09-04-05-04-MatrixTheory

Created on 20241206.

Last modified on 2024 年 12 月 21 日.

目录

Chapter 1 Overall

Matrix Theory

Augmented matrix

Chapter 2 Matrix Space

2.0.1 Matrix

2.0.1.1 Defination

If we have a serious of $\boldsymbol{x}$, we have a serious of b, like this:

$$A_{mn} \cdot X_{nt} = B_{mt} \Longrightarrow \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ & \ddots & \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \cdot \left[\begin{array}{c:c:c} x_{11} & \cdots & x_{1t} \\ \vdots & & \vdots \\ x_{n1} & \cdots & x_{nt} \end{array}\right] = \left[\begin{array}{c:c:c} b_{11} & \cdots & b_{1t} \\ \vdots & \cdots & \vdots \\ b_{m1} & \cdots & b_{mt} \end{array}\right] \tag{2.1}$$

The normal definition of the product of two matrix is as above.

2.0.1.2 Complex Matrices

2.0.1.2.1 Conjugate Transposition for matrix $\boldsymbol{C} \in \mathbb{C}^{m \times n}$, we mark the Conjugate Transposition as C^H, where $c_{ji}^T = \bar{c}_{ij}$

in representation, $\boldsymbol{C} = \boldsymbol{A} + i\boldsymbol{B}$. Usually 4 real matrix multiplications are needed to calculate $(C + iD)(E + iF)$, actually 3 multiplications are enough. $(C + iD)(E + iF) = (C + D)(E - F) + CF - DE + i(DE + CF)$

2.0.1.3 Multiplication

There are 6 views sorting with the loop order, we fully understand that. for example, we can think the order jki(j is the outer, i is the inner) as follows

$$\begin{aligned} & i : | \ \ \cdot = | \\ & k : [\,|\,|\,] \ \ | = \sum | \\ & j : [\,|\,|\,] \ \ [\,|\,|\,] = [\,|\,|\,] \end{aligned} \tag{2.2}$$

We collect the 6 vews into one table as fallows.

Table 2.1: $\boldsymbol{A}_{ik}\boldsymbol{X}_{kj} = \boldsymbol{B}_{ij}$

Order	innerLoop	MiddleLoop	dataAccess	view	comment
ijk	S-S:dot	rowV-M	$\boldsymbol{A}_{\alpha:}, [\boldsymbol{X}_{:\beta}], \boldsymbol{B}_{\alpha:}$	$[-]\cdot[\|\|\|] = [-]$	dot view $\rightharpoonup\downharpoonleft$
jik	S-S:dot	M-columnV	$[\boldsymbol{A}_{\alpha:}], \boldsymbol{X}_{:\beta}, \boldsymbol{B}_{:\beta}$	$[\equiv]\cdot[\|] = [\|]$	dot view $\downharpoonleft\rightharpoonup$
ikj	S-rowV:saxpy	rowV-M:gaxpy	$\boldsymbol{A}_{\alpha:}, [\boldsymbol{X}_{\beta:}], \boldsymbol{B}_{\alpha:}$	$[-]gaxpy[\equiv] = [-]$	useOfA $\rightharpoonup\downharpoonleft$
jki	colV-S:saxpy	M-colV:gaxpy	$[\boldsymbol{A}_{:\alpha}], \boldsymbol{X}_{:\beta}, \boldsymbol{B}_{:\alpha}$	$[\|\|\|\|]gaxpy[\|] = [\|]$	useOfB $\downharpoonleft\rightharpoonup$
kij	S-rowV:saxpy	colV-rowV:outP	$\boldsymbol{A}_{:\alpha}, \boldsymbol{X}_{\beta:}, \sum \boldsymbol{B}_{row}$	$\sum[\|]outProd[-] = \sum[\equiv]$	*on* $A \downharpoonleft outProd \rightharpoonup$
kji	colV-S:saxpy	colV-rowV:outP	$\boldsymbol{A}_{:\alpha}, \boldsymbol{X}_{\beta:}, \sum \boldsymbol{B}_{col}$	$\sum[\|]outProd[-] = \sum[\|\|\|\|]$	*on* $X \downharpoonleft outProd \rightharpoonup$

[1] S for scalar, V for vector, M for matrix；colV for column vector；outP for out product.
[2] $[-]gaxpy[\equiv] = [-]$ is $\sum[\cdot]gaxpy[-] = \sum[-]$.
[3] $[||||]gaxpy[|] = [|]$ is $\sum[|]gaxpy[\cdot] = \sum[|]$.

