MATRIX DECOMPOSITION ALGORITHM

Homepage: https://sites.google.com/site/doc4code/

Email: sj6219@hotmail.com

2012/10/27

이 문서에서는 행렬을 분해하는 방법에 대해서 공부해 보겠다.

Ken Shoemake 가 작성한 소스를 분석해서, 그 원리를 설명하겠다.

소스는

https://sites.google.com/site/doc4code/source/MatrixDecomposition.zip

에 복사해 두었다.

그런데, 분석하기 위해서는 미리 알아두어야 할 선형 수학의 수준이 꽤 높다.

행렬과 관련한 기본 지식이 필요한데, 어차피 따로 공부해야 하니까 이 문서에서는 자세히 설명하지 않도록 하겠다.

그리고, 모르는 용어가 있으면 위키피디아나 구글을 검색하면 대부분 찾을 수 있을 것이다.

SPECTRAL DECOMPOSITION

spectral decomposition 은 주어진 **square matrix** A 를 분해해서 다음 식을 만족하는 **invertible matrix** U 와 **diagonal matrix** Λ 으로 나타내는 것이다.

$$A = U\Lambda U^{-1}$$

spectral decomposition 을 **eigendecomposition** 이라고 부르기도 한다.

$$\Lambda = \begin{bmatrix} \Lambda_{11} & 0 & \cdots & 0 \\ 0 & \Lambda_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \Lambda_{nn} \end{bmatrix}$$

$$U = \begin{bmatrix} U_{11} & U_{12} & \cdots & U_{1n} \\ U_{21} & U_{22} & \cdots & U_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ U_{n1} & U_{n2} & \cdots & U_{nn} \end{bmatrix}$$

이 때, $\Lambda_{11}, \Lambda_{22}, \cdots, \Lambda_{nn}$ 를 A 의 **eigenvalues** 라고 한다. 또, $\begin{bmatrix} U_{11} \\ U_{21} \\ \vdots \\ U_{n1} \end{bmatrix}, \begin{bmatrix} U_{12} \\ U_{22} \\ \vdots \\ U_{n2} \end{bmatrix}, \cdots, \begin{bmatrix} U_{1n} \\ U_{2n} \\ \vdots \\ U_{nn} \end{bmatrix}$를 A 의 **eigenvectors** 라고 한다.

일반적인 행렬의 **spectral decomposition** 를 하는 것은 쉽지 않고, 가능하지 않은 경우도 있다. 그러나, A 가 **symmetric matrix** 인 경우에는 항상 가능하고, 알고리즘이 더 간단하다. 그리고, 이때 U 는 **orthogonal matrix** 가 되는 성질이 있다.

여기서는 A 가 **symmetric matrix** 인 경우에 한하여, 분해하는 방법을 알아보겠다.

JACOBI METHOD (GIVENS ROTATION)

Jacobi Method 는 n×n **symmetric matrix** A 를 아래조건을 만족하는 **orthogonal matrix** U 와 **diagonal matrix** Λ로 분해할 때 사용된다.

$$A = U\Lambda U^{T}$$

Jacobi Method 에 대해서는 **[2] Matrix Computations** 에 자세하게 설명되어 있다.

p, q 가 n 이하의 자연수라고 하고, θ 를 실수라고 할 때, 행렬 $J(p, q, \theta)$를 아래와 같이 정의하자.

$$J(p,q,\theta) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & \cos\theta & \cdots & \sin\theta & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -\sin\theta & \cdots & \cos\theta & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}, \quad 1 \le p < q \le N$$

이런 행렬을 Givens Rotation 이라고 부른다.

즉, $J(p,q,\theta)$는 p 번째 축과 q 번째 축을 포함하는 평면에 대해서 θ만큼 회전하는 **orthogonal matrix** 이다.

Jacobi method 에는

$$X_0 = A$$

로 하고,

$$X_{i+1} = J(p_i, q_i, \theta_i)^T X_i J(p_i, q_i, \theta_i)$$

를 반복한다. 이 때, p_i, q_i, θ_i를 적당히 설정하면 X_∞는 **diagonal matrix** Λ에 수렴하게 된다.

그리고, U는 $U = J(p_0, q_0, \theta_0) J(p_1, q_1, \theta_1) \cdots J\left(p_\infty, q_\infty, \theta_\infty\right)$ 로 구하면 된다.

