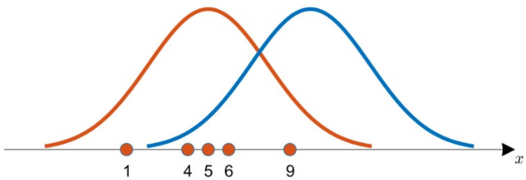
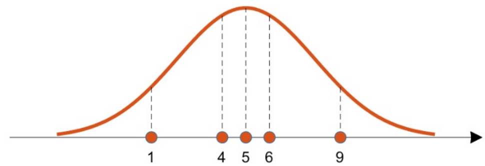
Maximum Likelihood Estimation - MLE





Note) Gaussian Distribution 이라고 가정하면,

이에 대한 Parameter는 μ, σ가 존재하며, 이들에 대한 각각의 편미분을 해야한다.

$$\mu = \frac{(x_1 + x_2 + \cdots + x_n)}{n}, \quad \sigma = \sqrt{\frac{(x_1-\mu)^2 + \cdots + (x_n-\mu)^2}{n}}$$

MLE는 모수적인 Data 밀도 추정 방법이다. ⇒ 관측되는 Data들을 가장 잘 Modeling 하는 Probability Distribution의 Parameter 찾는 알고리즘.

↳ 모수 : Input Data의 Distribution을 가정하는것을 의미한다. ⇒ 모수의 결정 여부 (• 모수적 방법 • 비모수적 방법

↳ Parameter $\theta = (\theta_1, \theta_2, \dots \theta_m)$으로 구성된 확률 밀도함수 $P(x|\theta)$에서 관측된 Sample Data set을 $(x_1, x_2, \dots x_n)$이라고 하자.

이때, Sample Data에서 Parameter $\theta = (\theta_1, \theta_2, \dots \theta_m)$을 추정하는 방법을 의미한다.

우리는 Data를 통해, 해당 Data들이 Sampling 되었을것이라 생각되는 Distribution의 Property를 Estimation 할 수 있다.

이때, Distribution에 대한 형태를 가정하고, 해당 Distribution의 Property를 추정하는 것이다.

↳ "모수"

Likelihood : 얻은 Data가 해당 Distribution으로 부터 나왔을 가능도를 의미한다. (Probability가 가정된 상황에서 관측값이 나오는 확률값)

↳ 이를 계산하기 위해 각 Sample Data에서 Candidate Distribution에 대한 height를 계산하여 다 곱한것을 이용한다.

↳ 모든 Data들의 Sampling이 독립적으로 이뤄지는 Event이기 때문이다.

$$P(x|\theta) = \prod_{k=1}^{n} P(x_k|\theta)$$ ⇒ 전체 표본 집합의 결합 확률 밀도 함수 = Likelihood Function

↳ 이 결과가 가장 큰 θ를 $\hat{\theta}$으로 여긴다.

log scale (계산 편의) : $L(\theta|x) = \log P(x|\theta) = \sum_{i=1}^{n} \log P(x_i|\theta)$

↳ $\frac{\partial}{\partial\theta} L(\theta|x) = \sum_{i=1}^{n} \frac{\partial}{\partial\theta} \log P(x_i|\theta) = 0$을 구하여 Likelihood Function을 Maximize 하는 θ를 찾는다.

Multivariate Gaussian Distribution

$\rightarrow X \sim N(\mu, \Sigma)$

Multivariate Gaussian Distribution의 DPF는 아래와 같다.

$$P(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right]$$

$\rightarrow \dfrac{1}{\sqrt{(2\pi)^n \cdot |\Sigma|}}$ 로 쓸수 있다.

- n: Data Dimension
- μ: Mean
- Σ: Covariance matrix $\rightarrow$ X가 Multivariable 이기에, Σ의 (i,j)는 $Cov(X_i, X_j)$ 이다.

