Mathematics for Machine Learning

Junhyeok Lee (wnsx0000@gmail.com)

May 7, 2026

목차

1. Intro

1.1. Intro

1.1.1. ML

ML(Machine Learning)은 데이터로부터 의미있는 정보를 자동으로 추출하는 general-purpose methodology을 디자인하는 것을 그 목적으로 한다. ML은 data, model, learning의 조합으로 이야기할 수 있고, 이를 잘 이해하고 새로운 solution을 제안하려면 수학적인 배경지식을 가지는 게 도움이 된다.

ML과 관련된 수학적 설명에서 용어의 정의가 문맥에 따라 차이가 존재할 수 있고, 본 요약본에서도 마찬가지이다.

- algorithm은 입력 데이터를 기반으로 예측하는 시스템을 의미할 수 있는데 여기에서는 이를 **Predictor**라고 부른다. 반면 predictor 내부의 parameter 값을 결정하는 시스템을 의미하기도 하는데 여기에서는 이른 **Training System**이라고 부른다.
- vector는 숫자들의 배열, 방향과 크기가 있는 화살표, 수학적 의미의 벡터 등으로 이해될 수 있다. 본 요약본에서는 데이터가 이미 컴퓨터가 읽을 수 있는 수치형 vector로 변환된 상태인 것으로 가정한다.
- model은 입력 데이터를 활용해 예측하는 프로세스를 단순하게 추상화한 것이고, 이런 model의 paramter를 최적화하는 것을 learning이라고 한다. 또한 **Utility Function**을 사용해 training data에 대해 얼마나 잘 예측하는지를 평가한다. 이때 learning은 training data뿐만 아니라, 새로운 데이터에 일반화가 가능하도록 하는 것을 그 목적으로 한다.

2. Vector & Matrix

linear algebra는 수치화된 데이터를 바라보는 가장 강력한 수학적 언어로, vector와 vector를 조작하기 위한 방법들을 다룬다.

2.1. Understanding of Multiplication

matrix-vector multiplication Ax=b, 그리고 matrix-matrix multiplication AB=C에 대해 더 이해해 보자. 참고로 vector라 하면 기본적으로 column vector를 의미한다.

2.1.1. Matrix Multiplication

Matrix Multiplication은 $c_{ij} = \sum_{l=1}^{n} a_{il} b_{lj}$로 계산된다. 추후 서술하는 것과 같이 matrix multiplication은 inner product라고도 한다.

matrix multiplication에 대해서는 associativity(결합 법칙)와 distributivity(분배 법칙)가 성립한다.

참고로 matrix multiplication할 때 분수가 나오면 scalar로 따로 빼놓고 계산해야 실수가 적다.

2.1.2. Linear Combination of Colums

연립일차방정식에서 주로 등장하는 Ax=b 꼴의 matrix-vector multiplication을 나타내는 방법은 아래와 같이 두 가지다.

1. Row-wise way(dot product)
Row-wise way는 Ax=b를 수식 그대로 coefficient matrix A와 variable vector x의 단순 곱(dot product)으로 보는 방식이다. 즉, A의 각 row가 방정식 하나의 coefficent를 나타내고, 각 방정식들에 대한 공통해를 찾는다는 관점이다. 수식으로 나타내면 당연하게도 아래와 같다.

기하적으로 생각해 보면 각 방정식에 대응되는 선 또는 면 등을 좌표계에 그리고, 좌표계에서 일치하는 부분이 x인 것으로 이해할 수 있다.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

2. Column-wise way
Column-wise way는 Ax=b를 A의 column들에 대한 linear combination(일차결합)으로 보는 방식이다(linear combination of columns of A). 수식으로 나타내면 아래와 같다. ML/DL 관점에서는 Ax=b를 이런 방식으로 나타내는 것이 intuition을 기르기에 좋고, ai 관련 논문이나 서적 등에서도 이 방식을 주로 사용한다.

기하적으로 생각해 보면 각 column vector들을 좌표계에 나타내고, 이것들을 적절히 scalar배 해서 b를 만드는 것으로 이해할 수 있다.

$$x_1 \begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \end{bmatrix} + x_2 \begin{bmatrix} a_{12} \\ a_{22} \\ a_{32} \end{bmatrix} + x_3 \begin{bmatrix} a_{13} \\ a_{23} \\ a_{33} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

column-wise way를 사용하면 Ax=b를 b에 대한 linear combination으로 이해할 수 있다는 장점도 있고, Ax=b의 차원이 달라지는 등의 변화가 생겼을 때에도 vector 하나만 추가해 주면 되므로 수식적인 이해와 수정이 용이하다는 장점도 있다. 또한 각 column vector들을 그리기만 하면 되므로 좌표계에 나타낼 때도 편리하다. 반면 row-wise way의 경우 각 방정식을 선이나 면 등의 형태로 좌표계에 나타내야 하는데, 특히 차원이 높아질수록 그리거나 이해하기 어렵다. 이에 따라 본 수업에서는 vector나 데이터셋 등을 나타낼 때 column-wise way를 사용한다.

column-wise way를 연립방정식 $Ax = b$에 적용해 보면, 임의의 vector b에 대해 Ax=b의 해가 존재하는지는 각 column vector들의 linear combination으로 b를 나타낼 수 있는지로 생각할 수 있다. 다시 말해, b가 임의의 n차원 vector일 때 column vector들의 부분집합이 해당 vector space의 basis라면 임의의 b에 대해서 해가 존재한다.

2.1.3. Linear Combination of Rows

앞에서 설명한 것처럼 Ax는 linear combination of columns of A이다. 동일한 방식으로 더 생각해 보면 $x^TA = b^T$는 linear combination of rows of A인 것으로 이해할 수 있다. 물론 이때 b는 $A^Tx = b$이므로 $Ax = b$에서의 b와는 다른 값이다.

$$\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix}$$

$$x_1 \begin{bmatrix} a_{12} & a_{12} & a_{13} \end{bmatrix} + x_1 \begin{bmatrix} a_{21} & a_{22} & a_{23} \end{bmatrix} + x_1 \begin{bmatrix} a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix}$$

물론 당연하게도 $x^TA = b^T$는 $A^Tx = b$와 같은데, 이는 단순히 A를 transpose한 것이므로 linear combination of rows of A로 이해할 수 있다.

즉, 정리하면 Ax꼴은 A의 column vector들의 linear combination으로, xA 꼴은 A의 row vector들의 linear combination으로 이해할 수 있다.

2.2. Matrix Multiplication

2.2.1. Matrix Multiplication

column vector, row vector에 대한 linear combination의 측면에서 matrix multiplication을 이해해보자. 우선 기본적으로 matrix A, B, C에 대한 matrix multipliaction AB=C는 A, B, C의 각 원소 a, b, c에 대해서 아래와 같이 나타낼 수 있다.

$$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$$

이때 C의 각 row와 column은 A와 B를 활용한 linear combination으로도 나타낼 수 있다. 당연하게도 matrix multiplication을 생각해 보면 C의 각 row는 대응되는 A의 row와 B에 대한 multiplication, C의 각 column는 A와 대응되는 B의 column에 대한 multiplication이다. 즉, A의 i번째 row를 a_i^r, B의 i번째 column을 b_i^c, C의 i번째 row 또는 column을 c_i^r, c_i^c이라 하면 아래의 수식이 성립한다.

$$Ab_i^c = c_i^c, \quad a_i^r B = c_i^r$$

다시 말해, AB=C에서 C의 각 column은 A의 column vector의 linear combination이고, C의 각 row는 B의 row vector의 linear combination이다.

Ax가 A column의 linear combination으로 나타낼 수 있다는 것은 Ax가 A의 column space에 포함된다는 것이고, 이를 수식으로 나타내면 $Ax \in C(A)$이다. 또한 multiplication은 결합법칙이 성립하므로 당연하게도 $ABCx \in C(A)$이다. 단순히 $BCx = x'$으로 묶는 것으로 증명된다.

2.2.2. Matrix Multiplication 다르게 나타내기

1. Column과 Row의 곱으로 나타내기
앞에서 언급한 것처럼 $n \times m$ matrix A와 $m \times k$ matrix B의 multiplication $AB = C$는 아래와 같이 나타낼 수 있다.

$$C_{ij} = \sum_{k=1}^{m} A_{ik} B_{kj}$$

이때 A의 i번째 column을 a_i^c, B의 i번째 row를 b_i^r라고 하면 아래와 같이 나타낼 수도 있다. 수식적으로는 이런 방식에서 특정 원소를 구하는 계산을 나타내보면 위의 수식과 같고, 직관적으로도 각 곱의 결과는 $n \times k$ 행렬이므로 이를 모두 더하면 AB와 같다.

$$AB = \sum_{k=1}^{m} a_k^c b_k^r$$

이런 식의 표기는 A의 각 열 또는 B의 각 행의 중요도에 따라 연산 결과에서 반영 비율을 바꿔보는 등의 접근를 가능하게 한다.

2. Submatrix의 곱으로 나타내기
matrix를 여러 submatrix 또는 블럭으로 나누고, 각 블럭을 성분으로 취급해 matrix multiplication을 계산할 수 있음.

$$AB = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{22}b_{12} + a_{22}b_{22} \end{bmatrix}$$

$$AB = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right] \cdot \left[\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}\right] = \left[\begin{array}{c|c} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ \hline A_{21}B_{11} + A_{22}B_{21} & A_{22}B_{12} + A_{22}B_{22} \end{array}\right]$$

2.3. Rank

2.3.1. Rank

1. Rank
matrix의 **Rank**는 해당 matrix의 column space의 dimension이다. 즉, linear independent한 column의 개수

이자, image의 dimension이다. $rk(A) = 2$와 같이 표기한다.

임의의 matrix A의 rank는 A^T의 rank와 같다. 즉, rank는 해당 matrix의 row space의 dimension이고, linear independent한 row의 개수로 구할 수도 있다. 정리하면 **rank는 linear independent한 row 또는 column의 개수로 계산할 수 있다.** 이때 하나의 matrix에 대해서 independent한 column의 개수와 independent한 row의 개수가 동일하므로, row와 column 중 더 짧은 쪽보다 항상 rank가 작다.

elementary operation은 rank를 보존한다. 이에 따라 임의의 matrix에 elemination을 적용해 pivot이 존재하는 column의 개수로 rank를 구할 수 있다.

2. Full Column/Row Rank
임의의 $m \times n$ matrix A에 대해 column/row와 rank가 같은 경우를 **Full Column/Row Rank**라고 한다.

1) Full Column Rank
모든 column들이 independent한 경우이다. 즉, pivot이 모든 column에 존재한다.

이 경우 $Ax = b$에서 pivot variable이 열의 개수와 같고, free variable이 없다. free variable이 없기도 하고, dimension theorm에 의해 N(A)는 영공간이다. $x_{\text{complete}} = x_{\text{particular}}$ 이다. b가 A의 linear combination인 것을 고려하면 $n < m$인 경우 항상 해가 존재하는 것은 아니고, $n = m$인 경우에는 invertible이므로 해가 존재한다.

2) Full Row Rank
모든 row들이 independent한 경우이다. 즉, pivot이 모든 row에 존재한다.

이 경우 $Ax = b$에서 모든 b에 대해 해가 존재한다. $n > m$인 경우 rank(column space의 dimension)가 m이고 column은 길이 m의 vector이므로 colum space는 R^m이다. 이에 따라 임의의 길이 m vector b를 column의 linear combination으로 만들 수 있으므로 항상 해가 존재한다. $n = m$인 경우 invertible이므로 해가 존재한다.

3) Full Rank
$m \times n$ matrix A가 가질 수 있는 가장 큰 rank를 가질 때, Full Rank라고 한다. 즉, full column rank 또는 full row rank인 경우를 말한다.

full rank가 아닌 matrix는 Rank Deficient라고 한다.

2.4. Invertibility

2.4.1. Invertibility

1. Invertibility
$n \times n$ 행렬 A에 대해서 $AB = BA = I$인 $n \times n$ 행렬 B를 A의 **Inverse Matrix(역행렬)**이라고 하고, A^{-1}로 표기한다. 또한 어떤 matrix가 inverse matrix가 존재하면 **Invertible(가역)**이라고 한다.

뒤에서도 다루겠지만, invertible인 matrix를 regular/invertible/nonsingular matrix라고 하고, non-invertible인 matrix를 singular/non-invertible matrix라고 한다.

$n \times n$ matrix의 invertibility는 아래와 같은 기준으로 판단이 가능하다.

1. rank가 n이면 invertible
2. determinant가 0이 아니면 invertible
3. Ax=0을 만족시키는 0이 아닌 벡터 x가 존재하면 non-invertible

$n \times n$ matrix A가 invertible하기 위한 필요충분조건은 A의 rank가 n인 것이다. 즉, non-singular matrix인 것과 invertible, rank가 n인 것은 동치 명제이다.

$n \times n$ matrix A가 invertible이면 rank가 n이고 nullity가 0이므로, systems of linear equation Ax=b는 유일한 해 $A^{`1}b$를 가짐. 또한 systems of linear equation의 해가 유일하면 nullity가 0인 것이므로 A는 invertible이다.

2. Inverse Matrix 구하기
invertible인 matrix A의 inverse matrix는 $(A|I)$에 elementary row operation을 적용해 구할 수 있다. A에 해당하는 부분을 항등행렬로 만들면 결과로 나오는 $(I|B)$에서 B가 A의 inverse matrix이다.

