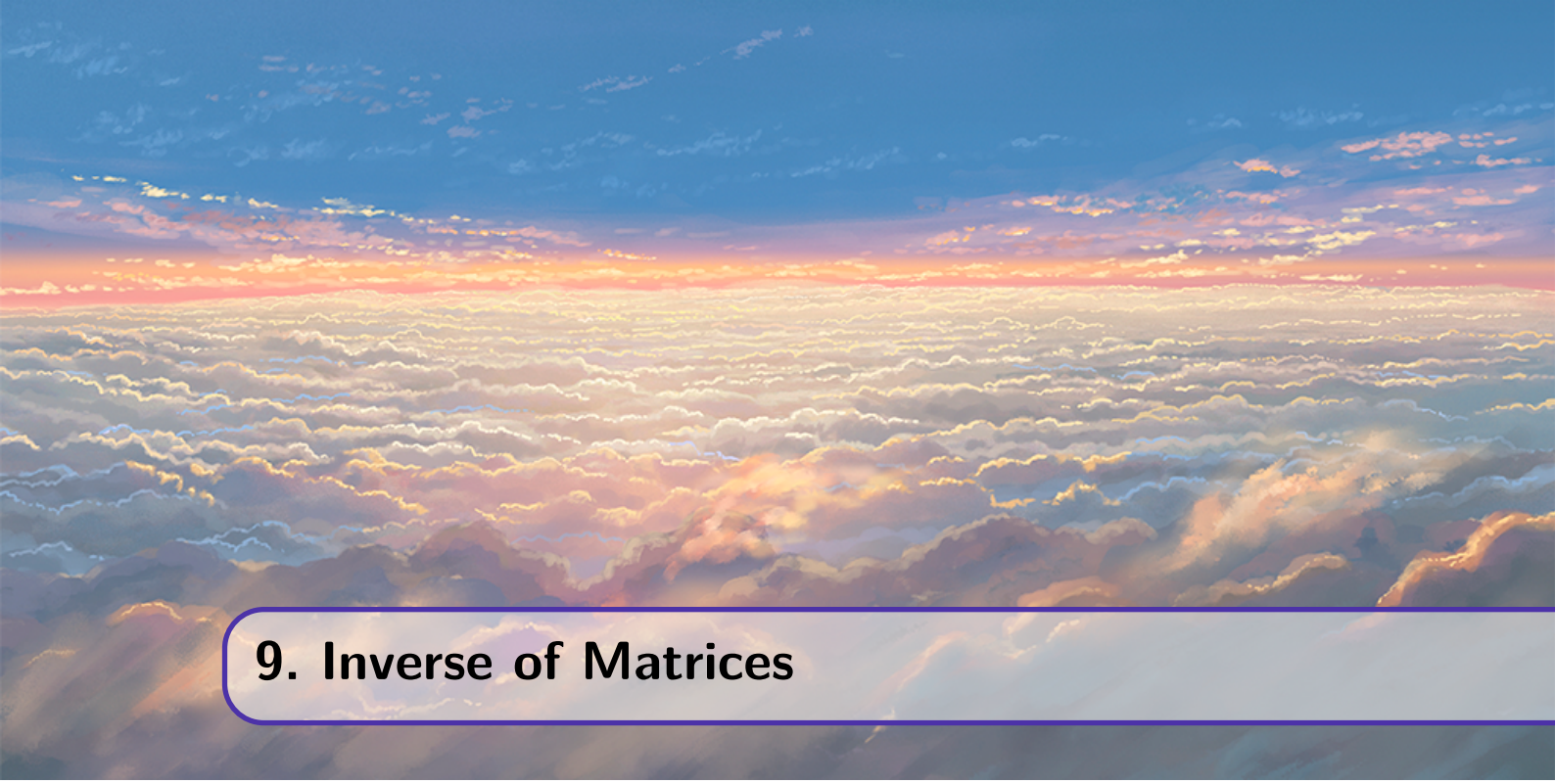
- if $b > 0$, calculate

$$x_2 = \frac{-b-\sqrt{b^2-4ac}}{2a}, \quad x_1 = \frac{c}{ax_2}$$

For the example, we get

```
    >> a = 1; b = 1e5 + 1e-5; c = 1;
>> x2 = (-b - chop(sqrt(b^2 - 4*a*c), 6)) / (2*a)
ans =
-1.0000e+05
>> x1 = c / (a*x2)
ans =
-1.0000e-05
```

∎



9. Inverse of Matrices

9.1 Left Inverse, Right Inverse, Inverse

Definition 9.1.1 — A **的左逆**. 当一个矩阵 X 满足

$$XA = I$$

X 被称为 A 的*左逆*; 当左逆存在时，则称 A 是*可左逆*的;

如果左逆矩阵存在, 则左逆矩阵有**无穷多**个.

- **Example 9.1**

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

矩阵 A 是可左逆的，其左逆矩阵有两个

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

■

Definition 9.1.2 — A **的右逆**. 当左逆存在时，则称 A 是可左逆的;

如果右逆矩阵存在, 则右逆矩阵有**无穷多**个.

- **Example 9.2**

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

矩阵 B 可右逆，以下矩阵都是 B 的右逆

$$D=\frac{1}{2}\begin{bmatrix}1&-1\\-1&1\\1&1\end{bmatrix},E=\begin{bmatrix}1&0\\0&1\\0&0\end{bmatrix},G=\begin{bmatrix}1&-1\\0&0\\0&1\end{bmatrix}$$

■

Theorem 9.1.1 一个大小为 $m\times n$ 的矩阵, 其左逆或右逆的维度为 $n\times m$.

Theorem 9.1.2 A 的左逆为 X 当且仅当 X^T 是 A^T 的右逆.

Proof.

$$A^TX^T=(XA)^T=I$$

■

Theorem 9.1.3 A 的右逆为 X 当且仅当 X^T 是 A^T 的左逆.

Proof.

$$X^TA^T=(AX)^{\mathrm{T}}=I$$

■

Theorem 9.1.4 如果矩阵 A 存在左逆和右逆，则左逆和右逆一定相等

Proof.

$$\begin{aligned}&XA=I,AY=I\\\Rightarrow&X=XI=X(AY)=(XA)Y=Y\\\Rightarrow&X=Y\end{aligned}$$

■

Definition 9.1.3 — **逆 A^{-1}.** 如果矩阵 A 存在左逆和右逆, 此时 X 称为矩阵的*逆*，记作 A^{-1} 当矩阵的逆存在时，则称矩阵 A *可逆*.

9.2 Linear Equation Systems

Definition 9.2.1 — **Linear Equation Systems.** 有 n 个变量的 m 个方程为

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

写成矩阵形式为：Ax = b . 其中 A 为系数矩阵, x 为 n 维列向量.

该方程组可能**无解**，**有唯一解**和**无穷解**.

9.2.1 线性方程组求解

Theorem 9.2.1 如果矩阵 A 可左逆，假设 X 是矩阵 A 的左逆，则**至多**一个解, 如有解则 $x = Xb$.

Proof.

$$Ax = b \Rightarrow x = XAx = Xb$$

列满秩时 (下面证明), 列向量线性无关, 所以其零空间中只有零解，方程 $Ax = b$ 可能有一个唯一解 (b 在 A 的列空间中, 此特解就是全部解, 因为通常的特解可以通过零空间中的向量扩展出一组解集，而此时零空间只有 0 向量) , 也可能无解 (b 不在 A 的列空间中) . ■

Theorem 9.2.2 如果矩阵 A 可右逆，假设 Y 是矩阵 A 的右逆，则**至少**一个解, 即 $x = \mathrm{Y}b$.

Proof. 设 $x = Yb$

$$x = Yb \Rightarrow Ax = AYb = b$$

右逆就是研究 $m \times n$ 矩阵 A 行满秩的情况, 此时 $\mathrm{n} > \mathrm{m} = \mathrm{rank(A)}$. 对称的, 其左零空间中仅有零向量，即没有行向量的线性组合能够得到零向量. ($N(A^T) = \{0\}$) ■

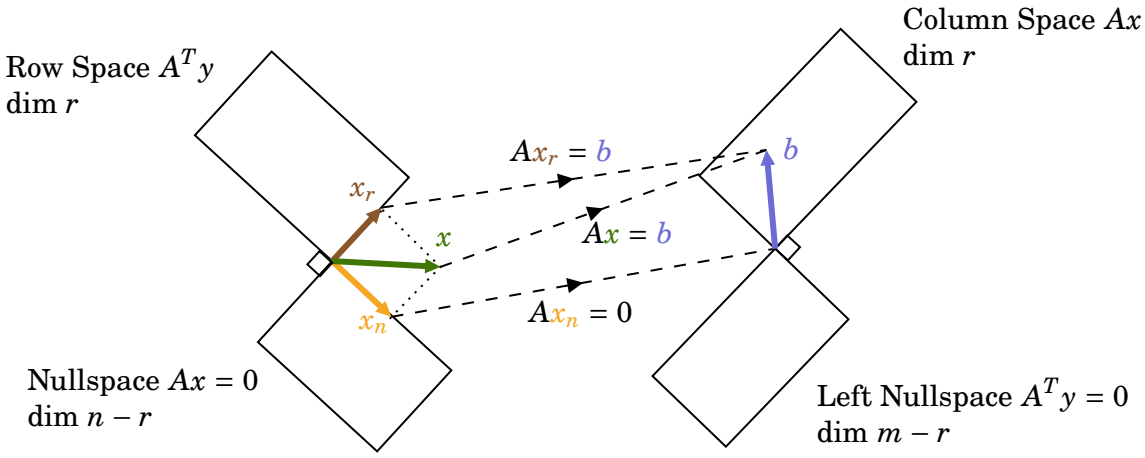
Theorem 9.2.3 如果矩阵 A 可逆的，假设 X 是矩阵 A 的逆，则

$$Ax = b \Rightarrow x = A^{-1}b$$

唯一解.

9.3 Fundamental Theorem of Linear Algebra

Figure 9.1: Four Subspace of Matrix A



The set of linear equations is called *over-determined* if $m > n$, *under-determined* if $m \leq n$, and *square* if $m = n$.

A set of equations with zero right-hand side, $Ax = 0$, is called a *homogeneous* set of equations. Any homogeneous set of equations has $x = 0$ as a solution.

$\boldsymbol{r = m}$	$\boldsymbol{r = n}$	Square and invertible	$\boldsymbol{Ax = b}$ has 1 solution
$\boldsymbol{r = m}$	$r < n$	Short and wide	$\boldsymbol{Ax = b}$ has ∞ solutions
$r < m$	$\boldsymbol{r = n}$	Tall and thin	$\boldsymbol{Ax = b}$ has 0 or 1 solution
$r < m$	$r < n$	Not full rank	$\boldsymbol{Ax = b}$ has 0 or ∞ solutions

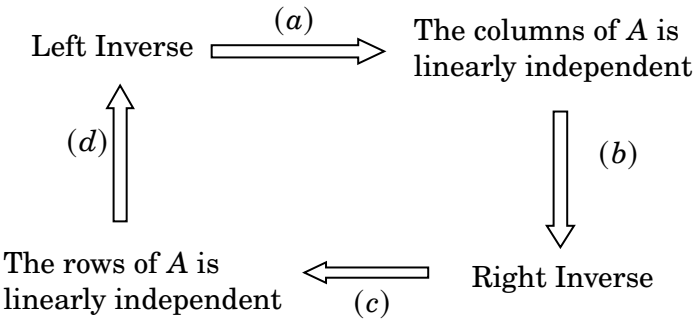
9.4 Invertible Matrices

Theorem 9.4.1 对于方阵 $A \in \mathbb{R}^{n\times n}$，以下条件都是等价的:

1. A 可左逆
2. A 的列向量线性无关
3. A 可右逆
4. A 的行向量线性无关
5. A 可逆

此时矩阵 A 为非奇异矩阵，由条件 1 与 3，可得 A 为可逆矩阵.

Proof. 可以通过以下方式证明:



- 性质 (a) 对任意矩阵 $A \in \mathbb{R}^{m\times n}$ 都成立
- 性质 (b) 对方阵矩阵 $A \in \mathbb{R}^{n\times n}$ 都成立
- 对于性质 (c) 与 (d), 可利用 A^T 证明

■

Theorem 9.4.2 (a): A 可左逆，则 A 列向量线性无关.

Proof. 假设 A 的左逆是 B，则

$$\begin{aligned} & Ax = 0 \\ \Rightarrow & BAx = 0 \\ \Rightarrow & Ix = 0 \end{aligned}$$

假设 A 的列向量 $A = [a_1, a_2, \cdots, a_n]$

$$Ax = x_1a_1 + x_2a_2 + \cdots + x_na_n = 0$$

则当该等式 $Ax = 0$ 成立时，其解 $x = 0$, 则 A 的列向量线性无关.

Corollary 9.4.3 如果 $A \in \mathbb{R}^{m\times n}$ 有左逆，则有 $m \geq n = r$.

即 A 是高或方的矩阵, 如 $A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$.

此时 A 的行向量可能线性相关，而 A 的列向量线性无关. $N(A) = \{0\}$.

假设 A 的列向量 $A = [a_1, a_2, \cdots, a_n]$

$$Ax = x_1a_1 + x_2a_2 + \cdots + x_na_n = b$$
$$Ay = y_1a_1 + y_2a_2 + \cdots + y_na_n = b$$

$$Ax - Ay = A(x - y) = 0 \Rightarrow x = y$$

当 $b \in \mathbb{R}^m, b \notin \{y \mid y = Ax, x \in \mathbb{R}^n\}$ 时（即 b 不在 A 的列空间，$m \geq n$ 时），线性方程组无解. $Ax = b$ 至多一个解，如有解则 $x = Xb$. ■

Theorem 9.4.4 矩阵的行秩等于列秩.

Proof. 令 A 是一个 $m \times n$ 的矩阵，其列秩为 r. 因此矩阵 A 的列空间的维度是 r .

令 $c_1, c_2, \ldots, c_r$ 是 A 的列空间的一组基,构成 $m \times r$ 矩阵 C 的列向量 $C = [c_1, c_2, \ldots, c_r]$，并使得 A 的每个列向量是 C 的 r 个列向量的线性组合.

由矩阵乘法的定义，存在一个 $r \times n$ 矩阵 R, 使得 $A = CR$. (A 的 (i, j) 元素是 c_i 与 R 的第 j 个行向量的点积.)

现在，由于 $A = CR$, A 的每个行向量是 R 的行向量的线性组合，这意味着 A 的行向量空间被包含于 R 的行向量空间之中. 因此 A的行秩 $\leq R$的行秩. 但 R 仅有 r 行, 所以 R的行秩 $\leq r = A$的列秩. 这就证明了 A的行秩 $\leq A$的列秩.

把上述证明过程中的“行”与“列”交换，利用对偶性质同样可证 A的列秩 $\leq A$的行秩. 更简单的方法是考虑 A 的转置矩阵 A^{T}, 则 A的列秩 $= A^{\mathrm{T}}$的行秩 $\leq A^{\mathrm{T}}$的列秩 $= A$的行秩. 这证明了 A 的列秩等于 A 的行秩. 证毕. ■

Theorem 9.4.5 (c): 矩阵 $A \in \mathbb{R}^{m \times n}$ 有右逆 X, 则 A 行向量线性无关.

Proof.

$$\mathrm{X}^T A^T = (AX)^T = I$$

则有 X^T 是 A^T 的左逆, A^T 的列向量线性无关.

即 $A^T \in \mathbb{R}^{n \times m}$.

Corollary 9.4.6 如果 $A \in \mathbb{R}^{m \times n}$ 有左逆，则有 $r = m \leq n$.

即 A 是宽或方的矩阵.

此时 A 的列向量可能线性相关，而 A 的行向量线性无关.

$N(A^T) = \{0\}, \dim N(A) = n - r, r = m$. ($Ax = b$ 有无穷解)

根据定理“矩阵的行秩等于列秩”，A^T 的列向量线性无关，则矩阵 A 有 m 个线性无关列向量（行向量），即通过 Gram-Schmidt 正交化可得 m 个正交基.

$\forall b \in \mathbb{R}^m$, 有 $b \in \{y \mid y = Ax, x \in \mathbb{R}^n\}$ $(m \leq n)$, 方程 $Ax = b$ 有解，其解为 $x = Xb$. ■

Theorem 9.4.7 (b): 若方阵 A 列向量线性无关，则 A 可右逆.

Proof. 假设 $A \in \mathbb{R}^{n \times n}$ 为方阵且列向量线性无关

$$A = [a_1, a_2, \cdots, a_n]$$

则对于任意向量 b $\in \mathbb{R}^n$, 则向量组 $[a_1, a_2, \ldots, a_n,$ b] 线性相关，存在不全为 o 的系数，使得以下等式成立

$$x_1a_1 + x_2a_2 + \cdots + x_na_n + x_{n+1}b = 0$$

因为 A 列向量线性无关，则 $x_{n+1} \neq 0$（假设 $x_{n+1} = 0$ 会推出违反线性无关假设的结论），即 b 是 A 列向量的线性组合;

$$b = -\frac{x_1}{x_{n+1}}a_1 - \frac{x_2}{x_{n+1}}a_2 - \cdots - \frac{x_n}{x_{n+1}}a_n$$

存在向量 $c_1, \ldots, c_n \in \mathbb{R}^n$, 使得

$$\begin{aligned} Ac_1 &= e_1 \\ Ac_2 &= e_2 \\ &\cdots \\ Ac_n &= e_n \end{aligned}$$

则矩阵 $C = [c_1 c_2 \ldots c_n]$ 是矩阵 A 的右逆, $AC = I$.

■

9.5 转置和共轭转置的逆

Theorem 9.5.1 — 转置 A^T 和共轭转置 A^{H}. 如果矩阵 A 为非奇异矩阵，则其转置 A^T 和共轭转置 A^{H} 都为非奇异矩阵，则有

$$\left(A^T\right)^{-1} = \left(A^{-1}\right)^T, \quad \left(A^H\right)^{-1} = \left(A^{-1}\right)^H$$

Proof.

$$\left(AA^{-1}\right)^T = I \Rightarrow \underbrace{\left(A^{-1}\right)^T}_{\text{the inverse of } A^T} A^T = I$$

■

Corollary 9.5.2 如果矩阵 A 和矩阵 B 都为非奇异矩阵，则乘积 AB 也为非奇异矩阵

$$(AB)^{-1} = B^{-1}A^{-1}$$

Proof.

$$(AB) \quad \underbrace{B^{-1}A^{-1}}_{\text{the inverse of AB}} = I$$

■

9.6 Gram Matrix 非奇异的性质

Gram 矩阵的定义见 6.4.1.

Corollary 9.6.1 — Gram Matrix 可逆等价于 A 列线性无关. 矩阵 $A \in \mathbb{R}^{m\times n}, \mathrm{G} = A^T A$
矩阵 A 列向量线性无关 $\Leftrightarrow$ Gram 矩阵 G 非奇异.

Proof. " $\Rightarrow$ ":
假设矩阵 A 列向量线性无关, $A^T A$ 奇异. 则存在 $A^T Ax = 0, x \neq 0$, 可得 $x^T A^T Ax = \|Ax\|_2^2 = 0$, 即 $Ax = 0$ 与列向量线性无矛盾.
" $\Leftarrow$ ":
假设 $A^T A$ 非奇异, 矩阵 A 列向量线性相关. 则有 $Ax = 0, x \neq 0$, 可得 $A^T Ax = 0$, 即 $A^T A$ 是奇异矩阵. ■

9.7 伪逆

Definition 9.7.1 — Pseudo-inverse.

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

$$A^{\dagger} = V\Sigma^{+}U^T = [v_1 \cdots v_r \cdots v_n] \begin{bmatrix} \sigma_1^{-1} & & \\ & \ddots & \\ & & \sigma_r^{-1} \end{bmatrix} [u_1 \cdots u_r \cdots u_m]^{\mathrm{T}}$$

Theorem 9.7.1 伪逆 $A^{\dagger}$ 为 A 的右逆

Proof.

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

■

Theorem 9.7.2 当 A 为方阵时，右逆等于矩阵的逆

Proof.

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

■

Theorem 9.7.3

Corollary 9.7.4 以下三个结论为等价的，对于实矩阵 A

- A 是可左逆的
- A 的列向量线性无关
- $A^T A$ 为非奇异矩阵

Corollary 9.7.5 以下三个结论为等价的，对于实矩阵 A

- A 是可右逆的
- A 的行向量线性无关
- AA^T 为非奇异矩阵

By choosing good bases, A multiplies $\boldsymbol{v}_i$ in the row space to give $\sigma_i \boldsymbol{u}_i$ in the column space. A^{-1} must do the opposite!

If $A\boldsymbol{v} = \sigma \boldsymbol{u}$ then $A^{-1}\boldsymbol{u} = \boldsymbol{v}/\sigma$. The singular values of A^{-1} are $1/\sigma$, just as the eigenvalues of A^{-1} are $1/\lambda$. The bases are reversed. The u 's are in the row space of A^{-1}, the v 's are in the column space.

The pseudoinverse $A^{+}$is an n by m matrix. **If A^{-1} exists, then $A^{+}$is the same as** A^{-1}. In that case $m = n = r$ and we are inverting $U\Sigma V^{\mathrm{T}}$ to get $V\Sigma^{-1}U^{\mathrm{T}}$.

The new symbol $A^{+}$is needed when $r < m$ or $r < n$. Then A has no two-sided inverse, but it has a *pseudo*inverse $A^{+}$with that same rank r :

$$A^{+}\boldsymbol{u}_i = \frac{1}{\sigma_i}\boldsymbol{v}_i \quad \text{for } i \le r \quad \text{and} \quad A^{+}\boldsymbol{u}_i = \boldsymbol{0} \quad \text{for } i > r$$

The vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r$ in the column space of A go back to $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r$ in the row space.

The other vectors $\boldsymbol{u}_{r+1}, \ldots, \boldsymbol{u}_m$ are in the left nullspace, and A^+sends them to zero. When we know what happens to all those basis vectors, we know A^+.

Notice the pseudoinverse of the diagonal matrix Σ. Each σ in Σ is replaced by σ^{-1} in Σ^+.The product $\Sigma^+\Sigma$ is as near to the identity as we can get. It is a projection matrix, $\Sigma^+\Sigma$ is partly I and otherwise zero. We can invert the σ 's, but we can't do anything about the zero rows and columns.

Figure 9.2: $A\boldsymbol{x}^\dagger$ in the column space goes back to $A^\dagger A\boldsymbol{x}^\dagger = \boldsymbol{x}^\dagger$ in the row space

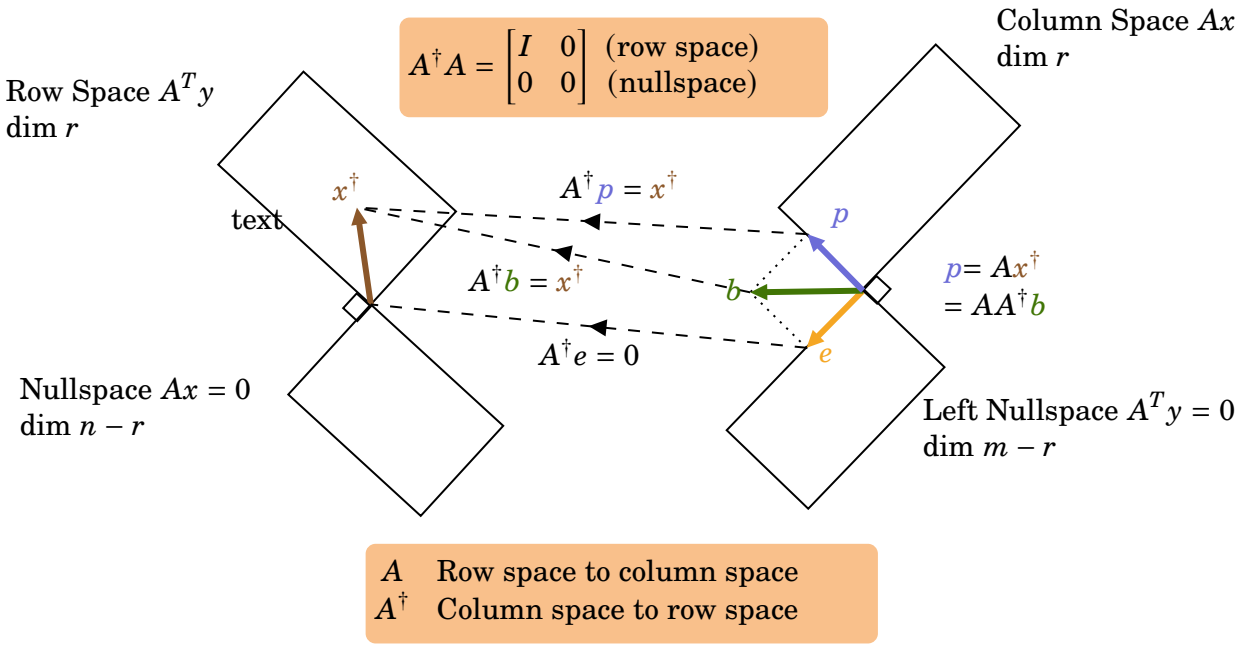
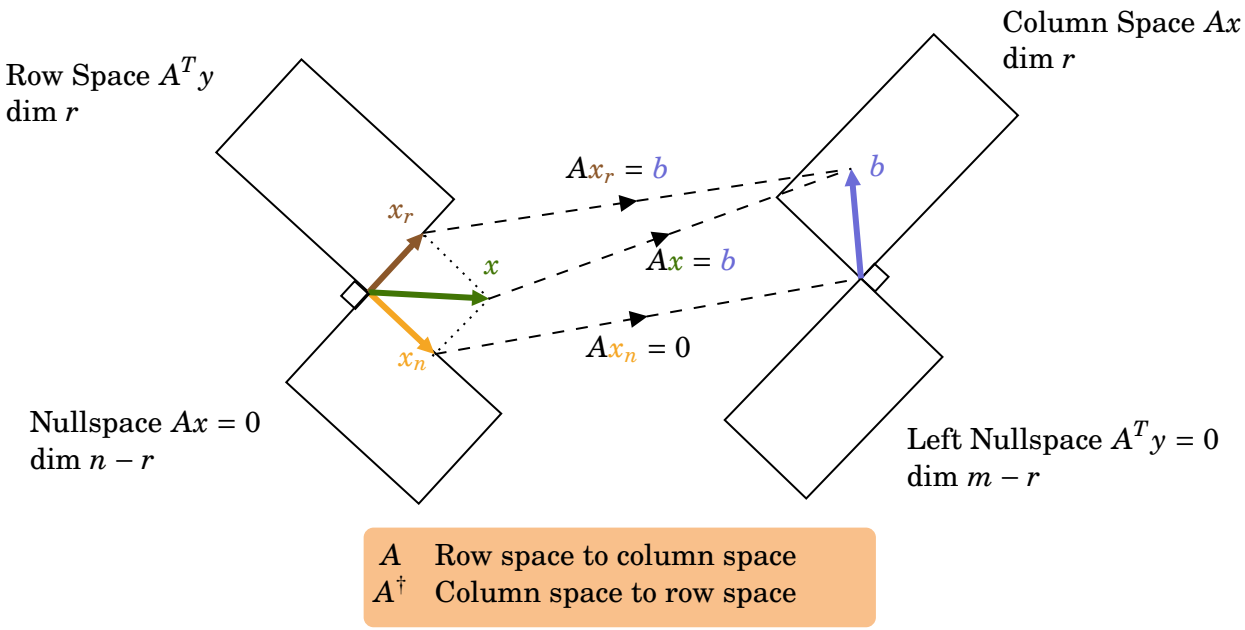


Figure 9.3: Projection from row space to column space



10. Orthogonal Matrices

10.1 预备知识

10.1.1 标准正交向量

参见 5.3.2。

10.1.2 Gram 矩阵与标准正交的关系

For the definition of Gram matrices, refer to 6.4.1. 关于它与非奇异的性质，参见 9.6.

Theorem 10.1.1 如果 A 的 Gram 矩阵为单位矩阵，则 $A \in \mathbb{R}^{m\times n}$ 具有标准正交列.

Proof.

$$
\begin{aligned}
A^TA &= \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix}^T \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix} \\
&= \begin{bmatrix} a_1^Ta_1 & a_1^Ta_2 & \cdots & {a_1}^Ta_n \\ a_2^Ta_1 & {a_2}^Ta_2 & \cdots & a_2^Ta_n \\ \vdots & \vdots & \ddots & \vdots \\ a_n^Ta_1 & a_n^Ta_2 & \cdots & a_n^Ta_n \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \\
&= I
\end{aligned}
$$

■

10.1.3 矩阵-向量乘积与标准正交的关系

如果 $A \in \mathbb{R}^{m\times n}$ 具有标准正交列，则线性函数 $f(x) = Ax$

Theorem 10.1.2 保持原内积.

$$\langle Ax, Ay\rangle = x^T y$$

Proof.

$$\langle Ax, Ay\rangle = (Ax)^T(Ay) = x^T A^T A y = x^T y$$

■

Theorem 10.1.3 保持原范数.

$$\|Ax\|_2 = \|x\|_2$$

Proof.

$$\|Ax\|_2 = \left((Ax)^T(Ax)\right)^{1/2} = \left(x^T x\right)^{1/2} = \|x\|_2$$

■

Theorem 10.1.4 保持原距离.

$$\|Ax - Ay\|_2 = \|x - y\|_2$$

Proof.

$$\|Ax - Ay\|_2 = \left((Ax - Ay)^T(Ax - Ay)\right)^{1/2} = \left((x - y)^T(x - y)\right)^{1/2} = \|x - y\|_2$$

■

Theorem 10.1.5 保持原角度.

$$\angle(Ax, Ay) = \angle(x, y)$$

Proof.

$$\angle(Ax, Ay) = \arccos\left(\frac{(Ax)^T(Ay)}{\|Ax\|_2\|Ay\|_2}\right) = \arccos\left(\frac{x^T y}{\|x\|_2\|y\|_2}\right) = \angle(x, y)$$

■

10.1.4 左可逆性与正交的关系

如果矩阵 $A \in \mathbb{R}^{m\times n}$ 有标准正交列，则：

Theorem 10.1.6 A 是左可逆的，其左逆为 A^T.

Proof. 根据定义：

$$A^T A = I$$

■

Theorem 10.1.7 A 有线性无关的列向量.

Proof.

$$Ax = 0 \quad \Rightarrow \quad A^T Ax = x = 0$$

■

Theorem 10.1.8 A 是高的或者方的，即 $m \geq n$.

Proof. 列向量 $a_1, a_2, \ldots, a_n \in \mathbb{R}^m$, 由维度定理可得 $n \leq m$。 ■

10.2 正交矩阵

Definition 10.2.1 — **正交矩阵**. 所有列两两相互正交的**方形**实矩阵。

Theorem 10.2.1 — **正交矩阵满足非奇异性**. 即如果矩阵 A 是正交的，则 A 是可逆的，左逆等于右逆，且它的逆为 A^T.

$$\left.\begin{array}{l} A^T A = I \\ A \text{ 是方的} \end{array}\right\} \quad \Rightarrow \quad AA^T = I$$

Corollary 10.2.2 A^T 也是一个正交矩阵。

Corollary 10.2.3 A 的行是标准正交的，即 a_i 范数为 1 且相互正交。

如果 $A \in \mathbb{R}^{m \times n}$ 有标准正交列以及 $m > n$, 则 $AA^T \neq I$。

10.3 Permutation Matrices

Notation 10.1. $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ 为 $(1, 2, \ldots, n)$ 的一个重新排序的排列。
将 π 与一个置换矩阵 $A \in \mathbb{R}^{n \times n}$ 联系起来:

$$A_{i\pi_i} = 1, \quad A_{ij} = 0 \text{ 如果} j \neq \pi_i$$

Definition 10.3.1 — **置换**. Ax 是 x 的一个置换:

$$Ax = \left(x_{\pi_1}, x_{\pi_2}, \ldots, x_{\pi_n}\right)$$

A 在每一行和每一列中都有一个等于 1 的元素。

Theorem 10.3.1 置换矩阵满足正交性，即所有置换矩阵都是正交的.

Corollary 10.3.2

$$A^T A = I$$

Proof. 因为 A 的每一行有一个元素等于 1

$$\left(A^T A\right)_{ij} = \sum_{k=1}^{n} A_{ki} A_{kj} = \begin{cases} 1 & i = j \\ 0 & \text{其它} \end{cases}$$

■

Corollary 10.3.3 $A^T = A^{-1}$ 是逆置换矩阵.

▪ **Example 10.1** 若 $\{1, 2, 3, 4\}$ 的置换为:

$$(\pi_1, \pi_2, \pi_3, \pi_4) = (2, 4, 1, 3)$$

相应的置换矩阵及其逆矩阵为

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

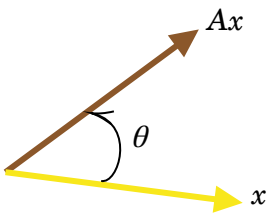
A^T 是与置换相关的置换矩阵

$$(\tilde{\pi}_1, \tilde{\pi}_2, \tilde{\pi}_3, \tilde{\pi}_4) = (3, 1, 4, 2)$$

■

10.4 平面旋转

Figure 10.1: An example of rotation



▪ **Example 10.2 — Rotation Matrices in $\mathbb{R}^2$.** 在一个 2 维平面的旋转可以用矩阵表示为

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

■

▪ **Example 10.3 — Rotation Matrices in $\mathbb{R}^3$.**

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

描述了在 $\mathbb{R}^3$ 中 (x_1, x_3) 平面的旋转。 ■

10.5 Householder Matrix

Definition 10.5.1 — **Householder matrix, reflator matrix.**

$$A = I - 2aa^T$$

其中，向量 a 满足 $\|a\|_2 = 1$ 。

Theorem 10.5.1 反射矩阵 (reflector matrix) 是对称的.

$$A^T = A$$

Theorem 10.5.2 反射矩阵 (reflector matrix) 是正交的。

Proof.

$$A^T A = \left(I - 2aa^T\right)\left(I - 2aa^T\right) = I - 4aa^T + 4aa^T aa^T = I$$

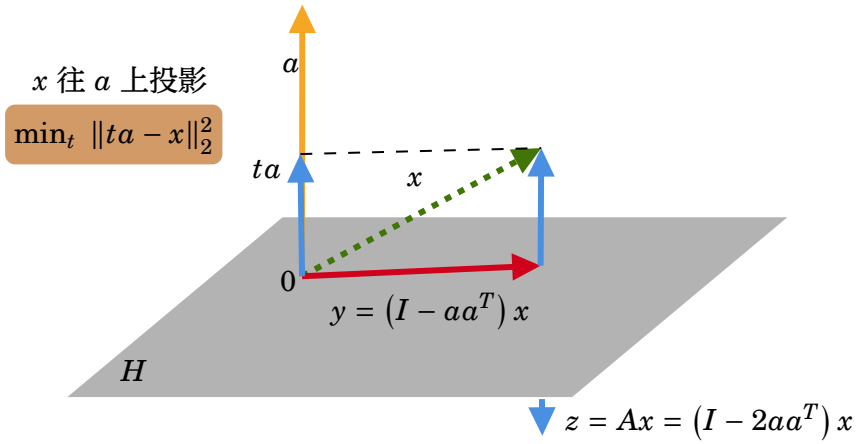
■

Theorem 10.5.3 反射矩阵是对合矩阵 (Involutory matrix)，即它的逆是它本身。

$$H = H^{-1}$$

10.5.1 The Geometry of Householder Transformation

Figure 10.2: Elementary Reflection



$H = \{u \mid a^T u = 0\}$ 是与 a 正交的向量的 (超) 平面。

Corollary 10.5.4 如果 $\|a\|_2 = 1$， x 在 H 上的投影为

$$y = x - \left(a^T x\right) a = x - a\left(a^T x\right) = \left(I - aa^T\right) x$$

Proof. 1. $y \in H$.

$$a^T y = a^T\left(x - a\left(a^T x\right)\right) = a^T x - \left(a^T a\right)\left(a^T x\right) = a^T x - a^T x = 0$$

2. 考虑任意 $z \in H (z \neq y)$, 证明 $\|x - z\| > \|x - y\|$

$$\begin{aligned}\|x-z\|_2^2 &= \|x-y+y-z\|_2^2 \\ &= \|x-y\|_2^2 + 2(x-y)^T(y-z) + \|y-z\|_2^2 \\ &= \|x-y\|_2^2 + 2\left(a^T x\right)a^T(y-z) + \|y-z\|_2^2 \\ &= \|x-y\|_2^2 + \|y-z\|_2^2 \quad (\text{因为} a^T y = a^T z = 0) \\ &\geq \|x-y\|_2^2\end{aligned}$$

■

Corollary 10.5.5 x 通过超平面的反射由反射算子的乘积给出

$$z = y + (y-x) = \left(I - 2aa^T\right)x$$

10.6 正交矩阵乘积

若 $A_1, \ldots, A_k \in \mathbb{R}^{n\times n}$ 是正交矩阵，那么它们的乘积为:

$$A = A_1 A_2 \cdots A_k$$

Corollary 10.6.1 — 正交矩阵乘积的正交性.

$$\begin{aligned}A^T A &= (A_1 A_2 \cdots A_k)^T (A_1 A_2 \cdots A_k) \\ &= A_k^T \cdots A_2^T A_1^T A_1 A_2 \cdots A_k \\ &= I\end{aligned}$$

10.7 具有正交矩阵的线性方程

系数正交矩阵 $A \in \mathbb{R}^{n\times n}$ 的线性方程;

$$Ax = b$$

解为:

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

10.7.1 The Complexity of the Multiplication Ax of Orthogonal Matrix A

可以在 $2n^2$ 个 flop 内计算矩阵向量乘法。

如果 A 有特殊性质，代价将会小于 n^2。例如，

- 置换矩阵：0 flop。
- 反射算子 (给定 a)：$4n$ flops。
- 平面旋转：$O(1)$ flop。

10.8 列标准正交的高矩阵

Theorem 10.8.1 假设矩阵 $A \in \mathbb{R}^{m\times n}$ 是高的 $(m > n)$, 具有标准正交列, 则有 A^T 具有标准正交行.

Theorem 10.8.2 A^T 是 A 的一个左逆。

$$A^T A = I$$

Theorem 10.8.3 A 没有右逆。

10.9 值域范围、列空间

Definition 10.9.1 — **向量集合张成的空间.** 一个向量集合张成的空间是其所有线性组合的集合.

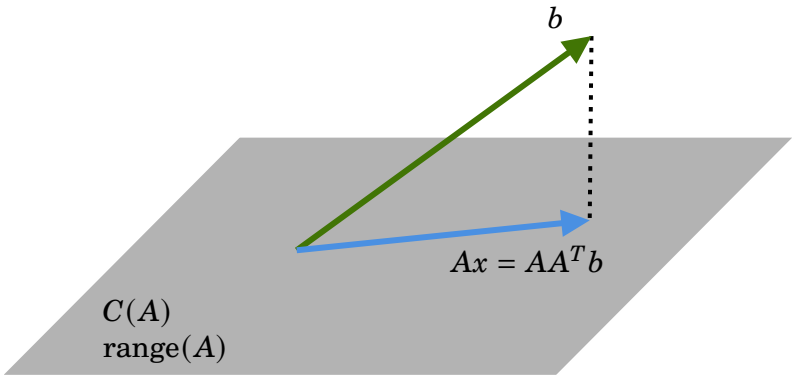
$$\text{span}\,(a_1, a_2, \cdots, a_n) = \{x_1 a_1 + x_2 a_2 + \cdots + x_n a_n \mid x \in \mathbb{R}^n\}$$

Definition 10.9.2 — **A 的范围（列空间）.**

$$\text{range}(A) = \{Ax \mid x \in \mathbb{R}^n\}$$

10.9.1 投影到列标准正交的矩阵 A 的列空间

Figure 10.3: Projection onto the column space of A, A has orthonormal columns



Problem 10.1 假设矩阵 $A \in \mathbb{R}^{m\times n}$ **具有标准正交列**, 求投影 b 在 $C(A)$ 上的投影。 ∎

即向量 Ax 与 b 有最短距离

$$\min_x \|Ax - b\|_2^2$$

$$\begin{aligned} f(x) &= \|Ax - b\|_2^2 \\ &= (Ax - b)^T (Ax - b) = x^T A^T Ax - 2x^T A^T b + b^T b \\ &= x^T x - 2x^T A^T b + b^T b \left(\because A^T A = I\right) \end{aligned}$$

$$\nabla f(x) = 2x - 2A^T b = 0 \Rightarrow x = A^T b$$

$AA^T b$ 称为向量 $\mathrm{b} \in \mathbb{R}^m$ 在 range(A) 上的正交投影。

Theorem 10.9.1

$$Ax = AA^T b \in \text{range}(A)$$

且是 b 在 $C(A)$ 上的投影。

Proof. 1.

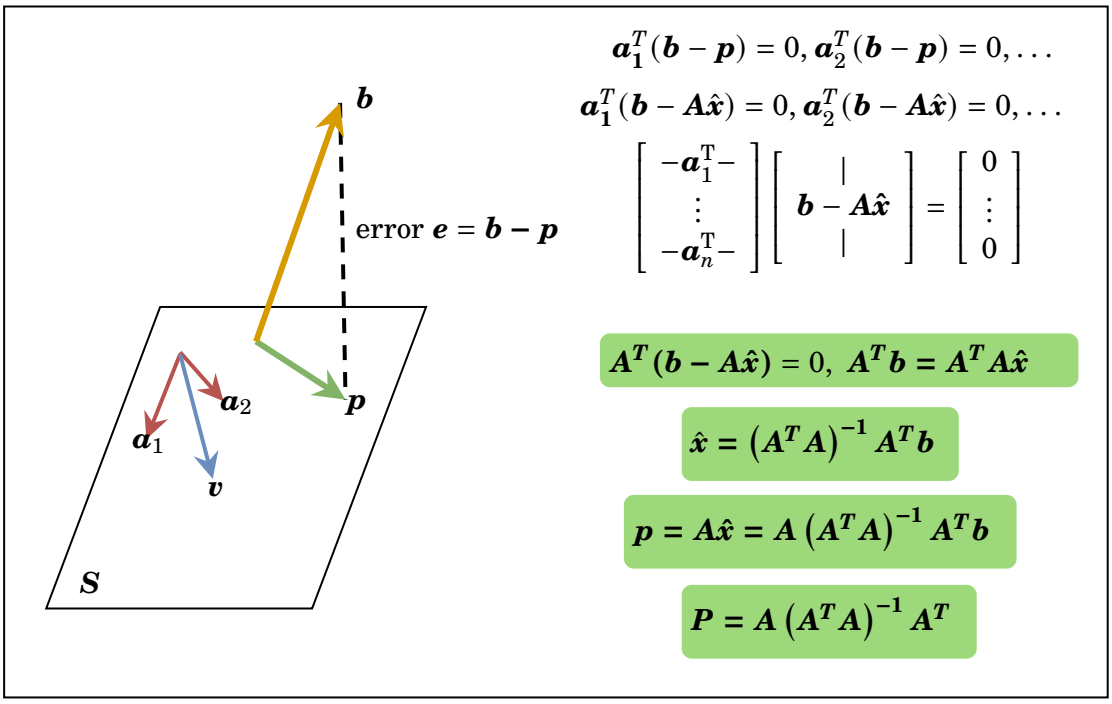
$$Ax = AA^T b \in \text{range}(A)$$

2. 可以证明 $\hat{x} = A^T b$ 满足 $\|A\hat{x} - b\| < \|Ax - b\|$, 对于所有 $x \neq \hat{x}$。
b 到 range(A) 内任意点 Ax 的距离的平方和为:

$$\begin{aligned}
\|Ax - b\|_2^2 &= \|A(x - \hat{x}) + A\hat{x} - b\|_2^2 \quad \left(\text{其中} \hat{x} = A^T b\right) \\
&= \|A(x - \hat{x})\|_2^2 + \|A\hat{x} - b\|_2^2 + 2(x - \hat{x})^T A^T (A\hat{x} - b) \\
&= \|A(x - \hat{x})\|_2^2 + \|A\hat{x} - b\|_2^2 \\
&= \|(x - \hat{x})\|_2^2 + \|A\hat{x} - b\|_2^2 \\
&\geq \|A\hat{x} - b\|_2^2
\end{aligned}$$

当且仅当 $x = \hat{x}$, 等号成立。
第 3 行成立是因为 $A^T(A\hat{x} - b) = \hat{x} - A^T b = 0$ 。 ■

Figure 10.4: Projection of $\boldsymbol{b}$ into the column space of A, A is any matrice. Sourced from [**Strang1993IntroductionTL**]



11. QR 分解与 Householder 变换

11.1 Triangular Matrices

Definition 11.1.1 — Lower Triangular Matrices. 矩阵 $A \in \mathbb{R}^{n\times n}$ 为下三角 *(Lower Triangular)* 矩阵, $A_{ij} = 0, j > i$ 。

$$A = \begin{bmatrix} A_{11} & 0 & \cdots & 0 & 0 \\ A_{21} & A_{22} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ A_{n-1,1} & A_{n-1,2} & \cdots & A_{n-1,n-1} & 0 \\ A_{n1} & A_{n2} & \cdots & A_{n,n-1} & A_{nn} \end{bmatrix}$$

Definition 11.1.2 — Upper Triangular Matrices. A^T 为上三角 *(Upper Triangular)* 矩阵。

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1,n-1} & A_{1,n} \\ 0 & A_{22} & \cdots & A_{2,n-1} & A_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & A_{n-1,n-1} & A_{n-1,n} \\ 0 & 0 & \cdots & 0 & A_{nn} \end{bmatrix}$$

Definition 11.1.3 — 单位上三角矩阵，单位下三角矩阵. 对角元素 a_{ii} 都等于 1 的上（下）三角矩阵。

11.1.1 高斯消元法

Problem 11.1 当 A 是具有非零对角元素的下三角矩阵时，解 $Ax = b$ 。 ∎

使用前向回代 (Forward Substitution) 算法求解。

时间复杂度：$1 + 3 + 5 + \ldots + (2n - 1) = n^2$ flops

Problem 11.2 当 A 是具有非零对角元素的上三角矩阵, 解 $Ax = b$ 。 ∎

Algorithm 4: Forward Substitution

Input: $A \in R^{n\times n}$(A是下三角矩阵), $b \in R^{n\times 1}$
Output: $x \in R^{n\times 1}$

$$x_1 = \frac{b_1}{A_{11}}$$
$$x_2 = \frac{b_2 - A_{21}x_1}{A_{22}}$$
$$x_3 = \frac{b_3 - A_{31}x_1 - A_{32}x_2}{A_{33}}$$
$$\cdots$$
$$x_n = \frac{b_n - A_{n1}x_1 - A_{n2}x_2 - \cdots - A_{n,n-1}x_{n-1}}{A_{nn}}$$

使用后向回代 (Back Substitution) 算法来求解.

Algorithm 5: Backward Substitution

Input: $A \in R^{n\times n}$(A是上三角矩阵), $b \in R^{n\times 1}$
Output: $x \in R^{n\times 1}$

$$x_n = \frac{b_n}{A_{nn}}$$
$$x_{n-1} = \frac{b_{n-1} - A_{n-1,n}x_n}{A_{n-1,n-1}}$$
$$x_{n-2} = \frac{b_{n-2} - A_{n-2,n-1}x_{n-1} - A_{n-2,n}x_n}{A_{n-2,n-2}}$$
$$\cdots$$
$$x_1 = \frac{b_1 - A_{12}x_2 - A_{13}x_3 - \cdots - A_{1n}x_n}{A_{11}}$$

时间复杂度：$1 + 3 + \ldots + 2n - 1 = n^2$ flops

11.1.2 The Inverses of Triangular Matrices

Theorem 11.1.1 对角元素非零的三角矩阵 A 是非奇异的，即:

$$Ax = 0 \quad \Rightarrow \quad x = 0$$

Theorem 11.1.2 — **高斯消元法**. A 的逆可以通过逐列解方程 $AX = I$ 来计算得到

$$A \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} = \begin{bmatrix} e_1 & e_2 & \cdots & e_n \end{bmatrix}$$

Theorem 11.1.3 下三角矩阵的逆是下三角矩阵，上三角矩阵的逆是上三角矩阵。

上/下三角矩阵 $A \in \mathbb{R}^{n\times n}$ 逆的复杂度

$$n^2 + (n-1)^2 + \cdots + 1 \approx \frac{1}{3}n^3 \text{ flops}$$

11.2 QR Factorization

如果矩阵 $A \in \mathbb{R}^{m\times n}$ 的列向量线性无关，则可以将其分解为

Theorem 11.2.1 — QR Factorization.

$$\begin{aligned} A_{n\times n} &= \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix} \\ &= \begin{bmatrix} q_1 & q_2 & \cdots & q_n \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & \cdots & R_{1n} \\ 0 & R_{22} & \cdots & R_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_{nn} \end{bmatrix} \\ &= Q_{n\times n} R_{n\times n} \end{aligned}$$

向量 $q_1, \ldots, q_n \in \mathbb{R}^m$ 是标准正交向量:

$$\|q_i\|_2 = 1, \quad q_i^T q_j = 0, \text{ if } \quad i \neq j$$

对角元素 R_{ii} 是非零的。若 $R_{ii} < 0$, 改变 $R_{ii}, \cdots, R_{in}$ 和向量 q_i 的符号。大多数定义要求 $R_{ii} > 0$, 使得 Q 和 R 是唯一的。

Corollary 11.2.2 $Q \in \mathbb{R}^{m\times n}$ 具有标准正交列 $\left(Q^T Q = I\right)$.

Corollary 11.2.3 如果 A 是方阵 (m = n), 则 Q 是正交的 $\left(Q^T Q = QQ^T = I\right)$.

Corollary 11.2.4 $R \in \mathbb{R}^{n\times n}$ 的上三角矩阵.

Corollary 11.2.5 R 是非奇异的 (对角元素是非零的).

Corollary 11.2.6

$$R = Q^{-1}A \Rightarrow R = Q^T A$$

QR 分解可通过 Gram-Schmidit 正交化法（参见5.4）、Householder 变换等进行。

Algorithm 6: QR Decomposition Using Gram-Schmidt Algorithm

1. 设矩阵 A 的列向量依次为 $a_1, a_2, \cdots, a_n$ ，由于 A 为非奇异矩阵, 则列向量线性无关
2. 对列向量 $a_1, a_2, \ldots, a_n$ 按照 Gram-Schmidt 方法进行正交化，然后单位化
3. 单位化得到的标准正交向量 $q_1, q_2, \ldots, q_n$, 即得到标准正交矩阵 Q
4. 根据 $R = Q^{-1}A \Rightarrow R = Q^T A$, 得到上三角矩阵 R
5. QR 分解 $A = QR$

■ **Example 11.1** 矩阵 A 的 QR 分解过程

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

令 $a_1=(1,1,0)^T, a_2=(1,-1,0)^T, a_3=(0,1,2)^T$, 由 Schmidt 方法正交单位化后, 得到 $q_1=\left(\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},0\right)^T, q_2=\left(\frac{1}{\sqrt{2}},-\frac{1}{\sqrt{2}},0\right)^T, q_3=(0,0,1)^T$ 。

所以 $a_1=\sqrt{2}q_1,\quad a_2=\sqrt{2}q_2,\quad a_3=\frac{1}{\sqrt{2}}q_1-\frac{1}{\sqrt{2}}q_2+2q_3$ 。

$$A=QR=\begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0\\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\sqrt{2} & 0 & \frac{1}{\sqrt{2}}\\ 0 & \sqrt{2} & -\frac{1}{\sqrt{2}}\\ 0 & 0 & 2\end{bmatrix}$$

■

■ **Example 11.2**

$$\begin{aligned}\begin{bmatrix}-1 & -1 & 1\\ 1 & 3 & 3\\ -1 & -1 & 5\\ 1 & 3 & 7\end{bmatrix}&=\begin{bmatrix}-1/2 & 1/2 & -1/2\\ 1/2 & 1/2 & -1/2\\ -1/2 & 1/2 & 1/2\\ 1/2 & 1/2 & 1/2\end{bmatrix}\begin{bmatrix}2 & 4 & 2\\ 0 & 2 & 8\\ 0 & 0 & 4\end{bmatrix}\\ &=\begin{bmatrix}q_1 & q_2 & q_3\end{bmatrix}\begin{bmatrix}R_{11} & R_{12} & R_{13}\\ 0 & R_{22} & R_{23}\\ 0 & 0 & R_{33}\end{bmatrix}\\ &=QR\end{aligned}$$

■

11.2.1 QR 分解的存在唯一性[1]

可以证明，QR 分解具有唯一性。

Theorem 11.2.7 — $m=n$ **矩阵** A **进行 QR 分解的唯一性.** If $\boldsymbol{A}=\boldsymbol{Q}_1\boldsymbol{R}_1=\boldsymbol{Q}_2\boldsymbol{R}_2$ are two QR decompositions of full rank, square A, then

$$\begin{aligned}\boldsymbol{Q}_2&=\boldsymbol{Q}_1\boldsymbol{S}\\ \boldsymbol{R}_2&=\boldsymbol{S}\boldsymbol{R}_1\end{aligned}$$

for some square diagonal $\boldsymbol{S}$ with entries ±1.

If we require the diagonal entries of $\boldsymbol{R}$ to be positive, then the decomposition is unique.

Theorem 11.2.8 — $m<n$ **矩阵** A **进行 QR 分解的唯一性.** If

$$\boldsymbol{A}=\boldsymbol{Q}_1\begin{bmatrix}\boldsymbol{R}_1 & \boldsymbol{N}_1\end{bmatrix}=\boldsymbol{Q}_2\begin{bmatrix}\boldsymbol{R}_2 & \boldsymbol{N}_2\end{bmatrix}$$

are two QR decompositions of a full rank, $m\times n$ matrix $\boldsymbol{A}$ with $m<n$, then

$$\boldsymbol{Q}_2=\boldsymbol{Q}_1\boldsymbol{S},\quad \boldsymbol{R}_2=\boldsymbol{S}\boldsymbol{R}_1,\quad \text{and}\quad \boldsymbol{N}_2=\boldsymbol{S}\boldsymbol{N}_1$$

for square diagonal S with entries ±1.

If we require the diagonal entries of $\boldsymbol{R}$ to be positive, then the decomposition is unique.

Proof. Let $\boldsymbol{Q}_1\begin{bmatrix}\boldsymbol{R}_1 & \boldsymbol{N}_1\end{bmatrix}=\boldsymbol{Q}_2\begin{bmatrix}\boldsymbol{R}_2 & \boldsymbol{N}_2\end{bmatrix}$ with $\boldsymbol{Q}_i$ being $m\times m$ and orthogonal, $\boldsymbol{R}_i$ being $m\times m$ and upper triangular, and $\boldsymbol{N}_i$ being an arbitrary $m\times(n-m)$ matrix.

[1]Reference: SciComp, Kyle Kloster's Website

Then multiplying through yields $\boldsymbol{Q}_1\boldsymbol{R}_1 = \boldsymbol{Q}_2\boldsymbol{R}_2$, two QR decompositions of a full rank, $m \times m$ matrix.

Using the theorem above, we get that $\boldsymbol{Q}_2 = \boldsymbol{Q}_1\boldsymbol{S}$ and $\boldsymbol{R}_2 = \boldsymbol{S}\boldsymbol{R}_1$ for a diagonal matrix $\boldsymbol{S}$ with entries ± 1.

Looking at the right-most partition of the original product yields $\boldsymbol{Q}_1\boldsymbol{N}_1 = \boldsymbol{Q}_2\boldsymbol{N}_2$. But we've shown $\boldsymbol{Q}_2 = \boldsymbol{Q}_1\boldsymbol{S}$, so now we have $\boldsymbol{Q}_1\boldsymbol{N}_1 = \boldsymbol{Q}_1\boldsymbol{S}\boldsymbol{N}_2$.

Left-multiplying by $\boldsymbol{Q}_1^T$ and then by $\boldsymbol{S}$ then proves $\boldsymbol{N}_2 = \boldsymbol{S}\boldsymbol{N}_1$, completing the theorem. ■

Theorem 11.2.9 — $m > n$ 矩阵 A 进行 QR 分解的唯一性. If $\boldsymbol{A} = \begin{bmatrix} \boldsymbol{Q}_1 & \boldsymbol{U}_1 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_1 \\ 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}_2 & \boldsymbol{U}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_2 \\ 0 \end{bmatrix}$ are two QR decompositions of a full rank, $m \times n$ matrix $\boldsymbol{A}$ with $m > n$, then

$$\boldsymbol{Q}_2 = \boldsymbol{Q}_1\boldsymbol{S}, \quad \boldsymbol{R}_2 = \boldsymbol{S}\boldsymbol{R}_1, \quad \text{and} \quad \boldsymbol{U}_2 = \boldsymbol{U}_1\boldsymbol{T}$$

for square diagonal $\boldsymbol{S}$ with entries ± 1, and square orthogonal $\boldsymbol{T}$.

If we require the diagonal entries of $\boldsymbol{R}$ to be positive, then $\boldsymbol{Q}$ and $\boldsymbol{R}$ are unique.

Proof. Let $\boldsymbol{A}$ be full rank and $m \times n$ with $m > n$. Suppose it has decompositions

$$\boldsymbol{A} = \tilde{\boldsymbol{Q}}_1\tilde{\boldsymbol{R}}_1 = \tilde{\boldsymbol{Q}}_2\tilde{\boldsymbol{R}}_2$$

for $m \times m$ orthogonal matrices $\tilde{\boldsymbol{Q}}_i$, $m \times n$ and upper-triangular matrices $\tilde{\boldsymbol{R}}_i$. (We know we can do this because the QR decomposition always exists).

