

Math about AI

1. Linear algebra review

Basic concepts

vector : 1-D array of numbers, 여기서는 column vector로 정의

Tensors : matrix의 3 이상의 dimensions

$$A \in R^{m \times n \times r}$$

identity matrix: I (ij)

diagonal matrix: 대각선 제외하고 모두 0

inner product & outer product : 어떤 모양이냐에 유의

- Inner product (or dot product)

$$x^T y \in \mathbb{R} = [x_1 \ x_2 \ \dots \ x_n] \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \sum_{i=1}^n x_i y_i.$$

1 x n n x 1 scalar

- Outer Product

$$xy^T \in \mathbb{R}^{m \times n} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} [y_1 \ y_2 \ \dots \ y_n] = \begin{bmatrix} x_1 y_1 & x_1 y_2 & \dots & x_1 y_n \\ x_2 y_1 & x_2 y_2 & \dots & x_2 y_n \\ \vdots & \vdots & \ddots & \vdots \\ x_m y_1 & x_m y_2 & \dots & x_m y_n \end{bmatrix}$$

m x 1 1 x n m x n

About Matrix

Vector Product with matrix: 같은 식임에도 관점에 따라 vector / linear combination으로 해석가능

- If we write \mathbf{A} by rows:

$$y = Ax = \begin{bmatrix} \cdots & a_1^T & \cdots \\ \cdots & a_2^T & \cdots \\ \vdots & & \\ \cdots & a_m^T & \cdots \end{bmatrix}_{m \times n} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}_{n \times 1} = \begin{bmatrix} a_1^T x \\ a_2^T x \\ \vdots \\ a_m^T x \end{bmatrix}_{m \times 1} \rightarrow a_1 x_1 + a_2 x_2 + \dots$$

scalar

- If we write \mathbf{A} by columns:

$$y = Ax = \begin{bmatrix} | & | & | & \cdots & | \\ a^1 & a^2 & \cdots & a^n & | \\ | & | & | & \cdots & | \end{bmatrix}_{n \times m} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}_{n \times 1} = \begin{bmatrix} a^1 \\ a^2 \\ \vdots \\ a^n \end{bmatrix}_{m \times 1} x_1 + \begin{bmatrix} a^1 \\ a^2 \\ \vdots \\ a^n \end{bmatrix}_{m \times 1} x_2 + \dots + \begin{bmatrix} a^1 \\ a^2 \\ \vdots \\ a^n \end{bmatrix}_{m \times 1} x_n$$

coefficient (or weight)

- \mathbf{y} is a **linear combination** of the columns of \mathbf{A}

- \mathbf{y} is a projection from \mathbf{x} via a transformation \mathbf{A} .



matrix \mathbf{A} 를 row 형태로 쓰면 벡터 형식으로 얻을 수 있고, \mathbf{A} 를 column 형태로 쓰면 linear combination 형태의 결과를 얻을 수 있다.

Matrix Multiplication

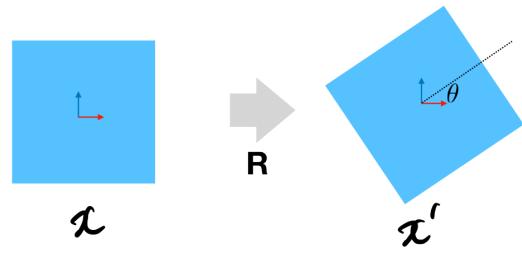
:

매트릭스의 곱셈은 곱해지는 vector나 matrix를 회전시키거나 변형시키는 방법이라고 이해할 수 있다.

- In 2D

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

$$\mathbf{x}' = \mathbf{Rx}$$



- In 3D

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix} \quad R_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \quad R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Hadamard Product(Element-wise Product)

$$(A \circ B)_{ij} = (A \odot B)_{ij} = (A)_{ij}(B)_{ij}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \circ \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & a_{13}b_{13} \\ a_{21}b_{21} & a_{22}b_{22} & a_{23}b_{23} \\ a_{31}b_{31} & a_{32}b_{32} & a_{33}b_{33} \end{bmatrix}$$

A와 B가 같은 dimension일 때 가능하며, 같은 위치의 요소를 곱한 값을 가지는 matrix를 얻는다. 결과값도 A,B와 같은 dimension m * n을 가진다

About Norms

Trace

: n x n의 square matrix일 때 사용하는 함수로 대각선의 요소를 더한 값을 의미

$$tr(A) = \sum_{i=1}^n a_{ii} = a_{11} + a_{22} + \dots + a_{nn}$$

matrix를 transpose한 값이 같고 $tr(A+B)=tr A + tr B$ 이다.

상수 배 했을 때 함수 안밖으로 빼는게 가능하며, 안에서 교환법칙도 성립한다. $tr AB = tr B$

$$tr AB = \sum_{i=1}^m (AB)_{ii} = \sum_{i=1}^m \sum_{j=1}^n a_{ij}b_{ji} = \sum_{j=1}^n \sum_{i=1}^m b_{ji}a_{ij} = \sum_{j=1}^n (BA)_{jj} = tr BA$$

Norms : 지점 간의 거리를 측정하는 장치

$$\|x\|_p = \left(\sum_{i=1}^n |x_i|^p \right)^{\frac{1}{p}}$$

기본적인 형태로 p 는 원하는 실수를 넣으면 된다.

$p=2$ 일 때는 Euclidean or L2 norm / $p=1$ 일 때는 L1 norm

following properties를 만족하는 어떤 함수는 norm이다. (**hold properties**)

$$\begin{aligned} f(\mathbf{x}) &= 0 \Rightarrow \mathbf{x} = 0 \\ f(\mathbf{x} + \mathbf{y}) &\leq f(\mathbf{x}) + f(\mathbf{y}) \\ \forall a \in R, f(a\mathbf{x}) &= |a|f(\mathbf{x}) \end{aligned}$$

Frobenius norm (matrix norm)

$$\|A\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n A_{ij}^2} = \sqrt{\text{tr}(AA^T)}$$

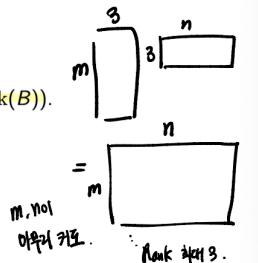
$A = B - C$ 라고 할 때, B와 C 두 matrix가 얼마나 유사한지를 측정하는 지표로 사용된다.

cf) trace function

$$\text{tr}(AB) = \sum_{i=1}^m (AB)_{ii} = \sum_{i=1}^m \sum_{j=1}^n a_{ij} b_{ji}$$

Rank of A Matrix

- For $A \in \mathbb{R}^{m \times n}$, $\text{rank}(A) \leq \min(m, n)$. If $\text{rank}(A) = \min(m, n)$, then A is said to be **full rank**.
- For $A \in \mathbb{R}^{m \times n}$, $\text{rank}(A) = \text{rank}(A^T)$.
- For $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times p}$, $\text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B))$.
- For $A, B \in \mathbb{R}^{m \times n}$, $\text{rank}(A + B) \leq \text{rank}(A) + \text{rank}(B)$.



Special matrix

orthogonal matrix

구성된 column들이 서로 직교해서 내적한 값이 0이며, 각 vector의 norm = 1

직교 행렬의 inverse는 transpose와 같다.

직교 행렬은 magnitude는 유지하고 방향을 바꾸므로 $\|Ux\| = \|x\|$ 로 동일하다!!

+ Rotation matrix (special orthogonal group)

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdots R \in SO(2)$$

- orthogonal matrix

- $\det(R) = 1$
⇒ basis가 나타내는 면적이 determinant인데 이게 1이여야 magnitude가 유지됨

두 개의 조건을 만족하면 곱했을 때 주어진 벡터를 회전시키는 rotation matrix가 된다.

SO(N)은 special orthogonal group이며, N은 크기를 의미 ... 예시는 2D

Skew symmetric matrix

: whose transpose equals its negative

$$A = \begin{bmatrix} 0 & 2 & -45 \\ -2 & 0 & -4 \\ 45 & 4 & 0 \end{bmatrix} \quad A^T = -A = \begin{bmatrix} 0 & 22 & 45 \\ 2 & 0 & 4 \\ -45 & -4 & 0 \end{bmatrix} = A^T$$

$$a \times b = [a] \times b = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix}$$

여기서 a는 $[a_1 \ a_2 \ a_3]$ 인 벡터로 **skew-symmetric**으로 전환해서 벡터 간의 outer product를 matrix와 vector 간의 곱으로 재정의하는 역할을 한다

⇒ 이는 cross products에서 matrix multiplication으로 넘어가는 handy way

Gram Matrix

$$A = [a^1 \ a^2 \ \dots \ a^n] \text{이라고 할 때, 여기서 } a^i \in R^m$$

$$G = A^T A = \begin{bmatrix} a_1^T a_1 & a_1^T a_2 & \dots & a_1^T a_n \\ a_2^T a_1 & a_2^T a_2 & \dots & a_2^T a_n \\ \vdots & \vdots & & \vdots \\ a_n^T a_1 & a_n^T a_2 & \dots & a_n^T a_n \end{bmatrix}$$

Gram matrix는 A의 column들의 inner products를 모두 가지고 있다고 할 수 있다.

특이 케이스로, 만약 G가 identity matrix이면, 나머지가 0이고 같은 column들이 내적하는 가운데만 1이므로 여기서 G는 orthogonal matrix이다.

Gram matrix는 “주어진 그림에 고흐 스타일을 적용해줘” 같은 style transfer에서 사용된다.

Null space

$$N(A) = \{x \in R^n : Ax = 0\}$$

주어진 matrix A ($m \times n$)에서 특정 x를 곱했을 때 $Ax = 0$ 을 만족하는 vector x들의 집합이다.

nullity : null space의 dimension을 the nullity of A라고 부른다

Rank-Nullity Theorem : $\text{Rank}(A) + \text{Nullity}(A) = n$ (n is the number of columns)

⇒ 생각해보면 당연!! 보통 rank를 구할 때 echelon form으로 변형해서 구하는데, 변형한 형태에서 rank에 해당하는 vector를 제외한 나머지는 0으로 같다.

즉, column 개수에 rank를 뺀 게 nullity가 되는 건 당연한 거다~

Q: why do we care?

A: 주어진 A에 대한 optimal한 x를 찾는 AI 문제들을 수식화하기 위해서 많이 사용한다.