자, 이제 주어진 X_i에서 $J(p_i, q_i, \theta_i)$를 어떻게 계산하는 지 알아보자.

$$X_i = \begin{bmatrix} M_{11} & M_{12} & \cdots & M_{1n} \\ M_{12} & M_{22} & \cdots & M_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ M_{1n} & M_{2n} & \cdots & M_{nn} \end{bmatrix}$$

라고 표기하면, p_i, q_i는 다음 조건을 만족하는 자연수 p, q이면 된다.

$$1 \le p < q \le n$$

$$|M_{pq}| > 0$$

가능하면 이 조건을 만족하는 p, q 중 $|M_{pq}|$의 값이 최대인 값을 p_i, q_i의 값으로 선택하다.

또,

$$X_{i+1} = J(p_i, q_i, \theta_i)^T X_i J(p_i, q_i, \theta_i)$$

이고,

$$X_{i+1} = \begin{bmatrix} N_{11} & N_{12} & \cdots & N_{1n} \\ N_{12} & N_{22} & \cdots & N_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ N_{1n} & N_{2n} & \cdots & N_{nn} \end{bmatrix}$$

라고 표기할 때, $N_{pq} = 0$이 되는 θ_i를 구해야 한다.

$$N_{pq} = M_{pq}(\cos^2\theta_i - \sin^2\theta_i) + \left(M_{pp} - M_{qq}\right)\cos\theta_i \sin\theta_i$$

만약, $M_{pq} = 0$ 이라면 $\cos\theta_i, \sin\theta_i$의 값은 각각 1, 0 이면 된다.

아니라면,

$$\tan^2\theta_i + 2\tau\tan\theta_i - 1 = 0$$

$$\tau = \frac{\left(M_{qq} - M_{pp}\right)}{2M_{pq}}$$

이어야 한다.

그러므로,

$$\tan\theta_i = -\tau \pm \sqrt{1+\tau^2}$$

$$\cos\theta_i = \frac{1}{\sqrt{1+\tan^2\theta_i}}$$

$$\sin\theta_i = \tan\theta_i \cos\theta_i$$

가 된다.

알고리즘으로 구현하면,

```
float [n][n] JacobiMatrix(float M[n][n], int p, int q)
{
        float c;
        float s;
        float J[n][n];

        if M[p][q] != 0
                float tau = (M[q][q]-M[p][p]) / (2 * M[p][q])
                float t = (tau >= 0) ? 1/(tau + sqrt(1+tau*tau)) : -1/(-tau + sqrt(1+tau*tau))
                c = 1/sqrt(1+t*t)
                s = t * c
        else
                c = 1
                s = 0

        for (int i=0; i < n; ++i)
                for (int j = 0; j < n; ++j)
                        J[i][j] = 0
                J[i][i] = 1
```

```
        J[p][p] = c
        J[q][q] = c
        J[p][q] = s
        J[q][p] = -s
        return J
}
```

그래서, $J(p_i, q_i, \theta_i) = \text{JacobiMatrix}(X_i, p_i - 1, q_i - 1)$를 이용해 구하면 된다.

CONVERGENCE PROOF

이번엔, 위 Jacobi Method 에서 X_∞가 수렴한다는 것을 증명하자.

n×n **Square matrix**

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nn} \end{bmatrix}$$

에 대해, trace 를 tr(A)로 표기하고, 다음과 같이 정의한다.

$$\mathrm{tr}(A) = \sum_{i=1}^{n} A_{ii}$$

그런데, $\mathrm{tr}(A^T A) = \sum_{i=1}^{n} \sum_{j=1}^{n} {A_{ij}}^2$ 인 성질이 있는 것을 알 수 있다.

또, m×n matrix A 와 n×m matrix B 에 대해서,

$$\mathrm{tr}(AB) = \sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij} B_{ji} = \mathrm{tr}(BA)$$

가 성립한다.

Square matrix A, B 와 **unitary matrix** U 에 대해서, $B = U^T A U$ 관계가 성립하면, B 는 A 와 unitarily equivalent 라고 부른다.

B 가 A 와 unitarily equivalent 하면

$$\mathrm{tr}(B^T B)$$

$$= tr(U^T A^T U U^T AU)$$

$$= tr(U^T A^T AUU^T)$$

$$= tr(A^T A)$$

이므로, B 와 A 의 각 원소의 제곱의 합은 같은 성질이 있다.