이때 elementary row operation을 적용하는 것은 아래와 같이 elementary matrix를 곱하는 것과 같으므로, $EA = I$, $E = B$이다. 즉, E 자체가 A의 inverse matrix이다.

$$E_n \cdots E_2 E_1 (A|I) = E(A|I) = (EA|E) = (I|B)$$

2×2 matrix A에 대해서, inverse matrix는 다음과 같고, $\det A \neq 0$이면 A는 invertible이다. 계산해 보면 당연하게 성립한다.

$$A^{-1} = \frac{1}{\det A} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & -a_{11} \end{bmatrix}$$

1×1 matrix A의 inverse matrix는 해당 원소의 값을 역수 취한 것이다.

3. Inverse Matrix의 성질
inverse matrix의 성질로는 아래와 같은 것들이 있다.

1. 정의에 따라 $AA^{-1} = A^{-1}A = I$가 성립한다.
2. $(AB)^{-1} = B^{-1}A^{-1}$이 성립한다.
3. $(A+B)^{-1} \neq A^{-1} + B^{-1}$

또한, invertible matrix A에 대한 inverse matrix는 유일하다.

2.4.2. Singular Matrix

1. Singular Matrix
column 또는 row가 independent하지 않은 matrix를 $n \times n$ **Singular Matrix(특이 행렬)**라고 하고, dependent한 matrix를 Non-singular Matrix라고 한다.

rank가 n이 아니면 singular, n이면 non-singular이다. 즉, singular matrix는 non-invertible이고, non-singular matrix는 invertible이다. invertible과 determinant가 0이 아닌 것은 필요충분이므로, determinant가 0이면 singular matrix, determinant가 0이 아니면 non-singular matrix이다.

정리하면, **singular인 것과 non-invertible인 것과 determinant가 0인 것은 동치 명제**이다.

2. Singular와 Non-singluar의 곱
$n \times n$ singular matrix A와 $n \times n$ non-singular matrix B의 곱 $AB = C$은 항상 singular matrix이다. 아래와 같이 여러 가지 방법으로 증명이 가능하다.

1) determinant는 행렬 곱을 보존하므로, 둘 중 하나의 determinant가 0이면 $\det(AB) = \det(A)\det(B) = 0$이다.

2) 행렬을 선형변환으로 생각해 보면, 둘 중에 하나라도 일대일대응이 아니면 AB도 일대일대응이 아니게 된다.

3) C의 각 column는 A의 column에 대한 linear combination이므로 independent할 수 없다. $rank(AB) = min(rank(A), rank(B))$인 것으로도 이해할 수 있다.

4) 귀류법으로도 접근이 가능하다. C가 non-singular라고 하면 차원정리에 의해 Cx=0인 0이 아닌 x가 존재하지 않는다. A가 singular이므로 $Ay = 0$인 0이 아닌 y가 존재하고, B는 non-singular이므로 $y = Bx$인 0이 아닌 x가 존재한다. $ABx = Ay = 0 = Cx$이므로 모순이다.

2.5. Transpose

2.5.1. Transpose

$n \times m$ matrix A에 대한 **Transpose(전치)**는 A의 row와 column을 뒤바꾸는 연산으로, A^T와 같이 표기한다. 즉, $(A^T)_{ij} = Aji$이다.

transpose는 아래와 같은 성질을 가진다.

1. $(A^T)^T = A$
2. $(A+B)^T = A^T + B^T$이다.
3. $(AB)^T = B^T A^T$이다. 당연하게도 $(ABC)^T = C^T B^T A^T$이다.
4. $(A^T)^{-1} = (A^{-1})^T$이다. $A^T(A^T)^{-1} = I$와, $AA^{-1} = I$를 transpose하고 비교해 증명할 수 있다. 이에 따라 A가 invertible이면 A^T도 invertible이다.

참고로, $rank(A) = rank(A^TA) = rank(AA^T)$이다. A와 A^TA는 null space가 같으므로 당연하다. $Ax = 0$이면 $A^TAx = 0$이고, $A^TAx = 0, x^TA^TAx = (Ax)^T(Ax) = 0$인데 dot product해서 영벡터인 것은 영벡터뿐이므로 $Ax = 0$이 성립한다. 또한 transpose해도 rank가 같으므로 이는 AA^T에 대해서도 성립한다.

transpose를 사용해 inner product와 outer product를 정의할 수 있다. $n \times 1$ vector x와 y에 대해 inner product는 $x \cdot y = x^Ty$로 정의되고(결과는 1×1), outer product는 xy^T로 정의된다(결과는 $n \times n$).

3. Vector Space

3.1. Vector Space

3.1.1. Group

Group은 다음과 같이 닫혀 있고, 결합법칙이 성립하며, 항등원과 역원이 존재하는 연산이 성립하는 집합으로, $G := (\mathcal{G}, \otimes)$으로 표기한다.

> **Definition 2.7** (Group). 집합 $\mathcal{G}$ 과 연산 $\otimes : \mathcal{G} \times \mathcal{G} \to \mathcal{G}$ 가 $\mathcal{G}$ 에 대해 정의되어 있을 때, 아래의 조건을 만족하는 $G := (\mathcal{G}, \otimes)$ 를 그룹이라고 부른다.
>
> 1. Closure of $\mathcal{G}$ on $\otimes : \forall x, y \in \mathcal{G} : x \otimes y \in \mathcal{G}$
> 2. Associativity: $\forall x, y, z \in \mathcal{G} : (x \otimes y) \otimes z = x \otimes (y \otimes z)$ (결합법칙)
> 3. Neutral element: $\exists e \in \mathcal{G}, \forall x \in \mathcal{G} : x \otimes e = x$ and $e \otimes x = x$ (항등원)
> 4. Inverse element: $\forall x \in \mathcal{G}, \exists y \in \mathcal{G} : x \otimes y = e$ and $y \otimes x = e$. (역원) 보통 x^{-1} 을 x의 inverse element(역원)이라고 표기합니다. 여기서 e 는 항등원을 나타냅니다.

이때 $\forall x, y \in \mathcal{G} : x \otimes y = y \otimes x$가 성립하는 G는 **Abelian Group**이라고 한다. 즉, commutative(교환법칙)가 성립하는 경우를 말한다.

예를 들어, invertible matrix의 집합은 matrix multiplication에 대해 group이지만, matrix multiplication은 commutative가 성립하지 않으므로 abelian group은 아니다.

3.1.2. Vector Space

1. Vector Space
Vector Space(벡터공간)는 scalar 곱과 vector 합에 대해 닫혀 있는 vector의 집합이다. 즉, 아래의 조건을 만족시키는 두 연산이 정의된 집합 V는 vector space이다.

1. 모든 $x \in V$, $y \in V$에 대해 $x + y \in V$이다.
2. 모든 scalar a와 모든 $x \in V$에 대해 $ax \in V$이다.

vector space는 vector 합에 대한 abelian group이고, 집합 외부의 체(field)에 속한 scalar를 활용한 곱이 정의된 집합이라고도 할 수 있다. 이때 vector 합을 inner operation, 스칼라 곱을 outer operation이라고도 한다. (inner/outer product와는 관련이 없다.)

정의에 의해 영벡터만 존재하는 집합도 vector space이고, 이를 **Zero Vector Space(영벡터공간, 영공간)**라고 한다. zero vector space의 dimension은 0이다.

R^n은 n개의 실수 값을 속성으로 하는 vector들의 대한 vector space이고, linear algebra의 algorithm은 주로 R^n 위에서 공식화된다.

2. Subspace
어떤 vector space에 대한 **Subspace(부분공간)**은 해당 vector space에서 정의하는 scalar 곱과 vector space에 대해 닫혀 있는 부분집합이다. 즉, 아래의 조건을 만족시키는 vector space의 부분집합 W는 해당 vector

space의 subspace이다.

1. 모든 $x \in W$, $y \in W$에 대해 $x + y \in W$이다.
2. 모든 scalar a와 모든 $x \in W$에 대해 $ax \in W$이다.
3. W는 V와 동일한 0 vector를 포함한다.

matrix A의 column을 span해 생성된 vector space는 **Column Space(열공간)**라고 하고, $C(A)$로 표기한다. 마찬가지로 row를 span해 생성된 vector space는 **Row Space(행공간)**라고 하고, $R(A) = C(A^T)$로 표기한다.

정의에 의해 0 vector만 존재하는 vector space는 모든 vector space의 subspace이다.

vector space에 대한 엄밀한 정의가 존재하기는 하지만, 여기에서는 이 정도로만 정리한다.

geometric vectors, polynomials, audio signals, tuples 등은 모두 vector space로 정의할 수 있고, 각각은 vector이다.

matrix A에 대해 $Ax = b_1, Ax = b_2$는 해가 존재하고 $Ax = b_3$는 해가 존재하지 않으려면, 당연하게도 b_1과 b_2은 A의 column space에 존재하고 b_3는 존재하지 않으면 된다. 가장 쉬운 예시는 A의 각 열이 b_1, b_2인 것이고, b_3는 b_1과 b_2의 linear combination으로 만들 수 없는 상황이다.

참고로 P와 L이 vector space V의 subspace일 때, 당연하게도 '$P \cup L$은 subspace이다'는 거짓이고, '$P \cap L$은 subspace이다'는 참이다. vector 몇 개로 증명이 가능하다.

matrix들에 대한 vector space는 주로 M으로 표기한다. upper triangle matrix에 대한 vector space P, symmetric matrix에 대한 vector space S, $P \cap S$인 diagonal matrix에 대한 vector space 모두 M의 subspace이다.

3.1.3. Column Space and Null Space

1. Column Space
$m \times n$ matrix A의 **Column Space**(C(A))는 $Ay = x$를 만족시키는 vector x의 집합인 vector space이다. 즉, A의 column의 linear combination으로 생성되는 vector space로, R^n의 subspace이다. column space는 image, range라고도 한다.

column space가 subspace임은 vector 합과 scalar 곱에 대해 닫혀 있는 것으로 쉽게 증명할 수 있다.

당연하게도 Ax=b가 해를 가지려면 b는 A의 column space에 존재해야 한다.

A의 column space의 dimension은 A의 rank와 같다. 즉, $dim(C(A)) = rank(A)$이다.

2. Null Space
$m \times n$ matrix A의 **Null Space(N(A))**는 $Ax = 0$을 만족시키는 vector x의 집합인 vector space이다. 즉, $Ax = 0$을 풀어 null space를 구할 수 있고, A의 null space는 R^m의 subspace이다. kernel이라고도 한다.

null space가 subspace임은 vector 합과 scalar 곱에 대해 닫혀 있는 것으로 쉽게 증명할 수 있다. 이때 x는 $n \times 1$ vector이므로 null space는 R^n의 subspace이다.

Ax=0을 만족하는 x($x \neq 0$)가 존재하면 A는 non-invertible이다. 즉, null space가 zero vector space가 아니면 A는 non-invertible이다. 이는 A column의 linear combination으로 0이 만들어질 수 있다는 것이고, linear independent하지 않으므로 당연하다.

$m \times n$ matrix A가 가지는 null space의 dimension은 A의 n-rank와 같다. 즉, $dim(N(A)) = n - rank(A)$이다.

참고로, 'Ax=b를 만족시키는 x의 집합이 항상 subspace이다.'는 거짓이다. 뒤에서 살펴볼 것처럼 subspace인 경우도 있고, 그렇지 않은 경우도 있다.

3.2. Basis&Dimension

3.2.1. Linaer Independence

어떤 vector들의 집합 $\{v_1, v_2, \cdots, v_n\}$에 대해서 다음 수식을 만족시키는 0이 아닌 scalar $c_1, c_2, \cdots, c_n$이 존재하지 않으면 이 집합은 **Linear Independent**하다고 한다. 즉, 해당 vector들의 linear combination으로 영벡터를 만들 수 없으면 linear independent하다.

$$c_1v_1 + c_2v_2 + \cdots + c_nv_n = 0$$

영벡터를 만들 수 있다는 것은 해당 집합 내에서 일부 vector들의 linear combination으로 다른 vector를 만들 수 있다는 것이다. 다시 말해, linear independent하다는 것은 해당 집합 내에 불필요한 vector가 존재하지 않는다는 것으로 이해할 수 있다.

vector $v_1, \cdots, v_n$으로 column을 구성한 $m \times n$ matrix A를 생각하자. 해당 vector 집합이 linear indepedent한 것(rank가 n)과 N(A)가 zero vector space인 것(nullity가 0)은 필요충분이다. 이에 따라 A에 대해 elimination을 적용해 rank를 계산하여 independent 여부를 확인할 수 있다.

3.2.2. Span

어떤 vector들의 집합 $\{v_1, v_2, \cdots, v_n\}$에 대해서 해당 vector들의 linear combination으로 vector space S를 생성할 수 있다. 이때 해당 vector들의 집합을 span하여 S를 생성했다고 하고, 이때 생성에 사용된 vector 집합을 generating set, 생성된 vector space를 **Span(생성공간)**이라고 한다.

어떤 vector 집합 β를 span했을 때 vector space V가 생성되는지는, 그 vector space에 속하는 임의의 벡터 v를 β의 linear combination으로 나타낼 수 있음을 보이면 된다. 이때 elimination을 적용해 RREF로 나타내면 더 단순할 수 있다.

어떤 vector 집합을 span해서 생성한 집합은 항상 vector space이다.

예를 들어, matrix A column들의 집합을 span하면 A의 column space가 만들어진다.