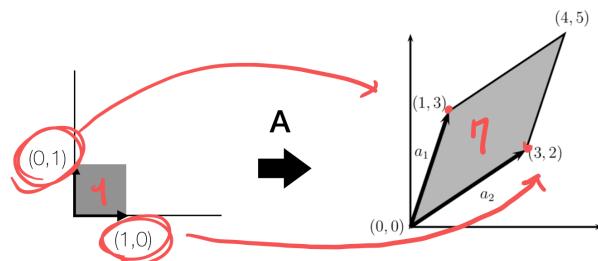
Q: How do we find the null space? \Rightarrow SVD

Determinant: span된 space의 볼륨을 측정하는 도구이다.

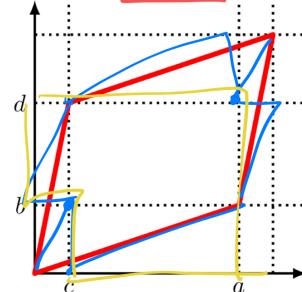
계산은 대각선으로 곱하면서 + / - 해서 합치는 과정 \leftarrow 아마 알거야

Here, the rows of the matrix are

$$\text{determinant}(A) = (1 \times 2 - 3 \times 3) = 7 \quad a_1 = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \quad a_2 = \begin{bmatrix} 3 \\ 2 \end{bmatrix}.$$



A wordless proof that the parallelogram defined by vectors (a, b) and (c, d) has area $ad - bc$.



matrix를 통해 변형된 평행사변형의 넓이를 측정하는 도구로, 기존의 벡터들이 형성하고 있던 영역이 matrix A를 곱했을 때 어떻게 바뀌는지를 알 수 있게 도와준다.

property - way to check invertible

$\det |A| = 0$ 이라면 A는 singular하다 (non-invertible, 역행렬이 존재하지 않음) \leftarrow if and only if 관계

$|A| \neq 0$, 즉 non-singular 하다면, $|A^{-1}| = 1 / |A|$ 가 성립해서 inverse의 determinant를 구할 수 있다

Quadratic forms: 그대로 해석하면 이차식이라는 뜻

주어진 square matrix A ($n \times n$)과 vector x ($\dim n$)이 있을 때, $x^T A x$ 를 **quadratic form**이라고 부른다.

$$x^T A x = \sum_{i=1}^n x_i (Ax)_i = \sum_{i=1}^n x_i (\sum_{j=1}^n A_{ij} x_j) = \sum_{i=1}^n \sum_{j=1}^n A_{ij} x_i x_j = \text{scalar output}$$

\Rightarrow quadratic form을 이용하면, 의미하는 바 그대로 2차식을 matrix와 vector의 곱형태로 나타낼 수 있다.

Eigenvalues and Eigenvectors

Eigenvector란 주어진 square matrix A에 대해 어떤 vector v를 곱했는데 A가 v의 방향은 바꾸지 않고, scale만 바꾸는 아주 특이한 경우를 의미한다!

$$Av = \lambda v \quad \text{scalar } \lambda \text{ is known as the eigenvalue} \quad \|v\|_2 = 1$$

if v is an eigenvector of A, so is any rescaled vector sv. \leftarrow 방향은 같으니 s배해도 eigenvector

그러므로 unit length인 eigenvector로 바꿔 넣어줄 수 있다!

Characteristic Polynomial: eigenvalue를 찾는 식

$$Av = \lambda v \Rightarrow (\lambda I - A)v = 0$$

⋮

$$\det(\lambda I - A)v = 0 \cdots \text{non-invertible}$$

$$P_A(\lambda) = \det(\lambda I - A) = 1 \times \lambda^n + c_{n-1} \times \lambda^{n-1} + \cdots + c_0 \quad (\text{the characteristic polynomial of } A)$$

⇒ if nonzero solution for v exists, 그에 대한 조건으로 non-invertible, 즉 가역행렬이 존재하면 안된다.

가역행렬이 존재할 경우, 0 벡터를 만드는 v 는 0벡터로 유일하기 때문이다.

- The characteristic polynomial is:

$$\det(\lambda I - A) = \det \begin{bmatrix} \lambda - 2 & -1 \\ -1 & \lambda - 2 \end{bmatrix} = 3 - 4\lambda + \lambda^2 = 0$$

- It has roots $\lambda = 1$ and $\lambda = 3$ which are the two eigenvalues of A .

- We can then solve for eigenvectors using $Av = \lambda v$:

$$v_{\lambda=1} = [1, -1]^T \quad \text{and} \quad v_{\lambda=3} = [1, 1]^T$$

Properties of eigenvalues and eigenvectors

$$\operatorname{tr} A = \sum_{i=1}^n \lambda_i \quad |A| = \prod_{i=1}^n \lambda_i$$

⇒ A 의 대각선 합은 eigenvalues의 합 / determinant A 는 eigenvalues의 곱

- the rank of A is equal to number of non-zero eigenvalues of A ← 중요한 척도

EigenDecomposition

$n \times n$ square matrix A 가 n linearly independent eigenvectors $\{v_1, \dots, v_n\}$ 과 그에 따른 eigenvalues $\{\lambda_1, \dots, \lambda_n\}$ 을 가지고 있다고 가정해보자 (cf: 선형독립으로 rank가 n 이라 eigenvalue도 n 개라는 걸 확인 가능...)

- concatenate eigenvectors as column to form matrix V
- concatenate eigenvalues to form vector $\lambda = [\lambda_1, \dots, \lambda_n]^T$

the eigendecomposition of A is given by:

$$AV = V \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}$$

A V V^T V Λ
 $\begin{bmatrix} A & & & & \end{bmatrix} \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}$
A Eigen vectors of A Eigen vectors of A Eigen values of A

$$AV = V \operatorname{diag}(\lambda)$$

$$\Rightarrow A = V \operatorname{diag}(\lambda) V^{-1}$$

AV = V diag(lambda) 는 $Av = \lambda v$ 의 식이 vector 형태로 합쳐져 있는 것으로 이해할 수 있다.

Details

Every **symmetric matrix** of dimension n has a set of **n orthogonal eigenvectors**

$$A = Q \Lambda Q^T$$

⇒ so can be decomposed into real-valued eigenvectors and eigenvalues like right equation:

- Q가 orthogonal하기에 inverse 와 transpose 동일

Singular Value Decomposition (SVD)

$$A = U \sum V^T \quad (A \in R^{m \times n})$$

- U는 $m \times m$ orthogonal matrix
- V는 $n \times n$ orthogonal matrix
- sigma는 $m \times n$ diagonal matrix로 sigma를 구성하는 요소들을 특이값이라 부른다. (크기 순서대로 내려온다 가정)

1) $m > n$

$$A = U \Sigma V^T = \begin{pmatrix} m \times n \\ \vdots & \dots & \vdots \\ u_1 & \dots & u_n \\ \vdots & \dots & \vdots \end{pmatrix} \begin{pmatrix} \dots & \vdots & \dots \\ u_m & & u_m \\ \vdots & & \vdots \end{pmatrix} \begin{pmatrix} n \times n \\ \sigma_1 & & & \sigma_n \\ \ddots & & & \\ & & 0 & \\ & & & 0 \\ m \times n & & & n \times n \end{pmatrix} \begin{pmatrix} n \times n \\ \dots & \vdots & \dots \\ v_1^T & & v_n^T \\ \vdots & & \vdots \\ v_n^T & & \vdots \end{pmatrix}$$

We can instead re-write the above as:

$$A = U_R \Sigma_R V^T$$

마지막 줄은 양곱해짐

⇒ $m < n$ 형태도 유사하게 나타난다.

Relation between SVD and Eigenvalue & EigenDecomposition

A를 SVD로 나타냈다고 하자, 그럼 $A^T A$ transpose와 A를 곱한 것을 봄보자

A square matrix $\longrightarrow A^T A = (U \Sigma V^T)^T (U \Sigma V^T)$

$$(V^T)^T (\Sigma)^T U^T (U \Sigma V^T) = V \Sigma^T U^T U \Sigma V^T = V \underbrace{\Sigma^T}_{\text{I}} \underbrace{\Sigma}_{\text{I}} V^T$$

Hence $A^T A = V \Sigma^2 V^T$ ← EigenDecomposition 형태

⇒ EigenDecomposition 형태이므로, V의 columns는 $A^T A$ 의 eigenvectors이며, 가운데의 시그마 제곱인 matrix의 diagonal entries는 $A^T A$ 의 eigenvalues이다.

λ the eigenvalues of $A^T A$, then $\sigma_i^2 = \lambda_i$

여기서 오메가는 처음 SVD에서의 가운데 diagonal matrix의 요소

비슷한 방법으로 AA^T 를 살펴보면

$$\begin{aligned} AA^T &= (\mathbf{U} \Sigma V^T) (\mathbf{U} \Sigma V^T)^T \\ &= (\mathbf{U} \Sigma V^T)(V^T)^T (\Sigma)^T \mathbf{U}^T = \mathbf{U} \Sigma \mathbf{V}^T \mathbf{V} \Sigma^T \mathbf{U}^T = \mathbf{U} \Sigma \Sigma^T \mathbf{U}^T \end{aligned}$$

⇒ 이번에는 \mathbf{U} 가 AA^T 의 eigenvectors가 되며, eigenvalue는 동일하게 시그마의 제곱 내에 있다.

Understanding Transformation Via SVD

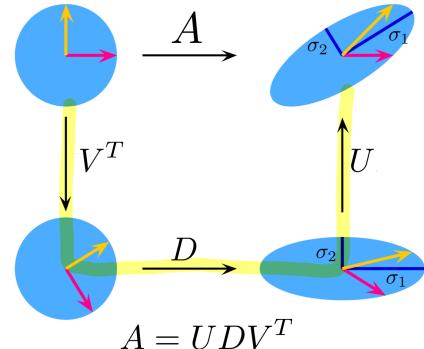
SVD를 geometric한 방식으로 이해해 보자. \mathbf{U}

$$\begin{aligned} A &= UDV^T = (UV^T)(VDV^T) \\ &\cdots (V \text{ is orthogonal}) \\ &= R(\theta)R(-\phi)R(\phi) \end{aligned}$$

그림으로 보면 SVD는 A 로 벡터를 변형시키는 과정이, V^T 로 다른 basis로 옮긴 후 D 에 의해 변형이 되고

\mathbf{U} 로 다시 원래의 basis로 돌리는 과정으로 이해 할 수 있다.