Jacobi Method 로 돌아가서,

$$\begin{bmatrix} N_{pp} & N_{pq} \\ N_{qp} & N_{qq} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}^T \begin{bmatrix} M_{pp} & M_{pq} \\ M_{qp} & M_{qq} \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

$\begin{bmatrix} N_{pp} & N_{pq} \\ N_{qp} & N_{qq} \end{bmatrix}$는 $\begin{bmatrix} M_{pp} & M_{pq} \\ M_{qp} & M_{qq} \end{bmatrix}$와 unitarily equivalent 하므로,

$$N_{pp}{}^2 + N_{qq}{}^2 + 2N_{pq}{}^2 = M_{pp}{}^2 + M_{qq}{}^2 + 2M_{pq}{}^2$$

또, X_i와 X_{i+1}도 unitarily equivalent 하므로 원소의 제곱의 합은 일정하다. Givens rotation 을 반복할 때마다, p_i행 q_i열 원소와 q_i행 p_i열의 원소의 절대값은 0 으로 줄어들고, p_i행 p_i열과 q_i행 q_i열의 원소의 절대값은 그만큼 커지게 된다. 그러므로 대각선에 위치하지 않은 원소의 제곱의 합은 0 에 가까워진다.

마지막으로 대각선에 위치하지 않은 원소의 값이 0 에 수렴한다는 것을 증명해보자. p_i, q_i의 제곱 값이 대각선에 위치 않은 원소 중에서 가장 크므로, 대각선에 위치하지 않은 원소의 제곱의 합은 최소한 $1 - \frac{n(n-1)}{2}$의 비율로 줄어든다. 그러므로, 0 에 수렴하게 된다.

POLAR DECOMPOSITION

polar decomposition 은 **square matrix** A 를 다음 조건을 만족하는 **orthogonal matrix** Q 와 **positive semidefinite symmetric matrix** S 로 분해한다.

$$A = QS$$

positive semidefinite matrix 는 모든 **eigenvalues** 가 음이 아닌 matrix 이다.

즉 **symmetric matrix** S 를 다음과 같이 **specular decomposition** 했을 때,

$$S = U\begin{bmatrix} \Lambda_{11} & 0 & \cdots & 0 \\ 0 & \Lambda_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \Lambda_{nn} \end{bmatrix}U^T$$

$\Lambda_{11}, \Lambda_{22}, \cdots, \Lambda_{nn}$의 값이 음이 아니면 S 는 **positive semidefinite symmetric matrix** 이다.

INVERTIBLE MATRIX

우선 A 가 n×n **invertible matrix** 일 때, **polar decomposition** 을 하는 방법에 대해서 알아보자.

이 방법은 **[3] Computing the polar decomposition--with application** 를 읽어보면, 그 원리와 방법에 대해 자세히 나와있다. 그 원리를 이해하기는 상당히 복잡하므로, 여기에서는 그 결과만 이용하도록 하겠다. 원리를 이해하려면, 매트릭스에 대한 기초 공부를 먼저 해야 한다.

이 방법에서는

$$X_0 = A$$

로 설정한다. 그 다음,

$$X_{i+1} = \frac{1}{2}(X_i + X_i^{-T})$$

을 반복한다. 이 때, X_i^{-T}는 $(X_i^T)^{-1}$를 의미한다. 그러면 X_∞는 Q 에 수렴한다.

이렇게, Q 를 구한 후, S 는

$$S = \frac{1}{2}(Q^T A + A^T Q)$$

를 이용해 계산한다.

그런데,

$$X_{i+1} = \frac{1}{2}\left(\gamma_i X_i + \frac{X_i^{-T}}{\gamma_i}\right)$$

로 하고, γ_i 를 적당한 값으로 설정하면 수렴하는 속도가 더 빨라진다.

여기에서는

$$\gamma_i = \left(\left\|X_i^{-1}\right\|_1 \left\|X_i^{-1}\right\|_\infty / \left(\|X_i\|_1 \|X_i\|_\infty\right)\right)^{\frac{1}{4}}$$

을 이용해 계산한다. 그리고, $\|M\|_1, \|M\|_\infty, \|M\|_F$를 **matrix norm** 이라고 부르는데, m×n matrix M 에 대하여, 다음과 같이 정의한다.

$$M = \begin{bmatrix} M_{11} & M_{12} & \cdots & M_{1n} \\ M_{21} & M_{22} & \cdots & M_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ M_{m1} & M_{m2} & \cdots & M_{mn} \end{bmatrix}$$

$$\|M\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{m} |M_{ij}| \text{ (Column sum norm)}$$

$$\|M\|_\infty = \max_{1 \le i \le n} \sum_{j=1}^{n} |M_{ij}| \text{ (Row sum norm)}$$

$$\|M\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} M_{ij}^2} \text{ (Frobenius norm)}$$

HOUSEHOLDER REFLECTION

non-invertible matrix 의 경우에 처리 방법을 알기 위해서는 먼저 **householder reflection** 에 대해 알아두어야 한다.

v 를 n 차원 vector 라고 할 때, 다음 형식으로 표현된 n×n matrix 를 **householder matrix** 이라고 한고, **v** 를 **householder vector** 라고 부른다.

$$P = I - \frac{2\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}}$$

P 는 원점을 지나고, 수직 방향이 **v** 인 평면에 대한 대칭 이동의 변환 행렬이다.