Since $m > n$, we can write $\tilde{\boldsymbol{Q}}_i = \begin{bmatrix} \boldsymbol{Q}_i & \boldsymbol{U}_i \end{bmatrix}$ and $\tilde{\boldsymbol{R}}_i = \begin{bmatrix} \boldsymbol{R}_i \\ 0 \end{bmatrix}$ where $\boldsymbol{Q}_i$ is $m \times n$ and $\boldsymbol{U}_i$ is $m \times (m-n)$. Then

$$\boldsymbol{A} = \tilde{\boldsymbol{Q}}_i\tilde{\boldsymbol{R}}_i = \begin{bmatrix} \boldsymbol{Q}_i & \boldsymbol{U}_i \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_i \\ 0 \end{bmatrix} = \boldsymbol{Q}_i\boldsymbol{R}_i$$

where $\boldsymbol{R}_i$ is square, upper-triangular, invertible (because $\boldsymbol{A}$ is full rank), and the columns of $\boldsymbol{Q}_i$ are orthonormal so $\boldsymbol{Q}_i$ satisfies $\boldsymbol{Q}_i^T\boldsymbol{Q}_i = \boldsymbol{I}$. Then we have

$$\boldsymbol{Q}_1\boldsymbol{R}_1 = \boldsymbol{Q}_2\boldsymbol{R}_2$$

and left-multiplying by $\boldsymbol{Q}_2^T$ and right-multiplying by $\boldsymbol{R}_1^{-1}$ yields

$$\boldsymbol{Q}_2^T\boldsymbol{Q}_1 = \boldsymbol{R}_2\boldsymbol{R}_1^{-1}$$

Note that the right-hand side of Eqn (2) is upper-triangular (since $\boldsymbol{R}_i$ is). On the other hand, left-multiplying Eqn (1) by $\boldsymbol{Q}_1^T$ and right-multiplying by $\boldsymbol{R}_2^{-1}$ gives $Q_1^TQ_2 = R_1R_2^{-1}$, and taking the transpose yields a lower-triangular expression for $\boldsymbol{Q}_2^T\boldsymbol{Q}_1$. Therefore $\boldsymbol{Q}_1^T\boldsymbol{Q}_2 = \boldsymbol{R}_1\boldsymbol{R}_2^{-1}$ is both lower- and upper-triangular, and so it is diagonal. Call it $\boldsymbol{D}$. Then right-multiplying Eqn (1) by $\boldsymbol{R}_2^{-1}$ yields

$$\boldsymbol{Q}_2\boldsymbol{R}_2\boldsymbol{R}_2^{-1} = \boldsymbol{Q}_2 = \boldsymbol{Q}_1\boldsymbol{R}_1\boldsymbol{R}_2^{-1} = \boldsymbol{Q}_1\boldsymbol{D}$$

and so $\boldsymbol{Q}_2 = Q_1\boldsymbol{D}$. Multiplying this by its transpose and using orthogonality of $\boldsymbol{Q}_i$ we get $\boldsymbol{I} = \boldsymbol{Q}_2^T\boldsymbol{Q}_2 = (\boldsymbol{Q}_1\boldsymbol{D})^T(\boldsymbol{Q}_1\boldsymbol{D}) = \boldsymbol{D}^T\boldsymbol{Q}_1^T\boldsymbol{Q}_1\boldsymbol{D} = \boldsymbol{D}^T\boldsymbol{D} = \boldsymbol{D}^2$. This proves $\boldsymbol{D}^2 = \boldsymbol{I}$, so

$\boldsymbol{D} = \boldsymbol{S}$, a diagonal matrix with entries ±1. So $\boldsymbol{Q}_2 = \boldsymbol{Q}_1\boldsymbol{S}$. Left multiplying Eqn (1) by $\boldsymbol{Q}_2^T = \boldsymbol{S}\boldsymbol{Q}_1^T$ then yields

$$\boldsymbol{S}\boldsymbol{Q}_1^T\boldsymbol{Q}_1\boldsymbol{R}_1 = \boldsymbol{S}\boldsymbol{R}_1 = \boldsymbol{Q}_2^T\boldsymbol{Q}_2\boldsymbol{R}_2 = \boldsymbol{R}_2$$

proving that $\boldsymbol{R}_2 = \boldsymbol{S}\boldsymbol{R}_1$.

■

11.2.2 复矩阵的 QR 分解

Theorem 11.2.10 如果 $A \in \mathbb{C}^{m\times n}$ 的列向量是线性无关的，则可以将其分解为

$$A = QR$$

$Q \in \boldsymbol{C}^{m\times n}$ 具有正交列。$\left(Q^HQ = I\right)$
$R \in \boldsymbol{C}^{n\times n}$ 具有实非零对角元素的上三角矩阵。

大多数情况下，会优先选择对角线元素 R_{ii} 为正数。
如果没有特别说明，之后默认矩阵 A 都是实数的。

11.3 QR 分解的应用

可用 QR 分解求解以下问题：

- 线性方程
- 最小二乘问题
- 带约束的最小二乘问题

11.3.1 QR 分解和求解线性方程组 $Ax = b$

QR 分解的思想可以用于加快求逆矩阵速度。

Corollary 11.3.1

$$Ax = b \Rightarrow x = A^{-1}b$$
$$QRx = b \Rightarrow x = R^{-1}\left(Q^Tb\right)$$

Algorithm 8: Solving linear equations via QR factorization

Input: $n \times n$ invertible matrix A

1. QR factorization. Compute the QR factorization $A = QR$
2. Compute Q^Tb
3. Back substitution. Solve the triangular equation $Rx = Q^Tb$ using back substitution

对于普通矩阵使用 QR 分解求解线性方程组，正交分解需要 $2n^3$ flops, 计算 Q^Tb 需要 $2n^2$ flops, 第三步回代求解 $Rx = Q^Tb$ 需要 n^2 flops.(总时间复杂度是 $O(n^3)$).

对于稀疏矩阵求解线性方程组，时间复杂度接近 $\text{nnz}(A)$。内存使用和复杂度在很大程度上取决于系数矩阵的稀疏模式。内存使用量通常是 $\text{nnz}(A)+n$ 的适度倍数，$\text{nnz}(A)+n$ 是指定问题数据 A 和 b 所需的标量数量，通常远小于 n^2+n(如果 A 和 b 不是稀疏时存储它们所需的标量数)。求解稀疏线性方程的 flop 数通常也更接近 $\text{nnz}(A)$，而不是 n^3（矩阵 A 不稀疏时的阶数）。

11.3.2 QR 分解和求解伪逆 $A^{\dagger}$、逆 A^{-1}

Definition 11.3.1 — 线性无关列向量的矩阵 A 的伪逆.

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

Theorem 11.3.2

$$\begin{aligned} A^{\dagger} &= \left((QR)^T (QR)\right)^{-1} (QR)^T \\ &= \left(R^T Q^T QR\right)^{-1} R^T Q^T \\ &= \left(R^T R\right)^{-1} R^T Q^T \quad \left(Q^T Q = I\right) \\ &= R^{-1} R^{-T} R^T Q^T \quad (R\text{是非奇异的}) \\ &= R^{-1} Q^T \end{aligned}$$

Corollary 11.3.3 对于方阵非奇异矩阵 A, 其逆为

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

Corollary 11.3.4 对于方阵非奇异矩阵 $A = QR$

$$RA^{-1} = Q^T$$

Proof.

$$\begin{aligned} & A = QR \\ \Rightarrow & AA^{-1} = QRA^{-1} \\ \Rightarrow & I = Q \underbrace{RA^{-1}}_{Q^{-1}} \end{aligned}$$

■

Algorithm 9: Computing the inverse via QR factorization

Input: $n \times n$ invertible matrix A

1 QR factorization. Compute the QR factorization $A = QR$
2 **for** $i = 1, \ldots, n$ **do**
3 Solve the triangular equation $Rb_i = \tilde{q}_i$ using back substitution.
4 **end**

对于通过 QR 分解求 A^{-1}，QR 分解需要 $2n^3$ flops，n 次回代需要 n^3 flops，时间复杂度是 $O(3n^3)$.

11.3.3 A 的列空间和 Q 的列空间相同

矩阵 $A \in \mathbb{R}^{m \times n}$ 的值域范围定义为:

$$\text{range}(A) = \{Ax \mid x \in \mathbf{R}^n\}$$

Theorem 11.3.5 假设 A 有线性无关的列向量，且其 QR 因子为 Q, R，则 Q 和 A 的值域范围相同（有相同的列空间）。

即 Q 的列向量是标准正交的，并且和 A 的列向量张成相同的空间。

Proof.

$$\begin{aligned} y \in \text{range}(A) &\Leftrightarrow y = Ax, x \in \mathbb{R}^n \\ &\Leftrightarrow y = QRx, z = Rx \\ &\Leftrightarrow y = Qz, z \in \mathbb{R}^n \\ &\Leftrightarrow y \in \text{range}(Q) \end{aligned}$$

■

11.3.4 往 A 列空间上的投影也是往 Q 列空间上的投影

结合 $A = QR$ 和 $A^\dagger = R^{-1}Q^T$, 可得:

Theorem 11.3.6

$$AA^\dagger = QRR^{-1}Q^T = QQ^T$$

Theorem 11.3.7

$$R^{\mathrm{T}}R\hat{\boldsymbol{x}} = R^{\mathrm{T}}Q^{\mathrm{T}}\boldsymbol{b} \text{ or } R\hat{\boldsymbol{x}} = Q^{\mathrm{T}}\boldsymbol{b} \text{ or } \quad \hat{\boldsymbol{x}} = R^{-1}Q^{\mathrm{T}}\boldsymbol{b}$$

可以用于求解最小二乘法，而不用直接求解 $Ax = b$（当 A 不可逆时）

注意在 $AA^\dagger$ 中乘积的顺序与 $A^\dagger A = I$ 的差异。

$$\begin{aligned} &\min_y \|Qy - x\|_2^2 \\ \Rightarrow &Q^T(Qy - x) = 0 \\ \Rightarrow &Q^TQy = Q^Tx \\ \Rightarrow &y = Q^Tx \end{aligned}$$

QQ^Tx 是 x 在 Q 值域（列空间）上的投影，也是往 A 的列空间上的投影（见10.9.1）.

The pseudoinverse $A^\dagger$ is the n by m matrix that makes $AA^\dagger$ and $A^\dagger A$ into projections.

Trying for $AA^{-1} = A^{-1}A = I$, $AA^\dagger$ = projection matrix onto the column space of A (refer to projection onto the column space of A)

A^+A = projection matrix onto the row space of A

11.4 QR Algorithm Using Gram-Schmidt Algorithm

Gram-Schmidt QR 算法将逐列计算 Q 和 R.

Figure 11.1: Projecting onto the column space of A is also projecting onto the column space of Q

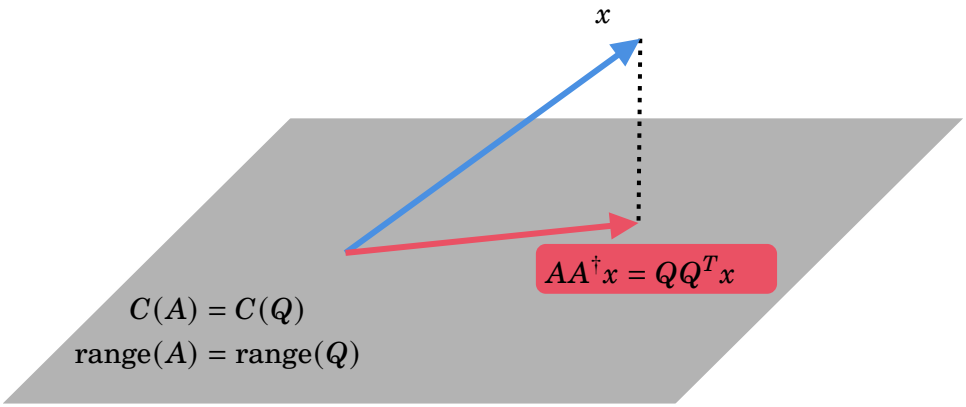
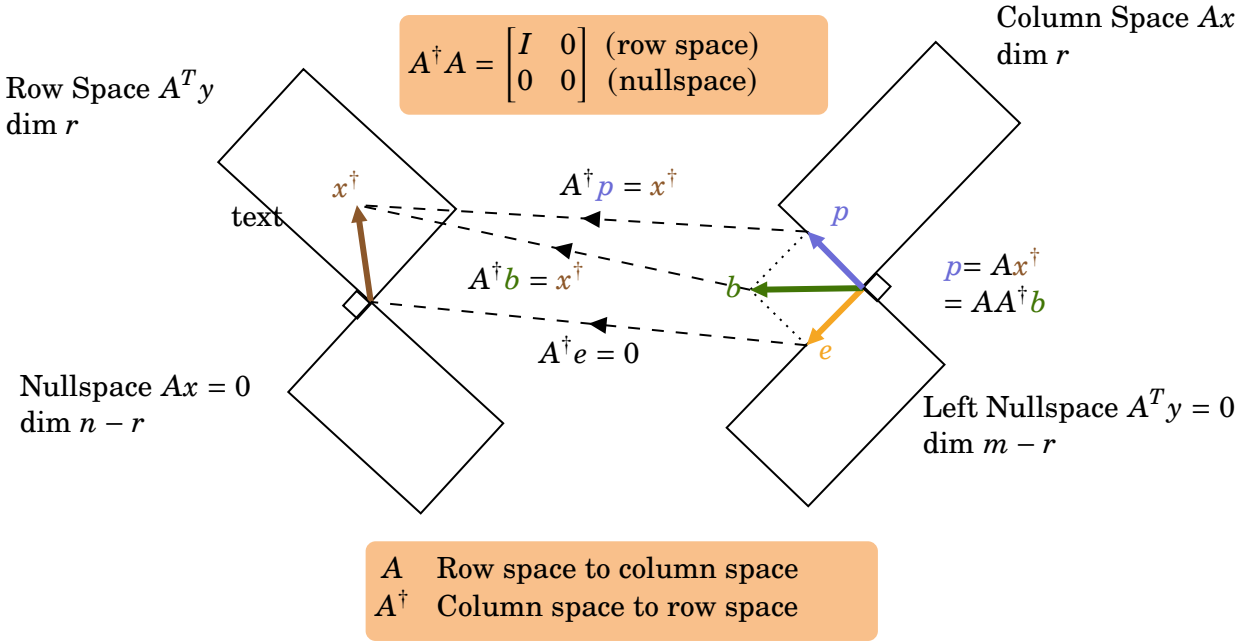


Figure 11.2: $A\boldsymbol{x}^{\dagger}$ in the column space goes back to $A^{\dagger}A\boldsymbol{x}^{\dagger} = \boldsymbol{x}^{\dagger}$ in the row space



k 步后我们得到了 QR 的部分分解：

$$A = \begin{bmatrix} a_1 & a_2 & \cdots & a_k \end{bmatrix} = \begin{bmatrix} q_1 & q_2 & \cdots & q_k \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & \cdots & R_{1k} \\ 0 & R_{22} & \cdots & R_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_{kk} \end{bmatrix}$$

Corollary 11.4.1 QR 的部分分解列向量 $q_1, \ldots, q_k$ 是标准正交的。

Corollary 11.4.2 对角线元素 $R_{11}, R_{22}, \ldots, R_{kk}$ 是正的。

Corollary 11.4.3 列向量 $q_1, \ldots, q_k$ 和 $a_1, \ldots, a_k$ 张成的空间相同。

Theorem 11.4.4 $A = QR$ 矩阵中的 R 矩阵为

$$R_{1k} = q_1^T a_k, R_{2k} = q_2^T a_k, \ldots, R_{k-1,k} = q_{k-1}^T a_k$$

Gram-Schmidt 正交化的两条基本公式

$$\widetilde{q}_i = a_i - \left(q_1^T a_i\right) q_1 - \cdots - \left(q_{i-1}^T a_i\right) q_{i-1}$$

$$\begin{aligned} a_i &= \left(q_1^T a_i\right) q_1 + \cdots + \left(q_{i-1}^T a_i\right) q_{i-1} + \underbrace{\|\tilde{q}_i\|_2 \, q_i}_{\widetilde{q}_i} \\ &= R_{1i} q_1 + \cdots + R_{ii} q_i \end{aligned}$$

Proof. 假设已经实现 $k-1$ 列的 QR 分解，方程 $A = QR$ 的第 k 列可以计算为：

$$a_k = R_{1k} q_1 + R_{2k} q_2 + \cdots + R_{k-1,k} q_{k-1} + R_{kk} q_k$$

无论如何选择 $R_{1k}, \ldots, R_{k-1,k}$，向量

$$\tilde{q}_k = a_k - R_{1k} q_1 - R_{2k} q_2 - \cdots - R_{k-1,k} q_{k-1} \neq 0$$

都将是非零的.（由于 $a_1, \ldots, a_k$ 是线性无关的，即 $q_1, \ldots, q_{i-1}, q_i$ 是线性无关的，假设 Gram-Schmidt 算法过程中没有出现中途退出的状况）

因此

$$a_k \notin \operatorname{span}\{q_1, \ldots, q_{k-1}\} = \operatorname{span}\{a_1, \ldots, a_{k-1}\}$$

q_k 是 $\tilde{q}_k$ 的单位化：选择 $R_{kk} = \|\tilde{q}_k\|_2$，以及 $q_k = \left(\frac{1}{R_{kk}}\right) \tilde{q}_k$ 。

$\tilde{q}_k$ 和 q_k 正交于 $q_1, \ldots, q_{k-1}$，则 $R_{1k}, \ldots, R_{k-1,k}$ 为：

$$R_{1k} = q_1^T a_k, R_{2k} = q_2^T a_k, \ldots, R_{k-1,k} = q_{k-1}^T a_k$$

■

Algorithm 10: QR Decomposition Using Gram-Schmidt Algorithm

Input: 矩阵 $A \in \mathbb{R}^{m\times n}$，列向量 $a_1, \ldots, a_n$ 线性无关
Output: 分解得到的 Q、R 矩阵

$R_{11} = \|a_1\|_2$
$q_1 = \frac{1}{R_{11}} a_1$
for $k = 2$ *to* n **do**
  for $l = 1$ *to* $k-1$ **do**
    $R_{l,k} = q_l^T a_k$
  end
  $\tilde{q}_k = a_k - \left(R_{1k}q_1 + R_{2k}q_2 + \cdots + R_{k-1,k}q_{k-1}\right)$
  $R_{kk} = \|\tilde{q}_k\|_2$
  $q_k = \dfrac{1}{R_{kk}}\tilde{q}_k$
end

11.4.1 Gram-Schmidt Algorithm

■ **Example 11.3**

$$\begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} = \begin{bmatrix} -1 & -1 & 1 \\ 1 & 3 & 3 \\ -1 & -1 & 5 \\ 1 & 3 & 7 \end{bmatrix} = \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ 0 & R_{22} & R_{23} \\ 0 & 0 & R_{33} \end{bmatrix}$$

Q 和 R 的第一列：

$$\tilde{q}_1 = a_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}, R_{11} = \|\tilde{q}_1\| = 2, q_1 = \frac{1}{R_{11}}\tilde{q}_1 = \begin{bmatrix} -1/2 \\ 1/2 \\ -1/2 \\ 1/2 \end{bmatrix}$$

Q 和 R 的第二列：计算得到 $R_{12} = q_1^T a_2 = 4$ 。
正交化计算：

$$\tilde{q}_2 = a_2 - R_{12}q_1 = \begin{bmatrix} -1 \\ 3 \\ -1 \\ 3 \end{bmatrix} - 4\begin{bmatrix} -1/2 \\ 1/2 \\ -1/2 \\ 1/2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

将其单位化得到：

$$R_{22} = \|\tilde{q}_2\| = 2, \quad q_2 = \frac{1}{R_{22}}\tilde{q}_2 = \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix}$$

Q 和 R 的第三列: 计算得到 $R_{13} = q_1^T a_3 = 2$ 以及 $R_{23} = q_2^T a_3 = 8$ 。

正交化计算: $\tilde{q}_3 = a_3 - R_{13}q_1 - R_{23}q_2 = \begin{bmatrix} 1 \\ 3 \\ 5 \\ 7 \end{bmatrix} - 2\begin{bmatrix} -1/2 \\ 1/2 \\ -1/2 \\ 1/2 \end{bmatrix} - 8\begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix} = \begin{bmatrix} -2 \\ -2 \\ 2 \\ 2 \end{bmatrix}$

将其单位化得到:

$$R_{33} = \|\tilde{q}_3\| = 4, \quad q_3 = \frac{1}{R_{33}}\tilde{q}_3 = \begin{bmatrix} -1/2 \\ -1/2 \\ 1/2 \\ 1/2 \end{bmatrix}$$

最终结果:

$$\begin{bmatrix} -1 & -1 & 1 \\ 1 & 3 & 3 \\ -1 & -1 & 5 \\ 1 & 3 & 7 \end{bmatrix} = \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ 0 & R_{22} & R_{23} \\ 0 & 0 & R_{33} \end{bmatrix}$$

$$= \begin{bmatrix} -1/2 & 1/2 & -1/2 \\ 1/2 & 1/2 & -1/2 \\ -1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & 1/2 \end{bmatrix} \begin{bmatrix} 2 & 4 & 2 \\ 0 & 2 & 8 \\ 0 & 0 & 4 \end{bmatrix}$$

■

11.4.2 基于 Gram-Schmidt 方法进行 QR 分解的时间复杂度

Gram-Schmidt 方法第 k 次循环的复杂度:

- a_k 有 k-1 个 $q_i^T a_k$ 内积操作: $(\mathrm{k}-1)(2\,\mathrm{m}-1)$ flops
- 计算 $\tilde{q}_k$: $2(\mathrm{k}-1)\mathrm{m}$ flops.

$$\widetilde{q}_i = a_i - \left(q_1^T a_i\right) q_1 - \cdots - \left(q_{i-1}^T a_i\right) q_{i-1}$$

- 计算 R_{kk} 和 q_k : 3 m flops。$R_{kk} = \|\tilde{q}_k\|_2, q_k = \tilde{q}_k / R_{kk}$

第 k 次循环的总和: $(4m-1)(k-1) + 3m$ flops

$A \in \mathbb{R}^{m\times n}$ 分解的复杂度:

$$\sum_{k=1}^{n}((4m-1)(k-1)+3m) = (4m-1)\frac{n(n-1)}{2} + 3mn$$

$$\approx 2mn^2 \text{ flops}$$

对于稀疏矩阵（空间存储小于 $m \times n$ ）的 QR 分解，时间复杂度可以低于 $O(2mn^2)$.

11.5 The Numerical Instability of QR Decomposition based on Gram-Schmidt Algorithm

Gram-Schmidt 算法复杂度为 $2mn^2$ flops. 在实际情况中**不推荐使用** (容易被舍入误差影响)。

修正 Gram-Schmidt 算法复杂度为 $2mn^2$ flops，有更好的数值计算性能。修正之处在于求解投影时每次只减去一个投影。

Householder 算法复杂度为 $2mn^2 - (2/3)n^3$ flops 。将 Q 表示为初等正交矩阵的乘积。是最广泛使用的算法 (在 MATLAB 和 Julia 中的 qr 函数使用该算法)。

本书中认为 QR 分解的复杂度为 $Q(2mn^2)$.

■ **Example 11.4**

```
[m,n]=size(A);
Q = zeros(m,n);
R = zeros(n,n);
for k = 1:n
    R(1:k-1,k)=Q(:,1:k-1)'*A(:,k);
    v= A(:,k)-Q(:,1:k-1)*R(1:k-1,k);
    R(k,k) = norm(v);
    Q(:,k)=v/R(k,k);
end

```

Listing 11.1: Gram Schmidt

Algorithm 11: Gram-Schmidt 的 MATLAB 算法

Input: 矩阵 A

Output: QR 分解得到的矩阵 Q、R

for $k = 1$ *to* n **do**

  $R_{\mathrm{jk}} = q_j^T a_k, j < k, j = 1, \ldots, k-1$

  $v = \tilde{q}_k = a_k - Q_{:,1:k-1} R_{1:k-1,k}$

  $R_{\mathrm{kk}} = \|\tilde{q}_k\|_2$

  $q_k = \left(\frac{1}{R_{kk}}\right) \tilde{q}_k$

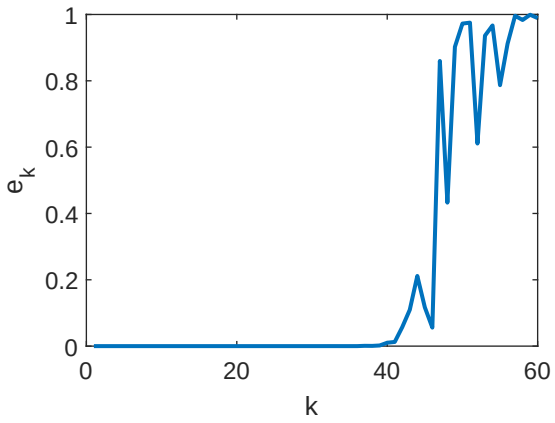
end

构造一个矩阵 $A = USV$, 其中 U 和 V 是正交矩阵，S 是对角矩阵

$$S_{ii} = 10^{-10(i-1)/(n-1)}, \quad i = 1, \cdots, n$$

把 Gram-Schmidt 算法应用到一个大小为 m = n = 50 的方形矩阵 A 上。

Figure 11.3: $\max_{1 \le i < k} \left| q_i^T q_k \right|$ (Gram-Schmidt Algorithm)



图中显示了 q_k 与前面列之间的正交性的偏差:

$$e_k = \max_{1 \le i < k} \left| q_i^T q_k \right|, \quad k = 2, \ldots, n$$

失去正交性是由于**浮点数存储的舍入误差**。

```
for j = 1:n
    v=A(:,j);
    for i=1:j-1
        R(i,j)=Q(:,i)'*v;
        v=v-R(i,j)*Q(:,i);
    end
    R(j,j)= norm(v);
    Q(:,j)=v/R(j,j);
end

```

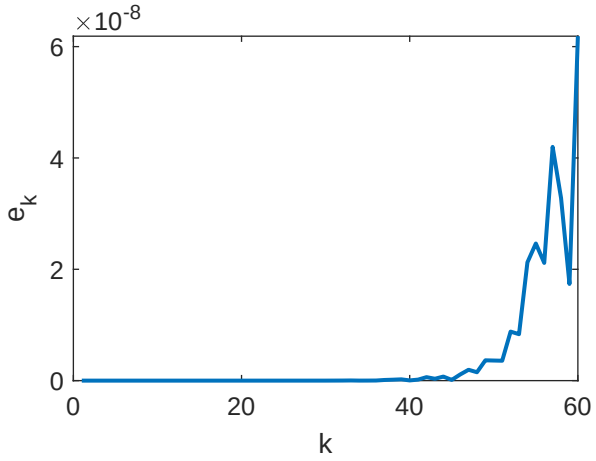
Listing 11.2: modified Gram-Schmidt

Algorithm 12: Modified Gram-Schmidt Algorithm

Input: 矩阵 A
Output: QR 分解得到的矩阵 Q、R

for $j = 1$ *to* n **do**
  $v = \tilde{q}_k = a_k$
  for $i = 1$ *to* $j-1$ **do**
    $R_{ij} = q_i^T v$
    $v = v - R_{ij} q_i$
  end
  $R_{\mathrm{kk}} = \|\tilde{q}_k\|_2$
  $q_k = \left(\frac{1}{R_{kk}}\right) \tilde{q}_k$
end

Figure 11.4: 在同一组数据中使用修正 Gram-Schmidt 算法求得的误差，$e_k = \max_{1 \le i < k} \left| q_i^T q_k \right|, k = 2, \ldots, n$



修正 Gram-Schmidt 算法的误差更小。 ∎

11.6 QR Decomposition Using Householder Transformation

Householder 算法是 QR 分解常用的算法 (MATLAB 和 Julia 中的 qr 函数)。与 Gram-Schmidt 相比，对舍入误差更有鲁棒性。

Householder 算法计算一个“完整的”QR 分解：

$$A_{m\times n} = \begin{bmatrix} Q_{m\times n} & \tilde{Q}_{m\times(m-n)} \end{bmatrix} \begin{bmatrix} R_{n\times n} \\ 0_{(m-n)\times n} \end{bmatrix}, \quad \begin{bmatrix} Q & \tilde{Q} \end{bmatrix} \text{是列正交的矩阵}$$

Proof.

$$\begin{aligned} A &= \begin{bmatrix} Q & \tilde{Q} \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix} \\ &= QR + \tilde{Q}0 \\ &= QR \end{aligned}$$

■

where R is an $n \times n$ upper triangular matrix, 0 is an $(m-n) \times n$ zero matrix, Q is $m \times n$, $\tilde{Q}$ is $m \times (m-n)$, and Q and $\tilde{Q}$ both have orthogonal columns.

$A \in R^{m\times n}, m \geq n$，当 $m < n$ 时列线性相关，无法进行 QR 分解。

完整的 Q 因子被构造成正交矩阵的乘积：

$$\begin{bmatrix} Q & \tilde{Q} \end{bmatrix} = H_1 H_2 \cdots H_n$$

每个 $H_i \in \mathbb{R}^{m\times m}$ 是对称正交的 Householder 矩阵。

11.6.1 Householder Matrix

Theorem 11.6.1 $H = I - 2vv^T$, 其中 $\|v\|_2 = 1$, Hx 是 x 关于超平面 $\{u \mid v^T u = 0\}$ 反对称.

H 是对称的

$$H^T = H$$

H **是正交的**

$$H^T H = I$$

Proof.

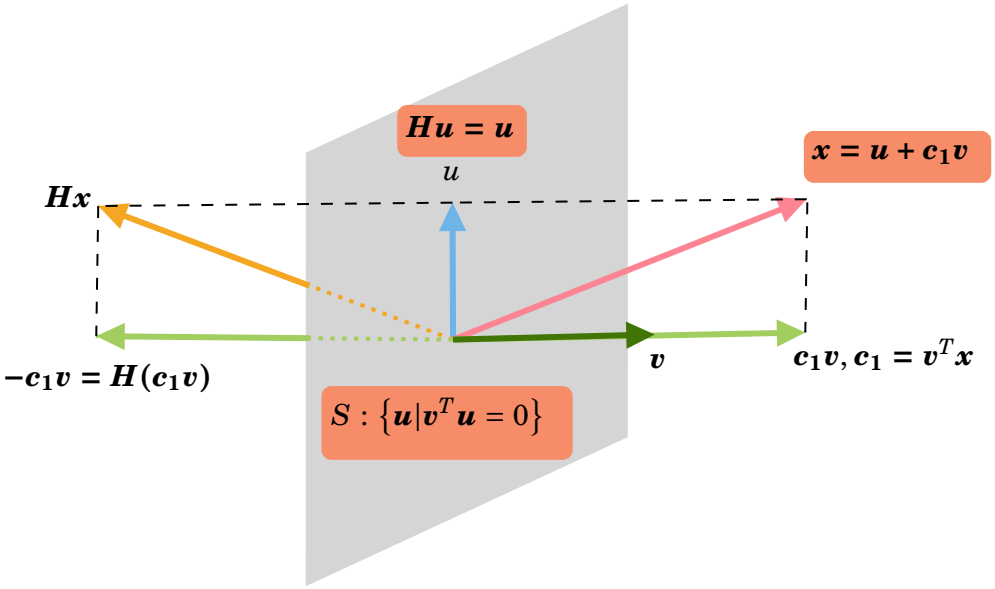
$$\begin{aligned} Hv &= (I - 2vv^T)(c_1 v) \\ &= c_1 v - 2vv^T c_1 v \\ &= c_1 v - 2c_1 vv^T v \\ &= -c_1 v \end{aligned}$$

$$\begin{aligned} Hu &= (I - 2vv^T)u \\ &= u - 2vv^T u \\ &= u \quad (v^T u = 0) \end{aligned}$$

$$\begin{aligned} Hx &= H(u + c_1 v) \\ &= u - c_1 v \\ &= x - 2(v^T x)v \end{aligned}$$

■

Figure 11.5: Reflection of x



Theorem 11.6.2 矩阵向量积 Hx 能化简为

$$Hx = x - 2\left(v^T x\right) v$$

Hx 的算法复杂度

如果 v 和 x 的长度是 p, 复杂度是 $4p$ flops 。

11.6.2 构造反射算子

给定非零 p 维向量 $y = (y_1, y_2, \ldots, y_p)$, 定义

Definition 11.6.1

$$w = \begin{bmatrix} y_1 + \operatorname{sign}(y_1)\|y\|_2 \\ y_2 \\ \vdots \\ y_p \end{bmatrix}$$

$$v = \frac{1}{\|w\|_2} w$$

$\operatorname{sign}(x)$ 是符号函数，$\operatorname{sign}(0) = 0$。

Theorem 11.6.3 向量 w 满足

$$\|w\|_2^2 = 2y^T w$$

$$\begin{aligned} \|w\|_2^2 &= w^T w \\ &= 2\left(\|y\|_2^2 + |y_1|\,\|y\|_2\right) \\ &= 2y^T\left(y + \operatorname{sign}(y_1)\|y\|_2 e_1\right) \\ &= 2y^T w \end{aligned}$$

Definition 11.6.2 — Constructed Householder Matrix. 将 y 变换为单位基向量 $e = [1, 0, 0, \cdots, 0]^T$ 乘以一个常数的 Householder 矩阵为

$$H = I - 2\frac{ww^T}{\|w\|_2^2}$$

Proof.

$$H = I - 2vv^T = I - 2\frac{ww^T}{\|w\|_2^2} \quad (v = \frac{1}{\|w\|_2}w)$$

■

Theorem 11.6.4 Householder 矩阵 $H = I - 2vv^T = I - 2\frac{ww^T}{\|w\|_2^2}$ 将 y 映射为

$$Hy = \begin{bmatrix} -\operatorname{sign}(y_1)\|y\|_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = r_1$$

Proof.

$$\begin{aligned} Hy &= y - \frac{2(w^T y)}{\|w\|_2^2}w \\ &= y - w \\ &= -\operatorname{sign}(y_1)\|y\|_2 e_1 \\ &= \begin{bmatrix} -\operatorname{sign}(y_1)\|y\|_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \end{aligned}$$

■

即 Householder 变换的结果 Hy 只保留第一个元素的值，其余元素的值变为 0. 又因为 H 是正交、对称的，它与 QR 分解求解 R 有一定联系。

构造的 Householder 矩阵几何意义

关于超平面 $\{x \mid w^T x = 0\}$, 其法向量 w, v:

$$w = y + \operatorname{sign}(y_1)\|y\|_2 e_1, v = \frac{w}{\|w\|_2}$$

反射算子 H 将 y 映射到向量 $-\operatorname{sign}(y_1)\|y\|_2 e_1$ 。

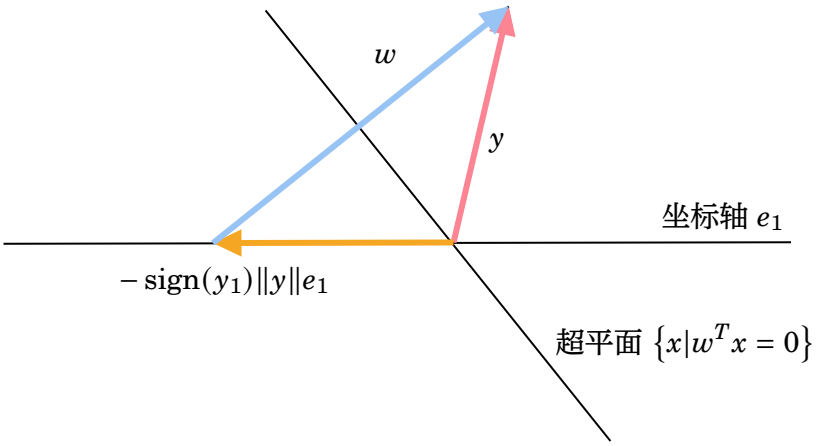
11.6.3 Householder 三角化

计算反射算子 $H_1, \ldots, H_n$ 将 A 简化为上三角矩阵形式:

$$H_n H_{n-1} \cdots H_1 A = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

第 k 个步骤之后，矩阵 $H_k H_{k-1} \ldots H_1 A$ 具有以下结构:
对于 $i > j$ 和 $j \le k$, 第 i,j 个位置的元素为零。

Figure 11.6: 构造的 Householder 矩阵的几何意义



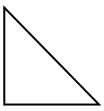
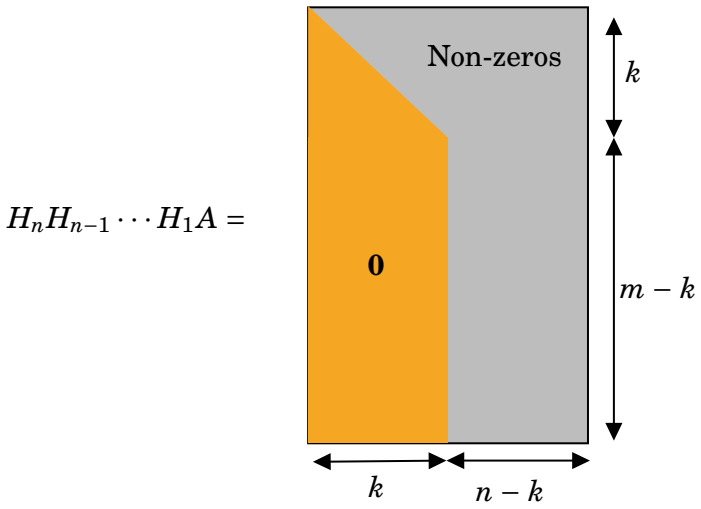


Figure 11.7: The structure of $H_kH_{k-1}\dots H_1A$



其过程如下：

$$A = \begin{bmatrix} X & X & X & X & X & X & X & X \\ X & X & X & X & X & X & X & X \\ X & X & X & X & X & X & X & X \\ X & X & X & X & X & X & X & X \\ X & X & X & X & X & X & X & X \\ X & X & X & X & X & X & X & X \\ X & X & X & X & X & X & X & X \\ X & X & X & X & X & X & X & X \end{bmatrix}$$

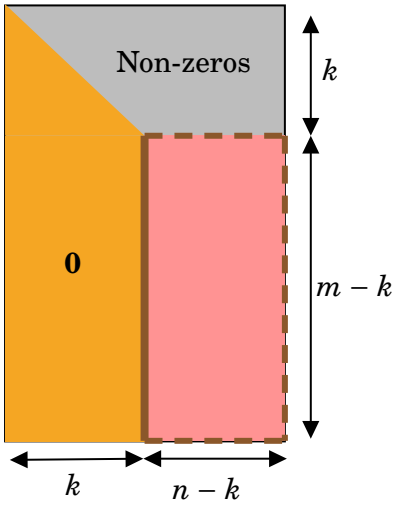
在第一次处理之后，A_1 第一列只剩下第一个元素不为 0.

$$H_1 A = A_1 = \begin{bmatrix} X & X & X & X & X & X & X & X \\ 0 & X & X & X & X & X & X & X \\ 0 & X & X & X & X & X & X & X \\ 0 & X & X & X & X & X & X & X \\ 0 & X & X & X & X & X & X & X \\ 0 & X & X & X & X & X & X & X \\ 0 & X & X & X & X & X & X & X \\ 0 & X & X & X & X & X & X & X \end{bmatrix}$$

第二次处理对于 $H_1 A_{2:m,1:n}$（$A_{1_{2:m,1:n}}$）进行处理。

$$H_2 A_1 = A_2 = \begin{bmatrix} X & X & X & X & X & X & X & X \\ 0 & X & X & X & X & X & X & X \\ 0 & 0 & X & X & X & X & X & X \\ 0 & 0 & X & X & X & X & X & X \\ 0 & 0 & X & X & X & X & X & X \\ 0 & 0 & X & X & X & X & X & X \\ 0 & 0 & X & X & X & X & X & X \\ 0 & 0 & X & X & X & X & X & X \end{bmatrix}$$

Figure 11.8: 迭代计算过程中 A 的结构



Algorithm 13: QR Decomposition Using Householder Transformation

Input: Matrix A

Output: $v_1, \cdots, v_n, v_k \in \mathbb{R}^{m-k+1}$ (Q), $A = \begin{bmatrix} R \\ 0 \end{bmatrix}$ (R)

1 **for** k *in* $1 : n$ **do**

2 令 $y = A_{k:m,k} \in \mathbb{R}^{m-k+1}$, 计算向量 v_k

$$w = y + \text{sign}\,(y_1)\,\|y\| e_1$$

$$v_k = \frac{1}{\|w\|} w$$

3 将 $A_{k:m,k:n} \in \mathbb{R}^{(m-k+1)\times(n-k+1)}$ 与反射矩阵 $I - 2v_k v_k^T$ 相乘

$$A_{k:m,k:n} := A_{k:m,k:n} - 2v_k \left(v_k^T A_{k:m,k:n}\right)$$

4 **end**

11.6.4 Household-QR Algorithm

Theorem 11.6.5 在算法步骤 2 中，将 $A_{k:m,k:n}$ 与反射算子 $I - 2v_k v_k^T$ 相乘

$$\left(I - 2v_k v_k^T\right) A_{k:m,k:n} = A_{k:m,k:n} - 2v_k \left(v_k^T A_{k:m,k:n}\right)$$

等价于用 $H_k \in \mathbb{R}^{m\times m}$ 乘以 $A \in \mathbb{R}^{m\times n}$

$$H_k = \begin{bmatrix} I & 0 \\ 0 & I - 2v_k v_k^T \end{bmatrix} = I - 2 \begin{bmatrix} 0 \\ v_k \end{bmatrix} \begin{bmatrix} 0 \\ v_k \end{bmatrix}^T$$

算法的最终结果将下列矩阵来代替 $A \in \mathbb{R}^{m\times n}$

$$\begin{bmatrix} R \\ 0 \end{bmatrix}$$

返回向量 $v_1, \ldots, v_n$, 其中 v_k 的长度为 $m - k + 1$ 。

11.6.5 An Example for Householder Algorithm

Problem 11.3

$$A = \begin{bmatrix} -1 & -1 & 1 \\ 1 & 3 & 3 \\ -1 & -1 & 5 \\ 1 & 3 & 7 \end{bmatrix} = H_1 H_2 H_3 \begin{bmatrix} R \\ 0 \end{bmatrix}$$

计算反射算子 H_1, H_2, H_3 来将矩阵 A 三角化

$$H_3 H_2 H_1 A = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ 0 & R_{22} & R_{23} \\ 0 & 0 & R_{33} \\ 0 & 0 & 0 \end{bmatrix}$$

R 的第一列：计算将 A 的第一列映射到 e_1 乘积的反射算子

$$y = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}, \quad w = y - \|y\|_2 e_1 = \begin{bmatrix} -3 \\ 1 \\ -1 \\ 1 \end{bmatrix}, \quad v_1 = \frac{1}{\|w\|_2} w = \frac{1}{2\sqrt{3}} \begin{bmatrix} -3 \\ 1 \\ -1 \\ 1 \end{bmatrix}$$

用 $I - 2v_1 v_1^T$ 和 A 的乘积代替 A:

$$A := \left(I - 2v_1 v_1^T\right) A = \begin{bmatrix} 2 & 4 & 2 \\ 0 & 4/3 & 8/3 \\ 0 & 2/3 & 16/3 \\ 0 & 4/3 & 20/3 \end{bmatrix}$$

R 的第二列：计算将 $A_{2:4,2}$ 映射到 e_1 乘积的反射算子

$$y = \begin{bmatrix} 4/3 \\ 2/3 \\ 4/3 \end{bmatrix}, \quad w = y + \|y\|_2 e_1 = \begin{bmatrix} 10/3 \\ 2/3 \\ 4/3 \end{bmatrix}, \quad v_2 = \frac{1}{\|w\|_2} w = \frac{1}{\sqrt{30}} \begin{bmatrix} 5 \\ 1 \\ 2 \end{bmatrix}$$

用 $I - 2v_2 v_2^T$ 和 $A_{2:4,2:3}$ 的乘积代替 $A_{2:4,2:3}$：

$$A := \begin{bmatrix} 1 & 0 \\ 0 & I - 2v_2 v_2^T \end{bmatrix} A = \begin{bmatrix} 2 & 4 & 2 \\ 0 & -2 & -8 \\ 0 & 0 & 16/5 \\ 0 & 0 & 12/5 \end{bmatrix}$$

R 的第三列：计算将 $A_{3:4,3}$ 映射到 e_1 乘积的反射算子

$$y = \begin{bmatrix} 16/5 \\ 12/5 \end{bmatrix}, \quad w = y + \|y\|_2 e_1 = \begin{bmatrix} 36/5 \\ 12/5 \end{bmatrix}, \quad v_3 = \frac{1}{\|w\|_2} w = \frac{1}{\sqrt{10}} \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

用 $I - 2v_3 v_3^T$ 和 $A_{3:4,3}$ 的乘积代替 $A_{3:4,3}$：

$$A := \begin{bmatrix} I & 0 \\ 0 & I - 2v_3 v_3^T \end{bmatrix} A = \begin{bmatrix} 2 & 4 & 2 \\ 0 & -2 & -8 \\ 0 & 0 & -4 \\ 0 & 0 & 0 \end{bmatrix}$$

总的求解式为

$$
\begin{aligned}
H_3H_2H_1A &= \begin{bmatrix} I_2 & 0 \\ 0 & I_2 - 2v_3v_3^T \end{bmatrix} \begin{bmatrix} I_1 & 0 \\ 0 & I_3 - 2v_2v_2^T \end{bmatrix} \left(I_4 - 2v_1v_1^T\right) A \\
&= \begin{bmatrix} I_2 & 0 \\ 0 & I_2 - 2v_3v_3^T \end{bmatrix} \begin{bmatrix} I_1 & 0 \\ 0 & I_3 - 2v_2v_2^T \end{bmatrix} \begin{bmatrix} 2 & 4 & 2 \\ 0 & 4/3 & 8/3 \\ 0 & 2/3 & 16/3 \\ 0 & 4/3 & 20/3 \end{bmatrix} \\
&= \begin{bmatrix} I_2 & 0 \\ 0 & I_2 - 2v_3v_3^T \end{bmatrix} \begin{bmatrix} 2 & 4 & 2 \\ 0 & -2 & -8 \\ 0 & 0 & 16/5 \\ 0 & 0 & 12/5 \end{bmatrix} \\
&= \begin{bmatrix} 2 & 4 & 2 \\ 0 & -2 & -8 \\ 0 & 0 & -4 \\ 0 & 0 & 0 \end{bmatrix}
\end{aligned}
$$

■

11.6.6 Complexity of Householder Algorithm

$$
H_k = \begin{bmatrix} I & 0 \\ 0 & I - 2v_kv_k^T \end{bmatrix} = I - 2 \begin{bmatrix} 0 \\ v_k \end{bmatrix} \begin{bmatrix} 0 \\ v_k \end{bmatrix}^T
$$

Householder 方法第 k 次循环的复杂度:

- $v_k^T A_{k:m,k:n}$ 的乘积：$(2(m-k+1)-1)(n-k+1)$ flops
- v_k 的外积：$(m-k+1)(n-k+1)$ flops
- $A_{k:m,k:n}$ 的减法：$(m-k+1)(n-k+1)$ flops

第 k 次循环的总和：$4(m-k+1)(n-k+1)$ flops

计算 R 和 $v_1, \ldots, v_n$ 的总复杂度

$$
\begin{aligned}
\sum_{k=1}^{n} 4(m-k+1)(n-k+\mathbf{2}) &\approx \int_0^n 4(m-t)(n-t+1)dt \\
&\approx 2mn^2 - \frac{2}{3}n^3 \text{ flops}
\end{aligned}
$$

$\sum_{k=1}^{n} 4(m-k+1)(n-k+\mathbf{2})$ 是因为

11.7 Householder 变换进行 QR 分解的 Q 因子

Householder 算法返回向量 $v_1, \ldots, v_n$, 其定义为:

Definition 11.7.1 — $v_1, \ldots, v_n$ **的完整表示.**

$$
\begin{bmatrix} Q & \tilde{Q} \end{bmatrix} = H_1H_2 \cdots H_n
$$

通常不需计算矩阵 $\begin{bmatrix} Q & \tilde{Q} \end{bmatrix}$。向量 $v_1, \ldots, v_n$ 是 $\begin{bmatrix} Q & \tilde{Q} \end{bmatrix}$ 简单表示 (economical representation)。

Theorem 11.7.1 $\begin{bmatrix} Q & \tilde{Q} \end{bmatrix}$ 或其转置的乘积可以计算为：

$$\begin{bmatrix} Q & \tilde{Q} \end{bmatrix} x = H_1 H_2 \cdots H_n x$$
$$\begin{bmatrix} Q & \tilde{Q} \end{bmatrix}^T y = H_n H_{n-1} \cdots H_1 y$$

11.7.1 Multiplication with Q factor

Definition 11.7.2 — **矩阵-向量积 $H_k x$.** 矩阵-向量积 $H_k x$ 定义为：

$$H_k x = \begin{bmatrix} I_{k-1} & 0 \\ 0 & I - 2v_k v_k^T \end{bmatrix} \begin{bmatrix} x_{1:k-1} \\ x_{k:m} \end{bmatrix} = \begin{bmatrix} x_{1:k-1} \\ x_{k:m} - 2\left(v_k^T x_{k:m}\right) v_k \end{bmatrix}$$

11.7.2 矩阵-向量积 $H_k x$ 算法复杂度

$H_k x$ 乘积的复杂度为：4(m − k + 1) flops。

$H_1 H_2, \ldots H_n$ 或其转置的乘积的复杂度为：

$$\sum_{k=1}^{n} 4(m - k + 1) \approx 4mn - 2n^2 \text{ flops}$$

其复杂度约等于 $m \times n$ 矩阵的矩阵-向量乘积 ($2mn$ flops)。

11.8 Fast Orthogonalization (Givens and Householder)

This section is quoted from [**Strang1993IntroductionTL**].

There are three ways to reach the important factorization $A = QR$.

Gram-Schmidt works to find the orthonormal vectors in Q. Then R is upper triangular because of the order of Gram-Schmidt steps.

Now we look at better methods (Householder and Givens), which use a product of specially simple Q 's that we know are orthogonal.

Elimination gives $A = LU$, orthogonalization gives $A = QR$. We don't want a triangular L, we want an orthogonal Q.L is a product of E' s from elimination, with 1's on the diagonal and the multiplier ℓ_{ij} below. Q **will be a product of orthogonal matrices.**

There are two simple orthogonal matrices to take the place of the E's. The *reflection matrices* $I - 2u\boldsymbol{u}^{\mathrm{T}}$ are named after *Householder*. The *plane rotation matrices* are named after *Givens*. The simple matrix that rotates the xy plane by θ is Q_{21}:

Definition 11.8.1 — **Givens Rotation in the 1-2 plane.**

$$Q_{21} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Use Q_{21} the way you used E_{21}, to produce a zero in the $(2,1)$ position. That determines the angle θ. Bill Hager gives this example in Applied Numerical Linear Algebra:

■ **Example 11.5 — 使用 Givens 进行 QR 分解.**

$$Q_{21}A = \begin{bmatrix} .6 & .8 & 0 \\ -.8 & .6 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 90 & -153 & 114 \\ 120 & -79 & -223 \\ 200 & -40 & 395 \end{bmatrix} = \begin{bmatrix} 150 & -155 & -110 \\ \mathbf{0} & 75 & -225 \\ 200 & -40 & 395 \end{bmatrix}$$

The zero came from $-8(90)+.6(120)$. No need to find θ, what we needed was $\cos\theta$

$$\cos\theta = \frac{90}{\sqrt{90^2+120^2}} \quad \text{and} \quad \sin\theta = \frac{-120}{\sqrt{90^2+120^2}}$$

Now we attack the $(3,1)$ entry. The rotation will be in rows and columns 3 and 1 . The numbers $\cos\theta$ and $\sin\theta$ are determined from 150 and 200 , instead of 90 and 120 .

$$Q_{31}Q_{21}A = \begin{bmatrix} .6 & 0 & .8 \\ 0 & 1 & 0 \\ -.8 & 0 & .6 \end{bmatrix} \begin{bmatrix} 150 & \cdot & \cdot \\ 0 & \cdot & \cdot \\ 200 & \cdot & \cdot \end{bmatrix} = \begin{bmatrix} 250 & -125 & 250 \\ \mathbf{0} & 75 & -225 \\ \mathbf{0} & 100 & 325 \end{bmatrix}$$

One more step to R. The $(3,2)$ entry has to go. The numbers $\cos\theta$ and $\sin\theta$ now come from 75 and 100 . The rotation is now in rows and columns 2 and 3:

$$Q_{32}Q_{31}Q_{21}A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & .6 & .8 \\ 0 & -.8 & .6 \end{bmatrix} \begin{bmatrix} 250 & -125 & \cdot \\ 0 & 75 & \cdot \\ 0 & 100 & \cdot \end{bmatrix} = \begin{bmatrix} 250 & -125 & 250 \\ \mathbf{0} & 125 & 125 \\ \mathbf{0} & \mathbf{0} & 375 \end{bmatrix}$$

We have reached the upper triangular R. What is Q? Move the plane rotations Q_{ij} to the other side to find $A = QR$-just as you moved the elimination matrices E_{ij} to the other side to find $A = LU$:

Theorem 11.8.1

$$Q_{32}Q_{31}Q_{21}A = R \quad \text{means} \quad A = \left(Q_{21}^{-1}Q_{31}^{-1}Q_{32}^{-1}\right)R = QR$$

The inverse of each Q_{ij} is Q_{ij}^{T} (rotation through $-\theta$). The inverse of E_{ij} was not an orthogonal matrix! LU **and** QR **are similar but** L **and** Q **are not the same.** ■

Householder reflections are faster than rotations because each one clears out a whole column below the diagonal. Watch how the first column a_1 of A becomes column r_1 of R :

■ **Example 11.6 — Reflection by** H_1.

$$H_1 = I - 2u_1u_1^T$$

$$H_1\boldsymbol{a}_1 = \begin{bmatrix} \|\boldsymbol{a}_1\| \\ 0 \\ \vdots \\ 0 \end{bmatrix} \text{ or } \begin{bmatrix} -\|\boldsymbol{a}_1\| \\ 0 \\ \vdots \\ 0 \end{bmatrix} = r_1$$

The length was not changed, and u_1 is in the direction of $a_1 - r_1$. We have $n-1$ entries in the unit vector $\boldsymbol{u}_1$ to get $n-1$ zeros in $\boldsymbol{r}_1$. (Rotations had one angle θ to get one zero.) When we reach column k, we have $n-k$ available choices in the unit vector u_k. This leads to $n-k$ zeros in $\boldsymbol{r}_k$. We just store the u's and $\boldsymbol{r}$'s to know the final Q and R :

Theorem 11.8.2 — H 的逆是它本身.

$$(H_{n-1}\dots H_1)\,A = R \quad \text{means} \quad A = (H_1\dots H_{n-1})\,R = QR$$

■

11.9 Recap：QR Decomposition

11.9.1 分治策略

求解线性方程组 $Ax = b$, 矩阵 $A \in \mathbb{R}^{n\times n}$ 分解成“结构简单”的矩阵相乘:

$$A = A_1 A_2 \cdots A_k$$

■ **Example 11.7** — 求解 k 个线性方程组 $A_1A_2\cdots A_kx = b$.

$$\begin{aligned} A_1\underbrace{(A_2\cdots A_kx)}_{z_1} &= b \\ A_2\underbrace{(A_3\cdots A_kx)}_{z_2} &= z_1 \\ &\cdots \\ A_{k-1}\underbrace{(A_kx)}_{z_{k-1}} &= z_{k-2} \\ A_kx &= z_{k-1} \end{aligned}$$

■

■ **Example 11.8** — QR 分解 $Ax = b$.

$$\begin{aligned} Qy &= b \\ Rx &= y \end{aligned}$$

■

通常**分解复杂度远大于求解复杂度**。

11.9.2 非奇异矩阵的 QR 分解

Theorem 11.9.1 任意非奇异矩阵 $A \in \mathbb{R}^{n\times n}$, 都可以进行 QR 分解。

Corollary 11.9.2 $Q \in \mathbb{R}^{n\times n}$ 是一个正交矩阵

Corollary 11.9.3 $R \in \mathbb{R}^{n\times n}$ 是一个上三角矩阵并且对角元素都为正数

11.9.3 使用 QR 分解求 A^{-1} 可以转换成 $R^{-1}Q^T$

Theorem 11.9.4 $A^{-1} = (QR)^{-1} = R^{-1}Q^{-1} = R^{-1}Q^T$

11.9.4 QR 分解求解 A^{-1}

计算非奇异矩阵 $\mathrm{A} \in \mathbb{R}^{n\times n}$ 的逆 A^{-1}, $X = [x_1, x_2, \cdots, x_n], x_i \in \mathbb{R}^n, i = 1, \cdots, n$, $I = [e_1, e_2, \cdots, e_n], e_i \in \mathbb{R}^n, i = 1, \cdots, n$:

Theorem 11.9.5 — QR 分解求解 A^{-1}.

$$Rx_1 = Q^T e_1, Rx_2 = Q^T e_2, \cdots, Rx_n = Q^T e_n$$

$$\begin{aligned} & AX = I \\ \Rightarrow & QRX = I \\ \Rightarrow & RX = Q^T I \\ \Rightarrow & Rx_1 = Q^T e_1, Rx_2 = Q^T e_2, \cdots, Rx_n = Q^T e_n \end{aligned}$$

The Complexity of $QRX = I$

复杂度: $2n^3 + n^3 \approx 3n^3$ flops

- QR 分解复杂度: $2n^3$
- 回代法：一次回代 n^2, 则 n 次回代 n^3

11.9.5 QR 分解求解线性方程组

使用 QR 分解求解线性方程组 $Ax = b$, 矩阵 $A \in \mathbb{R}^{n \times n}$ 为非奇异矩阵

Algorithm 14: QR 分解求解线性方程组

1. 首先对 A 进行 QR 分解，得到 $A = QR$
2. 计算 $y = Q^T b$
3. 通过回代法求解 $Rx = y$

The Complexity of Solving Linear Equation Systems Using QR Decomposition

复杂度: $2n^3 + 3n^2 \approx 2n^3$ flops

- QR 分解复杂度: $2n^3$
- 矩阵向量乘法: $2n^2$
- 回代法：n^2

12. LU 分解

12.1 Solving Linear Equation Systems

12.1.1 Linear Equation Systems

- **Example 12.1**

$$A\boldsymbol{x}=\boldsymbol{b} \Leftrightarrow \begin{array}{rrrrrrr} & x+ & 2 & y+ & 3 & z= & 6 \\ 2 & x+ & 5 & y+ & 2 & z= & 4 \\ 6 & x- & 3 & y+ & & z= & 2 \end{array}$$

■

见 Row Picture，Column Picture 的概念。

12.1.2 Elimination

- **Example 12.2 — Elimination.** Before

$$\begin{array}{rrrrrr} & x- & 2 & y & = & 1 \\ 3 & x+ & 2 & y & = & 11 \end{array}$$

After

$$\begin{array}{rrrrr} x- & 2 & y & = & 1 \\ & 8 & y & = & 8 \end{array}$$

■

Definition 12.1.1 — Pivot. The first nonzero in the row that does elimination. **Zero is not allowed as a pivot.**

Definition 12.1.2 — Multiplier. (Entry to eliminate) divided by (pivot)

消元法通过 elimination matrices E 进行消元操作，使得主对角线以下的元素为 0. Multiply the j^{th} equation by ℓ_{ij} and subtract from the i^{th} equation. (This eliminates x_j from equation i.) We need a lot of these simple matrices E_{ij}, one for every nonzero to be eliminated below the main diagonal.

Definition 12.1.3 — Elementary matrix, Elimination matrix. The elementary matrix or elimination matrix E_{ij} has the extra nonzero entry $-\ell$ in the i, j position. Then E_{ij} subtracts a multiple ℓ of row j from row i.

Theorem 12.1.1 消元法的本质是

$$Ax = b \Rightarrow EAx = Eb$$

Definition 12.1.4 — Permutation matrices P. P_{ij} is the identity matrix with rows i and j reversed. When this *permutation matrix* P_{ij} multiplies a matrix, it exchanges rows i and j.

