연세대학교 통계 데이터 사이언스 학회 ESC 23-2 SUMMER WEEK3

Orthogonalization/Triangularization, QR Factorization, Least Square Problem

[ESC 방학세션 5조] 김종민 이혜림 전인태 허정웅

Contents

1. QR Factorization

QR Factorization

QR Factorization 정의

- 실수 행렬 A를 orthonormal matrix Q와 upper triangular matrix R의 곱으로 나타내는 행렬 분해 방법

$$\begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix} = \begin{bmatrix} q_1 & q_2 & \cdots & q_n \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & & \vdots \\ & & \ddots & \\ & & & r_{nn} \end{bmatrix}$$

QR Factorization을 쓰는 이유

-least squares problem을 풀 때, QR 분해를 활용해 효과적으로 해를 찾을 수 있다

-eigenvalue, eigenvector 계산에 활용, 이를 통해 행렬의 중요한 특성을 파악하는데 도움

-linear system을 풀 때 더 효율적으로 풀 수 있다

-큰 행렬을 다룰 때 수치적으로 안정적인 해를 얻는 데 유용

QR Factorization

Reduced QR Factorization

- The successive spaces spanned by the columns $a_1, a_2, \ldots$of A

$$\langle a_1 \rangle \subseteq \langle a_1, a_2 \rangle \subseteq \langle a_1, a_2, a_3 \rangle \subseteq \cdots$$

- A $\in \mathbb{C}^{m \times n} (m \geq n)$ 가 full column rank 가진다고 가정했을 때,
  다음 성질을 만족하는 **orthonormal vector** $q_1, q_2, \cdots$를 찾는 것이 목표!

$$\langle q_1, q_2, \ldots, q_j \rangle = \langle a_1, a_2, \ldots, a_j \rangle \qquad j = 1, \ldots, n.$$

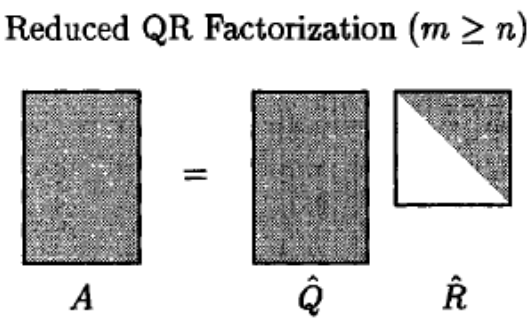
QR Factorization

Reduced QR Factorization

- $\langle q_1, q_2, \ldots, q_j \rangle = \langle a_1, a_2, \ldots, a_j \rangle$를 만족하는 matrix form

$$\begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix} = \begin{bmatrix} q_1 & q_2 & \cdots & q_n \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & & \vdots \\ & & \ddots & \\ & & & r_{nn} \end{bmatrix}$$

A Q R



- Q is $m \times n$ with orthonormal columns
- R is $n \times n$ and upper triangular

→ Reduced QR Factorization

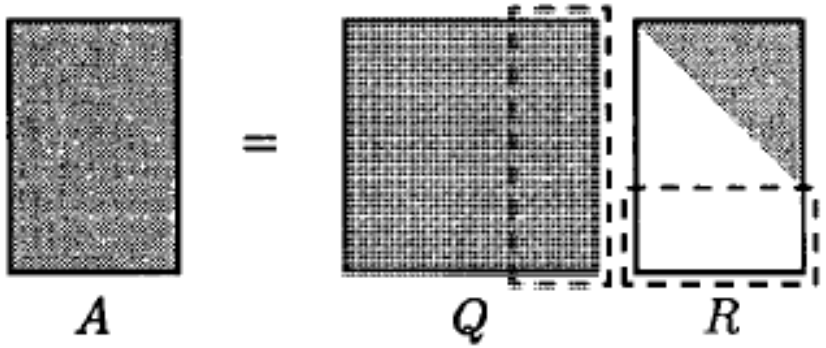
- $a_1, \ldots, a_k$는 $q_1, \ldots, q_k$의 linear combinations으로 표현 가능, 만약 R행렬이 invertible하다면 $q_1, \ldots, q_k$도 $a_1, \ldots, a_k$ 의 linear combinations으로 표현 가능

QR Factorization

Full QR Factorization

- A가 m×n 행렬일 때, m×m의orthonormal matrix Q와, m×n의 upper triangular matrix R로 분해

Full QR Factorization $(m \geq n)$



- Reduced QR에 (m-n)개의 orthonormal columns를 Q에 추가, (m-n)개의 row of zeros를 R에 추가한 것
- $j \geq n$에 대해서 Q의 열벡터 q_j는 range(A)에 대해 orthogonal하다

$col(A) = \langle q_1, \ldots, q_n \rangle$

QR Factorization

Existence and Uniqueness

Theorem 1.

Every A∈ $\mathbb{C}^{m\times n}(m \geq n)$ has a full QR factorization, hence also a reduced QR factorization

Proof

A가 full rank를 가정할 때, Gram-Schmidt 방법을 사용하면 QR분해가 존재함을 알 수 있다. QR 분해가 존재하지 않을 때는 Gram-Schmidt 과정 중 $v_j = 0$ 이 될 때이다.

하지만 $v_j = 0$은 $a_j \in \langle q_1, q_2, \dots, q_{j-1} \rangle = \langle a_1, a_2, \dots, a_{j-1} \rangle$을 의미하므로, 이는 A가 full rank를 가진다는 가정에 모순이 된다. 따라서 A가 full rank라면 항상 QR 분해가 존재한다.

또한 A가 full rank를 가지지 않을 때에는 $v_j = 0$인 j가 한번 이상 존재할 것이다. 그러나 이때는 단순히 $\langle q_1, q_2, \dots, q_{j-1} \rangle$에 orthogonal한 normalized vector를 q_j로 선택해서 Gram-Schmidt 방법을 계속하면 된다

QR Factorization

Existence and Uniqueness

Theorem 2.

Each A∈ $\mathbb{C}^{m\times n}(m \geq n)$ of full rank has a unique reduced QR factorization with $r_{jj} > 0$

Proof

Gram-Schmidt orthogonalization을 통해 QR 분해를 할 때, A가 full rank를 가진다면 R 행렬의 대각원소는 0이 될 수 없다 ($v_j \neq 0 \rightarrow |r_{jj}| = \|v_j\|_2 \neq 0$). 따라서 r_{jj}를 양수로 고정한다면, r_{jj} 를 결정할 수 있다. 또한 각 q_j를 구해가는 과정에서, r_{ij}, q_j는 완전히 결정할 수 있다. 따라서 A가 full rank를 가진다면 unique한 QR 분해를 할 수 있다.

QR Factorization

QR 분해에 사용되는 방법들

- Gram-Schmidt orthogonalization
- Householder transformations (elementary reflectors)
- Givens transformations(plane rotations)

2. Gram-Schmidt Orthogonalization

Introduction

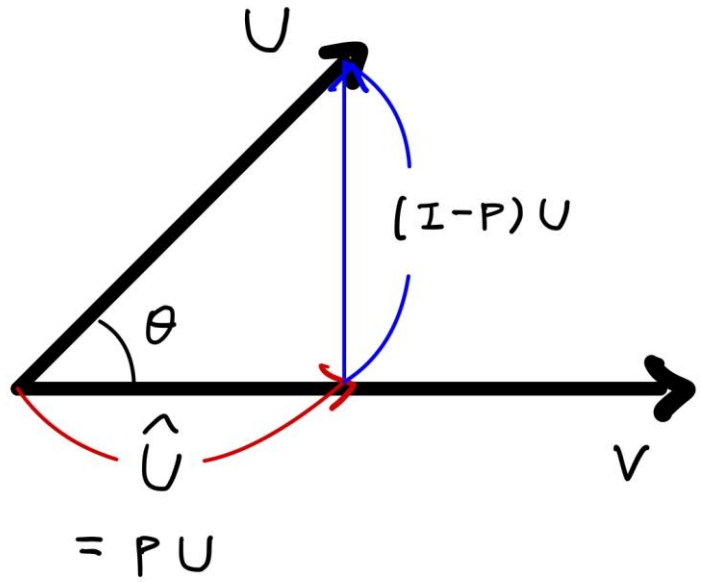
What is Projection?