자, 이번에는 어떤 3 차원 vector $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$를 z 축 방향으로 변환하는 **householder vector** 를 구해 보자.

z 축 방향 단위 벡터를 $\mathbf{e_3} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ 로 표기하면, $\left(\mathrm{I} - \frac{2\mathrm{vv}^\mathrm{T}}{\mathrm{v}^\mathrm{T}\mathrm{v}}\right)\mathbf{x}$가 $\mathbf{e_3}$의 배수여야 한다. 또, **v** 는 원점, **x**, $\mathbf{e_3}$을 포함하는 평면 위에 있어야 한다.

그래서, $\mathbf{v} = \mathbf{x} + \alpha\mathbf{e_3}$라고 하면

$$\left(\mathrm{I} - \frac{2\mathbf{v}\mathbf{v}^\mathrm{T}}{\mathbf{v}^\mathrm{T}\mathbf{v}}\right)\mathbf{x}$$

$$= \mathbf{x} - \frac{2\mathbf{v}(\mathbf{v} \cdot \mathbf{x})}{\mathbf{v} \cdot \mathbf{v}} = \mathbf{x} - \frac{2(\mathbf{x} + \alpha\mathbf{e_3})\big((\mathbf{x} + \alpha\mathbf{e_3}) \cdot \mathbf{x}\big)}{\mathbf{x} \cdot \mathbf{x} + 2\alpha \mathrm{x}_3 + \alpha^2}$$

$$= \mathbf{x} - \frac{2(\mathbf{x} + \alpha\mathbf{e_3})(\mathbf{x} \cdot \mathbf{x} + \alpha \mathrm{x}_3)}{\mathbf{x} \cdot \mathbf{x} + 2\alpha \mathrm{x}_3 + \alpha^2} = \left(1 - \frac{2(\mathbf{x} \cdot \mathbf{x} + \alpha \mathrm{x}_3)}{\mathbf{x} \cdot \mathbf{x} + 2\alpha \mathrm{x}_3 + \alpha^2}\right)\mathbf{x} - \frac{2\alpha(\mathbf{x} \cdot \mathbf{x} + \alpha \mathrm{x}_3)}{\mathbf{x} \cdot \mathbf{x} + 2\alpha \mathrm{x}_3 + \alpha^2}\mathbf{e_3}$$

여기서 $\mathbf{x}$의 계수가 0 이어야 하므로,

$$1 - \frac{2(\mathbf{x} \cdot \mathbf{x} + \alpha \mathrm{x}_3)}{\mathbf{x} \cdot \mathbf{x} + 2\alpha \mathrm{x}_3 + \alpha^2} = 0$$

$$\alpha = \pm|\mathbf{x}|$$

$$\mathbf{v} = \mathbf{x} \pm |\mathbf{x}|\mathbf{e_3}$$

이것을 알고리즘으로 구현하면,

```
float[3] make_reflector(float x[3])
{
    float v[3];
    float length = sqrt(dot(x, x))
    v = x;
    if (x[2] >= 0)
        v[2] += length
    else
        v[2] -= length
    return v * sqrt(2 / dot(v, v))
}
```

make_reflector 함수는 길이가 $\sqrt{2}$인 **householder vector** 를 계산해준다.

위에서 x[2]와 length 의 값이 비슷할 때, x[2]에서 length 를 빼면 float 의 cancellation 으로 인한 오차가 생길 수 있다. (벡터 $\mathbf{x}$ 가 z 축 방향일 때, $\mathbf{v} = \mathbf{x} - |\mathbf{x}|\mathbf{e_3}$를 사용하면 오차가 커진다.).
그래서, **v** 의 2 개의 답 중 오차를 줄일 수 있는 쪽의 답을 선택한다.

RANK 2

이번에는 A 가 3×3 matrix 이고, rank 가 2 일 때의 계산 방법에 대해 알아보겠다.