Definition 12.1.5 — Augmented matrix.

$$\begin{bmatrix} A & b \end{bmatrix}$$

Computing A^{-1} by Gauss-Jordan Elimination

$$\text{Multiply } \begin{bmatrix} A & I \end{bmatrix} \text{ by } A^{-1} \text{ to get } \begin{bmatrix} I & A^{-1} \end{bmatrix}$$

■ **Example 12.3**

$$\begin{bmatrix} K & e_1 & e_2 & e_3 \end{bmatrix} = \begin{bmatrix} \mathbf{2} & \mathbf{-1} & \mathbf{0} & 1 & 0 & 0 \\ \mathbf{-1} & \mathbf{2} & \mathbf{-1} & 0 & 1 & 0 \\ \mathbf{0} & \mathbf{-1} & \mathbf{2} & 0 & 0 & 1 \end{bmatrix} \quad \text{Start Gauss-Jordan on } K$$

$$\rightarrow \begin{bmatrix} 2 & -1 & 0 & 1 & 0 & 0 \\ \mathbf{0} & \frac{3}{2} & \mathbf{-1} & \frac{1}{2} & \mathbf{1} & \mathbf{0} \\ 0 & -1 & 2 & 0 & 0 & 1 \end{bmatrix} \quad \left(\frac{\mathbf{1}}{\mathbf{2}} \text{ row } 1 + \text{ row } \mathbf{2}\right)$$

$$\rightarrow \begin{bmatrix} 2 & -1 & 0 & 1 & 0 & 0 \\ 0 & \frac{3}{2} & -1 & \frac{1}{2} & 1 & 0 \\ \mathbf{0} & \mathbf{0} & \frac{4}{3} & \frac{1}{3} & \frac{2}{3} & \mathbf{1} \end{bmatrix} \quad \left(\frac{\mathbf{2}}{3} \text{ row } \mathbf{2} + \text{row } 3\right)$$

$$\begin{pmatrix} \text{Zero above} \\ \text{third pivot} \end{pmatrix} \rightarrow \begin{bmatrix} 2 & -1 & 0 & 1 & 0 & 0 \\ 0 & \frac{3}{2} & 0 & \frac{3}{4} & \frac{3}{2} & \frac{3}{4} \\ 0 & 0 & \frac{4}{3} & \frac{1}{3} & \frac{2}{3} & 1 \end{bmatrix} \quad \left(\frac{3}{4} \text{ row } \mathbf{3} + \text{ row } \mathbf{2}\right)$$

$$\begin{pmatrix} \text{Zero above} \\ \text{second pivot} \end{pmatrix} \rightarrow \begin{bmatrix} 2 & 0 & 0 & \frac{3}{2} & 1 & \frac{1}{2} \\ 0 & \frac{3}{2} & 0 & \frac{3}{4} & \frac{3}{2} & \frac{3}{4} \\ 0 & 0 & \frac{4}{3} & \frac{1}{3} & \frac{2}{3} & 1 \end{bmatrix} \quad \left(\frac{2}{3} \text{ row } 2 + \text{ row } 1\right)$$

它变成 Reduced echelon form R.

$$\begin{matrix} (divided\ by\ 2) \\ (divided\ by\ \frac{3}{2}) \\ (divided\ by\ \frac{4}{3}) \end{matrix} \begin{bmatrix} 1 & 0 & 0 & \frac{3}{4} & \frac{1}{2} & \frac{1}{4} \\ 0 & 1 & 0 & \frac{1}{2} & 1 & \frac{1}{2} \\ 0 & 0 & 1 & \frac{1}{4} & \frac{1}{2} & \frac{3}{4} \end{bmatrix} = \begin{bmatrix} I & x_1 & x_2 & x_3 \end{bmatrix} = \begin{bmatrix} I & K^{-1} \end{bmatrix}$$

1. K is symmetric across its main diagonal. Then K^{-1} is also symmetric.
2. K is tridiagonal (only three nonzero diagonals). But K^{-1} is a dense matrix with no zeros. That is another reason we don't often compute inverse matrices. The inverse of a band matrix is generally a dense matrix.
3. The product of pivots is $2\left(\frac{3}{2}\right)\left(\frac{4}{3}\right) = 4$. This number 4 is the determinant of K.

K^{-1} involves division by the determinant of K $\quad K^{-1} = \frac{1}{4}\begin{bmatrix} 3 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 3 \end{bmatrix}$.

This is why an invertible matrix cannot have a zero determinant: we need to divide.

■

12.2 LU 分解

$(E_{32}E_{31}E_{21})\,A = U$ becomes $A = \left(E_{21}^{-1}E_{31}^{-1}E_{32}^{-1}\right)U$ which is $A = LU$

Theorem 12.2.1 When a row of A starts with zeros, so does that row of L. When a column of A starts with zeros, so does that column of U.

■ **Example 12.4 — The key reason why A equals LU.** Ask yourself about the pivot rows that are subtracted from lower rows. Are they the original rows of A? No, elimination probably changed them.

Are they rows of U? Yes, the pivot rows never change again.

When computing the third row of U, we subtract multiples of earlier rows of U (not rows of A!):

$$\text{Row 3 of } U = (\text{Row 3 of } A) - \ell_{31}(\text{Row 1 of } U) - \ell_{32}(\text{Row 2 of } of\, U)$$

Rewrite this equation to see that the row $\begin{bmatrix} \ell_{31} & \ell_{32} & 1 \end{bmatrix}$ is multiplying the matrix U :

$$(\text{Row 3 of } A) = \ell_{31}(\text{Row 1 of } U) + \ell_{32}(\text{Row 2 of } U) + 1(\text{Row 3 of } U)$$

This is exactly row 3 of $A = LU$.

That row of L holds $\ell_{31}, \ell_{32}, 1$. All rows look like this, whatever the size of A. With no row exchanges, we have $A = LU$. ■

Definition 12.2.1 — A 的 LU 分解.

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1k} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots & & \vdots \\ a_{k1} & \cdots & a_{kk} & \cdots & a_{kn} \\ \vdots & & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nk} & \cdots & a_{nn} \end{pmatrix} = LU$$

$$\text{where } L = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \ddots & 0 \\ \vdots & \vdots & \ddots & 0 \\ l_{n1} & l_{n2} & \cdots & 1 \end{pmatrix}, U = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & 0 & u_{nn} \end{pmatrix}$$

所以

$$A=\begin{pmatrix} a_{11} & \cdots & a_{1r} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots & & \vdots \\ a_{r1} & \cdots & a_{rr} & \cdots & a_{rn} \\ \vdots & & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nr} & \cdots & a_{nn} \end{pmatrix}$$
$$=\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & 0 & \ddots & 0 \\ l_{r1} & \cdots & 1 & \ddots & \vdots \\ \vdots & & \vdots & \ddots & 0 \\ l_{n1} & \cdots & l_{nr} & \cdots & 1 \end{pmatrix}\cdot\begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & 0 & u_{nn} \end{pmatrix}$$

12.2.1 $A=LDU$

$A=LU$ 是不对称的。但是可以改写为对称形式。

Split U into $\begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{bmatrix}\begin{bmatrix} 1 & u_{12}/d_1 & u_{13}/d_1 & \cdot \\ & 1 & u_{23}/d_2 & \cdot \\ & & \ddots & \vdots \\ & & & 1 \end{bmatrix}$.

■ **Example 12.5**

$$(A=LU)\begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix}\begin{bmatrix} 2 & 8 \\ 0 & 5 \end{bmatrix}$$

splits further into

$$(A=LDU)\begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix}\begin{bmatrix} 2 & \\ & 5 \end{bmatrix}\begin{bmatrix} 1 & 4 \\ 0 & 1 \end{bmatrix}$$

.

■

Theorem 12.2.2 当 A 是对称矩阵的时候，且消元的时候不需要行交换：
$S=LDL^T$

■ **Example 12.6**

$$\begin{bmatrix} 1 & 2 \\ 2 & 7 \end{bmatrix}=\begin{bmatrix} \mathbf{1} & 0 \\ \mathbf{2} & \mathbf{1} \end{bmatrix}\quad\begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}\quad\begin{bmatrix} \mathbf{1} & \mathbf{2} \\ 0 & \mathbf{1} \end{bmatrix}$$

■

12.2.2 L、U 矩阵的性质

根据矩阵的乘法原理, 有

Theorem 12.2.3 A 的第一行元素 a_{1j} 为

$$a_{1j}=u_{1j}, j=1,\cdots,n$$

Corollary 12.2.4 U 的第一行元素 u_{1j} 为

$$u_{1j} = a_{1j}, j = 1, \cdots, n$$

Proof.

$$A = \begin{pmatrix} \boldsymbol{a_{11}} & \cdots & \boldsymbol{a_{1r}} & \cdots & \boldsymbol{a_{1n}} \\ \vdots & \ddots & \vdots & & \vdots \\ a_{r1} & \cdots & a_{rr} & \cdots & a_{rn} \\ \vdots & & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nr} & \cdots & a_{nn} \end{pmatrix}$$

$$= \begin{pmatrix} \boldsymbol{1} & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & 0 & \ddots & 0 \\ l_{r1} & \cdots & 1 & \cdots & \vdots \\ \vdots & \ddots & \vdots & \ddots & 0 \\ l_{n1} & \cdots & l_{nr} & \cdots & 1 \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{u_{11}} & \cdots & \boldsymbol{u_{1r}} & \cdots & \boldsymbol{u_{1n}} \\ 0 & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & u_{rr} & \cdots & u_{rn} \\ \vdots & \ddots & 0 & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & u_{nn} \end{pmatrix}$$

■

Corollary 12.2.5 L 的第一列元素 l_{i1} 为

$$l_{i1} = \frac{a_{i1}}{u_{11}}, i = 2, 3, \cdots, n$$

Proof.

$$A = \begin{pmatrix} \boldsymbol{a_{11}} & \cdots & a_{1r} & \cdots & a_{1n} \\ \boldsymbol{\vdots} & \ddots & \vdots & & \vdots \\ \boldsymbol{a_{r1}} & \cdots & a_{rr} & \cdots & a_{rn} \\ \boldsymbol{\vdots} & & \vdots & \ddots & \vdots \\ \boldsymbol{a_{n1}} & \cdots & a_{nr} & \cdots & a_{nn} \end{pmatrix}$$

$$= \begin{pmatrix} \boldsymbol{1} & 0 & 0 & \cdots & 0 \\ \boldsymbol{\vdots} & \ddots & 0 & \ddots & 0 \\ \boldsymbol{l_{r1}} & \cdots & 1 & \ddots & \vdots \\ \boldsymbol{\vdots} & & \vdots & \ddots & 0 \\ \boldsymbol{l_{n1}} & \cdots & l_{nr} & \cdots & 1 \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{u_{11}} & \cdots & u_{1r} & \cdots & u_{1n} \\ 0 & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & u_{rr} & \cdots & u_{rn} \\ \vdots & \ddots & 0 & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & u_{nn} \end{pmatrix}$$

■

Theorem 12.2.6 A 的第 r 行主对角线以右元素元素 $a_{1j}(j = 1, \cdots, n)$ 为

$$a_{rj} = \sum_{k=1}^{r} l_{rk} u_{kj}, r = 1, 2, \cdots, n, j = r, \cdots, n$$

Proof.

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1r} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots & & \vdots \\ a_{r1} & \cdots & \boldsymbol{a_{rr}} & \cdots & \boldsymbol{a_{rn}} \\ \vdots & & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nr} & \cdots & a_{nn} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & 0 & \ddots & 0 \\ l_{r1} & \cdots & \boldsymbol{1} & \cdots & 0 \\ \vdots & \cdots & \vdots & \ddots & 0 \\ l_{n1} & \cdots & l_{nr} & \cdots & 1 \end{pmatrix} \cdot \begin{pmatrix} u_{11} & \cdots & u_{1r} & \cdots & u_{1n} \\ 0 & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \boldsymbol{u_{rr}} & \cdots & \boldsymbol{u_{rn}} \\ \vdots & \ddots & 0 & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & u_{nn} \end{pmatrix}$$

■

Corollary 12.2.7 U 第 r 行主对角线以右元素 u_{rj}

$$u_{rj} = a_{rj} - \sum_{k=1}^{r-1} l_{rk} u_{kj}, j = r, \cdots, n$$

Proof.

$$a_{rj} = \sum_{k=1}^{r} l_{rk} u_{kj}, r = 1, 2, \cdots, n, j = r, \cdots, n$$

$$\Rightarrow a_{rj} = \sum_{k=1}^{r-1} l_{rk} u_{kj} + l_{rr} u_{rj}, r = 1, 2, \cdots, n, j = r, \cdots, n$$

$$\Rightarrow u_{rj} = a_{rj} - \sum_{k=1}^{r-1} l_{rk} u_{kj}, j = r, \cdots, n$$

■

Corollary 12.2.8 U 的对角线元素 u_{rr}

$$u_{rr} = a_{rr} - \sum_{k=1}^{r-1} l_{rk} u_{kr}$$

Theorem 12.2.9 A 的第 r 列元素主对角线以下元素 $a_{ir} (i = r+1, \cdots, n)$ 为

$$a_{ir} = \sum_{k=1}^{r} l_{ik} u_{kr}, i = r+1, \cdots, n, r = 1, 2, \cdots, n-1$$

Proof.

$$A=\begin{pmatrix} a_{11} & \cdots & a_{1r} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots & & \vdots \\ a_{r1} & \cdots & \boldsymbol{a_{rr}} & \cdots & a_{rn} \\ \vdots & & \vdots & \ddots & \vdots \\ a_{n1} & \cdots & \boldsymbol{a_{nr}} & \cdots & a_{nn} \end{pmatrix}$$

$$=\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & 0 & \ddots & 0 \\ l_{r1} & \cdots & \boldsymbol{1} & \cdots & 0 \\ \vdots & \cdots & \vdots & \ddots & 0 \\ l_{n1} & \cdots & \boldsymbol{l_{nr}} & \cdots & 1 \end{pmatrix} \cdot \begin{pmatrix} u_{11} & \cdots & u_{1r} & \cdots & u_{1n} \\ 0 & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \boldsymbol{u_{rr}} & \cdots & u_{rn} \\ \vdots & \ddots & 0 & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & u_{nn} \end{pmatrix}$$

■

Corollary 12.2.10 显然, $r=1$ 时

$$a_{i1}=l_{i1}u_{11}, i=2,3,\cdots,n$$

Corollary 12.2.11 L 第 r 列主对角线以下元素 l_{ir}

$$l_{ir}=\frac{a_{ir}-\sum_{k=1}^{r-1}l_{ik}u_{kr}}{u_{rr}}, i=r+1,\cdots,n$$

12.2.3 Solving $Ax=b$ Using LU Decomposition and its Complexity

求解 $Ax=b, A$ 为非奇异矩阵，LU 算法为求解方程组 $Ax=b$ 的标准解法.

复杂度: $\frac{2}{3}n^3+2n^2\approx\frac{2}{3}n^3$ flops

Algorithm 15: Solving $Ax=b$ Using LU Decomposition

1. 对矩阵 A 进行 LU 分解 $\left(\frac{2}{3}n^3 \text{ flops }\right)$
2. 回代法: 求解 $Ly=b$ $\left(n^2 \text{ flops }\right)$
3. 回代法: 求解 $Ux=y$ $\left(n^2 \text{ flops }\right)$

12.2.4 Example of LU Decomposition

- **Example 12.7** 对矩阵 A 进行 LU 分解

$$A=\begin{bmatrix} 8 & 2 & 9 \\ 4 & 9 & 4 \\ 6 & 7 & 9 \end{bmatrix}$$

$$A=\begin{bmatrix} 8 & 2 & 9 \\ 4 & 9 & 4 \\ 6 & 7 & 9 \end{bmatrix}=\begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix}\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

计算 U 的第一行和 L 的第一列

$$(u_{11}, u_{12}, u_{13}) = (8, 2, 9), (l_{21}, l_{31}) = \left(\frac{1}{2}, \frac{3}{4}\right)$$

然后计算 U 的第二行和 L 的第二列

$$u_{22} = a_{22} - l_{21}u_{12} = 8, u_{23} = a_{23} - l_{21}u_{13} = -\frac{1}{2}, l_{32} = \frac{a_{32} - l_{31}u_{12}}{u_{22}} = \frac{11}{16}$$

最后计算 U 的第三行

$$u_{33} = a_{33} - l_{31}u_{13} - l_{32}u_{23} = -\frac{83}{32}$$

■

12.3 Problem of LU Decomposition

■ **Example 12.8**

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 2 \\ 0 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & L_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

计算 U 的第一行和 L 的第一列

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 2 \\ 0 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & l_{32} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

然后计算 U 的第二行和 L 的第二列

$$\begin{matrix} u_{22} = a_{22} - l_{21}u_{12} = 0 \\ u_{23} = a_{23} - l_{21}u_{13} = 2 \end{matrix} \quad l_{32} = \frac{a_{32} - l_{31}u_{12}}{u_{22}} = \frac{1}{\mathbf{0}}$$

即该矩阵无法 LU 分解！ ■

12.4 $PA = LU$

Theorem 12.4.1 非奇异矩阵 $\mathrm{A} \in \mathbb{R}^{n \times n}$，则可分解为 $A = P^T LU$
P 是一个置换矩阵，L 为下三角矩阵并且对角线元素全为 1，U 为上三角矩阵

$PA = LU$ 分解方法不唯一，随着 P 的选择不同，L、U 也不同。

■ **Example 12.9** — $PA = LU$. $A = \begin{bmatrix} 0 & 5 & 5 \\ 2 & 9 & 0 \\ 6 & 8 & 8 \end{bmatrix}, P_1 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, P_2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

易知

$$P_1^T = P_1^{-1} = P_1, P_2^T = P_2^{-1} = P_2$$

计算可得

$$P_1 A = \begin{bmatrix} 6 & 8 & 8 \\ 2 & 9 & 0 \\ 0 & 5 & 5 \end{bmatrix}, P_2 A = \begin{bmatrix} 2 & 9 & 0 \\ 0 & 5 & 5 \\ 6 & 8 & 8 \end{bmatrix}$$

LU 分解不唯一:

$$P_1A=\begin{bmatrix}6&8&8\\2&9&0\\0&5&5\end{bmatrix}=\begin{bmatrix}1&0&0\\\frac{1}{3}&1&0\\0&\frac{15}{19}&1\end{bmatrix}\begin{bmatrix}6&8&8\\0&\frac{19}{3}&\frac{-8}{3}\\0&0&\frac{135}{19}\end{bmatrix}=L_1U_1\Rightarrow A=P_1L_1U_1$$

$$P_2A=\begin{bmatrix}2&9&0\\0&5&5\\6&8&8\end{bmatrix}=\begin{bmatrix}1&0&0\\0&1&0\\3&\frac{-19}{5}&1\end{bmatrix}\begin{bmatrix}2&9&2\\0&5&5\\0&0&27\end{bmatrix}=L_2U_2\Rightarrow A=P_2L_2U_2$$

■

Theorem 12.4.2 这个方法等价于对 A 进行行初等变换然后对 PA 进行分解 $PA=LU$

The Complexity of $PA=LU$
复杂度: $\frac{2}{3}n^3$ flops

12.5 舍入误差的影响

■ **Example 12.10**

$$\begin{bmatrix}10^{-5}&1\\1&1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}=\begin{bmatrix}1\\0\end{bmatrix}$$

解得:

$$x_1=-\frac{1}{1-10^{-5}},x_2=\frac{1}{1-10^{-5}}$$

使用 LU 分解求解上述上述方程，并且使用以下两个置换矩阵:

$$P_1=\begin{bmatrix}1&0\\0&1\end{bmatrix}\quad\text{or}\quad P_2=\begin{bmatrix}0&1\\1&0\end{bmatrix}$$

计算过程中，中间结果四舍五入到小数点后四位。
选择 1: $P_1=I$。

$$\begin{bmatrix}10^{-5}&1\\1&1\end{bmatrix}=\begin{bmatrix}1&0\\10^5&1\end{bmatrix}\begin{bmatrix}10^{-5}&1\\0&1-10^5\end{bmatrix}$$

L 和 U 四舍五入到小数点后四位

$$L=\begin{bmatrix}1&0\\10^5&1\end{bmatrix},\quad U=\begin{bmatrix}10^{-5}&1\\0&-10^5\end{bmatrix}$$

向前回代

$$\begin{bmatrix}1&0\\10^5&1\end{bmatrix}\begin{bmatrix}z_1\\z_2\end{bmatrix}=\begin{bmatrix}1\\0\end{bmatrix}\Rightarrow z_1=1,z_2=-10^5$$

向后回代

$$\begin{bmatrix}10^{-5}&1\\0&-10^5\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}=\begin{bmatrix}1\\-10^{-5}\end{bmatrix}\Rightarrow x_1=0,x_2=1$$

x_1 的误差为 100%。

选择 2：行进行交换。

$$\begin{bmatrix} 1 & 1 \\ 10^{-5} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 10^{-5} & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1-10^{-5} \end{bmatrix}$$

L 和 U 四舍五入到小数点后四位

$$L = \begin{bmatrix} 1 & 0 \\ 10^{-5} & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

向前回代

$$\begin{bmatrix} 1 & 0 \\ 10^{-5} & 1 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \Rightarrow z_1 = 0, z_2 = 1$$

向后回代

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \Rightarrow x_1 = -1, x_2 = 1$$

x_1, x_2 的误差约为 10^{-5}。

■

不同置换矩阵 P ，算法可能导致产生不同的误差的结果; 由于数值存储存在误差: 第一种 P_1 行交换，算法不稳定; 第二种 P_2 行交换, 算法是稳定得到“准确”近似解;

在数值分析中，一些比较简单的规则去挑选置换矩阵 P，使得算法结果比较稳定。

$$\begin{bmatrix} 10^{-5} & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 \\ 10^{-5} & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

12.6 稀疏线性方程组

Theorem 12.6.1 如果矩阵 A 是系稀疏矩阵, 则它一般可以被分解为

$$A = P_1 L U P_2$$

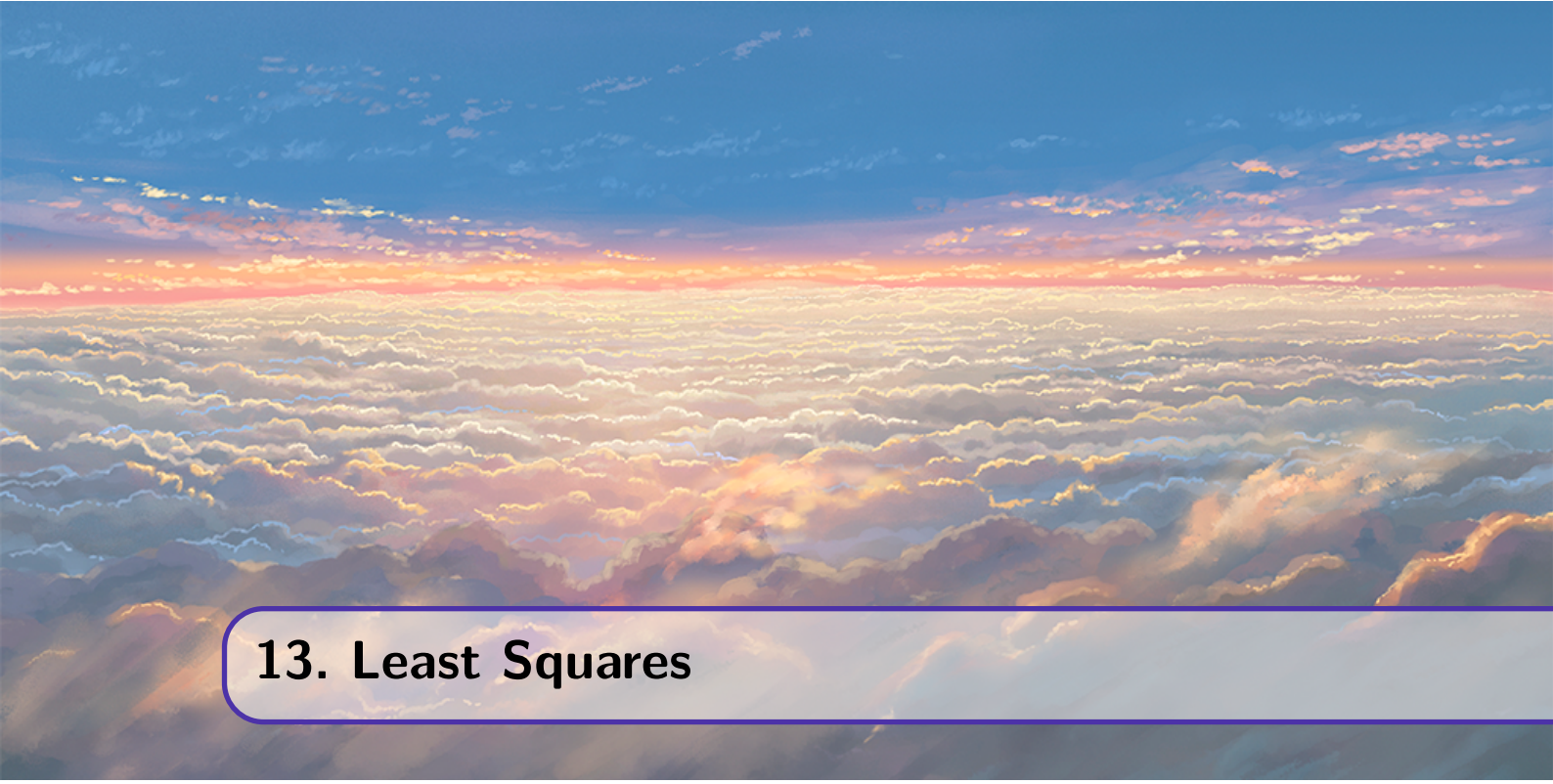
矩阵 P_1, P_2 都为置换矩阵

Corollary 12.6.2 对矩阵 A 进行行变换和列变换得到: $\tilde{A} = P_1^T A P_2^T$

然后进行分解: $\widetilde{\mathrm{A}} = LU$

P_1 和 P_2 的选择会影响 L 和 U 的稀疏度.

II Least Squares

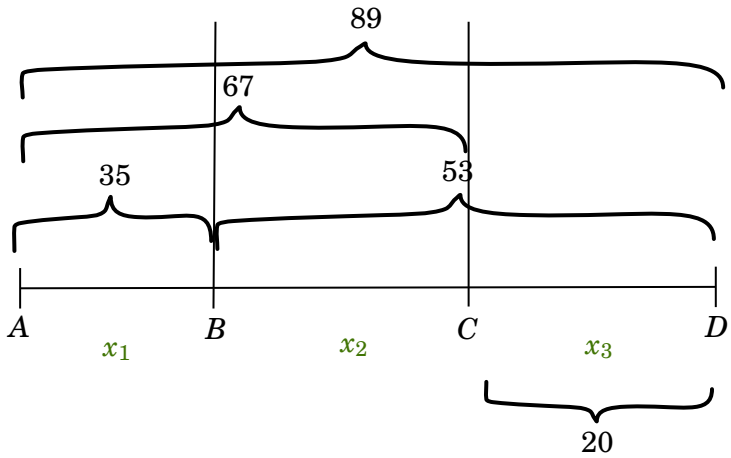


13. Least Squares

13.1 An Example: Measurement Problem

Problem 13.1 已知测量量路段长度：$AD = 89, AC = 67, BD = 53, AB = 35, CD = 20$，$x_1, x_2$ 和 x_3 的长度是多少? ▪

Figure 13.1: Measurement Problem



由 x_1, x_2 和 x_3 的关系可得方程组:

$$\begin{cases} x_1 + x_2 + x_3 = 89 \\ x_1 + x_2 = 67 \\ x_2 + x_3 = 53 \\ x_1 = 35 \\ x_3 = 20 \end{cases} \Leftrightarrow Ax = b, A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, b = \begin{bmatrix} 89 \\ 67 \\ 53 \\ 35 \\ 20 \end{bmatrix}$$

取后三个式子求解方程组，回代前两个式子

$$\begin{cases} x_2 + x_3 = 53 \\ x_1 = 35 \\ x_3 = 20 \end{cases} \Rightarrow x_1 = 35, x_2 = 33, x_3 = 20.$$
$$\begin{cases} x_1 + x_2 + x_3 = 88 \neq 89 \\ x_1 + x_2 = 68 \neq 67 \end{cases}$$

由于测量存在误差，方程组之间相互矛盾，该超定方程组无解。

Problem 13.2 — **最小二乘问题**. 寻找该方程组的近似解，并尽可能逼近方程组的目标 b, 即残差向量 $r = Ax - b$ 某种度量下尽可能小

$$\min_x \|Ax - b\|_2^2 = \|r\|_2^2 \quad (\ell_2\text{范数度量残差})$$

■

使用 ℓ_2、ℓ_∞ 等也可以度量误差，但是函数在零点处不光滑，不能求导。

Problem 13.3 — **求解最小二乘解**. 给定 $\mathrm{A} \in \mathbb{R}^{m\times n}$, $\mathrm{b} \in \mathbb{R}^m$, 求解 $x \in \mathbb{R}^n$ 让目标函数最小

$$\min_x \|Ax - b\|_2^2 = \min_x \sum_{i=1}^{m} \left(\sum_{j=1}^{n} A_{ij}x_j - b_i \right)^2$$

■

Notation 13.1 (最小二乘法的解). 最小二乘法的解为 $\hat{x}$

$$\hat{x} = \arg\min_x \|Ax - b\|_2^2 = \arg\min_x \sum_{i=1}^{m} \left(\sum_{j=1}^{n} A_{ij}x_j - b_i \right)^2$$

■ **Example 13.1** $f(x) = \|Ax - b\|_2^2, \mathrm{A} = \begin{bmatrix} 2 & 0 \\ -1 & 1 \\ 0 & 2 \end{bmatrix}, b = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$

求解

$$\hat{x} = \arg\min_x \|Ax - b\|_2^2$$

解:

$$f(x) = \|Ax - b\|_2^2 = (2x_1 - 1)^2 + (-x_1 + x_2)^2 + (2x_2 + 1)^2$$

$$\frac{\partial f}{\partial x_1} = 10x_1 - 2x_2 - 4, \frac{\partial f}{\partial x_2} = -2x_1 + 10x_2 + 4$$

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = 0 \Rightarrow \hat{x} = \left(\frac{1}{3}, -\frac{1}{3} \right)^T$$

■

Theorem 13.1.1 设最小二乘法的解为 $\hat{x}$ ，满足:

$$\|A\hat{x} - b\|_2^2 \leq \|Ax - b\|_2^2, \forall x \in \mathbf{R}^n$$

当残差 $\hat{r} = A\hat{x} - b = 0$ 时，则 $\hat{x}$ 是线性方程组 $Ax = b$ 的解; 否则其为误差最小平

方和意义下方程组的近似解。

13.2 求解最小二乘法

给定 $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m, x \in \mathbb{R}^n$ 目标函数：

$$f(x) = \|Ax - b\|_2^2 = \sum_{i=1}^{m}\left(\sum_{j=1}^{n} A_{ij}x_j - b_i\right)^2$$

为使目标函数最小，求最优解 $\hat{x}$：$\hat{x} = \arg\min_x f(x)$

Theorem 13.2.1 可微函数 $f(x)$ 的最优解 $\hat{x}$ 满足条件：梯度 $\nabla f(\hat{x}) = \mathbf{0}$，即：

$$\nabla f(\hat{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\hat{x}) \\ \vdots \\ \frac{\partial f}{\partial x_n}(\hat{x}) \end{bmatrix} = 2A^T(A\hat{x} - b) = 0$$

Theorem 13.2.2 — 正规方程与最小二乘解.

$$A^TAx = A^Tb$$

A 的列向量**线性无关**时，则 $\hat{x} = \left(A^TA\right)^{-1}A^Tb$。

Proof. 设函数 $g_i(x) = \sum_{j=1}^{n} A_{ij}x_j - b_i$，则有

$$g_i(x) = \sum_{j=1}^{n} A_{ij}x_j - b_i \Rightarrow \begin{pmatrix} A_{1,1} & \cdots & A_{1,k} & \cdots & A_{1,n} \\ \vdots & & \vdots & & \vdots \\ \boldsymbol{A_{j,1}} & \boldsymbol{\cdots} & \boldsymbol{A_{j,k}} & \boldsymbol{\cdots} & \boldsymbol{A_{j,n}} \\ \vdots & & \vdots & & \vdots \\ A_{m,1} & \cdots & A_{m,k} & \cdots & A_{m,n} \end{pmatrix}\begin{pmatrix} \boldsymbol{x_1} \\ \vdots \\ \boldsymbol{x_j} \\ \vdots \\ \boldsymbol{x_n} \end{pmatrix} - \begin{pmatrix} b_1 \\ \vdots \\ \boldsymbol{b_j} \\ \vdots \\ b_n \end{pmatrix}$$

$$f(x) = \|Ax - b\|_2^2 = \sum_{i=1}^{m}\left(\sum_{j=1}^{n} A_{ij}x_j - b_i\right)^2 = \sum_{i=1}^{m}\left(g_i(x)\right)^2$$

函数 $f(x)$ 对变量 x_k 偏导为

$$\frac{\partial f(x)}{\partial x_k} = \sum_{i=1}^{m}\left((2g_i(x))\left(\frac{\partial g_i(x)}{\partial x_k}\right)\right)$$

又因为

$$\frac{\partial g_i(x)}{\partial x_k} = A_{ik}$$

所以

$$\begin{aligned}\frac{\partial f}{\partial x_k}(x) &= \sum_{i=1}^{m} 2\left(g_i(x)\right)\left(A_{ik}\right) \\ &= 2\sum_{i=1}^{m}\left(\left(\sum_{j=1}^{n} A_{ij}x_j - b_i\right)\left(A_{ik}\right)\right) \\ &= 2\sum_{i=1}^{m}\left(\left(\sum_{j=1}^{n} A_{ij}x_j\right)\left(A_{ik}\right)\right) - 2\sum_{i=1}^{m}\left(\left(b_i\right)\left(A_{ik}\right)\right)\end{aligned}$$

注意有

$$\sum_{j=1}^{n} A_{ij}x_j = \begin{pmatrix} A_{1,1} & \cdots & A_{1,k} & \cdots & A_{1,n} \\ \vdots & & \vdots & & \vdots \\ A_{j,1} & \cdots & A_{j,k} & \cdots & A_{j,n} \\ \vdots & & \vdots & & \vdots \\ A_{m,1} & \cdots & A_{m,k} & \cdots & A_{m,n} \end{pmatrix}\begin{pmatrix} x_1 \\ \vdots \\ x_j \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} Result_1 \\ \vdots \\ Result_k \\ \vdots \\ Result_m \end{pmatrix} = Ax$$

$$\sum_{i=1}^{m}\left(\underbrace{\left(\sum_{j=1}^{n} A_{ij}x_j\right)}_{Result}\left(A_{ik}\right)\right) = \begin{array}{c} A_{1,k} \times Result_1 \\ + \\ \vdots \\ + \\ A_{i,k} \times Result_i \\ + \\ \vdots \\ + \\ A_{m,k} \times Result_m \end{array} = \begin{pmatrix} A_{1,k} \\ \vdots \\ A_{i,k} \\ \vdots \\ A_{m,k} \end{pmatrix}^T Ax = a_k^T Ax$$

类似地，有

$$\sum_{i=1}^{m}\left(\left(b_i\right)\left(A_{ik}\right)\right) = \begin{array}{c} A_{1,k} \times b_1 \\ + \\ \vdots \\ + \\ A_{i,k} \times b_i \\ + \\ \vdots \\ + \\ A_{m,k} \times b_m \end{array} = a_k^T b$$

所以

$$\begin{aligned}\frac{\partial f}{\partial x_k}(x) &= 2a_k^T Ax - 2a_k^T b \\ &= 2a_k^T(Ax - b)\end{aligned}$$

所以函数 $f(x)$ 的梯度

$$\begin{aligned}\nabla f(x) &= \begin{bmatrix} \frac{\partial f}{\partial x_1}(x) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x) \end{bmatrix} \\ &= 2\begin{bmatrix} a_1^T(Ax-b) \\ a_2^T(Ax-b) \\ \vdots \\ a_n^T(Ax-b) \end{bmatrix} \\ &= 2\left[a_1, a_2, \cdots, a_n\right]^T(Ax-b) \\ &= 2A^T(Ax-b)\end{aligned}$$

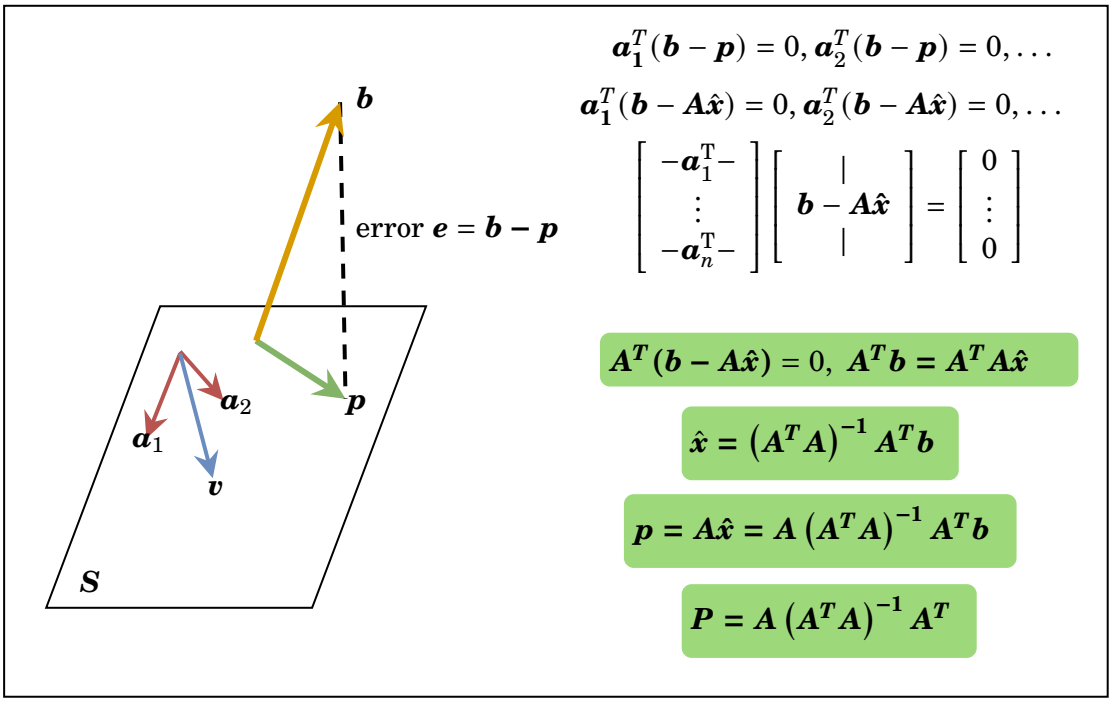
$$\nabla f(x) = 2\left(A^TAx - A^Tb\right) = 0 \Rightarrow A^TAx = A^Tb$$

A 的列向量无关时，则 $\hat{x} = \left(A^TA\right)^{-1}A^Tb$。

■

13.3 The Geometry of Least Squares: 投影与 A 列空间的关系

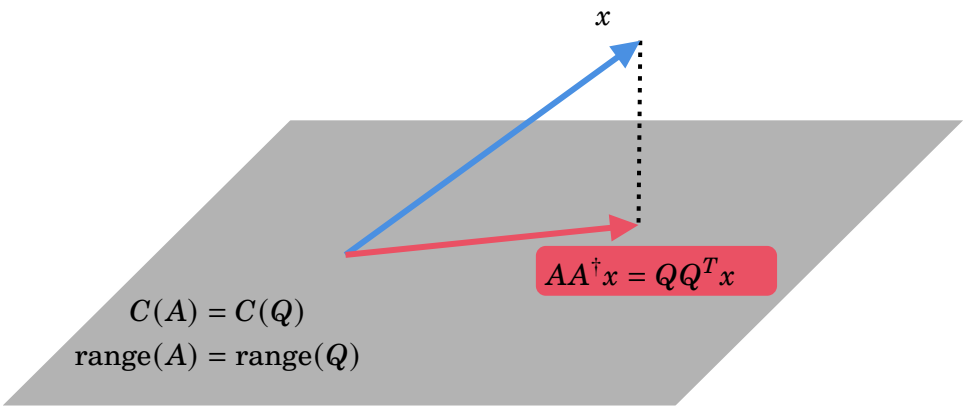
Figure 13.2: Projection of $\boldsymbol{b}$ into the column space of A, A is any matrice. Sourced from [**Strang1993IntroductionTL**]



矩阵 $\mathrm{A} \in \mathbb{R}^{m\times n}$ 的列 $a_1, a_2, \ldots, a_n \in \mathbb{R}^m$ 的最小二乘法问题

$$\hat{x} = \arg\min_x \|Ax-b\|_2^2, \|Ax-b\|_2^2 = \left\|\sum_{j=1}^n a_j x_j - b\right\|_2^2$$

Figure 13.3: Projecting onto the column space of A is also projecting onto the column space of Q



向量 b 在 $\text{range}(A)$ 上的投影是 $A\left(A^TA\right)^{-1}A^Tb$。

残余向量 $\hat{r} = A\hat{x} - b$ 满足 $A^T\hat{r} = A^T(A\hat{x} - b) = 0$。残余向量 $\hat{r}$ 正交于 A 的每一列, 因此正交于 $\text{range}(A)$。

Theorem 13.3.1 — 投影与 A 列空间的关系. $A\hat{x} \in \text{range}(A)$ 是 A 的列空间中最接近 b 的向量。

$\hat{r} = A\hat{x} - b$ 正交于 A 的列空间（值域空间）$\text{range}(A)$。

13.4 正规方程

Theorem 13.4.1 — 最小二乘法问题的正规方程.

$$A^TAx = A^Tb$$

等价于

$$\nabla f(x) = 0, f(x) = \|Ax - b\|_2^{2\prime\prime}$$

系数矩阵 A^TA 是 A 的 Gram 矩阵，最小二乘法问题所有的解都满足正规方程。

Theorem 13.4.2 如果 A 的列线性无关，则

A^TA 为非奇异矩阵，正规方程此时有唯一解。

13.5 QR 分解求解最小二乘法

Theorem 13.5.1 — QR 分解求解最小二乘法. 若 $A \in \mathbb{R}^{m\times n}$ 的列向量线性无关，则存在 $A = QR$ 分解，$Q \in \mathbb{R}^{m\times n}$ ，$R \in \mathbb{R}^{n\times n}$

最小二乘法问题的解

$$\begin{aligned}\hat{x} &= \left(A^T A\right)^{-1} A^T b \\ &= \left((QR)^T (QR)\right)^{-1} (QR)^T b \\ &= \left(R^T Q^T QR\right)^{-1} R^T Q^T b \\ &= \left(R^T R\right)^{-1} R^T Q^T b \\ &= R^{-1} Q^T b\end{aligned}$$

■ **Example 13.2**

$$A = \begin{bmatrix} 3 & -6 \\ 4 & -8 \\ 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} -1 \\ 7 \\ 0 \end{bmatrix}$$

首先对 A 进行 QR 分解

$$Q = \begin{bmatrix} 3/5 & 0 \\ 4/5 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = \begin{bmatrix} 5 & -10 \\ 0 & 1 \end{bmatrix}$$

计算 $d = Q^T b = (5, 2)$
求解 $Rx = d$

$$\begin{bmatrix} 5 & -10 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$$

解得 $x_1 = 5, x_2 = 2$

■

13.5.1 The Complexity of Solving Least Square Problem via QR Decomposition

算法复杂度:

- 首先对 A 进行 QR 分解 $A = QR$ $\left(2mn^2 \text{ flops }\right)$
- 计算矩阵向量乘积 $d = Q^T b$(2mn flops)
- 通过回代求解 $Rx = d$ $\left(n^2 \text{ flops }\right)$
- 复杂度: $2mn^2$ flops

13.6 求解正规方程可能带来的严重误差

直接求解正规方程组求解:

$$A^T Ax = A^T b$$

可能会造成严重的舍入误差。

■ **Example 13.3** 一个列向量“几乎”线性相关的矩阵

$$A = \begin{bmatrix} 1 & -1 \\ 0 & 10^{-5} \\ 0 & 0 \end{bmatrix}, b = \begin{bmatrix} 0 \\ 10^{-5} \\ 1 \end{bmatrix}$$

将中间结果四舍五入到小数点后 8 位.

方法 1： 通过 Gram 矩阵求解

$$A^TA = \begin{bmatrix} 1 & -1 \\ -1 & 1+10^{-10} \end{bmatrix} \approx \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, A^Tb = \begin{bmatrix} 0 \\ 10^{-10} \end{bmatrix} \Rightarrow x = \begin{bmatrix} 10^{-10} \\ 10^{-10} \end{bmatrix}$$

经过四舍五入之后，Gram 矩阵为奇异矩阵。

方法 2 : 通过对 A 进行 QR 分解

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, R = \begin{bmatrix} 1 & -1 \\ 0 & 10^{-5} \end{bmatrix}$$

$$\begin{aligned} \hat{x} &= \left(A^TA\right)^{-1}A^Tb = R^{-1}Q^Tb \\ \Rightarrow & Rx = Q^Tb \\ \Rightarrow & \begin{bmatrix} 1 & -1 \\ 0 & 10^{-5} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 10^{-5} \end{bmatrix} \\ \Rightarrow & x = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \end{aligned}$$

■

方法 2 比方法 1 更稳定，因为它避免构造 Gram 矩阵。

13.7 梯度下降法

给定 $A \in \mathbb{R}^{m\times n}$, b $\in \mathbb{R}^m, x \in \mathbb{R}^n$ 目标函数:

$$f(x) = \|Ax-b\|_2^2 = \sum_{i=1}^{m}\left(\sum_{j=1}^{n}A_{ij}x_j - b_i\right)^2$$

为使目标函数最小, 可求最优解 $\hat{x}$: $\hat{x} = \arg\min_x f(x)$。

Problem 13.4 $A \in \mathbb{R}^{\mathrm{m}\times n}$ 列向量线性相关或 n 非常大 $A^TA \in \mathbb{R}^{n\times n}$ 不可逆，无法直接代入求得最小二乘解。 ■

通过迭代求解目标的最优解过程:

$$x^{(1)}, x^{(2)}, \cdots, x^{(k)} \rightarrow \hat{x}$$

设 $x^{(k)}$ 是第 k 步迭代，期望更新 $x^{(k+1)}$ ，满足 $f\left(x^{(k+1)}\right) < f\left(x^{(k)}\right)$.

设函数 $f(x)$ 可微，根据泰勒公式，在 $x^{(k)}$ 的一阶公式为

$$f\left(x^{(k+1)}\right) = f\left(x^{(k)}\right) + \left\langle \nabla f\left(x^{(k)}\right), x^{(k+1)} - x^{(k)} \right\rangle + o\left(\left\|x^{(k+1)} - x^{(k)}\right\|\right)$$

如果 $\left\|x^{(k+1)} - x^{(k)}\right\|_2$ 足够小, 则有

$$f\left(x^{(k+1)}\right) - f\left(x^{(k)}\right) \approx \left\langle \nabla f\left(x^{(k)}\right), x^{(k+1)} - x^{(k)} \right\rangle$$

Corollary 13.7.1 根据柯西不等式 1.6.1

$\left|\left\langle \nabla f\left(x^{(k)}\right), x^{(k+1)}-x^{(k)}\right\rangle\right| \leq \left\|\nabla f\left(x^{(k)}\right)\right\|_2 \left\|x^{(k+1)}-x^{(k)}\right\|_2$

所以有

$$\left\langle \nabla f\left(x^{(k)}\right), x^{(k+1)}-x^{(k)}\right\rangle \geq -\left\|\nabla f\left(x^{(k)}\right)\right\|_2 \left\|x^{(k+1)}-x^{(k)}\right\|_2$$

当 $x^{(k+1)}-x^{(k)}=-\alpha_k \nabla f\left(x^{(k)}\right), \alpha_k>0$ 时，等式成立。

由于 $-\left\|\nabla f\left(x^{(k)}\right)\right\|_2 \left\|x^{(k+1)}-x^{(k)}\right\|_2$ 是非负的，此时 $f\left(x^{(k+1)}\right)-f\left(x^{(k)}\right) \leq 0$。

迭代公式为

$$x^{(k+1)}=x^{(k)}-\alpha_k \nabla f\left(x^{(k)}\right), f\left(x^{(k+1)}\right)<f\left(x^{(k)}\right)$$

Definition 13.7.1 — **梯度下降法求解最小二乘法.**

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}\|Ax-b\|_2^2, \quad A\in\mathbb{R}^{m\times n}, b\in\mathbb{R}^m$$

令

$$f(x)=\frac{1}{2}\|Ax-b\|_2^2$$

则 f 为凸函数, 并有 $\nabla f(x)=A^T(Ax-b)$。

则 $A^TA\in\mathbb{R}^{n\times n}$ 。如果列向量**线性相关**会导致其**不可逆**或 n 非常大。可以通过梯度下降法迭代

$$x^{(k+1)}=x^{(k)}-\alpha_k \nabla f\left(x^{(k)}\right), f\left(x^{(k+1)}\right)<f\left(x^{(k)}\right)$$

求解

$$x^{(k+1)}=x^{(k)}-\alpha^{(k)}A^T\left(Ax^{(k)}-b\right)$$

Algorithm 16: 梯度下降法

```
初始化 x^(0), k = 0
while Not Convergent do
    p^(k) = A^T (A x^(k) − b)
    x^(k+1) = x^(k) − α^(k) p^(k)
    k ← k + 1
end
```

13.8 估计学习率（步长）α

Problem 13.5

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}\|Ax-b\|_2^2, \quad A\in\mathbb{R}^{m\times n}, b\in\mathbb{R}^m$$

令 $f(x)=\frac{1}{2}\|Ax-b\|_2^2$

$$x^{(k+1)} = x^{(k)} - \alpha^{(k)} A^T \left(Ax^{(k)} - b\right)$$

需要估计 $\alpha^{(k)}$。
为了估计 $\alpha^{(k)}$, 通过线性搜索估计:

$$\alpha^{(k)} = \arg\min_{\alpha \in \Re} f\left(x^{(k)} - \alpha A^T \left(Ax^{(k)} - b\right)\right)$$

即 $\alpha^{(k)}$ 是最优步长。在上面的优化式中 $x^{(k)}$、A、b 均视为定值。

Theorem 13.8.1 — 线性搜索估计的最优步长.

$$\alpha^{(k)} = \frac{\left\|A^T \left(Ax^{(k)} - b\right)\right\|_2^2}{\left\|AA^T \left(Ax^{(k)} - b\right)\right\|_2^2}$$

Proof. 令 $g(\alpha) = f\left(x^{(k)} - \alpha A^T \left(Ax^{(k)} - b\right)\right)$ 是关于 α 的凸函数, 则有

$$\min_{\alpha} g(\alpha) \Rightarrow g'(\alpha) = 0 \Rightarrow \alpha^{(k)} = \frac{\left\|A^T \left(Ax^{(k)} - b\right)\right\|_2^2}{\left\|AA^T \left(Ax^{(k)} - b\right)\right\|_2^2}$$

$$\begin{aligned}
&f(x) = \frac{1}{2}\|Ax - b\|_2^2, g\left(\alpha^{(k)}\right) = f\left(x^{(k)} - \alpha^{(k)} A^T \left(Ax^{(k)} - b\right)\right) \\
\Rightarrow &g\left(\alpha^{(k)}\right) = \frac{1}{2}\left\|A\left(x^{(k)} - \alpha^{(k)} A^T \left(Ax^{(k)} - b\right)\right) - b\right\|_2^2 \\
&= \frac{1}{2}\left\|\left(Ax^{(k)} - b\right) - \left(\alpha^{(k)} A^T \left(Ax^{(k)} - b\right)\right)\right\|_2^2 \\
&= \frac{1}{2}\left(\left(Ax^{(k)} - b\right)^T \left(Ax^{(k)} - b\right) + \left(\alpha^{(k)} A^T \left(Ax^{(k)} - b\right)\right)^T \left(\alpha^{(k)} A^T \left(Ax^{(k)} - b\right)\right)\right) \\
&- \left(Ax^{(k)} - b\right)^T \left(\alpha^{(k)} A^T \left(Ax^{(k)} - b\right)\right) \\
\Rightarrow &g'\left(\alpha^{(k)}\right) = \alpha^{(k)} \left(A^T \left(Ax^{(k)} - b\right)\right)^T \left(A^T \left(Ax^{(k)} - b\right)\right) - \left(Ax^{(k)} - b\right)^T \left(A^T \left(Ax^{(k)} - b\right)\right) = 0 \\
\Rightarrow &\alpha^{(k)} = \frac{\left\|A^T \left(Ax^{(k)} - b\right)\right\|_2^2}{\left\|AA^T \left(Ax^{(k)} - b\right)\right\|_2^2}
\end{aligned}$$

■

Algorithm 17: 使用线性搜索估计步长的梯度下降法

初始化 $x^{(0)}$, $k = 0$
while *Not Convergent* **do**
  $p^{(k)} = A^T \left(Ax^{(k)} - b\right)$
  $\alpha^{(k)} = \dfrac{\left\|p^{(k)}\right\|_2^2}{\left\|Ap^{(k)}\right\|_2^2}$
  $x^{(k+1)} = x^{(k)} - \alpha^{(k)} p^{(k)}$
end

14. Least squares data fitting

14.1 Model fitting

Problem 14.1 suppose x and a scalar quantity y are related as

$$y \approx f(x)$$

x is the *explanatory variable* or *independent variable*, y is the *outcome*, or *response variable*, or *dependent variable*.

We don't know f, but have some idea about its general form

▪

Definition 14.1.1 — Model fitting. find an approximate model $\hat{f}$ for f, based on observations

we use the notation $\hat{y}$ for the model prediction of the outcome y

$$\hat{y} = \hat{f}(x)$$

14.1.1 Prediction error

we have data consisting of N examples (samples, measurements, observations)

$$x^{(1)}, \ldots, x^{(N)}, \quad y^{(1)}, \ldots, y^{(N)}$$

model prediction for example i is $\hat{y}^{(i)} = \hat{f}\left(x^{(i)}\right)$

Definition 14.1.2 — prediction error, residual. the *prediction error* or *residual* for example i is

$$r^{(i)} = y^{(i)} - \hat{y}^{(i)} = y^{(i)} - \hat{f}\left(x^{(i)}\right)$$

the model $\hat{f}$ fits the data well if the N residuals $r^{(i)}$ are small
prediction error can be quantified using the mean square error (MSE)

Definition 14.1.3 — mean square error (MSE).

$$\frac{1}{N}\sum_{i=1}^{N}\left(r^{(i)}\right)^2$$

the square root of the MSE is the *RMS error*.

14.2 Regression

we first consider the regression model

Problem 14.2

$$\hat{f}(x) = x^T\beta + v$$

Here, the independent variable x is an n-vector, the elements of x are the *regressors*.

■

The model is parameterized by the weight vector β and the offset (intercept) v.

Theorem 14.2.1 The prediction error for example i is

$$\begin{aligned} r^{(i)} &= y^{(i)} - \hat{f}\left(x^{(i)}\right) \\ &= y^{(i)} - \left(x^{(i)}\right)^T\beta - v \end{aligned}$$

Theorem 14.2.2 The MSE is

$$\frac{1}{N}\sum_{i=1}^{N}\left(r^{(i)}\right)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \left(x^{(i)}\right)^T\beta - v\right)^2$$

14.3 Least squares regression

Problem 14.3 choose the model parameters v, β that minimize the MSE

$$\frac{1}{N}\sum_{i=1}^{N}\left(v + \left(x^{(i)}\right)^T\beta - y^{(i)}\right)^2$$

■

this is a least squares problem: minimize $\left\|A\theta - y^{\mathrm{d}}\right\|^2$ with

$$A = \begin{bmatrix} 1 & \left(x^{(1)}\right)^T \\ 1 & \left(x^{(2)}\right)^T \\ \vdots & \vdots \\ 1 & \left(x^{(N)}\right)^T \end{bmatrix}, \quad \theta = \begin{bmatrix} v \\ \beta \end{bmatrix}, \quad y^{\mathrm{d}} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix}$$

Notation 14.1. *We write the solution as* $\hat{\theta} = (\hat{v}, \hat{\beta})$.

14.4 Linear-in-parameters model

Problem 14.4 we choose the model $\hat{f}(x)$ from a family of models

$$\hat{f}(x) = \theta_1 f_1(x) + \theta_2 f_2(x) + \cdots + \theta_p f_p(x)$$

the functions f_i are scalar valued basis functions (chosen by us), the basis functions often include a constant function (typically, $f_1(x) = 1$).