Table 2.2: $\boldsymbol{A}_{ik}\boldsymbol{X}_{kj} = \boldsymbol{B}_{ij}$

Order	InnerLoop	MiddleLoop	OuterLoop
ijk	$(rowV, colV) = S$	$(rowV, [colV]) = rowV$	collection
jik	$(rowV, colV) = S$	$([rowV], colV) = rowV$	collection
ikj	$(S, colV) = colV$	$(rowV, [colV]) = \sum colV$	collection
jki	$(colV, S) = colV$	$([colV], colV) = \sum colV$	collection
kij	$(S, rowV) = rowV$	$(colV, rowV) = [rowV]$	collection and $\sum[rowV]$
kji	$(colV, S) = colV$	$(colV, rowV) = [colV]$	collection and $\sum[colV]$

[1] $\sum$ comes with k.

2.0.1.4 Transposition

Defination: $a_{ij}^T = a_{ji}$

Proposition 2.1. $(\boldsymbol{AB})^T = \boldsymbol{B}^T\boldsymbol{A}^T$

Proof: $L = (a_{ik}b_{kj})^T = c_{ij}^T = c_{ji} = b_{jk}a_{ki} = R$ □

Proposition 2.2. *We take a look a the product with reflect* $T : \boldsymbol{x} \to \boldsymbol{T}\cdot\boldsymbol{x}$. $(\boldsymbol{Tx})^T\boldsymbol{Ty} = \boldsymbol{x}^T(\boldsymbol{T}^T\boldsymbol{T})\boldsymbol{y} = [(\boldsymbol{TT}^T)\boldsymbol{x}]^T\boldsymbol{y}$. $0 \leqslant \|\boldsymbol{TT}^T\| < 1$, $\boldsymbol{T}$ *is a contractive mapping.*

2.0.2 operation

2.0.2.1 procuct

$\boldsymbol{Ax} = \boldsymbol{y}$

2.0.2.2 dot procuct AX = B

Focus on each element of B.

2.0.2.2.1 vector vector

for vector, $\boldsymbol{x}.*\boldsymbol{y} = \boldsymbol{x}^T\boldsymbol{y}$,

2.0.2.2.2 matrix matrix for matrix, this is the definition of the multiplication of the matrix, $\boldsymbol{A}_{mn} . * \boldsymbol{B}_{mn} = [a_{ij} \cdot b_{ij}]_{mn}$

2.0.2.3 outer procuct $\mathbf{AX} = \mathbf{B}$

Focus on each element of X, with X is seperated as row by row.

2.0.2.3.1 vector vector $\mathbf{AX} = \mathbf{B}$ $\boldsymbol{x}\boldsymbol{y}^T := [x_i]_{m1} \cdot [y_j]_{1n} = [x_i y_j]_{mn}$

In row view, we have $i \rightarrow$: $\boldsymbol{A}_{i:} = x_i \cdot \boldsymbol{y}^T$, this notation means that for each i, we do the follows. And $\boldsymbol{A}_{i:}$ means the ith row of the row seperation of $\boldsymbol{A}$

In column view, we have $j \rightarrow$: $\boldsymbol{A}_{:j} = \boldsymbol{x} \cdot y_j$

2.0.2.3.2 matrix matrix $[||||] outerProduct [—] = [\quad]$, we just sum each matrix $\boldsymbol{M}$, where $\boldsymbol{M} = [||] outerProduct [—]$.

$\boldsymbol{X}_{mk} \cdot \boldsymbol{Y}_{kn} = k \rightarrow$: $outerProduct\ of (\boldsymbol{X}_{:k}, \boldsymbol{Y}_{k:})$

We carefully focus on the use of each element of the matrix $\boldsymbol{Y}$, like $A_{11}, A_{12}, A_{13}, \cdots$, we can see it is true.