Inner Product : $U^T V = \|U\| \, \|V\| \, cos\theta \;\Rightarrow\; cos\theta = \frac{U^T V}{\|U\| \, \|V\|}$

Project $U \; on \; V$: $\widehat{U} = \frac{V}{\|V\|} \|U\| \, cos\theta \; = \frac{U^T V}{V^T V} V = V(V^T V)^{-1} V^T U$

Projector : $\mathrm{P} = V(V^T V)^{-1} V^T \Rightarrow projection \; onto \; range(P)$

Complementary Projector : $(\mathrm{I} - \mathrm{P}) \Rightarrow projection \; onto \; null(P)$



Gram-Schmidt Orthogonalization

Gram-Schmidt Orthogonalization

1. a_j 에서 $q_1 \dots q_{j-1}$ 와 orthogonal 한 부분 추출

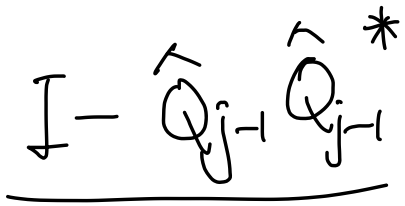
$$v_j = a_j - (q_1^* a_j) q_1 - (q_2^* a_j) q_2 - \dots - (q_{j-1}^* a_j) q_{j-1} = P_j a_j$$

2. Normalize 하기

$$q_j = \frac{v_j}{||v_j||} = \frac{P_j a_j}{||P_j a_j||} \; where \; P_j \; is \; projector \; onto \; the \; space \; orthogonal \; to < q_1, \dots, q_{j-1} >$$

3. 정리

$$q_n = \frac{a_n - \sum_{i=1}^{n-1} r_{in} q_i}{r_{nn}} \; where \; r_{ij} = q_i^* a_j \quad (i \neq j), \; |r_{ii}| = ||a_j - \sum_{i=1}^{j-1} r_{ij} q_j|| = ||v_j||$$



Gram-Schmidt Orthogonalization

Pseudo Code

$for\ j = 1\ \ to\ \ n$

$v_j = a_j$

$for\ i = 1\ \ to\ \ j - 1$

$r_{ij} = q_i^* a_j$

$v_j = v_j - r_{ij} q_i$

$r_{jj} = ||v_j||$

$q_j = v_j / r_{jj}$

Gram-Schmidt Orthogonalization

Modified Gram-Schmidt Orthogonalization

Classic G.S. : v_j 를 한번에 $p_1, \dots, p_{j-1}$ 에 orthogonal 하게 만듦 ⇒ unstable

Modified G.S. : $v_1, \dots, v_j$ 를 순차적으로 $q_1 \to q_2 \to \cdots \to q_{j-1}$에 orthogonal 하게 만듦

$\Rightarrow projector\ P_j = P'_{q_{j-1}} P'_{q_{j-2}} \dots P'_{q_1}$ $\quad where\ P'_{q_i}\ is\ projection\ onto\ space\ orthogonal\ to\ q_i\ ,\quad (P'_{q_i} = I - q_i\, q_i^*)$

$\Rightarrow v_j = P_j a_j = P'_{q_{j-1}} P'_{q_{j-2}} \dots P'_{q_1}\, a_j$

$v_j^{(1)} = a_j$

$v_j^{(2)} = P'_{q_1} v_j^{(1)} = v_j^{(1)} - q_1 q_1^* v_j^{(1)}$

$v_j^{(j)} = P'_{q_{j-1}} v_j^{(j-1)} = v_j^{(j-1)} - q_{j-1} q_{j-1}^* v_j^{(j-1)} = v_j$

Gram-Schmidt Orthogonalization

Pseudo Code

$for\ i = 1\ \ to\ \ n$

$\quad v_i = a_i$

$for\ i = 1\ \ to\ \ n$

$\quad r_{ii} = ||v_i||$

$\quad q_i = v_i / r_{ii}$

$\quad for\ j = i + 1\ \ to\ \ n$

$\quad\quad r_{ij} = q_i^* v_j$

$\quad\quad v_j = v_j - r_{ij} q_i$

R_1

$r_{11} = ||v_1|| \rightarrow q_1 = \frac{a_1}{r_{11}}$

$r_{12} = q_1^* a_2 \rightarrow v_2^{(2)} = a_2 - q_1^* a_2 q_1$

$r_{13} = q_1^* a_3 \rightarrow v_3^{(2)} = a_3 - q_1^* a_3 q_1$

$r_{1n} = q_1^* a_n \rightarrow v_n^{(2)} = a_n - q_1^* a_n q_1$

R_2

$r_{22} = ||v_2|| \rightarrow q_2 = \frac{v_2}{r_{22}} = a_2 - \frac{r_{12}\, q_1}{r_{22}}$

$r_{23} = q_2^* a_3 \rightarrow v_3^{(3)} = a_3 - q_1^* a_3 q_1 - q_2^* a_3 q_2$

$r_{2n} = q_2^* a_n \rightarrow v_n^{(3)} = a_n - q_1^* a_n q_1 - q_2^* a_n q_2$

$\Rightarrow\ v_n = v_n^{(n)} = a_n - q_1^* a_n q_1 - q_2^* a_n q_2 - \cdots - q_{n-1}^* a_n q_{n-1}$

$q_n = \frac{a_n - \sum_{i=1}^{n-1} r_{in} q_i}{r_{nn}}$

Gram-Schmidt Orthogonalization

Pseudo Code

$$\begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix} \begin{bmatrix} \frac{1}{r_{11}} & \frac{-r_{12}}{r_{11}} & \frac{-r_{13}}{r_{11}} & \cdots \\ & 1 & & \\ & & 1 & \\ & & & \ddots \end{bmatrix} = \begin{bmatrix} q_1 & v_2^{(2)} & \cdots & v_n^{(2)} \end{bmatrix}$$

$$R_2 = \begin{bmatrix} 1 & & & \\ & \frac{1}{r_{22}} & \frac{-r_{23}}{r_{22}} & \cdots \\ & & 1 & \\ & & & \ddots \end{bmatrix}, \quad R_3 = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \frac{1}{r_{33}} & \cdots \\ & & & \ddots \end{bmatrix}, \ldots.$$

$$AR_1R_2 \ldots R_n = \hat{Q}$$

$$R_1R_2 \ldots R_n = \hat{R}^{-1}$$

```
for (i in 1:3) {
  v[, i] = a[, i]
}

for (i in 1:3) {
  r[i, i] = norm(v[, i], type = "2")
  q[, i] = v[, i] / r[i, i]

  if (i == 3) {
    break
  }

  for (j in (i + 1):3) {
    r[i, j] = sum(q[, i] * v[, j])
    v[, j] = v[, j] - r[i, j] * q[, i]
  }
}
```

Gram-Schmidt Orthogonalization

Example

$$\begin{matrix} 12 & -51 & 4 \\ 6 & 167 & -68 \\ -4 & 24 & -41 \end{matrix}$$

```
> q
           [,1]       [,2]        [,3]
[1,]  0.8571429 -0.3942857 -0.33142857
[2,]  0.4285714  0.9028571  0.03428571
[3,] -0.2857143  0.1714286 -0.94285714
> r
     [,1] [,2] [,3]
[1,]   14   21  -14
[2,]    0  175  -70
[3,]    0    0   35
> 
```

```
for (i in 1:3) {
  v[, i] = a[, i]
}

for (i in 1:3) {
  r[i, i] = norm(v[, i], type = "2")
  q[, i] = v[, i] / r[i, i]

  if (i == 3) {
    break
  }

  for (j in (i + 1):3) {
    r[i, j] = sum(q[, i] * v[, j])
    v[, j] = v[, j] - r[i, j] * q[, i]
  }
}
```

Gram-Schmidt Orthogonalization

Cost of Modified Gram-Schmidt

```
for (i in 1:n) {
  r[i, i] = norm(v[, i], type = "2")
  q[, i] = v[, i] / r[i, i]

  if (i == n) {
    break
  }

  for (j in (i + 1):n) {
    r[i, j] = sum(q[, i] * v[, j])
    v[, j] = v[, j] - r[i, j] * q[, i]
  }
}
```

- $r_{ij} = q_i^* v_j$

  $multiplication\ cost : m \quad (multiplication\ of\ m\ rows)$
  $addition\ cost : m - 1 \quad (addtion\ of\ m\ rows)$

- $v_j = v_j - r_{ij} q_i$

  $multiplication\ cost : m \quad (multiplication\ of\ m\ rows)$
  $subtraction\ cost : m \quad (subtraction\ of\ m\ rows)$

- $total\ loop$

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} 4m \sim \sum_{i=1}^{n} (i) \times 4m \sim 2mn^2$$

$\Rightarrow 2mn^2\ cost$

$\frac{n^2}{2}$

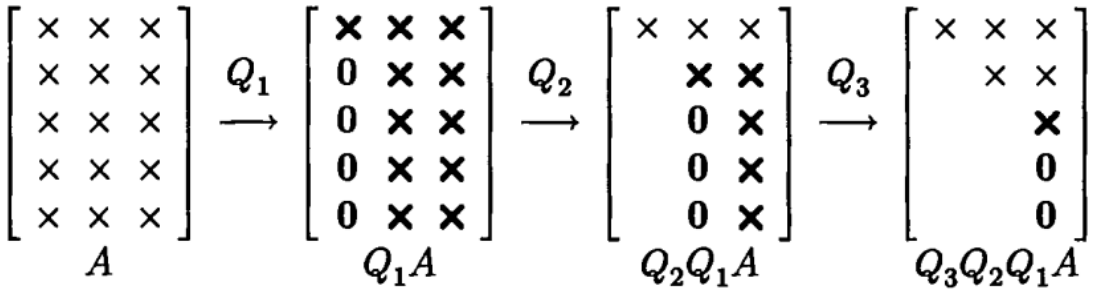
3.QR with Householder Reflections

QR with Householder Reflections

Basic Idea of QR from Householder

QR분해에서 Upper Triangular 행렬인 R을 만드는 다른 방법은 없을까?

⇒ 각 열의 대각성분 아래의 entry를 모두 0으로 바꿔주는 행렬을 찾자!



즉, 행렬 $A \in \mathbb{C}^{m\times n}$ $(m \geq n)$에 대하여 Q_k는 행렬 A의 k번째 열을 변형, 대각성분 아래의 entry인 $A_{k+1,k}$부터 $A_{m,k}$를 모두 0으로 바꾸는 역할을 하면 된다.

Q_k로 인해 $A_{1,k}$부터 $A_{k,k}$까지의 성분이 변형되어도 괜찮을까 ?

Q_k는 k번째 이외의 열은 건드려서는 안되며, $A_{1,k}$부터 $A_{k,k}$까지의 성분은 변형되어도 행렬 A를 upper triangular하게 만드는 것과 무관하므로 상관없다!