⇒ 즉, 다른 차원으로 건너가 변형이 일어나고 다시 돌아옴



Applications of SVD

1. matrix의 rank를 찾는 방법으로 SVD가 사용된다 (null space를 찾는데에도 사용된다고 앞에서 언급함)

$$\begin{aligned} \mathbf{A} &= \mathbf{U}_R \Sigma_R \mathbf{V}_R^T & k &= \min(m, n) \\ m \times n && m \times k & k \times n \\ && k \times k & & \\ \Sigma &= \begin{pmatrix} \sigma_1 & & & \\ & \ddots & & \\ 0 & & \sigma_k & \\ & & & \ddots \\ & & & 0 \end{pmatrix} & \Sigma &= \begin{pmatrix} \sigma_1 & & 0 & & \\ & \ddots & & \ddots & \\ 0 & & \sigma_k & 0 & \dots \\ & & & & 0 \end{pmatrix} \\ && & & \end{aligned}$$

If $\sigma_i \neq 0 \forall i$, then $\text{rank}(\mathbf{A}) = k$ (Full rank matrix)

⇒ in general, $\text{rank}(\mathbf{A}) = r$, where r is the number of non-zero singular values

2. Pseudo-inverse를 구하는데 사용

일반적으로 pseudo-inverse를 구하는 방식은 아래와 같지만 꽤 tricky함

$$\begin{aligned} \mathbf{A}^\dagger &= (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \quad \cdots \text{이 형태로 형성} \\ \mathbf{A}^\dagger \mathbf{A} &= (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{A} = (\mathbf{A}^T \mathbf{A})^{-1} (\mathbf{A}^T \mathbf{A}) = \mathbf{I} \\ \mathbf{A}^\dagger &= (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T = \mathbf{A}^{-1} \mathbf{A}^{-T} \mathbf{A}^T = \mathbf{A}^{-1} \mathbf{I} = \mathbf{A}^{-1} \quad \cdots \text{if } \mathbf{A} \text{ is invertible} \end{aligned}$$

SVD의 형태를 이용하면 더 쉽게 접근할 수 있다.

- When A is square

$$A^{-1} = VS^{-1}U^T$$

$$S = \begin{bmatrix} s_1 & & & \\ & s_2 & & \\ & & \ddots & \\ & & & s_n \end{bmatrix}$$

$$S^{-1} = \begin{bmatrix} \frac{1}{s_1} & & & \\ & \frac{1}{s_2} & & \\ & & \ddots & \\ & & & \frac{1}{s_n} \end{bmatrix}$$

$$\begin{aligned} A^{-1}A &= (VS^{-1}U^T)(USV^T) \\ &= VS^{-1}(U^TU)SV^T \\ &= V(S^{-1}S)V^T \\ &= VV^T \\ &= I \end{aligned}$$

- When A is non-square

$$A^\dagger = VS^\dagger U^T$$

$$S = \begin{bmatrix} s_1 & & & \\ & \ddots & & \\ & & s_k & \\ & & & 0 & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix}$$

$$S^\dagger = \begin{bmatrix} \frac{1}{s_1} & & & & & \\ & \ddots & & & & \\ & & \frac{1}{s_k} & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix}$$

↳ 0인 부분 존재

⇒ 공통적으로 singular value의 역수를 S의 pseudo inverse의 singular value로 취하고 있다.

이렇게 inverse를 구하는 방식은 least squares problem의 solution을 제공하는 역할도 한다

3. Low-Rank Approximation - image compression

$$\begin{aligned} A &= \begin{pmatrix} \vdots & \ddots \\ u_1 & \dots \\ \vdots & \ddots \end{pmatrix} \begin{pmatrix} \vdots \\ u_m \end{pmatrix} \begin{pmatrix} \sigma_1 & & & \\ & 0 & & \\ & & \ddots & \\ & & & \sigma_n \\ & & & 0 \\ & & & & \ddots \\ & & & & & 0 \end{pmatrix} \begin{pmatrix} \dots & v_1^T & \dots \\ \vdots & \vdots & \vdots \\ \dots & v_n^T & \dots \end{pmatrix} \\ &= \begin{pmatrix} \vdots & \dots & \vdots \\ u_1 & \dots & u_n \\ \vdots & \dots & \vdots \end{pmatrix} \begin{pmatrix} \dots & \sigma_1 v_1^T & \dots \\ \vdots & \vdots & \vdots \\ \dots & \sigma_n v_n^T & \dots \end{pmatrix} \\ &= \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \dots + \sigma_n u_n v_n^T \quad \text{Sum of "n" outer} \end{aligned}$$

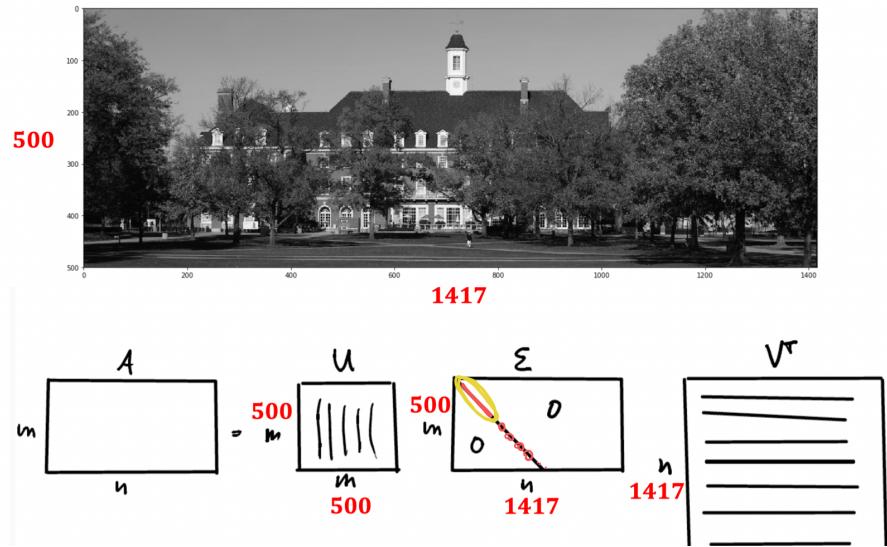
⇒ 그림처럼 A의 SVD를 outer product의 합으로도 쓸 수 있는데, 식을 살펴보면 singular vecotr가 0이냐 아니냐에 따라 벡터의 존재 유무, 즉 rank가 결정된다고 볼 수 있다.

예를 들어 A를 rank 3인 matrix로 간략화하고 싶다면 S의 원소(singular value)를 3개만 남기고 나머지를 다 0으로 만들면 된다.(singular value는 일반적으로 크기 순으로 쓰므로 위에서부터 남기면 된다)

$$\min_{A_k} \|A - A_k\| \text{ such that } \text{rank}(A_k) \leq k.$$

the best rank-k approximation은 원본 A와의 크기 차이가 얼마나지 않게 하는 것이며, 위에서 언급했듯이 가장 큰 singular value들을 순서대로 남기면 된다.

ex) Image compression



보통 image는 굉장히 많은 픽셀을 가지고 있으므로, rank도 어마어마하게 큰데 용량을 줄이거나 하는 문제로 rank를 줄일 때, 즉 사진을 압축할 때 많이 사용하는 방식이다.

2. Probability review

Random Variable

def: variable이라는 이름과 다르게 함수로, 특정 random한 이벤트의 결과를 scalar 값으로 mapping하는 함수

$$\Omega = \{\text{head}, \text{tail}\} \quad X = \begin{cases} 1, & \text{if heads} \\ 0, & \text{if tails} \end{cases} \quad P(X = 1) = 1/2$$

⇒ P는 가능성 **possibility**를 의미, random variable이 해당 값을 가질 경우를 care한다.

⇒ random variable을 쓰는 이유는 일어나는 event의 outcome을 길게 쓸 필요없이 scalar 값으로 표현이 가능한 편의성 때문에 사용한다. (그래서 대응되는 scalar값은 크게 중요치는 않다)

Discrete vs Continuous

- Discrete

$$X = \begin{cases} 1, & \text{if heads} \\ 0, & \text{if tails} \end{cases} \quad P[X = x] = f_X(x) \cdots \text{Discrete sample space}$$

⇒ discrete라는 이름에 걸맞게 상황이 잘 나눠져 있으며, 해당 value에 맞는 possibility가 function으로 딱 정해져 있다. - 이 함수를 Probability **mass** function (PMF) 라고 정의한다

- Continuous

$$X = \text{Exact amount of rain tomorrow}$$

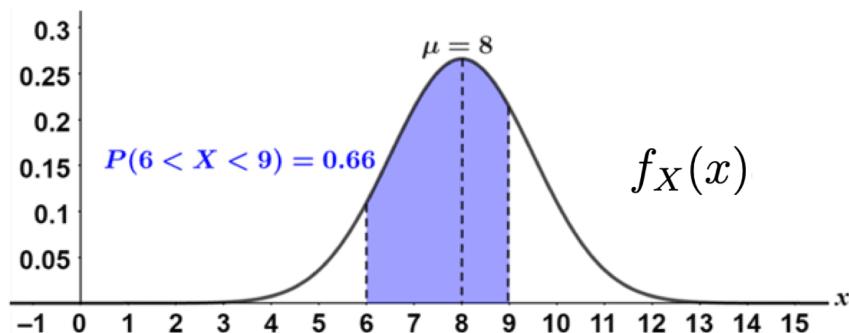
$$P[X \in A] = \int_A f_X(x)dx \cdots \text{Continuous sample space}$$

⇒ Continuous의 경우, probability가 continuous하게 정의되어 있으므로, 특정 Range인 X를 지정해서 함수를 적분해서 probability를 구해야 한다. - 이 함수를 Probability **density** function (PDF)라고 한다.