The coefficients $\theta_1, \ldots, \theta_p$ are the model parameters. ∎

the model $\hat{f}(x)$ is linear in the parameters θ_i

Corollary 14.4.1 if $f_1(x) = 1$, this can be interpreted as a regression model

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

with parameters $v = \theta_1, \beta = \theta_{2:p}$ and new features $\tilde{x}$ generated from x :

$$\tilde{x}_1 = f_2(x), \quad \ldots, \quad \tilde{x}_p = f_p(x)$$

14.5 Least squares model fitting

Problem 14.5 fit linear-in-parameters model to data set $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(N)}, y^{(N)}\right)$ ∎

Theorem 14.5.1 residual for data sample i is

$$r^{(i)} = y^{(i)} - \hat{f}\left(x^{(i)}\right) = y^{(i)} - \theta_1 f_1\left(x^{(i)}\right) - \cdots - \theta_p f_p\left(x^{(i)}\right)$$

Problem 14.6 — least squares model fitting. least squares model fitting: choose parameters θ by minimizing MSE

$$\frac{1}{N}\left(\left(r^{(1)}\right)^2 + \left(r^{(2)}\right)^2 + \cdots + \left(r^{(N)}\right)^2\right)$$

∎

this is a least squares problem:

Problem 14.7 minimize $\left\|A\theta - y^{\mathrm{d}}\right\|^2$ with

$$A = \begin{bmatrix} f_1\left(x^{(1)}\right) & \cdots & f_p\left(x^{(1)}\right) \\ f_1\left(x^{(2)}\right) & \cdots & f_p\left(x^{(2)}\right) \\ \vdots & & \vdots \\ f_1\left(x^{(N)}\right) & \cdots & f_p\left(x^{(N)}\right) \end{bmatrix}, \quad \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_p \end{bmatrix}, \quad y^{\mathrm{d}} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix}$$

∎

14.5.1 Example: polynomial approximation

Problem 14.8

$$\hat{f}(x) = \theta_1 + \theta_2 x + \theta_3 x^2 + \cdots + \theta_p x^{p-1}$$

∎

This is a linear-in-parameters model with basis functions $1, x, \ldots, x^{p-1}$

Problem 14.9 — least squares model fitting. choose parameters θ by minimizing MSE

$$\frac{1}{N}\left(\left(y^{(1)} - \hat{f}\left(x^{(1)}\right)\right)^2 + \left(y^{(2)} - \hat{f}\left(x^{(2)}\right)\right)^2 + \cdots + \left(y^{(N)} - \hat{f}\left(x^{(N)}\right)\right)^2\right)$$

∎

Problem 14.10 — least squares model fitting in matrix notation. minimize $\left\|A\theta - y^{\mathrm{d}}\right\|^2$ with

$$A = \begin{bmatrix} 1 & x^{(1)} & \left(x^{(1)}\right)^2 & \cdots & \left(x^{(1)}\right)^{p-1} \\ 1 & x^{(2)} & \left(x^{(2)}\right)^2 & \cdots & \left(x^{(2)}\right)^{p-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x^{(N)} & \left(x^{(N)}\right)^2 & \cdots & \left(x^{(N)}\right)^{p-1} \end{bmatrix}, \quad y^{\mathrm{d}} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix}$$

■

14.6 Piecewise-affine function

Definition 14.6.1 — knot points. $a_1 < a_2 < \cdots < a_k$ on the real axis

Proposition 14.6.1 piecewise-affine function is continuous, and affine on each interval $[a_k, a_{k+1}]$

Theorem 14.6.2 piecewise-affine function with knot points $a_1, \ldots, a_k$ can be written as

$$\hat{f}(x) = \theta_1 + \theta_2 x + \theta_3 (x - a_1)_+ + \cdots + \theta_{2+k} (x - a_k)_+$$

where $u_+ = \max\{u, 0\}$

14.6.1 Piecewise-affine function fitting

piecewise-affine model is in linear in the parameters θ, with basis functions

$$f_1(x) = 1, \quad f_2(x) = x, \quad f_3(x) = (x - a_1)_+, \quad \ldots, \quad f_{k+2}(x) = (x - a_k)_+$$

■ **Example 14.1** fit piecewise-affine function with knots $a_1 = -1, a_2 = 1$ to 100 points

■

14.7 Time series trend

Definition 14.7.1 — trend line. Suppose N data samples from time series: $y^{(i)}$ is value at time i, for $i = 1, \ldots, N$
straight-line fit $\hat{y}^{(i)} = \theta_1 + \theta_2 i$ is the *trend line*

Definition 14.7.2 — de-trended time series. $y^{\mathrm{d}} - \hat{y}^{\mathrm{d}} = \left(y^{(1)} - \hat{y}^{(1)}, \ldots, y^{(N)} - \hat{y}^{(N)}\right)$ is the de-trended time series

Problem 14.11 least squares fitting of trend line: minimize $\left\|A\theta - y^{\mathrm{d}}\right\|^2$ with

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ \vdots & \vdots \\ 1 & N \end{bmatrix}, \quad y^{\mathrm{d}} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \vdots \\ y^{(N)} \end{bmatrix}$$

■

14.7.1 Example: world petroleum consumption

time series of world petroleum consumption (million barrels/day) versus year
left figure shows data samples and trend line
right figure shows de-trended time series

14.7.2 Trend plus seasonal component

Proposition 14.7.1 model time series can be decomposed as a linear trend plus a periodic component with period P

$$\hat{y}^{\mathrm{d}} = \hat{y}^{\mathrm{lin}} + \hat{y}^{\mathrm{seas}}$$

with $\hat{y}^{\mathrm{lin}} = \theta_1(1, 2, \ldots, N)$ and

$$\hat{y}^{\mathrm{seas}} = (\theta_2, \theta_3, \ldots, \theta_{P+1}, \theta_2, \theta_3, \ldots, \theta_{P+1}, \ldots, \theta_2, \theta_3, \ldots, \theta_{P+1})$$

Definition 14.7.3 — offset. the mean of $\hat{y}^{\mathrm{seas}}$ serves as a *constant offset*.

Definition 14.7.4 — de-trended, seasonally adjusted time series. Residual $y^{\mathrm{d}} - \hat{y}^{\mathrm{d}}$ is the de-trended, seasonally adjusted time series

Problem 14.12 least squares formulation: minimize $\left\|A\theta - y^{\mathrm{d}}\right\|^2$ with

$$A_{1:N,1} = \begin{bmatrix} 1 \\ 2 \\ \vdots \\ N \end{bmatrix}, \quad A_{1:N,2:P+1} = \begin{bmatrix} I_P \\ I_P \\ \vdots \\ I_P \end{bmatrix}, \quad y^{\mathrm{d}} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix}$$

■

14.8 Auto-regressive (AR) time series model

Problem 14.13

$$\hat{z}_{t+1} = \beta_1 z_t + \cdots + \beta_M z_{t-M+1}, \quad t = M, M+1, \ldots$$

$z_1, z_2, \ldots$ is a time series, $\hat{z}_{t+1}$ is a prediction of z_{t+1}, made at time t. Prediction $\hat{z}_{t+1}$ is a linear function of previous M values $z_t, \ldots, z_{t-M+1}$. M is the memory of the model. ■

Problem 14.14 Least squares fitting of AR model:
given oberved data $z_1, \ldots, z_T$, minimize

$$(z_{M+1} - \hat{z}_{M+1})^2 + (z_{M+2} - \hat{z}_{M+2})^2 + \cdots + (z_T - \hat{z}_T)^2$$

■

Problem 14.15 — Least square formulation for AR model. this is a least squares problem:
minimize $\left\|A\beta - y^{\mathrm{d}}\right\|^2$ with

$$A = \begin{bmatrix} z_M & z_{M-1} & \cdots & z_1 \\ z_{M+1} & z_M & \cdots & z_2 \\ \vdots & \vdots & & \vdots \\ z_{T-1} & z_{T-2} & \cdots & z_{T-M} \end{bmatrix}, \quad \beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_M \end{bmatrix}, \quad y^{\mathrm{d}} = \begin{bmatrix} z_{M+1} \\ z_{M+2} \\ \vdots \\ z_T \end{bmatrix}$$

■

14.9 Generalization and validation

Definition 14.9.1 — Generalization ability. ability of model to predict outcomes for new, unseen data

Definition 14.9.2 — Model Validation (Out-of-sample Validation). To assess generalization ability, divide data in two sets: training set and test (or validation) set. Use training set to fit model and use test set to get an idea of generalization ability

Definition 14.9.3 — Over-fit model. model with low prediction error on training set, bad generalization ability. Prediction error on training set is much smaller than on test set.

14.9.1 Cross-validation

It is an extension of out-of-sample validation.

Algorithm 18: Cross-validation

```
1 divide data in K sets (folds); typical values are K = 5, K = 10
2 for i = 1 to K do
3 |   fit model i using fold i as test set and other data as training set
4 |   compare parameters and train/test RMS errors for the K models
5 end
```

14.10 Boolean (two-way) classification

Problem 14.16 A data fitting problem where the outcome y can take two values $+1, -1$.

Values of y represent two categories (true/false, spam/not spam, ...). Model $\hat{y} = \hat{f}(x)$ is called a *Boolean classifier*.

■

Use least squares to fit model $\tilde{f}(x)$ to training set $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(N)}, y^{(N)}\right)$
$\tilde{f}(x)$ can be a regression model $\tilde{f}(x) = x^T\beta + v$ or linear in parameters

$$\tilde{f}(x) = \theta_1 f_1(x) + \cdots + \theta_p f_p(x)$$

Take sign of $\tilde{f}(x)$ to get a Boolean classifier

$$\hat{f}(x) = \text{sign}(\tilde{f}(x)) = \begin{cases} +1 & \text{if } \tilde{f}(x) \geq 0 \\ -1 & \text{if } \tilde{f}(x) < 0 \end{cases}$$

14.10.1 Example: handwritten digit classification

Illustrations of this example are needed to better understand this example.

- MNIST data set used in homework - 28×28 images of handwritten digits ($n = 28^2 = 784$ pixels) - data set contains 60000 training examples; 10000 test examples - we only use the 493 pixels that are nonzero in at least 600 training examples - Boolean classifier distinguishes digit zero ($y = 1$) from other digits ($y = -1$)

Classifier with basic regression model

$$\hat{f}(x) = \text{sign}(\tilde{f}(x)) = \text{sign}\left(x^T\beta + v\right)$$

- x is vector of 493 pixel intensities - figure shows distribution of $\tilde{f}\left(x^{(i)}\right) = \left(x^{(i)}\right)^T\hat{\beta} + \hat{v}$ on training set

blue bars to the left of dashed line are false negatives (misclassified digits zero)
red bars to the right of dashed line are false positives (misclassified non-zeros)

Classifier with additional nonlinear features

$$\hat{f}(x) = \text{sign}(\tilde{f}(x)) = \text{sign}\left(\sum_{i=1}^{p}\theta_i f_i(x)\right)$$

- basis functions include constant, 493 elements of x, plus 5000 functions

$$f_i(x) = \max\left\{0, r_i^T x + s_i\right\} \qquad \text{with randomly generated } r_i, s_i$$

- figure shows distribution of $\tilde{f}\left(x^{(i)}\right)$ on training set

14.11 Multi-class classification

Problem 14.17 a data fitting problem where the outcome y can takes values $1, \ldots, K$
values of y represent K labels or categories.
multi-class classifier $\hat{y} = \hat{f}(x)$ maps x to an element of $\{1, 2, \ldots, K\}$ ∎

14.11.1 Least squares multi-class classifier

Algorithm 19: Least squares multi-class classifier

```
for k = 1, ..., K do
    compute Boolean classifier to distinguish class k from not k
        f̂_k(x) = sign(f̃_k(x))
end
```

Define multi-class classifier as

Definition 14.11.1 — multi-class classifier.

$$\hat{f}(x) = \underset{k=1,\ldots,K}{\text{argmax}}\tilde{f}_k(x)$$

14.12 statistics interpretation for least squares

Problem 14.18

$$y = X\beta + \epsilon$$

- β is (non-random) p-vector of unknown parameters
- X is $n \times p$ (data matrix or design matrix, i.e., result of experiment design)
- If there is an offset v, we include it in β and add a column of ones in X
- ϵ is a random n-vector (random error or disturbance)
- y is an observable random n-vector

■

This notation differs from previous sections but is common in statistics.

We discuss methods for estimating parameters β from observations of y.

14.12.1 Assumptions

Proposition 14.12.1 X is tall $(n > p)$ with linearly independent columns

Proposition 14.12.2 random disturbances ϵ_i have zero mean

$$\mathbf{E}\epsilon_i = 0 \quad \text{for } i = 1, \ldots, n$$

Proposition 14.12.3 random disturbances have equal variances σ^2

$$\mathbf{E}\epsilon_i^2 = \sigma^2 \quad \text{for } i = 1, \ldots, n$$

Proposition 14.12.4 random disturbances are uncorrelated (have zero covariances)

$$\mathbf{E}\left(\epsilon_i \epsilon_j\right) = 0 \quad \text{for } i, j = 1, \ldots, n \text{ and } i \neq j$$

Theorem 14.12.5 last three assumptions can be combined using matrix and vector notation:

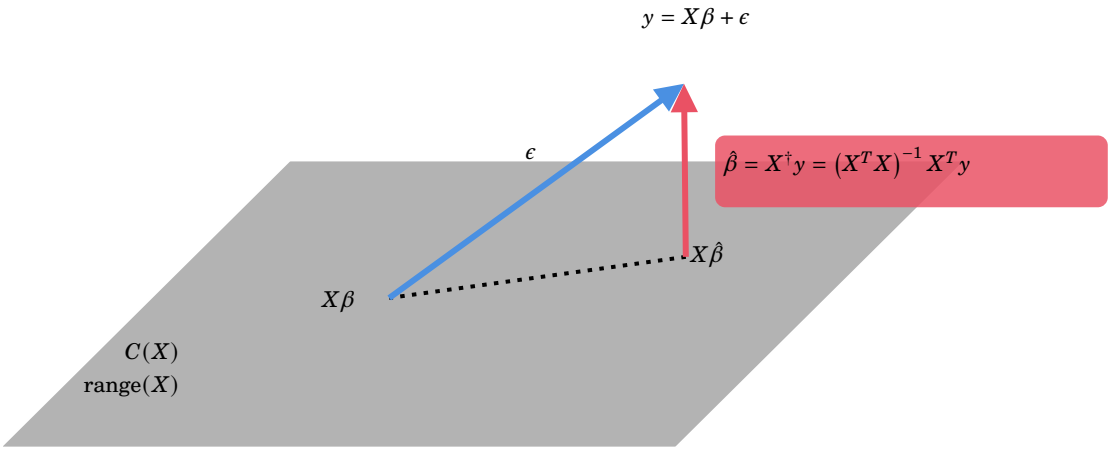
$$\mathbf{E}\epsilon = 0, \quad \mathbf{E}\epsilon\epsilon^T = \sigma^2 I$$

14.12.2 Least squares estimator

Theorem 14.12.6 least squares estimate $\hat{\beta}$ of parameters β, given the observations y, is

$$\hat{\beta} = X^\dagger y = \left(X^T X\right)^{-1} X^T y$$

Figure 14.1: Least squares estimator



$X\hat{\beta}$ is the orthogonal projection of y on range(X).
Residual $e = y - X\hat{\beta}$ is an (observable) random variable.

14.12.3 Mean and covariance of least squares estimate

Theorem 14.12.7

$$\hat{\beta} = X^{\dagger}(X\beta + \epsilon) = \beta + X^{\dagger}\epsilon$$

Theorem 14.12.8 least squares estimator is unbiased: $\mathbf{E}\hat{\beta} = \beta$

Theorem 14.12.9 covariance matrix of least squares estimate is

$$\begin{aligned}\mathbf{E}(\hat{\beta}-\beta)(\hat{\beta}-\beta)^T &= \mathbf{E}\left(\left(X^{\dagger}\epsilon\right)\left(X^{\dagger}\epsilon\right)^T\right)\\ &= \mathbf{E}\left(\left(X^TX\right)^{-1}X^T\epsilon\epsilon^TX\left(X^TX\right)^{-1}\right)\\ &= \sigma^2\left(X^TX\right)^{-1}\end{aligned}$$

Theorem 14.12.10 covariance of $\hat{\beta}_i$ and $\hat{\beta}_j (i \neq j)$ is

$$\mathbf{E}\left(\left(\hat{\beta}_i-\beta_i\right)\left(\hat{\beta}_j-\beta_j\right)\right) = \sigma^2\left(\left(X^TX\right)^{-1}\right)_{ij}$$

Corollary 14.12.11 for $i = j$, this is the variance of $\hat{\beta}_i$

14.12.4 Estimate of σ^2

Theorem 14.12.12 $\mathbf{E}\|\epsilon\|^2 = n\sigma^2$

Theorem 14.12.13 $\mathbf{E}\|e\|^2 = (n-p)\sigma^2$

Theorem 14.12.14 $\mathbf{E}\|X(\hat{\beta}-\beta)\|^2 = p\sigma^2$

Definition 14.12.1 — estimate $\hat{\sigma}$ of σ. define estimate $\hat{\sigma}$ of σ as

$$\hat{\sigma} = \frac{\|e\|}{\sqrt{n-p}}$$

Theorem 14.12.15 $\hat{\sigma}^2$ is an unbiased estimate of σ^2 :

$$\mathbf{E}\hat{\sigma}^2 = \frac{1}{n-p}\mathbf{E}\|e\|^2 = \sigma^2$$

Proof. first expression is immediate: $\mathbf{E}\|\epsilon\|^2 = \sum_{i=1}^{n}\mathbf{E}\epsilon_i^2 = n\sigma^2$

to show that $\mathbf{E}\|X(\hat{\beta}-\beta)\|^2 = p\sigma^2$, first note that

$$\begin{aligned} X(\hat{\beta}-\beta) &= XX^\dagger y - X\beta \\ &= XX^\dagger(X\beta+\epsilon) - X\beta \\ &= XX^\dagger\epsilon \\ &= X\left(X^TX\right)^{-1}X^T\epsilon \end{aligned}$$

on line 3 we used $X^\dagger X = I$ (however, note that $XX^\dagger \neq I$ if X is tall)

Theorem 14.12.16 squared norm of $X(\beta-\hat{\beta})$ is

$$\|X(\hat{\beta}-\beta)\|^2 = \epsilon^T\left(XX^\dagger\right)^2\epsilon = \epsilon^T XX^\dagger\epsilon$$

first step uses symmetry of $XX^\dagger$; second step, $X^\dagger X = I$
expected value of squared norm is

$$\begin{aligned} \mathbf{E}\|X(\hat{\beta}-\beta)\|^2 = \mathbf{E}\left(\epsilon^T XX^\dagger\epsilon\right) &= \sum_{i,j}\mathbf{E}\left(\epsilon_i\epsilon_j\right)\left(XX^\dagger\right)_{ij} \\ &= \sigma^2\sum_{i=1}^{n}\left(XX^\dagger\right)_{ii} \\ &= \sigma^2\sum_{i=1}^{n}\sum_{j=1}^{p}X_{ij}\left(X^\dagger\right)_{ji} \\ &= \sigma^2\sum_{j=1}^{p}\left(X^\dagger X\right)_{jj} \\ &= p\sigma^2 \end{aligned}$$

expression $\mathbf{E}\|e\|^2 = (n-p)\sigma^2$ on page 9.38 now follows from

$$\|\epsilon\|^2 = \|e + X\hat{\beta} - X\beta\|^2 = \|e\|^2 + \|X(\hat{\beta}-\beta)\|^2$$

■

14.12.5 Linear estimator

Problem 14.19 linear regression model (page 9.34), with same assumptions as before (p. 9.35):

$$y = X\beta + \epsilon$$

▪

Definition 14.12.2 a linear estimator of β maps observations y to the estimate

$$\hat{\beta} = By$$

Estimator is defined by the $p\times n$ matrix B, least squares estimator is an example with $B = X^\dagger$.

14.12.6 Unbiased linear estimator

Theorem 14.12.17 if B is a left inverse of X, then estimator $\hat{\beta} = By$ can be written as:

$$\hat{\beta} = By = B(X\beta + \epsilon) = \beta + B\epsilon$$

Corollary 14.12.18 this shows that the linear estimator is *unbiased* ($\mathbf{E}\hat{\beta} = \beta$) if $BX = I$.

Theorem 14.12.19 covariance matrix of unbiased linear estimator is

$$\mathbf{E}\left((\hat{\beta} - \beta)(\hat{\beta} - \beta)^T\right) = \mathbf{E}\left(B\epsilon\epsilon^T B^T\right) = \sigma^2 BB^T$$

Theorem 14.12.20 if c is a (non-random) p-vector, then estimate $c^T\hat{\beta}$ of $c^T\beta$ has variance $\mathbf{E}\left(c^T\hat{\beta} - c^T\beta\right)^2 = \sigma^2 c^T BB^T c$

Corollary 14.12.21 least squares estimator is an example with $B = X^\dagger$ and $BB^T = \left(X^T X\right)^{-1}$

14.13 Best linear unbiased estimator

Theorem 14.13.1 if B is a left inverse of X then for all p-vectors c

$$c^T BB^T c \geq c^T \left(X^T X\right)^{-1} c$$

Proof. use $BX = I$ to write BB^T as

$$\begin{aligned} BB^T &= \left(B - \left(X^T X\right)^{-1} X^T\right)\left(B^T - X\left(X^T X\right)^{-1}\right) + \left(X^T X\right)^{-1} \\ &= \left(B - X^\dagger\right)\left(B - X^\dagger\right)^T + \left(X^T X\right)^{-1} \end{aligned}$$

hence,

$$\begin{aligned} c^T BB^T C &= c^T \left(B - X^\dagger\right)\left(B - X^\dagger\right)^T c + c^T \left(X^T X\right)^{-1} c \\ &= \left\|\left(B - X^\dagger\right)^T c\right\|^2 + c^T \left(X^T X\right)^{-1} c \\ &\geq c^T \left(X^T X\right)^{-1} c \end{aligned}$$

with equality if $B = X^\dagger$ ∎

Corollary 14.13.2 left-hand side gives variance of $c^T\hat{\beta}$ for linear unbiased estimator

$$\hat{\beta} = By$$

Corollary 14.13.3 right-hand side gives variance of $c^T \hat{\beta}_{\text{ls}}$ for least squares estimator

$$\hat{\beta}_{\text{ls}} = X^{\dagger} y$$

Theorem 14.13.4 — Gauss-Markov theorem. Least squares estimator is the "best linear unbiased estimator" (BLUE).

This is known as the Gauss-Markov theorem.

15. Multi-objective Least Squares

15.1 Definition of Multi-objective Least Squares

Problem 15.1 假设有以下多个目标

$$J_1(x) = \|A_1x - b_1\|_2^2, \cdots, J_k(x) = \|A_kx - b_k\|_2^2$$

矩阵 $A_i \in \mathbb{R}^{m_i \times n}$, 向量 $b_i \in \mathbb{R}^{m_i}$;
寻找一个向量 $x \in \mathbb{R}^n$ 使得这 k 个目标 $_i(x), i = 1, \cdots, k$ 最小。
可以将上述多目标规划问题转换为加权最小二乘法问题。 ∎

Problem 15.2 — 加权最小二乘法问题.

$$\min_x J(x)$$

$$\begin{aligned} J(x) &= \lambda_1 \|A_1x - b_1\|_2^2 + \cdots + \lambda_k \|A_kx - b_k\|_2^2 \\ &= \left\|\sqrt{\lambda_1}\,(A_1x - b_1)\right\|_2^2 + \cdots + \left\|\sqrt{\lambda_k}\,(A_kx - b_k)\right\|_2^2 \end{aligned}$$

$\lambda_i > 0, i = 1, \cdots, k$, 表示不同目标的相对重要程度。
利用 ℓ_2 范数平方的可加性, 目标函数 $J(x)$ 可以写成紧密形式: ∎

Problem 15.3 — 加权最小二乘法问题紧密形式.

$$J(x) = \left\| \begin{bmatrix} \sqrt{\lambda_1}\,(A_1x - b_1) \\ \vdots \\ \sqrt{\lambda_k}\,(A_kx - b_k) \end{bmatrix} \right\|_2^2$$

∎

进一步可简化为

Problem 15.4 — 加权最小二乘法问题矩阵形式.

$$J(x) = \|\tilde{A}x - \tilde{b}\|_2^2$$

其中

$$\tilde{A}=\begin{bmatrix}\sqrt{\lambda_1}A_1\\ \vdots\\ \sqrt{\lambda_k}A_k\end{bmatrix},\quad \tilde{b}=\begin{bmatrix}\sqrt{\lambda_1}b_1\\ \vdots\\ \sqrt{\lambda_k}b_k\end{bmatrix}$$

■

因此将多目标问题转化为单目标问题，使用最小二乘法进行求解。

Problem 15.5 — **双目标规划问题.**

$$\min_x \|A_1x-b_1\|_2^2+\lambda\|A_2x-b_2\|_2^2, A_1, A_2\in\mathbb{R}^{10\times5}$$

λ 的变化会影响解的状态。 ■

15.2 求解多目标最小二乘问题

Problem 15.6

$$J(x)=\|\tilde{A}x-\tilde{b}\|_2^2, \tilde{A}=\begin{bmatrix}\sqrt{\lambda_1}A_1\\ \vdots\\ \sqrt{\lambda_k}A_k\end{bmatrix},\quad \tilde{b}=\begin{bmatrix}\sqrt{\lambda_1}b_1\\ \vdots\\ \sqrt{\lambda_k}b_k\end{bmatrix}$$

■

Theorem 15.2.1 如果 $\tilde{A}$ 的列向量线性无关时，则该问题的解唯一。

$$\begin{aligned}\hat{x}&=\left(\tilde{A}^T\tilde{A}\right)^{-1}\tilde{A}^T\tilde{b}\\ &=\left(\lambda_1{A_1}^TA_1+\cdots+\lambda_k{A_k}^TA_k\right)^{-1}\left(\lambda_1{A_1}^Tb_1+\cdots+\lambda_k{A_k}^Tb_k\right)\end{aligned}$$

可对 $\tilde{A}$ 进行 QR 分解计算 $\hat{x}$ 。每一个矩阵 A_i 的行向量可以线性相关。

15.3 正则化数据拟合

Problem 15.7 线性模型拟合数据 $\left(x^{(1)},y^{(1)}\right),\cdots,\left(x^{(N)},y^{(N)}\right)$

$$\hat{f}(x)=\theta_1f_1(x)+\cdots\theta_pf_p(x), \theta=\left[\theta_1,\cdots,\theta_p\right]^T$$

$f_1(x)$ 为常数函数，且恒等于 1 。 ■

较大的参数 θ_i 会让模型对 $f_i(x)$ 的变化更加敏感。让参数 $\theta_2,\ldots,\theta_p$ 更小，可以避免模型过拟合。

即可引出两个目标函数:

Problem 15.8

$$J_1(\theta)=\sum_{k=1}^{N}\left(\hat{f}\left(x^{(k)}\right)-y^{(k)}\right)^2,\quad J_2(\theta)=\sum_{j=2}^{p}\theta_j^2$$

首要目标 $J_1(\theta)$ 是误差的平方和。 ■

改写成加权最小二乘的形式

Problem 15.9

$$\min_\theta J_1(\theta)+\lambda J_2(\theta)=\sum_{k=1}^{N}\left(\hat{f}\left(x^{(k)}\right)-y^{(k)}\right)^2+\lambda\sum_{j=2}^{p}\theta_j^2$$

正则化参数 $\lambda>0$。 ■

该问题等价于最小二乘法问题:

Problem 15.10

$$\min_{\theta} \left\| \begin{bmatrix} A_1 \\ \sqrt{\lambda} A_2 \end{bmatrix} \theta - \begin{bmatrix} y_{d\times 1} \\ 0 \end{bmatrix} \right\|_2^2$$

$$A_1 = \begin{bmatrix} 1 & f_2\left(x^{(1)}\right) & \dots & f_p\left(x^{(1)}\right) \\ 1 & f_2\left(x^{(2)}\right) & \dots & f_p\left(x^{(2)}\right) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & f_2\left(x^{(N)}\right) & \dots & f_p\left(x^{(N)}\right) \end{bmatrix}, A_2 = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}, y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(N)} \end{bmatrix}$$

■

参数 λ 越大，会迫使 θ 得到接近零解。

15.4 图像逆问题

Problem 15.11

$$y = Ax_{ex} + v$$

向量 $x_{ex} \in \mathbb{R}^n$ 表示未知原始信息 (需要估计)，向量 $v \in \mathbb{R}^m$ 表示未知的误差或者噪声，向量 $y \in \mathbb{R}^m$ 为观测的已知数据，矩阵 $A \in \mathbb{R}^{m\times n}$ 将测量值 y 和原始信息 x_{ex} 之间的关系。

■

使用最小二乘估计法进行估计

Problem 15.12 — 最小二乘法进行估计.

$$\min_x \|Ax - y\|_2^2$$

利用未知 x_{ex} 先验信息，对目标进行约束，构成多目标优化问题。 ■

例如可以使用岭回归进行求解。

Definition 15.4.1 — Tikhonov 正则化.

$$\min_x \|Ax - y\|_2^2 + \lambda \|x\|_2^2, \lambda > 0$$

目标在于使 $\|Ax - y\|_2^2$ 足够小，同时 x 的能量也要小。

Theorem 15.4.1 — 岭回归问题的标准方程. 该优化模型等价于求解

$$\left(A^T A + \lambda I\right) x = A^T y$$

即使矩阵 A 的列线性相关时，也有唯一解。

15.4.1 差分矩阵平滑最小二乘解

Definition 15.4.2 — 图像转换为列向量存储. 二维图像 $X \in \mathbb{R}^{M\times N}$ ，可按列存储成向量 $x \in \mathbb{R}^{MN}$:

$$x = \begin{bmatrix} X_{1:M,1} \\ X_{1:M,2} \\ \vdots \\ X_{1:M,N} \end{bmatrix}$$

■ **Example 15.1 — reshape.**

$$X = \begin{bmatrix} 1 & 2 \\ 4 & 5 \end{bmatrix} \Rightarrow x = \begin{bmatrix} 1 \\ 4 \\ 2 \\ 5 \end{bmatrix}$$

■

对于前面的优化问题

$$y = Ax_{ex} + v$$

可以假设

Proposition 15.4.2 — 图像逆问题先验假设. 图像具有光滑性：**图像相邻两个像素值之间变化不大**。

- 水平方向：$X[n_1, n_2 + 1] \approx X[n_1, n_2]$
- 垂直方向：$X[n_1 + 1, n_2] \approx X[n_1, n_2]$

用差分矩阵进行平滑。

Definition 15.4.3 — 垂直差分矩阵. 垂直差分矩阵是大小为 $N \times N$ 的块矩阵，每块大小 $(M-1) \times M$。

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

■ **Example 15.2**

$$X = \begin{bmatrix} 1 & 2 \\ 4 & 5 \end{bmatrix} \Rightarrow x = \begin{bmatrix} 1 \\ 4 \\ 2 \\ 5 \end{bmatrix}, D_v x \Rightarrow \begin{bmatrix} 4-1 \\ 5-2 \end{bmatrix}$$

■

Definition 15.4.4 水平差分矩阵：大小为 $(N-1) \times N$ 的块矩阵，每块大小 $M \times M$：

$$D_h = \begin{bmatrix} -I_{M,M} & I_{M,M} & 0 & \cdots & 0 & 0 & 0 \\ 0 & -I_{M,M} & I_{M,M} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & -I_{M,M} & I_{M,M} \end{bmatrix}$$

■ **Example 15.3**

$$X = \begin{bmatrix} 1 & 2 \\ 4 & 5 \end{bmatrix} \Rightarrow x = \begin{bmatrix} 1 \\ 4 \\ 2 \\ 5 \end{bmatrix}, D_h x \Rightarrow \begin{bmatrix} 2-1 \\ 5-4 \end{bmatrix}$$

■

定义优化问题

Problem 15.13

$$\hat{x} = \arg\min_x \|Ax - y\|_2^2 + \lambda \|D_v x\|_2^2 + \lambda \|D_h x\|_2^2, \lambda > 0$$

$\|Ax - y\|_2^2$ 称为保证项: 保证 $A\hat{x} \approx y$.
$\lambda \|D_v x\|_2^2 + \lambda \|D_h x\|_2^2$ 为惩罚项，惩罚相邻像素值的差异变化

$$\|D_h x\|_2^2 + \|D_v x\|_2^2 = \sum_{i=1}^{M} \sum_{j=1}^{N-1} \left(X_{i,j+1} - X_{ij}\right)^2 + \sum_{i=1}^{M-1} \sum_{j=1}^{N} \left(X_{i+1,j} - X_{ij}\right)^2$$

■

求解这个模型可以得到图像逆退化的近似变换。

15.5 信号去噪

Problem 15.14 — 信号去噪问题. 观察信号向量 $y \in \mathbb{R}^n$ ，

$$y = x_{ex} + v$$

$x_{ex} \in \mathbb{R}^n$ 是未知信号, $v \in \mathbb{R}^n$ 是噪声。
目标是找一个近似信号 x 变换缓慢，信号既有光滑性，同时逼近 y ，其优化模型为

$$\min_x \|x - y\|_2^2 + \lambda \sum_{i=1}^{n-1} (x_{i+1} - x_i)^2, \lambda > 0$$

■

Definition 15.5.1 — 差分矩阵. 令矩阵 $D \in \mathbb{R}^{(n-1)\times n}$ 为差分矩阵:

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

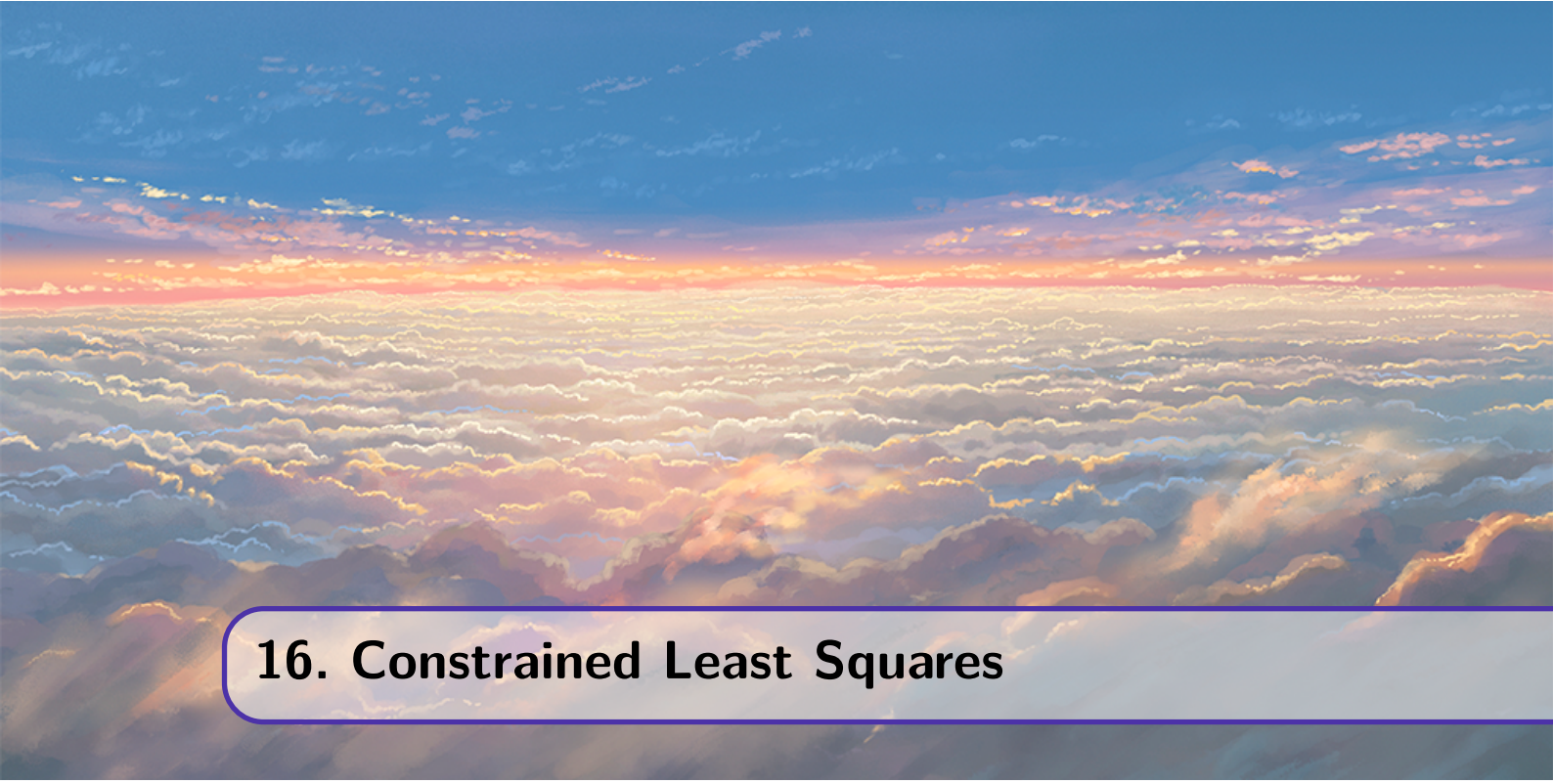
则有 $\sum_{i=1}^{n-1} (x_{i+1} - x_i)^2 = \|Dx\|_2^2$
优化模型等价于

$$\min_x \left\| \begin{bmatrix} I \\ \sqrt{\lambda} D \end{bmatrix} x - \begin{bmatrix} y \\ 0 \end{bmatrix} \right\|_2^2$$

优化模型等价于求解线性方程

$$\left(I + \lambda D^T D\right) x = y$$

当 $\lambda \to 0, \hat{x}(\lambda) \to y$; 当 $\lambda \to \infty, \hat{x}(\lambda) \to avg(y)1$



16. Constrained Least Squares

16.1 Karush–Kuhn–Tucker Conditions

Problem 16.1

$$\begin{aligned} &\min_x \left\{ f(x) = 2x_1^2 + x_2^2 \right\} \\ &\text{s.t.} \quad h(x) = x_1 + x_2 - 1 = 0 \end{aligned}$$

直接利用无约束优化问题求解：

$$\nabla f(x) = \begin{bmatrix} 4x_1 \\ 2x_2 \end{bmatrix} = 0 \Rightarrow x = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

显然不满足约束条件 $x_1 + x_2 - 1 = 0 + 0 - 1 \neq 0$, 不是优化问题的解。 ∎

由约束条件可得 $x_1 = 1 - x_2$ ，代入目标函数则有

$$f(x) = 3x_2^2 - 4x_2 + 2$$

即当 $\hat{x}_2 = \frac{2}{3}$ 时，目标函数值最小，并有 $\hat{x}_1 = 1 - \hat{x}_2 = \frac{1}{3}$ 。

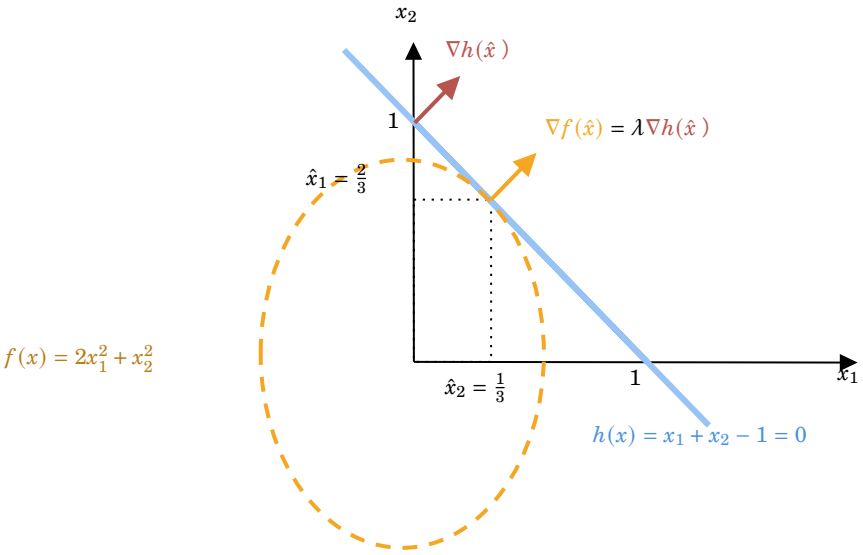
为了惩罚这个解未能满足约束条件, 引入拉格朗日函数 *(Lagrange Function)*

$$L(x, \lambda) = f(x) - \lambda h(x) = 2x_1^2 + x_2^2 + \lambda \left(1 - x_1 - x_2\right)$$

对拉格朗日函数进行求导

$$\left.\begin{aligned} \frac{\partial L}{\partial x_1} &= 4x_1 - \lambda = 0 \\ \frac{\partial L}{\partial x_2} &= 2x_2 - \lambda = 0 \\ \frac{\partial L}{\partial \lambda} &= 1 - x_1 - x_2 = 0 \end{aligned}\right\} \Rightarrow \hat{x}_1 = \frac{1}{3}, \hat{x}_2 = \frac{2}{3}, \hat{\lambda} = \frac{4}{3}$$

Figure 16.1: The Geometry of the problem



Definition 16.1.1 — **拉格朗日乘子法和 Lagrange Functions.**

$$\begin{aligned} \min_x / \max_x & f(x) \\ \text{s.t. } & h_i(x) = 0, i \in I \triangleq \{1, \cdots, p\} \\ & g_j(x) \leq 0, j \in J \triangleq \{1, \cdots, q\} \end{aligned}$$

$\lambda_i \in \mathbb{R}, i \in \mathrm{I}, u_j \in \mathbb{R}^+, j \in J$ 称为拉格朗日乘子 *(Lagrange Multipliers)* 。
引入拉格朗日函数

$$L(x, \lambda, u) = f(x) - \sum_{i \in I} \lambda_i h_i(x) - \sum_{j \in J} u_j g_j(x)$$

设 $\lambda = \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_p \end{bmatrix}, u = \begin{bmatrix} u_1 \\ \vdots \\ u_q \end{bmatrix}$,$h(x), g(x)$ 如果可以视为一个向量
则拉格朗日乘子式可写成

$$L(x, \lambda, u) = f(x) - \lambda^T h(x) - u^T g(x)$$

拉格朗日乘子法求得的解要使拉格朗日函数的梯度为 0。

Theorem 16.1.1 — **拉格朗日函数求导.** 对拉格朗日函数进行求导，有

$$\nabla_x L(x, \lambda, u) = \nabla_x f(x) - \sum_{i \in I} \lambda_i \nabla_x h_i(x) - \sum_{j \in J} u_j \nabla_x g_j(x) = 0$$

$$\nabla_x f(x) = \sum_{i \in I} \lambda_i \nabla_x h_i(x) + \sum_{j \in J} u_j \nabla_x g_j(x)$$

与图16.1中的结论相符。

Theorem 16.1.2 — Karush-Kuhn-Tucker Conditions. 对于优化问题

$$\begin{aligned}\min_x / \max_x & f(x)\\ \text{s.t. } & h_i(x)=0, i\in I \triangleq \{1,\cdots,p\}\\ & g_j(x)\le 0, j\in J\triangleq\{1,\cdots,q\}\end{aligned}$$

KKT 条件包括：

- $h_i(x)=0, i\in I$
- $\lambda_i h_i(x)=0, i\in I$
- $g_j(x)\le 0, j\in J$
- $u_j g_j(x)=0, j\in J$
- $u_j\ge 0, j\in J$

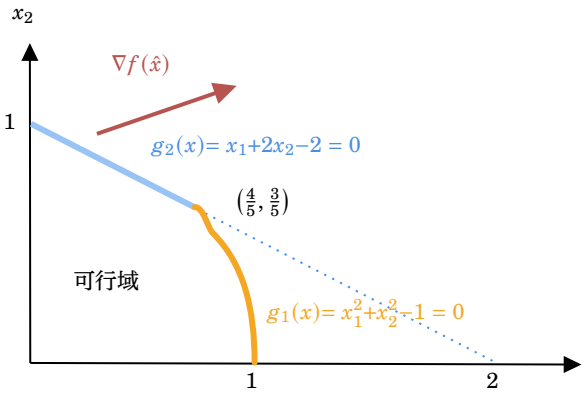
16.1.1 An Example for Karush-Kuhn-Tucker Conditions

Problem 16.2 求解

$$\begin{aligned}\max_x & \{f(x)=20x_1+10x_2\}\\ \text{s.t. } & g_1(x)=x_1^2+x_2^2\le 1\\ & g_2(x)=x_1+2x_2\le 2\\ & g_3(x)=-x_1\le 0\\ & g_4(x)=-x_2\le 0\end{aligned}$$

■

Figure 16.2: An Example for Karush-Kuhn-Tucker Conditions



对拉格朗日函数求导

$$\nabla_x L(x,u)=\nabla_x f(x)-u_1\nabla_x g_1(x)-u_2\nabla_x g_2(x)-u_3\nabla_x g_3(x)-u_4\nabla_x g_4(x)=0, u_j\ge 0$$

计算各个部分，可得

$$\nabla_x f(x)=\begin{bmatrix}20\\10\end{bmatrix}, \nabla_x g_1(x)=\begin{bmatrix}2x_1\\2x_2\end{bmatrix}, \nabla_x g_2(x)=\begin{bmatrix}1\\2\end{bmatrix}, \nabla_x g_3(x)=\begin{bmatrix}-1\\0\end{bmatrix}, \nabla_x g_4(x)=\begin{bmatrix}0\\-1\end{bmatrix}$$

检测边界条件，可知

$$\nabla_x f(x)=u_1\nabla_x g_1(x), \nabla_x f(x)\ne u_2\nabla_x g_2(x), \nabla_x f(x)\ne u_3\nabla_x g_3(x), \nabla_x f(x)\ne u_4\nabla_x g_4(x)$$

即 $\nabla_x f(x)$ 只可能是 $u_1\nabla_x g_1(x)$ 的线性组合。

所以

$$u_2 = u_3 = u_4 = 0, \quad u_1 \neq 0$$

即

$$\nabla_x f(x) = u_1 \nabla_x g_1(x)$$
$$\begin{bmatrix} 20 \\ 10 \end{bmatrix} = u_1 \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix}$$
$$x_1 = 2x_2$$

由 KKT 条件得 $u_j g_j(x) = 0, j \in J$。由于 $u_2 = u_3 = u_4 = 0, \quad u_1 \neq 0$，所以 $g_1(x) = 0$。将上式代入

$$g_1(x) = x_1^2 + x_2^2 - 1 = 5x_2^2 - 1 = 0$$

由于 $x_2 \geq 0$, 可得 $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \frac{\sqrt{5}}{5} \begin{bmatrix} 2 \\ 1 \end{bmatrix}, u_1 = 5\sqrt{5}.$

16.2 优化问题：最小范数优化问题

Problem 16.3

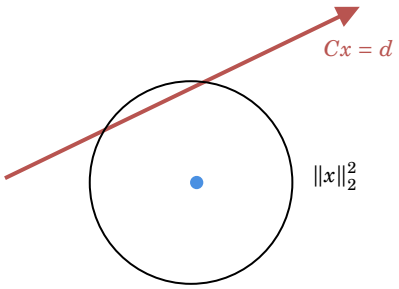
$$\min_x \|x\|_2^2$$
$$\text{s.t.} Cx = d$$

矩阵 $\mathrm{C} \in \mathbb{R}^{p \times n}$, 向量 $d \in \mathbb{R}^p$. ∎

在大多数应用中 $p < n, Cx = d$ 是一个**欠定方程**。这时，$Cx = d$ 表示为一条直线。它的几何意义是在直线 $Cx = d$ 中，寻找范数最小的解。

假如 $p = n$，$Cx = d$ 表示一个点。

Figure 16.3: 最小范数优化问题 ($p < n$)



假设矩阵 C 行向量线性无关时，有：

- 对任意一个 $\mathrm{d}, Cx = d$ 至少有一个解;
- 矩阵 C 为宽的或者方的 $(p \leq n)$;
- 当 $p < n$，有无穷多个解。

16.2.1 最小范数优化问题的例子

Problem 16.4 矩阵 $C \in \mathbb{R}^{2 \times 10}$, 向量 $d \in \mathbb{R}^2$:

$$Cx = d$$

$$\underbrace{\begin{bmatrix} 19/2 & 17/2 & 15/2 & \cdots & 1/2 \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix}}_{C} x = \underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{d}$$

这个方程只有一组含有两个非 0 元素的解。

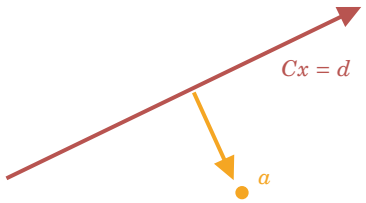
$$x = \begin{bmatrix} 1 \\ -1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad x = \begin{bmatrix} 0 \\ 1 \\ -1 \\ \vdots \\ 0 \end{bmatrix}, \cdots$$

16.3 优化问题：点到线的最短距离

Problem 16.5 给定点 $a \neq 0$，点到线最短距离问题:

$$\begin{array}{ll} \min_x & \|x-a\|_2^2 \\ \text{s.t.} & Cx = d \end{array}$$

Figure 16.4: 点到线的最短距离



令 $y = x - a$, 则点到线的最短距离问题，可等价为最小范数问题:

Problem 16.6

$$\begin{array}{ll} \min_y & \|y\|_2^2 \\ \text{s. t.} & Cy = d - Ca \end{array}$$

则 $x = y + a$ 是点到线最短距离问题的解。

16.4 最小范数优化问题推导

Problem 16.7

$$\begin{array}{ll} \min_x & \frac{1}{2}\|x\|_2^2 \\ \text{s.t.} & Cx = d \end{array}$$

求其最优解。
引入拉格朗日函数

$$L(x, \lambda) = \frac{1}{2}\|x\|_2^2 - \lambda^T (Cx - d)$$

λ 是一个向量。

对拉格朗日函数求导

$$\nabla_x L(x,\lambda) = x - C^T\lambda = 0 \Rightarrow x = C^T\lambda$$

由矩阵 C 行线性无关可得

$$Cx = CC^T\lambda = d \Rightarrow \lambda = \left(CC^T\right)^{-1} d$$

则有

$$\hat{x} = C^T\lambda = C^T\left(CC^T\right)^{-1} d = C^\dagger d$$

以上得到的是 $\hat{x} = C^\dagger d$ 是问题最优解的必要条件。还需要证明它是最优解。

Theorem 16.4.1 $\hat{x} = C^\dagger d$ 是问题最优解。

Proof. 1. 首先证明解 $\hat{x}$ 满足等式 $\hat{x} = C^T\lambda = C^T\left(CC^T\right)^{-1} d = C^\dagger d$.

$$C\hat{x} = CC^T\left(CC^T\right)^{-1} d = d$$

2. 在 $Cx = d, x \neq \hat{x}$ 的情况下

$$\begin{aligned}
\hat{x}^T(x-\hat{x}) &= d^T\left(CC^T\right)^{-1} C(x-\hat{x}) \\
&= d^T\left(CC^T\right)^{-1}(Cx - C\hat{x}) \\
&= d^T\left(CC^T\right)^{-1}(d-d) \\
&= 0
\end{aligned}$$

3. 在 $Cx = d, x \neq \hat{x}$ 的情况下，证明 $\|x\|_2^2 > \|\hat{x}\|_2^2$.

$$\begin{aligned}
\|x\|_2^2 &= \|\hat{x} + x - \hat{x}\|_2^2 \\
&= \|\hat{x}\|_2^2 + 2\hat{x}^T(x-\hat{x}) + \|x-\hat{x}\|_2^2 \\
&= \|\hat{x}\|_2^2 + \|x-\hat{x}\|_2^2 \\
&> \|\hat{x}\|_2^2
\end{aligned}$$

■

16.5 QR 分解求解最小范数优化问题

对矩阵 $C^T \in \mathbb{R}^{n\times p}$ 进行 QR 分解，$C^T = QR$

$$\begin{aligned}
\hat{x} &= C^T\left(CC^T\right)^{-1} d \\
&= QR\left(R^TQ^TQR\right)^{-1} d \\
&= QR\left(R^TR\right)^{-1} d \\
&= Q\left(R^{-1}\right)^T d
\end{aligned}$$

Algorithm 20: QR 分解求解最小范数优化问题

Input: QR 分解得到的矩阵 Q、R，向量 d
Output: $\hat{x}$
1 求解 $R^T z = d$
2 求解 $\hat{x} = Qz$

Proof.

$$\hat{x} = Q \underbrace{\left(R^T\right)^{-1} d}_{z}$$

$$z = \left(R^T\right)^{-1} d \Rightarrow R^T z = d$$

$$\hat{x} = Qz$$

■

16.5.1 QR 分解求解最小范数优化问题的复杂度

总复杂度：$\approx 2np^2$ flops

- 矩阵 $C^T = QR$ 分解，$C^T = QR$ $\left(2np^2 \text{ flops }\right)$
- 回代法求解 $R^T z = d$ $\left(p^2 \text{ flops }\right)$
- 计算 $\hat{x} = Qz (2np \text{ flops })$

16.5.2 QR 分解求解最小范数优化问题的例子

▪ **Example 16.1**

$$C = \begin{bmatrix} 1 & -1 & 1 & 1 \\ 1 & 0 & 1/2 & 1/2 \end{bmatrix}, \quad d = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

对矩阵 C^T 进行 QR 分解，$C^T = QR$

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

通过回代法求解 $R^T Z = d$

$$\begin{bmatrix} 2 & 0 \\ 1 & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

解得 $z_1 = 0, z_2 = \sqrt{2}$。
可得 $\hat{x} = Qz = (1, 1, 0, 0)$ ▪

16.6 最小二乘法约束问题

Problem 16.8 — **分段多项式拟合问题.**

$$\min_x \|Ax - b\|_2^2$$
$$\text{s.t.} Cx = d$$

矩阵 $A \in \mathbb{R}^{m \times n}, C \in \mathbb{R}^{p \times n}$, 向量 $b \in \mathbb{R}^m, d \in \mathbb{R}^p$ ▪

在大多数应用中 $p < n, Cx = d$ 是一个欠定方程。它的几何意义是在直线 $Cx = d$ 中，寻找 $\|Ax - b\|_2^2$ 最小的解。(此时 $\|Ax - b\|_2^2$ 表示一个椭圆)。

特殊情况：

- 当 $p = 0$ 时，则其转化为无约束的最小二乘法问题。
- 当 $A = I, b = 0$ 时，则其为最小范数问题。

16.7 优化问题：分段多项式拟合问题

Problem 16.9 假设样本点 $(x_1, y_1), \ldots, (x_N, y_N) \in \mathbb{R}^2$ ，有

$$x_1, \ldots, x_M \leq a, x_{M+1}, \ldots, x_N > a$$

设 d 阶多项式

$$f(x) = \theta_1 + \theta_2 x + \cdots + \theta_d x^{d-1}$$
$$g(x) = \theta_{d+1} + \theta_{d+2} x \ldots + \theta_{2d} x^{d-1}$$

两个多项式 $f(x), g(x)$ 对于样本点 $(x_1, y_1), \ldots, (x_N, y_N)$ 进行拟合

$$f(x_i) \approx y_i, x_i \leq a$$
$$g(x_i) \approx y_i, x_i > a$$

拟合要求：函数值和导数值必须在分段位置 a 连续

$$f(a) = g(a), f'(a) = g'(a)$$

对于 $\begin{cases} f(x_i) \approx y_i, x_i \leq a \\ g(x_i) \approx y_i, x_i > a \end{cases}$ ，可以构造矩阵

Notation 16.1.

$$A = \begin{bmatrix} \mathbf{1} & \boldsymbol{x_1} & \cdots & \boldsymbol{x_1^{d-1}} & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 1 & x_M & \cdots & x_M^{d-1} & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 1 & x_{M+1} & \cdots & x_{M+1}^{d-1} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & 1 & x_N & \cdots & x_N^{d-1} \end{bmatrix}, \theta = \begin{bmatrix} \boldsymbol{\theta_1} \\ \vdots \\ \boldsymbol{\theta_d} \\ \theta_{d+1} \\ \vdots \\ \theta_{2d} \end{bmatrix}, b = \begin{bmatrix} \boldsymbol{y_1} \\ \vdots \\ y_M \\ y_{M+1} \\ \vdots \\ y_N \end{bmatrix}$$

则可以转化为

$$A\theta \approx b$$

对于约束条件 $f(a) = g(a), f'(a) = g'(a)$，可以构造矩阵

Notation 16.2.

$$C = \begin{bmatrix} 1 & a & \cdots & a^{d-1} & -1 & -a & \cdots & -a^{d-1} \\ 0 & 1 & \cdots & (d-1)a^{d-2} & 0 & -1 & \cdots & -(d-1)a^{d-2} \end{bmatrix}, \theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_d \\ \theta_{d+1} \\ \vdots \\ \theta_{2d} \end{bmatrix}, d = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

所以

$$C\theta = d$$

对于分段多项式拟合问题

Problem 16.10 — 分段多项式拟合问题. 假设样本点 $(x_1, y_1), \ldots, (x_N, y_N) \in \mathbb{R}^2$，有 $x_1, \ldots, x_M \leq a, x_{M+1}, \ldots, x_N > a$。

$$\begin{array}{ll} \min_\theta & \sum_{i=1}^{M} \left(f\left(x_i\right) - y_i\right)^2 + \sum_{i=M+1}^{N} \left(g\left(x_i\right) - y_i\right)^2 \\ \text{s.t.} & f(a) = g(a), f'(a) = g'(a) \end{array}$$

■

最终可以转换成矩阵形式

Problem 16.11 — 分段多项式拟合问题（矩阵形式）.

$$\begin{aligned} &\min_\theta \|A\theta - b\|_2^2 \\ &\text{s.t.} C\theta = d \end{aligned}$$

■

16.8 先验假设

Proposition 16.8.1 堆叠矩阵的列向量线性无关

$$\begin{bmatrix} A \\ C \end{bmatrix} \in \mathbb{R}^{(m+p)\times n}$$

Proposition 16.8.2 矩阵 $C \in \mathbb{R}^{p\times n}$ 行线性无关

假设 1 是一个比 A 可右逆更弱的条件。
假设 $p \leq n \leq m + p$。

16.9 优化问题：最小二乘法的带约束 KKT 条件

Problem 16.12

$$\begin{aligned} &\min_x \frac{1}{2}\|Ax - b\|_2^2 \\ &s.t. Cx = d \end{aligned}$$

■

引入拉格朗日函数

$$L(x, z) = \frac{1}{2}\|Ax - b\|_2^2 - z^T(d - Cx), z \in \mathbb{R}^p$$

对拉格朗日函数求导

$$\begin{aligned} \nabla_x L(x, z) &= A^T(Ax - b) + C^T z = 0 \\ \nabla_z L(x, z) &= Cx - d = 0 \end{aligned} \tag{16.1}$$

使用矩阵导数

$$\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left(\mathbf{X}^T \mathbf{B} \mathbf{X}\right) = \mathbf{B}\mathbf{X} + \mathbf{B}^T\mathbf{X}$$
$$\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left(\mathbf{B}\mathbf{X}\mathbf{X}^T\right) = \mathbf{B}\mathbf{X} + \mathbf{B}^T\mathbf{X}$$
$$\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left(\mathbf{X}\mathbf{X}^T\mathbf{B}\right) = \mathbf{B}\mathbf{X} + \mathbf{B}^T\mathbf{X}$$
$$\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left(\mathbf{X}\mathbf{B}\mathbf{X}^T\right) = \mathbf{X}\mathbf{B}^T + \mathbf{X}\mathbf{B}$$
$$\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left(\mathbf{B}\mathbf{X}^T\mathbf{X}\right) = \mathbf{X}\mathbf{B}^T + \mathbf{X}\mathbf{B}$$
$$\frac{\partial}{\partial \mathbf{X}} \operatorname{Tr}\left(\mathbf{X}^T\mathbf{X}\mathbf{B}\right) = \mathbf{X}\mathbf{B}^T + \mathbf{X}\mathbf{B}$$

可以求得

$$\begin{aligned}\frac{\partial \frac{1}{2}\|Ax-b\|_2^2}{\partial x} &= \frac{\partial \frac{1}{2}\left(x^TA^TAx - 2x^TA^Tb + b^Tb\right)}{\partial x} \\ &= \frac{\partial \frac{1}{2}\operatorname{tr}\left(x^TA^TAx\right)}{\partial x} - \frac{\partial x^TA^Tb}{\partial x} \\ &= \frac{1}{2}\cdot 2A^TAx - A^Tb \\ &= A^T(Ax-b)\end{aligned}$$

可以转换成矩阵形式

$$\begin{bmatrix} A^TA & C^T \\ C & 0 \end{bmatrix}\begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} A^Tb \\ d \end{bmatrix}$$

16.10 KKT 最优条件

Problem 16.13

$$\min_x \frac{1}{2}\|Ax-b\|_2^2$$
$$s.t. Cx = d$$

■

Theorem 16.10.1 — 优化条件的 Karush-Kuhn-Tucker(KKT) 等式. 令 $\hat{x}$ 是上述约束优化问题的解，则有

$$\begin{bmatrix} A^TA & C^T \\ C & 0 \end{bmatrix}\begin{bmatrix} \hat{x} \\ z \end{bmatrix} = \begin{bmatrix} A^Tb \\ d \end{bmatrix}, \begin{bmatrix} \hat{x} \\ z \end{bmatrix} \in \mathbb{R}^{n+p}$$

则它是最优解。

特殊情况：
- 最小二乘法问题：当 $p=0$ 时，即为正规方程 $A^TA\hat{x} = A^Tb$
- 最小范数问题: 当 $A=I, b=0$ 时，可以推导得到 $C\hat{x} = b, \hat{x} + C^Tz = 0$