QR with Householder Reflections

Basic Idea of QR from Householder

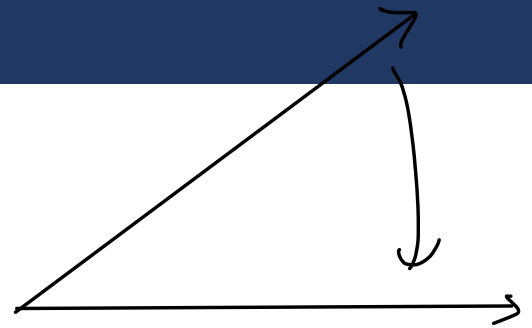
행렬 A에 n개의 열이 있으므로, 행렬 A를 upper triangular하게 바꾸는 데에 n개의 Q_i가 필요할 것이다. 즉, $(Q_n Q_{n-1} \cdots Q_1)A = R$이다.

그런데 QR 분해에 의하면, $A \in \mathbb{C}^{m \times n}$를 QR할 때, Q는 unitary 행렬이다. (실수 field에서 정의되었다면 orthogonal 행렬) 그렇다면 $Q^*A = R$이어야 하고, 따라서 $Q^* = (Q_n Q_{n-1} \cdots Q_1)$ 임을 유도할 수 있다. 즉, $Q = (Q_1^* Q_2^* \cdots Q_n^*)$와 같이 표현되어져야 한다. 각각의 Q_i들도 unitary 함은 후술.

Householder Method는 unitary(or orthogonal)한 행렬들을 바탕으로 triangular 행렬을 찾는 **Orthogonal Triangularization**이라 할 수 있다.

QR with Householder Reflections

Basic Idea of QR from Householder



어떻게 행렬 Q_k를 만들 수 있을까?

다음을 만족시키는 Blocked Matrix를 Q_k로 설정하자.

각 행렬의 크기는 $Q_k \in \mathbb{C}^{m\times m}$, $I \in \mathbb{R}^{(k-1)\times(k-1)}$, $F \in \mathbb{C}^{(m-k+1)\times(m-k+1)}$이다.

$$Q_k = \begin{array}{c} \\ k-1 \\ m-k+1 \end{array} \overset{\begin{array}{cc} k-1 & m-k+1 \end{array}}{\begin{bmatrix} I & O \\ O & F \end{bmatrix}}$$

각 sub matrix의 역할을 살펴보자.

- I의 역할

  $\prod_{i-1}^{k-1} Q_i A$로 인하여, 행렬의 $(k-1)$번째까지의 열은 모두 0으로의 교체가 완료되었으므로 Q_k에 의해 변화가 발생하면 안된다. 따라서 $I \in \mathbb{R}^{(k-1)\times(k-1)}$가 사용되었다.

- F의 역할

  F는 행렬의 k번째 열의 대각성분 아래 성분들을 0으로 교체하는 역할을 한다. $(k+1)$~ 번째 열의 변형은 신경쓰지 말자. Q_i $(i \geq k+1)$가 바꿔줄 것이다.

QR with Householder Reflections

Basic Idea of QR from Householder

F를 찾는 아이디어

Q_k의 sub matrix F를 찾아보자. 행렬 A의 k번째 열에서 대각성분 이하의 성분들만 분리하여 x라 정의하자. 즉, $x = \left(\mathrm{A}_{k,k} \middle| A_{k+1,k} \middle| \cdots \middle| A_{m,k}\right)^{\mathrm{T}}$. F에 의해 선형변환된 벡터 Fx는 $A^{(k)}$의 대각성분, 즉, Fx의 첫 번째 성분을 제외한 모든 성분이 0으로 변형되어야 하며, 기존 벡터 x의 norm을 보존해야 한다. 즉, $||x|| = ||Fx||$.

$$x = \begin{bmatrix} \times \\ \times \\ \times \\ \vdots \\ \times \end{bmatrix} \quad \xrightarrow{F} \quad Fx = \begin{bmatrix} \|x\| \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \|x\| e_1.$$

참고

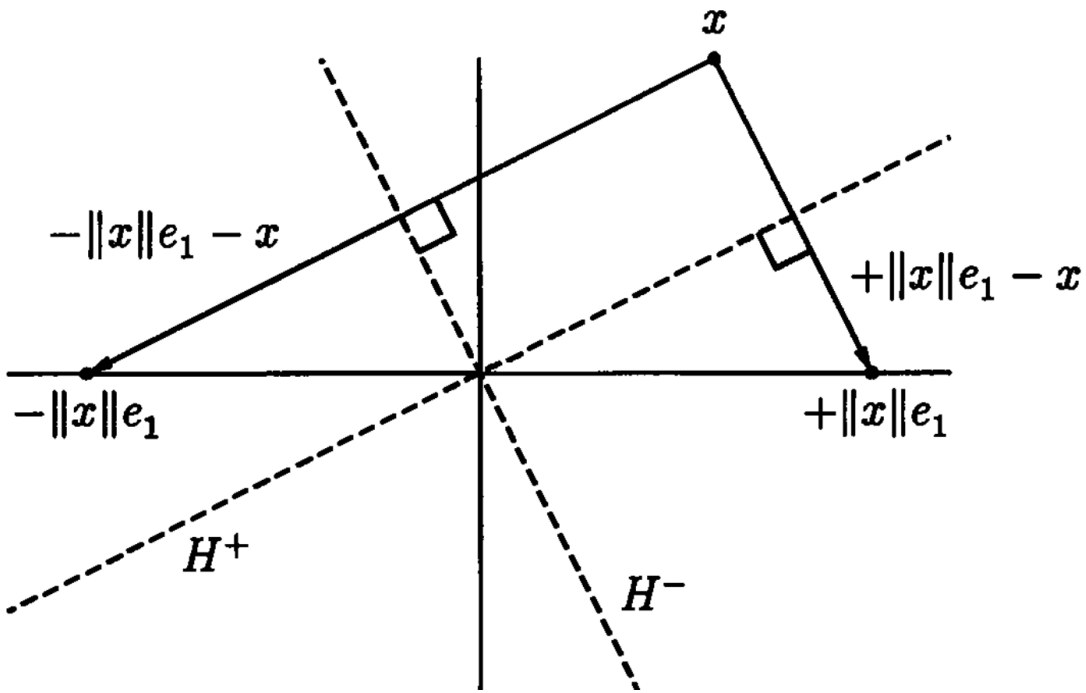
norm을 보존하는 선형변환은 rotation이나 reflection의 기능을 수행한다. 즉 F와, I와 F만을 포함한 각각의 Q_i 역시 orthogonal (or unitary) 할 것이라는 직관을 얻을 수 있다. 또한 위의 norm-preserving matrix로부터, reflection을 통해 QR을 찾는 방법과 rotation을 통해 QR을 찾는 방법이 모두 가능할 것임을 유추할 수 있다. 이는 각각 Householder reflection과 Givens Rotation이다.

QR with Householder Reflections

Basic Idea of QR from Householder

Householder Reflector, F

만약 2차원 벡터 x에서 두번째 성분을 0으로 바꾼 채 norm을 보존시킨다면, 새롭게 얻어진 벡터 $Fx = ||x||e_1$은 x축과 나란한 모양을 가질 것이다. 그런데 사실 Fx는 norm-preserving과 0으로의 교체 역할만 하면 되므로 Fx의 방향 자체는 상관이 없다. 따라서 $Fx = \pm||x||e_1$ 중 상황에 맞춰 적절한 sign을 선택해주면 된다.



QR with Householder Reflections

Basic Idea of QR from Householder

F를 찾기 위한 공식 유도

1. 벡터 x에 대하여, $\text{proj}_H x$를 구한다. (이하 $proj_H(\cdot) = P(\cdot)$라 함.)

$$\text{proj}_H x = \frac{vv^*}{v^*v}x$$

2. Hyperplane (2D에서는 직선) H에 대해 수직인 벡터 d를 구한다.

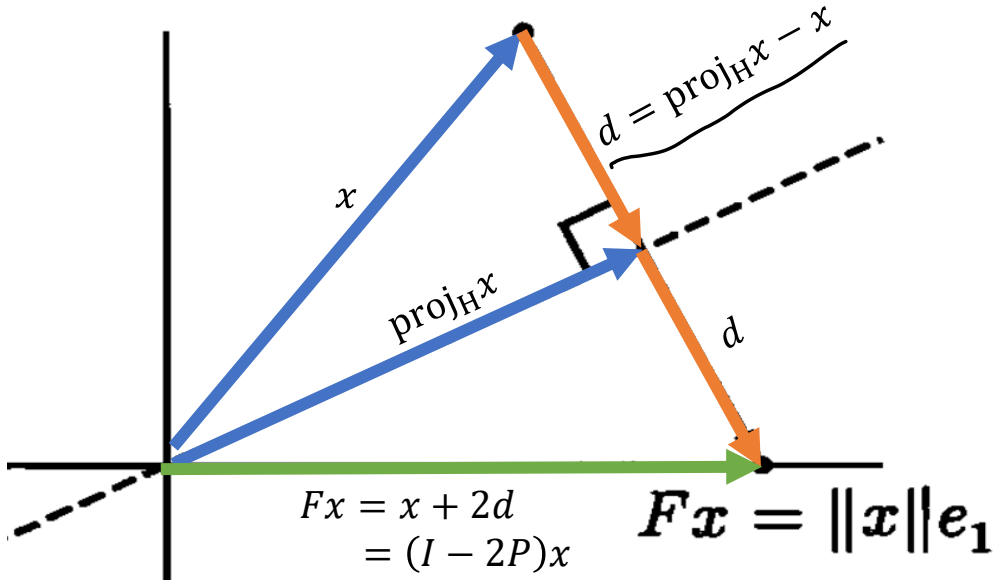
$$d = Px - x$$

3. 벡터 d가 H에서 멈추지 않고 reflection의 효과를 가지도록 두배로 연장한다. 이후, x와 합성하여 Fx를 찾아준다.

$$Fx = x + 2d = x + (2Px - 2x)$$
$$= -x + 2Px = -(I - 2P)x$$

4. 앞서 살펴보았듯, Fx의 부호는 $+, -$ 모두 가능하므로 부호를 정리해준다.

$$\therefore Fx = \pm(I - 2P)x = \pm(x - 2\text{proj}_H x)$$



QR with Householder Reflections

Application of Householder Method for QR Decomposition

Let $A = \begin{pmatrix} 12 & -51 & 4 \\ 6 & 167 & -68 \\ -4 & 24 & -41 \end{pmatrix}$. Let's denote the first column of A as $\mathbf{a}_1 = \begin{pmatrix} 12 \\ 6 \\ -4 \end{pmatrix}$.