3.2.3. Basis&Dimension

1. Basis
어떤 vector들의 집합 $\beta = \{v_1, v_2, \cdots, v_n\}$이 1) linear indepedent이면서 2) span했을 때 vector space V를 생성한다면, 이때 β는 V의 **Basis(기저)**라고 한다. 즉, minimal generating set이자, maximal linear independent set이다. basis 중 vector space에 대해 표준적으로 존재하는 것을 **Standard Basis** 또는 Canonical Basis라고 한다.

이 조건을 만족시키는 vector 집합은 여러 개일 수 있으므로 당연하게도 어떤 vector space에 대한 basis는 유일하지 않은데, 동일한 vector space에 대해서라면 모든 basis는 그 원소의 개수가 전부 같다.

basis는 generating set을 찾은 다음, 해당 set의 vector를 column으로 하는 matrix A를 구성한 뒤, A에 대한 RREF를 구하면 pivot column 위치에 있던 vector들이 basis가 된다.

2. Dimension
vector space V의 basis가 n개의 vector로 이루어질 때 n을 V의 **Dimension(차원)**이라고 한다.

U가 vector space V의 subspace이면 $\dim U \leq \dim V$이다.

3.2.4. Fundamental Theorem of Linear Mappings

Fundamental Theorem of Linear Mappings 또는 Rank-Nullity theorem 또는 Dimension theorem은 다음의 수식이 성립함을 의미한다.

$$\dim(V) = \dim(\ker(f)) + \dim(\mathrm{Im}(f)) = \text{nullity} + \text{rank}$$

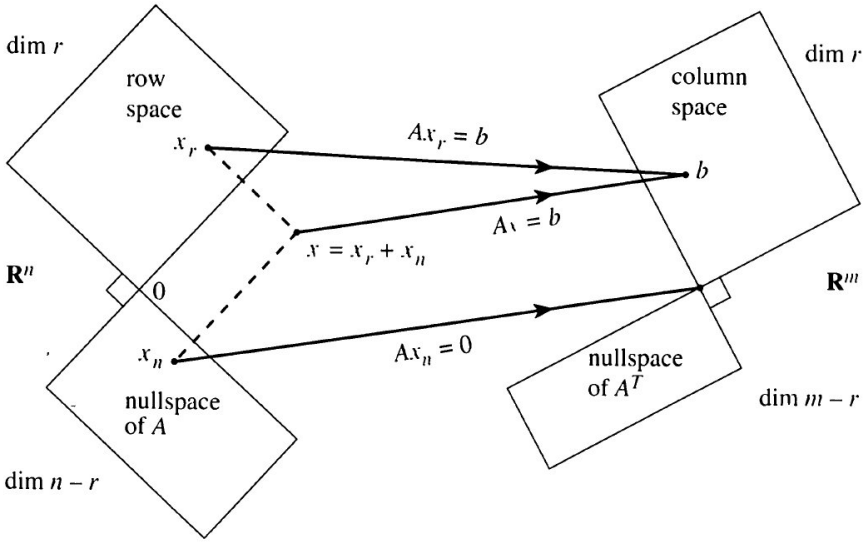
이때 V는 domain(정의역)이다. rank가 $\dim(V)$보다 작으면 null space에 영벡터가 아닌 원소가 존재하며, 이 경우 $Ax = 0$은 무한히 많은 해를 가진다.

3.3. Four Fundamental Spaces

3.3.1. Four Fundamental Spaces

rank가 r인 $m \times n$ matrix A는 다음 그림과 같이 4가지 fundamental subspace로 이해할 수 있다. orthogonality에 대한 내용은 뒤에서 따로 다룬다.

즉, Ax=b에서 x는 항상 A의 row space와 null space에 속한 원소의 합으로 나타낼 수 있고, b는 column space와 left null space에 속한 원소의 합으로 나타낼 수 있다.



1. Column Space ($C(A)$)
Column Space는 앞에서 다룬 것처럼 길이 m vector인 column들의 linear combination으로 생성되는 subspace이다. 이때 길이 m인 vector들이므로 R^m의 subspace이다.

column space는 pivot variable에 대응되는 위치에 있는 column(RREF의 pivot column과는 다름.)들의 집합을 basis로 가진다. dimension은 r이다.

2. Null Space ($N(A)$)
Null Space는 앞에서 다룬 것처럼 Ax=0을 만족시키는 길이 n vector들로 구성된 subspace로, 이에 따라 R^n의 subspace이다.

null space는 special solution들의 집합을 basis로 가진다. dimension은 n-r이다.

3. Row Space ($C(A^T)$)
Row Space는 길이 n vector인 row들의 linear combination으로 생성되는 subspace이다. 이때 길이 n인 vector들이므로 R^n의 subspace이다.

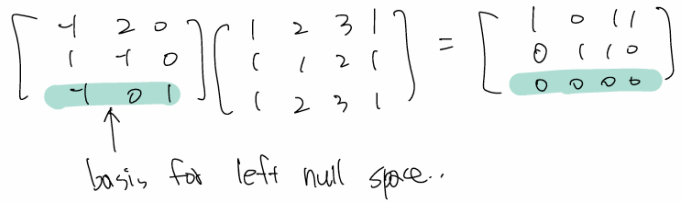
row space는 A에 대한 elimination을 통해 얻은 R에서 전부 0이 아닌 row의 집합을 basis로 가진다. dimension은 r이다(column space와 같다.).

4. Null Space of A^T ($N(A^T)$)
A^T의 Null Space 또는 Left Null Space는 $A^T x = 0$을 만족시키는 길이 m vector들로 구성된 subspace로, 이에 따라 R^m의 subspace이다. A^T의 null space는 $A^T y = 0$, $y^T A = 0$을 만족시키는 y의 집합이므로 left null space라고 부른다.

left null space의 basis는 다음과 같은 방법으로 구할 수 있다. dimension은 m-r이다.

- A^T null space의 basis를 직접 구한다.
- $[A|I]$에 elimination을 적용해 $[R|E]$로 변형한 뒤(EA=R 성립), R에서 모든 성분이 0인 row와 동일한 위치에 있는 E의 row들의 집합을 구성하면 그게 basis이다.

  이때 E는 invertible이므로 각 row는 independent하고, 전부 0인 row는 m-r개이므로 이렇게 뽑은 row들은 basis가 된다.



4. Systems of Linear Equation

linear algebra의 많은 문제들은 연립일차방정식(Systems of linear equations)의 형태로 나타낼 수 있다. 그리고 일반적으로 systems of linear equations은 matrix A와 vector x, b에 대해 Ax=b 꼴로 나타낼 수 있다.

systems of linear equations이 가지는 solution(해)은 없거나, 오직 하나이거나, 무수히 많다. 또한 이는 기하학적으로 이해할 수 있는데, 각 equation을 R^n에서 나타냈을 때 서로 만나지 않거나, 한 점에서 만나거나, 무수히 많은 점에서 만나게 된다.

4.1. Elimination

4.1.1. Elimination

1. Elimination
Elimination(소거법)은 선형대수학에서 systems of linear equation의 해를 구하는 가장 기본적인 방법이면서, 행렬의 다양한 속성을 이해하는 데에 활용되는 기법이다.

Gaussian Elimination(가우스 소거법)은 matrix에 elementary operation을 적용해 Row Echelon Form(행사다리꼴)으로 만드는 elimination이고, **Gauss-Jordan Elimination(가우스-조던 소거법)**은 동일하게 elementary operation을 적용해 RREF(Reduced Row Echelon Form, 기약행사다리꼴)로 만드는 elimination이다. 이렇게 elementary operation을 적용하는 것을 Elementary Transform이라고 한다.

참고로, 본 교재에서는 gaussian elimination을 matrix A를 RREF로 만드는 elimination이라고 정의한다.

row echelon form은 아래의 조건 중 1, 2번 조건만 만족시키는 matrix이고, RREF는 아래의 조건을 모두 만족시키는 matrix이다. 이때 **Pivot(피벗)**은 각 row에서 처음으로 0이 아닌 원소이다. 당연하게도 matrix A를 row echelon form으로 변형했을 때 pivot의 개수는 A의 rank와 같다.

1. 0이 아닌 성분을 가지는 row는 모든 성분이 0인 행보다 위쪽에 위치한다.
2. 0이 아닌 성분을 가지는 row의 pivot은 위쪽 row의 pivot보다 오른쪽에 위치한다.
3. pivot이 존재하는 column에서 pivot을 제외한 다른 모든 원소의 값은 0이다. (pivot의 위아래가 모두 0이다.)
4. 모든 privot의 값은 1이다.

2. Elementary Operation and Elementary Matrix
matrix의 row[column]에 대한 아래의 세 연산을 **Elementary Row[Column] Operation(기본행[열]연산)**이라 한다. 또한 row와 column에 대한 이 연산들을 통틀어 Elementary Operation이라 한다.

1. A의 두 row[column]을 교환하는 것.
2. A의 한 row[column]에 0이 아닌 scalar를 곱하는 것.
3. A의 한 row[column]에 다른 row[column]의 scalar 배를 더하는 것.

Elementary Matrix(기본행렬)은 identity matrix I_n에 elementary operation을 한 번 적용하여 얻은 matrix이다. elementary operation을 적용하는 것은 그 matrix에 대응되는 elementary matrix(동일한 elementary operation을 I_n에 적용한 것)를 곱하는 것과 같고, 이에 따라 elementary operation을 적용하는 것을 수식적으로 나타낼 수 있다.

이때 elementary row operation을 적용하는 것은 elementary matrix를 왼쪽에 곱하는 것과 같고, elementary column operation을 적용하는 것은 elementary matrix를 오른쪽에 곱하는 것과 같다.

세 번째 elementary operation에 대한 matrix를 Elemination Marix(소거행렬)로, 첫 번째 elementary operation에 대한 matrix를 Permutation Matrix(치환행렬)로 부르기도 한다.

elementary row operation은 row space를 보존하지만, column space는 보존하지 않는다. 또한 이는 elementary matrix를 왼쪽에 곱하는 것과 같으므로 null space를 보존한다. 반면 elementary column operation은 column space는 보존하지만 row space 및 null space는 보존하지 않는다. 물론 두 operation 모두 rank는 보존한다.

참고로, 본 교재에서는 3번 elementary operation을 'scalar 배를 더하는 것' 대신 단순히 '더하는 것'으로 정의한다.

3. Systems of Linear Equation의 풀이
Ax=b 꼴의 systems of linear equation은 Augmented Matrix(첨가행렬) $(A|b)$에 대해 elimination을 적용해 RREF로 변형하는 것으로 풀 수 있다.

1. matrix의 각 row에 대해 위에서 아래로 내려가면서 elementary row operation을 적용해 pivot의 값이 1이고, pivot과 동일한 column의 성분들 중 아래쪽에 있는 것들을 0으로 만든다.
2. 이후 아래에서 위로 올라가면서 privot과 동일한 column의 성분들 중 위쪽에 있는 것을 0으로 만든다.

4. Systems of Linear Equation의 해의 존재 여부 판단
systems of linear equation $Ax = b$의 해가 임의의 b에 대해 존재하는지는 아래의 방법 중 하나로 판단할 수 있다.

1. A가 invertible이면 해가 존재한다. 또한 이 경우 $x = A^{-1}b$으로 해를 계산할 수 있다.
2. $rank(A) = rank(A|b)$이면 해가 존재한다. 이 경우 A column의 linear combination으로 b를 만들 수 있다.
3. $(A|b)$를 RREF로 나타냈을 때 모순인 수식이 나오는 행이 있으면 해가 존재하지 않는다.

systems of linear equation을 풀 때, elimination 대신 직관적으로 column의 linear combination으로 나타낼 수 있는지 확인할 수도 있는데, 이런 직관을 가지면 좋다고 한다.

Ax=b에 대한 solution은 gaussian elimination을 사용하거나, A가 invertible할 때 $x = A^{-1}b$를 계산하거나, $x = (A^TA)^{-1}A^Tb$임을 이용해 구할 수 있지만 이 방법들 모두 수백만 개의 변수가 존재하는 실제 상황에서는 적용하기에 cost가 너무 크다. 그래서 실제로는 Richardson method, Jacobi method, Gauss-Seidel method, Successive over-relaxation method와 같은 Stationary Iterative methods, 또는 Conjugate gradients, Generalized minimal residual, Biconjugate gradient와 같은 Krylov subspace methods를 통해 간접적으로 해를 구한다고 한다.

4.2. General Solution

4.2.1. Special/Particular Solution of Ax=b

Ax=b의 general solution을 찾아보자.

1. Pivot Variable vs. Free Variable
elimination을 적용했을 때 row echelon form에서 pivot이 존재하는 column을 **Pivot Column**, pivot이 존재하지 않는 column을 **Free Column**이라고 한다. 당연하게도 pivot column과 동일한 위치의 column들의 집합은 column space의 basis이다.

참고로, 본 교재에서는 pivot variable을 Basic Variable이라고 명명한다.

이에 따라 Ax=b에서 x에 해당하는 각 원소를 다음과 같이 구분할 수 있다.

- pivot column에 대응되는 x의 원소는 **Pivot Variable**이라고 한다.

  matrix A의 rank는 row echelon form으로 변형했을 때 pivot의 개수와 같으므로, pivot variable의 개수는 rank와 같다.
- free column에 대응되는 x의 원소는 **Free Variable**이라고 한다. 이때 free variable은 뒤에서 다루는

것처럼 해를 구할 때 어떤 값이든 될 수 있기 때문에 free이다.