Proof. 假设 x 满足 $Cx=d, (\hat{x}, z)$ 满足 KKT 等式 $\begin{bmatrix} A^TA & C^T \\ C & 0 \end{bmatrix}\begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} A^Tb \\ d \end{bmatrix}$ 的定义，即 x 是另一个解。

所以有

$$\begin{aligned}\|Ax-b\|_2^2 &= \|A(x-\hat{x})+A\hat{x}-b\|_2^2 \\ &= \|A(x-\hat{x})\|_2^2+\|A\hat{x}-b\|_2^2+2(x-\hat{x})^TA^T(A\hat{x}-b) \\ &= \|A(x-\hat{x})\|_2^2+\|A\hat{x}-b\|_2^2-\underbrace{2(x-\hat{x})^T}_{(Cx=C\hat{x}=d)}\underbrace{C^Tz}_{\left(A^TA\hat{x}+C^Tz=A^Tb\right)} \\ &= \|A(x-\hat{x})\|_2^2+\|A\hat{x}-b\|_2^2 \\ &\geq \|A\hat{x}-b\|_2^2\end{aligned}$$

x 和 $\hat{x}$ 都是 KKT 等式的解，所以 $Cx=C\hat{x}=d$.
由拉格朗日函数求导16.1，有

$$A^TA\hat{x}+C^Tz=A^Tb$$

所以 $\hat{x}$ 是最优解。 ■

Theorem 16.10.2 — KKT 等式最优解的唯一性. 最优解 $\hat{x}$ 是唯一的。

Proof. 假设矩阵 A 列线性无关，C 行线性无关，即

$$\begin{aligned}A^TA(\hat{x}-x)=0 \Rightarrow x=\hat{x} \quad (A^TA\text{可逆}) \\ C^T(\hat{z}-z)=0 \Rightarrow \hat{z}=z\end{aligned}$$

所以对于方程

$$A^TA\hat{x}+C^Tz=A^Tb$$

$\begin{bmatrix} x \\ z \end{bmatrix}$ 也是唯一的。 ■

Theorem 16.10.3 如果矩阵 A 列线性无关，C 行线性无关，则矩阵

$$\begin{bmatrix} A^TA & C^T \\ C & 0 \end{bmatrix}$$

为非奇异矩阵。

Proof.

$$\begin{aligned}\begin{bmatrix} A^TA & C^T \\ C & 0 \end{bmatrix}\begin{bmatrix} x \\ z \end{bmatrix}=0 &\Rightarrow x^T\left(A^TAx+C^Tz\right)=0, Cx=0 \\ &\Rightarrow \|Ax\|_2^2+(Cx)^Tz=\|Ax\|_2^2=0, Cx=0 \\ &\Rightarrow Ax=0, Cx=0 \\ &\Rightarrow x=0 \quad (A\text{列线性无关})\end{aligned}$$

由于 $A^TAx+C^Tz=0$, 则当 $x=0$ 时，有 $z=0$ 。
所以 C 行线性无关。 ■

Theorem 16.10.4 如果矩阵 A 列线性无关和 C 行线性无关不同时成立，则矩阵

$$\begin{bmatrix} A^TA & C^T \\ C & 0 \end{bmatrix}$$

为奇异矩阵。

Proof. 如果 A 列线性相关，则存在 $x \neq 0$, 使得 $Ax = 0$, 则

$$\begin{bmatrix} A^TA & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} x \\ 0 \end{bmatrix} = 0$$

如果 C 行线性相关，则存在 $z \neq 0$, 使得 $C^Tz = 0$, 则

$$\begin{bmatrix} A^TA & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} 0 \\ z \end{bmatrix} = 0$$

因此该矩阵为奇异矩阵。 ■

16.11 LU 分解求解 KKT 最优条件

Problem 16.14

$$\min_x \frac{1}{2}\|Ax - b\|_2^2$$
$$s.t. Cx = d$$

$$\begin{bmatrix} A^TA & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} A^Tb \\ d \end{bmatrix}$$

求其最优解。 ■

Algorithm 21: LU 分解求解 KKT 最优条件

1 计算 $H = A^TA$
2 计算 $c = A^Tb$
3 LU 分解求解

$$\begin{bmatrix} H & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} c \\ d \end{bmatrix}$$

16.11.1 LU 分解求解 KKT 最优条件的时间复杂度

Theorem 16.11.1 用 LU 分解法求解下列线性方程

$$\begin{bmatrix} H & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} c \\ d \end{bmatrix}$$

需要的时间复杂度是 $\frac{2}{3}(p+n)^3$ flops。

第一步的时间复杂度是 $2mn^2$flops，第二步的时间复杂度是 $2mn$ flops，总时间复杂度为 $2mn^2 + \frac{2}{3}(p+n)^3$ flops 。

16.12 QR 分解求解 KKT 最优条件

Problem 16.15

$$\min_x \frac{1}{2}\|Ax-b\|_2^2$$
$$s.t. Cx = d$$

$$\begin{bmatrix} A^TA & C^T \\ C & 0 \end{bmatrix}\begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} A^Tb \\ d \end{bmatrix}$$

求其最优解。 ∎

由于 $\hat{x}$ 满足 $C\hat{x} = d$ ，则有

$$C^TC\hat{x} = C^Td$$

列出拉格朗日函数

$$L(x,z) = \frac{1}{2}\|Ax-b\|_2^2 - z^T(d-Cx)$$

由 KKT 条件可得

$$\begin{aligned} \nabla_x L(x,z) = A^T(A\hat{x}-b) + C^Tz + C^TC\hat{x} - C^Td = 0 \\ \Rightarrow \left(A^TA + C^TC\right)\hat{x} + C^T(z-d) = A^Tb \\ \nabla_z L(x,z) = C\hat{x} - d = 0 \end{aligned}$$

令 $w = z - d$ ，KKT 条件写成矩阵形式

$$\begin{bmatrix} A^TA + C^TC & C^T \\ C & 0 \end{bmatrix}\begin{bmatrix} \hat{x} \\ w \end{bmatrix} = \begin{bmatrix} A^Tb \\ d \end{bmatrix}$$

假设 1($\begin{bmatrix} A \\ C \end{bmatrix}$ 列向量无关，命题16.8.1) 保证了 $A^TA + C^TC$ 是非奇异的，即存在以下 QR 因子分解

$$\begin{bmatrix} A \\ C \end{bmatrix} = QR = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix}R = \begin{bmatrix} Q_1R \\ Q_2R \end{bmatrix}$$

代入 QR 分解，可得

$$\begin{bmatrix} R^TR & R^TQ_2^T \\ Q_2R & 0 \end{bmatrix}\begin{bmatrix} \hat{x} \\ w \end{bmatrix} = \begin{bmatrix} R^TQ_1^Tb \\ d \end{bmatrix}$$

将第一个方程两边乘 R^{-T} 和并令变量 $y = R\hat{x}$ ，可得

$$\begin{bmatrix} \left((R^T)^{-1}R^T\right)R & \left((R^T)^{-1}R^T\right)Q_2^T \\ Q_2R & 0 \end{bmatrix}\begin{bmatrix} \hat{x} \\ w \end{bmatrix} = \begin{bmatrix} \left((R^T)^{-1}R^T\right)Q_1^Tb \\ d \end{bmatrix}$$

$$\begin{bmatrix} R & Q_2^T \\ Q_2R & 0 \end{bmatrix}\begin{bmatrix} \hat{x} \\ w \end{bmatrix} = \begin{bmatrix} Q_1^Tb \\ d \end{bmatrix}$$

$$\begin{bmatrix} I & Q_2^T \\ Q_2 & 0 \end{bmatrix}\begin{bmatrix} R\hat{x} \\ w \end{bmatrix} = \begin{bmatrix} Q_1^Tb \\ d \end{bmatrix}$$

$$\begin{bmatrix} I & Q_2^T \\ Q_2 & 0 \end{bmatrix} \begin{bmatrix} y \\ w \end{bmatrix} = \begin{bmatrix} Q_1^T b \\ d \end{bmatrix}$$

矩阵 $C = Q_2 R \Rightarrow Q_2 = CR^{-1}, Q_2$ 行线性无关

$$Q_2^T u = R^{-T} C^T u = 0 \Rightarrow C^T u = 0 \Rightarrow u = 0$$

因为 C 行线性无关的 (假设 2(16.8.2))。

利用 Q_2^T 的 QR 分解来求解

$$\begin{bmatrix} I & Q_2^T \\ Q_2 & 0 \end{bmatrix} \begin{bmatrix} y \\ w \end{bmatrix} = \begin{bmatrix} Q_1^T b \\ d \end{bmatrix}$$

方程第 1 行可得 $y = Q_1^T b - Q_2^T w$ ，并代入第二行

$$Q_2 y = d \Rightarrow Q_2 Q_2^T w = Q_2 Q_1^T b - d$$

用 QR 分解 $Q_2^T = \tilde{Q}\tilde{R}$ 来解这个关于 w 的方程:

$$\tilde{R}^T \tilde{R} w = \tilde{R}^T \tilde{Q}^T Q_1^T b - d$$

上式可以简化为:

$$\tilde{R} w = \tilde{Q}^T Q_1^T b - \tilde{R}^{-T} d$$

$$\begin{bmatrix} A^T A + C^T C & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} \hat{x} \\ w \end{bmatrix} = \begin{bmatrix} A^T b \\ d \end{bmatrix} \quad \tilde{R} w = \tilde{Q}^T Q_1^T b - \tilde{R}^{-T} d$$

算法过程见22。

Algorithm 22: QR 分解求解 KKT 最优条件

1 计算两个 QR 分解

$$\begin{bmatrix} A \\ C \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} R, Q_2^T = \tilde{Q}\tilde{R}$$

2 用前代法求解 $\tilde{R}^T u = d$ ，计算 $c = \tilde{Q}^T Q_1^T b - u$

3 用回代法求解 $\tilde{R} w = c$ ，计算 $y = Q_1^T b - Q_2^T w$

4 用回代法计算 $R\hat{x} = y$。

总时间复杂度: QR 分解有 $2(p+m)n^2 + 2np^2$ 次 flops。

16.13 KKT 最优条件求解复杂度：QR vs LU

假设 $p < n$:

- LU 复杂度: $2mn^2 + (2/3)(p+n)^3 < 2mn^2 + (16/3)n^3$
- QR 复杂度: $2(p+m)n^2 + 2np^2 < 2mn^2 + 4n^3$

稳定性: QR 分解避免直接计算 $A^T A$。

16.14 Supplement Material: Karush-Kuhn-Tucker (KKT) 条件[1]

16.14.1 等式约束优化问题

Problem 16.16 — 等式约束优化问题. 给定一个目标函数 $f : \mathbb{R}^n \to \mathbb{R}$, 我们希望找到 $\mathbf{x} \in \mathbb{R}^n$, 在满足约束条件 $g(\mathbf{x}) = 0$ 的前提下, 使得 $f(\mathbf{x})$ 有最小值

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g(\mathbf{x}) = 0 \end{aligned}$$

■

为方便分析, 假设 f 与 g 是连续可导函数。Lagrange 乘数法是等式约束优化问题的典型解法。定义

Definition 16.14.1 — Lagrangian 函数.

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})$$

其中 λ 称为 *Lagrange 乘数*。

Theorem 16.14.1 Lagrange 乘数法将原本的约束优化问题转换成等价的无约束优化问题

$$\min_{\mathbf{x},\lambda} L(\mathbf{x}, \lambda)$$

Theorem 16.14.2 — 拉格朗日乘子法最优解必要条件. 计算 L 对 $\mathbf{x}$ 与 λ 的偏导数并设为零，可得最优解的必要条件：

$$\nabla_{\mathbf{x}} L = \frac{\partial L}{\partial \mathbf{x}} = \nabla f + \lambda \nabla g = \mathbf{0}$$

$$\nabla_{\lambda} L = \frac{\partial L}{\partial \lambda} = g(\mathbf{x}) = 0$$

其中第一式为*定常方程式 (stationary equation)*, 第二式为*约束条件*。

解开上面 $n+1$ 个方程式可得 $L(\mathbf{x}, \lambda)$ 的驻点 (stationary point) $\mathbf{x}^\star$ 以及 λ 的值 (正负数皆可能)。

16.14.2 不等式约束优化问题

接下来我们将约束等式 $g(\mathbf{x}) = 0$ 推广为不等式 $g(\mathbf{x}) \leq 0$ 。考虑这个问题

Problem 16.17 — 不等式约束优化问题.

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g(\mathbf{x}) \leq 0 \end{aligned}$$

约束不等式 $g(\mathbf{x}) \leq 0$ 称为*原始可行性 (primal feasibility)*, 据此我们定义*可行域 (feasible region)* $K = \{\mathbf{x} \in \mathbb{R}^n \mid g(\mathbf{x}) \leq 0\}$。 ■

假设 $\mathbf{x}^\star$ 为满足约束条件的最佳解, 分开两种情况讨论:

- $g(\mathbf{x}^*) < 0$, 最佳解位于 K 的内部, 称为内部解 (interior solution), 这时约束条件是*无效的 (inactive)*;
- $g(\mathbf{x}^*) = 0$, 最佳解落在 K 的边界, 称为边界解 (boundary solution), 此时约束条件是*有效的 (active)*.

[1] Cited from `https://zhuanlan.zhihu.com/p/38163970`.

这两种情况的最佳解具有不同的必要条件。

- 内部解：在约束条件无效的情形下，$g(\mathbf{x})$ 不起作用，约束优化问题退化为无约束优化问题，因此驻点 $\mathbf{x}^\star$ 满足 $\nabla f = \mathbf{0}$ 且 $\lambda = 0$ 。
- 边界解：在约束条件有效的情形下，约束不等式变成等式 $g(\mathbf{x}) = 0$，这与前述 Lagrange 乘数法的情况相同。

对于边界解，我们可以证明

Theorem 16.14.3 驻点 $\mathbf{x}^\star$ 发生于 $\nabla f \in \operatorname{span} \nabla g$。

换句话说，

Corollary 16.14.4 存在 λ 使得 $\nabla f = -\lambda \nabla g$。
注意这里 λ 的正负号是有其意义的。

因为我们希望最小化 f，梯度 ∇f (函数 f 在点 $\mathbf{x}$ 的最陡上升方向) 应该指向可行域 K 的内部 (因为最优解最小值是在边界取得的)，但 ∇g 指向 K 的外部 (即 $g(\mathbf{x}) > 0$ 的区域，因为你的约束是小于等于 0)，因此 $\lambda \geq 0$，称为对偶可行性 *(dual feasibility)*。

因此，不论是内部解或边界解，$\lambda g(\mathbf{x}) = 0$ 恒成立，称为互补松弛性 *(complementary slackness)*。

整合上述两种情况，

Theorem 16.14.5 — Karush-Kuhn-Tucker (KKT) 条件. 最佳解的必要条件包括：Lagrangian 函数 $L(\mathbf{x}, \lambda)$ 的定常方程式、原始可行性、对偶可行性，以及互补松弛性：

$$\begin{aligned} \nabla_{\mathbf{x}} L = \nabla f + \lambda \nabla g &= \mathbf{0} \\ g(\mathbf{x}) &\leq 0 \\ \lambda &\geq 0 \\ \lambda g(\mathbf{x}) &= 0 \end{aligned}$$

这些条件合称为 *Karush-Kuhn-Tucker (KKT)* 条件。

如果我们要最大化 $f(\mathbf{x})$ 且受限于 $g(\mathbf{x}) \leq 0$，那么对偶可行性要改成 $\lambda \leq 0$ 。

上面结果可推广至多个约束等式与约束不等式的情况。考虑标准约束优化问题 (或称非线性规划)：

Definition 16.14.2 — 标准约束优化问题 (非线性规划).

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_j(\mathbf{x}) = 0, \quad j = 1, \ldots, m \\ & h_k(\mathbf{x}) \leq 0, \quad k = 1, \ldots, p \end{aligned}$$

Theorem 16.14.6 — 标准约束优化的 KKT 条件. 定义 Lagrangian 函数

$$L\left(\mathbf{x}, \{\lambda_j\}, \{\mu_k\}\right) = f(\mathbf{x}) + \sum_{j=1}^{m} \lambda_j g_j(\mathbf{x}) + \sum_{k=1}^{p} \mu_k h_k(\mathbf{x})$$

其中 λ_j 是对应 $g_j(\mathbf{x}) = 0$ 的 Lagrange 乘数，μ_k 是对应 $h_k(\mathbf{x}) \leq 0$ 的 Lagrange 乘数 (或称 KKT 乘数)。

KKT 条件包括

$$\begin{aligned} \nabla_{\mathbf{x}} L &= \mathbf{0} \\ g_j(\mathbf{x}) &= 0, \quad j = 1, \ldots, m \\ h_k(\mathbf{x}) &\leq 0 \\ \mu_k &\geq 0 \\ \mu_k h_k(\mathbf{x}) &= 0, \quad k = 1, \ldots, p \end{aligned}$$

16.14.3 拉格朗日乘子法的例子

Problem 16.18 考虑这个问题

$$\begin{aligned} \min \quad & x_1^2 + x_2^2 \\ \text{s.t.} \quad & x_1 + x_2 = 1 \\ & x_2 \leq \alpha \end{aligned}$$

其中 $(x_1, x_2) \in \mathbb{R}^2, \alpha$ 为实数。 ∎

写出 Lagrangigan 函数

$$L(x_1, x_2, \lambda, \mu) = x_1^2 + x_2^2 + \lambda(1 - x_1 - x_2) + \mu(x_2 - \alpha)$$

KKT 方程组如下:

$$\begin{aligned} \frac{\partial L}{\partial x_i} &= 0, \quad i = 1, 2 \\ x_1 + x_2 &= 1 \\ x_2 - \alpha &\leq 0 \\ \mu &\geq 0 \\ \mu(x_2 - \alpha) &= 0 \end{aligned}$$

求偏导可得 $\frac{\partial L}{\partial x_1} = 2x_1 - \lambda = 0$ 且 $\frac{\partial L}{\partial x_2} = 2x_2 - \lambda + \mu = 0$, 分别解出 $x_1 = \frac{\lambda}{2}$ 且 $x_2 = \frac{\lambda}{2} - \frac{\mu}{2}$ 。代入约束等式 $x_1 + x_2 = \lambda - \frac{\mu}{2} = 1$ 或 $\lambda = \frac{\mu}{2} + 1$ 。合并上面结果,

$$x_1 = \frac{\mu}{4} + \frac{1}{2}, \quad x_2 = -\frac{\mu}{4} + \frac{1}{2}$$

最后再加入约束不等式 $-\frac{\mu}{4} + \frac{1}{2} \leq \alpha$ 或 $\mu \geq 2 - 4\alpha$ 。分开三种情况讨论。

1. $\alpha > \frac{1}{2}$: 不难验证 $\mu = 0 > 2 - 4\alpha$ 满足所有的 KKT 条件, 约束不等式是无效的, $x_1^\star = x_2^\star = \frac{1}{2}$ 是内部解，目标函数的极小值是 $\frac{1}{2}$。
2. $\alpha = \frac{1}{2}$: 如同 1, $\mu = 0 = 2 - 4\alpha$ 满足所有的 KKT 条件, $x_1^\star = x_2^\star = \frac{1}{2}$ 是边界解, 因为 $x_2^\star = \alpha$。
3. $\alpha < \frac{1}{2}$: 这时约束不等式是有效的, $\mu = 2 - 4\alpha > 0$, 则 $x_1^\star = 1 - \alpha$ 且 $x_2^\star = \alpha$, 目标函数的极小值是 $(1 - \alpha)^2 + \alpha^2$。

16.15 Supplement Material: 浅谈最优化问题的 KKT 条件[2]

Theorem 16.15.1 — KKT 条件. 对于具有等式和不等式约束的一般优化问题

$$\begin{aligned} & \min f(\mathbf{x}) \\ \text{s.t. } & g_j(\mathbf{x}) \leq 0 (j = 1, 2, \cdots, m) \\ & h_k(\mathbf{x}) = 0 (k = 1, 2, \cdots, l) \end{aligned}$$

[2]Cited from https://zhuanlan.zhihu.com/p/26514613.

KKT 条件给出了判断 $\mathbf{x}^*$ 是否为最优解的**必要条件**, 即:

$$\begin{cases} \frac{\partial f}{\partial x_i} + \sum_{j=1}^{m} \mu_j \frac{\partial g_j}{\partial x_i} + \sum_{k=1}^{l} \lambda_k \frac{\partial h_k}{\partial x_i} = 0, (i = 1, 2, \ldots, n) \\ h_k(\mathbf{x}) = 0, (k = 1, 2, \cdots, l) \\ \mu_j g_j(\mathbf{x}) = 0, (j = 1, 2, \cdots, m) \\ \mu_j \geq 0 \end{cases}$$

16.15.1 等式约束优化问题

等式约束优化问题是指

Problem 16.19 — 等式约束优化问题.

$$\begin{aligned} &\min f\left(x_1, x_2, \ldots, x_n\right) \\ &\text{s.t. } h_k\left(x_1, x_2, \ldots, x_n\right) = 0 \end{aligned}$$

■

我们令 $L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \sum_{k=1}^{l} \lambda_k h_k(\mathbf{x})$, 函数 $L(x, y)$ 称为 *Lagrange 函数*, 参数 λ 称为 *Lagrange 乘子*.

再联立方程组:

$$\begin{cases} \frac{\partial L}{\partial x_i} = 0 (i = 1, 2, \cdots, n) \\ \frac{\partial L}{\partial \lambda_k} = 0 (k = 1, 2, \cdots, l) \end{cases}$$

得到的解为可能极值点，由于我们用的是必要条件，具体是否为极值点需根据问题本身的具体情况检验. 这个方程组称为**等式约束的极值必要条件**.

上式我们对 n 个 x_i 和 l 个 λ_k 分别求偏导, 回想一下在*无约束优化问题*

$$f\left(x_1, x_2, \ldots, x_n\right) = 0$$

中, 我们根据极值的必要条件, 分别令 $\frac{\partial f}{\partial x_i} = 0$, 求出可能的极值点.

因此可以联想到：等式约束下的 Lagrange 乘数法引入了 l 个 Lagrange 乘子，或许我们可以**把 λ_k 也看作优化变量**（x_i 就叫做*优化变量*）. 相当于将优化变量个数增加到 $(n + l)$ 个, x_i 与 λ_k 一视同仁, 均为优化变量, 均对它们求偏导.

16.15.2 不等式约束优化问题

以上我们讨论了等式约束的情形，接下来我们来介绍不等式约束的优化问题.

我们先给出其主要思想：**转化**的思想——**将不等式约束条件变成等式约束条件**. 具体做法是**引入***松弛变量*. 松弛变量也是优化变量，也需要一视同仁求偏导.

具体而言, 我们先看一个一元函数的例子:

■ **Example 16.2**

$$\begin{aligned} &\min f(x) \\ &\text{s.t. } g_1(x) = a - x \leq 0 \\ &\quad\ g_2(x) = x - b \leq 0 \end{aligned}$$

■

优化问题中，我们必须求得一个确定的值，因此不妨令所有的不等式均取到等号，即 $\leq$ 的情况.

对于约束 g_1 和 g_2, 我们分别引入两个松弛变量 a_1^2 和 b_1^2, 得到 $h_1\left(x, a_1\right) = g_1 + a_1^2 = 0$ 和 $h_2\left(x, b_1\right) = g_2 + b_1^2 = 0$.

注意, 这里直接加上平方项 a_1^2、b_1^2 而非 a_1、b_1, 是因为 g_1 和 g_2 这两个不等式的左边必须加上一个正数才能使不等式变为等式. 若只加上 a_1 和 b_1, 又会引入新的约束 $a_1 \geq 0$ 和 $b_1 \geq 0$, 这不符合我们的意愿.

$$\begin{aligned} g_1(x) = a - x \leq 0 \\ g_2(x) = x - b \leq 0 \end{aligned} \Rightarrow \begin{aligned} h_1(x, a_1) = g_1(x) + a_1^2 = a - x + a_1^2 = 0 \\ h_2(x, b_1) = g_2(x) + b_1^2 = x - b + b_1^2 = 0 \end{aligned}$$

由此我们将不等式约束转化为了等式约束, 并得到 Lagrange 函数

$$L(x, a_1, b_1, \mu_1, \mu_2) = f(x) + \mu_1 \left(a - x + a_1^2\right) + \mu_2 \left(x - b + b_1^2\right)$$

我们再按照等式约束优化问题（极值必要条件）对其求解, 联立方程

$$\begin{cases} \dfrac{\partial F}{\partial x} = \dfrac{\partial f}{\partial x} + \mu_1 \dfrac{dg_1}{dx} + \mu_2 \dfrac{dg_2}{dx} = \dfrac{df}{dx} - \mu_1 + \mu_2 = 0 \\ \dfrac{\partial F}{\partial \mu_1} = g_1 + a_1^2 = 0, \\ \dfrac{\partial F}{\partial \mu_2} = g_2 + b_1^2 = 0 \\ \dfrac{\partial F}{\partial a_1} = 2\mu_1 a_1 = 0, \\ \dfrac{\partial F}{\partial b_1} = 2\mu_2 b_1 = 0 \\ \mu_1 \geq 0, \quad \mu_2 \geq 0 \end{cases}$$

对于不等式约束前的乘子, 我们要求其大于等于 0. ($\mu_1 \geq 0, \mu_2 \geq 0$)

得出方程组后, 便开始动手解它. 看到第 3 行的两式 $\mu_1 a_1 = 0$ 和 $\mu_1 a_1 = 0$ 比较简单, 我们就从它们入手吧.

对于 $\mu_1 a_1 = 0$, 我们有两种情况:

情形 1： $\mu_1 = 0, a_1 \neq 0$

此时由于乘子 $\mu_1 = 0$, 因此 g_1 与其相乘为零, 可以理解为约束 g_1 不起作用, 且有 $g_1(x) = a - x < 0$.

情形 2: $\mu_1 \geq 0, a_1 = 0$

此时 $g_1(x) = a - x = 0$ 且 $\mu_1 > 0$, 可以理解为约束 g_1 起作用, 且有 $g_1(x) = 0$.

合并情形 1 和情形 2 得: $\mu_1 g_1 = 0$**, 且在约束起作用时** $\mu_1 > 0, g_1(x) = 0$**; 约束不起作用时** $\mu_1 = 0, g_1(x) < 0$.

同样地, 分析 $\mu_2 b_1 = 0$, 可得出约束 g_2 起作用和不起作用的情形, 并分析得到 $\mu_2 g_2 = 0$.

由此,

Theorem 16.15.2 — 一元一次优化式的 KKT 条件. 方程组（极值必要条件）转化为

$$\begin{cases} \frac{df}{dx} + \mu_1 \frac{dg_1}{dx} + \mu_2 \frac{dg_2}{dx} = 0 \\ \mu_1 g_1(x) = 0, \mu_2 g_2(x) = 0 \\ \mu_1 \geq 0, \mu_2 \geq 0 \end{cases}$$

这是一元一次的情形. 类似地,

Corollary 16.15.3 对于多元多次不等式约束问题

$$\begin{aligned} &\min f(\mathbf{x}) \\ &\text{s.t. } g_j(\mathbf{x}) \leq 0 (j = 1, 2, \cdots, m) \end{aligned}$$

有

$$\begin{cases} \frac{\partial f(x^*)}{\partial x_i} + \sum_{j=1}^{m} \mu_j \frac{\partial g_j(x^*)}{\partial x_i} = 0 (i = 1, 2, \ldots, n) \\ \mu_j g_j(x^*) = 0 (j = 1, 2, \ldots, m) \\ \mu_j \geq 0 (j = 1, 2, \ldots, m) \end{cases}$$

上式便称为不等式约束优化问题的 KKT（Karush-Kuhn-Tucker）条件. μ_j 称为 KKT 乘子, 且约束起作用时 $\mu_j \geq 0, g_j(x) = 0$；约束不起作用时 $\mu_j = 0, g_j(x) < 0$.

16.15.3 KKT 乘子必须大于等于 0

Problem 16.20 还剩最后一个问题没有解决：为什么 KKT 乘子必须大于等于零？ ∎

我将用几何性质来解释. 由于

$$\frac{\partial f(x^*)}{\partial x_i} + \sum_{j=1}^{m} \mu_j \frac{\partial g_j(x^*)}{\partial x_i} = 0 (i = 1, 2, \ldots, n)$$

用梯度表示

$$\nabla f(\mathbf{x}^*) + \sum_{j \in J} \mu_j \nabla g_j(\mathbf{x}^*) = 0$$

J 为起作用约束的集合.

移项可得

$$-\nabla f(\mathbf{x}^*) = \sum_{j \in J} \mu_j \nabla g_j(\mathbf{x}^*)$$

注意到梯度为向量.

Theorem 16.15.4

$$-\nabla f(\mathbf{x}^*) = \sum_{j \in J} \mu_j \nabla g_j(\mathbf{x}^*)$$

在约束极小值点 $\mathbf{x}^*$ 处，函数 $f(\mathbf{x}^*)$ 的负梯度一定可以表示成：所有起作用约束在该点的梯度（等值线的法向量）的线性组合.

Corollary 16.15.5 — 梯度的性质. 复习课本中梯度的性质：某点梯度的方向就是函数等值线 $f(\mathbf{x}) = C$ 。（在这点的法线方向, 等值线就是地理的等高线。）

为方便作图, 假设现在**只有两个约束条件起约束作用**, 我们作出图形如图16.5.

注意我们上面推导过, 约束起作用时 $g_j(\mathbf{x}) = 0$, 所以此时约束在几何上应该是一簇**约束平面**.

我们假设在 $\mathbf{x}^*$ 取得极小值点, 若同时满足 $g_1(\mathbf{x}) = 0$ 和 $g_2(\mathbf{x}) = 0$, 则 $\mathbf{x}^k$ 一定在这两个平面的交线上, 且 $-\nabla f(\mathbf{x}^*) = \sum_{j \in J} \mu_j \nabla g_j(\mathbf{x}^*)$, 即 $-\nabla f(\mathbf{x}^k)$、$\nabla g_1(\mathbf{x}^k)$ 和 $\nabla g_2(\mathbf{x}^k)$ 共面.

图16.6是在点 $\mathbf{x}^k$ 处沿 $x_1 O x_2$ 面的截面, 过点 $\mathbf{x}^k$ 作目标函数的负梯度 $-\nabla f(\mathbf{x}^k)$, 它垂直于目标函数的等值线 $f(\mathbf{x}) = C$ ，且指向目标函数 $f(\mathbf{x})$ 的最速减小方向.

Figure 16.5

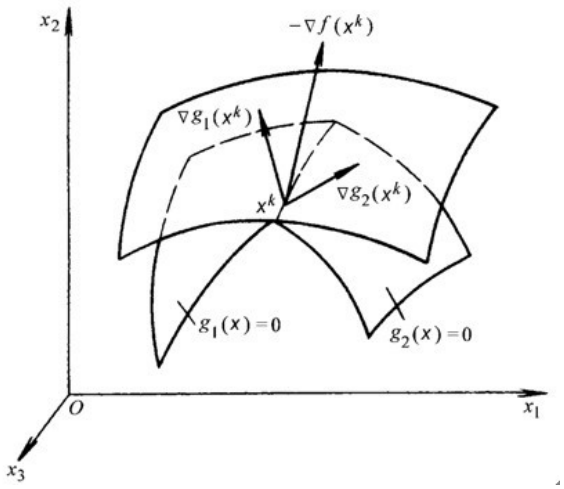
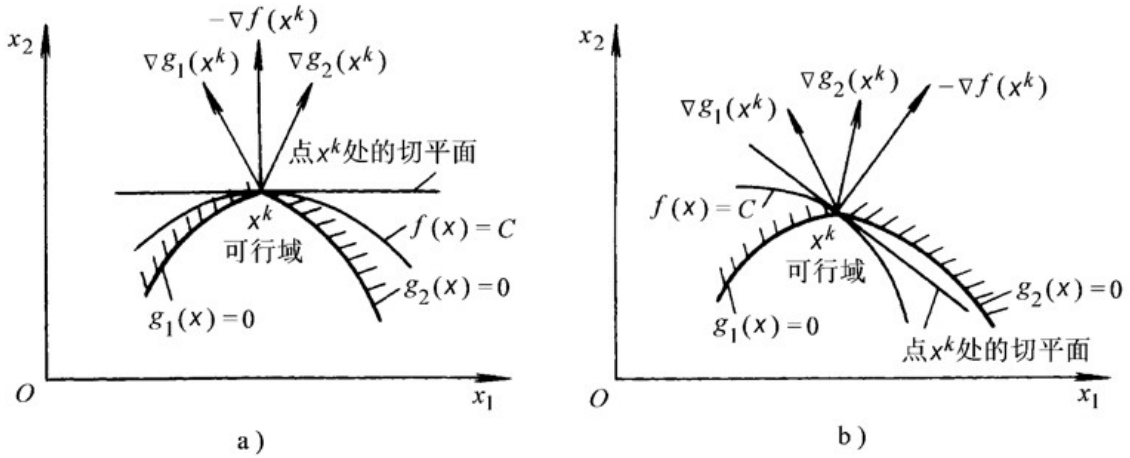


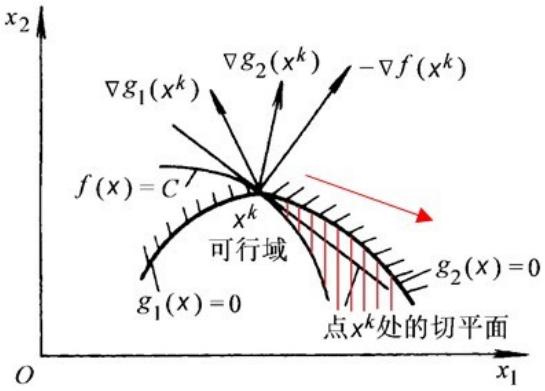
Figure 16.6



Corollary 16.15.6 一点的梯度与等值线相互垂直。

再作约束函数 $g_1(\mathbf{x}) = 0$ 和 $g_2(\mathbf{x}) = 0$ 的梯度 $\nabla g_1\left(\mathbf{x}^k\right)$ 和 $\nabla g_2\left(\mathbf{x}^k\right)$, 它们分别垂直 $g_1(\mathbf{x}) = 0$ 和 $g_2(\mathbf{x}) = 0$ 两曲面在 $\mathbf{x}^k$ 的切平面, 并形成一个雉形夹角区域. 此时, 可能有 a、 b 两种情形:

Figure 16.7



b)

我们先来看情形 b : 若 3 个向量的位置关系如16.7所示, 即 $-\nabla f$ 落在 ∇g_1 和 ∇g_2 所形成的锥角区外的一侧. 此时, 作等值面 $f(\mathbf{x}) = C$ 在点 $\mathbf{x}^k$ 的切平面 (它与 $-\nabla f\left(\mathbf{x}^k\right)$ 垂直), 我们发现: 沿着与负梯度 $-\nabla f$ 成锐角的方向移动 (如下图红色箭头方向), 只要在红色区域取值, 目标函数 $f(\mathbf{x})$ 总能减小. 而红色区域是可行域 ($f(\mathbf{x}) = C$, C 取不同的常数能得到不同的等值线, 因此能取到红色区域), 因此既可减小目标函数值, 又不破坏约束条件. 这说明 $\mathbf{x}^k$ 仍可沿约束曲面移动而不破坏约束条件, 且目标函数值还能够减小. 所以 $\mathbf{x}^k$ 不是稳定的最优点, 即不是局部极值点.

反过头来看情形 a: $-\nabla f$ 落在 ∇g_1 和 ∇g_2 形成的锥角内. 此时, 同样作 $f(\mathbf{x}) = C$ 在点 $\mathbf{x}^k$ 与 $-\nabla f$ 垂直的切平面. 当从 $\mathbf{x}^k$ 出发沿着与负梯度 $-\nabla f$ 成锐角的方向移动时, 虽然能使目标函数值减小, 但此时任何一点都不在可行区域内. 显然, 此时 $\mathbf{x}^k$ 就是局部最优点 $\mathbf{x}^*$, 再做任何移动都将破坏约束条件, 故它是稳定点.

由于 $-\nabla f\left(\mathbf{x}^*\right)$ 和 $\nabla g_1\left(\mathbf{x}^*\right)$ 、$\nabla g_2\left(\mathbf{x}^*\right)$ 在一个平面内, 所以前者可看成是后两者的线性组合. 又由上面的几何分析知, $-\nabla f\left(\mathbf{x}^*\right)$ 在 $\nabla g_1\left(\mathbf{x}^*\right)$ 和 $\nabla g_2\left(\mathbf{x}^*\right)$ 的夹角之间, 所以线性组合的系数为正, 有

$$-\nabla f\left(\mathbf{x}^*\right) = \mu_1 \nabla g_1\left(\mathbf{x}^*\right) + \mu_2 \nabla g_2\left(\mathbf{x}^*\right), \text{ 且} \mu_1 > 0, \mu_2 > 0.$$

这就是 $\mu_j > 0$ 的原因. 类似地, 当有多个不等式约束同时起作用时, 要求 $-\nabla f\left(\mathbf{x}^*\right)$ 处于 $\nabla g_j\left(\mathbf{x}^*\right)$ 形成的超角锥 (高维图形, 我姑且称之为“超”) 之内.

16.15.4 总结: 同时包含等式和不等式约束的一般优化问题

Theorem 16.15.7 — 同时包含等式和不等式约束的一般优化问题的 KKT 条件.

$$\begin{aligned} &\min f(\mathbf{x}) \\ &\text{s.t. } g_j(\mathbf{x}) \leq 0 (j = 1, 2, \cdots, m) \\ &h_k(\mathbf{x}) = 0 (k = 1, 2, \cdots, l) \end{aligned}$$

KKT 条件 ($\mathbf{x}^*$ 是最优解的必要条件) 为

$$\begin{cases} \frac{\partial f}{\partial x_i} + \sum_{j=1}^{m} \mu_j \frac{\partial g_j}{\partial x_i} + \sum_{k=1}^{l} \lambda_k \frac{\partial h_k}{\partial x_i} = 0, (i = 1, 2, \ldots, n) \\ h_k(\mathbf{x}) = 0, (k = 1, 2, \cdots, l) \\ \mu_j g_j(\mathbf{x}) = 0, (j = 1, 2, \cdots, m) \\ \mu_j \geq 0 \end{cases}$$

对于等式约束的 Lagrange 乘子，并没有非负的要求。

以后求其极值点，不必再引入松弛变量，直接使用 KKT 条件判断。

16.16 Supplement Material: 凸优化、拉格朗日乘子法和 KKT 条件[3]

16.16.1 凸集的概念

Definition 16.16.1 — **点、(直) 线、线段.** $x_1 \neq x_2$ 是 $\mathbb{R}^n$ 中的两点, $\theta \in \mathbb{R}$, 那么 $y = x_2 + \theta(x_1 - x_2)$ 表示穿过两点的线。

当 $0 \leqslant \theta \leqslant 1$ 时, y 是 x_1 到 x_2 的线段，y 也可以表示成 $y = \theta x_1 + (1-\theta)x_2$。

Definition 16.16.2 — **仿射集.** 一个集合 $C \subseteq \mathbb{R}^n$ 是仿射集如果其中任意两个不同的点的连线仍包含于 C 。

Definition 16.16.3 — **凸集.** 一个集合 $C \subseteq \mathbb{R}^n$ 是凸集, 如果任意两点之间的线段仍包含于 C 。即 $\forall x_1, x_2 \in C$, 任意 $0 \leqslant \theta \leqslant 1$, 有 $\theta x_1 + (1-\theta)x_2 \in C$ 。

Theorem 16.16.1 两个凸集的交集仍是凸集。

Definition 16.16.4 — **凸函数.** 一个函数 $f : \mathbb{R}^n \to \mathbb{R}$ 是凸函数如果定义域是凸集而且对任意定义域的 $x, y, 0 \leqslant \theta \leqslant 1$ 有

$$f(\theta x + (1-\theta)y) \leqslant \theta f(x) + (1-\theta)f(y)$$

二维时候的几何意义是, 如果两点的线段总位于函数曲线之上, 那么该函数是凸函数。

Theorem 16.16.2 f 是凸的, 那么 $-f$ 是凹的。

Theorem 16.16.3 仿射函数既凸又凹。

Theorem 16.16.4 — **α-sublevel 集.** 给定凸函数 f，则 $\{x \in D(f) : f(x) \leqslant \alpha\}$ 是一个凸集。

Proof. 对任意 $x, y \in D(f)$ 使得

$$f(x) \leqslant \alpha, f(y) \leqslant \alpha$$

[3] Cited from https://zhuanlan.zhihu.com/p/59928816.

有

$$f(\theta x + (1-\theta)y) \leqslant \theta f(x) + (1-\theta)f(y) \leqslant \theta\alpha + (1-\theta)\alpha$$

也就是说如果 x, y 属于 $\alpha-$ sublevel 集，那么二者之间的线段上的点也可以使得 $f \leqslant \alpha$，即二者之间的线段也包含于 $\alpha-$ sublevel 集。 ■

16.16.2 凸优化

Definition 16.16.5 — 优化问题. 优化问题有如下形式：

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leqslant b_i, i = 1, \ldots, m \end{array}$$

$x = (x_1, \ldots, x_n)$ 是优化变量，函数 $f_0 : \mathbb{R}^n \to \mathbb{R}$ 是目标函数，函数 $f_i : \mathbb{R}^n \to \mathbb{R}$ 是约束函数。b_i 是约束。使得 $f_0(x)$ 在约束条件下最小的 x^* 叫做最优点，或问题的解。

Definition 16.16.6 — 凸优化. 凸优化问题有如下形式：

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in C \end{array}$$

f 是凸函数，C 是凸集。

Corollary 16.16.5 凸集可以表示成某些凸集的交集。

所以凸优化问题一般表示为

Definition 16.16.7 — 凸优化问题.

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g_i(x) \leqslant 0, i = 1, \ldots, m \\ & h_j(x) = 0, j = 1, \ldots, p \end{array}$$

其中 g_i 是凸函数，h_j 是仿射函数。

$g_i(x) \leqslant 0$ 叫做 $0-$*sublevel* 集，是凸集。

Theorem 16.16.6 $h_j(x) = 0$ 也是凸集。

Proof.

$$h\left(\theta x_1 + (1-\theta)x_2\right) = \theta h\left(x_1\right) + (1-\theta)h\left(x_2\right) = 0$$

■

凸优化问题的特点是，**所有局部最优点都是全局最优点**。

Definition 16.16.8 — 二次规划. 如果一个凸优化问题的 g_i 都是仿射函数，且 f 是凸二次函数，那么它叫做二次规划：

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^\top Px + c^\top x + d \\ \text{subject to} & g_i(x) \leqslant 0, i = 1, \ldots, m \\ & h_j(x) = 0, j = 1, \ldots, p \end{array}$$

其中 P 是一个对称半正定矩阵 (使得 $\frac{1}{2}x^\top Px \geqslant 0$) 。

16.16.3 拉格朗日对偶性

Theorem 16.16.7 对于没有限制的凸函数, 最优点 x^* 一定满足 $\nabla_x f(x^*) = 0$。

然而对于有限制条件的凸优化问题却不是这样。拉格朗日对偶性可以将有限制的凸优化问题转化为没有限制的问题, 来求解凸优化问题。

Definition 16.16.9 — **拉格朗日函数.** 给定一个凸优化问题, 拉格朗日算子是一个函数 $\mathcal{L}: \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$, 定义为:

$$\mathcal{L}(x, \alpha, \beta) = f(x) + \sum_{i=1}^{m} \alpha_i g_i(x) + \sum_{i=1}^{p} \beta_i h_i(x)$$

$x \in \mathbb{R}^n$ 叫做主变量 *(primal variable)*。$\alpha \in \mathbb{R}^m, \beta \in \mathbb{R}^p$ 统称对偶变量或拉格朗日乘子。

Theorem 16.16.8 总存在一个拉格朗日乘子, 使得没有限制的拉格朗日算子相对于 x 的最小值, 等于原凸优化问题的最优值

后文会证明。

主问题

为了说明拉格朗日算子和原凸优化问题的关系，需要引入主问题和对偶问题。

考虑优化问题:

Problem 16.21

$$\min_x \left[\max_{\alpha,\beta,\alpha_i \geqslant 0} \mathcal{L}(x, \alpha, \beta) \right] = \min_x \theta_{\mathcal{P}}(x)$$

括号内的 $\theta_{\mathcal{P}}: \mathbb{R}^n \to \mathbb{R}$ 叫做主目标 *(primal objective)*, 等号右边的没有限制的最小化问题叫做主问题 *(primal problem)*。用 x^* 表示主问题的解,$p^* = \theta_{\mathcal{P}}(x^*)$ 表示主目标的最优值。 ∎

Definition 16.16.10 — **primal feasible.** x 是主可行的 *(primal feasible)*如果 $g_i(x) \leqslant 0, h_j(x) = 0$ 。

$$\begin{aligned}
\theta_{\mathcal{P}}(x) &= \max_{\alpha,\beta,\alpha_i \geqslant 0} \mathcal{L}(x, \alpha, \beta) \\
&= \max_{\alpha,\beta,\alpha_i \geqslant 0} \left[f(x) + \sum_{i=1}^{m} \alpha_i g_i(x) + \sum_{i=1}^{p} \beta_i h_i(x) \right] \\
&= f(x) + \max_{\alpha,\beta,\alpha_i \geqslant 0} \left[\sum_{i=1}^{m} \alpha_i g_i(x) + \sum_{i=1}^{p} \beta_i h_i(x) \right]
\end{aligned}$$

观察最后一个式子:

- 如果任何一个 $g_i(x) > 0$, 那么最大化 $\theta_{\mathcal{P}}(x)$ 只需要将相应的 α_i 设置为无限大。
- 如果 $g_i(x) \leqslant 0$, 因为 $\alpha_i \geqslant 0$, 所以 $\theta_{\mathcal{P}}(x)$ 最大时, 必然有 $\alpha_i = 0$ 。

相似地:

- 如果 $h_i \neq 0$, 那么最大化 $\theta_{\mathcal{P}}(x)$ 只需要将相应的 β_i 设置为 $h_i(x)$ 的相同符号且绝对值无限大。
- 如果 $h_i(x) = 0$, 那么 $\sum_{i=1}^{p} \beta_i h_i(x)$ 项的最大值为 0 。

综上, 有

$$\theta_{\mathcal{P}}(x)=\begin{cases} f(x)+0 & x \text{ 为主可行的} \\ f(x)+\infty & x \text{ 不是主可行的} \end{cases}$$

因此当 x 主可行时, 主问题16.21的最优值等于原凸优化问题16.16.7的最优值。

对偶问题

Definition 16.16.11 — 对偶问题. 对调主问题的最大最小操作:

$$\max_{\alpha,\beta,\alpha\geqslant 0}\left[\min_{x}\mathcal{L}(x,\alpha,\beta)\right]=\max_{\alpha,\beta,\alpha\geqslant 0}\theta_{\mathcal{D}}(\alpha,\beta)$$

$\theta_{\mathcal{D}}(\alpha,\beta):\mathbb{R}^m\times\mathbb{R}^p\to\mathbb{R}$ 是对偶目标 *(dual objective)*，等号右边的有限制的最大化问题叫做对偶问题。用 (α^*,β^*) 表示对偶问题的解，$d^*=\theta_{\mathcal{D}}(\alpha^*,\beta^*)$ 表示对偶目标的最优值。

Definition 16.16.12 — 对偶可行. 如果 $\alpha_i(x)\geqslant 0$ ，(α,β) 是对偶可行的。

Theorem 16.16.9 如果 (α,β) 是对偶可行的，那么 $\theta_{\mathcal{D}}(\alpha,\beta)\leqslant p^*$.

Proof.

$$\begin{aligned}\theta_{\mathcal{D}}(\alpha,\beta)&=\min_{x}\mathcal{L}(x,\alpha,\beta)\\&\leqslant\mathcal{L}\left(x^*,\alpha,\beta\right)\\&=f\left(x^*\right)+\sum_{i=1}^{m}\alpha_i g_i\left(x^*\right)+\sum_{i=1}^{p}\beta_i h_i\left(x^*\right)\\&\leqslant f\left(x^*\right)=p^*\end{aligned}$$

■

Theorem 16.16.10 — 弱对偶性. 对任意主问题和对偶问题，有 $d^*\leqslant p^*$.

Theorem 16.16.11 — 强对偶性. 对任意主问题和对偶问题, 如果满足某个条件 (constraint qualifications), 那么 $d^*=p^*$ 。最常用的 constraint qualification 是 Slater's condition: 所有的不等式限制都严格满足 (即 $g_i(x)<0$) 。

Theorem 16.16.12 — Slater 条件. 设定义在 $\mathcal{D}$ 上的函数 $f_i(\cdot),i=1,2,\cdots,n$ 为凸函数, $g_j(\cdot),j=1,2,\cdots,m$ 为仿射函数, 考虑凸优化问题

$$\min_{\mathbf{x}} f_0(\mathbf{x}),\quad \text{s.t. } f_i(\mathbf{x})\leq 0, g_i(\mathbf{x})\leq 0$$

如果存在点 $\mathrm{x}\in\operatorname{relint}\mathcal{D}$ （即 $\mathcal{D}$ 的相对内点）, 则强对偶性成立.

实践中, 几乎所有的凸优化问题都满足某种 constraint qualification, 所以主问题和对偶问题有相同的最优值。

Theorem 16.16.13 — 互补松弛性 (complementary slackness, KKT complementarity). 如果强对偶性满足, 那么 $\alpha_i^*g\left(x_i^*\right)=0,i=1,\ldots,m$

Proof.

$$\begin{aligned} p^* = d^* = \theta_{\mathcal{D}}\left(\alpha^*, \beta^*\right) &= \min_x \mathcal{L}\left(x, \alpha^*, \beta^*\right) \\ &\leqslant \mathcal{L}\left(x^*, \alpha^*, \beta^*\right) \\ &= f\left(x^*\right) + \sum_{i=1}^{m} \alpha_i^* g_i\left(x^*\right) + \sum_{i=1}^{p} \beta_i^* h_i\left(x^*\right) \\ &\leqslant f\left(x^*\right) = p^* \end{aligned}$$

因为第一个和最后一个表达式相等, 所以中间所有的小于等于号都是等号, 有

$$\sum_{i=1}^{m} \alpha_i^* g_i\left(x^*\right) + \sum_{i=1}^{p} \beta_i^* h_i\left(x^*\right) = 0$$

因为 $\alpha_i^* \geqslant 0$，$h_i\left(x^*\right) = 0$, 所以 α_i^* 和 $g_i\left(x^*\right)$ 至少有一个是 0 。 ■

Theorem 16.16.14 设 $x^* \in \mathbb{R}^n, \alpha^* \in \mathbb{R}^m, \beta^* \in \mathbb{R}^p$, 下列条件为 KKT 条件:

1. (主可行) $g_i\left(x^*\right) \leqslant 0, i = 1, \ldots, m, h_j\left(x^*\right) = 0, j = 1, \ldots, p$
2. (对偶可行) $\alpha_i^* \geqslant 0, i = 1, \ldots, m$
3. (互补松弛性) $\alpha_i^* g\left(x_i^*\right) = 0, i = 1, \ldots, m$
4. (Lagrangian Stationary) $\nabla_x \mathcal{L}\left(x^*, \alpha^*, \beta^*\right) = 0$

Theorem 16.16.15 对于凸优化问题, 有:

x^* 原始最优，(α^*, β^*) 对偶最优，且有

强对偶性 $\Leftrightarrow$ 满足KKT条件

16.16.4 总结

给定一个凸优化问题, 拉格朗日算子将凸优化问题的目标函数和限制考虑进一个函数中, 在拉格朗日算子基础上可以定义主问题和对偶问题。

主问题是先调整 (α, β) 使拉格朗日算子最大 (变为主目标), 再调整 x 使主目标最小。x 主可行时, 主目标等于原凸优化问题的目标函数, 主目标的最小值等于原凸优化问题的最小值。

对偶问题是先调整 x 使拉格朗日算子最小 (变为对偶目标, 因为拉格朗日算子是关于 x 的没有限制的凸函数, 所以变为对偶目标时其对 x 的偏导数为 0), 再调整使对偶目标最大。(α, β) 对偶可行时, 对偶目标小于等于主目标的最小值。

弱对偶性指对偶目标的最大值小于等于主目标的最小值。强对偶性指对偶目标的最大值等于主目标的最小值。

对偶目标的最大值 = 主目标的最小值 = 原凸优化问题的最小值 $\Leftrightarrow^{\text{当且仅当}}$ 满足KKT条件

因此，当对偶问题比原凸优化问题容易求解时，可以通过求解对偶问题来解原凸优化问题。

16.17 Supplementary Material: KKT Conditions, Linear Programming and Nonlinear Programming [4]

16.17.1 Karush-Kuhn-Tucker Theorem(s)

[4] Written by Christopher Griffine. (Linear Programming)

Theorem 16.17.1 Let $z : \mathbb{R}^n \to \mathbb{R}$ be a differentiable objective function, $g_i : \mathbb{R}^n \to \mathbb{R}$ be differentiable constraint functions for $i = 1, \ldots, m$ and $h_j : \mathbb{R}^n \to \mathbb{R}$ be differentiable constraint functions for $j = 1, \ldots, l$. If $\mathbf{x}^* \in \mathbb{R}^n$ is an optimal point satisfying an appropriate regularity condition for the following optimization problem:

$$P \begin{cases} \max & z(x_1, \ldots, x_n) \\ s.t. & g_1(x_1, \ldots, x_n) \leq 0 \\ & \vdots \\ & g_m(x_1, \ldots, x_n) \leq 0 \\ & h_1(x_1, \ldots, x_n) = 0 \\ & \vdots \\ & h_l(x_1, \ldots, x_n) = 0 \end{cases}$$

then there exists $\lambda_1, \ldots, \lambda_m \in \mathbb{R}$ and $\mu_1, \ldots \mu_l \in \mathbb{R}$ so that:

$$\text{Primal Feasibility} : \begin{cases} g_i(\mathbf{x}^*) \leq 0 & \text{for } i = 1, \ldots, m \\ h_j(\mathbf{x}^*) = 0 & \text{for } j = 1, \ldots, l \end{cases}$$

$$\text{Dual Feasibility} : \begin{cases} \nabla z(\mathbf{x}^*) - \sum_{i=1}^{m} \lambda_i \nabla g_i(\mathbf{x}^*) - \sum_{j=1}^{l} \mu_j \nabla h_j(\mathbf{x}^*) = \mathbf{0} \\ \lambda_i \geq 0 \quad \text{for } i = 1, \ldots, m \\ \mu_j \in \mathbb{R} \quad \text{for } j = 1, \ldots, l \end{cases}$$

$$\text{Complementary Slackness} : \left\{ \lambda_i g_i(\mathbf{x}^*) = 0 \quad \text{for } i = 1, \ldots, m \right.$$

Theorem 16.17.2 Let $z : \mathbb{R}^n \to \mathbb{R}$ be a differentiable concave function, $g_i : \mathbb{R}^n \to \mathbb{R}$ be differentiable convex functions for $i = 1, \ldots, m$ and $h_j : \mathbb{R}^n \to \mathbb{R}$ be affine functions for $j = 1, \ldots, l$. Suppose there are $\lambda_1, \ldots, \lambda_m \in \mathbb{R}$ and $\mu_1, \ldots \mu_l \in \mathbb{R}$ so that:

$$\text{Primal Feasibility} : \begin{cases} g_i(\mathbf{x}^*) \leq 0 & \text{for } i = 1, \ldots, m \\ h_j(\mathbf{x}^*) = 0 & \text{for } j = 1, \ldots, l \end{cases}$$

$$\text{Dual Feasibility} : \begin{cases} \nabla z(\mathbf{x}^*) - \sum_{i=1}^{m} \lambda_i \nabla g_i(\mathbf{x}^*) - \sum_{j=1}^{l} \mu_j \nabla h_j(\mathbf{x}^*) = \mathbf{0} \\ \lambda_i \geq 0 \quad \text{for } i = 1, \ldots, m \\ \mu_j \in \mathbb{R} \quad \text{for } j = 1, \ldots, l \end{cases}$$

$$\text{Complementary Slackness} : \left\{ \lambda_i g_i(\mathbf{x}^*) = 0 \quad \text{for } i = 1, \ldots, m \right.$$

then $\mathbf{x}^*$ is a global maximizer for

$$P\begin{cases}\max\ z(x_1,\ldots,x_n)\\ s.t.\ g_1(x_1,\ldots,x_n)\le 0\\ \qquad\vdots\\ \qquad g_m(x_1,\ldots,x_n)\le 0\\ \qquad h_1(x_1,\ldots,x_n)=0\\ \qquad\vdots\\ \qquad h_l(x_1,\ldots,x_n)=0\end{cases}$$

The values $\lambda_1,\ldots,\lambda_m$ and $\mu_1,\ldots,\mu_l$ are sometimes called *Lagrange multipliers* and sometimes called *dual variables*. Primal Feasibility, Dual Feasibility and Complementary Slackness are called the *Karush-Kuhn-Tucker* (KKT) conditions.

The regularity condition mentioned in Theorem 16.17.1 is sometimes called a *constraint qualification*. A common one is that the gradients of the binding constraints are all linearly independent at $\mathbf{x}^*$. There are many variations of constraint qualifications. We will not deal with these in these notes.

Suffice it to say, all the problems we consider will automatically satisfy a constraint qualification, meaning the KKT theorem holds.

Theorem 16.17.1 holds as a necessary condition even if $z(\mathbf{x})$ is not concave or the functions $g_i(\mathbf{x})$ $(i=1,\ldots,m)$ are not convex or the functions $h_j(\mathbf{x})$ $(j=1,\ldots,l)$ are not linear. In this case though, the fact that a triple: $(\mathbf{x},\boldsymbol{\lambda},\boldsymbol{\mu})\in\mathbb{R}^n\times\mathbb{R}^m\times\mathbb{R}^l$ does not ensure that this is an optimal solution for Problem P.

Looking more closely at the dual feasibility conditions, we see something interesting. Suppose that there are *no* equality constraints (i.e., not constraints of the form $h_j(\mathbf{x})=0$). Then the statements:

$$\nabla z(\mathbf{x}^*)-\sum_{i=1}^{m}\lambda_i\nabla g_i(\mathbf{x}^*)-\sum_{j=1}^{l}\mu_j\nabla h_j(\mathbf{x}^*)=\mathbf{0}$$

$$\lambda_i\ge 0\quad\text{for } i=1,\ldots,m$$

imply that:

$$\nabla z(\mathbf{x}^*)=\sum_{i=1}^{m}\lambda_i\nabla g_i(\mathbf{x}^*)$$

$$\lambda_i\ge 0\quad\text{for } i=1,\ldots,m$$

Specifically, this says that the *gradient of z at $\mathbf{x}^*$ is a positive combination of the gradients of the constraints at $\mathbf{x}^*$*. But more importantly, since we also have *complementary slackness*, we know that if $\mathbf{g}_i(\mathbf{x}^*)\neq 0$, then $\lambda_i=0$ because $\lambda_i g_i(\mathbf{x}^*)=0$ for

$i = 1, \ldots, m$. Thus, what dual feasibility is really saying is that *gradient of z at $\mathbf{x}^*$ is a positive combination of the gradients of the **binding** constraints at $\mathbf{x}^*$*. Remember, a constraint is binding if $g_i(\mathbf{x}^*) = 0$, in which case $\lambda_i \geq 0$.