2.0.2.3.3 question

Question 2.1. *power function 001 solve* $(\boldsymbol{x}\boldsymbol{y}^T)^k$. *If k=1, easy. if k>1, ans =* $(\boldsymbol{y}^T\boldsymbol{x})^{k-1}\boldsymbol{x}\boldsymbol{y}^T$

Question 2.2. *power function 002*

solve $(\boldsymbol{X}\boldsymbol{Y}^T)^k, X, Y \in \mathbb{R}^{n \times 2}$. *Same trick like power function 001.*

2.0.2.4 saxpi

2.0.2.4.1 scalar scalar $y = ax + y$

2.0.2.4.2 scalar vector $\boldsymbol{y} = a \cdot \boldsymbol{x} + \boldsymbol{y}$

2.0.2.4.3 matrix vector $\boldsymbol{y} = \boldsymbol{A} \cdot \boldsymbol{x} + \boldsymbol{y}$

2.0.2.4.3.1 view row: $[—] \cdot | = [—]$ *This is the basic view of the dot product of the matrix. in view row first, we have:*

Algorithm 1: saxpyMatrixVectorRowAlgo1

Input: $\boldsymbol{A}_{mn}, \boldsymbol{x}, \boldsymbol{y}$

Output: $\boldsymbol{y}$

Initialization: $i = 0, j = 0$;

for $i \leftarrow 0$ **to** $m - 1$ **do**

  for $j \leftarrow 0$ **to** $n - 1$ **do**

    $y_i \leftarrow A_{ij}x_j + y_i$

  end

end

return $\boldsymbol{y}$;

We seperate $\boldsymbol{A}$ *as row,* $\boldsymbol{A}_{mn} = [\boldsymbol{r}_i^T, ...]^T$*, the j range can be shinked, the algorithm is as follows. This means that, we operate each row at a time, and think each row is one whole object.*

Algorithm 2: saxpyMatrixVectorRowAlgo2

Input: $\boldsymbol{A}_{mn} = [\boldsymbol{r}_i^T, ...]^T, \boldsymbol{x}, \boldsymbol{y}$

Output: $\boldsymbol{y}$

Initialization: $i = 0, j = 0$;

for $i \leftarrow 0$ **to** $m - 1$ **do**

  $y_i \leftarrow \boldsymbol{r}_i^T \cdot \boldsymbol{x} + y_i$

end

return $\boldsymbol{y}$;

2.0.2.4.3.2 view column: $[||||] outerProduct [—] = [\ ]$ $\boldsymbol{A}_{mn}\boldsymbol{x} = \boldsymbol{y}$*, we seperate A column by column, x row by row, use outer product , focus on the use of x.*

in column view, we add each column of $\boldsymbol{A}$ *to the same output column to get the new* $\boldsymbol{y}$*, and the weight of each column comes from each row of* $\boldsymbol{x}$

Algorithm 3: saxpyMatrixVectorColumnAlgo1

Input: $\boldsymbol{A}_{mn}, \boldsymbol{x}, \boldsymbol{y}$

Output: $\boldsymbol{y}$

Initialization: $i = 0, j = 0$;

for $j \leftarrow 0$ **to** $n - 1$ **do**

  for $i \leftarrow 0$ **to** $m - 1$ **do**

    $y_i \leftarrow A_{ij}x_j + y_i$

  end

end

return $\boldsymbol{y}$;

Also with column seperation of $\boldsymbol{A}_{mn} = [\boldsymbol{c}_i, ...]$, *we have the vector view algorithm:*

Algorithm 4: saxpyMatrixVectorColumnAlgo2

Input: $\boldsymbol{A}_{mn} = [\boldsymbol{c}_i, ...], \boldsymbol{x}, \boldsymbol{y}$

Output: $\boldsymbol{y}$

```
Initialization: i = 0, j = 0;
for j ← 0 to n − 1 do
    y ← c_i · x_j + y
end
return y;
```

2.0.3 properties

2.0.3.1 geometry properties

2.0.3.2 4 subspace

$R(\boldsymbol{A})$: A 的列空间

$N(\boldsymbol{A})$: A 的右零空间，即满足 $\boldsymbol{Ax} = \boldsymbol{0}$ 的所有 $\boldsymbol{x}$ 在的空间

$R(\boldsymbol{A}^T)$: A 的行空间

$N(\boldsymbol{A}^T)$: A 的左零空间

Lemma 2.1. $N(\boldsymbol{A}) = R(\boldsymbol{A}^T)^{\perp}$

proof: $\forall \boldsymbol{x} \in N(\boldsymbol{A})$, $\boldsymbol{\beta} \in R(\boldsymbol{A}^T)$, *we have* $\boldsymbol{x} \cdot \boldsymbol{\beta} = 0$, *which means that* $\boldsymbol{x} \in R(\boldsymbol{A}^T)^{\perp}$, $\square$.