대표적인 예시로는 normal (gaussian) distribution이 있다



$X = \text{Exact amount of weight of a mouse}$



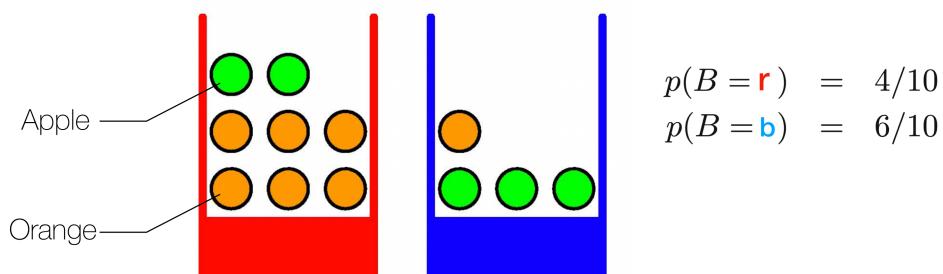
⇒ 그림처럼 continuous sample space의 경우, probability를 얘기할 때 특정 range를 통해 가야 한다.
적분한 부분 (그림의 underneath)가 probability가 되고, PDF의 Y-value는 probability density라 부른다.

PDF에는 두 가지 조건이 있는데,

- $f(x)$ should be non-negative for all values of the random variable.
- the area underneath of $f(x)$ should be equal to 1.

Example of Probability Theory

1. two colored boxes with apples and oranges

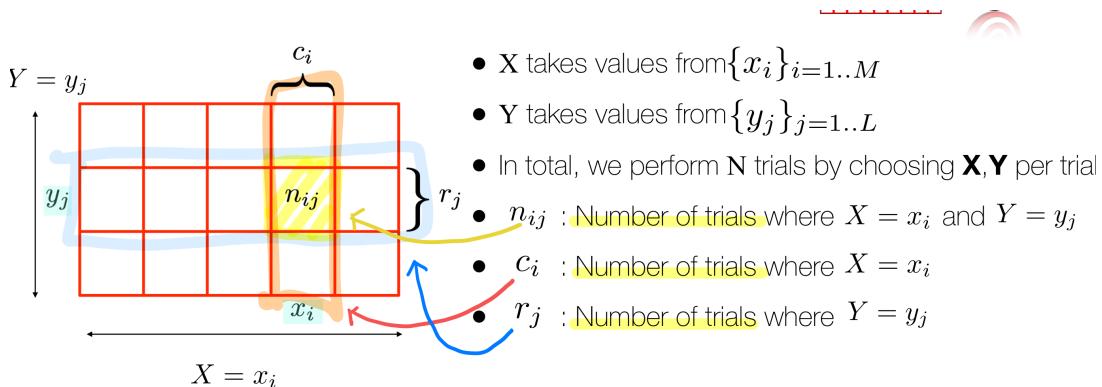


⇒ 여기서 $B = r$ or b 는 box를 선택하는 random variable의 값, $F = a$ or o 는 apple이랑 orange로 정의

- Q1: what's the overall probability that we pick an apple? $P(F=a) = ?$
first, consider the cases that we pick red box or blue box.
Pick red box = $0.4 \times 2/8 = 0.1$, Pick blue box = $0.6 \times 3/4 = 0.45$
sum the probability of each case: 0.55

- Q2. Given that we have chosen an apple, what's the probability that we chose the red box?
질문 1번에서는 red box를 고르고 거기서 apple을 골랐는데, 여기서는 반대로 내가 apple을 고른 상황 중 red box에 담겨 있을 가능성을 의미하는 것이다. ⇒ $P(B=r | F=a) = ?$
생각해보면 red box + apple을 고른 경우를 apple을 고른 경우로 나누면 되지 않을까? 왜냐하면 이미 given chosen apple이라고 apple을 고른 경우를 제시했기 때문에, apple을 고를 확률을 상정할 필요X
 $P(B=r|F=a) = P(B=r) P(F=a|B=r) / P(F=a) = 0.1 / 0.55 = 2/11$ (분모는 red box에서 apple 고르기)

2. Frequentist's view with two random variable X and Y



$$\text{joint probability} : p(X = x_i, Y = y_j) = \frac{n_{ij}}{N}$$

$$\text{marginal probability} : p(X = x_i) = \frac{c_i}{N}, p(Y = y_i) = \frac{r_j}{N}$$

$$\text{sum role(or)} : p(X = x_i) = \sum_{j=1}^L p(X = x_i, Y = y_j)$$

$$\text{conditional probability} : p(Y = y_j | X = x_i) = \frac{n_{ij}}{c_i} = \frac{p(X = x_i, Y = y_j)}{p(X = x_i)}$$

$$\begin{aligned} \text{Product rule(and)} &: p(X = x_i, Y = y_j) = p(Y = y_j | X = x_i)p(X = x_i) \\ &= \frac{n_{ij}}{c_i} \cdot \frac{c_i}{N} = p(X = x_i | Y = y_j)p(Y = y_j) \end{aligned}$$

sum role을 or이라고 부르는 이유는 한 조각, 한 조각을 or로 붙여서 저 안에 해당되면 되기 때문이다.
conditional probability는 x_i 를 골랐는데 이 영역인 y_j 일 가능성으로, 앞에서 푼 Q2와 동일한 내용이다.
Product rule을 and라고 부르는 이유는 식에서 볼 수 있는 것처럼, x_i 를 고를 가능성과 그 x_i 중 y_j 를 고를 가능성을 곱해 해당되는 특정 영역(i,j)을 고를 가능성으로 좁히는 교집합(and)과 같은 역할이기 때문이다.

Bayes' Theorem

$$\text{Bayes' Theorem} : p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}$$

위에서 언급한 Product rule에서 기인한 형태이다. 중요한 이유는 순서를 바꾸면 matrix를 계산하기 편해진다.

Independence in Probability Theory

two events X and T are **independent** if and only if $P(X, T) = P(X)P(T)$

⇒ X가 어떤 결과를 내는지에 따라 Y가 달라지지 않는다는 의미이다.(반대도 동일),

⇒ 만약 달라졌다면? $P(X, Y) = P(Y|X)P(X)$ or $P(X|Y)P(Y)$ 로 나타내야 한다. (한 이벤트가 발생한다는 걸 특정지어(X) 그 속에서 다음 이벤트(Y)가 생길 가능성을 곱하는 형태)

- Equivalently:

$$P(X|Y) = \frac{P(X,Y)}{P(Y)} = P(X)$$

↳ Y 중 X, Y인 것 = X인 것
⇒ Y는 X에 영향X

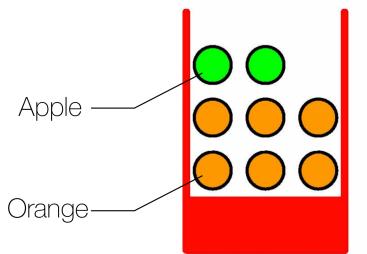
and similarly

$$P(Y|X) = \frac{P(X,Y)}{P(X)} = P(Y)$$

↳ X 중 X, Y인 것

⇒ 이런 식으로 좁혀서 쓸 수 있다.

Back to our example



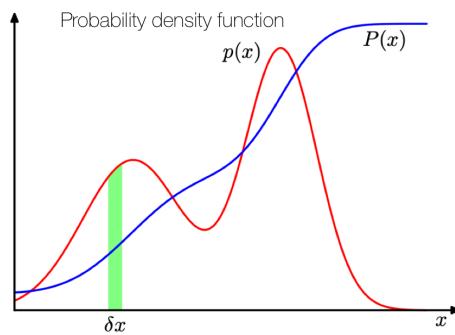
$$\begin{aligned} p(B = r) &= 4/10 \\ p(B = b) &= 6/10 \\ p(F = a|B = r) &= 1/4 \\ p(F = o|B = r) &= 3/4 \\ p(F = a|B = b) &= 3/4 \\ p(F = o|B = b) &= 1/4 \end{aligned}$$

- Q1. what's the overall probability that we pick an apple? ⇒ $P(F=a) = ?$
B의 case 모두에 대해 $F=a$ 를 고를 경우의 확률을 더하면 된다.

$$P(F=a) = P(F=a|B=r)P(B=r) + P(F=a|B=b)P(B=b) = 1/10 + 9/20 = 11/20$$
+ by sum rule, $p(F=o) = 1 - 11/20 = 9/20$
- Q2. Given that we have chosen an orange, what's the probability that we chose the red box?

via bayes' theorem: $p(B = r | F = o) = p(F = o | B = r) p(B = r) / p(F = o) = 3/4 \times 4/10 \times 20/9 = 2/3$

summary of PDF with continuous random variables



- Cumulative distribution function

$$P(z) = \int_{-\infty}^z p(x) dx$$

- $p(x)$ must satisfy:

$$\begin{aligned} p(x) &\geq 0 \\ \int_{-\infty}^{\infty} p(x) dx &= 1 \end{aligned}$$

- Probability that x lies (a,b):

$$p(x \in (a, b)) = \int_a^b p(x) dx$$

Sum rule $p(x) = \int p(x,y) dy$

Product rule $p(x,y) = p(y|x)p(x).$

Basic concept

Expectation: 기댓값은 어떤 확률 변수의 평균 값을 의미, 변수의 분포에 따라 값이 어떻게 퍼져있는지를 나타내며, 특정 확률 변수의 값이 반복적으로 관찰될 때 예상되는 평균 결과를 수치적으로 표현합니다.

ex) 주사위의 기댓값은 3.5

For a function $f(x)$ under a probability distribution $p(x)$

χ

Discrete variable $\mathbb{E}[f] = \sum_x p(x)f(x)$

Continuous variable $\mathbb{E}[f] = \int p(x)f(x)dx$

Conditional expectation $\mathbb{E}[f|y] = \sum_x p(x|y)f(x)$

Conditional distribution

Variance: 분산, 데이터나 확률변수가 **기대값(평균)**을 기준으로 얼마나 흩어져 있는지를 측정하는 값입니다. 분산은 확률 분포의 퍼짐 정도를 나타내므로, 값이 클수록 데이터가 평균에서 멀리 분포하고, 값이 작을수록 평균에 가깝게 분포합니다.

$$\begin{aligned} \text{var}[f] &= \mathbb{E}[(f(x) - \mathbb{E}[f(x)])^2] && \text{IE } \rightarrow \\ \text{var}[f] &= \mathbb{E}[f(x)^2] - \mathbb{E}[f(x)]^2 && \text{linearity 가능} \\ \text{var}[x] &= \mathbb{E}[x^2] - \mathbb{E}[x]^2 \\ &\quad (\text{when } f(x) = x) \end{aligned}$$

Covariance: 두 변수가 얼마나 함께 변하는지를 의미한다. ← 식을 보면 variance에서 변형한 형태

$$\begin{aligned} \text{cov}[x, y] &= \mathbb{E}_{x,y} [\{x - \mathbb{E}[x]\} \{y - \mathbb{E}[y]\}] \\ &= \mathbb{E}_{x,y}[xy] - \mathbb{E}[x]\mathbb{E}[y] \end{aligned}$$