Then the reflected x, namely Fx will be $||\mathbf{a}_1||\mathbf{e}_1 = \alpha_1\mathbf{e}_1 = \begin{pmatrix} \alpha_1 \\ 0 \\ 0 \end{pmatrix}$, where $\alpha_1 = ||\mathbf{a}_1|| = \sqrt{12^2 + 6^2 + (-4)^2} = 14$.

Now, $\mathbf{u} = \mathbf{a}_1 \pm \alpha_1\mathbf{e}_1 = \mathbf{a}_1 \pm F\mathbf{a}_1 = \begin{pmatrix} 12 \\ 6 \\ -4 \end{pmatrix} \pm \begin{pmatrix} 14 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 12 \pm 14 \\ 6 \\ -4 \end{pmatrix}$. To reduce the cancellation error, it is recommended to choose the sign that can make the first component as big as possible. Let's choose $+$. Then we have $\mathbf{u}_1 = \begin{pmatrix} 26 \\ 6 \\ -4 \end{pmatrix}$.

QR with Householder Reflections

Application of Householder Method for QR Decomposition

Let $\mathbf{u}_1$ get normalized by dividing itself with its norm and denote it as $\mathbf{v}_1$. Then we have $\mathbf{v}_1 = \frac{\mathbf{u}_1}{||\mathbf{u}_1||} = \frac{1}{\sqrt{182}} \begin{pmatrix} 13 & 3 & -2 \end{pmatrix}^T$. Therefore, We can get

$$Q_1 = I - \frac{2}{\sqrt{182}\sqrt{182}} \begin{pmatrix} 13 \\ 3 \\ -2 \end{pmatrix} \begin{pmatrix} 13 & 3 & -2 \end{pmatrix}$$

$$= I - \begin{pmatrix} \frac{13}{7} & \frac{3}{7} & -\frac{2}{7} \\ \frac{3}{7} & \frac{9}{91} & -\frac{6}{91} \\ -\frac{2}{7} & -\frac{6}{91} & \frac{4}{91} \end{pmatrix}$$

$$= \begin{pmatrix} -\frac{6}{7} & -\frac{3}{7} & \frac{2}{7} \\ -\frac{3}{7} & \frac{82}{91} & \frac{6}{91} \\ \frac{2}{7} & \frac{6}{91} & \frac{87}{91} \end{pmatrix}.$$

Now observe: $Q_1 A = \begin{pmatrix} -14 & -21 & 14 \\ 0 & \frac{2261}{13} & -\frac{854}{13} \\ 0 & \frac{252}{13} & -\frac{553}{13} \end{pmatrix}$.

QR with Householder Reflections

Application of Householder Method for QR Decomposition

Now take the $(1,1)$ minor, and then apply the process again to

$$M_{11} = \begin{pmatrix} \frac{2261}{13} & -\frac{854}{13} \\ \frac{252}{13} & -\frac{553}{13} \end{pmatrix}.$$

Let $\mathbf{a}_2 = \begin{pmatrix} \frac{2261}{13} \\ \frac{252}{13} \end{pmatrix}$. Then $\alpha_2 = ||\mathbf{a}_2|| = 175$.

Now, $\mathbf{u}_2 = \mathbf{a}_2 \pm \alpha_2 \mathbf{e}_2 = \begin{pmatrix} \frac{2261}{13} \\ \frac{252}{13} \end{pmatrix} \pm \begin{pmatrix} 175 \\ 0 \end{pmatrix}$. Let's choose $+$ and normalize $\mathbf{u}_2$. Then we have $\mathbf{v}_2 = \frac{1}{5\sqrt{13}} \begin{pmatrix} 18 & 1 \end{pmatrix}^T$. Therefore we can get

$$\begin{aligned} F_2 &= I - \frac{2}{5\sqrt{13} \cdot 5\sqrt{13}} \begin{pmatrix} 18 \\ 1 \end{pmatrix} \begin{pmatrix} 18 & 1 \end{pmatrix} \\ &= I - \frac{2}{25 \cdot 13} \begin{pmatrix} 324 & 18 \\ 18 & 1 \end{pmatrix} \\ &= \begin{pmatrix} -\frac{323}{325} & -\frac{36}{325} \\ -\frac{36}{325} & \frac{323}{325} \end{pmatrix}. \end{aligned}$$

QR with Householder Reflections

Application of Householder Method for QR Decomposition

Therefore, from $Q_2 = \left(\begin{array}{c|c} I_{1\times 1} & 0 \\ \hline 0 & F_2 \end{array}\right)$, we can get $Q_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -\frac{323}{325} & -\frac{36}{325} \\ 0 & -\frac{36}{325} & \frac{323}{325} \end{pmatrix}$.

$$\therefore Q_2 Q_1 A = \begin{pmatrix} -14 & -21 & 14 \\ 0 & -175 & 70 \\ 0 & 0 & -35 \end{pmatrix}.$$

Now for the last column, we have nothing to transform as the upper-triangular matrix has already made. Let's verify whether $Q{=}Q_1^* Q_2^*$ is unitary and such QR really recovers A. (Not shown above, both Q_1 and Q_2 are unitary.)

$$\begin{aligned} Q = Q_1^* Q_2^* &= \begin{pmatrix} -\frac{6}{7} & -\frac{3}{7} & \frac{2}{7} \\ -\frac{3}{7} & \frac{82}{91} & \frac{6}{91} \\ \frac{2}{7} & \frac{6}{91} & \frac{87}{91} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -\frac{323}{325} & -\frac{36}{325} \\ 0 & -\frac{36}{325} & \frac{323}{325} \end{pmatrix} \\ &= \begin{pmatrix} -\frac{6}{7} & \frac{69}{175} & \frac{58}{175} \\ -\frac{3}{7} & \frac{82}{91} & \frac{6}{91} \\ \frac{2}{7} & \frac{6}{91} & \frac{87}{91} \end{pmatrix} = \begin{pmatrix} -\frac{6}{7} & \frac{69}{175} & \frac{58}{175} \\ -\frac{3}{7} & -\frac{158}{175} & -\frac{6}{175} \\ \frac{2}{7} & -\frac{6}{35} & \frac{33}{35} \end{pmatrix}, \end{aligned}$$

which is unitary as $Q^* = Q^{-1}$.

QR with Householder Reflections

Application of Householder Method for QR Decomposition

Now for the last column, we have nothing to transform as the upper-triangular matrix has already made. Let's verify whether $Q{=}Q_1^*Q_2^*$ is unitary and such QR really recovers A. (Not shown above, both Q_1 and Q_2 are unitary.)

$$Q = Q_1^*Q_2^* = \begin{pmatrix} -\frac{6}{7} & -\frac{3}{7} & \frac{2}{7} \\ -\frac{3}{7} & \frac{82}{91} & \frac{6}{91} \\ \frac{2}{7} & \frac{6}{91} & \frac{87}{91} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -\frac{323}{325} & -\frac{36}{325} \\ 0 & -\frac{36}{325} & \frac{323}{325} \end{pmatrix}$$
$$= \begin{pmatrix} -\frac{6}{7} & \frac{69}{175} & \frac{58}{175} \\ -\frac{3}{7} & \frac{82}{91} & \frac{6}{91} \\ \frac{2}{7} & \frac{6}{91} & \frac{87}{91} \end{pmatrix} = \begin{pmatrix} -\frac{6}{7} & \frac{69}{175} & \frac{58}{175} \\ -\frac{3}{7} & -\frac{158}{175} & -\frac{6}{175} \\ \frac{2}{7} & -\frac{6}{35} & \frac{33}{35} \end{pmatrix},$$

which is unitary as $Q^* = Q^{-1}$.