Continuing from the previous remark, in the general case when we have some equality constraints, then dual feasibility says:

$$\begin{aligned} \nabla z(\mathbf{x}^*) &= \sum_{i=1}^{m} \lambda_i \nabla g_i(\mathbf{x}^*) + \sum_{j=1}^{l} \mu_j \nabla h_j(\mathbf{x}^*) \\ \lambda_i &\geq 0 \quad \text{for } i = 1, \ldots, m \\ \mu_j &\in \mathbb{R} \quad \text{for } j = 1, \ldots, l \end{aligned}$$

Since equality constraints are *always binding* this says that the *gradient of z at $\mathbf{x}^*$ is a linear combination of the gradients of the **binding** constraints at $\mathbf{x}^*$*.

16.17.2 Linear Programming and KKT Conditions - An Example

Consider the following linear programming problem:

$$P \begin{cases} \max \; x_1 + x_2 \\ \quad s.t. x_1 + 2x_2 \leq 4 \\ \qquad 2x_1 + x_2 \leq 6 \\ \qquad x_1, x_2 \geq 0 \end{cases} \tag{16.2}$$

Assuming I didn't know how to solve this problem using the Simplex method, I could look at Theorem 16.17.2 and remark that the constraints and objective are all concave (and convex) since they are linear. Therefore, it suffices to find a KKT point and this point must be optimal. We can re-write Problem P in the form of Theorem 16.17.1 as:

$$P \begin{cases} \max \; z(x_1, x_2) \equiv x_1 + x_2 \\ \quad s.t. g_1(x_1, x_2) \equiv x_1 + 2x_2 - 4 \leq 0 \\ \qquad g_2(x_1, x_2) \equiv 2x_1 + x_2 - 6 \leq 0 \\ \qquad g_3(x_1, x_2) \equiv -x_1 \leq 0 \\ \qquad g_4(x_1, x_2) \equiv -x_2 \leq 0 \end{cases}$$

To write the KKT conditions, observe the following:

$$\nabla z = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad \nabla g_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad \nabla g_2 = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \qquad \nabla g_3 = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \qquad \nabla g_4 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

We can now write the KKT conditions for this problem as:

$$\text{Primal Feasibility}: \begin{cases} x_1 + 2x_2 \leq 4 \\ 2x_1 + x_2 \leq 6 \\ x_1, x_2 \geq 0 \end{cases}$$

$$\text{Dual Feasibility}: \begin{cases} \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \lambda_1 \begin{bmatrix} 1 \\ 2 \end{bmatrix} - \lambda_2 \begin{bmatrix} 2 \\ 1 \end{bmatrix} - \lambda_3 \begin{bmatrix} -1 \\ 0 \end{bmatrix} - \lambda_4 \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ \lambda_1, \lambda_2, \lambda_3, \lambda_4 \geq 0 \end{cases}$$

$$\text{Complementary Slackness}: \begin{cases} \lambda_1(x_1 + 2x_2 - 4) = 0 \\ \lambda_2(2x_1 + x_2 - 6) = 0 \\ \lambda_3(-x_1) = 0 \\ \lambda_4(-x_2) = 0 \end{cases}$$

Consider Dual Feasibility for a moment. I can expand the matrices to obtain a system of equations:

$$\begin{aligned} 1 - \lambda_1 - 2\lambda_2 + \lambda_3 &= 0 \\ 1 - 2\lambda_1 - \lambda_2 + \lambda_4 &= 0 \\ \lambda_1, \lambda_2, \lambda_3, \lambda_4 &\geq 0 \end{aligned}$$

Or:

$$\begin{aligned} \lambda_1 + 2\lambda_2 - \lambda_3 &= 1 \\ 2\lambda_1 + \lambda_2 - \lambda_4 &= 1 \\ \lambda_1, \lambda_2, \lambda_3, \lambda_4 &\geq 0 \end{aligned}$$

Since $\lambda_3, \lambda_4 \geq 0$, they act like *surplus variables* and we can write the Dual Feasibility as:

$$\begin{cases} \lambda_1 + 2\lambda_2 \geq 1 \\ 2\lambda_1 + \lambda_2 \geq 1 \\ \lambda_1, \lambda_2 \geq 0 \end{cases} \tag{16.3}$$

It would now suffice to find values for x_1, x_2, λ_1, λ_2, λ_3, and λ_4 the satisfy the KKT conditions and we could solve the linear programming problem P.

We can show that the optimal point for this problem is $x = \frac{8}{3}$ and $y = \frac{2}{3}$ using a graphical method. Figure 16.8 shows the feasible region of the problem as well as the level curves of the objective function (curves for which $x_1 + x_2 = k$, where k is a fixed constant). The value k increases going left to right and bottom to top, so the optimal solution must occur at the intersection of the lines $x_1 + 2x_2 = 4$ and $2x_1 + x_2 = 6$. You can compare the value of the objective at the proposed optimal point $x = \frac{8}{3}$ and $y = \frac{2}{3}$ to the value of the objective at the other extreme points. For example, at the extreme point $x = 3$, $y = 0$, we see the value of the objective is only 3, as compared to $\frac{10}{3}$ at the point of optimality.

We can now see that for the KKT conditions to hold, we must have $\lambda_3 = \lambda_4 = 0$ because $x_1 > 0$ and $x_2 > 0$ at optimality and complementary slackness requires that:

Figure 16.8: The feasible region for the example linear programming problem illustrating level curves of the objective.

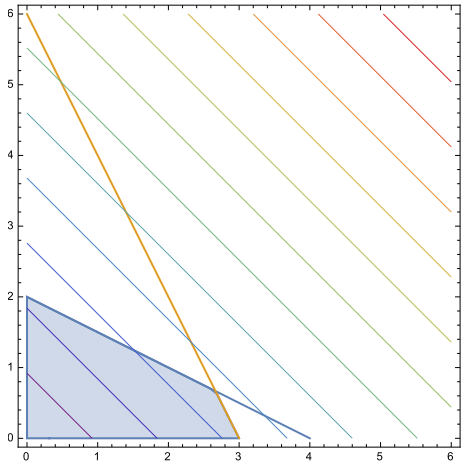
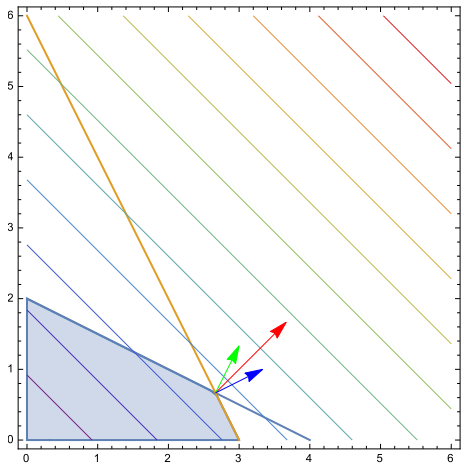


Figure 16.9: The gradients of the objective function and the binding constraints at optimality and their geometric relation are illustrated



$\lambda_3(-x_1) = 0$ and $\lambda_4(-x_2) = 0$ at optimality. This leaves λ_1 and λ_2. We know these must satisfy the equations in Expression 16.3. Therefore, we see that when $\lambda_1 = \lambda_2 = \frac{1}{3}$, Expression 2 is satisfied. Thus: we have written:

$$\begin{bmatrix}1\\1\end{bmatrix} = \frac{1}{3}\begin{bmatrix}1\\2\end{bmatrix} - \frac{1}{3}\begin{bmatrix}2\\1\end{bmatrix} \tag{16.4}$$

That is, we have expressed the gradient of the objective function (∇z) as a positive combination of the gradients of the *binding* constraints (∇g_1 and ∇g_2). This is shown in Figure 16.9, in which we see the gradient of the objective function (red) inside the (cone of) the gradients of the binding constraints (blue and green).

16.17.3 The Dual KKT Conditions

Recall that if we are given a linear programming problem of the form:

$$P\begin{cases}\max & \mathbf{c}^T\mathbf{x}\\ s.t. & \mathbf{A}\mathbf{x} \leq \mathbf{b}\\ & \mathbf{x} \geq 0\end{cases} \tag{16.5}$$

Then the dual problem for Problem P is:

$$D\begin{cases}\min & \mathbf{w}\mathbf{b}\\ s.t. & \mathbf{w}\mathbf{A} \geq \mathbf{c}^T\\ & \mathbf{w} \geq \mathbf{0}\end{cases} \tag{16.6}$$

Here we assume that $\mathbf{w}$ is a *row* vector (unlike our usual assumption that all vectors are column vectors). In our example problem, we have:

$$\mathbf{A} = \begin{bmatrix}1 & 2\\ 2 & 1\end{bmatrix} \qquad \mathbf{c} = \begin{bmatrix}1\\1\end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix}4\\6\end{bmatrix}$$

Let us write:

$$\mathbf{w} = \begin{bmatrix}\lambda_1 & \lambda_2\end{bmatrix}$$

Then we can write the dual problem as:

$$D\begin{cases}\min & \begin{bmatrix}\lambda_1 & \lambda_2\end{bmatrix}\begin{bmatrix}4\\6\end{bmatrix}\\ & \begin{bmatrix}\lambda_1 & \lambda_2\end{bmatrix}\begin{bmatrix}1 & 2\\2 & 1\end{bmatrix} \geq \begin{bmatrix}1 & 1\end{bmatrix}\\ & \lambda_1, \lambda_2 \geq 0\end{cases}$$

This can be rewritten more concretely as:

$$D\begin{cases}\min & 4\lambda_1 + 6\lambda_2\\ & \lambda_1 + 2\lambda_2 \geq 1\\ & 2\lambda_1 + \lambda_2 \geq 1\\ & \lambda_1, \lambda_2 \geq 0\end{cases}$$

Immediately we see a connection. The constraints of this problem look just like the simplified dual feasibility constraints. In fact, if we add surplus variables $\lambda_3, \lambda_4 \geq 0$, we see they match exactly and the problem in standard form is:

$$D \begin{cases} \min & 4\lambda_1 + 6\lambda_2 \\ & \lambda_1 + 2\lambda_2 - \lambda_3 = 1 \\ & 2\lambda_1 + \lambda_2 - \lambda_4 = 1 \\ & \lambda_1, \lambda_2, \lambda_3, \lambda_4 \geq 0 \end{cases}$$

We can complete our exploration of the relationship between these two problems by constructing the KKT conditions for the dual problem. First, re-write the dual as:

$$\begin{aligned} \max \quad & -4\lambda_1 - 6\lambda_2 \\ s.t. \; & -\lambda_1 - 2\lambda_2 + 1 \leq 0 \\ & -2\lambda_1 - \lambda_2 + 1 \leq 0 \\ & -\lambda_1 \leq 0 \\ & -\lambda_2 \leq 0 \end{aligned}$$

As before, we can write down the KKT conditions:

$$\text{Primal Feasibility} : \begin{cases} \lambda_1 + 2\lambda_2 \geq 1 \\ 2\lambda_1 + \lambda_2 \geq 1 \\ \lambda_1, \lambda_2 \geq 1 \end{cases}$$

$$\text{Dual Feasibility} : \begin{cases} \begin{bmatrix} -4 \\ -6 \end{bmatrix} - x_1 \begin{bmatrix} -1 \\ -2 \end{bmatrix} - x_2 \begin{bmatrix} -2 \\ -1 \end{bmatrix} - s_1 \begin{bmatrix} -1 \\ 0 \end{bmatrix} - s_2 \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ x_1, x_2, s_1, s_2 \geq 0 \end{cases}$$

$$\text{Complementary Slackness} : \begin{cases} x_1(-\lambda_1 - 2\lambda_2 + 1) = 0 \\ x_2(-2\lambda_1 - \lambda_2 + 1) = 0 \\ s_1(-\lambda_1) = 0 \\ s_2(-\lambda_2) = 0 \end{cases}$$

Again, we can re-write the dual feasibility conditions (for which we have intentionally chosen specific dual variable names) and see they become:

$$\begin{aligned} x_1 + 2x_2 + s_1 &= 4 \\ 2x_1 + x_2 + s_2 &= 6 \\ x_1, x_2, s_1, s_2 &\geq 0 \end{aligned}$$

Since $s_1, s_2 \geq 0$, they act like *slack variables* and thus we have:

$$\begin{aligned} x_1 + 2x_2 &\leq 4 \\ 2x_1 + x_2 &\leq 6 \\ x_1, x_2 &\geq 0 \end{aligned}$$

and we've recovered the constraints from the original primal problem as the dual feasibility conditions of the KKT conditions for the dual optimization problem. We can go further. Note:

$$\begin{aligned} -s_1 &= x_1 + 2x_2 - 4 = -g_1(x_1, x_2) \\ -s_2 &= 2x_1 + x_2 - 6 = -g_2(x_1, x_2) \end{aligned}$$

while from our earlier observations about the surplus variables λ_3 and λ_4 we know:

$$\begin{aligned} \lambda_3 &= -\lambda_1 - 2\lambda_2 + 1 \\ -\lambda_4 &= -2\lambda_1 - \lambda_2 + 1 \end{aligned}$$

Thus, complementary slackness from the primal problem is:

$$\begin{aligned} \lambda_3(-x_1) = 0 &\iff \lambda_3(x_1) = 0 \iff x_1(\lambda_1 + 2\lambda_2 - 1) = 0 \\ \lambda_4(-x_2) = 0 &\iff \lambda_4(x_2) = 0 \iff x_2(-2\lambda_1 - \lambda_2 + 1) = 0 \end{aligned}$$

Likewise:

$$\begin{aligned} \lambda_1(x_1 + 2x_2 - 4) = 0 &\iff \lambda_1(-s_1) = 0 \iff s_1(-\lambda_1) = 0 \\ \lambda_2(2x_1 + x_2 - 6) = 0 &\iff \lambda_2(-s_2) = 0 \iff s_2(-\lambda_2) = 0 \end{aligned}$$

Thus, the complementary slackness conditions of the primal problem are identical to the complementary slackness conditions of the dual problem. This fact is true for linear programming problems in general.

For an arbitrary linear programming problem and its dual, the KKT conditions for the primal and dual problems are **equivalent**, but the dual feasibility conditions for the primal problem are identical to the primal feasibility conditions for the dual problem and vice-versa. *Thus, two linear programming problems are dual to each other if and only if they share KKT conditions with the primal and dual feasibility conditions swapped.*

We will use these results on KKT conditions and linear programs to discuss the solution (i.e., Nash equilibria) of zero-sum games via linear programming. We will then generalize these results to derive a quadratic programming problem whose optimal solutions will yield Nash equilibria.

16.17.4 An Example from Nonlinear Programming

Let's recall a simple optimization problem from differential calculus (Math 140): Goats are an environmentally friendly and inexpensive way to control a lawn when there are lots of rocks or lots of hills. (Seriously, both Google and some U.S. Navy bases use goats on rocky hills instead of paying lawn mowers!)

Suppose I wish to build a pen to keep some goats. I have 100 meters of fencing and I wish to build the pen in a rectangle with the largest possible area. How long should the sides of the rectangle be? In this case, making the pen *better* means making it have the largest possible area.

The problem is illustrated in Figure 16.10. Clearly, we know that:

$$2x + 2y = 100 \tag{16.7}$$

because $2x + 2y$ is the perimeter of the pen and I have 100 meters of fencing to build my pen. The area of the pen is $A(x, y) = xy$. We can use Equation 16.7 to solve for x in terms of y. Thus we have:

$$y = 50 - x \tag{16.8}$$

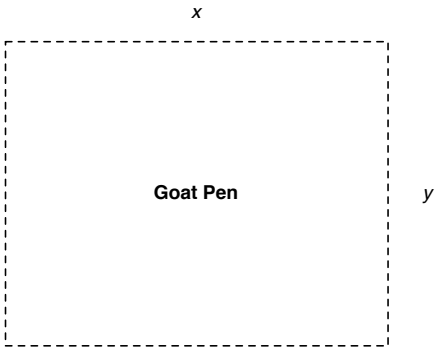


Figure 16.10: Goat pen with unknown side lengths. The objective is to identify the values of x and y that maximize the area of the pen (and thus the number of goats that can be kept).

and $A(x) = x(50 - x)$. To maximize $A(x)$, recall we take the first derivative of $A(x)$ with respect to x, set this derivative to zero and solve for x:

$$\frac{dA}{dx} = 50 - 2x = 0; \tag{16.9}$$

Thus, $x = 25$ and $y = 50 - x = 25$. We further recall from basic calculus how to confirm that this is a maximum; note:

$$\left.\frac{d^2A}{dx^2}\right|_{x=25} = -2 < 0 \tag{16.10}$$

Which implies that $x = 25$ is a *local maximum* for this function. Another way of seeing this is to note that $A(x) = 50x - x^2$ is an concave (a frowning parabola). As we could have guessed, a square will maximize the area available for holding goats.

We can re-write the problem to be in a more common form

$$\begin{cases} \max & A(x,y) = xy \\ s.t. & 2x + 2y = 100 \equiv 2x + 2y - 100 = 0 \\ & x \geq 0 \equiv -x \leq 0 \\ & y \geq 0 \equiv -y \leq 0 \end{cases} \tag{16.11}$$

Note we've added two inequality constraints $x \geq 0$ and $y \geq 0$ because it doesn't really make any sense to have negative lengths. In our problem, we now have: $g_1(x,y) = -x$ and $g_2(x,y) = -y$ and $h(x,y) = 2x + 2y - 100$ to be consistent with the notation in Theorem 16.17.1. It is worth noting, we could have used the constraint $2x+2y-100 \leq 0$ instead and obtained the same answer.

For the point of optimality ($x = 25, y = 25$), let us now compute the KKT conditions. Note first:

$$\nabla A = \begin{bmatrix} y \\ x \end{bmatrix} \qquad \nabla h = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \qquad \nabla g_1 = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \qquad \nabla g_2 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

Substituting $x^* = y^* = 25$, we can see that primal feasibility is obviously satisfied. Moreover:

$$\nabla A(x^*, y^*) = \begin{bmatrix} 25 \\ 25 \end{bmatrix}$$

We consider only dual feasibility and complimentary slackness.

$$\text{Dual Feasibility}: \begin{cases} \begin{bmatrix} 25 \\ 25 \end{bmatrix} - \lambda_1 \begin{bmatrix} -1 \\ 0 \end{bmatrix} - \lambda_2 \begin{bmatrix} 0 \\ -1 \end{bmatrix} - \mu \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ \lambda_1, \lambda_2 \geq 0 \\ \mu \in \mathbb{R} \end{cases}$$

$$\text{Complementary Slackness}: \begin{cases} \lambda_1(-x^*) = \lambda_1 \cdot (-25) = 0 \\ \lambda_2(-y^*) = \lambda_2 \cdot (-25) = 0 \end{cases}$$

Notice we substituted x^* and y^* into the complementary slackness equations. From complementary slackness, we see at once that $\lambda_1 = \lambda_2 = 0$. This means dual feasibility reduces to:

$$\begin{bmatrix} 25 \\ 25 \end{bmatrix} - \mu \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

The solution: $\mu = 25/2$ satisfies this equality. Notice the gradient of the objective and the gradient of the (single) binding constraint are parallel. Notice also unlike a linear programming problem, the optimal solution to this problem *does not* occur at the extreme point of the constraint set (which is really just the line segment $2x + 2y = 100$ with $x, y \geq 0$). This geometric interpretation is illustrated in Figure 16.11. The gradient of the objective is shown in red, the gradient of the binding constraint is shown in green. Note, the gradient of the binding constraint has been scaled for visual effect. As expected, the level curves shown are given by the (implicit) equation $xy = k$. As you go up and right, the value of k increases.

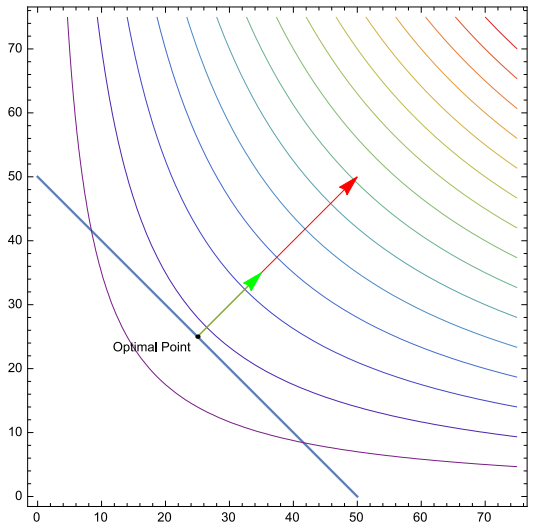


Figure 16.11: Visualization of the KKT conditions for a simple nonlinear programming problem. Notice the point of optimality is not at an extreme point. Moreover, the gradient of the binding constraint is parallel to the gradient of the objective, as expected. *Note, the gradient of the binding constraint has been scaled larger for visual effect.*

16.18 Supplementary Material: KKT Conditions [5]

16.18.1 Dual Problem

Given the following convex minimization problem:

Problem 16.22 — convex minimization problem.

$$\begin{array}{ll} \min_x & f(x) \\ \text{subject to} & h_i(x) \le 0, \quad i = 1, \dots, m \\ & \ell_j(x) = 0, \quad j = 1, \dots, r \end{array}$$

The Lagrangian is defined as $L(x, u, v) = f(x) + \sum_{u=1}^{m} u_i h_i(x) + \sum_{j=1}^{r} v_j \ell_j(x)$. The Lagrange dual function is defined as $g(u, v) = \min_x L(x, u, v)$. The dual problem is

$$\begin{array}{ll} \max_{u,v} & g(u, v) \\ \text{subject to} & u \ge 0 \end{array}$$

▪

The Lagrange dual function $g(u, v)$ is always *concave* regardless of whether the primal problem is convex or not.

Weak duality: $f^* \ge g^*$ holds for all problems, where f^* and g^* are primal and dual optimal values, respectively.

Slaters's condition, which says the primal has at least one strictly feasible point, is a sufficient condition for *strong duality* to hold. If $\exists x$ such that $h_i(x) < 0, i = 1, \dots, m$ and $\ell_j(x) = 0, j = 1, \dots, r, f^* = g^*$. This condition can be further refined to $h_i(x) < 0$ for all i such that h_i is nonaffine. As a result, Slater's condition is reduced to feasibility for LP's.

16.19 Karush-Kuhn-Tucker (KKT) Conditions

For the given problem 16.22, the KKT conditions are:

1. $0 \in \partial_x \left(f(x) + \sum_{i=1}^{m} u_i h_i(x) + \sum_{j=1}^{r} v_j \ell_j(x) \right)$ (stationary)
2. $u_i \cdot h_i(x) = 0$ for all i (complementary slackness)
3. $h_i(x) \le 0, \ell_j(x) = 0$ for all i, j (primal feasibility)
4. $u_i \ge 0$ for all i (dual feasibility)

Theorem 16.19.1 For x^* and u^*, v^* to be primal and dual solutions, KKT conditions are sufficient.

Proof. Proof:

Sufficiency: if $\exists x^*$ and u^*, v^* that satisfy the KKT conditions, $g(u^*, v^*) = f(x^*) + \sum_{i=1}^{m} u_i^* h_i(x^*) + \sum_{j=1}^{r} v_j^* \ell_j(x^*) = f(x^*)$

The first equality holds from stationarity, since $f(x) + \sum_{i=1}^{m} u_i h_i(x) + \sum_{j=1}^{r} v_j \ell_j(x)$ is convex, so any stationary point is a minimizer, and the second holds by complementary slackness.

By weak duality, x^* and u^*, v^* are optimal. It always implies that the duality gap is 0 . ■

Theorem 16.19.2 For a problem with strong duality (e.g. assume Slater's condition: convex problem and there exists x strictly satisfying nonaffine inequality constraints),

[5] These notes are taken from Machine Learning 10-725 (Convex Optimization).

x^* and u^*, v^* are primal and dual solutions $\iff$ x^* and u^*, v^* satisfy the KKT conditions

Proof. Sufficiency: Follows from Theorem 16.19.1.

Necessity: Let x^* and u^*, v^* be primal and dual solutions, and suppose we know strong duality holds.Then

$$\begin{aligned} f(x^*) &= g(u^*, v^*) \\ &= \min_x \left(f(x) + \sum_{i=1}^{m} u_i^* h_i(x) + \sum_{j=1}^{r} v_j^* \ell_j(x) \right) \\ &\leq f(x^*) + \sum_{i=1}^{m} u_i^* h_i(x^*) + \sum_{j=1}^{r} v_j^* \ell_j(x^*) \\ &\leq f(x^*) \end{aligned}$$

The LSH equals RHS, so all inequalities in the equation must be equalities. Looking at the KKT conditions one by one, primal and dual feasibility holds, by virtue of optimality. Stationarity comes from the fact that x^* minimizes $f(x) + \sum_{i=1}^{m} u_i^* h_i(x) + \sum_{j=1}^{r} v_j^* \ell_j(x)$. Since x^* is the minimizer, it must be a stationary point for this function. Complementary slackness comes from $f(x^*) + \sum_{i=1}^{m} u_i^* h_i(x^*) + \sum_{j=1}^{r} v_j^* \ell_j(x^*) = f(x^*)$, since we must have $\sum_{i=1}^{m} u_i^* h_i(x^*) = 0$ and they are each non-negative.

■

17. 非线性最小二乘法

17.1 非线性最小二乘法的定义

Problem 17.1 $f_1(x), \ldots, f_m(x)$ 是可微函数；
优化目标:

$$\min_x \sum_{i=1}^{m} (f_i(x))^2$$

■

Problem 17.2 设函数 $f(x) : \mathbb{R}^n \to \mathbb{R}^m$ ，其第 i 个分量为函数 $f_i(x)$
则有

$$f(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_m(x) \end{bmatrix}$$

优化目标

$$\min_x \sum_{i=1}^{m} (f_i(x))^2 = \|f(x)\|_2^2$$

■

如果 $f(x) = Ax - b$ ，问题简化为 (线性) 最小二乘问题。

17.1.1 例子：距离测量定位

Problem 17.3 向量 x_{ex} 表示二维或三维中的未知位置，通过测量到的已知点 $a_1, \cdots, a_m$ 的距离来估计 x_{ex}

$$\rho_i = \|x_{ex} - a_i\|_2 + v_i, \quad i = 1, \cdots, m$$

其中 v_i 是测量误差。 ■

非线性最小二乘法估计：通过最小化估计 $\hat{x}$ 的位置

$$\min_x \sum_{i=1}^{m} (\|x - a_i\|_2 - \rho_i)^2 = \|f(x)\|_2^2$$

函数 $f_i(x) = \|x - a_i\|_2 - \rho_i$ 是 $f(x)$ 的 i 个分量。

17.1.2 例子：多个相机视图定位

这个例子与遥感卫星成像等有关。

建立一个理想的相机模型，由参数 $A \in \mathbb{R}^{2\times3}, b \in \mathbb{R}^2, c \in \mathbb{R}^3, d \in \mathbb{R}$ 来描述。相机及其位置和方向用 A，b，c，d 来刻画。

目标位置 $x \in \mathbb{R}^3$ 在二维平面图像投影位置 $x' \in \mathbb{R}^2$ 。

$$x' = \frac{1}{c^T x + d}(Ax + b)$$

如果物体在摄像机前面，则 $c^T x + d > 0$ 。

Problem 17.4 位于 x_{ex} 位置的物体由 l 个相机观察 (由 A_i, b_i, c_i, d_i 描述)，目标在相机图像平面上位置 $y_i \in \mathbb{R}^2$

$$y_i = \frac{1}{c_i^T x_{ex} + d_i}(A_i x_{ax} + b_i) + v_i$$

v_i 为测量误差或量化误差。目的是从 l 个观测点 $y_1, \ldots, y_l$ 来估计三维位置 x_{ex} 。 ■

使用非线性最小二乘法估计

$$\min_x \sum_{i=1}^{l} \left\| \frac{1}{c_i^T x + d_i}(A_i x + b_i) - y_i \right\|_2^2$$

这是关于 $m = 2l$ 的非线性最小二乘法问题，$(y_i)_j$ 是 y_i 的第 j 个分量:

$$f_i(x) = \frac{(A_i x + b_i)_1}{c_i^T x + d_i} - (y_i)_1, \quad f_{l+i}(x) = \frac{(A_i x + b_i)_2}{c_i^T x + d_i} - (y_i)_2$$

17.1.3 例子：模型拟合

Problem 17.5

$$\min_\theta \sum_{i=1}^{N} \left(\hat{f}\left(x^{(i)}, \theta\right) - y^{(i)} \right)^2$$

$\left(x^{(i)}, y^{(i)}\right), i = 1, \cdots, N$ 表示样本。设函数 $\hat{f}(x, \theta)$ 的参数 $\theta = (\theta_1, \ldots, \theta_p)$。最小化的目标是估计函数参数 θ。 ■

假设函数 $\hat{f}(x, \theta)$ 关于 θ 的线性函数

$$\hat{f}(x, \theta) = \theta_1 f_1(x) + \cdots + \theta_p f_p(x)$$

当然 $\hat{f}(x, \theta)$ 也可以是关于 θ 的非线性函数。

■ **Example 17.1** 有四个变量 $\theta_1, \theta_2, \theta_3, \theta_4$ 的非线性最小二乘问题:

$$\min_\theta \sum_{i=1}^{N} \left(\theta_1 e^{\theta_2 x(i)} \cos\left(\theta_3 x^{(i)} + \theta_4\right) - y^{(i)} \right)^2$$

■

17.1.4 例子：正交距离回归

正交距离回归目标: 最小化 $\hat{f}(x,\theta)$ 图中数据点到曲线的均方距离。

例子: 三次多项式的正交距离回归:

$$\hat{f}(x,\theta) = \theta_1 + \theta_2 x + \theta_3 x^2 + \theta_4 x^3$$

17.1.5 非线性最小二乘法

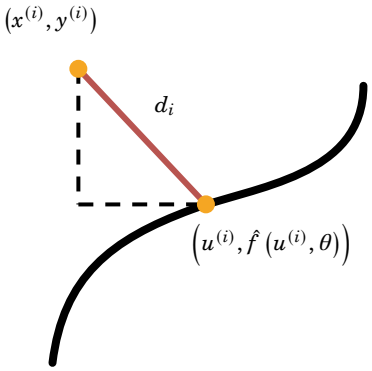
Problem 17.6

$$\min_{\theta, u^{(i)}} \sum_{i=1}^{N} \left(\left(\hat{f}\left(u^{(i)}, \theta\right) - y^{(i)} \right)^2 + \left\| u^{(i)} - x^{(i)} \right\|_2^2 \right)$$

这个模型需要优化参数 θ 和 N 个点 $u^{(i)}$。

第 i 项为数据点 $\left(x^{(i)}, y^{(i)}\right)$ 到点 $\left(u^{(i)}, \hat{f}\left(u^{(i)}, \theta\right)\right)$ 距离的平方。 ∎

Figure 17.1: 优化模型的几何意义



$$d_i^2 = \left(\hat{f}\left(u^{(i)}, \theta\right) - y^{(i)} \right)^2 + \left\| u^{(i)} - x^{(i)} \right\|_2^2$$

通过 $u^{(i)}$ 与图中点 $\left(x^{(i)}, y^{(i)}\right)$ 的平方距离来最小化 d_i^2。最小化 $\sum_i d_i^2$ ，即通过 $u^{(1)}, \ldots, u^{(N)}$ 和 θ 来最小化均方距离。

17.1.6 二分类

二分类的目标函数为 $\hat{f}(x,\theta) = \operatorname{sign}\left(\theta_1 f_1(x) + \theta_2 f_2(x) + \cdots + \theta_p f_p(x)\right)$

Problem 17.7 — 二分类问题.

$$\min_{\theta} \sum_{i=1}^{N} \left(\phi\left(\theta_1 f_1\left(x^{(i)}\right) + \cdots + \theta_p f_p\left(x^{(i)}\right)\right) - y^{(i)} \right)^2$$

$\left(x^{(i)}, y^{(i)}\right)$ 是数据点，$y^{(i)} \in \{-1, 1\}$ 。$\phi(u)$ 是 sigmoid 函数，它是 $\operatorname{sign}(u)$ 函数的一个可微的近似函数。

$$\phi(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}}$$

通过计算 θ 来求解非线性最小二乘问题，**大概率**能得到良好的结果。 ∎

17.2 梯度

17.2.1 Gradient and Directional Derivative

Definition 17.2.1 — $\mathbb{R}^n \to \mathbb{R}$ **函数 g 的梯度.** 可微函数 $g : \mathbb{R}^n \to \mathbb{R}$ 在 $z \in \mathbb{R}^n$ 的梯度 (gradient) 为

$$\nabla g(z) = \begin{bmatrix} \frac{\partial g}{\partial x_1}(z) \\ \frac{\partial g}{\partial x_2}(z) \\ \cdots \\ \frac{\partial g}{\partial x_n}(z) \end{bmatrix}$$

Definition 17.2.2 — **g 在 z 附近的仿射近似（一阶泰勒公式，线性化）.**

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

Definition 17.2.3 — **Directional Derivative.** For given z and nonzero v, define $h(t) = g(z + tv)$

The derivative of h at $t = 0$

$$\begin{aligned} h'(0) &= \frac{\partial g}{\partial x_1}(z)v_1 + \frac{\partial g}{\partial x_2}(z)v_2 + \cdots + \frac{\partial g}{\partial x_n}(z)v_n \\ &= \nabla g(z)^T v \end{aligned}$$

This is called the *directional derivative* of g (at z, in the direction v), v is a *descent direction* of g at z if $\nabla g(z)^T v < 0$.

17.2.2 Jacobian Matrices

Definition 17.2.4 — **Jacobian Matrices.** 可微函数 $f : \mathbb{R}^n \to \mathbb{R}^m$ 在 $z \in \mathbb{R}^n$ 的导数矩阵 (Jacobian 矩阵)

$$f(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_m(x) \end{bmatrix} \quad Df(z) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(z) & \frac{\partial f_1}{\partial x_2}(z) & \cdots & \frac{\partial f_1}{\partial x_n}(z) \\ \frac{\partial f_2}{\partial x_1}(z) & \frac{\partial f_2}{\partial x_2}(z) & \cdots & \frac{\partial f_2}{\partial x_n}(z) \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1}(z) & \frac{\partial f_m}{\partial x_2}(z) & \cdots & \frac{\partial f_m}{\partial x_n}(z) \end{bmatrix}$$

Definition 17.2.5 — **f 在 z 附近的仿射近似 (线性化).**

$$\begin{aligned} \hat{f}(x) &= f(z) + Df(z)(x - z) \\ &= \begin{bmatrix} f_1(z) \\ f_2(z) \\ \vdots \\ f_m(z) \end{bmatrix} + \left(\begin{bmatrix} \frac{\partial f_1}{\partial x_1}(z) & \frac{\partial f_1}{\partial x_2}(z) & \cdots & \frac{\partial f_1}{\partial x_n}(z) \\ \frac{\partial f_2}{\partial x_1}(z) & \frac{\partial f_2}{\partial x_2}(z) & \cdots & \frac{\partial f_2}{\partial x_n}(z) \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1}(z) & \frac{\partial f_m}{\partial x_2}(z) & \cdots & \frac{\partial f_m}{\partial x_n}(z) \end{bmatrix} \begin{bmatrix} x_1 - z_1 \\ x_2 - z_2 \\ \vdots \\ x_n - z_n \end{bmatrix} \right)_{m \times 1} \end{aligned}$$

用符号 $\hat{f}(x; z)$ 表示点 z 附近的线性化逼近。

17.2.3 Hessian

Definition 17.2.6 — Hessian Matrices. Hessian of g at z is a symmetric $n \times n$ matrix $\nabla^2 g(z)$ with elements

$$\nabla^2 g(z)_{ij} = \frac{\partial^2 g}{\partial x_i \partial x_j}(z)$$

This is also the derivative matrix $Df(z)$ of $f(x) = \nabla g(x)$ at z.

Definition 17.2.7 — Quadratic (second order) approximation of g around z.

$$g_q(x) = g(z) + \nabla g(z)^T (x - z) + \frac{1}{2}(x - z)^T \nabla^2 g(z)(x - z)$$

■ **Example 17.2** Affine function: $g(x) = a^T x + b$

$$\nabla g(x) = a, \quad \nabla^2 g(x) = 0$$

Quadratic function: $g(x) = x^T P x + q^T x + r$ with P symmetric

$$\nabla g(x) = 2Px + q, \quad \nabla^2 g(x) = 2P$$

Least squares cost: $g(x) = \|Ax - b\|^2 = x^T A^T A x - 2b^T A x + b^T b$

$$\nabla g(x) = 2A^T A x - 2A^T b, \quad \nabla^2 g(x) = 2A^T A$$

■

Theorem 17.2.1 — Linear combination properties. If $g(x) = \alpha_1 g_1(x) + \alpha_2 g_2(x)$, then

$$\nabla g(x) = \alpha_1 \nabla g_1(x) + \alpha_2 \nabla g_2(x)$$
$$\nabla^2 g(x) = \alpha_1 \nabla^2 g_1(x) + \alpha_2 \nabla^2 g_2(x)$$

Theorem 17.2.2 — Composition with affine mapping properties. if $g(x) = h(Cx + d)$, then

$$\nabla g(x) = C^T \nabla h(Cx + d)$$
$$\nabla^2 g(x) = C^T \nabla^2 h(Cx + d) C$$

■ **Example 17.3**

$$g(x_1, x_2) = e^{x_1 + x_2 - 1} + e^{x_1 - x_2 - 1} + e^{-x_1 - 1}$$

Gradient

$$\nabla g(x) = \begin{bmatrix} e^{x_1+x_2-1} + e^{x_1-x_2-1} - e^{-x_1-1} \\ e^{x_1+x_2-1} - e^{x_1-x_2-1} \end{bmatrix}$$

Hessian

$$\nabla^2 g(x) = \begin{bmatrix} e^{x_1+x_2-1} + e^{x_1-x_2-1} + e^{-x_1-1} & e^{x_1+x_2-1} - e^{x_1-x_2-1} \\ e^{x_1+x_2-1} - e^{x_1-x_2-1} & e^{x_1+x_2-1} + e^{x_1-x_2-1} \end{bmatrix}$$

Gradient and Hessian via composition property:
express g as $g(x) = h(Cx + d)$ with $h(y_1, y_2, y_3) = e^{y_1} + e^{y_2} + e^{y_3}$ and

$$C = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 0 \end{bmatrix}, \quad d = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

Gradient: $\nabla g(x) = C^T \nabla h(Cx+d)$

$$\nabla g(x) = \begin{bmatrix} 1 & 1 & -1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} e^{x_1+x_2-1} \\ e^{x_1-x_2-1} \\ e^{-x_1-1} \end{bmatrix}$$

Hessian: $\nabla^2 g(x) = C^T \nabla h^2(Cx+d)C$

$$\nabla^2 g(x) = \begin{bmatrix} 1 & 1 & -1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} e^{x_1+x_2-1} & 0 & 0 \\ 0 & e^{x_1-x_2-1} & 0 \\ 0 & 0 & e^{-x_1-1} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 0 \end{bmatrix}$$

■

17.2.4 Optimality conditions for twice differentiable g

Theorem 17.2.3 — Necessary condition for twice differentiable g. If $x^\star$ is locally optimal, then $\nabla g\,(x^\star) = 0$ and $\nabla^2 g\,(x^\star)$ is positive semidefinite

Theorem 17.2.4 — Sufficient condition. If $x^\star$ satisfies $\nabla g\,(x^\star) = 0$ and $\nabla^2 g\,(x^\star)$ is positive definite then $x^\star$ is locally optimal.

Definition 17.2.8 — Convex functions. g is called *convex* if $\nabla^2 g(x)$ is positive semidefinite everywhere.

Theorem 17.2.5 — Necessary and sufficient condition for convex functions. if g is convex then $x^\star$ is optimal if and only if $\nabla g\,(x^\star) = 0$

■ **Example 17.4** $g(x) = \log\left(e^x + e^{-x}\right)$

$$g'(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \quad g''(x) = \frac{4}{(e^x + e^{-x})^2}$$

$g''(x) \geq 0$ everywhere; $x^\star = 0$ is the unique optimal point ■

■ **Example 17.5** - $g(x) = x^4$

$$g'(x) = 4x^3, \quad g''(x) = 12x^2$$

$g''(x) \geq 0$ everywhere; $x^\star = 0$ is the unique optimal point ■

■ **Example 17.6** - $g(x) = x^3$

$$g'(x) = 3x^2, \quad g''(x) = 6x$$

$g'(0) = 0, g''(0) = 0$ but $x = 0$ is not locally optimal ■

■ **Example 17.7** - $g(x) = x^T P x + q^T x + r$ (P is symmetric positive definite)

$$\nabla g(x) = 2Px + q, \quad \nabla^2 g(x) = 2P$$

$\nabla^2 g(x)$ is positive definite everywhere, hence the unique optimal point is

$$x^\star = -(1/2)P^{-1}q$$

■

■ **Example 17.8** - $g(x) = \|Ax - b\|^2$ (A is a matrix with linearly independent columns)

$$\nabla g(x) = 2A^T Ax - 2A^T b, \quad \nabla^2 g(x) = 2A^T A$$

$\nabla^2 g(x)$ is positive definite everywhere, hence the unique optimal point is

$$x^\star = \left(A^T A\right)^{-1} A^T b$$

■

■ **Example 17.9**

$$g\left(x_1, x_2\right) = e^{x_1+x_2-1} + e^{x_1-x_2-1} + e^{-x_1-1}$$

Gradient

$$\nabla g(x) = \begin{bmatrix} e^{x_1+x_2-1} + e^{x_1-x_2-1} - e^{-x_1-1} \\ e^{x_1+x_2-1} - e^{x_1-x_2-1} \end{bmatrix}$$

Hessian

$$\nabla^2 g(x) = \begin{bmatrix} e^{x_1+x_2-1} + e^{x_1-x_2-1} + e^{-x_1-1} & e^{x_1+x_2-1} - e^{x_1-x_2-1} \\ e^{x_1+x_2-1} - e^{x_1-x_2-1} & e^{x_1+x_2-1} + e^{x_1-x_2-1} \end{bmatrix}$$

We can express $\nabla^2 g(x)$ as

$$\nabla^2 g(x) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} e^{x_1+x_2-1} & 0 & 0 \\ 0 & e^{x_1-x_2-1} & 0 \\ 0 & 0 & e^{-x_1-1} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 0 \end{bmatrix}$$

this shows that $\nabla^2 g(x)$ is positive definite for all x therefore $x^\star$ is optimal if and only if

$$\nabla g\left(x^\star\right) = \begin{bmatrix} e^{x_1^\star+x_2^\star-1} + e^{x_1^\star-x_2^\star-1} - e^{-x_1^\star-1} \\ e^{x_1^\star+x_2^\star-1} - e^{x_1^\star-x_2^\star-1} \end{bmatrix} = 0$$

two nonlinear equations in two variables ■

17.3 求解非线性最小二乘法：目标梯度

对于优化问题17.2

$$g(x) = \|f(x)\|_2^2 = \sum_{i=1}^{m} \left(f_i(x)\right)^2, f(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_m(x) \end{bmatrix}$$

g 对 x_j 的一阶导数为

$$\begin{aligned} \frac{\partial g}{\partial x_j}(z) &= 2\sum_{i=1}^{m} f_i(z) \frac{\partial f_i}{\partial x_j}(z) \\ &= 2\left[\frac{\partial f_1}{\partial x_j}(z), \frac{\partial f_2}{\partial x_j}(z), \cdots, \frac{\partial f_m}{\partial x_j}(z)\right] \begin{bmatrix} f_1(z) \\ f_2(z) \\ \vdots \\ f_m(z) \end{bmatrix} \end{aligned}$$

注意到

$$(Df(z))^T = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(z) & \frac{\partial f_2}{\partial x_1}(z) & \cdots & \frac{\partial f_m}{\partial x_1}(z) \\ \frac{\partial f_1}{\partial x_2}(z) & \frac{\partial f_2}{\partial x_2}(z) & \cdots & \frac{\partial f_m}{\partial x_2}(z) \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_1}{\partial x_n}(z) & \frac{\partial f_2}{\partial x_n}(z) & \cdots & \frac{\partial f_m}{\partial x_n}(z) \end{bmatrix} = [\nabla f_1(z), \nabla f_2(z), \cdots, \nabla f_m(z)]$$

g 在 z 的梯度为

$$\nabla g(z) = \begin{bmatrix} \frac{\partial g}{\partial x_1}(z) \\ \vdots \\ \frac{\partial g}{\partial x_n}(z) \end{bmatrix} = 2\sum_{i=1}^{m} f_i(z)\nabla f_i(z) = 2(Df(z))^T f(z)$$

Theorem 17.3.1 — 非线性最小二乘问题的最优必要条件.

$$\min_x g(x) = \sum_{i=1}^{m} f_i(x)^2 \quad f(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_m(x) \end{bmatrix}$$

如果要 x 使 $g(x)$ 最小，则必须满足:

$$\nabla g(x) = 2Df(x)^T f(x) = 0$$

Corollary 17.3.2 — 正规方程的最优必要条件. 推广到正规方程，如果 $f(x) = Ax - b$ ，那么 $Df(x) = A$ 和

$$\nabla g(x) = 2A^T(Ax - b)$$

对于一般函数 $f(x), \nabla g(x) = 0$ **不是最优解的充分条件**。

17.4 Gauss-Newton Algorithm

Problem 17.8

$$\min_x g(x) = \|f(x)\|_2^2 = \sum_{i=1}^{m} f_i(x)^2$$

从某个初始值 $x^{(1)}$ 开始，当迭代 $x^{(2)}, x^{(3)}, \ldots$.
函数 f 在 $x^{(k)}$ 附近的泰勒展开式是 ∎

$$\hat{f}\left(x; x^{(k)}\right) = f\left(x^{(k)}\right) + Df\left(x^{(k)}\right)\left(x - x^{(k)}\right)$$

用仿射近似 $\hat{f}\left(x, x^{(k)}\right)$ 代替最小二乘法问题中函数 $f(x)$

$$x^{(k+1)} = \arg\min_x \left\| \hat{f}\left(x; x^{(k)}\right) \right\|_2^2$$

第 k 次迭代问题:

$$x^{(k+1)} = \arg\min_x \left\| \hat{f}\left(x; x^{(k)}\right) \right\|_2^2$$
$$x^{(k+1)} = \arg\min_x h(x) = \left\| f\left(x^{(k)}\right) + Df\left(x^{(k)}\right)\left(x - x^{(k)}\right) \right\|_2^2$$

函数 $h(x)$ 的梯度:

$$\nabla h(x) = 2Df\left(x^{(k)}\right)^T \left(f\left(x^{(k)}\right) + Df\left(x^{(k)}\right)\left(x - x^{(k)}\right)\right) = 0$$

则有

$$Df\left(x^{(k)}\right)^T Df\left(x^{(k)}\right) x^{(k)} - Df\left(x^{(k)}\right)^T f\left(x^{(k)}\right) = Df\left(x^{(k)}\right)^T Df\left(x^{(k)}\right) x$$

$$Df\left(x^{(k)}\right)^T Df\left(x^{(k)}\right) x^{(k)} - Df\left(x^{(k)}\right)^T f\left(x^{(k)}\right) = Df\left(x^{(k)}\right)^T Df\left(x^{(k)}\right) x$$

如果 $Df\left(x^{(k)}\right)$ 的列是线性无关的，则其解为:

$$x^{(k+1)} = x^{(k)} - \left(Df\left(x^{(k)}\right)^T Df\left(x^{(k)}\right)\right)^{-1} Df\left(x^{(k)}\right)^T f\left(x^{(k)}\right)$$

高斯-牛顿法步骤 $\Delta x^{(k)} = x^{(k+1)} - x^{(k)}$:

$$\begin{aligned}\Delta x^{(k)} &= -\left(Df\left(x^{(k)}\right)^T Df\left(x^{(k)}\right)\right)^{-1} Df\left(x^{(k)}\right)^T f\left(x^{(k)}\right) \\ &= -\frac{1}{2}\left(Df\left(x^{(k)}\right)^T Df\left(x^{(k)}\right)\right)^{-1} \nabla g\left(x^{(k)}\right)\end{aligned}$$

(利用第 14.10 的 $\nabla g(x)$ 表达式)。
逼近函数 $\hat{f}\left(x; x^{(k)}\right)$ 的关于 $x^{(k+1)}$ 的代价:

$$\begin{aligned}\| \hat{f}\left(x^{(k+1)}; x^{(k)}\right)\|_2^2 &= \| f\left(x^{(k)}\right) + Df\left(x^{(k)}\right)\Delta x^{(k)} \|_2^2 \\ &= \left\| f\left(x^{(k)}\right)\right\|_2^2 + 2f\left(x^{(k)}\right)^T Df\left(x^{(k)}\right)\Delta x^{(k)} + \left\| Df\left(x^{(k)}\right)\Delta x^{(k)} \right\|_2^2\end{aligned}$$

$$2Df\left(x^{(k)}\right)^T \left(f\left(x^{(k)}\right) + Df\left(x^{(k)}\right)\left(x^{(k+1)} - x^{(k)}\right)\right) = 0 \quad \text{(目标梯度等于0)}$$
$$\Rightarrow Df\left(x^{(k)}\right)^T f\left(x^{(k)}\right) = -Df\left(x^{(k)}\right)^T Df\left(x^{(k)}\right)\Delta x^{(k)}$$
$$\Rightarrow \left(\Delta x^{(k)}\right)^T Df\left(x^{(k)}\right)^T f\left(x^{(k)}\right) = -\left(\Delta x^{(k)}\right)^T Df\left(x^{(k)}\right)^T Df\left(x^{(k)}\right)\Delta x^{(k)} = -\left\| Df\left(x^{(k)}\right)\Delta x^{(k)}\right\|_2^2$$

$$\left\| \hat{f}\left(x^{(k+1)}; x^{(k)}\right)\right\|_2^2 = \left\| f\left(x^{(k)}\right)\right\|_2^2 - \left\| Df\left(x^{(k)}\right)\Delta x^{(k)}\right\|_2^2$$

如果 $Df\left(x^{(k)}\right)$ 的列向量线性无关，且 $\Delta x^{(k)} \neq 0$:

$$\left\|Df\left(x^{(k)}\right)\Delta x^{(k)}\right\|_2^2 > 0 \Rightarrow \left\|\hat{f}\left(x^{(k+1)};x^{(k)}\right)\right\|_2^2 < \left\|f\left(x^{(k)}\right)\right\|_2^2$$

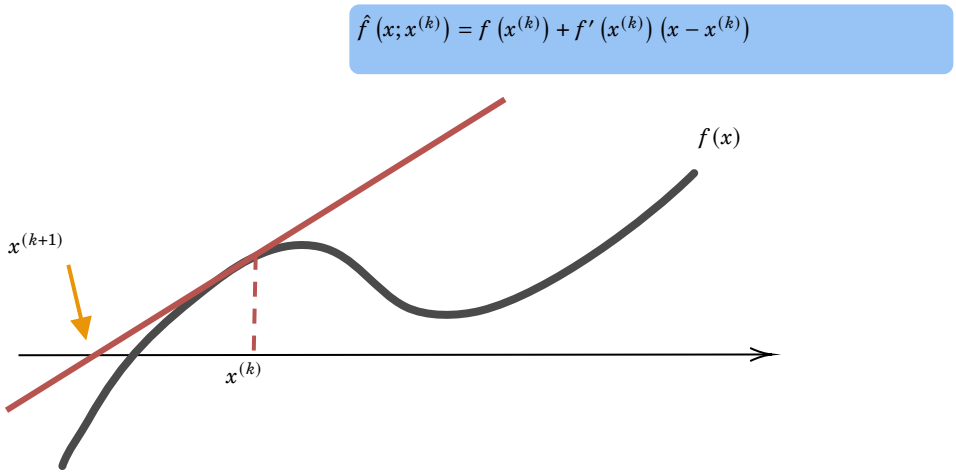
当 m = n 时，$Df\left(x^{(k)}\right)$ 的列向量线性无关，高斯-牛顿法可简化为

$$\begin{aligned}
x^{(k+1)} &= x^{(k)} - \left(Df\left(x^{(k)}\right)^T Df\left(x^{(k)}\right)\right)^{-1} Df\left(x^{(k)}\right)^T f\left(x^{(k)}\right) \\
&= x^{(k)} - \left(Df\left(x^{(k)}\right)\right)^{-1}\left(Df\left(x^{(k)}\right)^T\right)^{-1} Df\left(x^{(k)}\right)^T f\left(x^{(k)}\right) \\
&= x^{(k)} - \left(Df\left(x^{(k)}\right)\right)^{-1} f\left(x^{(k)}\right).
\end{aligned}$$

如果 $m = n = 1$ 时，迭代更新可进一步简化为

$$x^{(k+1)} = x^{(k)} - \frac{f\left(x^{(k)}\right)}{f'\left(x^{(k)}\right)}, f'\left(x^{(k)}\right) \neq 0$$

Figure 17.2: $m = n = 1$ 时的情形



17.4.1 高斯-牛顿法的问题

然而，$\hat{f}\left(x;x^{(k)}\right)$ 只是 $f(x)$ 的局部近似，仍然可能会有

存在

$$\left\|f\left(x^{(k+1)}\right)\right\|_2^2 > \left\|f\left(x^{(k)}\right)\right\|_2^2$$

的情况。

Jacobian 矩阵 Df $\left(x^{(k)}\right)$ 的可能" 列向量是线性相关。

此时 $\left(Df\left(x^{(k)}\right)^T Df\left(x^{(k)}\right)\right)$ 奇异，不可逆。

17.5 Levenberg–Marquardt Algorithm

Levenberg-Marquardt(LM) 算法解决高斯-牛顿法的两个难点：

1. 当列 $Df\left(x^{(k)}\right)$ 是线性相关时，如何更新 $x^{(k)}$ ？
2. 当高斯-牛顿法的更新过程，$\|f(x)\|_2^2$ 并没有减少。

17.5.1 推导 Levenberg–Marquardt 算法

LM 算法通过求解正则化最小二乘法更新 $x^{(k+1)}$ ：

Problem 17.9

$$x^{(k+1)}=\arg\min_x\left\|\hat{f}\left(x;x^{(k)}\right)\right\|_2^2+\lambda^{(k)}\left\|x-x^{(k)}\right\|_2^2$$

要求 x 靠近 $x^{(k)}$ ，即 $\hat{f}\left(x;x^{(k)}\right)\approx f(x)$. (The second term forces x to be close to $x^{(k)}$ where $\hat{f}\left(x;x^{(k)}\right)\approx f(x)$.) ∎

当 $\lambda^{(k)}>0$ 时，总有唯一解.

> Levenberg Marquardt Algorithm 无需再对 $Df\left(x^{(k)}\right)$ 限制。

Problem 17.10 第 k 次迭代求解的最小二乘正则化问题:

$$x^{(k+1)}=\arg\min_x\left\|f\left(x^{(k)}\right)+Df\left(x^{(k)}\right)\left(x-x^{(k)}\right)\right\|_2^2+\lambda^{(k)}\left\|x-x^{(k)}\right\|_2^2$$

∎

令目标函数导数等于 0 :

$$\begin{aligned}&2Df\left(x^{(k)}\right)^T\left(f\left(x^{(k)}\right)+Df\left(x^{(k)}\right)\left(x-x^{(k)}\right)\right)+2\lambda^{(k)}\left(x-x^{(k)}\right)=0\\\Rightarrow&\left(Df\left(x^{(k)}\right)^T Df\left(x^{(k)}\right)+\lambda^{(k)}I\right)\left(x-x^{(k)}\right)=-Df\left(x^{(k)}\right)^T f\left(x^{(k)}\right)\end{aligned}$$

$x^{(k+1)}$ 更新:

$$x^{(k+1)}=x^{(k)}-\left(Df\left(x^{(k)}\right)^T Df\left(x^{(k)}\right)+\lambda^{(k)}I\right)^{-1}Df\left(x^{(k)}\right)^T f\left(x^{(k)}\right)$$

Theorem 17.5.1 LM 算法步骤 $\Delta x^{(k)}=x^{(k+1)}-x^{(k)}$ 是:

$$\begin{aligned}\Delta x^{(k)}&=-\left(Df\left(x^{(k)}\right)^T Df\left(x^{(k)}\right)+\lambda^{(k)}I\right)^{-1}Df\left(x^{(k)}\right)^T f\left(x^{(k)}\right)\\&=-\frac{1}{2}\left(Df\left(x^{(k)}\right)^T Df\left(x^{(k)}\right)+\lambda^{(k)}I\right)^{-1}\nabla g\left(x^{(k)}\right)\end{aligned}$$

Theorem 17.5.2 对于 $\lambda^{(k)}=0$ ，简化为高斯-牛顿法

Theorem 17.5.3 对于较大的 $\lambda^{(k)}$ ：

$$\Delta x^{(k)} \approx -\frac{1}{2\lambda^{(k)}} \nabla g\left(x^{(k)}\right)$$

17.5.2 正则化参数

几种调整 $\lambda^{(k)}$ 的策略.

Algorithm 23: 在第 k 次迭代时，求解 $\hat{x}$

1
$$\hat{x} = \arg\min_x \left\| \hat{f}\left(x; x^{(k)}\right) \right\|_2^2 + \lambda^{(k)} \left\| x - x^{(k)} \right\|_2^2$$

2 **if** $\|f(\hat{x})\|_2^2 < \left\| f\left(x^{(k)}\right) \right\|_2^2$ **then**
3 　$x^{(k+1)} = \hat{x}$
4 　降低 λ
5 **end**
6 **else**
7 　无需更新 x $\left(x^{(k+1)} = x^{(k)}\right)$
8 　增加 λ
9 **end**

它的一些变化:

■ **Example 17.10** 比较预期损失降低的情况和实际损失降低的情况。(compare actual cost reduction with predicted cost reduction) ■

Theorem 17.5.4 在第 k 次迭代时，求解 $\hat{x}$

$$\hat{x} = \arg\min_x \left\| \hat{f}\left(x; x^{(k)}\right) \right\|_2^2 + \lambda^{(k)} \left\| x - x^{(k)} \right\|_2^2$$

等价于
具有“置信区间”的最小二乘法问题: $\min_x \left\| \hat{f}\left(x; x^{(k)}\right) \right\|_2^2$ subject to $\left\| x - x^{(k)} \right\|_2^2 \leq \gamma$

- 其中 β_1, β_2 是常数，$0 < \beta_1 < 1 < \beta_2$ 。
- 在步骤 2 中，可以使用 QR 分解来计算 $\hat{x}$ 。
- 迭代会在当 $\nabla g\left(x^{(k)}\right) = 2A^T f\left(x^{(k)}\right)$ 足够小时终止。

17.6 Newton Method

Proposition 17.6.1 Assume $f : \mathbf{R}^n \to \mathbf{R}^n$ is differentiable.