↑
상관성을 아는 거라
값은 크게 중요X

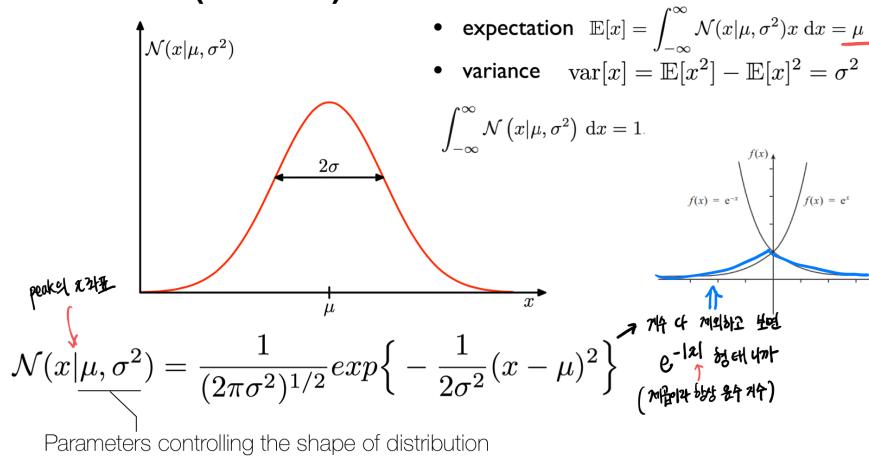
Note: $\text{cov}[x, x] = \text{var}[x]$

$\text{cov}[x,y] > 0$ 이면, x가 커질 때 y도 커짐 / $\text{cov}[x,y] < 0$ 이면, x가 커질 때 y는 작아짐

- In the case of two vectors of random variables \mathbf{x} and \mathbf{y} :

$$\begin{aligned} \text{cov}[\mathbf{x}, \mathbf{y}] &= \mathbb{E}_{\mathbf{x}, \mathbf{y}} [\{\mathbf{x} - \mathbb{E}[\mathbf{x}]\}\{\mathbf{y}^T - \mathbb{E}[\mathbf{y}^T]\}] \\ &= \mathbb{E}_{\mathbf{x}, \mathbf{y}}[\mathbf{x}\mathbf{y}^T] - \mathbb{E}[\mathbf{x}]\mathbb{E}[\mathbf{y}^T]. \end{aligned}$$

Gaussian (Normal) Distribution - PDF function의 대표적 예시



peak를 담당하는 parameter는 가우시안의 정중앙 부분으로 양옆의 모양이 같고, 당연하게도 기댓값이 된다.

파라미터를 조절해서 그래프(함수)를 만들기 때문에, 파라미터 2개를 찾는 것이 목적으로 많이 나온다.

Multivariate Gaussian

$$N(\mathbf{x}|\mu, \Sigma) = \frac{1}{(2\pi)^D/2} \frac{1}{|\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu) \right\},$$

Inverse of Covariance matrix
 Σ 의 역 행렬 (quadratic form)

Mahalanobis distance

Determinant

D-dim mean Covariance matrix (DxD)

x처럼 단일 변수가 아닌, 다변수 가우스 함수도 존재한다. expectation은 x와 같은 dimension, variance는 D x D인 matrix를 가진다. 이 matrix를 **covariance matrix**라고 부른다.

Covariance Matrix

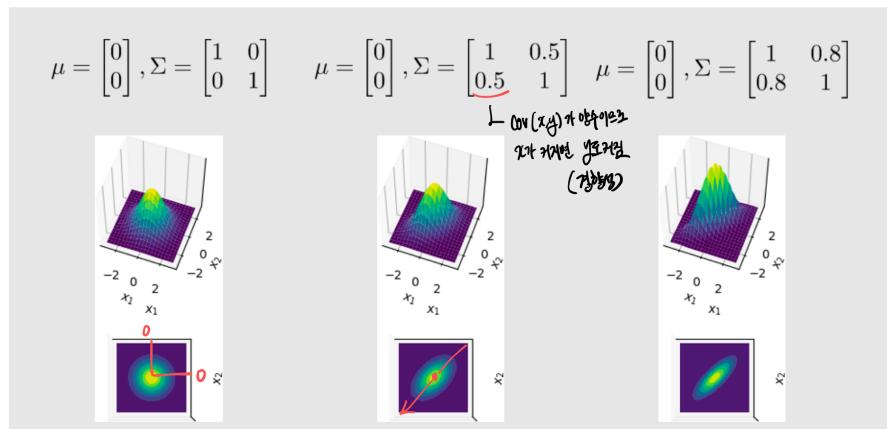
- For a random vector $X : \Omega \rightarrow \mathbb{R}^n$
- Covariance matrix is a $n \times n$ square matrix:

$$\begin{aligned} \Sigma &= \begin{bmatrix} Cov[X_1, X_1] & \dots & Cov[X_1, X_n] \\ \vdots & \ddots & \vdots \\ Cov[X_n, X_1] & \dots & Cov[X_n, X_n] \end{bmatrix} & cov[x, x] &= var[x] \\ &= \begin{bmatrix} E[X_1^2] - E[X_1]E[X_1] & \dots & E[X_1X_n] - E[X_1]E[X_n] \\ \vdots & \ddots & \vdots \\ E[X_nX_1] - E[X_n]E[X_1] & \dots & E[X_n^2] - E[X_n]E[X_n] \end{bmatrix} & cov[x, y] &= \mathbb{E}_{x,y}[(x - \mathbb{E}[x])(y - \mathbb{E}[y])] \\ &= \begin{bmatrix} E[X_1^2] & \dots & E[X_1X_n] \\ \vdots & \ddots & \vdots \\ E[X_nX_1] & \dots & E[X_n^2] \end{bmatrix} - \begin{bmatrix} E[X_1]E[X_1] & \dots & E[X_1]E[X_n] \\ \vdots & \ddots & \vdots \\ E[X_n]E[X_1] & \dots & E[X_n]E[X_n] \end{bmatrix} & & \\ &= E[XX^T] - E[X]E[X]^T & = E[(X - E[X])(X - E[X])^T]. \end{aligned}$$

- Symmetric
- ✓ • Positive semi-definite
• 1차원의 0 이상

$$\begin{aligned} cov[x, y] &= \mathbb{E}_{x,y}[(x - \mathbb{E}[x])(y - \mathbb{E}[y])] \\ &= \mathbb{E}_{x,y}[xy] - \mathbb{E}[x]\mathbb{E}[y] \end{aligned}$$

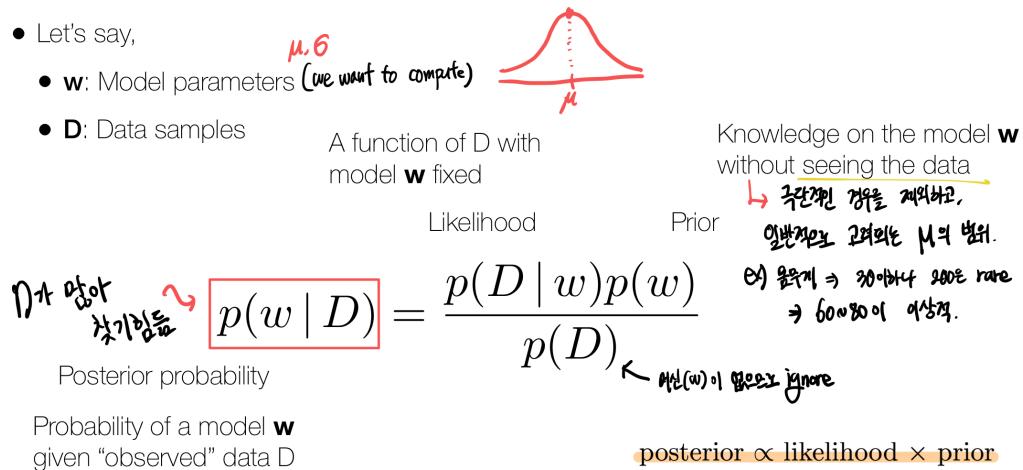
⇒ 1차원에서 scalar로만 나오던 covariance를 matrix form으로 확장한 형태로, 모든 X의 요소에 대해 하나하나 1x1로 covariance를 적용한 matrix이다. 대각 요소들이 0 이상이고 symmetric하다는 특징이 있다.



→ 각각의 변수에 대한 경향성을 담고 있다. 2번 예시처럼, 대각선이 아닌 요소들을 봤을 때, 각 변수 간의 관계를 짐작해 볼 수 있다. "cov[x,y]가 양수이므로 x와 y가 함께 증가하는군"

Revisiting Bayes' Theorem

: 주어진 아래 그림은 베이즈 정리를 활용해 확률을 통한 불확실성의 정량화를 나타내고 있다.



목표 : Data sample이 주어져 있고, 이를 통해 우리는 w 를 구해야 한다.

식의 구성

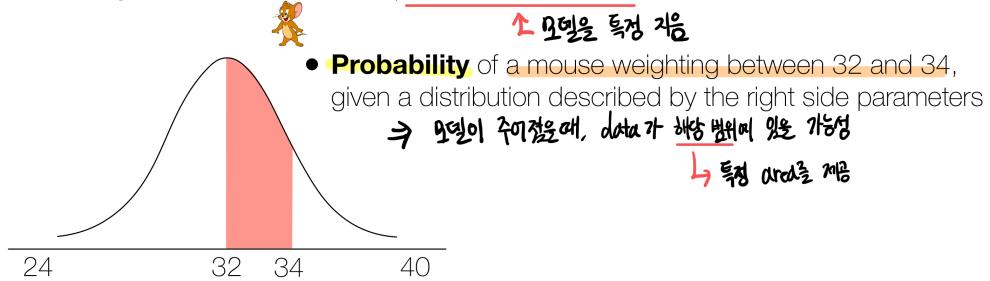
- **Prior (사전 확률) $p(w)$** : 데이터를 보기전, 모델 파라미터 w 에 대해 우리가 가지고 있는 선형적 지식을 의미한다. 극단적인 경우를 제외하고 일반적으로 우리가 고려하는 expectation의 예상 범위라 할 수 있다.
- **Likelihood $p(D|w)$** : 주어진 파라미터 w 에 대해 데이터를 얻을 가능성은 나타낸다. 이는 모델 w 가 고정되어 있을 때 데이터 D 가 나올 확률을 의미한다. w 를 구할 때 이 가능성을 보고 w 를 조절시키는 방식으로 사용한다.
- **Posterior (사후확률) $p(w|D)$** : 우리가 최종적으로 구해야 할 값, 주어진 데이터 D 를 관찰한 후, 모델 파라미터 w 를 가질 확률을 의미한다.

posterior은 D 가 많아서 조건을 만족하는 w 를 찾기 힘들다. 그래서 베이즈 정리를 통해 식을 바꾸어, 대략적인 w 를 선정하고 모델 w 를 계속 업데이트 해서 $p(w|D)$ 의 값을 높이는 방향으로 진행한다. ⇒ 식 대신에 그림에 표시된 비례식을 쓰기도 한다.