Also,

$$QR = Q(Q_2Q_1A) = \begin{pmatrix} -\frac{6}{7} & \frac{69}{175} & \frac{58}{175} \\ -\frac{3}{7} & -\frac{158}{175} & -\frac{6}{175} \\ \frac{2}{7} & -\frac{6}{35} & \frac{33}{35} \end{pmatrix} \begin{pmatrix} -14 & -21 & 14 \\ 0 & -175 & 70 \\ 0 & 0 & -35 \end{pmatrix}$$
$$= \begin{pmatrix} 12 & -51 & 4 \\ 6 & 167 & -68 \\ -4 & 24 & -41 \end{pmatrix}.$$

■

QR with Householder Reflections

Cost of Householder Reflections

```r
# Householder Reflections
Householder <- function(mat){
  options(digits=22)
  if (ncol(mat)-1 <= 1){
    return(mat)
  }

  reflectors <- list()
  for (i in 1:(ncol(mat)-1)){
    a <- mat[i:nrow(mat),i]
    a_norm <- norm(a, type="2")
    a_rotated <- c(a_norm, rep(0,nrow(mat)-i))

    if (a[1]<0){
      u <- a - a_rotated
    } else {
      u <- a + a_rotated
    }

    v <- u / norm(u, type="2")
    outer <- 2*(v %*% t(v))
    I <- matrix(0, nrow=nrow(outer), ncol=ncol(outer))
    diag(I) <- 1
    Fmat <- I - outer

    blocked_matrix <- function(mat, Fmat){
      I_11 <- matrix(0, nrow=nrow(mat)-nrow(Fmat), ncol=nrow(mat)-ncol(Fmat))
      diag(I_11) <- 1

      Q_top <- cbind(I_11, matrix(0, nrow=nrow(I_11), ncol=ncol(Fmat)))
      Q_bottom <- cbind(matrix(0, nrow=nrow(Fmat), ncol=ncol(I_11)), Fmat)
      Q <- rbind(Q_top, Q_bottom)
      return(Q)
    }
    reflectors[[i]] <- blocked_matrix(mat, Fmat)
    mat <- blocked_matrix(mat, Fmat) %*% mat
  }

  I <- diag(nrow(mat))
  for (q in reflectors){
    I <- I %*% t(q)
  }
  if (mat[1,1]<0){
    I <- -1*I
    mat <- -1*mat
  }
  qr <- list(Q=I, R=mat)
  return(qr)
}
```

– k번째 열을 위한 F 구하기

$Fx = (I - 2P)x \Longrightarrow$ 각 $m - k + 1$개의 원소 이용

$\Longrightarrow 2P : 2(m - k + 1)(m - k + 1)$

$\Longrightarrow 2P : 2(m - k + 1)(m - k + 1)$

$\Longrightarrow (I - 2P)x : (m - k + 1)(m - k + 1)$

– 총 열의 개수= n

– Cost

$$\sum_{k=1}^{n} 4(m - k + 1)(m - k + 1) \sim O(mn^2)$$

4.QR with Givens Rotation

반사
회전

QR with Givens Rotation

Basic Idea of QR from Givens Rotation

앞서 Householder 파트에서 행렬을 QR 분해하기 위하여 unitary matrix 를 곱해 줌으로써 upper triangular 행렬인 R을 찾아낼 수 있음을 살펴보았다. 또한 그러한 unitary 행렬은 분해하고자 하는 행렬의 각 열벡터를 반사/회전 변환하여 대각성분 아래 성분들을 모두 0으로 바꾸었다.

이제 반사가 아닌 회전을 통해 위의 목적을 달성해보자. 행렬의 대각성분을 pivot으로하여 0으로 변환하고자 하는 대각 아래 성분과 결합해 2차원 열벡터로 구성한 뒤 시각화해보자.

e.g. 아래 행렬 A의 $(3,1)$번째 성분인 $\sqrt{3}$을 0으로 변환

$$A = \begin{pmatrix} 4 & 1 & 4 \\ 3 & 2 & 5 \\ 0 & 3 & 6 \end{pmatrix} \Longrightarrow x = (4,\ 3)^T$$

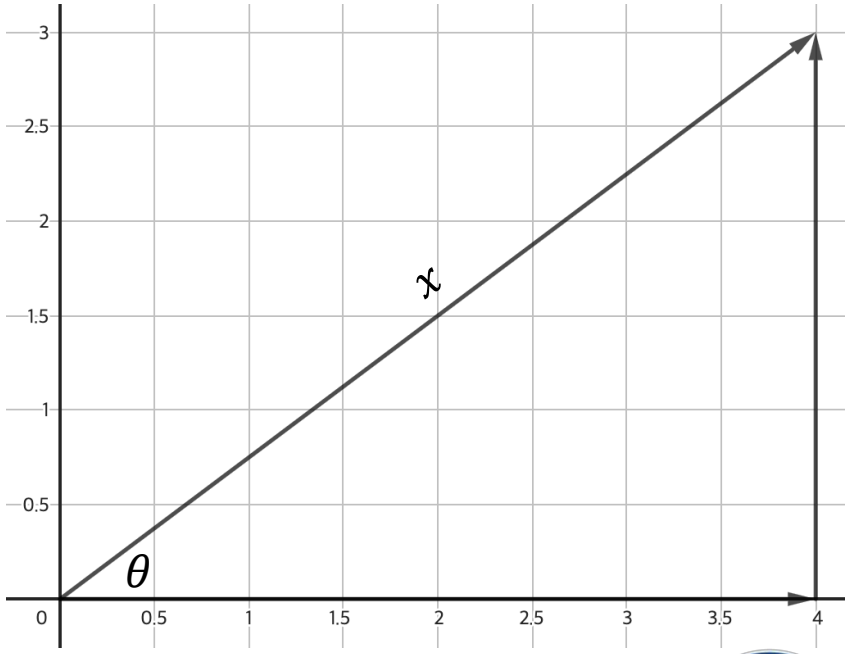
QR with Givens Rotation

Basic Idea of QR from Givens Rotation

벡터 x는 우측 Figure에서 대각선을 의미하고, 이를 바탕으로 직각 삼각형을 그릴 수 있다. 이제 x를 바탕으로 $\cos\theta$와 $\sin\theta$를 구해보자.

- $\cos\theta = \frac{||\text{proj}_{\text{X-axis}}x||}{||x||} = \frac{4}{5}$
- $\sin\theta = \frac{||\text{proj}_{\text{Y-axis}}x||}{||x||} = \frac{3}{5}$

어떻게 하면 벡터 x를 회전시켜, norm을 보존한 채 X축과 평행한 벡터로 수정할 수 있을까?



QR with Givens Rotation

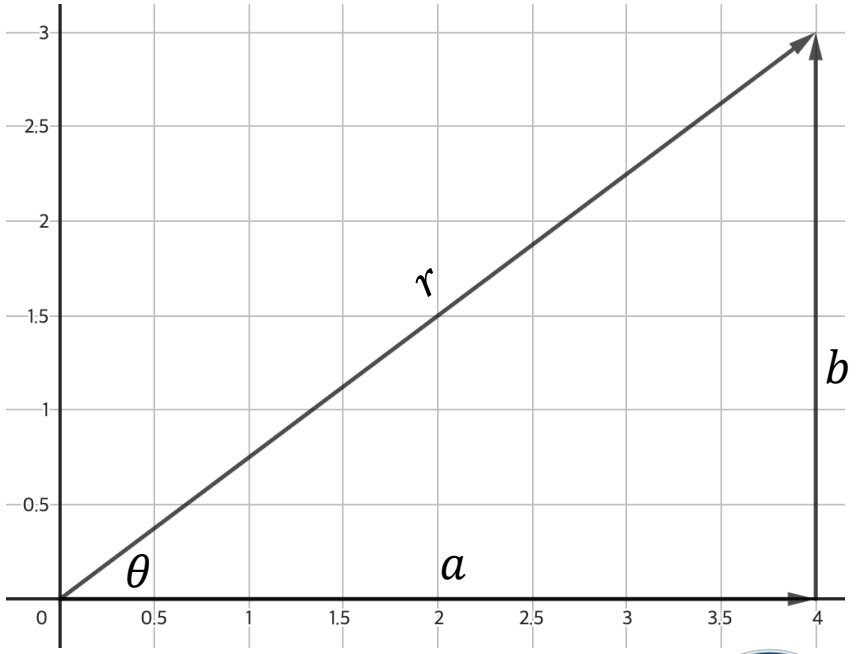
Basic Idea of QR from Givens Rotation

Norm-preserving Vector Rotation: along to X-axis

오른쪽 삼각형을 생각해보자. 2차원 벡터 x'이 벡터 x를 회전시켜 X축과 평행하게 새롭게 만들어낸 벡터라면, $||x'|| = r$일 것이며, $x' = (r, 0)^T$의 형태일 것이다. 이는 다음과 같은 간단한 2×2 행렬(즉, 선형변환)을 통해 얻어낼 수 있는데, 이 행렬을 Givens Matrix, 아래 행렬을 통한 회전 변환을 Givens Rotation이라 부른다.