뒤에서 다룰 것처럼, matrix A의 nullity는 그 basis인 special solution의 개수와 같고, speical solution의 개수는 free variable의 개수와 같으므로, free variable의 개수는 nullity와 같다.

pivot variable의 개수와 free variable의 개수를 dimension theorm으로 이해할 수 있다.

2. Special/Particular Solution
연립일차방정식 Ax=b에 대해서 Ax=b를 만족시키는 특정한 해를 **Particular Solution**, Ax=0을 만족시키는 특별한 해를 **Special Solution**이라고 한다. 이때 linear independent한 special solution은 free variable 만큼 존재하며, 그 집합은 A null space의 basis가 된다.

참고로, 여기에서의 정의가 더 일반적인 것처럼 보이지만, 본 교재에서는 particular solution을 'particular 또는 speical solution'으로 정의하고, speical solution은 따로 정의하지 않는다.

elimination의 elementary operation들은 null space를 바꾸지 않는다(참고로, column space는 바꾼다.). 즉, special solution은 elimination을 적용한 뒤에도 구할 수 있다. 이에 따라 연립방정식 Ax=0에 대한 **special solution은 다음과 같은 과정을 거쳐 구할 수 있다.**

1. A에 elimination을 적용해 RREF 또는 row echelon form인 matrix R로 변형한다.
2. $Rx = 0$에서 free variable들 중 하나에만 1, 나머지는 0을 대입해 각 경우에 대한 x를 구하면 그게 special solution이다. free variable이 하나인 경우 1을 대입하면 된다. pivot column의 linear combination으로 free column을 만들 수 있으므로, free variable에 임의의 값을 넣어도 Rx=0을 만족시키는 x를 항상 찾을 수 있다.

   이에 따라 free variable 개수만큼의 speical solution이 존재하게 된다.

   이때 RREF로 변형했으면 1을 대입하는 free variable에 대응되는 free column의 원소에서 부호만 바꾸면 special solution이므로 더 간단하다.

이렇게 구한 special solution들은 free variable 위치에 대해 하나만 1이고 나머지는 0이므로, 서로 linear indepedent이다. 또한 $Ax' = 0$을 만족시키는 임의의 x'은 special solution의 linear combination으로 나타낼 수 있으므로 special solution 집합의 span은 null space이다. 이는 pivot variable의 값은 free variable의 값들에 의해 유일하게 결정되므로, x'의 free variable 값과 같도록 special solution들의 linear combination을 구성하면 pivot variable의 값 또한 유일하게 결정되기 때문이다. 즉, **special solution의 집합은 null space의 basis**이다.

$$\boldsymbol{A}=\begin{bmatrix} p & p & f & p & f \\ | & | & | & | & | \end{bmatrix} \quad \boldsymbol{R}=\begin{bmatrix} \mathbf{1} & 0 & a & 0 & c \\ 0 & \mathbf{1} & b & 0 & d \\ 0 & 0 & 0 & \mathbf{1} & e \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \boldsymbol{s}_1=\begin{bmatrix} -a \\ -b \\ \mathbf{1} \\ 0 \\ \mathbf{0} \end{bmatrix} \quad \boldsymbol{s}_2=\begin{bmatrix} -c \\ -d \\ \mathbf{0} \\ -e \\ \mathbf{1} \end{bmatrix}$$

Ax=b에 대한 **particular solution은 $Ax = b$ 또는 $Rx = b'$에서 free variable 전부에 0을 대입하고 연립방정식을 풀어서 구할 수 있다.** pivot variable의 linear combination으로 모든 free column을 만들 수 있으므로, Ax=b의 해가 존재한다면 모든 free variable에 0을 대입했을 때 particular solution을 찾을 수 있다. 특히 RREF로 elimination을 했다면 pivot variable 값이 b의 값과 같아지므로 매우 단순하다.

3. Rank와의 관계
$m \times n$ matrix A에 대해 rank가 r일 때, n, m, r의 대소관계에 따라 Ax=b의 해가 어떻게 존재하는지를 살펴보자. 이는 앞서 살펴본 내용에 의하면 당연하다.

- r=m=n인 경우, Ax=b는 항상 유일한 해를 가진다.
- r=n<m인 경우(full column rank), Ax=b는 해가 없거나, 유일한 해를 가진다.
- n>m=r인 경우(full row rank), Ax=b는 항상 무한한 해를 가진다.
- r<m, r<n인 경우, Ax=b는 해가 없거나, 무한한 해를 가진다.

이는 four fundamental space로도 이해할 수 있다. A의 column space에 b가 존재하지 않으면 해가 존재하지 않고, 해가 존재할 때 A의 null space의 dimension이 0이 아니면 해가 무수히 존재한다.

본 수업에서는 Ax=0에서 0에 기본행렬을 곱해도 0이므로 elimination을 적용해도 null space가 바뀌지 않는다고

하는데, 각 elementary operation에 대해 null space가 바뀌지 않는다는 것으로 증명할 수도 있다.

4.2.2. General Solution of Ax=b

1. General Solution
Ax=b에 대한 **General(Complete) Solution**(x_c)은 Ax=b를 만족시키는 모든 해의 집합으로, 다음과 같이 하나의 particular solution(x_p)와 special solution($s_1, s_2 \cdots$)의 linear combination(x_n. null space의 임의의 vector)으로 나타낼 수 있다.

$$x_{\mathrm{c}} = x_{\mathrm{p}} + x_{\mathrm{n}} = x_{\mathrm{p}} + (c_1 s_1 + c_2 s_2 + \cdots)$$

이는 particular solution x_{p}에 대해 $Ax_{\mathrm{p}} = b$가 성립하고, special solution의 linear combination x_n에 대해 $Ax_n = 0$이 성립하므로, 이 둘을 연립해 $A(x_p + x_n) = b$로 정리할 수 있기 때문이다.

영벡터가 아닌 b에 대해 Ax=b의 general solution은 영벡터를 포함하지 않으므로 항상 subspace가 아니다. 영벡터가 포함된다면 A0=b이므로 모순이다.

2. General Solution 구하기
Ax=b에 대한 general solution은 앞아서 다룬 것처럼 다음과 같이 구할 수 있다. 물론 해가 없는 경우에는 구할 수 없다.

1. Ax=b에 대한 augmented matrix [A|b]에 elimination을 적용해 RREF 또는 row echelon form R로 변형한다.
2. free variable들 중 하나에만 1, 나머지는 0을 대입해 $Rx = 0$에서 각 경우에 대해 special solution을 구한다.
3. 모든 free variable에 0을 대입해 $Rx = b'$에서 particular solution을 구한다.
4. special solution의 linear combination과 particular solution을 더해 general solution을 얻는다.

또는 '프리드버그 선형대수학'에서 다뤘던 것처럼 다음과 같은 방법으로도 구할 수 있다.

1. Ax=b에 대한 augmented matrix [A|b]에 elimination을 적용해 RREF 또는 row echelon form으로 변형한다.
2. 각 pivot variable에 매개변수를 부여한다.
3. 각 free variable들을 해당 매개변수로 나타낸다.
4. 매개변수에 대해 정리해 general solution을 얻는다.

   결과로 도출된 general solution에서 매개변수가 곱해져 있지 않은 vector는 particular solution이고, 매개변수가 곱해져 있는 vector는 special solution이다.

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} -2t_1 + 2t_2 + 3 \\ t_1 - t_2 + 1 \\ t_1 \\ 2t_2 + 2 \\ t_2 \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \\ 0 \\ 2 \\ 0 \end{pmatrix} + t_1 \begin{pmatrix} -2 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} + t_2 \begin{pmatrix} 2 \\ -1 \\ 0 \\ 2 \\ 1 \end{pmatrix}$$

즉, general solution을 구하는 과정은 homogeneous(동차) equation Ax=0의 해를 찾고, nonhomogeneous(비동차) equation Ax=b의 특정 해 하나를 더하는 것과 같다.

4.2.3. Minus 1 Trick

Minus 1 Trick은 matrix A에 대한 RREF R에 한 원소만 -1인 row를 끼워 넣어 general solution을 구하는 trick이다. 특히 $Ax = 0$의 해를 빠르게 구하기 편리하다.

이때 row echelon form이면 안되고, RREF여야 한다. 하나의 free variable에 1, 나머지 free variable에 0을 넣어 각 경우에 대한 pivot variable의 값을 구하는 계산을 해 보면 당연하게 성립하는 것을 알 수 있다.

Example 2.8 (Minus-1 Trick)

식 (2.49)의 행렬을 다시 살펴보도록 하겠습니다.

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{1} & 3 & 0 & 0 & 3 \\ 0 & 0 & \mathbf{1} & 0 & 9 \\ 0 & 0 & 0 & \mathbf{1} & -4 \end{bmatrix} \tag{2.53}$$

위 행렬에 (2.52) 형태의 행을 추가하여 아래와 같은 5×5 행렬로 변환합니다.

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} \mathbf{1} & 3 & 0 & 0 & 3 \\ \mathbf{0} & \mathbf{-1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 0 & 0 & \mathbf{1} & 0 & 9 \\ 0 & 0 & 0 & \mathbf{1} & -4 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{-1} \end{bmatrix} \tag{2.54}$$

위 행렬로부터 대각 요소에 -1을 포함하는 열을 가지고, 바로 $\boldsymbol{Ax} = \mathbf{0}$ 의 해를 얻을 수 있습니다.

$$\left\{ \boldsymbol{x} \in \mathbb{R}^5 : \boldsymbol{x} = \lambda_1 \begin{bmatrix} 3 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \lambda_2 \begin{bmatrix} 3 \\ 0 \\ 9 \\ -4 \\ -1 \end{bmatrix}, \quad \lambda_1, \lambda_2 \in \mathbb{R} \right\} \tag{2.55}$$

5. Linear Transformation

5.1. Linear Transformation

5.1.1. Linear Transformation

1. Linear Transformation

Linear Transformation(선형변환)은 다음과 같은 조건 2가지를 만족시키는 함수이다. 즉, vector 합과 scalar 곱(선형성)을 보존하는 함수이다. Linear Mapping 또는 Vector Space Homomorphism이라고도 한다.

1. $T(v+w) = T(v) + T(W)$
2. $T(cv) = cT(v)$

linear transformation은 두 가지 관점에서 생각해볼 수 있다. 하나는 space(좌표계. 정의역, 치역)를 고려하지 않고, Ax와 같이 단순히 값을 보내는 것이다. 또 다른 하나는 space를 고려해서, 해당 space의 basis를 가져와서 표현하는 것이다. 두 번째 관점에 의해서는 어떤 linear transformation이 어떤 함수인지를 알려면 정의역의 basis를 보내보면 된다.

기하적 관점에서 linear transformation은 vector를 다른 basis에 대한 좌표계로 변환(또는 확장)한다고도 이해할 수 있는데, 이런 관점에서 보면 linear transformation과 뒤쪽에서 설명할 내용들을 더 잘 이해할 수 있다. 링크를 참고하자.

2. Linear Transformation의 Image와 Null Space

linear transformation의 image와 null spcae는 matrix에서와 동일하게, $f : V \to W$에 대해 image는 $\{w \in W \mid \exists v : f(v) = w\}$ null space는 $\{v \in V : f(v) = 0\}$로 정의된다. 이때 V를 domain, W를 codomain이라고 한다.

또한 다음과 같은 성질이 성립한다.

- null space는 공집합일 수 없다.
- image는 W의 subspace, null space는 V의 subspace이다.
- null space가 {0}인 것과, f가 injective인 것은 필요충분이다.

  finite vector space V, W와 linear transformation $f : V \to W$를 생각하자. $\dim(V) = \dim(W)$이고 nullity가 0이면 f는 bijective이다.

예를 들어, projection matrix를 곱하는 것은 linear transformation이다.

5.1.2. Speicial Mapping

1. Special Mapping
임의의 집합 V, W에 대해 mapping $\Phi : V \to W$를 생각하자. 다음과 같이 정의한다.

- **Injective(단사)**. $\forall x, y \in V, \Phi(x) = \Phi(y) \to x = y$
- **Surjective(전사)**. $\Phi(V) = W$
- **Bijective(전단사)**. injective and surjective.

이 정의를 사용해 다음과 같이 linear mapping의 speical case들을 정의할 수 있다.

- **Isomorphism(동형사상)**. $\Phi : V \to W$, linear and bijective.

  Φ가 isomorphism이면 Φ^{-1}도 isomorphism이다.
- Endomorphism. $\Phi : V \to V$, linear.
- Automorphism. $\Phi : V \to V$, linear and bijective.
- V에 대한 Identity Mapping 또는 Identity Automorphism(id_V). $\Phi : V \to V$, $x \to x$

2. Isomorphic
finite vector space V, W에 대해 $\dim(V) = \dim(W)$이면 V와 W는 **Isomorphic(동형)**하다고 한다. 이는 두 vector space 간에 isomorphism이 존재해 서로 손실없이 변환될 수 있음을 의미한다.

isomorphic에 의해 임의의 두 vector space V,W에 대한 mapping을 R^n, R^m에 대한 것으로 취급할 수 있도록 한다.

bijective이면 해당 mapping에 대한 역도 성립한다.

iso는 same, endo는 in, auto는 self를 의미하는 접두어이다.