$Df\left(x^{(k)}\right)$ is the derivative matrix of f at $x^{(k)}$.

Each iteration requires one evaluation of $f(x)$ and $Df(x)$, each iteration requires factorization of the $n \times n$ matrix $Df(x)$.

Proposition 17.6.2 We assume $Df(x)$ is nonsingular.

Algorithm 24: Levenberg Marquardt Algorithm

选择 $x^{(1)}$ 和 $\lambda^{(1)}$

while $k = 1, 2, \ldots$ **do**

求 $f\left(x^{(k)}\right)$

$A = Df\left(x^{(k)}\right)$

计算最小二乘法正则化问题的解:

$$\hat{x} = x^{(k)} - \left(A^T A + \lambda^{(k)} I\right)^{-1} A^T f\left(x^{(k)}\right)$$

计算 $x^{(k+1)}$ 和 $\lambda^{(k+1)}$:

$$\begin{cases} x^{(k+1)} = \hat{x}, & \lambda^{(k+1)} = \beta_1 \lambda^{(k)} \quad \left(\text{if } \|(\hat{x})\|_2^2 < \left\|f\left(x^{(k)}\right)\right\|_2^2\right) \\ x^{(k+1)} = x^{(k)}, & \lambda^{(k+1)} = \beta_2 \lambda^{(k)} \quad (\text{others}) \end{cases}$$

end

Algorithm 25: Newton's Method

choose $x^{(1)}$

while $k = 1, 2, \ldots$ **do**

$$x^{(k+1)} = x^{(k)} - Df\left(x^{(k)}\right)^{-1} f\left(x^{(k)}\right)$$

end

17.6.1 Interpretation of Newton Method

$$x^{(k+1)} = x^{(k)} - Df\left(x^{(k)}\right)^{-1} f\left(x^{(k)}\right)$$

linearize f (i.e., make affine approximation) around current iterate $x^{(k)}$

$$\hat{f}\left(x; x^{(k)}\right) = f\left(x^{(k)}\right) + Df\left(x^{(k)}\right)\left(x - x^{(k)}\right)$$

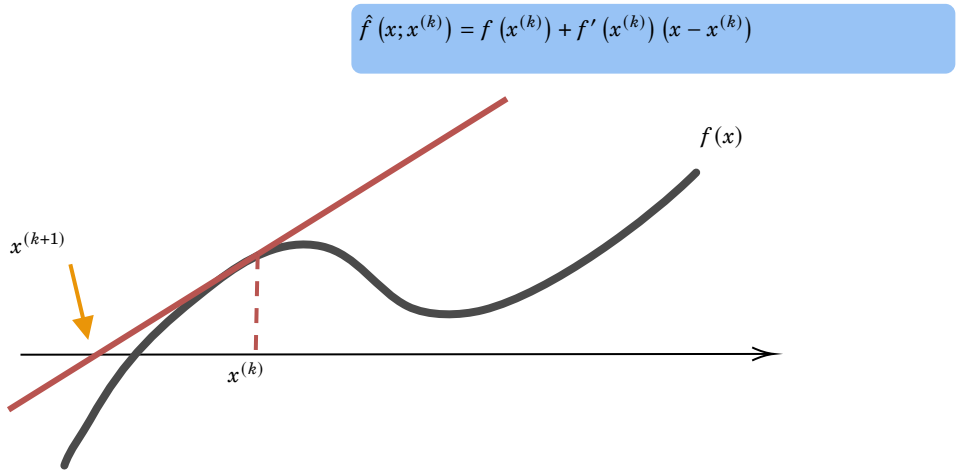
solve the linearized equation $\hat{f}\left(x; x^{(k)}\right) = 0$; the solution is

$$x = x^{(k)} - Df\left(x^{(k)}\right)^{-1} f\left(x^{(k)}\right)$$

take the solution x of the linearized equation as the next iterate $x^{(k+1)}$

17.6.2 One Variable

Figure 17.3: $m = n = 1$ 时的情形



affine approximation of f around $x^{(k)}$ is

$$\hat{f}\left(x; x^{(k)}\right) = f\left(x^{(k)}\right) + f'\left(x^{(k)}\right)\left(x - x^{(k)}\right)$$

solve the linearized equation $\hat{f}\left(x; x^{(k)}\right) = 0$ and take the solution as $x^{(k+1)}$:

$$x^{(k+1)} = x^{(k)} - \frac{f\left(x^{(k)}\right)}{f'\left(x^{(k)}\right)}$$

17.6.3 Relation to Gauss–Newton method

recall Gauss-Newton method for nonlinear least squares problem

Problem 17.11 — nonlinear least squares problem.

$$\text{minimize} \quad \|f(x)\|^2$$

■

where f is a differentiable function from $\mathbf{R}^n$ to $\mathbf{R}^m$.
Gauss-Newton update

$$x^{(k+1)} = x^{(k)} - \left(Df\left(x^{(k)}\right)^T Df\left(x^{(k)}\right)\right)^{-1} Df\left(x^{(k)}\right)^T f\left(x^{(k)}\right)$$

If $m = n$, then $Df(x)$ is square and this is the Newton update

$$x^{(k+1)} = x^{(k)} - Df\left(x^{(k)}\right)^{-1} f\left(x^{(k)}\right)$$

17.6.4 Examples

■ **Example 17.11**

$$f_1(x_1, x_2) = \log\left(x_1^2 + 2x_2^2 + 1\right) - 0.5 = 0$$

$$f_2(x_1, x_2) = x_2 - x_1^2 + 0.2 = 0$$

two equations in two variables; two solutions $(0.70, 0.29), (-0.70, 0.29)$.
Newton iteration

Algorithm 26: Newton iteration

1 evaluate $g = f(x)$ and

$$H = Df(x) = \begin{bmatrix} 2x_1/\left(x_1^2 + 2x_2^2 + 1\right) & 4x_2/\left(x_1^2 + 2x_2^2 + 1\right) \\ -2x_1 & 1 \end{bmatrix}$$

2 solve $Hv = -g$ (two linear equations in two variables)
3 update $x := x + v$

Results:
- $x^{(1)} = (1, 1)$: converges to $x^\star = (0.70, 0.29)$ in about 4 iterations
- $x^{(1)} = (-1, 1)$: converges to $x^\star = (-0.70, 0.29)$ in about 4 iterations
- $x^{(1)} = (1, -1)$ or $x^{(0)} = (-1, -1)$: does not converge

■

- Newton's method works very well if started near a solution, may not work otherwise
- can converge to different solutions depending on the starting point
- does not necessarily find the solution closest to the starting point

17.6.5 Convergence of Newton's method

Quadratic convergence explains fast convergence when started near solution.

Theorem 17.6.3 — Quadratic Convergence. if $f(x^\star) = 0$ and $Df(x^\star)$ is nonsingular, and $x^{(1)}$ is sufficiently close to $x^\star$, then

$$x^{(k)} \to x^\star, \quad \left\|x^{(k+1)} - x^\star\right\| \le c\left\|x^{(k)} - x^\star\right\|^2$$

for some $c > 0$

17.6.6 Newton's method for minimizing a convex function

if $\nabla^2 g(x)$ is positive definite everywhere, we can minimize $g(x)$ by solving

$$\nabla g(x) = 0$$

Algorithm:

Algorithm 27: Newton's method for minimizing a convex function

1 choose $x^{(1)}$
2 **while** $k = 1, 2, \ldots$ **do**
3 $$x^{(k+1)} = x^{(k)} - \nabla^2 g\left(x^{(k)}\right)^{-1} \nabla g\left(x^{(k)}\right)$$
4 **end**

$v = -\nabla^2 g(x)^{-1} \nabla g(x)$ is called the Newton step at x.

It converges if started sufficiently close to the solution. Newton step is computed by a Cholesky factorization of the Hessian.

Interpretations of Newton step

Definition 17.6.1 — **affine approximation of $f(x) = \nabla g(x)$ around $x^{(k)}$.**

$$\hat{f}\left(x; x^{(k)}\right) = \nabla g\left(x^{(k)}\right) + \nabla^2 g\left(x^{(k)}\right)\left(x - x^{(k)}\right)$$

is

Newton update $x^{(k+1)}$ is solution of linear equation $\hat{f}\left(x; x^{(k)}\right) = 0$.

Definition 17.6.2 — **quadratic approximation of $g(x)$ around $x^{(k)}$.**

$$g_q\left(x; x^{(k)}\right) = g\left(x^{(k)}\right) + \nabla g\left(x^{(k)}\right)^T \left(x - x^{(k)}\right) + \frac{1}{2}\left(x - x^{(k)}\right)^T \nabla^2 g\left(x^{(k)}\right)\left(x - x^{(k)}\right)$$

Newton update $x^{(k+1)}$ minimizes $g_q\left(x; x^{(k)}\right)$ (satisfies $\nabla g_q\left(x; x^{(k)}\right) = 0$)

17.6.7 Damped Newton method

Positive scalar t is called the *step size*, step 2 in algorithm is called *line search*.

Interpretation of line search

to determine a suitable step size, consider the function $h : \mathbf{R} \to \mathbf{R}$

$$h(t) = g\left(x^{(k)} + tv\right)$$

$h'(0) = \nabla g\left(x^{(k)}\right)^T v$ is the directional derivative at $x^{(k)}$ in the direction v
line search terminates with positive t if $h'(0) < 0$ (v is a descent direction)
if $\nabla^2 g\left(x^{(k)}\right)$ is positive definite, the Newton step is a descent direction

$$h'(0) = \nabla g\left(x^{(k)}\right)^T v = -v^T \nabla^2 g\left(x^{(k)}\right) v < 0$$

Algorithm 28: Damped Newton method

choose $x^{(1)}$
while $k = 1, 2, \ldots$ **do**
  compute Newton step $v = -\nabla^2 g\left(x^{(k)}\right)^{-1} \nabla g\left(x^{(k)}\right)$
  (Line Search) find largest t in $\left\{1, 0.5, 0.5^2, 0.5^3, \ldots\right\}$ that satisfies

$$g\left(x^{(k)} + tv\right) < g\left(x^{(k)}\right)$$

  $x^{(k+1)} = x^{(k)} + tv$
end

Newton method for nonconvex functions
if $\nabla^2 g\left(x^{(k)}\right)$ is not positive definite, it is possible that Newton step v satisfies

$$\nabla g\left(x^{(k)}\right)^T v = -\nabla g\left(x^{(k)}\right)^T \nabla^2 g\left(x^{(k)}\right)^{-1} \nabla g\left(x^{(k)}\right) > 0$$

- if Newton step is not descent direction, replace it with descent direction - simplest choice is $v = -\nabla g\left(x^{(k)}\right)$; practical methods make other choices

17.7 Newton method for nonlinear least squares

17.7.1 Hessian of nonlinear least squares cost

$$g(x) = \|f(x)\|^2 = \sum_{i=1}^{m} f_i(x)^2$$

Theorem 17.7.1 gradient:

$$\nabla g(x) = 2\sum_{i=1}^{m} f_i(x)\nabla f_i(x) = 2Df(x)^T f(x)$$

Theorem 17.7.2 second derivatives:

$$\frac{\partial^2 g}{\partial x_j \partial x_k}(x) = 2\sum_{i=1}^{m}\left(\frac{\partial f_i}{\partial x_j}(x)\frac{\partial f_i}{\partial x_k}(x) + f_i(x)\frac{\partial^2 f_i}{\partial x_j \partial x_k}(x)\right)$$

Theorem 17.7.3 Hessian

$$\nabla^2 g(x) = 2Df(x)^T Df(x) + 2\sum_{i=1}^{m} f_i(x)\nabla^2 f_i(x)$$

Theorem 17.7.4 (Undamped) Newton step at $x = x^{(k)}$:

$$\begin{aligned} v_{\text{nt}} &= -\nabla^2 g(x)^{-1} \nabla g(x) \\ &= -\left(Df(x)^T Df(x) + \sum_{i=1}^{m} f_i(x) \nabla^2 f_i(x) \right)^{-1} Df(x)^T f(x) \end{aligned}$$

Theorem 17.7.5 Gauss-Newton step at $x = x^{(k)}$

$$v_{\text{gn}} = -\left(Df(x)^T Df(x) \right)^{-1} Df(x)^T f(x)$$

Corollary 17.7.6 can be written as $v_{\text{gn}} = -H_{\text{gn}}^{-1} \nabla g(x)$ where $H_{\text{gn}} = 2Df(x)^T Df(x)$
H_{gn} is the Hessian without the term $\sum_i f_i(x) \nabla^2 f_i(x)$.

17.7.2 Comparison

Newton step:

- requires second derivatives of f
- not always a descent direction ($\nabla^2 g(x)$ is not necessarily positive definite)
- fast convergence near local minimum

Gauss-Newton step:

- does not require second derivatives
- a descent direction (if columns of $Df(x)$ are linearly independent):

$$\nabla g(x)^T v_{\text{gn}} = -2 v_{\text{gn}}^T Df(x)^T Df(x) v_{\text{gn}} < 0 \quad \text{if } v_{\text{gn}} \neq 0$$

- local convergence to $x^\star$ is similar to Newton method if

$$\sum_{i=1}^{m} f_i\left(x^\star\right) \nabla^2 f_i\left(x^\star\right)$$

is small (e.g., $f\left(x^\star\right)$ is small, or f is nearly affine around $x^\star$)

17.8 Unconstrained minimization problem

Problem 17.12 Assume that g is twice differentiable
minimize $g\left(x_1, x_2, \ldots, x_n\right)$
g is a function from $\mathbf{R}^n$ to $\mathbf{R}$ (the cost function or objective function), $x = (x_1, x_2, \ldots, x_n)$ is n-vector of optimization variables. ∎
to solve a maximization problem (i.e., maximize $g(x)$), minimize $-g(x)$.

17.9 Local and global optimum

Definition 17.9.1 — **globally optimal.** $x^\star$ is an optimal point (or a minimum) if

$$g\left(x^\star\right) \leq g(x) \quad \text{for all } x$$

It is also called globally optimal.

Definition 17.9.2 — locally optimal. $x^\star$ is a locally optimal point (local minimum) if for some $R > 0$ $g(x^\star) \leq g(x)$ for all x with $\|x - x^\star\| \leq R$

It is also called locally optimal.

IV Extensive Reading

18. Fourier Series, Fourier Transform

18.1 基本概念

Definition 18.1.1 — **波.** 波的基本属性：**频率、振幅、相位。**

Definition 18.1.2 — **Complex Numbers.**

$$C = R + jI$$

where R and I are real numbers and $j = \sqrt{-1}$. Here, R denotes the real part of the complex number and I its imaginary part.

Real numbers are a subset of complex numbers in which $I = 0$.

Definition 18.1.3 — **Complex Number in Polar Coordinates.**

$$C = |C|(\cos\theta + j\sin\theta) = |C|e^{j\theta}$$

where $|C| = \sqrt{R^2 + I^2}$ is the length of the vector extending from the origin of the complex plane to point (R, I), and θ is the angle between the vector and the real axis.

上式使用了欧拉公式

Theorem 18.1.1 — **Euler's Formular.**

$$e^{j\theta} = \cos\theta + j\sin\theta$$

Corollary 18.1.2 $e^{2\pi it}$ 可以表示每秒 1 圈的旋转.

Definition 18.1.4 — 正弦函数.

$$y = A\sin(\omega t + \varphi)$$

就是一个以 $\frac{2\pi}{\omega}$ 为周期的正弦函数, 其中 y 表示动点的位置, t 表示时间, A 为振幅, ω 为角频率, φ 为初相.

如何深入研究非正弦周期函数呢? 将周期函数展开成由简单的周期函数例如三角函数组成的级数. 具体地说, 将周期为 $T\left(=\frac{2\pi}{\omega}\right)$ 的周期函数用一系列以 T 为周期的正弦函数 $A_n\sin(n\omega t + \varphi_n)$ 组成的级数来表示, 记为

$$f(t) = A_0 + \sum_{n=1}^{\infty} A_n \sin(n\omega t + \varphi_n)$$

其中 $A_0, A_n, \varphi_n (n = 1, 2, 3, \cdots)$ 都是常数.

将周期函数按上述方式展开, 它的物理意义是很明确的, 这就是把一个比较复杂的周期运动看成是许多不同频率的简谐振动的叠加. 在电工学上, 这种展开称为谐波分析, 其中常数项 A_0 称为 $f(t)$ 的直流分量, $A_1\sin(\omega t + \varphi_1)$ 称为一次谐波 (又叫做基波), $A_2\sin(2\omega t + \varphi_2), A_3\sin(3\omega t + \varphi_3), \cdots$ 依次称为二次谐波, 三次谐波, 等等.

Definition 18.1.5 — 三角级数. 令 $\frac{a_0}{2} = A_0, a_n = A_n\sin\varphi_n, b_n = A_n\cos\varphi_n, \omega = \frac{\pi}{l}$(即 $T = 2l$), 则 $f(t) = A_0 + \sum_{n=1}^{\infty} A_n\sin(n\omega t + \varphi_n)$ 可以改写为

$$\frac{a_0}{2} + \sum_{n=1}^{\infty}\left(a_n\cos\frac{n\pi t}{l} + b_n\sin\frac{n\pi t}{l}\right)$$

其中 $a_0, a_n, b_n (n = 1, 2, 3, \cdots)$ 都是常数.

Definition 18.1.6 — 三角函数系（基波）.

$$1, \cos x, \sin x, \cos 2x, \sin 2x, \cdots, \cos nx, \sin nx, \cdots$$

在区间 $[-\pi, \pi]$ 上正交, 就是指在三角函数系 $(7-4)$ 中任何不同的两个函数的乘积在区间 $[-\pi, \boldsymbol{\pi}]$ 上的积分等于零, 即

$$\begin{array}{l}\int_{-\pi}^{\pi}\cos nx\,\mathrm{d}x = 0 \quad (n = 1, 2, 3, \cdots),\\ \int_{-\pi}^{\pi}\sin nx\,\mathrm{d}x = 0 \quad (n = 1, 2, 3, \cdots),\\ \cdots\end{array}$$

Definition 18.1.7 — 复数域的 Gram 矩阵. 如果 $A \in \mathbb{C}^{m\times n}$ 的 Gram 矩阵为单位矩阵, 则 A 具有正交列:

$$
\begin{aligned}
A^H A &= \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix}^H \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix} \\
&= \begin{bmatrix} a_1^H a_1 & a_1^H a_2 & \cdots & a_1^H a_n \\ a_2^H a_1 & a_2^H a_2 & \cdots & a_2^H a_n \\ \vdots & \vdots & & \vdots \\ a_n^H a_1 & a_n^H a_2 & \cdots & a_n^H a_n \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \\
&= I
\end{aligned}
$$

Gram 矩阵列有单位范数：$\|a_i\|_2^2 = a_i^H a_i = 1$ 。
Gram 矩阵列是相互正交的：对于 $i \neq j, a_i^H a_j = 0$ 。

Definition 18.1.8 — 酉矩阵. 列正交的方形复数矩阵称为酉矩阵。

Definition 18.1.9 — 酉矩阵的逆.

$$
\left.\begin{array}{l} A^H A = I \\ A \text{ 是方的} \end{array}\right\} \quad \Rightarrow \quad AA^H = I
$$

酉矩阵是具有逆 A^H 的非奇异矩阵。如果 A 是酉矩阵, 那么 A^H 也是酉矩阵。

18.2 Fourier Series

Definition 18.2.1 — 傅里叶级数.

$$
f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} \left(a_k \cos kx + b_k \sin kx\right).
$$

where

$$
\begin{cases} a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos nx \, \mathrm{d}x & (n = 0, 1, 2, 3, \cdots), \\ b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin nx \, \mathrm{d}x & (n = 1, 2, 3, \cdots). \end{cases}
$$

Proof. 设 $f(x)$ 是周期为 2π 的周期函数, 且能展开成三角级数

$$
f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} \left(a_k \cos kx + b_k \sin kx\right).
$$

先求 a_0. 上式一从 $-\pi$ 到 π 积分, 由于假设式右端级数可逐项积分，因此有

$$
\int_{-\pi}^{\pi} f(x)\mathrm{d}x = \int_{-\pi}^{\pi} \frac{a_0}{2} \, \mathrm{d}x + \sum_{k=1}^{\infty} \left[a_k \int_{-\pi}^{\pi} \cos kx \, \mathrm{d}x + b_k \int_{-\pi}^{\pi} \sin kx \, \mathrm{d}x \right]
$$

根据三角函数系的正交性，等式右端除第一项外，其余各项均为零，所以

$$
\int_{-\pi}^{\pi} f(x)\mathrm{d}x = \frac{a_0}{2} \cdot 2\pi
$$

于是得

$$a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\mathrm{d}x$$

其次求 a_n. 用 $\cos nx$ 乘式两端, 再从 $-\pi$ 到 π 积分, 得到

$$\begin{aligned}&\int_{-\pi}^{\pi} f(x)\cos nx\ \mathrm{d}x\\ =&\frac{a_0}{2}\int_{-\pi}^{\pi}\cos nx\ \mathrm{d}x+\sum_{k=1}^{\infty}\left[a_k\int_{-\pi}^{\pi}\cos kx\cos nx\ \mathrm{d}x+b_k\int_{-\pi}^{\pi}\sin kx\cos nx\ \mathrm{d}x\right]\end{aligned}$$

根据三角函数系的正交性, 等式右端除 $k=n$ 的一项外, 其余各项均为 0. 所以

$$\int_{-\pi}^{\pi} f(x)\cos nx\ \mathrm{d}x = a_n\int_{-\pi}^{\pi}\cos^2 nx\ \mathrm{d}x = a_n\pi$$

于是得

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos nx\ \mathrm{d}x \quad (n=1,2,3,\cdots)$$

类似地, 用 $\sin nx$ 乘 $(7-5)$ 式的两端, 再从 $-\pi$ 到 π 积分, 可得

$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin nx\ \mathrm{d}x \quad (n=1,2,3,\cdots)$$

■

Theorem 18.2.1 — 收敛定理, 狄利克雷 (Dirichlet) 充分条件. 设 $f(x)$ 是周期为 2π 的周期函数, 如果它满足:

1. 在一个周期内连续或只有有限个第一类间断点,
2. 在一个周期内至多只有有限个极值点

那么 $f(x)$ 的傅里叶级数收敛, 并且

- 当 x 是 $f(x)$ 的连续点时, 级数收敛于 $f(x)$;
- 当 x 是 $f(x)$ 的间断点时, 级数收敛于 $\frac{1}{2}\left[f\left(x^-\right)+f\left(x^+\right)\right]$.

收敛定理告诉我们: 只要函数在 $[-\pi,\pi]$ 上至多有有限个第一类间断点, 并且不作无限次振动, 函数的傅里叶级数在连续点处就收敛于该点的函数值, 在间断点处收敛于该点左极限与右极限的算术平均值. 可见, 函数展开成傅里叶级数的条件比展开成幂级数的条件低得多.

Corollary 18.2.2 记 $C=\left\{x \mid f(x)=\frac{1}{2}\left[f\left(x^-\right)+f\left(x^+\right)\right]\right\}$
在 C 上就成立 $f(x)$ 的傅里叶级数展开式

$$f(x) = \frac{a_0}{2}+\sum_{n=1}^{\infty}\left(a_n\cos nx+b_n\sin nx\right), x\in C$$

Definition 18.2.2 — 正弦级数. 当 $f(x)$ 为奇函数时, $f(x)\cos nx$ 是奇函数, $f(x)\sin nx$ 是偶函数, 故

$$\left.\begin{aligned}a_n &= 0 \quad (n=0,1,2,\cdots),\\ b_n &= \tfrac{2}{\pi}\textstyle\int_0^{\pi} f(x)\sin nx\ \mathrm{d}x \quad (n=1,2,3,\cdots).\end{aligned}\right\}$$

即知奇函数的傅里叶级数是只含有正弦项的正弦级数.

$$\sum_{n=1}^{\infty} b_n \sin nx$$

Definition 18.2.3 — **余弦级数.** 当 $f(x)$ 为偶函数时, $f(x)\cos nx$ 是偶函数, $f(x)\sin nx$ 是奇函数, 故

$$\left.\begin{array}{l} a_n = \frac{2}{\pi}\int_0^{\pi} f(x)\cos nx\,\mathrm{d}x \quad (n=0,1,2,\cdots) \\ b_n = 0 \quad (n=1,2,3,\cdots) \end{array}\right\}$$

即知偶函数的傅里叶级数是只含常数项和余弦项的余弦级数

$$\frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos nx$$

在实际应用 (如研究某种波动问题, 热的传导、扩散问题) 中, 有时还需要把定义在区间 $[0,\pi]$ 上的函数 $f(x)$ 展开成正弦级数或余弦级数.

根据前面讨论的结果, 这类展开问题可以按如下的方法解决: 设函数 $f(x)$ 定义在区间 $[0,\pi]$ 上并且满足收玫定理的条件, 我们在开区间 $(-\pi,0)$ 内补充函数 $f(x)$ 的定义, 得到定义在 $(-\pi,\pi]$ 上的函数 $F(x)$, 使它在 $(-\pi,\pi)$ 上成为奇函数 𝕀(偶函数). 按这种方式拓广函数定义域的过程称为奇延拓 (偶延拓). 然后将奇延拓 (偶延拓) 后的函数展开成傅里叶级数, 这个级数必定是正弦级数 (余弦级数). 再限制 x 在 $(0,\pi]$ 上, 此时 $F(x)\equiv f(x)$, 这样便得到 $f(x)$ 的正弦级数 (余弦级数) 展开式.

Theorem 18.2.3 设周期为 $2l$ 的周期函数 $f(x)$ 满足收敛定理的条件, 则它的傅里叶级数展开式为

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty}\left(a_n\cos\frac{n\pi x}{l} + b_n\sin\frac{n\pi x}{l}\right)(x\in C), \quad (8-1)$$

其中

$$\left.\begin{array}{l} a_n = \frac{1}{l}\int_{-l}^{l} f(x)\cos\frac{n\pi x}{l}\,\mathrm{d}x \quad (n=0,1,2,\cdots) \\ b_n = \frac{1}{l}\int_{-l}^{l} f(x)\sin\frac{n\pi x}{l}\,\mathrm{d}x \quad (n=1,2,3,\cdots), \\ C = \left\{x \mid f(x) = \frac{1}{2}\left[f\left(x^-\right) + f\left(x^+\right)\right]\right\} \end{array}\right\}$$

当 $f(x)$ 为奇函数时,

$$f(x) = \sum_{n=1}^{\infty} b_n \sin\frac{n\pi x}{l} \quad (x\in C)$$

其中

$$b_n = \frac{2}{l}\int_0^{l} f(x)\sin\frac{n\pi x}{l}\,\mathrm{d}x \quad (n=1,2,3,\cdots)$$

当 $f(x)$ 为偶函数时,

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n\cos\frac{n\pi x}{l}(x\in C)$$

其中

$$a_n = \frac{2}{l}\int_0^l f(x)\cos\frac{n\pi x}{l}\,\mathrm{d}x \quad (n = 0, 1, 2, \cdots)$$

Definition 18.2.4 — 傅里叶级数的复数形式.

$$\sum_{n=-\infty}^{\infty} c_n \mathrm{e}^{\frac{n\pi x}{l}j}$$

where $c_n = \frac{1}{2l}\int_{-l}^{l} f(x)\mathrm{e}^{-\frac{n\pi x}{l}j}\mathrm{d}x \quad (n = 0, \pm1, \pm2, \cdots)$.

Proof. 设周期为 $2l$ 的周期函数 $f(x)$ 的傅里叶级数为

$$\frac{a_0}{2} + \sum_{n=1}^{\infty}\left(a_n\cos\frac{n\pi x}{l} + b_n\sin\frac{n\pi x}{l}\right)$$

其中系数 a_n 与 b_n 为

$$\left.\begin{aligned} a_n &= \tfrac{1}{l}\textstyle\int_{-l}^{l} f(x)\cos\frac{n\pi x}{l}\,\mathrm{d}x \quad (n = 0, 1, 2, \cdots), \\ b_n &= \tfrac{1}{l}\textstyle\int_{-l}^{l} f(x)\sin\frac{n\pi x}{l}\,\mathrm{d}x \quad (n = 1, 2, 3, \cdots). \end{aligned}\right\}$$

利用欧拉公式

$$\cos t = \frac{\mathrm{e}^{t\mathrm{i}} + \mathrm{e}^{-t\mathrm{i}}}{2}, \sin t = \frac{\mathrm{e}^{t\mathrm{i}} - \mathrm{e}^{-t\mathrm{i}}}{2\mathrm{i}}$$

记

$$\frac{a_0}{2} = c_0, \quad \frac{a_n - b_n\mathrm{i}}{2} = c_n, \quad \frac{a_n + b_n\mathrm{i}}{2} = c_{-n} \quad (n = 1, 2, 3, \cdots)$$

则表示为

$$c_0 + \sum_{n=1}^{\infty}\left(c_n\mathrm{e}^{\frac{n\pi x}{l}i} + c_{-n}\mathrm{e}^{-\frac{n\pi x}{l}i}\right) = \left(c_n\mathrm{e}^{\frac{n\pi x}{l}i}\right)_{n=0} + \sum_{n=1}^{\infty}\left(c_n\mathrm{e}^{\frac{n\pi x}{l}i} + c_{-n}\mathrm{e}^{-\frac{n\pi x}{l}i}\right)$$

$$c_0 = \frac{a_0}{2} = \frac{1}{2l}\int_{-l}^{l} f(x)\mathrm{d}x$$

$$\begin{aligned} c_n &= \frac{a_n - b_n i}{2} \\ &= \frac{1}{2}\left[\frac{1}{l}\int_{-l}^{l} f(x)\cos\frac{n\pi x}{l}\,\mathrm{d}x - \frac{\mathrm{i}}{l}\int_{-l}^{l} f(x)\sin\frac{n\pi x}{l}\,\mathrm{d}x\right] \\ &= \frac{1}{2l}\int_{-l}^{l} f(x)\left(\cos\frac{n\pi x}{l} - \mathrm{i}\sin\frac{n\pi x}{l}\right)\mathrm{d}x \\ &= \frac{1}{2l}\int_{-l}^{l} f(x)\mathrm{e}^{-\frac{n\pi}{l}}\,\mathrm{d}x \quad (n = 1, 2, 3, \cdots); \end{aligned}$$

$$c_{-n} = \frac{a_n + b_n\mathrm{i}}{2} = \frac{1}{2l}\int_{-l}^{l} f(x)\mathrm{e}^{\frac{n\pi x}{l}i}\,\mathrm{d}x \quad (n = 1, 2, 3, \cdots)$$

将已得的结果合并写为

$$c_n = \frac{1}{2l}\int_{-l}^{l} f(x)\mathrm{e}^{-\frac{n\pi x}{l}}\,\mathrm{d}x \quad (n = 0, \pm 1, \pm 2, \cdots)$$

■

18.3 Fourier Transform

Definition 18.3.1 — The Fourier transform of a continuous function $f(t)$ of a continuous variable t.

$$F(\mu) = \Im\{f(t)\} = \int_{-\infty}^{\infty} f(t)e^{-j2\pi\mu t}dt$$

Using Euler's formula, we can write as

$$F(\mu) = \int_{-\infty}^{\infty} f(t)[\cos(2\pi\mu t) - j\sin(2\pi\mu t)]dt$$

傅里叶变换将一个复杂的波变换成不同的正弦波（基波），时频图（时域、频域、某一刻相位）。对于任意一个时域的曲线，均可以分解成为不同的基波的

$$\text{时域} f(t) \Leftrightarrow F(u, v) \text{频域}$$

Definition 18.3.2 — inverse Fourier transform.

$$f(t) = \int_{-\infty}^{\infty} F(\mu)e^{j2\pi\mu t}d\mu$$

18.4 Discrete Fourier Transform

Definition 18.4.1 — Discrete Fourier Transform.

$$F_m = \sum_{n=0}^{M-1} f_n e^{-j2\pi mn/M} \quad m = 0, 1, 2, \ldots, M-1$$

Definition 18.4.2 — inverse discrete Fourier transform (IDFT).

$$f_n = \frac{1}{M}\sum_{m=0}^{M-1} F_m e^{j2\pi mn/M} \quad n = 0, 1, 2, \ldots, M-1$$

Definition 18.4.3 — 离散傅里叶变换矩阵 W.

$$W = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \cdots & \omega^{-(n-1)} \\ 1 & \omega^{-2} & \omega^{-4} & \cdots & \omega^{-2(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{-(n-1)} & \omega^{-2(n-1)} & \cdots & \omega^{-(n-1)(n-1)} \end{bmatrix}$$

where $\omega = e^{2\pi j/n}$, $\quad j = \sqrt{-1}$

Corollary 18.4.1 矩阵 $(1/\sqrt{n})W$ 是酉矩阵:

$$\frac{1}{n}W^H W = \frac{1}{n}WW^H = I$$

Corollary 18.4.2 W 的逆 $W^{-1} = (1/n)W^H$ 。

Corollary 18.4.3 W 的共轭转置为

$$W^H = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^1 & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{bmatrix}$$

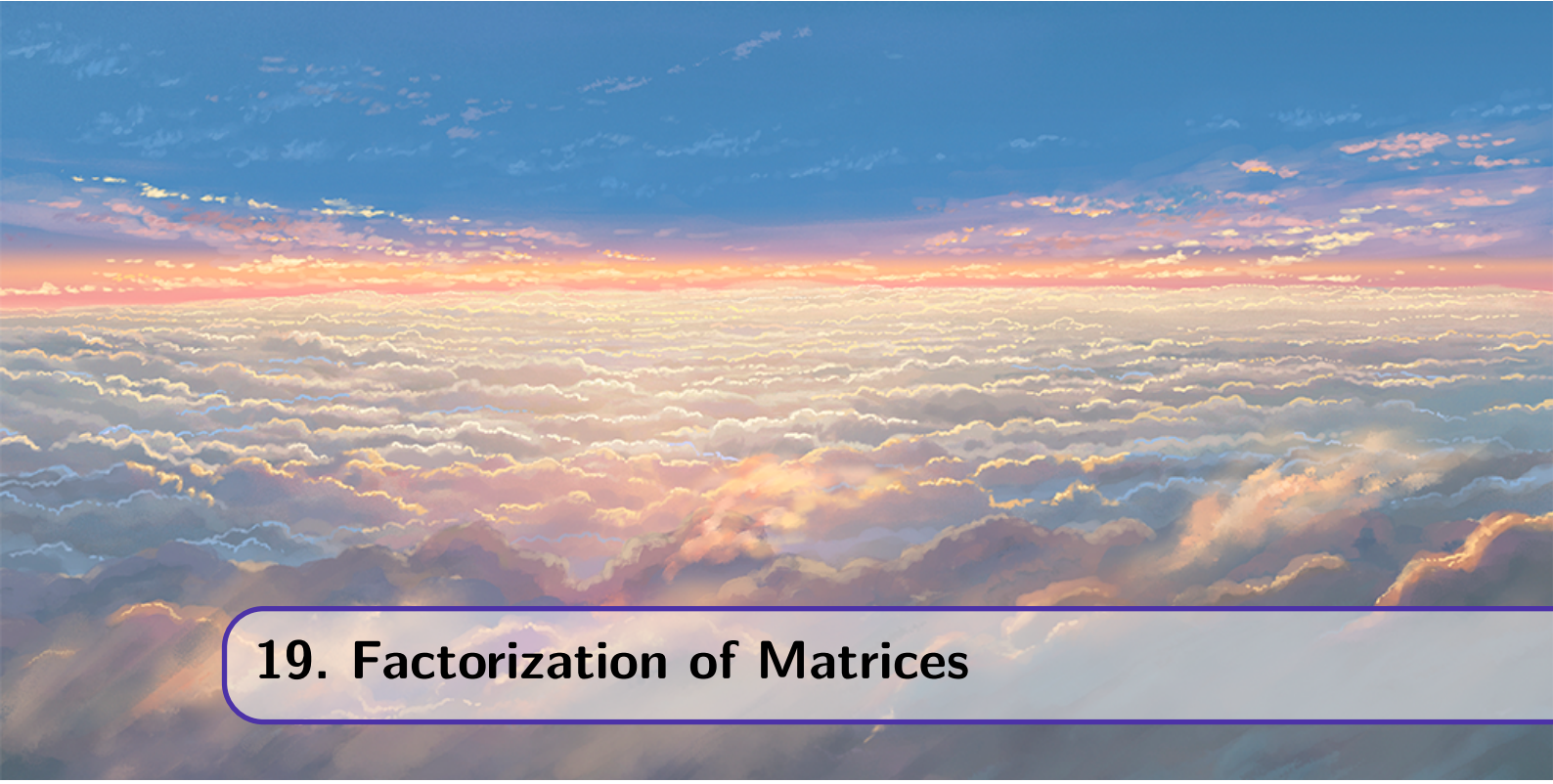
Corollary 18.4.4 Gram 矩阵的第 i, j 个元素为

$$\left(W^H W\right)_{ij} = 1 + \omega^{i-j} + \omega^{2(i-j)} + \cdots + \omega^{(n-1)(i-j)}$$

$\left(W^H W\right)_{ii} = n, \left(W^H W\right)_{ij} = \frac{\omega^{n(i-j)}-1}{\omega^{i-j}-1} = 0$ 如果 $i \neq j$
最后一步因为 $\omega^n = 1$ 。

Definition 18.4.4 — **离散傅里叶反变换.** n 维向量 x 的离散傅里叶反变换是

$$W^{-1}x = (1/n)W^H x$$



19. Factorization of Matrices

19.1 主要的矩阵分解

$$A = LU$$

$$A = LPU$$

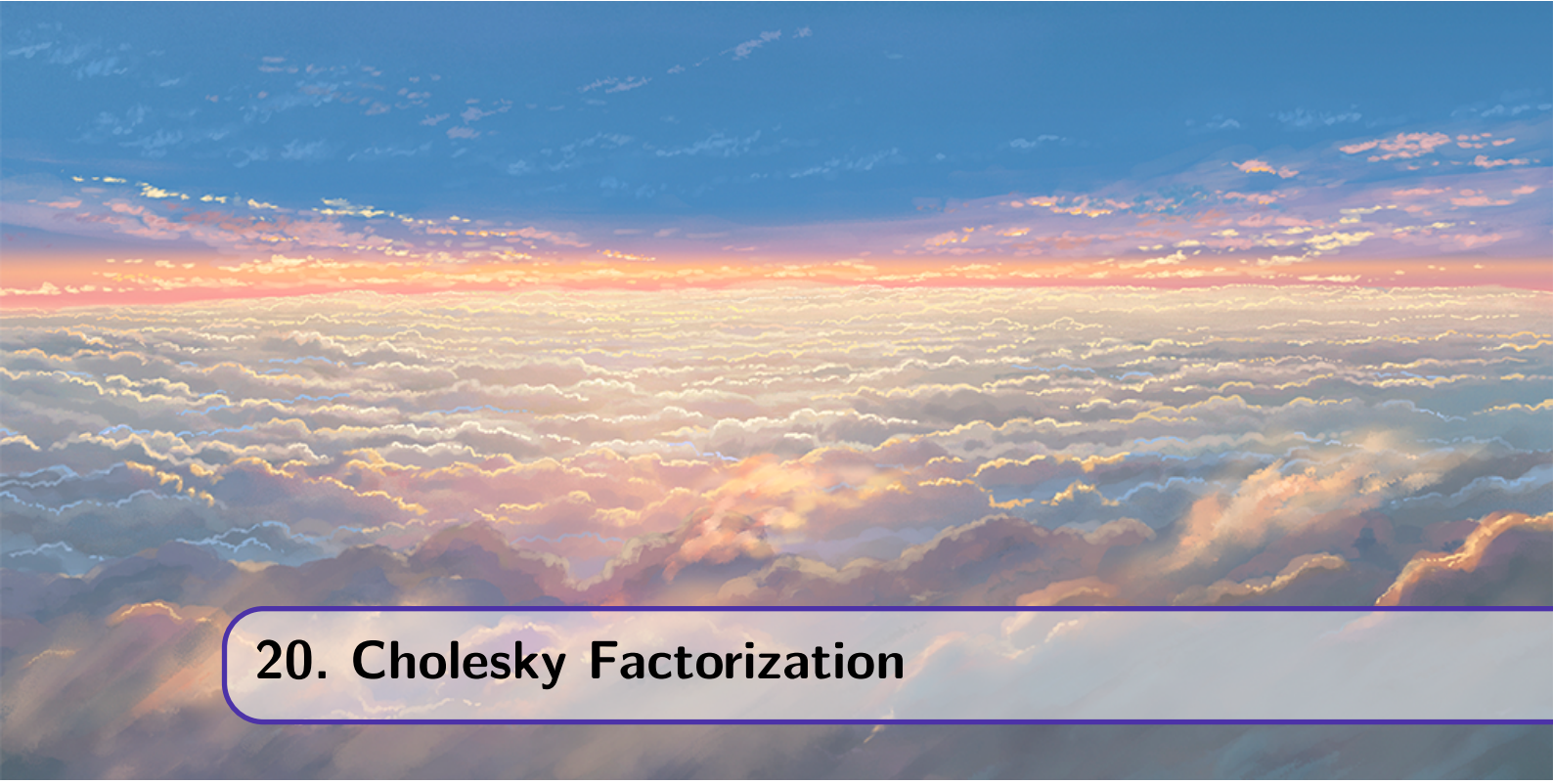
$$A = QR$$

$$A = X\Lambda X^{-1}$$

$$S = Q\Lambda Q^T$$

$$A = U\Sigma V^T$$

$$A = CMR$$



20. Cholesky Factorization

20.1 Positive Definite Matrices

Definition 20.1.1 — Positive semidefinite. a symmetric matrix $A \in \mathbf{R}^{n \times n}$ is positive semidefinite if $x^T A x \geq 0 \quad$ for all x

Definition 20.1.2 — positive definite. a symmetric matrix $A \in \mathbf{R}^{n \times n}$ is positive definite if $x^T A x > 0 \quad$ for all $x \neq 0$

20.2 Schur complement

Definition 20.2.1 — Schur complement. partition $n \times n$ symmetric matrix A as

$$A = \begin{bmatrix} A_{11} & A_{2:n,1}^T \\ A_{2:n,1} & A_{2:n,2:n} \end{bmatrix}$$

the Schur complement of A_{11} is defined as the $(n-1) \times (n-1)$ matrix

$$S = A_{2:n,2:n} - \frac{1}{A_{11}} A_{2:n,1} A_{2:n,1}^T$$

Theorem 20.2.1 if A is positive definite, then S is positive definite

Proof. Take any $x \neq 0$ and define $y = -\left(A_{2:n,1}^T x\right) / A_{11}$; then

$$x^T S x = \begin{bmatrix} y \\ x \end{bmatrix}^T \begin{bmatrix} A_{11} & A_{2:n,1}^T \\ A_{2:n,1} & A_{2:n,2:n} \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} > 0$$

because A is positive definite ∎

Theorem 20.2.2 Positive definite matrices are nonsingular.

Theorem 20.2.3 If A is positive semidefinite, but not positive definite, then it is singular.

Proof. to see this, suppose A is positive semidefinite but not positive definite
there exists a nonzero x with $x^T Ax = 0$
since A is positive semidefinite the following function is nonnegative:

$$\begin{aligned} f(t) &= (x - tAx)^T A(x - tAx) \\ &= x^T Ax - 2tx^T A^2 x + t^2 x^T A^3 x \\ &= -2t\|Ax\|^2 + t^2 x^T A^3 x \end{aligned}$$

$f(t) \geq 0$ for all t is only possible if $\|Ax\| = 0$, i.e., $Ax = 0$
hence there exists a nonzero x with $Ax = 0$, so A is singular ∎

Corollary 20.2.4 if $A \in \mathbf{R}^{n\times n}$ is positive semidefinite, then

$$B^T AB$$

is positive semidefinite for any $B \in \mathbf{R}^{n\times m}$

Corollary 20.2.5 if $A \in \mathbf{R}^{n\times n}$ is positive definite, then

$$B^T AB$$

is positive definite for any $B \in \mathbf{R}^{n\times m}$ with linearly independent columns

20.3 Examples

20.3.1 Graph Laplacian

Definition 20.3.1 — Laplacian of the Graph. the positive semidefinite matrix $A = BB^T$ is called the Laplacian of the graph

$$A_{ij} = \begin{cases} \text{degree of node } i & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and there is an arc } i \to j \text{ or } j \to i \\ 0 & \text{otherwise} \end{cases}$$

the degree of a node is the number of arcs incident to it

Theorem 20.3.1 if y is vector of node potentials, then $B^T y$ contains potential differences: $\left(B^T y\right)_j = y_k - y_l$ if arc j goes from node l to k

Theorem 20.3.2 — Dirichlet energy function. Define $y^T Ay = y^T BB^T y$ is the sum of

squared potential differences

$$y^T A y = \left\|B^T y\right\|^2 = \sum_{\text{arcs } i \to j} (y_j - y_i)^2$$

This is also known as the Dirichlet energy function.

20.4 Cholesky Factorization

Theorem 20.4.1 every positive definite matrix $A \in \mathbf{R}^{n \times n}$ can be factored as

$$A = R^T R$$

where R is upper triangular with positive diagonal elements, R is called the Cholesky factor of A.

complexity of computing R is $(1/3)n^3$ flops.

It can be interpreted as "square root" of a positive definite matrix and gives a practical method for testing positive definiteness.

Algorithm 29: Cholesky factorization of order $n - 1$

Input:

$$\begin{bmatrix} A_{11} & A_{1,2:n} \\ A_{2:n,1} & A_{2:n,2:n} \end{bmatrix} = \begin{bmatrix} R_{11} & 0 \\ R_{1,2:n}^T & R_{2:n,2:n}^T \end{bmatrix} \begin{bmatrix} R_{11} & R_{1,2:n} \\ 0 & R_{2:n,2:n} \end{bmatrix}$$
$$= \begin{bmatrix} R_{11}^2 & R_{11} R_{1,2:n} \\ R_{11} R_{1,2:n}^T & R_{1,2:n}^T R_{1,2:n} + R_{2:n,2:n}^T R_{2:n,2:n} \end{bmatrix}$$

1 compute first row of R :

$$R_{11} = \sqrt{A_{11}}, \quad R_{1,2:n} = \frac{1}{R_{11}} A_{1,2:n}$$

2 compute 2,2 block $R_{2:n,2:n}$ from

$$A_{2:n,2:n} - R_{1,2:n}^T R_{1,2:n} = R_{2:n,2:n}^T R_{2:n,2:n}$$

Proposition 20.4.2 the algorithm works for positive definite A of size $n \times n$

Proof. if A is positive definite then $A_{11} > 0$.

if A is positive definite, then

$$A_{2:n,2:n} - R_{1,2:n}^T R_{1,2:n} = A_{2:n,2:n} - \frac{1}{A_{11}} A_{2:n,1} A_{2:n,1}^T$$

is positive definite.

Hence the algorithm works for $n = m$ if it works for $n = m - 1$.

It obviously works for $n = 1$; therefore it works for all n. ∎

20.5 Examples for Cholesky Factorization

■ **Example 20.1**

$$\begin{bmatrix} 25 & 15 & -5 \\ 15 & 18 & 0 \\ -5 & 0 & 11 \end{bmatrix} = \begin{bmatrix} R_{11} & 0 & 0 \\ R_{12} & R_{22} & 0 \\ R_{13} & R_{23} & R_{33} \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ 0 & R_{22} & R_{23} \\ 0 & 0 & R_{33} \end{bmatrix}$$

$$= \begin{bmatrix} 5 & 0 & 0 \\ 3 & 3 & 0 \\ -1 & 1 & 3 \end{bmatrix} \begin{bmatrix} 5 & 3 & -1 \\ 0 & 3 & 1 \\ 0 & 0 & 3 \end{bmatrix}$$

$$\begin{bmatrix} 25 & 15 & -5 \\ 15 & 18 & 0 \\ -5 & 0 & 11 \end{bmatrix} = \begin{bmatrix} R_{11} & 0 & 0 \\ R_{12} & R_{22} & 0 \\ R_{13} & R_{23} & R_{33} \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ 0 & R_{22} & R_{23} \\ 0 & 0 & R_{33} \end{bmatrix}$$

first row of R

$$\begin{bmatrix} 25 & 15 & -5 \\ 15 & 18 & 0 \\ -5 & 0 & 11 \end{bmatrix} = \begin{bmatrix} 5 & 0 & 0 \\ 3 & R_{22} & 0 \\ -1 & R_{23} & R_{33} \end{bmatrix} \begin{bmatrix} 5 & 3 & -1 \\ 0 & R_{22} & R_{23} \\ 0 & 0 & R_{33} \end{bmatrix}$$

second row of R

$$\begin{bmatrix} 18 & 0 \\ 0 & 11 \end{bmatrix} - \begin{bmatrix} 3 \\ -1 \end{bmatrix} \begin{bmatrix} 3 & -1 \end{bmatrix} = \begin{bmatrix} R_{22} & 0 \\ R_{23} & R_{33} \end{bmatrix} \begin{bmatrix} R_{22} & R_{23} \\ 0 & R_{33} \end{bmatrix}$$

$$\begin{bmatrix} 9 & 3 \\ 3 & 10 \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 1 & R_{33} \end{bmatrix} \begin{bmatrix} 3 & 1 \\ 0 & R_{33} \end{bmatrix}$$

third column of R : $10 - 1 = R_{33}^2, i.e., R_{33} = 3$ ■

20.6 Solving equations with positive definite A

Problem 20.1 solve $Ax = b$ with A a positive definite $n \times n$ matrix ■

Algorithm 30: Solving equations with positive definite A
1 factor A as $A = R^T R$
2 solve $R^T Rx = b$
3 solve $R^T y = b$ by forward substitution
4 solve $Rx = y$ by back substitution

Complexity: $(1/3)n^3 + 2n^2 \approx (1/3)n^3$ flops

- factorization: $(1/3)n^3$
- forward and backward substitution: $2n^2$

20.7 Cholesky Factorization of Gram Matrix

Suppose B is an $m \times n$ matrix with linearly independent colum, and the Gram matrix $A = B^T B$ is positive definite.

There two methods for computing the Cholesky factor of A, given B

Algorithm 31: computing the Cholesky factor of A

Output: matrix R (the Cholesky factor of A)

1 compute $A = B^T B$

2 compute Cholesky factorization of A

$$A = R^T R$$

Algorithm 32: computing the Cholesky factor of A

Output: matrix R (the Cholesky factor of A)

1 compute QR factorization $B = QR$; since

$$A = B^T B = R^T Q^T Q R = R^T R$$

■ **Example 20.2**

$$B = \begin{bmatrix} 3 & -6 \\ 4 & -8 \\ 0 & 1 \end{bmatrix}, \quad A = B^T B = \begin{bmatrix} 25 & -50 \\ -50 & 101 \end{bmatrix}$$

1. Cholesky factorization:

$$A = \begin{bmatrix} 5 & 0 \\ -10 & 1 \end{bmatrix} \begin{bmatrix} 5 & -10 \\ 0 & 1 \end{bmatrix}$$

2. QR factorization

$$B = \begin{bmatrix} 3 & -6 \\ 4 & -8 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 3/5 & 0 \\ 4/5 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & -10 \\ 0 & 1 \end{bmatrix}$$

■

20.7.1 Comparison of the two methods

Numerical stability: QR factorization method is more stable

- QR method computes R without "squaring" B (i.e., forming $B^T B$)
- this is important when the columns of B are "almost" linearly dependent

Complexity:

- method 1: cost of symmetric product $B^T B$ plus Cholesky factorization

$$mn^2 + (1/3)n^3 \text{ flops}$$

- method 2: $2mn^2$ flops for QR factorization
- method 1 is faster but only by a factor of at most two (if $m \gg n$)

20.8 Sparse positive definite matrices

Cholesky factorization of dense matrices:

- $(1/3)n^3$ flops
- on a standard computer: a few seconds or less, for n up to several 1000

Cholesky factorization of sparse matrices:

- if A is very sparse, R is often (but not always) sparse
- if R is sparse, the cost of the factorization is much less than $(1/3)\,n^3$
- exact cost depends on n, number of nonzero elements, sparsity pattern
- very large sets of equations can be solved by exploiting sparsity

20.9 Sparse Cholesky factorization

Theorem 20.9.1 if A is sparse and positive definite, it is usually factored as

$$A = PR^T RP^T$$

P a permutation matrix; R upper triangular with positive diagonal elements

Interpretation: we permute the rows and columns of A and factor

$$P^T AP = R^T R$$

- choice of permutation greatly affects the sparsity R
- there exist several heuristic methods for choosing a good permutation

20.10 Solving sparse positive definite equations

solve $Ax = b$ with A a sparse positive definite matrix

Algorithm 33: Solving sparse positive definite equations

1. compute sparse Cholesky factorization $A = PR^T RP^T$
2. permute right-hand side: $c := P^T b$
3. solve $R^T y = c$ by forward substitution
4. solve $Rz = y$ by back substitution
5. permute solution: $x := Pz$

20.11 Quadratic form

Theorem 20.11.1 suppose A is $n \times n$ and Hermitian $\left(A_{ij} = \bar{A}_{ji}\right)$

$$\begin{aligned} x^H Ax &= \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} \bar{x}_i x_j \\ &= \sum_{i=1}^{n} A_{ii} \left|x_i\right|^2 + \sum_{i>j} \left(A_{ij} \bar{x}_i x_j + \bar{A}_{ij} x_i \bar{x}_j\right) \\ &= \sum_{i=1}^{n} A_{ii} \left|x_i\right|^2 + 2 \operatorname{Re} \sum_{i>j} A_{ij} \bar{x}_i x_j \end{aligned}$$

note that $x^H Ax$ is real for all $x \in \mathbf{C}^n$

20.12 Complex positive definite matrices

Definition 20.12.1 a Hermitian $n \times n$ matrix A is positive semidefinite if

$$x^H Ax \geq 0 \quad \text{for all } x \in \mathbf{C}^n$$

Definition 20.12.2 Hermitian $n \times n$ matrix A is positive definite if

$$x^H Ax > 0 \text{ for all nonzero } x \in \mathbf{C}^n$$

Theorem 20.12.1 — Cholesky factorization of complex matrices. every positive definite matrix $A \in \mathbf{C}^{n \times n}$ can be factored as

$$A = R^H R$$

where R is upper triangular with positive real diagonal elements

20.13 Regularized least squares model fitting

We revisit the data fitting problem with linear-in-parameters model

Problem 20.2 — Data fitting problem (Regularized least squares model fitting).

$$\text{minimize} \sum_{k=1}^{N} \left(\theta^T F\left(x^{(k)}\right) - y^{(k)}\right)^2 + \lambda \sum_{j=1}^{p} \theta_j^2$$

where

$$\begin{aligned} \hat{f}(x) &= \theta_1 f_1(x) + \theta_2 f_2(x) + \cdots + \theta_p f_p(x) \\ &= \theta^T F(x) \end{aligned}$$

$F(x) = (f_1(x), \ldots, f_p(x))$ is a p-vector of basis functions, $f_1(x), \ldots, f_p(x)$, $(x^{(1)}, y^{(1)})$, $\ldots$,$(x^{(N)}, y^{(N)})$ are N examples. ∎

To simplify notation, we add regularization for all coefficients $\theta_1, \ldots, \theta_p$

Next discussion can be modified to handle $f_1(x) = 1$, regularization $\sum_{j=2}^{p} \theta_j^2$.

Problem 20.3

$$\text{minimize} \quad \|A\theta - b\|^2 + \lambda \|\theta\|^2$$

∎

A has size $N \times p$ (number of examples $\times$ number of basis functions)

$$A = \begin{bmatrix} F\left(x^{(1)}\right)^T \\ F\left(x^{(2)}\right)^T \\ \vdots \\ F\left(x^{(N)}\right)^T \end{bmatrix} = \begin{bmatrix} f_1\left(x^{(1)}\right) & f_2\left(x^{(1)}\right) & \cdots & f_p\left(x^{(1)}\right) \\ f_1\left(x^{(2)}\right) & f_2\left(x^{(2)}\right) & \cdots & f_p\left(x^{(2)}\right) \\ \vdots & \vdots & & \vdots \\ f_1\left(x^{(N)}\right) & f_2\left(x^{(N)}\right) & \cdots & f_p\left(x^{(N)}\right) \end{bmatrix}$$

,b is the N-vector $b = \left(y^{(1)}, \ldots, y^{(N)}\right)$

We discuss methods for problems with $N \ll p$ (A is very wide).

Theorem 20.13.1 The equivalent "stacked" least squares problem has size $(p+N)\times p$

QR factorization method may be too expensive when $N \ll p$.

Theorem 20.13.2 from the normal equations:

$$\hat{\theta} = \left(A^T A + \lambda I\right)^{-1} A^T b = A^T \left(AA^T + \lambda I\right)^{-1} b$$

second expression follows from the "push-through" identity

$$\left(A^T A + \lambda I\right)^{-1} A^T = A^T \left(AA^T + \lambda I\right)^{-1}$$

Proof. this is easily proved, by writing it as $A^T \left(AA^T + \lambda I\right) = \left(A^T A + \lambda I\right) A^T$.
from the second expression for $\hat{\theta}$ and the definition of A,

$$\hat{f}(x) = \hat{\theta}^T F(x) = w^T A F(x) = \sum_{i=1}^{N} w_i F\left(x^{(i)}\right)^T F(x)$$

where $w = \left(AA^T + \lambda I\right)^{-1} b$. ■

Algorithm 34: Regularized least squares model fitting

1. compute the $N \times N$ matrix $Q = AA^T$, which has elements

$$Q_{ij} = F\left(x^{(i)}\right)^T F\left(x^{(j)}\right), \quad i, j = 1, \ldots, N$$

2. use a Cholesky factorization to solve the equation

$$(Q + \lambda I)w = b$$

$\hat{\theta} = A^T w$ is not needed; w is sufficient to evaluate the function $\hat{f}(x)$:

$$\hat{f}(x) = \sum_{i=1}^{N} w_i F\left(x^{(i)}\right)^T F(x)$$

complexity: $(1/3)\, N^3$ flops for factorization plus cost of computing Q.