Givens Matrix (for Givens Rotation)

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



QR with Givens Rotation

Basic Idea of QR from Givens Rotation

Givens Matrix의 원리

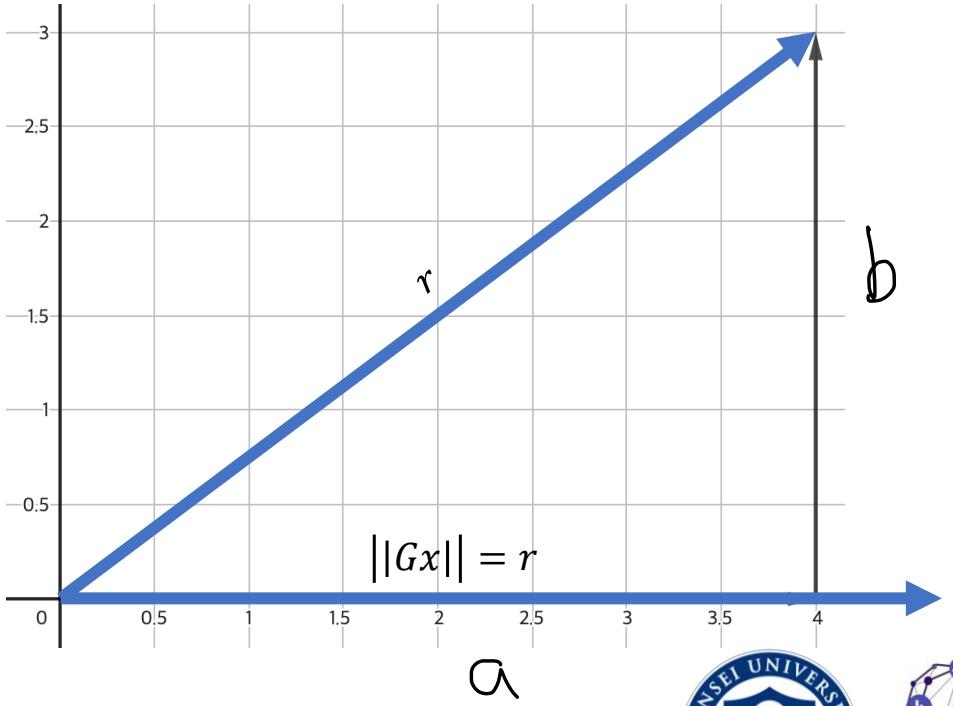
Givens Matrix G에 임의의 행렬 $x = (a, b)^T$를 곱한다고 하자.

다음 계산 과정을 통해 G의 변환원리를 이해할 수 있다.

뿐만 아니라 G의 열벡터 간 내적과, $\cos^2\theta + \sin^2\theta = 1$을 통해 G가 unitary한 행렬임도 확인할 수 있다.

$$Gx = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a\cos\theta + b\sin\theta \\ -a\sin\theta + b\cos\theta \end{bmatrix}$$

$$= \begin{bmatrix} a\frac{a}{r} + b\frac{b}{r} \\ -a\frac{b}{r} + b\frac{a}{r} \end{bmatrix} = \begin{bmatrix} \frac{a^2+b^2}{r} \\ \frac{-ab+ab}{r} \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}$$



QR with Givens Rotation

Basic Idea of QR from Givens Rotation

Givens Matrix의 확장

Givens Matrix는 기본적으로 2차원에서의 벡터 변환을 바탕으로 한다. 그러나 행렬 A가 가령 3×3이라면, 2×2 크기의 G를 곱할 수 없을 것이다. 이런 경우 다음의 규칙을 따라 G를 확장하면 된다.

e.g. 3×3 행렬 A의 $A_{2,1}$을 0으로 변환하는 과정

- $i = 2,\ j = 1$이라 설정.
- G를 행렬 A와 동일한 크기의 행렬이라 가정.
- $G_{i,i} = G_{1,1} = \cos\theta\,,\ G_{j,i} = G_{1,2} = \sin\theta$
  $G_{i,j} = G_{2,1} = -\sin\theta\,,\ G_{j,j} = G_{2,2} = \cos\theta$
  나머지 대각성분은 모두 1, 기타 성분은 0

$$G = \begin{pmatrix} c & s & 0 \\ -s & c & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

확장된 G도 unitary matrix일까?

확장된 G도 unitary (or orthogonal)하다.

$\sin^2\theta + \cos^2\theta = 1$과 각 열의 내적이 0임을 통해, 혹은 $G^T = G^{-1}$임을 통해 확인할 수 있다.

QR with Givens Rotation

Basic Idea of QR from Givens Rotation

QR Decomposition with G: how to find Q, R

앞선 예시를 통해 Givens Matrix를 통한 0으로의 성분 교체는 한번에 하나씩 밖에 되지 않음을 알 수 있었다. 그렇다면 3×3 크기의 행렬 A를 생각해보자. 0 성분이 없다면, 총 3개의 G를 곱함으로써 A를 upper triangular 행렬로 변환할 수 있을 것이다. 즉, 다음과 같다.

$$G_3 G_2 G_1 A = R$$

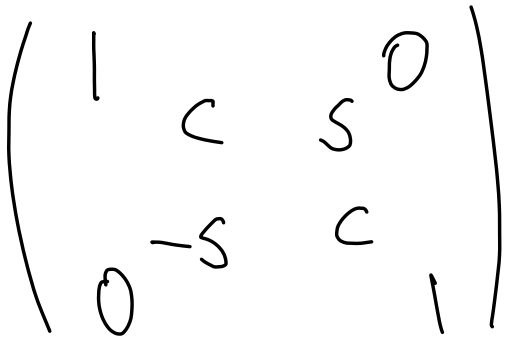
또한 우리는 각각의 G_i가 unitary 행렬임을 확인하였다. 따라서 다음과 같은 방법을 통해 Q를 찾아낼 수 있다.

$$R = Q^* A = (G_3 G_2 G_1) A$$

$$Q = (Q^*)^* = (G_3 G_2 G_1)^* = G_1^* G_2^* G_3^*$$

G의 순서는 중요할까?

예를 들어 $A_{2,1}$을 변환하기 위한 G_1과 $A_{3,1}$을 변환하기 위한 G_2 중 무엇을 먼저 진행해야할까? 정답은 "중요하지 않다"이다. 어차피 Givens Matrix는 한번에 한 성분밖에 0으로 바꾸지 못하기 때문에 어떤 성분을 먼저 변환할지는 중요하지 않다. 상황에 맞는 적절한 G만 찾아주면 된다.



$s^2 + c^2 = 1$

QR with Givens Rotation

Application of Givens Rotation for QR Decomposition

Let $A = \begin{pmatrix} 12 & -51 & 4 \\ 6 & 167 & -68 \\ -4 & 24 & -41 \end{pmatrix}$. First we need to form a rotation matrix that will zero $A_{21} = 6$. Let's form this matrix using the Givens rotation method, and call it G_1.

Pick the diagonal entry of A_{21}'s column as a pivot and let them construct a new vector $x_1 = \begin{pmatrix} 12 & 6 \end{pmatrix}^T$, to point along the X axis. By the Givens rotation matrix's law, let's create the unitary Givens rotation matrix, G_1:

$$G_1 = \begin{pmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} & 0 \\ -\frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Then we have $G_1 A$ as following:

$$G_1 A = \begin{pmatrix} 6\sqrt{5} & 13\sqrt{5} & -12\sqrt{5} \\ 0 & 77\sqrt{5} & -28\sqrt{5} \\ -4 & 24 & -41 \end{pmatrix}$$

QR with Givens Rotation

Application of Givens Rotation for QR Decomposition

Likewise, let's create a new vector $x_2 = \begin{pmatrix} 6\sqrt{5} & -4 \end{pmatrix}^T$.

Then we have G_2 and G_2G_1A as:

$$G_2 = \begin{pmatrix} \frac{3\sqrt{5}}{7} & 0 & -\frac{2}{7} \\ 0 & 1 & 0 \\ \frac{2}{7} & 0 & \frac{3\sqrt{5}}{7} \end{pmatrix} \Longrightarrow G_2G_1A = \begin{pmatrix} 14 & 21 & -14 \\ 0 & 77\sqrt{5} & -28\sqrt{5} \\ 0 & 14\sqrt{5} & -21\sqrt{5} \end{pmatrix}.$$

At last, to zero $(G_2G_1A)_{32}$, we can make a vector $x_3 = \begin{pmatrix} 77\sqrt{5} & 14\sqrt{5} \end{pmatrix}^T$. Then we have G_3 and $G_3G_2G_1A$ as:

$$G_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{11\sqrt{5}}{25} & \frac{2\sqrt{5}}{25} \\ 0 & -\frac{2\sqrt{5}}{25} & \frac{11\sqrt{5}}{25} \end{pmatrix} \Longrightarrow R = G_3G_2G_1A = \begin{pmatrix} 14 & 21 & -14 \\ 0 & 175 & -70 \\ 0 & 0 & -35 \end{pmatrix}.$$

Plus, from $Q^* = G_3G_2G_1$ we can get Q as follows:

$$Q^* = G_3G_2G_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{11\sqrt{5}}{25} & \frac{2\sqrt{5}}{25} \\ 0 & -\frac{2\sqrt{5}}{25} & \frac{11\sqrt{5}}{25} \end{pmatrix} \begin{pmatrix} \frac{3\sqrt{5}}{7} & 0 & -\frac{2}{7} \\ 0 & 1 & 0 \\ \frac{2}{7} & 0 & \frac{3\sqrt{5}}{7} \end{pmatrix} \begin{pmatrix} \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} & 0 \\ -\frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\therefore Q^* = \begin{pmatrix} \frac{6}{7} & \frac{3}{7} & -\frac{2}{7} \\ -\frac{69}{175} & \frac{158}{175} & \frac{6}{35} \\ \frac{58}{175} & -\frac{6}{175} & \frac{33}{35} \end{pmatrix}$$

$$\Longrightarrow QR = \begin{pmatrix} \frac{6}{7} & -\frac{69}{175} & \frac{58}{175} \\ \frac{3}{7} & \frac{158}{175} & -\frac{6}{175} \\ -\frac{2}{7} & \frac{6}{35} & \frac{33}{35} \end{pmatrix} \begin{pmatrix} 14 & 21 & -14 \\ 0 & 175 & -70 \\ 0 & 0 & -35 \end{pmatrix} = \begin{pmatrix} 12 & -51 & 4 \\ 6 & 167 & -68 \\ -4 & 24 & -41 \end{pmatrix} = A. \blacksquare$$

QR with Givens Rotation

Computational Method to get Givens Matrix

Instability of Givens Rotation

컴퓨터를 통해 Givens Matrix를 찾기 위해서, 컴퓨터는 벡터 $x = (a \;\; b)^T$에 대하여

$$\theta' = \arctan(b/a)$$

를 구한 뒤, 아래와 같은 규칙에 따라 θ를 찾는다. ($0 \leq \theta < 2\pi$에 위치하도록)

- 벡터 x가 1사분면에 있는 경우: $\theta = \theta'$
- 벡터 x가 2사분면에 있는 경우: $\theta = \pi + \theta'$
- 벡터 x가 3사분면에 있는 경우: $\theta = -pi + \theta'$
- 벡터 x가 4사분면에 있는 경우: $\theta = \theta'$

이를 통해 $\sin\theta$와 $\cos\theta$를 찾으므로 계속해서 반올림 오차가 발생할 수밖에 없다.