5.1.3. Ordered Basis

basis의 순서를 정렬해 n-tuple로 나타낸 것을 **Ordered Basis(순서기저)**라고 한다.

vector space의 임의의 vector v는 ordered basis $B = (b_1, \cdots, b_n)$에 대해 $x = \alpha_1 b_1 +, \cdots, +\alpha_n b_n$으로 나타낼 수 있다. 이때의 $\alpha_1, \cdots, \alpha_n$은 B에 대한 v의 **Coordinate**라고 하고, 다음을 B에 대한 v의 **Coordinate Vector** 또는 Coordinate Representation이라고 한다.

$$\alpha = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} \in R^n$$

5.1.4. Matrix Representations of Linear Mappings

vector space V, W와 그에 대응되는 ordered basis $B = (b_1, \cdots, b_n)$, $C = (c_1, \cdots, c_m)$를 생각하자. linear mapping $\Phi : V \to W$에 대한 유일한 matrix representation은 다음과 같이 정의된다. 이때 $j \in \{1, \cdots, n\}$이다.

$$\Phi(b_j) = \alpha_{1j} c_1 + \cdots + \alpha_{mj} c_m = \sum_{i=1}^{m} \alpha_{ij} c_i$$

α_{ij}를 원소로 하는 matrix A를 Φ의 **Transformation Matrix**라 하고, $A = [\Phi]_B^C$와 같이 표기한다. 즉, basis의 각 원소를 보냈을 때의 coordinate를 column으로 가지는 matrix이다. 모든 linear mapping은 ordered basis로 정의되는 유일한 transformation matrix를 가진다.

이런 정의 자체는 증명할 게 없이 당연하다. 임의의 vector를 vector space W에 보내면 W의 basis로 표현할 수 있다. 중요한 지점은 어떤 linear mapping에 대해 동일한 두 basis를 사용한 경우에 이런 matrix representation이 유일한지이다.

참고로, 영변환 $T(x) = 0$의 matrix representation은 zero matrix이고, 항등변환 $T(x) = x$의 matrix representation은 identity matrix이다.

5.1.5. Basis Change

1. Change of Coordinate Matrix

Change of Coordinate Matrix(좌표변환행렬) Q는, finite vector space V의 basis β, β'에 대해서 다음과 같이 정의된다. 이때 I_V는 V에서의 항등변환이고, Q는 β'의 각 원소의, β에 대한 coordinate를 column으로 한다. 이를 활용해 동일한 vector를 다른 basis로 표현했을 때의 coordinate를 찾을 수 있다.

$$Q = [I_V]^{\beta}_{\beta'}, \quad [v]_{\beta} = Q[v]_{\beta'}$$

또한 I_V는 invertible이므로 Q도 invertible이고, Q^{-1}는 반대 방향으로의 coordinate change를 수행하는 matrix이다.

이때 β이 standard basis인 경우 Q는 단순히 새로운 basis로 column을 채워 얻을 수 있다.

2. Basis Change of Transformation Matrix

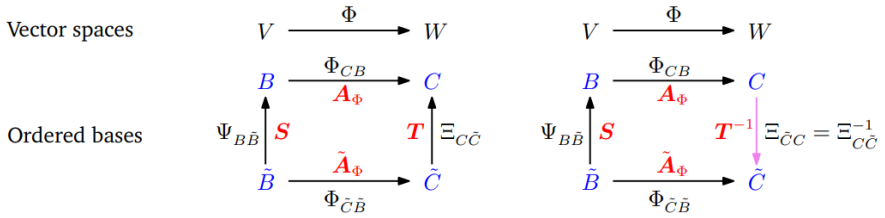
transformation matrix에 대한 **Basis Change**는 다음과 같이 change of coordination matrix를 곱하는 것으로 정의된다.

$$[T]_{\beta'} = [I_V]^{\beta'}_{\beta}[T]^{\beta}_{\beta}[I_V]^{\beta}_{\beta'} = Q^{-1}[T]_{\beta}Q$$

$$[T]^{\beta}_{\alpha} = ([I_W]^{\beta'}_{\beta})^{-1}[T]^{\beta'}_{\alpha'}[I_V]^{\alpha'}_{\alpha} = [I_W]^{\beta}_{\beta'}[T]^{\beta'}_{\alpha'}[I_V]^{\alpha'}_{\alpha}$$

$$A = T^{-1}\tilde{A}S$$

linear transformation의 관점에서 보면 다음과 같이 linear transformation의 합성으로도 이해할 수 있다.



β와 β'의 순서를 잘 맞춰주는 것으로 암기할 수 있다.

3. Equivalent & Similar

두 matrix A와 $\tilde{A}$에 대해 $A = T^{-1}\tilde{A}S$를 만족시키는 T와 S가 존재하면 두 matrix는 **Equivalent(동치)**하다고 하고, $A = S^{-1}\tilde{A}S$를 만족시키는 S가 존재하면 두 matrix는 **Similar(닮음)**하다고 한다. similar는 동일한 linear transform을 서로 다른 basis로 바라본 것을 의미한다.

당연하게도 equivalent하다고 모두 similar한 것은 아니지만, similar하면 equivalent하다.

similar한 matrix들은 eigen value의 개수와 값이 서로 같고, 이에 따라 trace와 determinant가 모두 같다. 또한 동일한 linear transform을 나타내므로 rank와 nullity도 같다. 이때 eigen vector는 다를 수 있다. 만약 $B = M^{-1}AM$이라면, $A\lambda = \lambda v$, $Bv = M^{-1}AMv = \lambda v$, $AMv = \lambda Mv$이다. 즉, A의 eigen value는 λ으로 동일하고 eigen vector는 Mv이다.

diagonalizable한 matrix A에 대해 $S^{-1}AS = \Lambda$이므로 A와 Λ는 similar하고, eigen value가 같다. 하지만 eigen value가 같다고 반드시 similar한 것은 아니다. eigen value가 같아도 대수적 중복도와 기하적 중복도가 일치하지 않는 matrix는 eigen decomposition이 불가능하다. 당연하게도 모든 eigen value의 대수적 중복도가 1인 경우에는 항상 similar하다.

5.1.6. Matrix Multiplication as Composition

transformation matrix에 대한 matrix multiplication과, linear transformation의 composition(합성)은 서로 대응된다. $f : V \to W$, $g : W \to X$인 두 linear transformation을 생각하면, $f \circ g : V \to X$ 또한 linear transformation이고 다음 수식이 성립한다.

$$A_{f \circ g} = A_f A_g$$

5.2. Pseudo Inverse Matrix

교재에서 다루진 않지만 함께 정리했다.

5.2.1. Left/Right Inverse Matrix

어떤 marix에 대해 왼쪽에 곱해져서 identity matrix를 만들면 **Left Inverse**, 오른쪽에 곱해져서 identitiy matrix를 만들면 **Right Inverse**이다.

- left inverse는 matrix가 full column rank일 때 구할 수 있다. $(A^TA)^{-1}A^TA = I$이므로 $(A^TA)^{-1}A^T$가 left inverse이다. 이때 A^TA의 inverse matrix를 사용하므로 full column rank여야 성립한다.

  left Inverse는 A의 column space에 있는 vector에 대해서는 A의 row space의 대응되는 값으로 그대로 되돌려주고, left null space에 있는 vector는 0으로 보낸다(left null space의 vector x에 대해서 그 정의에 의해 $A^Tx = 0$인 것을 생각하면 당연하다.).

  left inverse matrix를 오른쪽에 곱한 $A(A^TA)^{-1}A^T$는 A의 column space로의 projection matrix이다. 이는 left inverse matrix가 R^m에 존재하는 임의의 vector를 A의 row space로 보내고(A의 column space에 존재하지 않는 부분은 영벡터로 가므로 없어진다.), 보낸 vector를 A로 다시 보내는 것과 같다.

  least square 문제를 풀 때 곱하던 matrix가 A에 대한 left inverse matrix이다. left inverse matrix가 A의 R^m에 존재하는 임의의 vector를 column space로 보내는 것을 생각하면 이해할 수 있다.

- right inverse는 matrix가 full row rank일 때 구할 수 있다(r=m<n). $AA^T(AA^T)^{-1} = I$이므로, $A^T(AA^T)^{-1}$가 right inverse이다. 이때 마찬가지로 AA^T의 inverse matrix를 사용하므로 full row rank여야 성립한다.

  right inverse는 R^m의 vector를 R^n의 row space로 보내되, 해당 vector를 A에 넣었을 때 원래의 vector로 돌아오도록 하는 matrix이다. 이때 right inverse는 full row rank일 때 존재하므로, left null space는 존재하지 않는다.

  right inverse matrix를 왼쪽에 곱한 $A^T(AA^T)^{-1}A$는 A의 row space로의 projection matrix이다. column space로의 projection matrix에서 transpose 여부만 바뀐 것이므로 당연하다.

$x \neq y$이고, x와 y가 row space의 vector이면 $Ax \neq Ay$이다. 즉, row space에서 보내는 매핑은 항상 일대일 대응이고, 이에 따라 $m \times n$ matrix에 대해서도 inverse matrix와 유사한 left/right inverse matrix가 정의될 수 있다. 이는 귀류법으로 간단히 보일 수 있다. 만약 $Ax = Ay$라면 $A(x - y) = 0$인데, $x - y$는 null space의 원소이다. 근데 row space와 null space의 교집합은 $\{0\}$이므로 모순이다.

left/right inverse matrix는 four fundamental spaces에 대한 diagram을 그려서 생각하면 이해가 쉽다.

5.2.2. Pseudo Inverse

1. Pseudo Inverse
Pseudo Inverse 또는 Moore-Penrose Inverse는 임의의 $m \times n$ matrix에 대해 inverse matrix와 가장 유사하게 동작하는 matrix로, inverse, left/right inverse를 아우르는 가장 일반화된 개념이다.

pseudo inverse A^+는 다음과 같이 SVD와 유사하게 정의된다. 이때 Σ^+는 SVD의 Σ에서 singular value 중 0이 아닌 것들은 역수를 취하고, 0인 것들은 그대로 둔 뒤, transpose한 matrix이다. 즉, SVD에서와 같이 A^+는 rank r까지는 $A^+u = \frac{1}{\sigma}v$로 보내고, r+1부터는 영벡터로 보낸다.

$$A^+ = V\Sigma^+U^T$$

임의의 matrix는 row space의 vector를 column space의 vector로 일대일 매핑한다. 즉, non-invertible이어도 해당 부분으로 한정하면 일대일대응이다. pseudo inverse는 이런 매핑에서 vector를 반대로 보낸다. column space의 vector를 row space의 vector로 일대일 매핑하고, left null space에 해당하는 부분은 0으로 보내는 것으로 이해할 수 있다.

full rank인 square matrix인 경우 $A^+ = A^{-1}$이고, full column rank인 matrix에 대해서 A^+는 left inverse, full row rank인 matrix에 대해서 A^+는 right inverse이다. 물론 full rank가 아닌 경우에도 pseudo inverse는 존재하므로, pseudo inverse는 inverse, left/right inverse의 상위 개념이다.

2. Projection Matrix인가?
A^+A는 row space로의 projection matrix이다. 이 경우 vector에서 row space에 해당하는 부분은 유지되고, null space에 해당하는 부분은 제거되므로 당연하다. A가 full column rank인 경우 null space가 영공간이므로 $A^+A = I$가 되고, A^+는 left inverse이다.

AA^+는 column space로의 projection matrix이다. 이 경우 vector에서 column space에 해당하는 부분은 유지되고, left null space에 해당하는 부분은 제거되므로 당연하다. A가 full row rank인 경우 left null space가 영공간이므로 $AA^+ = I$가 되고, A^+는 right inverse이다.

6. Affine Space

6.1. Affine Space

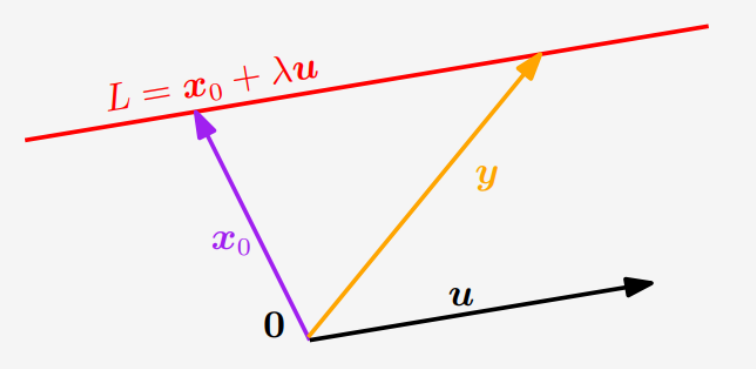
6.1.1. Affine Space

1. Affine Space
vector space V와 그 vector $x_0 \in V$, $U \subseteq V$에 대해 생각하자. 다음 수식을 만족하는 V에 대한 subset L을 subset **Affine Space** 또는 Linear Manifold라고 한다. 여기에서 V의 linear subspace인 U는 Direction 또는 **Direction space**라고 하고, x_0는 **Support Point**라고 한다. affine space는 영벡터를 포함하지 않으므로 vector space가 아니고, 각 원소를 vector가 아니라 **Point**(점)라고 한다. vector space는 affine space의 speical case인 것으로 이해할 수 있다.

$$L = x_0 + U = \{x_0 + u : u \in U\}$$

vector space V에 대해 affine space $L = x_0 + U$, $\tilde{L} = \tilde{x}_0 + \tilde{U}$가 존재한다고 할 때, $U \subseteq \tilde{U}$이고 $x_0 - \tilde{x}_0 \in \tilde{U}$이면 $L \subseteq \tilde{L}$이다.

예를 들어, R^3의 affine space로는 point, line, plane(평면)이 있다.