Chapter 3 Linear System

Normally, we consider vector space over the fields of real or complex numbers.

3.0.1 linear equation Ax = B

3.0.1.1 Defination

linear equation in n variables.$\sum_{i=1}^{i=n} a_i x^i = b$*, which can be written as* $\boldsymbol{a}^T\boldsymbol{x} = b$*. We collect m equations and write like this:*

$$\begin{bmatrix} \boldsymbol{a}_1^T \\ \vdots \\ \boldsymbol{a}_m^T \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{x} \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} \tag{3.1}$$

Noticed that x_1 *is only applied toe the first column of the left matrix, we can say that* $\boldsymbol{x}$ *is one point, or a specific composition, of the space spanned by the column vector of the matrix. Then it is easy to see that this equation has the solution, only if the vector* $\boldsymbol{b}$ *is in the space spanned by the column vector of the matrix.*

Or we can write like this:

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ & \ddots & \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} \tag{3.2}$$

The equation $\boldsymbol{Ax} = \boldsymbol{b}$ *has solution, means* y 可由 A 的列向量线性表出。

If $\mathbf{b} = \mathbf{0}$*, called homogeneous linear equations, homogeneous because* 所有非 *0* 项是 *1* 次的。*if* $\mathbf{b} \neq \mathbf{0}$*, it is inhomogeneous.* 显然 *0* 向量 *(zero solution, or trivial soltution)* 是一个解*.* A 的列向量正交，只有零解；若 A 的列向量线性相关，有多解，即可按多种方式回到原点。

3.0.1.2 Number of solution

3.0.1.2.1 非齐次线性 n 元线性方程组解的个数等解集结构的研究，期待在不求解的情况下有所了解，就需要研究系数矩阵表示的 n 维向量空间的性质。

构造增广矩阵 $[\boldsymbol{A},\boldsymbol{b}]$ 后，初等行变换化为阶梯型，如??所示，解的个数讨论。

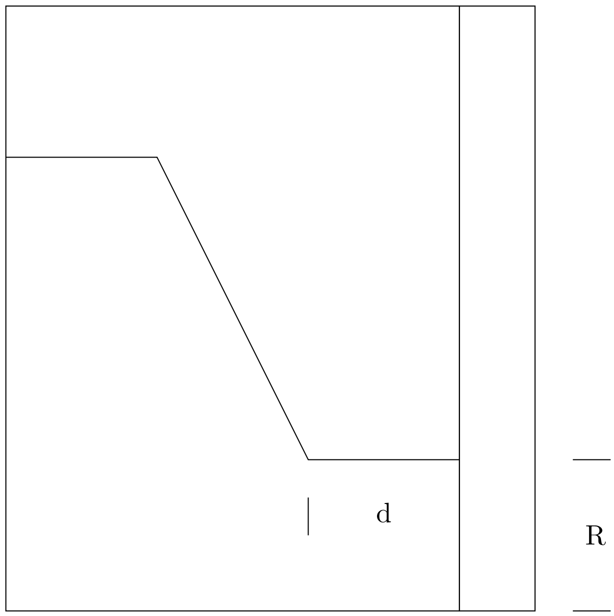

Figure 3.1: 【number of solution】

总共有 *n+1* 列，下面 r 行都是 *0.*

(1)$d=0$，即最后一个个主元在第 *n+1* 列，即存在方程 *0=1*，无解*, no solution*。
(2)$d=1$，即最后一个个主元在第 n 列，唯一解*, one solution,* $tr\boldsymbol{A}_{mn}=m$。
(3)$d>1$，即最后一个个主元在第 t 列，$t<n$。高度 R 所在的行号记为 r。有无穷个解。解可以这样写出，共 R 行，即 R 个主元，每个主元都用所在行的常数项 d 和 *n-r* 个自由元表示出来。

根据主元的构造过程，t 的列号一定大于等于 r。

当 A_{ii} 都是主元的时候*,*$d\neq 1$*,* $tr\boldsymbol{A}_{mn}<m$*,inifinity solution*，最后一行是解的超平面方程*,* 图中 d 是解的维度，$d=n-tr\boldsymbol{A}_{mn}$*,* 如 d 为 *3*，有 *3* 列独立的，即解空间是三维的。齐次方程组的未知数个数大于方程个数，有无数解。