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

우선 **householder matrix** V 와 A 를 곱해서, 그 곱이 다음 형식이 되도록 변환한다.

$$B = VA = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ 0 & 0 & 0 \end{bmatrix}$$

행렬 A 에서 rank 가 2 이므로, $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} A_{11} \\ A_{21} \\ A_{31} \end{bmatrix}, \begin{bmatrix} A_{12} \\ A_{22} \\ A_{32} \end{bmatrix}, \begin{bmatrix} A_{13} \\ A_{23} \\ A_{33} \end{bmatrix}$는 같은 평면에 위치한다. 이 때, 3 차원 벡터 **x** 를 $\begin{bmatrix} A_{11} \\ A_{21} \\ A_{31} \end{bmatrix} \mathrm{X} \begin{bmatrix} A_{12} \\ A_{22} \\ A_{32} \end{bmatrix}$로 정하면, **x** 는 그 평면에 수직방향이다.

그래서 **householder vector v** 를 **v** = make_reflector(**x**) 을 이용해 계산하면,

$(\mathrm{I} - \mathbf{v}\mathbf{v}^{\mathrm{T}})$는 **x** 를 z 방향으로 변환하는 대칭이동이다. 그런데, $\begin{bmatrix} A_{11} \\ A_{21} \\ A_{31} \end{bmatrix}$는 **x** 에 수직이므로,

$(\mathrm{I} - \mathbf{v}\mathbf{v}^{\mathrm{T}}) \begin{bmatrix} A_{11} \\ A_{21} \\ A_{31} \end{bmatrix}$는 z 방향에 수직이어야 한다. 따라서, $\begin{bmatrix} B_{11} \\ B_{21} \\ 0 \end{bmatrix} = (\mathrm{I} - \mathbf{v}\mathbf{v}^{\mathrm{T}}) \begin{bmatrix} A_{11} \\ A_{21} \\ A_{31} \end{bmatrix}$ 가 된다.

A 의 다른 열도 마찬가지로 변환되기 때문에, $\begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ 0 & 0 & 0 \end{bmatrix} = (\mathrm{I} - \mathbf{v}\mathbf{v}^{\mathrm{T}})\mathrm{A}$가 된다.

이제 B 에 **householder matrix** W 를 곱해서, 그 곱을 다음 형식으로 만들자.

$$C = BW = \begin{bmatrix} C_{11} & C_{12} & 0 \\ C_{21} & C_{22} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

3 차원 벡터 **y** 를 $\begin{bmatrix} B_{11} \\ B_{12} \\ B_{13} \end{bmatrix} \mathrm{X} \begin{bmatrix} B_{21} \\ B_{22} \\ B_{23} \end{bmatrix}$로 정하면, **y** 는 $\begin{bmatrix} B_{11} \\ B_{12} \\ B_{13} \end{bmatrix}$과 수직이다.

이번엔, **householder vector w** 를 **w** = make_reflector(**y**)로 구하면,

$$\begin{bmatrix} C_{11} \\ C_{12} \\ 0 \end{bmatrix} = (\mathrm{I} - \mathbf{w}\mathbf{w}^{\mathrm{T}}) \begin{bmatrix} B_{11} \\ B_{12} \\ B_{13} \end{bmatrix}$$

가 성립한다. 양쪽을 transpose 하면

$$[C_{11} \quad C_{12} \quad 0] = [B_{11} \quad B_{12} \quad B_{13}](I - \mathbf{w}\mathbf{w}^T)$$

가 된다.

마찬가지로,

$$\begin{bmatrix} C_{11} & C_{12} & 0 \\ C_{21} & C_{22} & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ 0 & 0 & 0 \end{bmatrix}(I - \mathbf{w}\mathbf{w}^T)$$

이제 2×2 **invertible matrix** $\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$는 Invertible matrix 이므로, 앞에서 배운 방법대로 polar decomposition 해서 2×2 **orthogonal matrix** Q 와 2×2 **positive definite symmetric matrix** S 로 분해되었다고 하자.

그러면

$$C = \begin{bmatrix} QS & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} Q & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix}$$

$$A = VCW = V\begin{bmatrix} Q & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix}W = V\begin{bmatrix} Q & 0 \\ 0 & 1 \end{bmatrix}WW\begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix}W$$

그래서 A 는 3x3 **orthogonal matrix** $V\begin{bmatrix} Q & 0 \\ 0 & 1 \end{bmatrix}W$와 3x3 **positive semi-definite symmetric matrix** $W\begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix}W$로 분해된다.

2×2 INVERTIBLE MATRIX

이번에는 2×2 **invertible matrix** A=$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$를 분해해 보자.