Probability vs. Likelihood

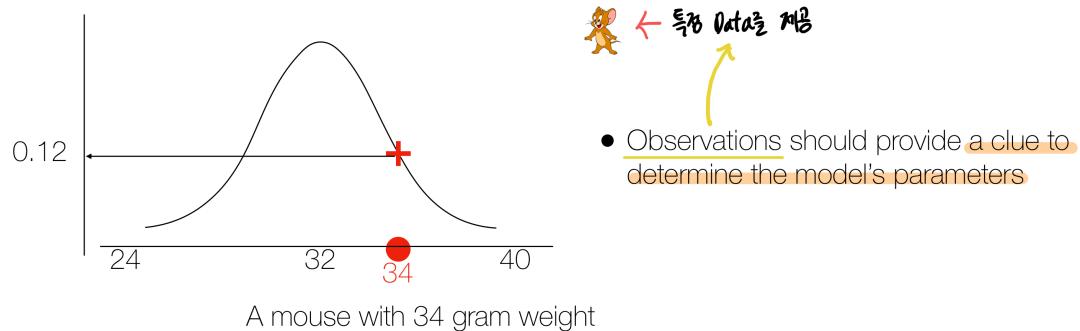
: $P(\text{data} \mid \text{distribution})$ vs. $L(\text{distribution} \mid \text{data})$

- **Probability:** $P(32 < \text{Weight} < 34 | \mu = 32, \sigma = 2.5)$

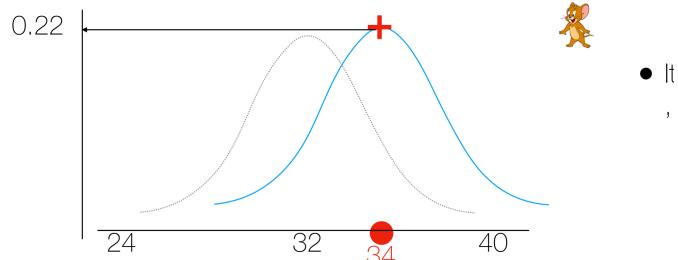


\Rightarrow Probability is the area under a fixed distribution

- **Likelihood:** $L(\mu = 32, \sigma = 2.5 | \text{weight} = 34)$



- **Likelihood:** $L(\mu = 34, \sigma = 2.5 | \text{weight} = 34)$



\Rightarrow Likelihood is the y-axis values for fixed data points with a movable distribution

\Rightarrow likelihood는 probability와 다르게 “mouse가 34g이다”라는 정보가 주어지면, 주어진 parameter로 만들어지는 distribution에 대해 특정 Y-value(probability degree)를 제공해준다.

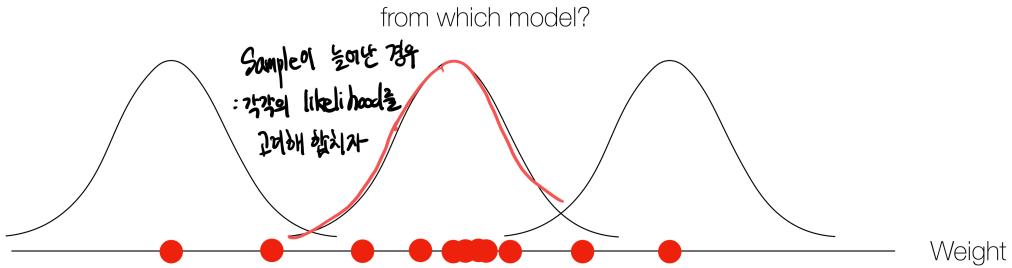
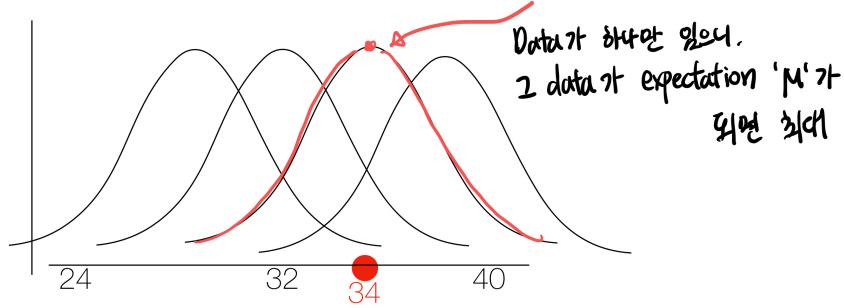
\Rightarrow 그림처럼 파라미터에 따라 값이 달라지므로, Likelihood는 파라미터에 대한 function이며, 제공 받은 Y-value 통해 파라미터 값을 조정해서 모델을 최적화 하는 방식으로 활용한다. -

MLE가 그 방법

Maximum Likelihood Estimation (MLE)

: 모델을 최적화하기 위해 사용하는 방식으로, model의 파라미터를 likelihood를 최대화하는가?를 기준으로 평가

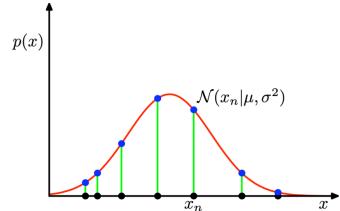
- Given observations, which model has the maximum likelihood?



⇒ 우리는 당연하게도 sample의 개수가 많은 경우를 생각해야 한다.

Maximizing Likelihood

Assuming independent and identically distribution (i.i.d)



- Maximum likelihood** $p(x|\mu, \sigma^2) = \prod_{n=1}^N \mathcal{N}(x_n|\mu, \sigma^2)$ y-axis values for fixed data points
- Log likelihood** $\ln p(x|\mu, \sigma^2) = -\frac{1}{2\sigma^2} \sum_{n=1}^N (x_n - \mu)^2 - \frac{N}{2} \ln \sigma^2 - \frac{N}{2} \ln(2\pi)$

Maximum likelihood 식에서 볼 수 있는 것처럼, 특정 값으로 parameter를 먼저 고정시켜 놓고 N개의 data에 대해 나온 Y-value를 모두 곱한다. ⇒ expectation과 variance, 두 파라미터를 조정해서 언제 likelihood가 최대가 되는지를 찾으면 됨다.

소수점이 크게 나오기 때문에, log를 사용하면 조금 더 편리하게 사용할 수 있다. 저 값을 크게만 만들면 된다.

- Sample mean via ML** $\mu_{ML} = \frac{1}{N} \sum_{n=1}^N x_n$ (평균) (By solving over μ)
(maximizing log likelihood)
- Sample variance via ML** $\sigma_{ML}^2 = \frac{1}{N} \sum_{n=1}^N (x_n - \mu_{ML})^2$ (By solving over σ^2)
(maximizing log likelihood)

<http://ermakov.github.io/machinelearning/2017/08/12/>

⇒ log likelihood로 얻을 수 있는 값으로, expectation은 data 각각의 x-value 평균값이고, data와 평균값을 빼서 normalization한 게 variance가 된다.

Information Theory

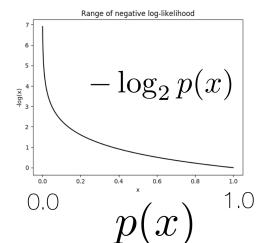
: How to measure the amount of information?

여기서의 information은 '놀랄 정도', 정보의 양은 사건이 발생했을 때 얼마나 예상 밖인지를 측정하는 요소.

- Let $h(x)$ is the **quantity of information** conveyed via the observation x
lower probability of x ,
 $p(x)$ means higher information $h(x)$ - 반비례 관계, 일어날 확률이 적을 수록 예상 밖의 일이라고 생각할 정도가 커지기 때문.
- Intuitively, we consider the sum of two independent observations: $h(x,y) = h(x) + h(y)$
for statistically independent event x,y : $p(x,y) = p(x)p(y)$
⇒ information과 probability 사이에 로그적 관계가 적용된다. 아래와 같이 정의

- Quantity of information $h(x)$:

$$h(x) = -\log_2 p(x)$$



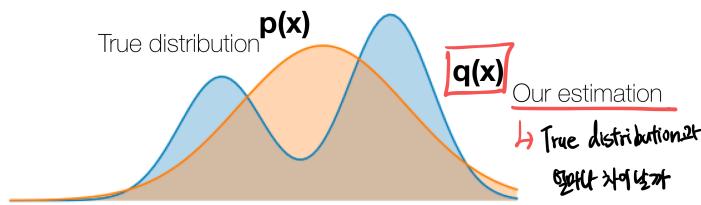
Entropy: the average amount of information when transmitting(전송) a random variable x

$$H[x] = - \sum_x p(x) \log_2 p(x) = \sum_x p(x) h(x)$$

⇒ expectation $E[h]$ 와 같은 역할을 본다고 할 수 있다.