$\det\boldsymbol{A}=0$*, no solution, or infinite solution.* $\det\boldsymbol{A}\neq 0$*, one solution.*

3.0.1.2.2 齐次线性 一定有 *0* 解，因而当有非 *0* 解时，有无穷个解。n 列时，系数矩阵的秩 $r<n$。

方程个数 $s<n$ 时，由于 $r\leqslant s<n$，易知有无穷个解。

3.0.2 solve equation

$\boldsymbol{A}_{mn}\boldsymbol{x}=\boldsymbol{y}$ 求解方法，如消元法、迭代法等。

3.0.2.1 elimination 消元法

3.0.2.1.1 Gaussian Elimination 基础步骤的 $O(n)$ 的，但是最终组合起来就是 $O(n^3)$ 的。

利用初等变换化（同解变换）为“阶梯形（或称上三角形）”，从下往上回代。

阶梯型：*1*）*0* 行在下方；*2*）每行首个非 *0* 元的列号随行号增大而严格增大。

简化阶梯型：*1*）阶梯型；*2*）主元是 *1*；*3*）主元所在列其他元素是 *0.*

简化阶梯型后，可直接写出一般解，如下方程，其中主变量是 x_1, x_3，其余是自由未知量。

$$\begin{bmatrix} 1 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix} \tag{3.3}$$
$$Ans: x_1 = x_2 + 2; x_3 = -1$$

3.0.3 排列

3.0.3.0.1 偶排列 如 *2431*,顺序对有 $24, 23$, 逆序对有 $21, 43, 41, 31$,逆序数是 *4*,记为 $\tau(2431) = 4$, 是偶数则为偶排列。

Lemma 3.1. 对换改变奇偶性，如 *2431* 是偶排列，对换 *4* 和 *1* 后得到的 *2134* 是奇排列。

证明：对换 ab,

若 ab 相邻：偏序函数原来查询 *(ab)*, 记为 $P(a,b)$, 对换后改为 $P(b,a)$，反号，而 b 更后面的元素相关的查询不受影响，因而改变符号；

若 ab 不相邻：记为 $ax_1\cdots x_tb$, 经过 t 次对换变为 $x_1\cdots x_tab$, 经过 *t+1* 次对换变为 $bx_1\cdots x_ta$, 即改变符号。若 ab 不相邻，还可以这样考虑：对换前后，与 a 和 b 有关的查询为 $(a,[x_i,b]),(x_i,b)$, 对换后即将其中 a 和 b 互换，影响的查询共有 *2t+1* 个，即改变符号。即证。

Chapter 4 Eigenvalue problem

4.0.1 Eigenvalue of Linear transformation

《矩阵理论-陈大新》

4.0.2 Eigenvalue of special matrix

《矩阵理论-陈大新》

4.0.3 最小多项式

《矩阵理论-陈大新》

4.0.4 圆盘定理

《矩阵理论-陈大新》

Chapter 5 polynomial

因式分解定理，多项式的根，多元多项式。

Chapter 6 operation

代数运算、分块运算、乘法、秩

Chapter 7　Transformation

坐标变换、像与核、特征向量、特征子空间、商空间

正交变换规范变换

酉相似

7.0.1 Elementary Transformation

初等变换。

1) 交换两行*:* $\boldsymbol{A} \xrightarrow{(i,j)} \boldsymbol{B}$

2) 某行乘以不为 *0* 的数*:* $\boldsymbol{A} \xrightarrow{\lambda(i)} \boldsymbol{B}$

3) 某行乘以不为 *0* 的数加到另一行上*:* $\boldsymbol{A} \xrightarrow{\lambda(i)+(j)} \boldsymbol{B}$

初等矩阵*:* 单位矩阵执行一系列初等变换得到的矩阵*.*

初等变换作用于矩阵 $\boldsymbol{A}$*,* 等于初等变换作用于单位阵之后得到的初等矩阵 $\boldsymbol{E}$ 再作用于 $\boldsymbol{A}$*.*