※ $Cov(X,Y) = E[(X-E[X])(Y-E[Y])]$

$= E[XY] - E[X]E[Y]$

$$\begin{bmatrix} Cov(X_1, X_1) & \cdots & Cov(X_1, X_n) \\ \vdots & & \vdots \\ Cov(X_n, X_1) & \cdots & Cov(X_n, X_n) \end{bmatrix} = \begin{bmatrix} E[(X_1-E[X_1])(X_1-E[X_1])] & \cdots \\ \vdots & \\ E[(X_n-E[X_n])(X_1-E[X_1])] & \cdots \end{bmatrix}$$

$$= E\begin{bmatrix} (X_1-\mu_1)^2 & \cdots & (X_1-\mu_1)(X_n-\mu_n) \\ \vdots & & \\ (X_n-\mu_n)(X_1-\mu_1) & \cdots & (X_n-\mu_n)^2 \end{bmatrix} = E\begin{bmatrix} (X_1-\mu_1) \\ \vdots \\ (X_n-\mu_n) \end{bmatrix}\begin{bmatrix} (X_1-\mu_1) & \cdots & (X_n-\mu_n) \end{bmatrix}$$

$$= E[(X-\mu)(X-\mu)^T]$$

Log Scale Multivariable Gaussian Distribution

여기서 μ는 vector · Σ는 Square Matrix가 된다.

Multivariate Gaussian Distribution의 DPF는 아래와 같다.

$$P(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left[-\frac{1}{2} (x-\mu)^T \Sigma^{-1} (x-\mu) \right]$$

$\frac{1}{\sqrt{(2\pi)^n \cdot |\Sigma|}}$ 로 쓸수 있다. ($|\Sigma|$ → Determinant.)

여기에 log를 취하면,

$$\log\left(\frac{1}{\sqrt{(2\pi)^n |\Sigma|}}\right) \log\left(\exp\left[-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right]\right)$$

$$= \log\left(\frac{1}{\sqrt{(2\pi)^n |\Sigma|}}\right) - \frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)$$

$$= \log\left(\left((2\pi)^n |\Sigma|\right)^{-\frac{1}{2}}\right) - \frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)$$

$$= -\frac{1}{2}\log\left((2\pi)^n |\Sigma|\right) - \frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)$$

$$= -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log(|\Sigma|) - \frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)$$

n은 Dataset Dimension으로 $-\frac{n}{2}\log(2\pi)$와 $-\frac{1}{2}\log(|\Sigma|)$는 Constant가 된다.

본 논문에서는 $-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)$ 라는 term이 Random Variable에 의존적이며, $\frac{1}{2}$는 - 가 붙어있기에, $(x-\mu)^T \Sigma^{-1} (x-\mu)$를 Minimize 하는 형태로 정리한다.

Levenberg - Marquardt Algorithm [LMA라고 하도록 한다.]

: Non-linear least square를 해결하는 대표적인 방법이다.

↳ Nonlinear Function optimization에 있어 가장 널리 쓰이는 방법이다.

LMA는 Gauss-Newton Method와 Gradient Descent Method 결합 형태이다.

↳ 이를 알기 위해선 다른 기법에 대해 알아야한다.

- 최소자승법 : least square Method

⇒ 관측값 $(x_i, y_i)_{i=1\dots n}$, Model parameter $P=(P_1, P_2, \dots P_m)$, Model : $y=f(x,P)$

여기서 관측값과 Model의 오차를 r_i라고 한다.

이때, 오차 제곱합이 최소가 되도록 하는 Parameter P를 구하는것

$$P^* = \arg\min \sum_{i=1}^{n} r_i^2 = \arg\min \sum_{i=1}^{n} (y_i - f(x_i, P))^2$$

※ 관측값의 신뢰도가 다르면, r_i^2에 가중치를 곱해 최소화 시키도록 한다.

이는 P에 대한 미분이 '0'이 되는 P를 구해 풀수 있다.

이때, Model이 linear하면, linear least square, Non-linear하면, Non-linear Least Square 한다.

⇒ Non linear에 대한 경우, Closed form solution이 없기에 Iterative Minimization을 사용한다.

즉, P_0에서 시작해 점차 Minimize하는 쪽으로 P를 갱신한다.

↳ 여기에 Gradient Descent, Gauss Newton, LMA 방법들이 속한다.