20.13.1 Example: multivariate polynomials

Definition 20.13.1 $\hat{f}(x)$ is a polynomial of degree d (or less) in n variables $x = (x_1, \ldots, x_n)$

$\hat{f}(x)$ is a linear combination of all possible monomials

$$x_1^{k_1} x_2^{k_2} \cdots x_n^{k_n}$$

where $k_1, \ldots, k_n$ are nonnegative integers with $k_1 + k_2 + \cdots + k_n \leq d$

Theorem 20.13.3 number of different monomials is

$$\left(\begin{array}{c} n+d \\ n \end{array} \right) = \frac{(n+d)!}{n!d!}$$

▪ **Example 20.3** for $n = 2, d = 3$ there are ten monomials

$$1, x_1, x_2, x_1^2, x_1 x_2, x_2^2, x_1^3, x_1^2 x_2, x_1 x_2^2, x_2^3$$

▪

20.14 Multinomial formula

Theorem 20.14.1

$$(x_0 + x_1 + \cdots + x_n)^d = \sum_{k_0 + \cdots + k_n = d} \frac{(d+1)!}{k_0! k_1! \cdots k_n!} x_0^{k_0} x_1^{k_1} \cdots x_n^{k_n}$$

sum is over all nonnegative integers $k_0, k_1, \ldots, k_n$ with sum d

Corollary 20.14.2 setting $x_0 = 1$ gives

$$(1 + x_1 + x_2 + \cdots + x_n)^d = \sum_{k_1 + \cdots + k_n \leq d} c_{k_1 k_2 \cdots k_n} x_1^{k_1} x_2^{k_2} \cdots x_n^{k_n}$$

the sum includes all monomials of degree d or less with variables $x_1, \ldots, x_n$

Definition 20.14.1 — coefficient $c_{k_1 k_2 \cdots k_n}$. coefficient $c_{k_1 k_2 \cdots k_n}$ is defined as

$$c_{k_1 k_2 \cdots k_n} = \frac{(d+1)!}{k_0! k_1! k_2! \cdots k_n!} \quad \text{with} \quad k_0 = d - k_1 - \cdots - k_n$$

20.15 Vector of monomials

Definition 20.15.1 write polynomial of degree d or less, with variables $x \in \mathbf{R}^n$, as

$$\hat{f}(x) = \theta^T F(x)$$

$F(x)$ is vector of basis functions

$$\sqrt{c_{k_1 \cdots k_n}} x_1^{k_1} x_2^{k_2} \cdots x_n^{k_n} \quad \text{for all } k_1 + k_2 + \cdots + k_n \leq d$$

Theorem 20.15.1 length of $F(x)$ is $p = (n+d)!/(n!d!)$

Theorem 20.15.2 multinomial formula gives simple formula for inner products

$$F(u)^T F(v) = \sum_{k_1+\cdots+k_n \leq d} c_{k_1 k_2 \cdots k_n} \left(u_1^{k_1} \cdots u_n^{k_n}\right) \left(v_1^{k_1} \cdots v_n^{k_n}\right)$$
$$= (1 + u_1 v_1 + \cdots + u_n v_n)^d$$

Only $2n + 1$ flops needed for inner product of length $p = (n + d)!/(n!d!)$.

■ **Example 20.4** vector of monomials of degree $d = 3$ or less in $n = 2$ variables

$$F(u)^T F(v) = \begin{bmatrix} 1 \\ \sqrt{3}u_1 \\ \sqrt{3}u_2 \\ \sqrt{3}u_1^2 \\ \sqrt{6}u_1 u_2 \\ \sqrt{3}u_2^2 \\ u_1^3 \\ \sqrt{3}u_1^2 u_2 \\ \sqrt{3}u_1 u_2^2 \\ u_2^3 \end{bmatrix}^T \begin{bmatrix} 1 \\ \sqrt{3}v_1 \\ \sqrt{3}v_2 \\ \sqrt{3}v_1^2 \\ \sqrt{6}v_1 v_2 \\ \sqrt{3}v_2^2 \\ v_1^3 \\ \sqrt{3}v_1^2 v_2 \\ \sqrt{3}v_1 v_2^2 \\ v_2^3 \end{bmatrix} = (1 + u_1 v_1 + u_2 v_2)^3$$

■

20.16 Least squares fitting of multivariate polynomials

Problem 20.4 fit polynomial of n variables, degree $\leq d$, to points $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(N)}, y^{(N)}\right)$

■

Algorithm 35: Least squares fitting of multivariate polynomials

1 compute the $N \times N$ matrix Q with elements

$$Q_{ij} = K\left(x^{(i)}, x^{(j)}\right) \text{ where } K(u, v) = \left(1 + u^T v\right)^d$$

2 use a Cholesky factorization to solve the equation $(Q + \lambda I)w = b$

the fitted polynomial is

$$\hat{f}(x) = \sum_{i=1}^{N} w_i K\left(x^{(i)}, x\right) = \sum_{i=1}^{N} w_i \left(1 + \left(x^{(i)}\right)^T x\right)^d$$

complexity: nN^2 flops for computing Q, plus $(1/3)N^3$ for the factorization, i.e.,

$$nN^2 + (1/3)N^3 \text{ flops}$$

20.17 Kernel methods

Definition 20.17.1 — Kernel function. a generalized inner product $K(u,v)$.

$K(u,v)$ is inner product of vectors of basis functions $F(u)$ and $F(v)$, $F(u)$ may be infinite-dimensional.

Kernel methods work with $K(u,v)$ directly, do not require $F(u)$.

■ **Example 20.5** the polynomial kernel function $K(u,v) = \left(1 + u^T v\right)^d$ ■

■ **Example 20.6** the Gaussian radial basis function kernel

$$K(u,v) = \exp\left(-\frac{\|u-v\|^2}{2\sigma^2}\right)$$

■

■ **Example 20.7** kernels exist for computing with graphs, texts, strings of symbols, ... ■

we apply the method to least squares classification:

- training set is 10000 images from MNIST data set ($\approx$ 1000 examples per digit)
- vector x is vector of pixel intensities (size $n = 28^2 = 784$)
- we use the polynomial kernel with degree $d = 3$:

$$K(u,v) = \left(1 + u^T v\right)^3$$

hence $F(z)$ has length $p = (n+d)!/(n!d!) = 80931145$

- we calculate ten Boolean classifiers

$$\hat{f}_k(x) = \text{sign}\left(\tilde{f}_k(x)\right), \quad k = 1, \ldots 10$$

$\hat{f}_k(x)$ distinguishes digit $k-1$ (outcome +1) form other digits (outcome -1)

the Boolean classifiers are combined in the multi-class classifier

$$\hat{f}(x) = \underset{k=1,\ldots,10}{\text{argmax}} \tilde{f}_k(x)$$

Algorithm 36: compute Boolean classifier for digit $k-1$ versus the rest

1 compute $N \times N$ matrix Q with elements

$$Q_{ij} = \left(1 + \left(x^{(i)}\right)^T x^{(j)}\right)^d, \quad i,j = 1, \ldots, N$$

2 define N-vector $b = \left(y^{(1)}, \ldots, y^{(N)}\right)$ with elements

$$y^{(i)} = \begin{cases} +1 & x^{(i)} \text{ is an example of digit } k-1 \\ -1 & \text{otherwise} \end{cases}$$

3 solve the equation $(Q + \lambda I)w = b$

the solution w gives the Boolean classifier for digit $k-1$ versus rest

$$\tilde{f}_k(x) = \sum_{i=1}^{N} w_i \left(1 + \left(x^{(i)}\right)^T x\right)^d$$

The matrix Q is the same for each of the ten Boolean classifiers
Hence, only the right-hand side of the equation

$$(Q + \lambda I)w = y^{\mathrm{d}}$$

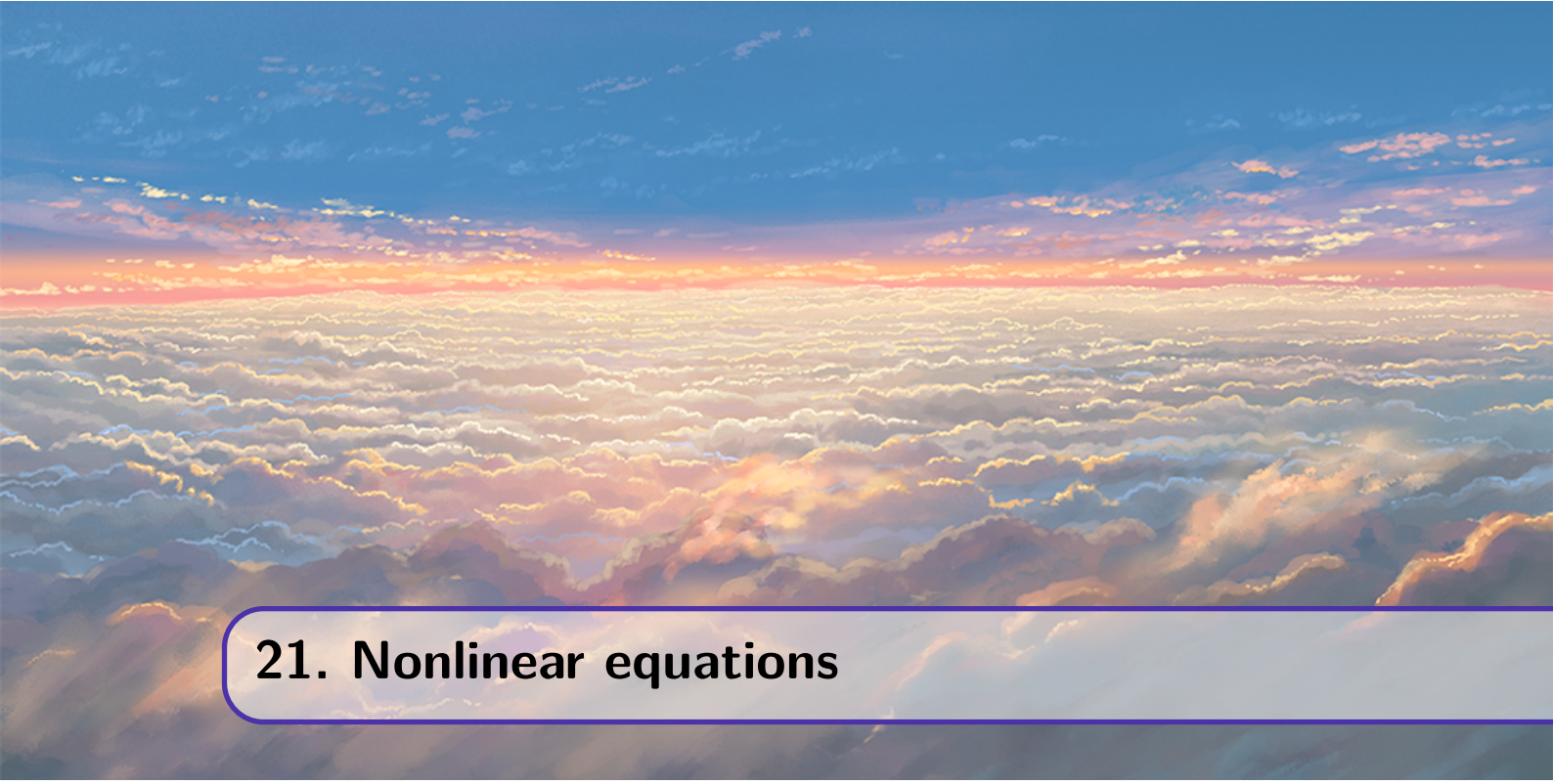
is different for each Boolean classifier.

20.17.1 Complexity

Complexity:

- constructing Q requires $N^2/2$ inner products of length n : nN^2 flops
- Cholesky factorization of $Q + \lambda I$: $(1/3)N^3$ flops
- solve the equation $(Q + \lambda I)w = y^{\mathrm{d}}$ for the 10 right-hand sides: $20N^2$ flops

Total complexity is $(1/3)N^3 + nN^2$.



21. Nonlinear equations

In this chapter we discuss methods for finding a solution of n nonlinear equations in n variables

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0 \\ f_2(x_1, x_2, \ldots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, x_2, \ldots, x_n) &= 0 \end{aligned}$$

To simplify notation, we will often express this as

$$f(x) = 0$$

where

$$f(x) = \begin{bmatrix} f_1(x_1, x_2, \ldots, x_n) \\ f_2(x_1, x_2, \ldots, x_n) \\ \vdots \\ f_n(x_1, x_2, \ldots, x_n) \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

We assume that f is at least continuous (i.e., $\lim_{y \to x} f(y) = f(x)$ for all x), and for some algorithms, stronger assumptions will be needed (for example, differentiability).

21.1 Bisection method

The first method we discuss only applies to problems with $n = 1$.

We start with an interval $[l, u]$ that satisfies $f(l)f(u) < 0$ (the function values at the end points of the interval have opposite signs). Since f is continuous, this guarantees that the interval contains at least one solution of $f(x) = 0$. In each iteration we evaluate f at the midpoint $(l + u)/2$ of the interval, and depending on the sign of $f((l + u)/2)$, replace l or u with $(l + u)/2$. If $f((u + l)/2)$ has the same sign as $f(l)$, we replace l with $(u + l)/2$. Otherwise we replace u. Thus we obtain a new interval

that still satisfies $f(l)f(u) < 0$. The method is called bisection because the interval is replaced by either its left or right half at each iteration.

Algorithm 4.1. Bisection ALGORITHM. given l, u with $l < u$ and $f(l)f(u) < 0$; a required tolerance $\epsilon > 0$ repeat 1. $x := (l+u)/2$. 2. Compute $f(x)$. 3. if $f(x) = 0$, return x. 4. if $f(x)f(l) < 0$, $u := x$, else, $l := x$. until $u - l \leq \epsilon$

The convergence of the bisection method is easy to analyze. If we denote by $\left[l^{(0)}, u^{(0)}\right]$ the initial interval, and by $\left[l^{(k)}, u^{(k)}\right]$ the interval after iteration k, then

$$u^{(k)} - l^{(k)} = \frac{u^{(0)} - l^{(0)}}{2^k},$$

because the length of the interval is divided by two at each iteration. This means that the exit condition $u^{(k)} - l^{(k)} \leq \epsilon$ will be satisfied if

$$\log_2 \left(\frac{u^{(0)} - l^{(0)}}{2^k} \right) = \log_2 \left(u^{(0)} - l^{(0)} \right) - k \leq \log_2 \epsilon,$$

i.e., as soon as $k \geq \log_2 \left(\left(u^{(0)} - l^{(0)} \right) / \epsilon \right)$. The algorithm therefore terminates after

$$\left\lceil \log_2 \left(\frac{u^{(0)} - l^{(0)}}{\epsilon} \right) \right\rceil$$

iterations. (By $\lceil \alpha \rceil$ we mean the smallest integer greater than or equal to α.) Since the final interval contains at least one solution $x^\star$, we are guaranteed that its midpoint

$$x^{(k)} = \frac{1}{2} \left(l^{(k)} + u^{(k)} \right)$$

is no more than a distance $u^{(k)} - l^{(k)}$ from $x^\star$. Thus we have

$$\left| x^{(k)} - x^\star \right| \leq \epsilon,$$

when the algorithm terminates.

The advantages of the bisection method are its simplicity and the fact that it does not require derivatives. It also does not require a starting point close to $x^\star$. The disadvantages are that it is not very fast, and that it does not extend to $n > 1$. Selecting an initial interval that satisfies $f(l)f(u) < 0$ may also be difficult.

Convergence rate The bisection method is R-linearly convergent (as defined in section 3.2). After k iterations, the midpoint $x^{(k)} = \left(u^{(k)} + l^{(k)} \right) / 2$ satisfies

$$\left| x^{(k)} - x^\star \right| \leq u^{(k)} - l^{(k)} \leq (1/2)^k \left(u^{(0)} - l^{(0)} \right),$$

(see equation (4.1)). Therefore the definition (3.2) is satisfied with $c = 1/2$ and $M = u^{(0)} - l^{(0)}$.

21.2 Newton's method for one equation with one variable

Newton's method is the most popular method for solving nonlinear equations. We first explain the method for $n = 1$, and then extend it to $n > 1$. We assume that f is differentiable.

Algorithm 4.2. NEWTON'S METHOD FOR ONE EQUATION WITH ONE VARIABLE. given initial x, required tolerance $\epsilon > 0$ repeat 1. Compute $f(x)$ and $f'(x)$. 2. if $|f(x)| \leq \epsilon$, return x. 3. $x := x - f(x)/f'(x)$. until maximum number of iterations is exceeded.

For simplicity we assume that $f'(x) \neq 0$ in step 3 . (In a practical implementation we would have to make sure that the code handles the case $f'(x) = 0$ gracefully.) The algorithm starts at some initial value $x^{(0)}$, and then computes iterates $x^{(k)}$ by repeating

$$x^{(k+1)} = x^{(k)} - \frac{f\left(x^{(k)}\right)}{f'\left(x^{(k)}\right)}, \quad k = 0, 1, 2, \ldots$$

This update has a simple interpretation. After evaluating the function value $f\left(x^{(k)}\right)$ and the derivative $f'\left(x^{(k)}\right)$, we construct the first-order Taylor approximation to f around $x^{(k)}$:

$$\hat{f}(y) = f\left(x^{(k)}\right) + f'\left(x^{(k)}\right)\left(y - x^{(k)}\right)$$

We then solve $\hat{f}(y) = 0$, i.e.,

$$f\left(x^{(k)}\right) + f'\left(x^{(k)}\right)\left(y - x^{(k)}\right) = 0$$

for the variable y. This is called the linearized equation. If $f'\left(x^{(k)}\right) \neq 0$, the solution exists and is given by

$$y = x^{(k)} - \frac{f\left(x^{(k)}\right)}{f'\left(x^{(k)}\right)}.$$

We then take y as the next value $x^{(k+1)}$. This is illustrated in figure 4.1.

▪ **Example 21.1** Examples We first consider the nonlinear equation

$$f(x) = e^x - e^{-x} - 1 = 0.$$

The derivative is $f'(x) = e^x + e^{-x}$, so the Newton iteration is

$$x^{(k+1)} = x^{(k)} - \frac{e^{x^{(k)}} - e^{-x^{(k)}} - 1}{e^{x^{(k)}} + e^{-x^{(k)}}}, \quad k = 0, 1, \ldots$$

If we start at $x = 4$, the algorithm converges very quickly to $x^\star = 0.4812$ (see figure 4.2). ▪

▪ **Example 21.2** As a second example, we consider the equation

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 0$$

The derivative is $f'(x) = 4/(e^x + e^{-x})^2$, so the Newton iteration is

$$x^{(k+1)} = x^{(k)} - \frac{1}{4}\left(e^{2x^{(k)}} - e^{-2x^{(k)}}\right), \quad k = 0, 1, \ldots$$

Figure 4.3 shows the iteration, starting at two starting points, $x^{(0)} = 0.85$, and $x^{(0)} = 1.15$. The method converges rapidly from $x^{(0)} = 0.85$, but does not converge from $x^{(0)} = 1.15$.

The two examples are typical for the convergence behavior of Newton's method: it works very well if started near a solution; it may not work when started far from a solution. ∎

21.3 Newton's method for sets of nonlinear equations

We now extend Newton's method to a nonlinear equation $f(x) = 0$ where $f : \mathbf{R}^n \to \mathbf{R}^n$ (a function that maps an n-vector x to an n-vector $f(x)$).

We start with an initial point $x^{(0)}$. At iteration k we evaluate $f\left(x^{(k)}\right)$, the derivative matrix $Df\left(x^{(k)}\right)$, and form the first-order approximation of f at $x^{(k)}$:

$$\hat{f}(y) = f\left(x^{(k)}\right) + Df\left(x^{(k)}\right)\left(y - x^{(k)}\right)$$

We set $\hat{f}(y) = 0$ and solve for y, which gives

$$y = x^{(k)} - Df\left(x^{(k)}\right)^{-1} f\left(x^{(k)}\right)$$

(assuming $Df\left(x^{(k)}\right)$ is nonsingular). This value is taken as the next iterate $x^{(k+1)}$. In summary,

$$x^{(k+1)} = x^{(k)} - Df\left(x^{(k)}\right)^{-1} f\left(x^{(k)}\right), \quad k = 0, 1, 2, \ldots$$

1 Algorithm 4.3. NEWTON'S METHOD FOR SETS OF NONLINEAR EQUATIONS. given an initial x, a required tolerance $\epsilon > 0$ repeat 1. Evaluate $g = f(x)$ and $H = Df(x)$. 2 . if $\|g\| \leq \epsilon$, return x. 3. Solve $Hv = -g$. 4. $x := x + v$. until maximum number of iterations is exceeded.

∎ **Example 21.3** Example As an example, we take a problem with two variables

$$f_1(x_1, x_2) = \log\left(x_1^2 + 2x_2^2 + 1\right) - 0.5, \quad f_2(x_1, x_2) = -x_1^2 + x_2 + 0.2.$$

There are two solutions, $(0.70, 0.29)$ and $(-0.70, 0.29)$. The derivative matrix is

$$Df(x) = \begin{bmatrix} 2x_1/\left(x_1^2 + 2x_2^2 + 1\right) & 4x_2/\left(x_1^2 + 2x_2^2 + 1\right) \\ -2x_1 & 1 \end{bmatrix}$$

Figure 4.4 shows what happens if we use three different starting points. The example confirms the behavior we observed for problems with one variable. Newton's method does not always work, but if started near a solution, it takes only a few iterations. The main advantage of the method is its very fast local convergence. The disadvantages are that it requires a good starting point, and that it requires the n^2 partial derivatives of f.

Many techniques have been proposed to improve the convergence properties. (A simple idea for $n = 1$ is to combine it with the bisection method.) These globally convergent Newton methods are designed in such a way that locally, in the neighborhood of a solution, they automatically switch to the standard Newton method.

A variation on Newton's method that does not require derivatives is the secant method, discussed in the next paragraph. ∎

21.4 Secant method

Although the idea of the secant method extends to problems with several variables, we will describe it only for $n = 1$.

The secant method can be interpreted as a variation of Newton's method in which we replace the first-order approximation

$$\hat{f}(y) = f\left(x^{(k)}\right) + f'\left(x^{(k)}\right)\left(y - x^{(k)}\right)$$

with the function

$$\hat{f}(y) = f\left(x^{(k)}\right) + \frac{f\left(x^{(k)}\right) - f\left(x^{(k-1)}\right)}{x^{(k)} - x^{(k-1)}}\left(y - x^{(k)}\right).$$

This is the affine function that agrees with $f(y)$ at $y = x^{(k)}$ and $y = x^{(k-1)}$. We then set $\hat{f}(y)$ equal to zero, solve for y, and take the solution as $x^{(k+1)}$ (figure 4.5).

1 Algorithm 4.4. SECANT METHOD FOR ONE EQUATION WITH ONE VARIABLE. given two initial points x, x^-, required tolerance $\epsilon > 0$ repeat 1. Compute $f(x)$ 2. if $|f(x)| \leq \epsilon$, return x. 3. $g := (f(x) - f(x^-))/(x - x^-)$. 4. $x^- := x$. 5. $x := x - f(x)/g$. until maximum number of iterations is exceeded.

21.5 Convergence analysis of Newton's method

Newton's method converges quadratically if $f'(x^\star) \neq 0$, and we start sufficiently close to $x^\star$. More precisely we can state the following result.

Suppose $I = [x^\star - \delta, x^\star + \delta]$ is an interval around a solution $x^\star$ on which f satisfies the following two properties. 1. There exists a constant $m > 0$ such that $|f'(x)| \geq m$ for all $x \in I$. 2. There exists a constant $L > 0$ such that $|f'(x) - f'(y) \leq L|x - y|$ for all $x, y \in I$. We will show that if $\left|x^{(k)} - x^\star\right| \leq \delta$, then

$$\left|x^{(k+1)} - x^\star\right| \leq \frac{L}{2m}\left|x^{(k)} - x^\star\right|^2.$$

In other words, the inequality (3.3) is satisfied with $c = L/(2m)$, so if $x^{(k)}$ converges to $x^\star$, it converges quadratically.

The inequality (4.3) is proved as follows. For simplicity we will denote $x^{(k)}$ as x and $x^{(k+1)}$ as x^+, i.e.,

$$x^+ = x - \frac{f(x)}{f'(x)}$$

We have

$$\begin{aligned}\left|x^+ - x^\star\right| &= \left|x - \frac{f(x)}{f'(x)} - x^\star\right| \\ &= \frac{\left|-f(x) - f'(x)\left(x^\star - x\right)\right|}{|f'(x)|} \\ &= \frac{\left|f\left(x^\star\right) - f(x) - f'(x)\left(x^\star - x\right)\right|}{|f'(x)|}\end{aligned}$$

(Recall that $f(x^\star) = 0$.) We have $|f'(x)| \geq m$ by the first property, and hence

$$\left|x^+ - x^\star\right| \leq \frac{|f(x^\star) - f(x) - f'(x)(x^\star - x)|}{m}.$$

We can use the second property to bound the numerator:

$$\begin{aligned}
\left|f(x^\star) - f(x) - f'(x)(x^\star - x)\right| &= \left|\int_x^{x^\star} (f'(u) - f'(x))\, du\right| \\
&\leq \int_x^{x^\star} |f'(u) - f'(x)|\, du \\
&\leq L \int_x^{x^\star} |u - x| du \\
&= \frac{L}{2}\left|x^\star - x\right|^2
\end{aligned}$$

Putting (4.4) and (4.5) together, we obtain $|x^+ - x| \leq L|x^\star - x|^2/(2m)$, which proves (4.3).

The inequality (4.3) by itself does not guarantee that $\left|x^{(k)} - x^\star\right| \to 0$. However, if we assume that $\left|x^{(0)} - x^\star\right| \leq \delta$ and that $\delta \leq m/L$, then convergence readily follows. Since $\left|x^{(0)} - x^\star\right| \leq \delta$, we can apply the inequality (4.3) to the first iteration, which yields

$$\left|x^{(1)} - x^\star\right| \leq \frac{L}{2m}\left|x^{(0)} - x^\star\right|^2 \leq \frac{L}{2m}\delta^2 \leq \frac{\delta}{2}.$$

Therefore $\left|x^{(1)} - x^\star\right| \leq \delta$, so we can apply the inequality to $k = 1$, and obtain a bound on the error in $x^{(2)}$,

$$\left|x^{(2)} - x^\star\right| \leq \frac{L}{2m}\left|x^{(1)} - x^\star\right|^2 \leq \frac{L}{2m}\frac{\delta^2}{4} \leq \frac{\delta}{8},$$

and therefore also

$$\left|x^{(3)} - x^\star\right| \leq \frac{L}{2m}\left|x^{(2)} - x^\star\right|^2 \leq \frac{L}{2m}\frac{\delta^2}{8^2} \leq \frac{\delta}{128}$$

et cetera. Continuing in this fashion, we have

$$\left|x^{(k+1)} - x^\star\right| \leq \frac{L}{2m}\left|x^{(k)} - x^\star\right|^2 \leq 2\left(\frac{1}{4}\right)^{2^k}\delta$$

which shows that the error converges to zero very rapidly. A final note on the practical importance of this (and most other) convergence results. If $f'(x^\star) \neq 0$ and $f'(x)$ is a. continuous function, then it is reasonable to assume that the assumptions we made are satisfied for some δ, m, and L. In practice, of course, we almost never know δ, m, L, so the convergence result does not provide any practical guidelines that might help us, for example, when selecting a starting point.

The result does provide some interesting qualitative or conceptual information. It establishes convergence of Newton's method, provided we start sufficiently close to a solution. It also explains the very fast local convergence observed in practice. As an interesting detail that we have not observed so far, the proof suggests that quadratic

convergence only occurs if $f'(x^\star) \neq 0$. A simple example will confirm this. Suppose we apply Newton's method to the nonlinear equation

$$f(x) = x^2 - a = 0$$

where a is a nonnegative number. Newton's method uses the iteration

$$\begin{aligned} x^{(k+1)} &= x^{(k)} - \frac{\left(x^{(k)}\right)^2 - a}{2x^{(k)}} \\ &= \frac{1}{2}\left(x^{(k)} + \frac{a}{x^{(k)}}\right), \quad k = 0, 1, 2, \ldots \end{aligned}$$

Figures 4.7 and 4.8 show the result for $a = 5$ and $a = 0$ respectively, with starting point $x^{(0)} = 5$. Newton's method converges in both cases, but much more slowly when $a = 0$ (and hence $f'(x^\star) = 0$).

Sets of equations The quadratic convergence result for Newton's method generalizes to functions of several variables. The precise statement is as follows. Suppose there is a neighborhood

$$I = \left\{x \mid \left\|x - x^\star\right\| \leq \delta\right\}$$

around a solution $x^\star$ on which f satisfies the following two properties. 1. There exists a constant $m > 0$ such that $\left\|Df(x)^{-1}\right\|_2 \leq 1/m$ for all $x \in I$. 2. There exists a constant $L > 0$ such that $\|Df(x) - Df(y)\|_2 \leq L\|x - y\|$ for all $x, y \in I$. (Note that the norms $\left\|Df(x)^{-1}\right\|_2$ and $\|Df(x) - Df(y)\|_2$ are matrix norms, defined in §6.2, and $\|x - y\|$ is a vector norm.) If $\left\|x^{(k)} - x^\star\right\| \leq \delta$, then

$$\left\|x^{(k+1)} - x^\star\right\| \leq \frac{L}{2m}\left\|x^{(k)} - x^\star\right\|^2.$$

Secant method Under similar assumptions as Newton's method (including, in particular, $f'(x^\star) \neq 0$ and a starting point sufficiently close to $x^\star$), the secant method converges superlinearly. We omit the precise statement and the proof.

22. Unconstrained minimization

22.1 Introduction

Let $g : \mathbf{R}^n \to \mathbf{R}$ be a scalar-valued function of n variables $x = (x_1, x_2, \ldots, x_n)$. We say $x^\star = (x_1^\star, x_2^\star, \ldots, x_n^\star)$ minimizes g if $g(x^\star) \leq g(x)$ for all n-vectors x. We use the notation

$$\text{minimize } g(x)$$

to denote the problem of finding an $x^\star$ that minimizes g. This is called an *un*constrained minimization problem, with variables $x_1, \ldots, x_n$, and with objective function or cost function g.

If $x^\star$ minimizes g, then we say $x^\star$ is a solution of the minimization problem, or a minimum of g. A minimum is also sometimes referred to as a global minimum. A vector $x^\star$ is a local minimum if there exists an $R > 0$ such that $g(x^\star) \leq g(x)$ for all x with $\|x - x^\star\| \leq R$. In other words, there is a neighborhood around $x^\star$ in which $g(x) \geq g(x^\star)$. In this chapter, minimum means global minimum. The word 'global' in 'global minimum' is redundant, but is sometimes added for emphasis.

The greatest α such that $\alpha \leq g(x)$ for all x is called the optimal value of the minimization problem, and denoted $\min g(x)$ or $\min_x g(x)$. If $x^\star$ is a minimum of g, then $g(x^\star) = \min g(x)$, and we say that the optimal value is attained at $x^\star$. It is possible that $\min g(x)$ is finite, but there is no $x^\star$ with $g(x^\star) = \min g(x)$ (see the examples below). In that case the optimal value is not attained. It is also possible that $g(x)$ is unbounded below, in which case we define the optimal value as $\min g(x) = -\infty$.

▪ **Example 22.1** - $g(x) = (x-1)^2$. The optimal value is $\min g(x) = 0$, and is attained at the (global) minimum $x^\star = 1$. There are no other local minima. ▪

▪ **Example 22.2** - $g(x) = e^x + e^{-x} - 3x^2$, shown in the left-hand plot of figure 5.1. The optimal value is -7.02. There are two (global) minima, at $x^\star = \pm 2.84$. There are no other local minima. ▪

▪ **Example 22.3** - $g(x) = e^x + e^{-x} - 3x^2 + x$, shown in figure 5.1 (right). The optimal value is -9.90, attained at the minimum $x^\star = -2.92$. There is another local minimum

at $x = 2.74$. ∎

∎ **Example 22.4** - $g(x) = e^{-x}$. The optimal value is $\min g(x) = 0$, but is not attained. There are no local or global minima. ∎

∎ **Example 22.5** $g(x) = -x + e^{-x}$. This function is unbounded below; the optimal value is $\min g(x) = -\infty$. There are no local or global minima. ∎

22.2 Gradient and Hessian

Definition 22.2.1 Gradient The gradient of a function $g(x)$ of n variables, at $\hat{x}$, is the vector of first partial derivatives evaluated at $\hat{x}$, and is denoted $\nabla g(\hat{x})$:

$$\nabla g(\hat{x}) = \left(\frac{\partial g}{\partial x_1}(\hat{x}), \frac{\partial g}{\partial x_2}(\hat{x}), \ldots, \frac{\partial g}{\partial x_n}(\hat{x}) \right)$$

The gradient is used in the first-order (or affine) approximation of g around $\hat{x}$,

$$\begin{aligned} \hat{g}(x) &= g(\hat{x}) + \sum_{i=1}^{n} \frac{\partial g}{\partial x_i}(\hat{x})\,(x_i - \hat{x}_i) \\ &= g(\hat{x}) + \nabla g(\hat{x})^T (x - \hat{x}) \end{aligned}$$

If $n = 1$, the gradient is simply the first derivative $g'(\hat{x})$, and the first-order approximation reduces to

$$\hat{g}(x) = g(\hat{x}) + g'(\hat{x})(x - \hat{x}).$$

Definition 22.2.2 Hessian The Hessian of a function $g(x)$ of n variables, at $\hat{x}$, is the matrix of second partial derivatives evaluated at $\hat{x}$, and is denoted as $\nabla^2 g(\hat{x})$:

$$\nabla^2 g(\hat{x}) = \begin{bmatrix} \frac{\partial^2 g}{\partial x_1^2}(\hat{x}) & \frac{\partial^2 g}{\partial x_1 \partial x_2}(\hat{x}) & \cdots & \frac{\partial^2 g}{\partial x_1 \partial x_n}(\hat{x}) \\ \frac{\partial^2 g}{\partial x_2 \partial x_1}(\hat{x}) & \frac{\partial^2 g}{\partial x_2^2}(\hat{x}) & \cdots & \frac{\partial^2 g}{\partial x_2 \partial x_n}(\hat{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 g}{\partial x_n \partial x_1}(\hat{x}) & \frac{\partial^2 g}{\partial x_n \partial x_2}(\hat{x}) & \cdots & \frac{\partial^2 g}{\partial x_n^2}(\hat{x}) \end{bmatrix}.$$

This is a symmetric matrix, because

$$\frac{\partial^2 g}{\partial x_i \partial x_j}(\hat{x}) = \frac{\partial^2 g}{\partial x_j \partial x_i}(\hat{x})$$

The Hessian is related to the second-order (or quadratic) approximation of g around $\hat{x}$, which is defined as

$$\begin{aligned} g_q(x) &= g(\hat{x}) + \sum_{i=1}^{n} \frac{\partial g}{\partial x_i}(\hat{x})\,(x_i - \hat{x}_i) + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial^2 g}{\partial x_i \partial x_j}(\hat{x})\,(x_i - \hat{x}_i)\left(x_j - \hat{x}_j\right) \\ &= g(\hat{x}) + \nabla g(\hat{x})^T (x - \hat{x}) + \frac{1}{2}(x - \hat{x})^T \nabla^2 g(\hat{x})(x - \hat{x}) \end{aligned}$$

If $n = 1$, the Hessian is the second derivative $g''(\hat{x})$, and the second-order approxima-

tion reduces to

$$g_{\mathrm{q}}(x) = g(\hat{x}) + g'(\hat{x})(x - \hat{x}) + \frac{1}{2}g''(\hat{x})(x - \hat{x})^2.$$

▪ **Example 22.6** As an example, the Hessian of the function

$$g\,(x_1, x_2) = e^{x_1+x_2-1} + e^{x_1-x_2-1} + e^{-x_1-1}$$

is

$$\nabla^2 g(x) = \begin{bmatrix} e^{x_1+x_2-1} + e^{x_1-x_2-1} + e^{-x_1-1} & e^{x_1+x_2-1} - e^{x_1-x_2-1} \\ e^{x_1+x_2-1} - e^{x_1-x_2-1} & e^{x_1+x_2-1} + e^{x_1-x_2-1} \end{bmatrix}$$

so the second-order approximation around $\hat{x} = 0$ is

$$g_{\mathrm{q}}(x) = \frac{1}{e}\left(3 + x_1 + (3/2)x_1^2 + x_2^2\right).$$

▪

Properties

We list here a few properties that often simplify the task of calculating gradients and Hessians. These facts are straightforward (although sometimes tedious) to verify, directly from the definition of gradient and Hessian. 1. Linear and affine functions. The gradient and Hessian of $g(x) = a^T x + b$ are

$$\nabla g(x) = a, \quad \nabla^2 g(x) = 0.$$

2. Quadratic functions. The gradient and Hessian of $g(x) = x^T P x + q^T x + r$, where P is a symmetric matrix, are

$$\nabla g(x) = 2Px + q, \quad \nabla^2 g(x) = 2P.$$

3. Sum of two functions. If $g(x) = g_1(x) + g_2(x)$, then

$$\nabla g(x) = \nabla g_1(x) + \nabla g_2(x), \quad \nabla^2 g(x) = \nabla^2 g_1(x) + \nabla^2 g_2(x).$$

4. Scalar multiplication. If $g(x) = \alpha f(x)$ where α is a scalar, then

$$\nabla g(x) = \alpha \nabla f(x), \quad \nabla^2 g(x) = \alpha \nabla^2 f(x)$$

5. Composition with affine function. If $g(x) = f(Cx + d)$ where C is an $m \times n$ matrix, d is an m-vector, and $f(y)$ is a function of m variables, then

$$\nabla g(x) = C^T \nabla f(Cx + d), \quad \nabla^2 g(x) = C^T \nabla^2 f(Cx + d) C.$$

Note that $f(Cx + d)$ denotes the function $f(y)$, evaluated at $y = Cx + d$. Similarly, $\nabla f(Cx + d)$ is the gradient $\nabla f(y)$, evaluated at $y = Cx + d$, and $\nabla^2 f(Cx + d)$ is the Hessian $\nabla^2 f(y)$, evaluated at $y = Cx + d$.

▪ **Example 22.7** Examples As a first example, consider the least-squares function

$$g(x) = \|Ax - b\|^2.$$

We can find the gradient and Hessian by expanding g in terms of its variables x_i, and then taking the partial derivatives. An easier derivation is from the properties listed above. We can express g as

$$\begin{aligned} g(x) &= (Ax - b)^T (Ax - b) \\ &= x^T A^T A x - b^T A x - x^T A^T b + b^T b \\ &= x^T A^T A x - 2b^T A x + b^T b. \end{aligned}$$

This shows that g is a quadratic function: $g(x) = x^T Px + q^T x + r$ with $P = A^T A$, $q = -2A^T b, r = b^T b$. From property 2,

$$\nabla g(x) = 2A^T Ax - 2A^T b, \quad \nabla^2 g(x) = 2A^T A.$$

An alternative derivation is based on property 5 . We can express g as $g(x) = f(Cx+d)$ where $C = A, d = -b$, and

$$f(y) = \|y\|^2 = \sum_{i=1}^{m} y_i^2$$

The gradient and Hessian of f are

$$\nabla f(y) = \begin{bmatrix} 2y_1 \\ 2y_2 \\ \vdots \\ 2y_m \end{bmatrix} = 2y, \quad \nabla^2 f(y) = \begin{bmatrix} 2 & 0 & \cdots & 0 \\ 0 & 2 & \cdots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & 2 \end{bmatrix} = 2I$$

Applying property 5 , we find that

$$\begin{aligned} \nabla g(x) &= A^T \nabla f(Ax - b) \\ &= 2A^T (Ax - b) \\ \nabla^2 g(x) &= A^T \nabla^2 f(Ax - b) A \\ &= 2A^T A \end{aligned}$$

the same expressions as we derived before. We can use the same method to find the gradient and Hessian of the function in (5.1),

$$g\,(x_1, x_2) = e^{x_1+x_2-1} + e^{x_1-x_2-1} + e^{-x_1-1}.$$

We can express g as $g(x) = f(Cx + d)$, where $f(y) = e^{y_1} + e^{y_2} + e^{y_3}$, and

$$C = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 0 \end{bmatrix}, \quad d = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

The gradient and Hessian of f are

$$\nabla f(y) = \begin{bmatrix} e^{y_1} \\ e^{y_2} \\ e^{y_3} \end{bmatrix}, \quad \nabla^2 f(y) = \begin{bmatrix} e^{y_1} & 0 & 0 \\ 0 & e^{y_2} & 0 \\ 0 & 0 & e^{y_3} \end{bmatrix}$$

so it follows from property 5 that

$$\nabla g(x) = C^T \nabla f(Cx + d) = \begin{bmatrix} 1 & 1 & -1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} e^{x_1+x_2-1} \\ e^{x_1-x_2-1} \\ e^{-x_1-1} \end{bmatrix}$$

and

$$\begin{aligned} \nabla^2 g(x) &= C^T \nabla^2 f(Cx + d) C \\ &= \begin{bmatrix} 1 & 1 & -1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} e^{x_1+x_2-1} & 0 & \\ 0 & e^{x_1-x_2-1} & 0 \\ 0 & 0 & e^{-x_1-1} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 0 \end{bmatrix} \end{aligned}$$

■

Local optimality It is much harder to characterize optimality if g is not convex (i.e., if there are points where the Hessian is not positive semidefinite). It is not sufficient to set the gradient equal to zero, because such a point might correspond to a local minimum, a local maximum, or a saddle point (see for example, the second function in figure 5.1). However, we can state some simple conditions for local optimality. - Necessary condition. If $x^\star$ is locally optimal, then $\nabla g\left(x^\star\right)=0$ and $\nabla^2 g\left(x^\star\right)$ is positive semidefinite. - Sufficient condition. If $\nabla g\left(x^\star\right)=0$ and $\nabla^2 g\left(x^\star\right)$ is positive definite, then $x^\star$ is locally optimal.

The function $g(x)=x^3$ provides an example that shows that 'positive definite' cannot be replaced by 'positive semidefinite' in the sufficient condition. At $x=0$ it satisfies $g'(x)=3x^2=0$ and $g''(x)=6x=0$, although $x=0$ is not a local minimum.

22.3 Newton's method for minimizing a convex function

We first consider the important case when the objective function g is convex. As we have seen in section 5.3, we can find the minimum by solving $\nabla g(x)=0$. This is a set of n nonlinear equations in n variables, that we can solve using any method for nonlinear equations, for example, Newton's method.

For simplicity we assume that $\nabla^2 g(x)$ is positive definite everywhere, which is a little stronger than requiring $\nabla^2 g(x)$ to be positive semidefinite.

Newton's method for solving nonlinear equations, applied to $\nabla g(x)=0$, is based on the iteration

$$x^{(k+1)}=x^{(k)}-\nabla^2 g\left(x^{(k)}\right)^{-1}\nabla g\left(x^{(k)}\right),\quad k=0,1,2,\ldots$$

1 Algorithm 5.1. NEWTON'S METHOD FOR UNCONSTRAINED MINIMIZATION. given initial x, tolerance $\epsilon>0$ repeat 1. Evaluate $\nabla g(x)$ and $\nabla^2 g(x)$. 2. if $\|\nabla g(x)\|\leq\epsilon$, return x. 3. Solve $\nabla^2 g(x)v=-\nabla g(x)$. 4. $x:=x+v$. until a limit on the number of iterations is exceeded

Since $\nabla^2 g(x)$ is positive definite, we can use the Cholesky factorization in step 3 . The vector v computed in the k th iteration is called the Newton step at $x^{(k)}$:

$$v^{(k)}=-\nabla^2 g\left(x^{(k)}\right)^{-1}\nabla g\left(x^{(k)}\right).$$

The Newton step can be interpreted in several ways. Interpretation as solution of linearized optimality condition In chapter 4 we have seen that the iterates in Newton's method for solving nonlinear equations can be interpreted as solutions of linearized problems. If we linearize the optimality condition $\nabla g(x)=0$ near $\hat{x}=x^{(k)}$ we obtain

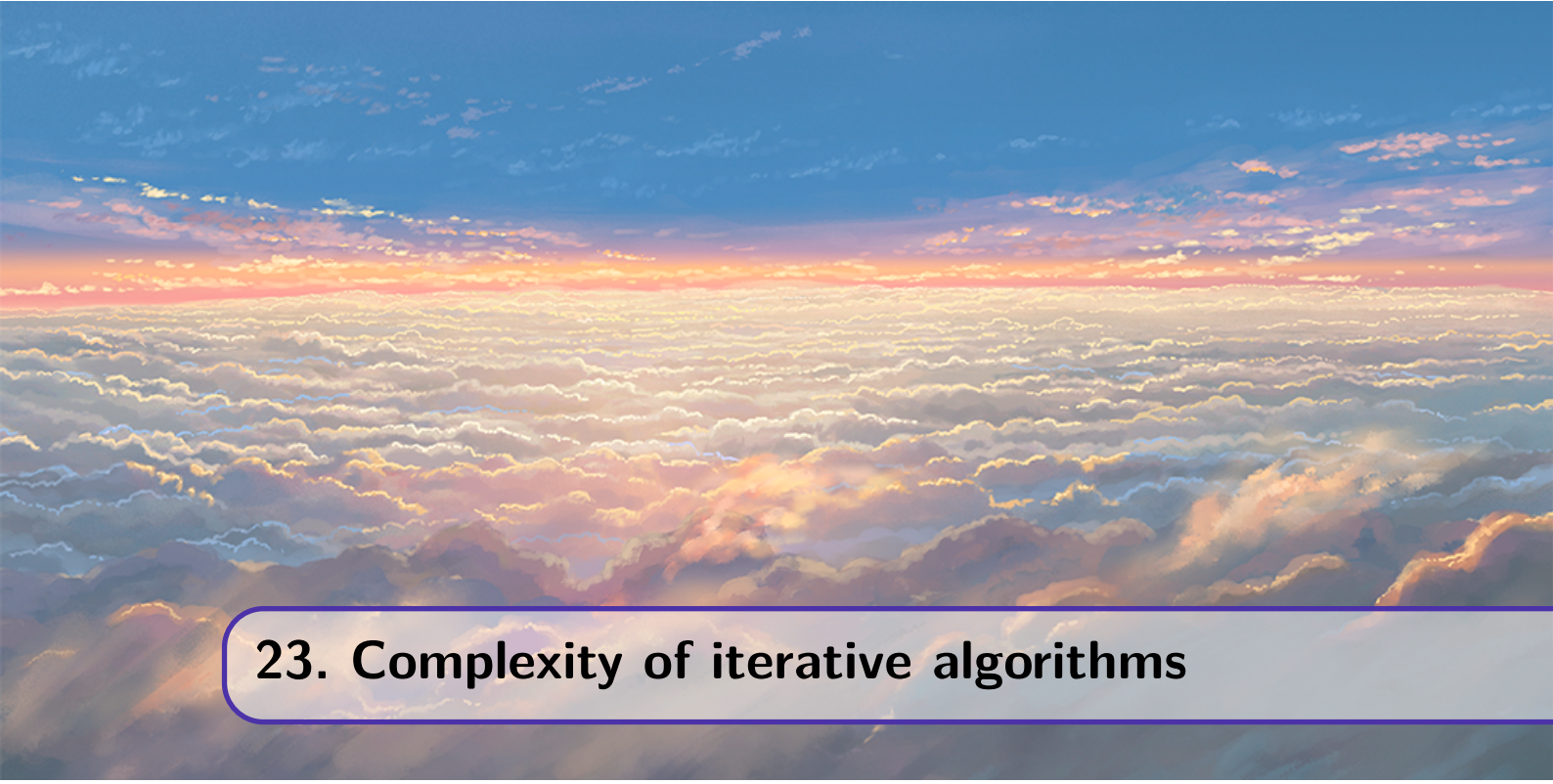
$$\nabla g(x)\approx\nabla g(\hat{x})+\nabla^2 g(\hat{x})(x-\hat{x})=0.$$

This is a linear equation in x, with solution

$$x=\hat{x}-\nabla^2 g(\hat{x})^{-1}\nabla g(\hat{x})=x^{(k)}+v^{(k)}.$$

So the Newton step $v^{(k)}$ is what must be added to $x^{(k)}$ so that the linearized optimality condition holds.

When $n = 1$ this interpretation is particularly simple. The solution of the linearized optimality condition is the zero-crossing of the derivative $g'(x)$, which is monotonically increasing since $g''(x) > 0$. Given our current approximation $x^{(k)}$ of the solution, we form a first-order Taylor approximation of $g'(x)$ at $x^{(k)}$. The zerocrossing of this approximation is then $x^{(k)} + v^{(k)}$. This interpretation is illustrated in figure 5.3.



23. Complexity of iterative algorithms

23.1 Iterative algorithms

The matrix algorithms we discussed so far are non-iterative. They require a finite number of floating-point operations, that can be counted and expressed as a polynomial function of the problem dimensions. In chapters 4 and 5 we discuss problems that are solved by iterative algorithms. By this is meant an algorithm that computes a sequence of values $x^{(0)}, x^{(1)}, x^{(2)}, \ldots$, with

$$x^{(k)} \to x^*$$

as $k \to \infty$, where the scalar or vector x^* is a solution of the problem. $x^{(0)}$ is called the starting point of the algorithm, and $x^{(k)}$ is called the k th iterate. Moving from $x^{(k)}$ to $x^{(k+1)}$ is called an iteration of the algorithm. The algorithm is terminated when $\left\|x^{(k)} - x^*\right\| \leq \epsilon$, where $\epsilon > 0$ is some specified tolerance, or when it is determined that the sequence is not converging (for example, when a limit on the number of iterations is exceeded).

The total cost of an iterative algorithm is more difficult to estimate than the cost of a non-iterative algorithm, because the number of iterations depends on the problem parameters and on the starting point. The efficiency of an iterative algorithm is therefore usually not expressed by giving its flop count, but by giving upper bounds on the number of iterations to reach a given accuracy. Deriving such bounds is the purpose of convergence analysis.

Iterative algorithms are often classified according to their rate of convergence. In the following paragraphs we give an overview of the most common definitions. For simplicity we will assume x^* is a scalar, and $x^{(k)} (k = 0, 1, 2, \ldots)$ is a sequence of numbers converging to $x^\star$.

Absolute and relative error The error after k iterations can be expressed as the absolute error, $\left|x^{(k)} - x^\star\right|$, or as the relative error $\left|x^{(k)} - x^\star\right| / |x^\star|$ (which is defined only if $x^\star \neq 0$). The relative error can also be expressed as the number of correct digits, for which several slightly different definitions exist. The definition that we will

adopt is the following: if the relative error $\left|x^{(k)} - x^\star\right| / |x^\star|$ is less than one, then the number of correct digits in $x^{(k)}$ is defined as

$$\left\lfloor -\log_{10}\left(\frac{\left|x^{(k)} - x^\star\right|}{|x^\star|}\right)\right\rfloor$$

(by $\lfloor \alpha \rfloor$ we mean the largest integer less than or equal to α). In other words, we take the logarithm with base 10 of the relative error (which is a negative number if the relative error is less than one), change its sign, and round it down to the nearest integer. This means that r correct digits correspond to relative errors in the interval $\left[10^{-r}, 10^{-r-1}\right)$.

23.2 Linear and R-linear convergence

A sequence $x^{(k)}$ with limit $x^\star$ is linearly convergent if there exists a constant $c \in (0, 1)$ such that

$$\left|x^{(k)} - x^\star\right| \leq c \left|x^{(k-1)} - x^\star\right|$$

for k sufficiently large. For example, the sequence $x^{(k)} = 1 + (1/2)^k$ converges linearly to $x^\star = 1$, because

$$\left|x^{(k+1)} - x^\star\right| = (1/2)^{k+1} = \frac{1}{2}\left|x^{(k)} - x^\star\right|$$

so the definition is satisfied with $c = 1/2$.

If $x^\star \neq 0$, we can give an intuitive interpretation of linear convergence in terms of the number of correct digits in $x^{(k)}$. Let

$$r^{(k)} = -\log_{10}\frac{\left|x^{(k)} - x^\star\right|}{|x^\star|}.$$

Except for rounding to an integer, $r^{(k)}$ is the number of correct digits in $x^{(k)}$. If we divide both sides of the inequality (3.1) by $|x^\star|$ and take logarithms, we obtain

$$r^{(k+1)} \geq r^{(k)} - \log_{10} c.$$

Ignoring the effect of rounding, we can say we gain at least $-\log_{10} c$ correct digits per iteration.

We can verify this using the example $x^{(k)} = 1+1/2^k$. As we have seen, this sequence is linearly convergent with $c = 1/2$, so we expect to gain roughly $-\log_{10} 1/2 = 0.3$ correct digits per iteration, or in other words, one correct digit per three or four iterations. This is confirmed by table 3.1, which shows the first ten values of $x^{(k)}$.

23.2.1 R-linear convergence

R-linear convergence Linear convergence is also sometimes defined as follows. A sequence $x^{(k)}$ with limit $x^\star$ is R-linearly convergent if there exists a positive M and $c \in (0, 1)$ such that

$$\left|x^{(k)} - x^\star\right| \leq Mc^k$$

for sufficiently large k. This means that for large k the error decreases at least as fast as the geometric series Mc^k. We refer to this as R-linear convergence to distinguish it from the first definition. Every linearly convergent sequence is also R-linearly convergent, but the converse is not true. For example, the error in an R-linearly convergent sequence does not necessarily decrease monotonically, while the inequality (3.1) implies that $\left|x^{(k)} - x^\star\right| < \left|x^{(k-1)} - x^\star\right|$ for sufficiently large k.

23.3 Quadratic convergence

A sequence $x^{(k)}$ with limit $x^\star$ is quadratically convergent if there exists a constant $c > 0$ such that

$$\left|x^{(k)} - x^\star\right| \leq c \left|x^{(k-1)} - x^\star\right|^2$$

for k sufficiently large. The sequence $x^{(k)} = 1 + (1/2)^{2^k}$ converges quadratically to $x^\star = 1$, because

$$\left|x^{(k+1)} - x^\star\right| = (1/2)^{2^{k+1}} = \left((1/2)^{2^k}\right)^2 = \left|x^{(k)} - x^\star\right|^2$$

so the definition is satisfied with $c = 1$. If $x^\star \neq 0$, we can relate the definition to the number of correct digits in $x^{(k)}$. If we define $r^{(k)}$ as above, we can write the inequality (3.3) as

$$r^{(k)} \geq 2r^{(k-1)} - \log_{10}\left(\left|x^\star c\right|\right).$$

Since $\left|x^{(k)} - x^\star\right| \to 0$, we must have $r^{(k)} \to +\infty$, so sooner or later the first term on the right-hand side will dominate the second term, which is constant. For sufficiently large k, the number of correct digits roughly doubles in each iteration. Table 3.2 shows the first few values of the sequence $x^{(k)} = 1 + (1/2)^{2^k}$ which converges quadratically with $c = 1$, and $x^\star = 1$. We start with one correct digit. It takes two iterations to get the second correct digit. The next iteration we gain one digit, then we gain two in one iteration, etc.

23.4 Superlinear convergence

A sequence $x^{(k)}$ with limit $x^\star$ is superlinearly convergent if there exists a sequence $c_k > 0$ with $c_k \to 0$ such that

$$\left|x^{(k)} - x^\star\right| \leq c_k \left|x^{(k-1)} - x^\star\right|$$

for sufficiently large k.

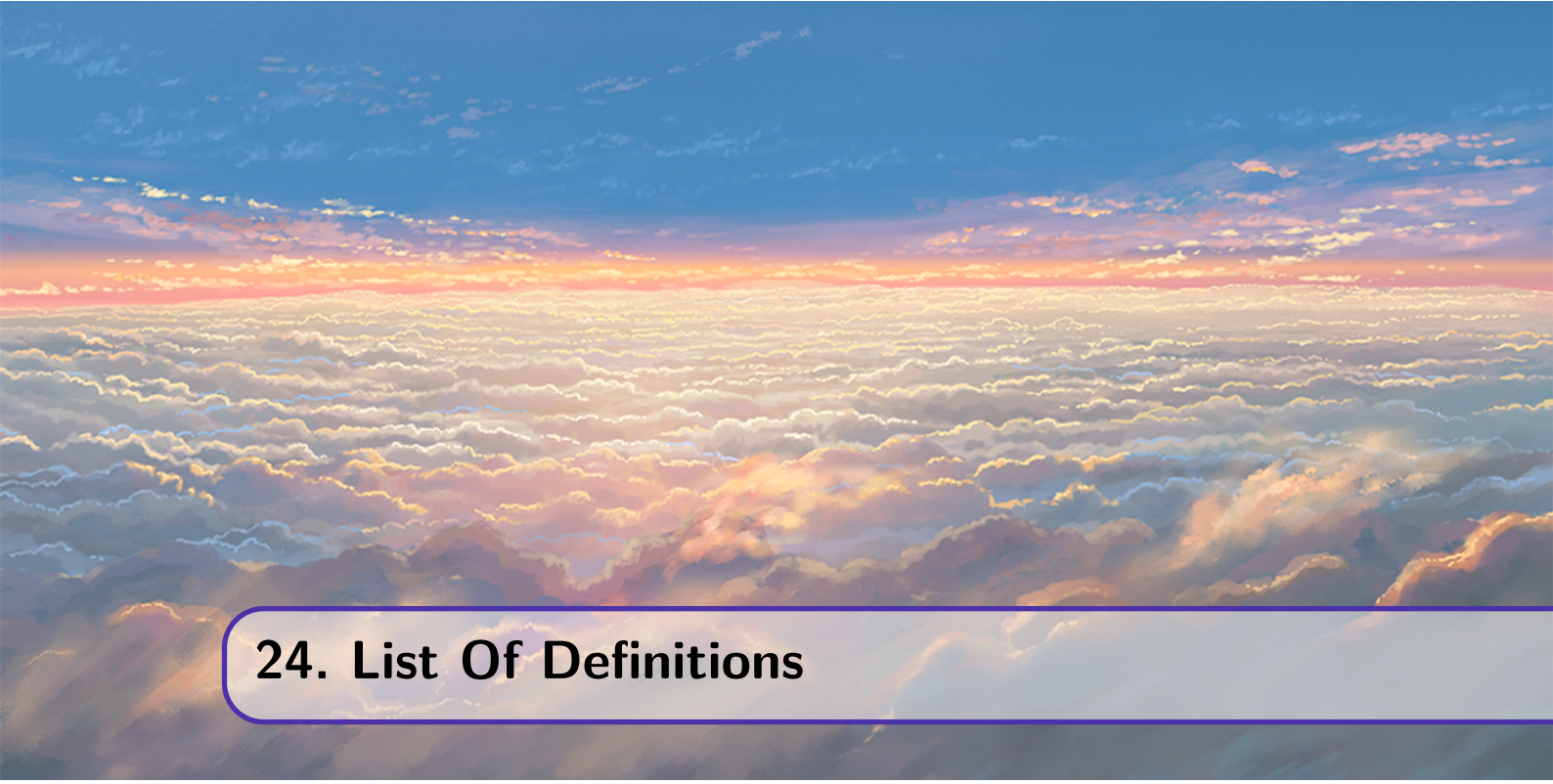
The sequence $x^{(k)} = 1 + (1/(k+1))^k$ is superlinearly convergent because

$$\left|x^{(k)} - x^\star\right| = \frac{1}{(k+1)^k} = \frac{k^{k-1}}{(k+1)^k}\frac{1}{k^{k-1}} = \frac{k^{k-1}}{(k+1)^k}\left|x^{(k-1)} - x^\star\right|,$$

so the definition is satisfied with $c_k = k^{k-1}/(k+1)^k$, which indeed goes to zero. If we define $r^{(k)}$ as above, we can write the inequality (3.3) as

$$r^{(k)} \geq r^{(k-1)} - \log_{10}(c_k),$$

and since $c_k \to 0$, $-\log_{10} c_k \to \infty$. For sufficiently large k, the number of correct digits we gain per iteration ($-\log_{10}(c_k)$) increases with k.



24. List Of Definitions