7.0.2 Linear Transformation

线性变换

7.0.3 Base Transformation

[Defination: similarity] Transformation Simplification

The motivation is about the base.Changing the bases of a transformation can help simplify the computation. We want to compure $\boldsymbol{y} = \boldsymbol{B}\boldsymbol{x}$*, in current base,* $\boldsymbol{B}$ *is difficult to compute, we have a simple transformation* $\boldsymbol{A}$*, and we want to find a good base transformation* $\boldsymbol{P}$ *where we have* $\boldsymbol{P}\boldsymbol{y} = \boldsymbol{A}\boldsymbol{P}\boldsymbol{x}$*, therefore we have* $\boldsymbol{B} = \boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{P}$*. If* $\exists \boldsymbol{P}$*, we note as* $\boldsymbol{A} \sim \boldsymbol{B}$*.*

Lemma 7.1. *If* $\boldsymbol{A} \sim \boldsymbol{B}$*, then* $\lambda_A = \lambda_B$*.*

[Prove] We use the defination of λ_A*, for* $\boldsymbol{x}$*, we have* $\boldsymbol{P}\boldsymbol{B}\boldsymbol{P}^{-1}\boldsymbol{x} = \lambda_A \boldsymbol{x}$*, which means* $\boldsymbol{B}(\boldsymbol{P}^{-1}\boldsymbol{x}) = \lambda_A(\boldsymbol{P}^{-1}\boldsymbol{x}) = \lambda_B(\boldsymbol{P}^{-1}\boldsymbol{x}) \quad \square$*.*

For the simpest $\boldsymbol{A}$ is diag.

Chapter 8 Orthogonal

8.1 Orthogonal matrix

正交矩阵和酉矩阵

$$\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{I}$$

which means, $\boldsymbol{A}^T = \boldsymbol{A}^{-1}$

For complex matrix, we use conjugate transpose, and note as Unitary matrix.

8.2 Orthogonal Bases

8.2.1 Gram-Schmidt Algorithm：GS 正交化

normalize a given set $\{\boldsymbol{\alpha}_i\}$ *to* $\{\boldsymbol{\beta}_i\}$*, such that* $\boldsymbol{\beta}_i\boldsymbol{\beta}_j = 0, \forall i, j$*, and* $span\{\boldsymbol{\alpha}_i\} = span\{\boldsymbol{\beta}_i\}$

Algorithm 5: Algorithm Normalize:GS

Input: $\{\boldsymbol{\alpha}_i\}$

Output: *a normalized base* $\{\boldsymbol{\beta}_i\}$

$\boldsymbol{\beta}_1 = \boldsymbol{\alpha}_1$;

$\boldsymbol{\beta}_2 = \boldsymbol{\alpha}_2 - k_1\boldsymbol{\beta}_1$;

let $\boldsymbol{\beta}_1\boldsymbol{\beta}_2 = 0 \Rightarrow k_1 = \frac{\boldsymbol{\beta}_1\boldsymbol{\alpha}_2}{\boldsymbol{\beta}_1\boldsymbol{\beta}_1}$;

keep doing, we have

$$\boldsymbol{\beta}_k = \boldsymbol{\alpha}_k - \sum_{i=1}^{k} \frac{\boldsymbol{\beta}_i\boldsymbol{\alpha}_k}{\boldsymbol{\beta}_i\boldsymbol{\beta}_i}\boldsymbol{\beta}_i$$

return $\{\boldsymbol{\beta}_i\}$;

8.2.1.1 description

In 1907, Erhard Schmidt introduced an orthogonalizaiton algoritm, and he claimed the procedure was essentially the same as a paper by J. P. Gram in 1883.

form an orthogonal sequence$\boldsymbol{q}_n$ *from a linearly independent sequence*$\boldsymbol{x}_n$ *of members from inner-product space by defining* $\boldsymbol{q}_n$ *inductively as:*

$$\boldsymbol{q}_1 = \boldsymbol{x}_1, \boldsymbol{q}_n = \boldsymbol{x}_n - \sum_{k=1}^{n-1} \frac{< \boldsymbol{q}_k, \boldsymbol{x}_n >}{||\boldsymbol{q}_k||^2} \boldsymbol{q}_k, n \geqslant 2.$$

8.2.1.2 proof

the construction is like this, first we have $\boldsymbol{q}_1 = \boldsymbol{x}_1$*, then* $\boldsymbol{q}_2 = \boldsymbol{x}_2 - k_1 \boldsymbol{q}_1$.