A 가 2×2 **orthogonal matrix** Q 와 2×2 **positive definite symmetric matrix** S 로 분해되었다고 가정하자.

$$A = QS$$

만약 A 의 determinant 가 양수라면 Q 는 rotation matrix 여야 한다. 이때, Q 는 다음과 같은 형식이다.

$$Q = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}, \qquad c^2 + s^2 = 1$$

$$S = Q^T A = \begin{bmatrix} cA_{11} + sA_{21} & cA_{12} + sA_{22} \\ -sA_{11} + cA_{21} & -sA_{12} + cA_{22} \end{bmatrix}$$

S 는 symmetric 이므로,

$$cA_{12} + sA_{22} = -sA_{11} + cA_{21}$$

$$c: s = A_{11} + A_{22}: A_{21} - A_{12}$$

이 관계식을 이용해 Q 를 구한다.

만약, determinant 가 음수라면 Q 는 reflection matrix 여야 한다. 이 때, Q 는 다음 형식을 갖는다.

$$Q = \begin{bmatrix} c & s \\ s & -c \end{bmatrix}, \qquad c^2 + s^2 = 1$$

그 이후 계산 과정은 비슷하므로 생략하겠다.

RANK 1

이번에는 A 가 3×3 matrix 이고, rank 가 1 일 때의 계산 방법에 대해 알아보겠다.

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

일단, **householder vector v** 를 **v** = make_reflector($\begin{bmatrix} A_{11} \\ A_{21} \\ A_{31} \end{bmatrix}$) 을 이용해 계산한다.

그 다음, householder matrix 에 A 를 곱하면,

$$\begin{bmatrix} 0 \\ 0 \\ B_{31} \end{bmatrix} = (I - \mathbf{v}\mathbf{v}^T) \begin{bmatrix} A_{11} \\ A_{21} \\ A_{31} \end{bmatrix}$$

그런데, $\begin{bmatrix} A_{12} \\ A_{22} \\ A_{32} \end{bmatrix}, \begin{bmatrix} A_{13} \\ A_{23} \\ A_{33} \end{bmatrix}$도 $\begin{bmatrix} A_{11} \\ A_{21} \\ A_{31} \end{bmatrix}$와 같은 방향이므로

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ B_{31} & B_{32} & B_{33} \end{bmatrix} = (I - \mathbf{v}\mathbf{v}^T)A$$

가 된다. 그 다음 **householder vector w** 를 **w** = make_reflector($\begin{bmatrix} B_{31} \\ B_{32} \\ B_{33} \end{bmatrix}$) 을 이용해 계산한다.

$$C = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & C_{33} \end{bmatrix} = B(I - \mathbf{w}\mathbf{w}^T)$$

그 다음부터는 Rank 가 2 일 때와 비슷하므로 생략하겠다.

SINGULAR VALUE DECOMPOSITION

singular value decomposition 은 m×n matrix A 를 다음 조건을 만족하는 m×m **orthogonal matrix** V, m×n **rectangular diagonal matrix** Σ, n×n **orthogonal matrix** W 로 분해한다.

$$A = V\Sigma W^T$$

m >=n 일 때

$$\Sigma = \begin{bmatrix} \Sigma_{11} & 0 & \cdots & 0 \\ 0 & \Sigma_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \Sigma_{nn} \\ \vdots & & & 0 \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix}$$

이다. $\Sigma_{11}, \Sigma_{22}, \cdots, \Sigma_{nn}$ 를 matrix A 의 **singular values** 라고 부른다.

또, m <= n 일 때

$$\Sigma = \begin{bmatrix} \Sigma_{11} & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \Sigma_{22} & \ddots & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & \vdots \\ 0 & \cdots & 0 & \Sigma_{mm} & 0 & \cdots & 0 \end{bmatrix}$$

이다.

우선 A 가 **square matrix** 일 때, 어떻게 분해하는지 알아보겠다.

일단 A 를 **polar decomposition** 해서 **orthogonal matrix** Q 와 **positive semidefinite symmetric matrix** S 로 분해한다.

$$A = QS$$

그 다음 **symmetric matrix** S 를 **spectral decomposition** 해서 **diagonal matrix** Σ 와 **orthogonal matrix** W 로 분해한다.

$$S = W\Sigma W^T$$

$$A = Q(W\Sigma W^T) = (QW)\Sigma W^T$$

위와 같이 A 를 **orthogonal matrix** QW, **diagonal matrix** Σ, **orthogonal matrix** W 로 분해한다.

마지막으로 n×m matrix $A_{n,m}$의 분해 방법에 대해 알아보자.