↳ Non-linear Function을 지역적으로 linear Function으로 근사하여 해를 구하는 원리

해에서 멀리 있는 경우, Gradient Descent, 해 근처에 있는 경우, Gauss-Newton을 이용한다

$$\mathbf{p}_{k+1} = \mathbf{p}_k - (J_\mathbf{r}^T J_\mathbf{r} + \mu_k \, diag(J_\mathbf{r}^T J_\mathbf{r}))^{-1} J_\mathbf{r}^T \mathbf{r}(\mathbf{p}_k), \quad k \geq 0$$

$$J_\mathbf{r}(\mathbf{p}) = \begin{bmatrix} \frac{\partial r_1(\mathbf{p})}{\partial p_1} & \cdots & \frac{\partial r_1(\mathbf{p})}{\partial p_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial r_n(\mathbf{p})}{\partial p_1} & \cdots & \frac{\partial r_n(\mathbf{p})}{\partial p_m} \end{bmatrix}, \quad \mathbf{r}(\mathbf{p}) = \begin{bmatrix} r_1(\mathbf{p}) \\ r_2(\mathbf{p}) \\ \vdots \\ r_n(\mathbf{p}) \end{bmatrix} = \begin{bmatrix} y_1 - f(x_1, \mathbf{p}) \\ y_2 - f(x_2, \mathbf{p}) \\ \vdots \\ y_n - f(x_n, \mathbf{p}) \end{bmatrix}$$

Closed - Form Solution

: 알려진 함수나 수학 연산으로 해를 구할 수 있는 식을 의미한다.

Non linear 에 대한 미분방정식이나 Coefficient가 변수라면 close-form Solution X

Singular Value Decomposition

임의의 mxn Matrix에 대한 SVD는 $A = U\Sigma V^T$로 정의할 수 있다.

- U는 mxm 직교행렬 ⇒ 행·열 Vector 모두 수직, 크기 : 1
- V는 nxn 직교행렬
- Σ는 mxn 직사각 대각행렬

$Q^{-1} = Q^T$
$Q^T Q = I$

여기서 U는 AA^T를 eigendecomposition 하여 얻은 othogonal Matrix, U의 column Vector ⇒ left singular Vector

V는 A^TA를 〃 V의 〃 ⇒ Right singular Vector

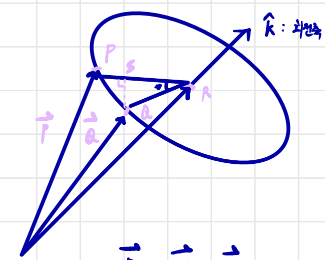
Σ는 AA^T, A^TA를 eigendecomposition 해서 나오는 eigenvalue들의 Square Root를 대각 원소로 하는 mxn 대각행렬로 그 대각원소들을 A의 singular Value라고 한다.

Rodrigues Formula

$$
\begin{aligned}
a &= (a_x, a_y, a_z) = k \\
c &= \cos\alpha \\
s &= \sin\alpha
\end{aligned}
$$

$$
R_a = \begin{bmatrix} (1-c)a_x^2 + c & (1-c)a_x a_y - s a_z & (1-c)a_x a_z + s a_y \\ (1-c)a_x a_y + s a_z & (1-c)a_y^2 + c & (1-c)a_y a_z - s a_x \\ (1-c)a_x a_z - s a_y & (1-c)a_y a_z + s a_x & (1-c)a_z^2 + c \end{bmatrix}
$$

Axis와 Angle이 주어졌을 때, 어떠한 Vector를 해당 Axis에 대해 해당 Angle만큼 Rotation한 Vector를 구하는 과정이다.



$\vec{P} = \vec{OR} + \vec{RP}$

$\vec{OP} - \vec{OR}$는 $\vec{RP}$이기에 $(\vec{OP} - \vec{OR})\cos\theta$로 나타낼수 있다.

여기서, $\overline{PR}$과 $\overline{SQ}$는 수직

$\vec{Q} = \vec{OR} + \vec{RS} + \vec{SQ}$

$\vec{SQ}$ ↳ $(\vec{P} \times \hat{k}) \cdot \sin\theta$

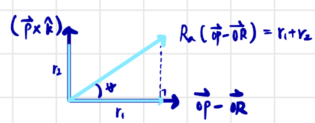
$\vec{OR}$ ↓ $(\vec{P} \cdot \hat{k})\hat{k}$

: $\vec{P}$를 $\hat{k}$에 Projection

⇒ 회전 영향 X

$(\vec{OP} - \vec{OR})$과 $(\vec{P} \times \hat{k})$이 두 축으로 Coordinate 구성.

⇒ 새로운 좌표계를 만든것임.



$\|r_1\| = \cos\theta \cdot \|R_a(\vec{OP} - \vec{OR})\|$

즉, $R_a(P) = \vec{OR} + R_a(\vec{OP} - \vec{OR})$이 된다.