for now, we have $span(\boldsymbol{q}_1, \boldsymbol{q}_2) = span(\boldsymbol{x}_1, \boldsymbol{x}_2)$*. With constraints* $< \boldsymbol{x}_1, \boldsymbol{x}_2 >= 0$,$k_1 = \frac{<\boldsymbol{q}_1, \boldsymbol{x}_n>}{||\boldsymbol{q}_1||^2}$. *Keep doing, like*

$\boldsymbol{q}_3 = \boldsymbol{x}_3 - k_1 \boldsymbol{q}_1 - k_2 \boldsymbol{q}_2$. *From the construction, we can see it is right.*

数学归纳法（*Mathematical Induction, MI*）

8.2.1.3 least squares problems

8.2.1.4 projection problem

8.2.1.5 example

8.2.1.6 exercises

8.2.2 Householder Reflection

point $\boldsymbol{p}$*, hyperplane with normal* $\boldsymbol{n}$*, the reflection of* $\boldsymbol{p}$ *about the plane:*

$$\begin{aligned} \boldsymbol{q} &= \boldsymbol{p} - 2 < \boldsymbol{p}, \boldsymbol{n} > \boldsymbol{n} \\ &= (\boldsymbol{I} - 2\boldsymbol{n}\boldsymbol{n}^T)\boldsymbol{p} \\ &= \boldsymbol{H}\boldsymbol{p} \end{aligned}$$

$\boldsymbol{H}$ *is Householder matrix.*

Lemma 8.1. $\boldsymbol{H}$ *is orthogonal.*

[prove]

$$(\boldsymbol{H})_{ij} = \begin{cases} -2n_i n_j, \ i \neq j \\ 1 - 2n_i n_j, \ i = j \end{cases}$$

we calculate $c = ((\frac{1}{2}\boldsymbol{H})(\frac{1}{2}\boldsymbol{H}^T))_{ij}$.*For* $i \neq j$, *we have,*

$$\begin{aligned} c = &+ (-n_i n_0)(-n_j n_0) + \cdots \\ &+ (\frac{1}{2} - n_i n_i)(-n_j n_i) + \cdots \\ &+ (-n_i n_j)(\frac{1}{2} - n_j n_j) + \cdots \\ &+ (-n_i n_{n-1})(-n_j n_{n-1}) = 0 \end{aligned}$$

For $i = j$, *we have,*

$$\begin{aligned} c = &+ (-n_i n_0)(-n_i n_0) + \cdots \\ &+ (\frac{1}{2} - n_i n_i)(\frac{1}{2} - n_i n_i) + \cdots \\ &+ (-n_i n_{n-1})(-n_i n_{n-1}) = \frac{1}{4} \quad \square. \end{aligned}$$

8.2.3 Application

When we need the reflection collinear to the vector $\boldsymbol{e}_1 = [1, 0, \cdots, 0]^T$, *the normal of the reflection superplane is*

$$\boldsymbol{n} = \frac{\boldsymbol{p} - ||\boldsymbol{p}||\boldsymbol{e}_1}{||\boldsymbol{p} - ||\boldsymbol{p}||\boldsymbol{e}_1||}$$

and

$$\boldsymbol{H}_1\boldsymbol{A} = \begin{bmatrix} \alpha_1 & \cdots \\ 0 & \\ \vdots & \boldsymbol{A}'_1 \\ 0 & \end{bmatrix} = \begin{bmatrix} \alpha_1 & \\ 0 & \boldsymbol{A}''_1 \\ \vdots & \\ 0 & \cdots \end{bmatrix}$$

$$\boldsymbol{H}_k = \begin{bmatrix} \boldsymbol{I}_{k-1} & 0 \\ 0 & \boldsymbol{H}'_k \end{bmatrix}, \boldsymbol{T}_k = \begin{bmatrix} \boldsymbol{I}_{k-1} & 0 \\ 0 & \boldsymbol{H}''_k \end{bmatrix}$$

where $\boldsymbol{H}'_k$ *is the Householder of* $\boldsymbol{A}'_k$, *it is obviously orthogonal. And we have* $\boldsymbol{H}_n \cdots \boldsymbol{H}_1\boldsymbol{A} = \boldsymbol{R}$, *therefore we have the QR decomposition of* $\boldsymbol{A}$

$$\boldsymbol{A} = \boldsymbol{H}_1^T \cdots \boldsymbol{H}_n^T\boldsymbol{R} = \boldsymbol{Q}\boldsymbol{R}$$

.