Kullback-Leibler (KL) Divergence

(a.k.a) relative entropy



: $p(x)$ 와 $q(x)$ 두 distribution 사이의 차이를 측정하고자 할 때 사용한다. - 보통 True distribution과 our estimation을 비교한다.(모델이 얼마나 좋은지 나쁜지를 가르는 척도)

$$\begin{aligned} \text{KL}(p||q) &= - \int p(\mathbf{x}) \ln q(\mathbf{x}) d\mathbf{x} - \left(- \int p(\mathbf{x}) \ln p(\mathbf{x}) d\mathbf{x} \right) \\ &= - \int p(\mathbf{x}) \ln \left\{ \frac{q(\mathbf{x})}{p(\mathbf{x})} \right\} d\mathbf{x}. \end{aligned}$$

Recall: entropy $H[x] = - \sum_x p(x) \log_2 p(x)$

$$= H(p, q) - H(p)$$

Cross-entropy Entropy

cross-entropy: p 에 대한 q 의 상대적 entropy를 의미

cross-entropy와 entropy의 차이로, 예상하는 $q(x)$ 가 $p(x)$ 에 비해 얼마나 안좋은지를 측정하는 수단

주어진 distribution이 continuous하므로 entropy에서 sigma 대신 integral을 사용한다는 점을 유의

Properties

- 교환법칙 성립 $x : KL(p||q) \neq KL(q||p)$
- $KL(p||q) \geq 0$ (cross-entropy는 항상 entropy - lower bound 보다 크기 때문에 0 이상, 0일 때는 $p=q$)

Mutual Information

: 두 Random Variable이 얼마나 independent한지를 측정하는 수단

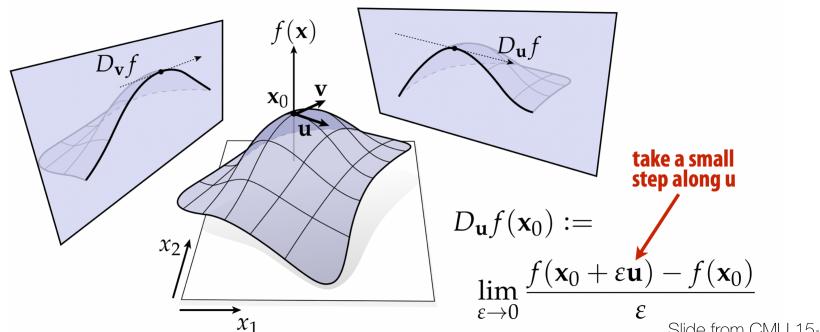
$$I[x, y] = KL(p(x, y)||p(x)p(y)) = - \iint p(x, y) \ln\left(\frac{p(x)p(y)}{p(x, y)}\right) dx dy$$

x 와 y 가 independent하면 $I(x, y) = 0$ (if and only if 관계), 그 외는 당연히 0보다 크다

3. Optimization Basics

Basic concept

Directional Derivative in Multivariate Functions



⇒ 다변수 함수의 미분의 경우 function을 어떤 line에 대해 slice하는 방식이다. x, y 로 정해진 basis가 아닌 특정 방향으로 미분할 수 있으며, usual derivative한 방식이다. D 아래에 본인이 미분할 방향 벡터를 적어놓는다.

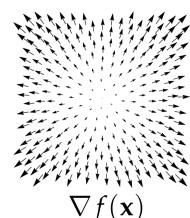
Gradient and Partial Derivative

Let a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$. The **gradient of f** is the **column vector of partial derivatives**

$$\nabla f(x) = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} = 2x$$

Gradient $\nabla f(x) := \begin{pmatrix} \frac{\partial f(x)}{\partial x_1} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{pmatrix}$ ← vector form

Nabla symbol



⇒ 특정함수에 대한 편미분을 모아놓은, 변수마다의 변화율을 나타내는 vector 형태이다. 여기서 나온 그래디언트 벡터는 특정 점에서 가장 함수가 빠르게 증가하는 방향을 가리킨다. (크기는 그 방향의 변화율)

Jacobian

야코비 행렬은 gradient와 유사하지만 미분하는 함수의 형태가 다르다. 함수가 하나의 output이 아닌 여러 output을 내며, 그렇기에 함수 f 속에 각 output과 대응되는 함수가 벡터형태로 있다.

Jacobian of vector function $f : R^n \rightarrow R^m \dots f = [f_1 \ f_2 \ \dots \ f_m]^T$

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1} & \dots & \frac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \nabla^T f_1 \\ \vdots \\ \nabla^T f_m \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \dots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

m × n matrix

Hessian Matrix

f 를 두 번 미분가능한 다변수 함수라고 할 때, hessian matrix는 f 의 이계도함수로 구성된 matrix이다.

$$\mathbf{H}_f = \nabla^2 f(\mathbf{x}) := \begin{pmatrix} \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_1} & \dots & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_1} & \dots & \frac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_n} \end{pmatrix}$$

n × n square matrix

⇒ 만약 모든 이계도함수가 continuous하다면, hessian matrix는 symmetric하다

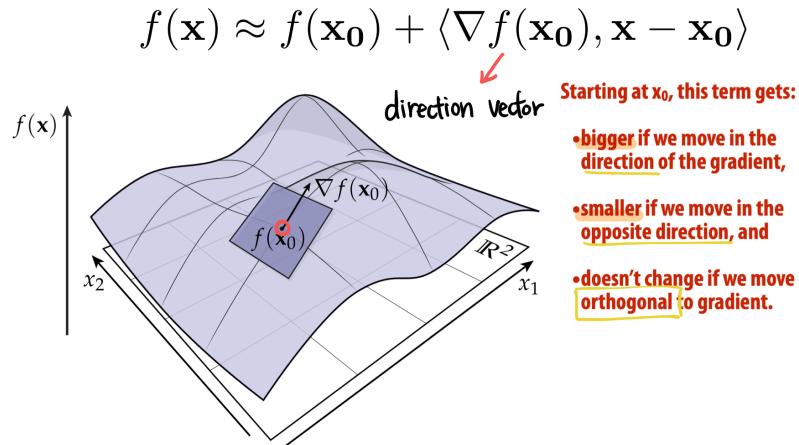
⇒ Hessian Matrix를 다르게 보면 f 의 gradient의 jacobian으로 볼 수 있다. (f 의 gradient는 한번 미분된 함수들로 구성된 벡터들, 이의 jacobian은 각 함수들을 모든 변수들에 대해 미분하는 걸로, 즉 변수별로 2번씩 미분한 행렬인 hessian matrix와 동일함)

Taylor Series for Multivariate Functions

plain taylor series: 함수 및 함수값 유추에 용이한 방법

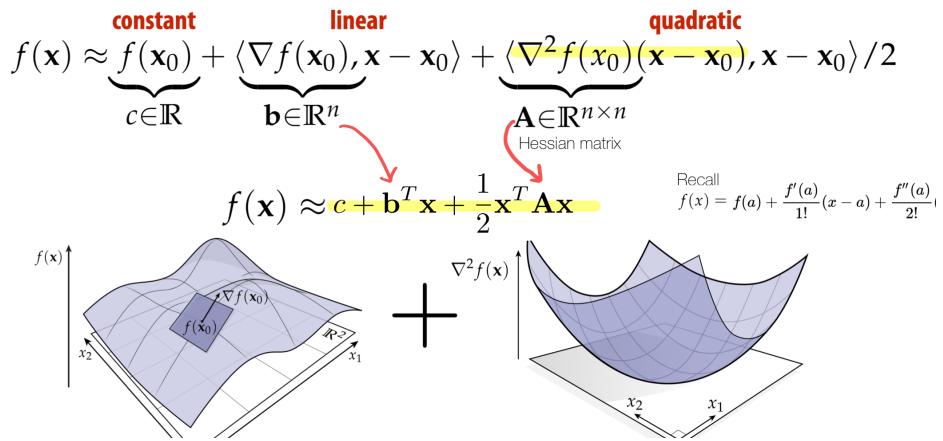
$$\begin{aligned} f(x) &= f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \dots + \frac{f'''(a)}{3!}(x-a)^3 + \dots \\ &= \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \end{aligned}$$

- 1st-order Approximation



⇒ 여기서는 gradient의 역할로서, $\mathbf{x}-\mathbf{x}_0$ 가 gradient의 방향으로 가면 함수값이 증가하고, 반대 방향으로 가면 함수값이 감소하고, 직교하면 변화가 없다는 일종의 함수 값의 변화를 추측하는 역할이다.

- 2nd-order Approximation



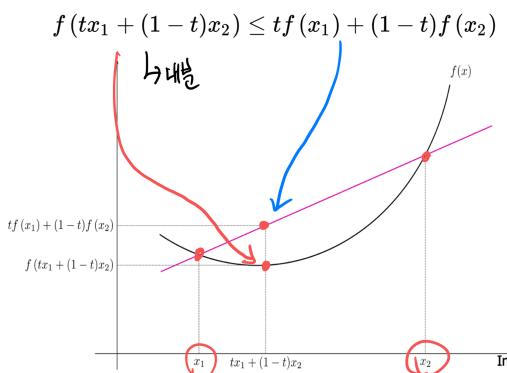
⇒ 단변수 함수일 때와 유사한 형태로 이계도함수 대신 Hessian matrix가 그 자리를 차지한다. 저 덧셈의 의미는 linear 형태로 대략적인 값을 잡아놓았다면, quadratic 형태는 real function과 linear 함수의 간격을 좁히는 역할을 한다고 볼 수 있다.

Optimization Problem

$$\begin{aligned} & \underset{\mathbf{x}}{\text{minimize}} \quad f(\mathbf{x}) \quad \cdots \text{objective function} \quad f : \mathbb{R}^n \rightarrow \mathbb{R} \\ & \text{subject to} \quad g_i(\mathbf{x}) \leq 0, \quad i = 1, \dots, m \quad \cdots \text{inequality constraints} \\ & \quad f_j(\mathbf{x}) = 0, \quad j = 1, \dots, p \quad \cdots \text{equality constraints} \end{aligned}$$

⇒ 이런 최소화를 해야하는 문제를 어떻게 해결할지가 메인으로 다룰 주제이다.

Convex Functions



A function $f : X \rightarrow \mathbb{R}$ is convex if X is convex and:

1. The epigraph of f is a convex set
(all points above a graph)

$$\text{epi } f = \{(x, t) \mid x \in X, f(x) \leq t\}$$

2. Jensen's Inequality: For all $x, y \in X$ and $0 \leq \alpha \leq 1$

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y)$$

3. The function f is twice differentiable and its Hessian

$\nabla^2 f(x)$ is positive semi-definite for all $x \in X$.
이제 함수가 양수인 느낌 Recall: $\mathbf{z}^T \mathbf{A} \mathbf{z} \geq 0$, for all \mathbf{z}

⇒ 쉽게 말해 아래로 볼록을 의미한다. convex function에서의 local minimum은 global minimum을 의미한다. 추가로 second derivative is positive everywhere.

⇒ 오른쪽은 다변수 함수에 대한 convexity를 의미. epigraph는 두 점을 고르고 그 점을 이었을 때, 나오는 직선이 그래프 아래에 위치하면 안된다는 조건을 의미한다.