$A_{n,m}$의 rank 를 r 이라고 하면, **householder matrix** $T_{n,n}$과 $U_{m,m}$에 의해 다음과 같이 분해될 수 있다.

$$A_{n,m} = T_{n,n}\begin{bmatrix} B_{r,r} & 0_{r,m-r} \\ 0_{n-r,r} & 0_{n-r,m-r} \end{bmatrix} U_{m,m}$$

$B_{r,r}$를 **singular value decomposition** 해서, **orthogonal matrix** $V_{r,r}$, **diagonal matrix** $\Sigma_{r,r}$, **orthogonal matrix** $W_{r,r}$ 로 분해되었다고 하자.

$$A_{n,m} = T_{n,n}\begin{bmatrix} V_{r,r}\Sigma_{r,r}{W_{r,r}}^T & 0_{r,m-r} \\ 0_{n-r,r} & 0_{n-r,m-r} \end{bmatrix} U_{m,m}$$

$$= T_{n,n}\begin{bmatrix} V_{r,r} & 0_{r,n-r} \\ 0_{n-r,r} & I_{n-r,n-r} \end{bmatrix}\begin{bmatrix} \Sigma_{r,r} & 0_{r,m-r} \\ 0_{n-r,r} & 0_{n-r,m-r} \end{bmatrix}\begin{bmatrix} {W_{r,r}}^T & 0_{r,m-r} \\ 0_{m-r,r} & I_{m-r,m-r} \end{bmatrix} U_{m,m}$$

그래서 n×n **orthogonal matrix** $T_{n,n}\begin{bmatrix} V_{r,r} & 0_{r,n-r} \\ 0_{n-r,r} & I_{n-r,n-r} \end{bmatrix}$, n×m **rectangular diagonal matrix** $\begin{bmatrix} \Sigma_{r,r} & 0_{r,m-r} \\ 0_{n-r,r} & 0_{n-r,m-r} \end{bmatrix}$, m×m **orthogonal matrix** $U_{m,m}\begin{bmatrix} W_{r,r} & 0_{r,m-r} \\ 0_{m-r,r} & I_{m-r,m-r} \end{bmatrix}$로 분해되었다.

SNUGGLE

Ken Shoemake 의 소스를 분석해 보면, **symmetric matrix** A 를

$$A = U\Lambda U^T$$

로 **spectral decomposition** 한 다음, 회전 행렬 U 의 회전 각도를 최소화하려고 한다.

여기서, U의 값은 여러 개의 값을 가질 수 있다. 그래서, 그 중에 회전 각도가 가능한 한 작은 U를 선택한다.

[5] Matrix Animation and Polar Decomposition 에 설명이 되어 있으나, 설명이 좀 어렵다.

우선, 회전 행렬 U 를 quaternion q=$q_w + q_x\mathbf{i} + q_y\mathbf{j} + q_z\mathbf{k}$ 으로 변환한다. 그 다음, w component 를 값을 최대화하면 회전 각도를 최소화할 수 있다.

$\Lambda = \begin{bmatrix} \Lambda_{11} & 0 & 0 \\ 0 & \Lambda_{22} & 0 \\ 0 & 0 & \Lambda_{33} \end{bmatrix}$라고 할 때, Λ_{11}, Λ_{22}, Λ_{33}중 몇 개가 같은 값이냐에 따라 알고리즘이 달라진다.

우선, Λ_{11}, Λ_{22}, Λ_{33}가 모두 같은 값이면 가장 쉽다. $1 + 0\mathbf{i} + 0\mathbf{j} + 0\mathbf{k}$ 를 회전 행렬의 quaternion 으로 하면 된다.

만약 3 개가 모두 다르다면, q 에 어떤 quaternion $p = p_w + p_x\mathbf{i} + p_y\mathbf{j} + p_z\mathbf{k}$ 를 곱해 본 후, 그 곱 qp 의 w component(실수 부분)이 최대가 되는 값을 찾는다.

p 는 다음에서 지정한 48 개의 값 중 하나여야 한다. p 의 각 component 를 $[p_w \quad p_x \quad p_y \quad p_z]$로 나타내면, 아래 패턴 중 하나여야 한다.

$$[\pm 1 \quad 0 \quad 0 \quad 0]$$

$$[\pm 1 \quad \pm 1 \quad 0 \quad 0]/\sqrt{2}$$

$$[\pm 1 \quad \pm 1 \quad \pm 1 \quad \pm 1]/2$$

각 패턴의 순서는 뒤바뀔 수 있다.

첫 번째 패턴은 순서의 조합에 따른 4 가지와 부호의 조합에 의해 2 가지씩 8 개이다.

두 번째 패턴은 순서의 조합에 따른 6 가지와 부호의 조합에 따른 4 가지씩 24 개이다.