We want to rewrite each line, we can apply a householder $\boldsymbol{H}$ *to* $\boldsymbol{A}''^T_k$, *and then transpolate the result,* $(\boldsymbol{H}\boldsymbol{A}''^T_k)^T = \boldsymbol{A}''_k\boldsymbol{H}$, *we mark the row-Householder as* $\boldsymbol{T}$, *and we have*

$\boldsymbol{H}_n \cdots \boldsymbol{H}_1\boldsymbol{A}\boldsymbol{T}_2 \cdots \boldsymbol{T}_n = \boldsymbol{B}$,

where $\boldsymbol{B}$ *is the bidiagonal, therefore,*

$$\boldsymbol{A} = \boldsymbol{H}_1^T \cdots \boldsymbol{H}_n^T \boldsymbol{B} \boldsymbol{T}_n^T \cdots \boldsymbol{T}_2^T$$

.

Chapter 9 special matrix

9.1 schur 定理

9.2 正规矩阵

9.3 实对称矩阵和 Hermite 矩阵

Chapter 10 Decomposition

10.1 Transformation Decomposition

If we have $\boldsymbol{y} = \boldsymbol{B}\boldsymbol{x} = \boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{P}\boldsymbol{x} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^T\boldsymbol{x}$*, where* $\boldsymbol{U}$ *is rotation or reflection,* $\boldsymbol{\Sigma}$ *is scaling, the transformation is simplified.*

10.2 Eigen Decomposition

For square matrix.

10.2.1 Defination by transformation

$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^{-1}$*, where* $\boldsymbol{U}$ *is rotation or reflection,* $\boldsymbol{\Sigma}$ *is scaling.*

10.2.2 Defination by defination

For the given transformation$\boldsymbol{A}$*, if* $\exists \boldsymbol{v}, \lambda$*, such that* $\boldsymbol{A}\boldsymbol{v} = \lambda\boldsymbol{v}$*, and maybe we have some equations with different* λ*, we collect as* $\boldsymbol{\lambda}$ *as a diagonal matrix, and the equations as* $\boldsymbol{A}\boldsymbol{Q} = \boldsymbol{Q}\boldsymbol{\lambda}$*, where* $\boldsymbol{Q} = [\boldsymbol{v}_0, \cdots, \boldsymbol{v}_{n-1}]$*, therefore we have* $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\lambda}\boldsymbol{Q}^T$*, we call this is the Eigen Decomposition of* $\boldsymbol{A}$*.*

10.3 QR

10.3.0.0.1 QR factorization *the construction is like this:*

$$[\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_n] = [\boldsymbol{q}_1, \boldsymbol{q}_2, \cdots, \boldsymbol{q}_n] \cdot \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & & \\ & & \ddots & \\ 0 & 0 & 0 & r_{nn} \end{bmatrix}$$

10.4 SVD: Singular Value Decomposition

10.4.1 Defination

When the matrix is not square, the form $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$.

Lemma 10.1. $\boldsymbol{C}$ *is the Gram matrix of* $\boldsymbol{A}$, $\boldsymbol{\Sigma}^2 = \boldsymbol{\lambda}_C$

[Prove] $\boldsymbol{C} = \boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^T$ □.

10.4.2 Algorithm 1: QR

Algorithm 6: SVD:QR

Input: $\boldsymbol{A}_{mn}$

Output: $\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}$

$\boldsymbol{C} = \boldsymbol{A}^T\boldsymbol{A}$;

use symmetric QR: $\boldsymbol{C} = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^T$;

$\boldsymbol{U} = \boldsymbol{A}\boldsymbol{V}\boldsymbol{\Sigma}^{-1}$;

return $\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}$;

Chapter 11 Form

11.0.1 Jordan

Jordan 型、根子空间分解、循环子空间、多项式矩阵相抵不变量、特征方阵与相似标准型

11.0.1.1 不变子空间

《矩阵理论-陈大新》

11.0.1.2 特征值全 0 矩阵的 Jordan 标准型

《矩阵理论-陈大新》

11.0.1.3 Jordan 标准型计算

11.0.2 二次

配方法构造、对称方阵的相合、相合不变量

Chapter 12 参考文献说明

《矩阵理论-陈大新》[?]：好的观点的来源。