QR with Givens Rotation

Cost of Givens Rotation

```r
# Givens Rotation
Givens <- function(mat){
  options(digits=22)
  G_list <- list()
  ind <- 1
  for (i in 2:nrow(mat)){
    for (j in 1:(ncol(mat)-1)){
      if ((mat[i,j]!=0)&(i>j)){
        x <- c(mat[j,j], mat[i,j])
        theta <- atan2(x[2], x[1])
        c <- cos(theta)
        s <- sin(theta)

        G <- diag(nrow(mat))
        G[j,j] <- c
        G[j,i] <- s
        G[i,j] <- -s
        G[i,i] <- c

        G_list[[ind]] <- G
        mat <- G %*% mat
        ind <- ind+1
      }
      else{
        next
      }
    }
  }
  Q_uni <- diag(nrow(mat))
  for (g in G_list){
    Q_uni <- g %*% Q_uni
  }

  Q <- t(Q_uni)
  R <- mat
  result <- list(Q=Q,R=mat)
  return(result)
}

```

– 1번째 열의 대각 아래 성분 0으로 변환

Reduce to zero: m − 1개의 원소
Row operations: 각 n − 1번 (행의 1번째 원소는 0으로 바뀜)

– k번째 열에 대한 일반화

Reduce to zero: m − k개의 원소
Row operations: 각 n − k번

– Cost

$$\sum_{k=1}^{n}(m-k)(n-k) = \frac{mn^2}{2} + T(m,n) \sim O(2mn^2) = O(mn^2)$$

Comparison of the Three Methods

Gram-Schmidt

$$QR = \begin{bmatrix} 0.857142857142857 0952762 & -0.394285714285714 2394021 & -0.331428571428571 23919910 \\ 0.428571428571428 5476381 & 0.902857142857142 6915610 & 0.034285714285713 55883541 \\ -0.285714285714285 6984254 & 0.171428571428571 4024019 & -0.942857142857143 06015507 \end{bmatrix} \begin{bmatrix} 14 & 20.999999999999964 4729 & -14.000000000000000 0000 \\ 0 & 175.000000000000284 2171 & -69.999999999999857 8915 \\ 0 & 0.000000000000000 0000 & 34.999999999999928 9457 \end{bmatrix}$$

Householder

$$QR = \begin{bmatrix} 0.857142857142856 9842539 & -0.394285714285714 1283798 & -0.331428571428571 40573255 \\ 0.428571428571428 5476381 & 0.902857142857142 5805387 & 0.034285714285714 28048038 \\ -0.285714285714285 6984254 & 0.171428571428571 4024019 & -0.942857142857142 94913276 \end{bmatrix} \begin{bmatrix} 13.999999999999982 2364 & 21.000000000000035 5271 & -14.000000000000017 7636 \\ 7.783517420333595848586e-16 & 174.99999999999971 5783 & -69.999999999999857 8915 \\ 5.335902659890751938656e-16 & -3.552713678800500929356e-15 & 35.000000000000000 0000 \end{bmatrix}$$

Givens

$$QR = \begin{bmatrix} 0.857142857142857 2062985 & -0.394285714285714 2394021 & 0.331428571428571 46124371 \\ 0.428571428571428 6031492 & 0.902857142857142 8025833 & -0.034285714285714 28048038 \\ -0.285714285714285 6984254 & 0.171428571428571 4301575 & 0.942857142857142 83811046 \end{bmatrix} \begin{bmatrix} 14 & 21.000000000000071 0543 & -14.000000000000017 7636 \\ 0 & 175.000000000000000 0000 & -70.000000000000000 0000 \\ 0 & 0.000000000000000 0000 & -34.999999999999928 9457 \end{bmatrix}$$

Comparison of the Three Methods

	Gram-Schmidt	Householder	Givens
장점	• 간단하게 구현 가능	• 반올림 오차가 누적되지 않아 안정적 • 행렬이 클 수록 더 정확한 결과	• 간단한 계산 • 빠른 수행속도
단점	• 열들이 선형종속에 가까울수록 오차 증폭	• Householder 행렬 자체의 계산 과정이 복잡함	• 행렬이 커질수록 계산 복잡도가 증가 • 계속되는 반올림으로 안정적이지 못한 결과

5.Least-squares problem

Least-squares problem

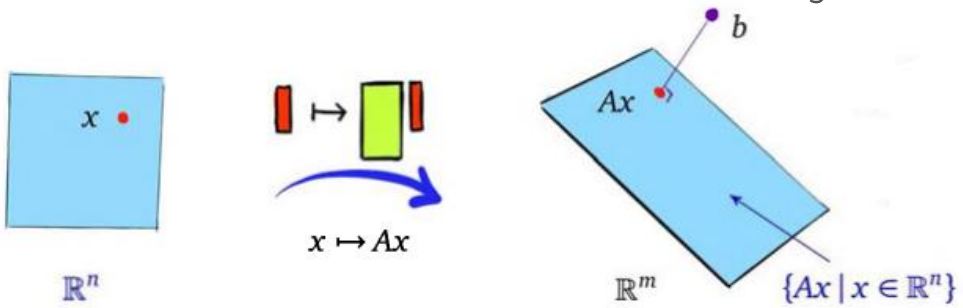
Least-squares problem?

Figure 1

$$\begin{bmatrix} 60 & 5.5 & 1 \\ 65 & 5.0 & 0 \\ 55 & 6.0 & 1 \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 66 \\ 74 \\ 78 \\ \vdots \end{bmatrix}$$

$$A \qquad \mathbf{x} = \mathbf{b}$$

Figure 2



- $A \in \mathbb{R}^{m \times n}, m \geq n, b \in \mathbb{R}^m$ 위와 같은 A, x, b matrix가 있을 때, 변수보다 방정식이 더 많게 됨
  → Ax=b 만족하는 Unique한 solution이 존재하지 않을 수 있음
- 정확한 해를 구하기 어려운 경우, $argmin_x \| Ax - b \|_2$ 를 만족하는 x를 구하는 것
- 기하학적으로 matrix x를 Ax로 변환 후 생성되는 평면과 b vector 사이의 거리로 생각 가능

Least-squares problem

Polynomial Examples: m개의 point $x_1, x_2 \cdots x_m$ & data $y, y_2 \cdots y_m$의 관계를 설명하는 함수를 찾는 두 방법

1) Interpolation

· m-1 degree를 갖는 아래와 같은 함수를 만든 후, 계수 c를 찾는 방법

$$p(x) = c_0 + c_1 x + \cdots + c_{m-1} x^{m-1}$$

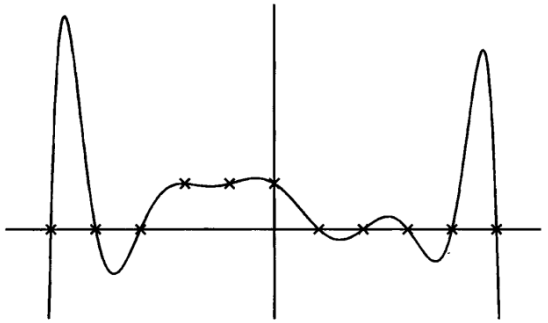


Figure 3

overfitting

· point의 수가 늘어날수록 다항식 차수가 늘어남
  → computation cost가 커지는 단점,
· 끝 point에서 큰 진동 나타남

2) Least Squares Fitting

· degree를 n으로 줄인 후, 정확한 solution이 아닌,
$\sum_{i=1}^{m} |p(x_i) - y_i|^2$ 최소화하는 c를 찾음

$$p(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}$$

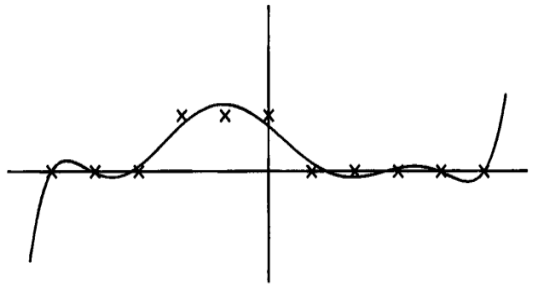


Figure 4

· 전반적인 behavior 더 잘 capture
· less sensitive to perturbation

Least-squares problem

Least-squares via Normal Equations

1) null-space 활용

· error $\coloneqq e = Ax - b$ 는 range of A (the span of the columns of A)와 orthogonal 함

· $A^T e = 0 \rightarrow A^T(Ax - b) = 0$ 성립, x에 대해 정리하면 아래와 같은 normal equation 도출