2. Parametric Equation
affine space에서 **Parameter**는 affine space 내의 특정 점의 위치를 정확하게 지정하기 위해 사용되는 scalar 값으로, affine space는 주로 parameter를 사용해 표현한다. vector space V에 대한 affine space $L = x_0 + U$가 있을 때, V의 subspace U의 ordered basis가 $(b_1, \cdots, b_k)$이라 하면 affine space L의 임의의 원소 x는 다음과 같이 표현할 수 있다.

$$x = x_0 + \lambda_1 b_1 + \cdots + \lambda_k b_k, \quad \lambda_1, \cdots, \lambda_k \in R$$

이런 표현식을 L에 대한 **Parametric Equation** 이라고 하고, $b_1, \cdots, b_k$를 **Directional Vector**라고 한다.

뒤에서 다루겠지만, R^n에 대해 n-1 차원 affine space는 Hyperplane이라고 한다. hyperplane은 support vector x_0와, n-1개의 원소를 포함하는 U의 basis로 구성된다.

Ax=b의 해는 공집합이거나, R^n에 대한 $n - rank$ 차원의 affine space인 것으로 이해할 수 있다. 또한 R^n에 대한 임의의 affine space는 어떤 nonhomogeneous equation Ax=b의 해이다. 또한 homogeneous equation Ax=0 또한 support vector가 0인 affine space라고 할 수 있다.

6.1.2. Affine Transformation

vector space V, W와 그 linear mapping $f : V \to W$, $a \in W$를 생각하자. **Affine Mapping** $\phi : V \to W$는 다음과 같이 정의된다. 이때 a를 **Translation(평행이동) Vector**라고 한다.

$$\phi(x) = a + f(x)$$

임의의 affine mapping ϕ는 유일한 translation $\tau : W \to W$와 유일한 linear mapping $f : V \to W$의 합성이다. 즉, $\phi = \tau \circ f$이다. linear mapping이후 평행이동 한 것이므로 당연하다.

affine mapping과 affine mapping의 합성도 affine mapping이다.

affine mapping은 기하학적 구조를 유지하며, dimension과 parallelism을 보존한다.

linear mapping은 평행이동을 허용하지 않고, affine mapping은 linear mapping에 평행이동을 결합해 평행이동을 허용하도록 일반화된 mapping이다.

7. Analytic Geometry

Analytic Geometry(해석기하학)는 좌표계를 사용하여 기하학적 도형(ex. 선, 원, 곡선 등)을 대수적 방정식으로 표현하고 연구하는 학문이다. 본 장에서는 linear algebra의 개념들에 대한 기하학적 해석과 직관적인 이해를 살펴본다.

7.1. Norm

7.1.1. Norm

1. Norm
Norm(노름)은 vector의 크기이자, vector를 기하학적으로 생각했을 때 원점부터 화살표 끝까지의 거리이다. 이때 $||\cdot|| : V \to \mathbb{R}$는 vector x를 그 길이 $||x||$로 보내는 mapping이다.

vector $x, y \in V$, scalar λ에 대해 생각했을 때 다음이 성립한다.

- Absolutely homogeneous: $||\lambda x|| = |\lambda| ||x||$
- Triangle inequality: $||x + y|| \leq ||x|| + ||y||$
- Positive definite: $||x|| \geq 0$이고, $||x|| = 0$인 것과 $x = 0$인 것은 필요충분이다.

어떤 vector v를 그 norm $||v||$으로 나눠 크기를 1로 만드는 것을 **Normalization**이라고 한다.

2. 다양한 Norm
다음과 같이 다양한 norm이 존재한다. 본 요약본에서는 euclidian norm을 기본으로 한다.

- **Euclidian norm(L2 norm)**은 다음과 같이 정의된다.

$$||x||_2 = \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{x^T x}$$

- Manhattan norm(L1 norm)은 다음과 같이 정의된다.

$$||x||_1 = \sum_{i=1}^{n} |x_i|$$

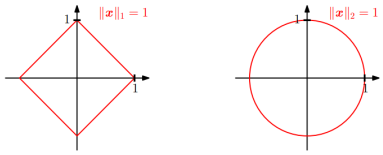


Figure 3.3 For different norms, the red lines indicate the set of vectors with norm 1. Left: Manhattan norm; Right: Euclidean distance.

7.2. Inner Product

inner product는 norm, distance, angle 등의 정의에 사용된다.

7.2.1. Inner Product

1. Bilinear Mapping
Bilinear mapping은 두 인자에 대해 linaer한 mapping으로, $x, y, z \in V$와 scalar $a, b \in \mathbb{R}$에 대해 다음을 만족시키는 mapping을 말한다.

$$f(ax + y, z) = af(x, z) + bf(y, z) f(x, ay + bz) = af(x, y) + bf(x, z)$$

2. Symmetric and Positive Definite
vector space V에 대해 그 vector 2개를 받아 하나의 scalar로 보내는 mapping $\Omega : V \times V \to \mathbb{R}$를 생각하자. symmetric과 positive definite은 다음과 같이 정의된다.

- vector $x, y \in V$에 대해 $\Omega(x, y) = \Omega(y, x)$가 성립하면 Ω는 **Symmetric**이다. 즉, 인자의 순서가 상관이 없는 mapping이다.
- 다음 수식이 성립하면 Ω는 **Postitive Definite**이다. 즉, 0을 제외한 모든 경우에 대해 0보다 큰 값을 도출하는 mapping이다.

$$\forall x \in V \backslash \{0\} : \Omega(x, x) > 0, \quad \Omega(0, 0) = 0$$

3. Inner Product
positive definite이면서 symmetric bilinear mapping인 $\Omega : V \times V \to \mathbb{R}$를 **Inner Product**라고 하고, $\langle x, y \rangle$로 표기한다.

vector space에 inner product가 추가된 vector space $(V, \langle \cdot, \cdot \rangle)$를 **Inner Product Space** 또는 Vector Space with Inner Product라고 한다. 이때 inner product로 dot product를 사용한다면 Euclidean Vector Space라고 한다.

가장 널리 쓰이는 inner product는 **Dot Product**로, 다음 수식과 같이 정의된다.

$$\langle x, y \rangle = x^T y = \sum_{i=1}^{n} x_i y_i$$

inner product의 결과는 scalar이므로, transpose해도 그 결과가 같다. 예를 들어, 두 vector $x, y \in \mathbb{R}^n$에 대해 $x^T y = y^T x$가 성립한다.

7.2.2. Positive Definite Matrix

1. Positive Definite Matrix
Positive Definite Matrix(양의 정부호 행렬)는 $x \neq 0$인 임의의 vector x에 대해서 $x^TAx > 0$이 성립하는 symmetric matrix이다.

positive definite matrix에 대해서는 다음과 같은 성질이 성립한다. 또한 어떤 matrix가 positive definite matrix인지는 이 세 가지 중 하나가 성립하는 것을 보이면 되고, 이때 하나가 성립하면 나머지도 성립한다. 즉, 이 세 가지 중 하나만 성립해서 $x \neq 0$인 x에 대해서 $x^TAx > 0$아다.

- 모든 eigen value가 양수이다.

  A에 대한 $Av = \lambda v$인 eigen vector v를 생각하자. $v^TAv = \lambda v^Tv > 0$ 이어야 하는데, eigen vector는 영벡터가 아니므로 $\lambda > 0$이 성립한다. 당연하게 그 역도 성립한다.

  determinant는 모든 eigen value를 곱해서 구할 수 있으므로 positive definite의 determinant 또한 양수이고, 항상 invertible하다.

- 모든 pivot이 양수이다.

  symmetric matrix에서는 eigen value의 부호 개수와 pivot의 부호 개수가 같으므로 당연하다.

- 해당 matrix의 왼쪽 위 끝의 원소를 포함하는 sub-matrix의 determinant가 모두 양수이다.

  한 row의 scalar배를 다른 row에 곱하는 연산은 determinant를 보존하므로, 이런 sub-matrix의 determinant를 작은 쪽부터 순차적으로 구해 보면 pivot의 부호에 따라 부호가 정해지는 것을 알 수 있다.

등호가 붙으면(어떤 eigen value가 0) 각각 Positive Semi-definite Matrix(양의 준정부호 행렬)이라고 한다. 또한 $x^TAx < 0$이 성립(모든 eigen value가 음수)하는 matrix는 Negative Definite Matrix(음의 정부호 행렬)이라고 하고, 등호가 붙으면 Negeative Semi-definite Matrix(음의 준정부호 행렬)라고 한다. 또한 eigen value에 양수와 음수가 섞여 있는 상태는 Indefinite Matrix(부정부호 행렬)라고 한다.

positive definite matrix A에 대해 다음 성질이 성립한다.

- A에 대해 영벡터를 제외하고 $x^TAx > 0$이 성립하므로 A의 nullity는 0이다.
- standard basis $e = \{e_1, \cdots, e_n\}$을 생각했을 때 $e_i^TAe_i > 0$이 항상 성립하므로, A의 대각성분은 모두 양수이다.

2. Inner Product와 Positive Definite
finite vector space에서 inner product와 positive definite은 동치이다.

finite vector space V의 ordered basis $B = \{b_1, \cdots, b_n\}$을 생각하자. B에 대한 임의의 vector $x, y \in V$의 coordinate를 각각 $\hat{x}, \hat{y}$라 하면, x와 y를 basis에 대한 linear combination으로 나타내면 inner product는 bilinearity에 의해 다음과 같이 정리된다. 이때 $A_{ij} = \langle b_i, b_j \rangle$이다.

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \left\langle \sum_{i=1}^{n} \psi_i \boldsymbol{b}_i, \sum_{j=1}^{n} \lambda_j \boldsymbol{b}_j \right\rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} \psi_i \langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle \lambda_j = \hat{\boldsymbol{x}}^\top \boldsymbol{A} \hat{\boldsymbol{y}}$$

이때 inner product는 positive defintie이고 symmetric이므로, A도 symmetric이면서 다음이 성립한다. 즉, 이때의 A는 positive definite이다.

$$\forall x \in V \backslash \{0\} : x^TAx > 0$$

또한 symmetric matrix A가 positive definite이면 대응되는 ordered basis B에 대해 inner product가 정의된다. 즉, B에 대해 A가 존재하면 inner product가 정의된다.

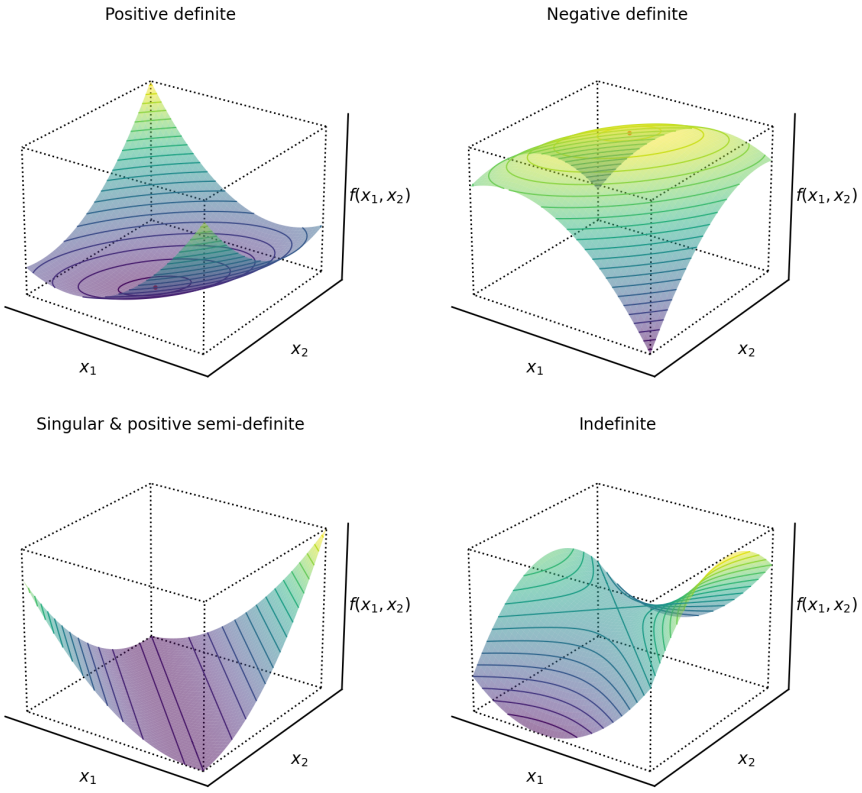
3. Positive Definite의 의미
positive definite의 정의를 나타내 보면 다음과 같이 $x_1, \cdots, x_n$에 대한 이차방정식이 0보다 크다는 것을 의미한다. 즉, 최솟값이 존재하는 경우이다. 또한 이때 변수 $x_1, \cdots, x_n$ 각각에 대한 완전제곱식 형태로 수식을 묶어 정리하면, $x_1^2, \cdots, x_n^2$의 계수가 pivot과 같다(2×2 matrix에 대해 계산해볼 수도 있고, LDL^T decomposition을 사용해 증명할 수 있다고 한다.).

$$x^T Ax = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \\ \cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n \end{bmatrix} = a_{11}x_1^2 + a_{22}x_2^2 + \cdots + a_{nn}x_n^2 + \cdots > 0$$

이를 2×2 matrix A에 대해 그래프를 그려보면, positive definite(밥그릇 모양), negative definite(뒤집힌 밥그릇 모양), positive semi-definite(골짜기 모양), indefinite(안장 모양) 각각에 대해 $x^T Ax$의 그래프는 다음과 같다. λ의 부호에 따라 그래프가 달라지는 것을 이해할 수 있다.

positive semi-definite의 경우 어떤 eigen value가 0이어서 해당 방향의 경우 이동해도 값이 변하지 않게 되고, 골짜기 모양의 그래프가 그려진다. indefinite의 경우 양수인 eigen value와 음수인 eigen value가 둘 다 존재해서 양의 무한대와 음의 무한대로 발산하는 상태가 공존하고, 안장 모양의 그래프가 그려진다.