세 번째 패턴은 순서의 조합에 따른 1 가지와 부호의 조합에 따른 16 가지씩 16 개이다.

그래서, p 는 총 48 개의 값을 가질 수 있다.

q 와 p 의 곱의 실수 부분은 $q_w p_w - q_x p_x - q_y p_y - q_z p_z$이다. 이 값을 가장 크게 만들 수 있는 p 의 값을 48 개 중에서 선택하면 된다.

만약, p 의 값으로 $(1 + 0\mathbf{i} - \mathbf{j} + 0\mathbf{k})/\sqrt{2}$ 가 선택되었다고 하자. 이 값에 해당하는 회전 행렬은 $\begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$이다. 이 때 $\begin{bmatrix} \Lambda_{11} & 0 & 0 \\ 0 & \Lambda_{22} & 0 \\ 0 & 0 & \Lambda_{33} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Lambda_{33} & 0 & 0 \\ 0 & \Lambda_{22} & 0 \\ 0 & 0 & \Lambda_{11} \end{bmatrix} \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$ 이므로, **symmetric matrix** 도 $\begin{bmatrix} \Lambda_{33} & 0 & 0 \\ 0 & \Lambda_{22} & 0 \\ 0 & 0 & \Lambda_{11} \end{bmatrix}$ 로 바꿔야 한다.

마지막으로, $\Lambda_{11} = \Lambda_{22} \neq \Lambda_{33}$ 인 경우에 대해서 알아보자.

이 때는 z 축 방향으로 자유롭게 회전할 수 있다.

그래서, qp 에 z 축 방향 회전운동 r 을 곱해서, 그 곱 qpr 의 실수 부분이 최대가 되는 값을 구한다.

r 이 z 축 방향 회전이므로 $r = c + 0\mathbf{i} + 0\mathbf{j} + s\mathbf{k}$, $c^2 + s^2 = 1$의 형태여야 한다.

만약, qp 의 곱이 $qp = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ 라면, $qpr = (wc - zs) + (xc + ys)\mathbf{i} + (yc - xs)\mathbf{j} + (zc - ws)\mathbf{k}$ 이 된다. $c = w/\sqrt{w^2 + z^2}, s = -z/\sqrt{w^2 + z^2}$ 일 때, qpr 의 실수 부분은 최대값 $\sqrt{w^2 + z^2}$를 가진다.

그래서, 이번에는 qp의 w component 의 제곱과 z component 의 제곱의 합이 최대화하도록

p의 값을 48 개 중에서 하나 고른다. 그런데, 48 개의 값 중에서 6 가지만 검사하면 된다. 나머지는 6 개중 하나와 같은 결과가 나온다. 그 값은

$$1 + 0\mathbf{i} + 0\mathbf{j} + 0\mathbf{k}$$

$$(1 + \mathbf{i} + \mathbf{j} + \mathbf{k})/2$$

$$(-1 + \mathbf{i} + \mathbf{j} + \mathbf{k})/2$$

와 각 값에 $0 + 1\mathbf{i} + 0\mathbf{j} + 0\mathbf{k}$를 곱한 값이다.

그리고, 각 경우에 $w^2 + z^2 - \frac{1}{2}$의 값은 $\pm\left(q_w{}^2 + q_z{}^2 - \frac{1}{2}\right)$, $\pm(q_x q_z - q_w q_y)$, $\pm(q_w q_x + q_y q_z)$이다.

이 값 중 가장 큰 값을 구한 후, 그 값에 해당하는 p 의 값을 선택한다. 만약 가장 큰 값이 – 부호에 해당하면, 그 p 의 값에 $0 + 1\mathbf{i} + 0\mathbf{j} + 0\mathbf{k}$를 곱한 값을 p 로 사용한다.

참고자료

[1] Matrix Analysis, Roger A. Horn and Charles R. Johnson

행렬에 대한 고급 수학을 공부하기에 좋은 책이다. 조금 어렵다.

[2] Matrix Computations, Gene H. Golub. Charles F. Van Loan. Third Edition

행렬 계산을 컴퓨터에서 구현하는 방법에 대해서 잘 설명되어 있다. 다 읽을 필요는 없고 중간 중간 필요한 부분만 읽으면 될 듯 하다.

[3] Computing the polar decomposition--with application, Nicholas. J. Higham

[4] Fast Polar Decomposition of An Arbitrary Matrix, Nicholas Higham and Robert S. Schreiber

[5] Matrix Animation and Polar Decomposition, Ken Shoemake and Tom Duff