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

2) Cost function 활용

· error $\coloneqq e = Ax - b$ 의 제곱합 최소화

$$\boldsymbol{f(x) = (Ax - b)^T (Ax - b) = x^T A^T Ax - 2x^T A^T b + b^T b}$$

$$\boldsymbol{\nabla f(x) = 2A^T Ax - 2A^T b \ (set \ \nabla f = 0)}$$

$$\boldsymbol{A^T Ax = A^T b}$$

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

→ Cholesky 분해 사용해 효율적으로 계산 후, Least-squares 문제 해결

Least-squares problem

Least-squares via Normal Equations

Cholesky factorization

$x^T A x \geq 0$

· 행렬 A가 symmetric, square, semi-positive definite일 때, 다음과 같이 분해 (L is lower-triangular matrix)

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

Figure 5

$$A = LL^T = \begin{bmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{bmatrix} \begin{bmatrix} L_{11} & L_{21} & L_{31} \\ 0 & L_{22} & L_{32} \\ 0 & 0 & L_{33} \end{bmatrix} \Rightarrow \begin{bmatrix} L_{11}^2 & & (\text{대칭}) \\ L_{21}L_{11} & L_{21}^2 + L_{22}^2 & \\ L_{31}L_{11} & L_{31}L_{21} + L_{32}L_{22} & L_{31}^2 + L_{32}^2 + L_{33}^2 \end{bmatrix} \rightarrow L = \begin{bmatrix} \sqrt{a_{11}} & 0 & 0 \\ a_{21}/L_{11} & \sqrt{a_{22} - L_{21}^2} & 0 \\ a_{31}/L_{11} & (a_{32} - L_{31}L_{21})/L_{22} & \sqrt{a_{33} - L_{31}^2 - L_{32}^2} \end{bmatrix}$$

· 위 결과와 matrix A의 원소를 1:1로 비교하여 L 구하면, 가장 오른쪽 matrix L을 얻을 수 있음

· 이를 일반화하면 다음과 같음

$$L_{jj} = \sqrt{a_{jj} - \sum_{k=1}^{j-1} L_{jk}^2}, \qquad L_{ij} = \frac{1}{L_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} L_{ik} L_{jk} \right)$$

Least-squares problem

$A^*Ax = A^*b$

$(R^*R)x = A^*b \longrightarrow R^*w = A^*b$

Algorithm via Normal Equations

1) Form the matrix A^*A and the vector A^*b
2) Compute the Cholesky factorization $A^*A = R^*R$ (A is square, symmetric and semi-positive definite)
3) Solve the lower-triangular system $R^*w = A^*b$ for w by forward-substitution (set $w = Rx$)
4) Solve the upper-triangular system $Rx = w$ for x by back-substitution

- cost $\approx mn^2 + \frac{1}{3}n^3$ flops
- 대부분의 경우 잘 작동, cholesky사용해 빠름
- decimal point error 발생 가능 & A가 poor conditioned 인 경우 문제 발생 가능
  → QR factorization 사용하여 보완

Least-squares problem

$P = A(A^*A)^{-1}A^*$

Least-squares via QR Factorization

- A = QR ($Q \in \mathbb{C}^{m \times n}$) 분해 후, subspace R(A) = R(Q) 성질 활용하여 orthogonal projector $P = QQ^T$ 구함
- Let $x_* = argmin_x \|Ax - b\|_2$ then

$$\boldsymbol{Ax_* = Pb}$$

$$\boldsymbol{QRx_* = QQ^Tb}$$

$$\boldsymbol{Rx_* = Q^Tb}$$

- QR 분해를 하여 orthogonal, well-conditioned 한 특성을 얻음으로 앞서 normal equation에서 발생한 문제 해결 가능

Least-squares problem

Algorithm via QR Factorization

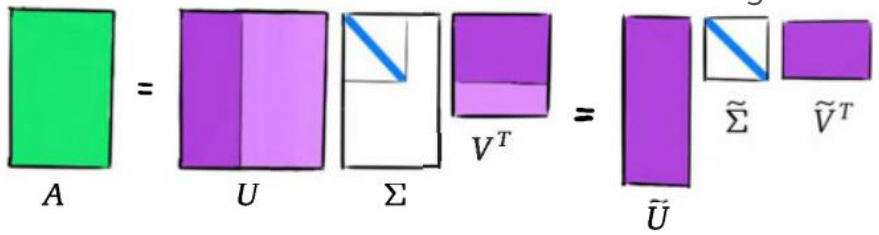
1) Compute the QR factorization $A = \hat{Q}\hat{R}$
2) Compute the orthogonal projector P := $\hat{Q}\hat{Q}^*$, since $Ax_* = Pb$, get $\hat{R}x_* = \hat{Q}^*b$
3) Compute the $y := \hat{Q}^*b$ and get $\hat{R}x_* = y$
4) Since $\hat{R}$ is an upper triangular matrix, find the solution of the equation $\hat{R}x_* = y$ for x by forward substitution

- cost $\approx 2mn^2 - \frac{2}{3}n^3$ flops
- numerical analyst 들이 least square문제를 풀기 위한 standard 방법으로 추천
- A가 full rank인 경우 많이 사용

Least-squares problem

Least-squares via SVD

Figure 6



$A \in \mathbb{R}^{m \times n}$
$\tilde{\boldsymbol{U}} \in \mathbb{R}^{m \times r}$
$\tilde{\boldsymbol{\Sigma}} \in \mathbb{R}^{r \times r}$
$\tilde{\boldsymbol{V}}^{\boldsymbol{T}} \in \mathbb{R}^{r \times n}$

· A가 full rank가 아닌 rank(A) = r인 경우 reduced SVD로 분해 후 사용

· A 분해 후 $\boldsymbol{Ax} = \boldsymbol{0}$ $\boldsymbol{iff}$ $\boldsymbol{V^T x} = \boldsymbol{0}$ 성립 → null space $N(A) = N(\tilde{V}^T)$ 성립, input $\boldsymbol{A} = \tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{V}}^{\boldsymbol{T}}$ to $\boldsymbol{A^T Ax} = \boldsymbol{A^T b}$ then,

$$\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{V}}^{\boldsymbol{T}}\boldsymbol{x} = \tilde{\boldsymbol{U}}^{\boldsymbol{T}}\boldsymbol{b}$$

· let $\boldsymbol{\omega} = \tilde{\boldsymbol{V}}^{\boldsymbol{T}}\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{r}}$ then, $\tilde{\boldsymbol{\Sigma}}\boldsymbol{\omega} = \tilde{\boldsymbol{U}}^{\boldsymbol{T}}\boldsymbol{b}$. This gives $\boldsymbol{\omega_i} = \frac{\boldsymbol{u_i^T b}}{\boldsymbol{\sigma_{ii}}}$

· multiply both sides of $\omega = \tilde{V}^T x$ by V. then,

$$\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{V}}^{\boldsymbol{T}}(\widetilde{\boldsymbol{V\omega}}) = \tilde{\boldsymbol{U}}^{\boldsymbol{T}}\boldsymbol{b}$$

· optimal solution is

$$\boldsymbol{x}_* = \tilde{\boldsymbol{V}}\boldsymbol{\omega}$$

Least-squares problem

$A^TAx = A^Tb$

Algorithm via SVD

1) Coupute the reduced SVD $A = \tilde{U}\tilde{\Sigma}\tilde{V}^T$
2) Compute the vector $\tilde{U}^T b$
3) Solve the diagonal system$\tilde{\Sigma}\omega = \tilde{U}^T b \; for \; \omega$
4) Set $x_* = \tilde{V}\omega$

- cost $\approx 2mn^2 + 11n^3$ flops
- SVD 계산에 많은 cost 소요, matrix A의 m >> n인 경우, cost ≈ QR
- A가 rank-deficient한 경우 사용

Least-squares problem

Comparison of Results

	Normal Equations	QR	SVD
Cost (flops)	$mn^2 + \frac{1}{3}n^3$	$2mn^2 - \frac{2}{3}n^3$	$2mn^2 + 11n^3$

- Speed만을 고려한다면 Normal Equations 방법이 best, but rounding error로 인해 unstable
- 일반적으로 full column rank인 경우 QR 방법이 best
- A가 rank-deficient 한 경우에는 SVD 방법이 좋음
- SVD에서 m >> n인 경우, cost가 QR과 유사함

END

References

[1] Darve, E., & Wootters, M. (2021). *Numerical Linear Algebra with Julia*. SIAM.

[2] Heath, M. T. (2018). *Scientific computing* (second). SIAM.

[3] Trefethen, L. N., & Bau , D. (1997). *Numerical linear algebra* (1st ed.). SIAM.

Figure

[1] [인공지능을 위한 선형대수] Least Squares Problem(최소자승법) 소개. (2019, September 24). WE GONNA MAKE IT. https://wegonnamakeit.tistory.com/33

[2] Darve, E., & Wootters, M. (2021). *Numerical Linear Algebra with Julia* (p.140). SIAM.

[3] Trefethen, L. N., & Bau , D. (1997). *Numerical linear algebra* (1st ed.) (p.79). SIAM.

[4] Trefethen, L. N., & Bau , D. (1997). *Numerical linear algebra* (1st ed.) (p.80). SIAM.

[5] 숄레스키 분해 - 공돌이의 수학정리노트 (Angelo's Math Notes). (n.d.). Angeloyeo.github.io. Retrieved July 26, 2023, from https://angeloyeo.github.io/2021/06/17/Cholesky_decomposition.html

[6] Darve, E., & Wootters, M. (2021). *Numerical Linear Algebra with Julia* (p.144). SIAM.