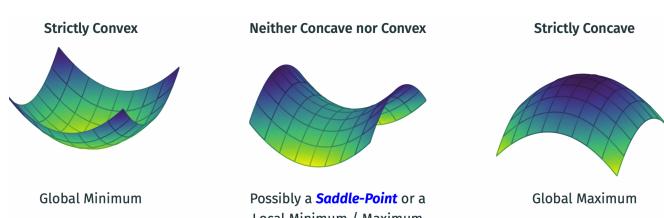
⇒ convex에 대해서 배우는 이유는 optimization하기 쉽기 때문이다. 왜냐하면 convex function의 경우 only one minimizer, 즉 Local minimum = global minimum이기 때문에 그냥 함수를 타고 내려가면 된다.

⇒

gradient descent에서 이에 관한 논의!!

How to recognize a minimum?

f 가 미분 가능하다고 가정해보자. 극점(extrema)이면, gradient $(x) = 0$ 이다. 이들이 후보가 된다.



convex한 경우는 local이 global minimum이지만, 다른 함수의 경우 안장점이 되거나, global maximum이 된다.

⇒ 여튼 convex에 관해서는 gradient = 0인 x 를 찾자

Gradient Descent

- 처음에 임의의 시작점인 x_0 를 잡는다.
- step to the next point x_{k+1} in the direction of the negative gradient
⇒ negative gradient

면 함수값이 작아지는 방향이기 때문에, 그 방향으로 x 값을 줄여나가야 한다.

$$x_{k+1} = x_k - \nabla f(x_k) \quad \dots \quad \text{repeat until } \|\nabla f(x_k)\| < \epsilon$$

- recall 1st order taylor series

$$f(\mathbf{x}) \approx f(\mathbf{x}_k) + \langle \nabla f(\mathbf{x}_k), \mathbf{x} - \mathbf{x}_k \rangle \quad (\text{Let } \mathbf{x} - \mathbf{x}_k = \mathbf{h})$$

$$f(\mathbf{x}_k + \mathbf{h}) \approx f(\mathbf{x}_k) + \mathbf{h}^T \nabla f(\mathbf{x}_k) \quad \|\mathbf{h}\| = 1$$

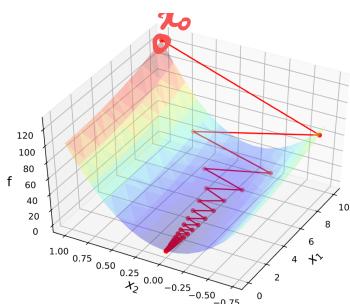
$$\arg \min_{\mathbf{h}} f(\mathbf{x}_k) + \mathbf{h}^T \nabla f(\mathbf{x}_k)$$

Minimum when $\mathbf{h} = -\frac{\nabla f(\mathbf{x}_k)}{\|\nabla f(\mathbf{x}_k)\|_2}$

위에서 언급한대로, our goal은 왼쪽의 그림처럼 최소값을 찾아야 한다.

이때 \mathbf{h} 는 gradient에 완전히 반대되는 형태일 수록 좋다.

⇒ 값이 최대한 많이 줄어들기 때문.



위의 식에서는 negative gradient 값으로 1배수 만큼 줄였는데 상수 배해서 줄여갈 수도 있다.

⇒ **step length**를 얼마로 하는지 관건인 문제이다. step-size를 크게하면 많은 계산없이 최소점에 도달할 수도 있다는 점이 있다. (대신 정확성은 떨어질 수 있다)

$$x_{k+1} = x_k - \alpha \nabla f(x_k) \\ (\alpha : \text{step length})$$

Least-Squares Problem

: most commonly하게 ML에서 observed되는 방식으로, Convex optimization의 subclass이다.

$$\text{minimize } \frac{1}{2} \sum_{j=1}^m r_j^2(x) \quad \cdots \quad r_j^2 = (z_j - f_j(x))^2 \leftarrow \text{Squared error}$$

⇒ true data와 estimation의 차이의 제곱들의 합을 최소화하는 문제로, unconstrained optimization problem이다. 저 함수 자체로 일종의 error function의 역할을 한다.

- ex) polynomial fitting problem

- Input: $\mathbf{x} \equiv (x_1, \dots, x_N)^T$ ↗ real value N data samples with their labels
 $\mathbf{t} \equiv (t_1, \dots, t_N)^T$ Label (target value)

- Model (a polynomial model):

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \dots + w_M x^M = \sum_{j=0}^M w_j x^j$$

↑
M-th degree polynomial

$$y(x_n, \mathbf{w}) = [1 \quad x_n \quad x_n^2 \quad \dots \quad x_n^M] \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix}$$

↑ Known constant values
minimize $\frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2$ Given sample \mathbf{x}_n . optimization targets

⇒ 여기서 주의해야 할 점은, $y(\mathbf{x}, \mathbf{w})$ 는 nonlinear function of \mathbf{x} 이지만, 우리가 신경써야 할 부분은 \mathbf{w} 이다.
즉, y 는

linear function of the coefficients \mathbf{w} 라는 점이다.

$$\text{minimize } \frac{1}{2} \|A\mathbf{w} - \mathbf{t}\|_2^2$$

위에는 \mathbf{x} 하나에 대해서 y 를 쓴 것이고, 실제로 주는 값들은 $x_1 \sim x_n$ 까지 굉장히 많다.

input들의 degree 0 ~ M까지를 하나의 Matrix로 만들면 $N \times M$ size의 matrix A 가 만들어진다고 볼 수 있다.

⇒ 이는 matrix의 norm을 최소화하는 문제와 동일하다

Linear Least-Squares Problem

위에서 본 것처럼 주어진 matrix A와 vector b에 대해서, vector x를 찾아야하는 optimization problem이 주어진다.
(w 대신 x를 사용하자 → coefficient를 구해야 하므로 w 대신 x를 쓰는게 가독성이 좋음)

$$A \in R^{m \times n}, b \in R^m, m \geq n$$

$$\underset{x \in R^n}{\text{minimize}} \frac{1}{2} \|Ax - b\|_2^2 = \frac{1}{2} (Ax - b)^T (Ax - b) = \frac{1}{2} (x^T A^T Ax - 2x^T A^T b + b^T b)$$

- Gradient: $\nabla f(x) = A^T Ax - A^T b$

- Solution when: $\nabla f(x) = 0$

$$\Rightarrow A^T Ax = A^T b \quad (\text{a.k.a. Normal Equation})$$

Recall: Matrix calculus

$$\frac{\partial x^T Ax}{\partial x} = \underline{(A + A^T)x}$$

$$\frac{\partial u^T Av}{\partial x} = \frac{\partial u}{\partial x} Av + \frac{\partial v}{\partial x} A^T u$$

⇒ 미분 하면 왼쪽 나옴

⇒ 최소값은 만드는 x를 찾기 위해 x에 대해 Gradient를 구하고, gradient를 0으로 만드는 x를 찾으면 된다.
왜냐하면 $f(x)$ 가 convex function이기 때문에 local minimum이 global minimum이다.

⇒ why convex function??

$$x^T A^T Ax = \|Ax\|_2^2 \geq 0$$

왼쪽의 식이 임의의 x에 대해 norm 형태를 이루고 있으므로, 항상 0이상 즉 positive semidefinite 조건을 만족한다.

⇒ 그러므로 $f(x)$ is a convex function

Solution via Pseudo-inverse

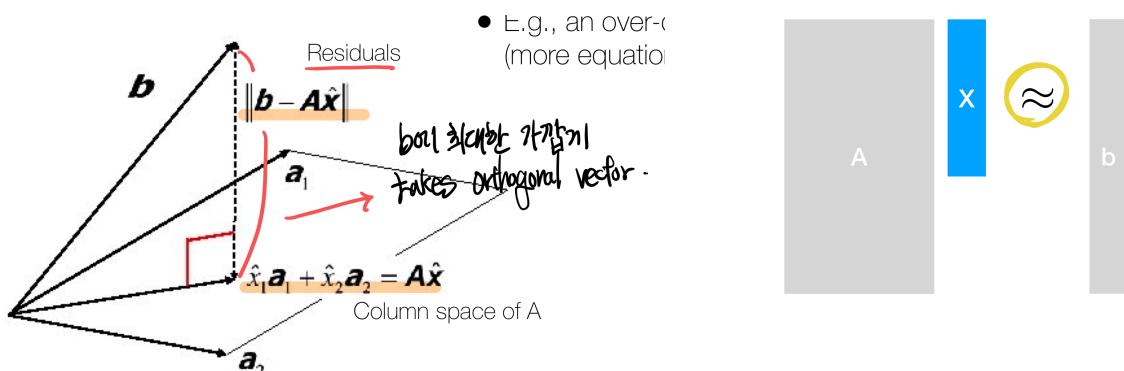
normal equation: $A^T A x = A^T b$

possible solution:

$$x = \frac{(A^T A)^{-1} A^T b}{A^\dagger : \text{Pseudo-inverse of } A}$$

only an inverse of $A^T A$

Geometric Interpretation



⇒ find the closet point in column space A from b, Ax에 orthogonal한 vecotr의 길이를 error vector로서 채택하면 된다.

Homogeneous Least-squares

- Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, find a vector $\mathbf{x} \in \mathbb{R}^n$ that solve the optimization problem

$$\mathbf{Ax} = 0 \quad (\text{or}) \quad \mathbf{Ax} = \mathbf{b}$$

- We assume that $m \geq n$
- To avoid the trivial solution $\mathbf{x} = 0$, we add the constraint $\|\mathbf{x}\| = 1$
- In the end, we solve the constrained least-squares:

$$\begin{aligned} & \text{minimize } f(\mathbf{x}) = \|\mathbf{Ax}\|_2^2 \quad \text{subject to } \|\mathbf{x}\| = 1 \\ & \Rightarrow \text{minimize } L(\mathbf{x}, \lambda) = \mathbf{x}^T \mathbf{A}^T \mathbf{A} \mathbf{x} + \lambda(1 - \mathbf{x}^T \mathbf{x}) \end{aligned}$$

$\mathbf{x}^T \mathbf{A}^T \mathbf{A} \mathbf{x}$ $\lambda(1 - \mathbf{x}^T \mathbf{x})$

Lagrange multiplier

⇒ solution: After $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$, the solution is the last column of \mathbf{V}