또한 각 column이 eigen vector인 matrix Q에 대해서 x=Qu라고 하면, $x^T Ax = u^T Q^T Q \Lambda Q^T Qu = u^T \Lambda u = \lambda_1 u_1^2 + \lambda_2 u_2^2$과 같이 각 vector 별로 하나의 항으로 정리할 수 있다. 위에서 내려다보는 것을 생각했을 때, $\lambda_1 u_1^2 + \lambda_2 u_2^2 = 1$인 경우 λ의 부호가 둘 다 양수이면 타원이 되고, λ의 부호 중 하나가 음수이면 쌍곡선이 된다.



positive definite matrix는 입력 vector와 보낸 vector의 inner product가 양수이므로 vector의 방향이 반대로 꺾이지 않도록 보내는 matrix로도 이해할 수 있다.

symmetrix matrix는 diagonalizable하고, pivot의 부호 개수와 eigen value의 부호 개수가 같다. 또한 symetric matrix에서 determinant는 pivot의 곱이므로, determinant의 부호에 따라 positive definite인지가 바뀐다. positive definite인 경우 determinant가 양수인데, 성분 값을 바꾸었을 때 determinant가 0이 되는 지점에서 positive semi-definite이 되고, 0에서 음수로 넘어가면 negative definite이거나 indefinite이 된다.

예를 들어, 2×2 matrix가 positive definite인지 보이려면, symmetric인지 확인하고 $x = (x_1, x_2)^T$를 사용해 항상 양수임을 확인할 수 있다.

7.2.3. Positive Definite 관련 성질

positive definite는 다음과 같은 성질들을 가진다.

- $n \times n$ matrix A가 positive definite이면 A^{-1}도 positive definite이다.

  inverse matrix는 eigen value가 역수이므로 당연하다.

- $n \times n$ matrix A, B가 positive definite이면 $A + B$도 positive definite이다.

- $m \times n$ matrix A에 대해 $n \times n$ matrix $A^T A$는 symmetric이면서 positive semi-definite이다. 이때 A가 full column rank인 경우 $Ax = 0$을 만족시키는 $x \neq 0$인 x가 존재하지 않으므로 positive definite이 된다.

  $A^T A$가 symmetric인 것은 자명하고, $x^T A^T Ax = (Ax)^T Ax \geq 0$ 이므로 항상 positive semi-definite이다.

7.2.4. Norm과 inner product

모든 inner product는 norm을 유도하지만, 모든 norm이 inner product로 유도되는 것은 아니다. 예를 들어, manhattan norm은 inner product로 유도되지 않는다.

$$||x|| = \sqrt{\langle x, x \rangle}$$

inner product로 유도되는 norm에 대해서는 다음과 같은 Cauchy-Schwarz Inequality(코시-슈바르츠 부등식)이 성립한다.

$$|\langle x, y \rangle| \leq ||x||\,||y||$$

7.2.5. Inner Product of Functions

function 사이의 inner product는 다음과 같이 definite integral(정적분)로 정의된다. 이에 따라 function에 대한 norm과 orthogonality를 계산할 수 있다.

$$\langle u, v \rangle = \int_a^b u(x)v(x)dx, \quad a, b < \infty$$

function에 대한 integral 값은 발산할 수 있으므로 유한한 값을 가지는 vector의 inner product와 차이가 있다. 또한 function에 대한 inner product의 정의를 명확하게 하려면 integral의 measures를 다뤄야 하는데, 여기에서는 다루지 않는다.

예를 들어, $\{1, \cos x, \cos 2x, \cdots\}$는 $-\pi$부터 π까지로 적분해보면 해당 집합은 orthogonal한 것을 알 수 있다. 이 집합으로 function을 projection하는 것이 fourier series의 기본 아이디어이다.

7.3. Distance & Angle

7.3.1. Distance

innver product space의 vector x, y에 대한 **Distance**는 다음과 같이 정의된다. 물론 inner product로 정의되지 않는 norm에 대해서는 해당 norm만으로 distance가 정의된다.

$$d(x, y) = ||x - y|| = \sqrt{\langle x - y, x - y \rangle}$$

또한 inner product로 dot product를 사용하는 경우의 distance를 **Euclidean Distance**라고 한다.

distance는 norm(그리고 inner product)으로 정의되므로 positive definite이면서 symmetric이다. 또한 $d(x, z) \leq d(x, y) + d(y, z)$으로, triangle inequality를 만족한다.

7.3.2. Angle

두 vector 간의 **Angle**은 다음과 같이 정의된다. angle은 두 vector의 방향이 얼마나 유사한지를 나타낸다.

$$\cos\theta = \frac{\langle x, y\rangle}{||x||\,||y||}$$

추가로, cauchy-schwarz inequaltiy에 의해서도 $||x|| \neq 0, ||y|| \neq 0$인 vector x, y에 대해서 다음과 같이 정의되는 것을 알 수 있다.

$$-1 \leq \frac{|\langle x, y\rangle|}{||x||\,||y||} \leq 1$$

7.4. Orthogonality

7.4.1. Orthogonality of Vector

두 vector x, y에 대해서 inner product $\langle x, y\rangle = 0$이면 이 두 vector는 **Orthogonal**하다고 하고, $x \perp y$로 표기한다. 또한 orthogonal한 vector x, y에 대해 $||x|| = ||y|| = 1$이면 **Orthonormal**하다고 한다.

이에 따라 영벡터는 모든 vector와 orthogonal하다.

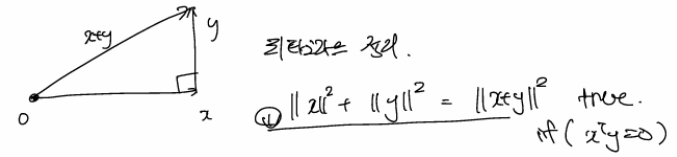
영벡터가 아닌 두 vector가 orthogonal하면 linear independent이다.

orthogonality는 inner product에 의해 정의되므로, 어떤 inner product에서 두 vector가 orthogonal해도, 다른 inner product에 대해서는 그렇지 않을 수도 있다.

orthogonal하다는 것은 서로 수직하다는 의미이다. 피타고라스 정리를 예시로 생각해보자. 다음 그림과 같이 x와 y가 수직이라면 $||x||^2 + ||y||^2 = ||x + y||^2$가 성립한다. x와 y 각각의 성분을 $x_1, \cdots, x_n$, $y_1, \cdots, y_n$이라고 했을 때, $||x||^2 = x_1^2 + \cdots + x_n^2 = x^T x$이 성립한다. 이에 따라 피타고라스 정리는 다음 수식과 같이 정리된다.

$$x^T x + y^T y = (x + y)^T (x + y) = x^T x + x^T y + y^T x + y^T y$$

정리하면 $x^T y + y^T x = 0$이 성립하는데, 계산해보면 이 두 항의 값은 $x_1 y_1 + \cdots + x_n y_n$으로 같은 것을 알 수 있다. 즉, 수직인 두 vector x, y에 대해서 $x^T y = 0$이 성립한다.

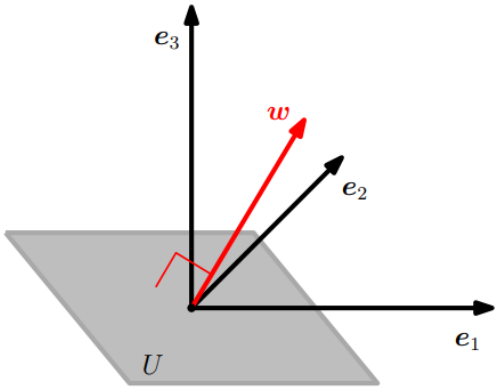


7.4.2. Orthogonality of Vector Space

1. Orthogoanlity of Vector Space
두 vector space S, T에 대해 모든 $v \in S$가 $w \in T$에 대해 orthogonal하면 S와 T는 orthogonal하다고 한다. 이는 두 vector space의 basis끼리 orthogonal한지 비교하는 것으로 판단할 수 있다.

당연하게도 zero vector space는 임의의 vector space와 orthogonal하다. 또한 intersection(교집합)이 존재하는 두 vector space는 orthogonal하지 않다.

dimension이 n인 inner product space V와 dimension이 k인 그 subspace $U \subseteq V$에 대해, U에 orthogonal한 **Orthogonal Complement** $U^\perp$는 dimension이 n-k인 V의 subspace이다. 즉, orthogonal하면서 V의 나머지 부분을 차지하는(direct sum) vector space이다. 이에 따라 임의의 vector $v \in V$는 U와 $U^\perp$의 basis로 표현될 수 있다. 예를 들어, 3차원에서 어떤 평면에 대한 orthogonal complement는 해당 평면에 수직인 직선이다. 참고로 이런 수직인 vector를 **Normal Vector(법선 벡터)**라고 한다.



2. Row Space and Null Space
A의 row space는 A의 null space와 orthogonal하다. null space에 속하는 임의의 vector는 Ax=0을 만족시키고, 이에 따라 A의 모든 row와의 inner product가 0이므로 당연하다.

이때 이 두 subspace는 R^n에 대해 **Orthogonal Complement(직교여공간)** 관계에 있다. 즉, row space와 null space는 서로가 자신의 모든 vector와 orthogonal한 모든 vector를 포함하는 subspace이다.

또한 row space의 dimension은 n-r이고 null space의 dimension은 r이므로 이 두 subspace는 R^n을 완전하게 구성한다. 즉, 임의의 vector $x \in R^n$는 $x = x_{\text{row}} + x_{\text{null}}$과 같이 쪼갤 수 있다. 이를 A에 넣으면 $Ax = A(x_{\text{row}} + x_{\text{null}}) = Ax_{\text{row}} = b$이다.

이때 임의의 vector $b \in C(A)$는 오직 하나의 row space 상 vector와 대응된다. 이에 대한 증명은 간단하다. 어떤 두 vector $x_1, x_2 \in C(A^T)$에 대해 $Ax_1 = Ax_2 = b$가 성립한다고 가정하면 $A(x_1 - x_2) = 0$이므로 $x_1 - x_2 \in N(A)$인데, $x_1 - x_2$는 그 자체로 linear combintion이므로 $x_1 - x_2 \in C(A^T)$이다. 즉, row space와 null space의 intersection으로는 영벡터만 있으므로 모순이다.

3. Column Space and Left Null Space
A의 column space는 A의 left null space와 orthogonal하다.

column space의 임의의 vector x_1에 대해서는 $Ay = x_1$이 성립하고, left null space의 임의의 vector x_2에 대해서는 $x_2^T A = 0$이 성립한다. $x_2^T A = 0$의 양변에 y를 곱하면 $x_2^T Ay = x_2^T x_1 = 0$이다.

7.4.3. Orthogonal Matrix

1. Orthogonal Matrix
Orthogonal Matrix는 orthonormal vector들로 column을 구성한 matrix이다. 즉, orthogonal matrix Q의 각 column을 $q_1, \cdots, q_n$이라고 하면 다음 수식이 성립한다. 이때의 vector들이 orthonormal이기는 하지만 관습적으로 orthogonal matrix라고 부른다고 한다.

$$q_i q_j = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$

다시 말해, square matrix Q에 대해 다음이 성립하면 orthogonal matrix라고 한다.

$$QQ^T = Q^T Q = I$$

2. Orthogonal Matrix의 성질
orthogonal matrix에 대해 다음 성질들이 성립한다.

- $m \times n$ matrix Q에 대해 $n \times n$ matrix인 $Q^T Q$는 identity matrix이다. 곱해보면 당연하다.
- Q가 square matrix이면, $Q^T = Q^{-1}$이다. $Q^T Q = I$가 성립하므로 당연하다.

- orthogonal matrix를 사용한 transformation은 norm과 angle을 보존한다.

  orthogonal matrix Q를 생각하면, $(Qx)^T(Qx) = x^TQ^TQx = x^Tx$이므로 Q는 norm을 보존하고, $(Qx)^T(Qy) = x^TQ^TQy = x^Ty$이므로 Q는 각도(inner product)를 보존한다. orthogonal matrix는 rotation matrix 또는 reflection matrix이다.

- Q의 column space로 projection하는 projection matrix P는 QQ^T이다. 정리해보면 당연하다. 또한 만약 Q가 square matrix이면 $P = I$이다.

  Q가 $n \times n$ square matrix이면서 orthogonal matrix이면 R^n에서 R^n으로 보내는 matrix이므로, projection 해도 이미 동일한 공간 안에 있고, 이에 따라 projection matrix가 I인 것으로도 이해할 수 있다.

$A^TA = AA^T$를 만족시키는 matrix는 Normal Matrix라고 한다. orthogonal matrix는 normal matrix이다.

orthogonal matrix가 구체적으로 왜 좋을까? 한 예시로, orthogonal matrix의 성질에 따라 뒤에서 설명할 least square problem을 더 쉽게 풀 수 있다. least square problem을 projection으로 풀 때 Ax=b를 $A^TA\hat{x} = A^Tb$로 변형하는데, A가 orthogonal matrix라면 $A^TA\hat{x} = \hat{x} = A^Tb$이다. 즉, 우변을 계산하기만 하면 된다.

7.4.4. Orthonormal Basis

vector space V의 basis $B = \{b_1, \cdots, b_n\}$에 대해 다음 수식이 성립할 때, 이를 **Orthonormal Basis(ONB)**라고 한다. 즉, 모든 원소의 norm이 1이고 서로 orthogonal한 basis를 말한다.

$$\langle b_i, b_i \rangle = 1, \ \ \langle b_i, b_j \rangle = 0 \ (i \neq j)$$

norm이 1이 아니고 orthogonal하기만 하면 Orthogonal Basis라고 한다.

7.5. Projection

ML에서는 어떤 feature에 대해 분석할 때 더 낮은 차원의 feature space로 보내 그 복잡도를 낮춘다. projection을 적용해 낮은 feature의 차원을 낮출 때 loss를 최소로 할 수 있다.

7.5.1. Projection

vector space V와 그 subspace $U \subseteq V$에 대해, linear mapping $f : V \to U$가 $f^2 = f$를 만족시킬 때 linear mapping f를 **Projection**이라고 한다.

또한 이런 linear mapping에 대한 transformation matrix P는 $P^2 = P$가 성립하고, 이를 **Projection Matrix**라고 한다.

7.5.2. A^TA 활용하기

$m \times n$ A에 대해 실제로 Ax=b를 푸는 상황을 공학적으로 생각해 보면, column은 고정되고 여러 개의 입력값들이 각 row에 들어오게 된다. 즉, $m > n$인 A에 대해 계산해야 하고, 항상 해가 존재하지는 않는다. 이런 경우 $Ax = b$를 $A^TAx = A^Tb$로 변형해 문제를 푸는 것이 유리하다. 뒤에서 설명할 것처럼, 이는 A의 column space에 b를 projection하고, 이에 따라 해가 존재하지 않는 원래 문제에 대해 가장 좋은 근사해를 구할 수 있다.

또한 A^TA는 다음과 같은 성질을 가진다. 특히 세 번째 성질이 중요하다.

- A^TA는 square matrix이다.
- A^TA는 symmetric matrix이다.
- A^TA와 A의 null space는 같다. 이에 따라 A가 full column rank를 가진다면 nullity가 0이고, A^TA의 nullity도 0이 되어 invertible이다.

  반대로 A가 full column rank를 가지지 않는다면 A^TA는 non-invertible이다.

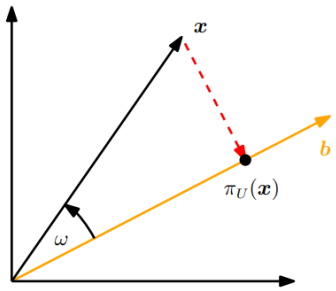
마지막 성질은 1. $N(A^TA)$의 임의의 vector가 $N(A)$에 속하는가, 그리고 2. $N(A)$의 임의의 vector가 $N(A^TA)$에 속하는가를 보여 증명할 수 있다.

1번은 양변의 왼쪽에 x^T를 곱하고 정리하면 $(Ax)^T Ax = u^T u = 0$이고, inner product가 0이 되는 vector는 영벡터밖에 없으므로 $Ax = 0$이 성립하는 것으로 증명할 수 있다. 2번은 $Ax = 0$인 x를 $A^T A$에 넣어 쉽게 증명할 수 있다.

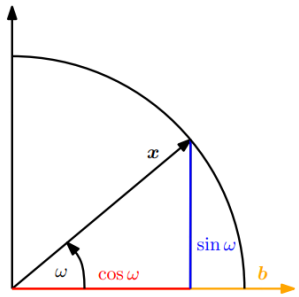
7.5.3. Orthogonal Projection

Ax=b의 해가 없다면, b를 C(A)로 projection하여 $A\hat{x} = p$를 구할 수 있다.

1. Projection onto One-dimension Subspaces (Lines)
vector b를 vector a가 span해 생성하는 subspace U로 **Orthogonal Projection**하는 것은 다음 그림과 같이 이해할 수 있다. b를 projection한 vector $\pi_U(b)$(projection point)는 U에 있으면서 b와 가장 가까운(distance가 최소인) vector이다. 즉, $\langle b - \pi_U(b), a\rangle = 0$이고, scalar x에 대해 $\pi_U(b) = xa$이다.



(a) Projection of $\boldsymbol{x} \in \mathbb{R}^2$ onto a subspace U with basis vector $\boldsymbol{b}$.



(b) Projection of a two-dimensional vector $\boldsymbol{x}$ with $\|\boldsymbol{x}\| = 1$ onto a one-dimensional subspace spanned by $\boldsymbol{b}$.

이를 활용해 projection matrix를 정의할 수 있고, 이 projection matrix를 곱해 b를 a가 span하는 line으로 projection할 수 있다. bilinearity를 활용해 일반적인 inner product에 대해 정의할 수 도 있지만, inner product로 dot product를 활용하는 경우 더 간단히 정리된다.

$\langle b - xa, a\rangle = \langle a, b - xa\rangle = 0$를 풀어서 정리하면 $a^T a$는 상수이므로 $x = \frac{a^T b}{a^T a}$이고, $\pi_U(b) = xa = ax = a\frac{a^T b}{a^T a} = \frac{aa^T}{a^T a}b$이다. 즉, a에 orthogonal projection을 하는 projection matrix는 다음과 같다.

$$P = \frac{aa^T}{a^T a}, \quad \pi_U(b) = \frac{aa^T}{a^T a}b$$

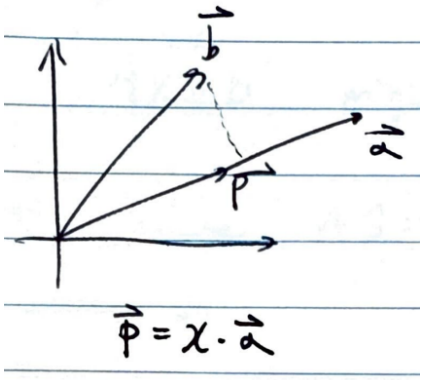
또한 $\pi_U(b) = xa$를 정리해 보면 그 norm은 다음과 같다.

$$||\pi_U(b)|| = |\cos w|\,||x||$$

projection matrix P에 대해 아래의 성질이 존재한다. 또한 수식적으로 봤을 때 b를 scalar배하면 p도 동일한 만큼 scalar배 되지만, a는 scalar배 해도 p는 변하지 않는다.

- $rank(P) = 1$이다. $a^T a$는 scalar이고 aa^T는 $n \times n$ matrix가 되는데, 모든 column이 a의 linear combination으로 만들어지므로 당연하다.
- $C(P)$는 a의 span이다. 이 또한 P의 모든 column이 a의 linear combination으로 만들어지므로 당연하다.
- $P^T = P$이다.
- $P^2 = P$이다. 이는 수식적으로도 증명이 되고, projection을 했는데 또 하는 것은 변화가 없으므로 당연하다. 또한 이에 따라 이미 projection된 vector는 P에 대한 eigen vector이고, 이때 eigen value는 1이다.

어떤 matrix가 projection matrix인지는 $P^2 = P$, $P^T = P$가 성립하는지로 보일 수 있다.



2. Projection onto General Subspaces
임의의 subspace에 대한 projection을 생각해보자.

어떤 subspace U에 vector b를 projection한 vector(projection point)를 $\pi_U(b)$라고 하면, $\pi_U(b)$는 subspace의 basis $\{a_1, \cdots, a_n\}$로 나타낼 수 있다. 이 basis로 column을 구성한 matrix A를 생각하면 $\pi_U(b) = Ax$로 나타낼 수 있고, distance가 최소여야 하므로 a_i에 대해 $\langle a_i, b - Ax \rangle = 0$이 성립한다. inner product로 dot product를 사용하는 경우 $A^T(b - Ax) = 0$로 나타낼 수 있다.

이제 $A^T(b - Ax) = 0$를 정리하면 $A^TAx = A^Tb$인데, 이를 Normal Equation이라고 한다(x를 구하는 경우 이 equation을 풀어 얻을 수 있다.). 이때 A를 basis로 구성했으므로 A^TA는 invertible이다. 즉, $x = (A^TA)^{-1}A^Tb$로 정리할 수 있다. 참고로, $(A^TA)^{-1}A^T$는 A에 대한 pseudo inverse이다.

이를 대입하면 $\pi_U(b) = Ax = A(A^TA)^{-1}A^Tb$이므로 subspace로의 projection matrix P는 다음과 같이 정의된다. 참고로 이때 $P = A(A^TA)^{-1}A^T$의 괄호는 풀고 싶게 생겼지만, A가 invertible하다는 보장이 없으므로(심지어 square matrix가 아닐 수도 있다.) 불가능하다.

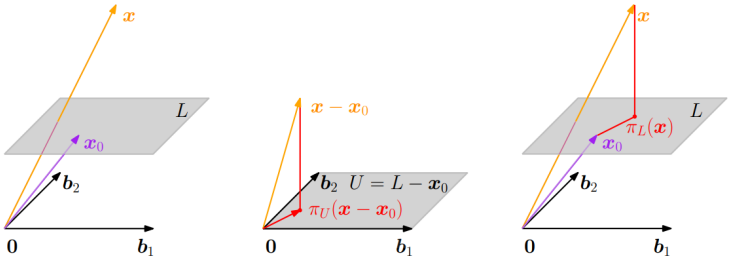
$$P = A(A^TA)^{-1}A^T$$

이때 basis가 ONB(orthonormal baiss)이면 $B^TB = I$이므로 normal equation과 projection matrix는 다음과 같이 간단해진다. 즉, inverse matrix를 계산하지 않아도 된다.

$$x = A^Tb, \quad P = AA^T$$

3. Projection onto Affine Spaces
subspace U와 affine space $L = x_0 + U$를 생각하자. vector $x \in U$에 대한 affine space로의 orthogonal projection은, $x - x_0$를 U에 projection하고, 이후 x_0를 더해주는 것으로 쉽게 구할 수 있다.

$$\pi_L(x) = x_0 + \pi_U(x - x_0)$$



(a) Setting.

(b) Reduce problem to projection π_U onto vector subspace.

(c) Add support point back in to get affine projection π_L.

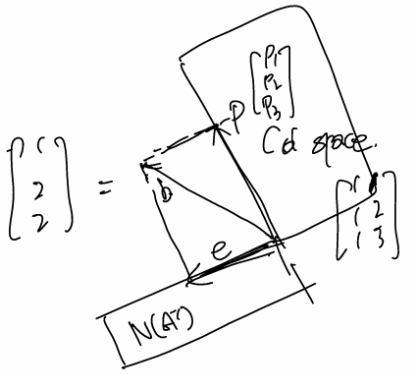
또한 다음과 같이 x와 affine space L 사이의 distance는 $x - x_0$와 subspace U 사이의 distance와 같다.

$$d(x, L) = ||x - (x_0 + \pi_U(x - x_0))|| = d(x - x_0, \pi_U(x - x_0))$$

U의 dimension이 1이면 A^TA가 scalar이므로 line으로의 projection과 같이 분수로 쓸 수 있다.

당연하게도 b를 A에 대해 projection할 때, $b \in C(A)$이면 그대로 b이다. 또한 b가 A에 직교하면 영벡터가 된다. 단순 대입으로 쉽게 증명할 수 있다. $b \in C(A)$이면 $Ac = b$라고 하자. $Pb = A(A^TA)^{-1}A^TAc = A(A^TA)^{-1}(A^TA)c = Ac = b$이다. 또한 b가 A에 직교하면 A의 모든 basis와 직교하고, $b \in N(A^T)$이므로 $A^Tb = 0$이다. 이것도 대입해 보면 $Pb = A(A^TA)^{-1}A^Tb = 0$이므로 영벡터인 것을 알 수 있다.

projection에서 $e = b - p$였다. 즉, $b = p + e$이다. fundamental space에서 보면 $C(A)$의 p와 $N(A^T)$의 e가 더해져서 공간 외부의 b를 만드는 것을 알 수 있다. 즉, projection matrix P를 곱하면 외부의 b에서 p가 된다. 이때 b에 어떤 matrix Q를 곱해 $e \in N(A^T)$로도 projection할 수 있다. 즉, $e = Qb$이다. 정리하면 $b = Pb + Qb$, $Q = I - P$이다. 즉, Q는 identity matrix에서 P를 뺀 것이다. 추가로 이런 Q도 projection matrix이므로 앞에서 다룬 성질이 성립한다.



참고로 projection matrix의 eigen value는 0 또는 1이다. 해당 공간에 없는 vector들에 대해서는 0, 해당 공간에 있는 vector들에 대해서는 1이다.

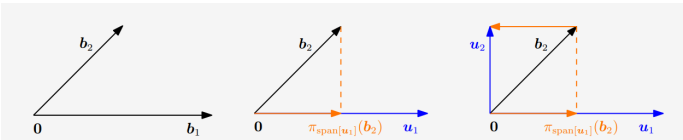
aa^T는 symmetric rank-1 matrix이다.

7.5.4. Gram Schmidt Process

Gram Schmidt Process(그람 슈미트 과정) 또는 Gram Schmidt Orthogonalization Method는 임의의 vector space V의 basis를 orthogonal basis로 변환하는 방법이다. 이 변환은 이렇게 얻은 orthogonal basis에 normalization을 적용해 ONB를 얻을 수 있다.

n-dimension vector space V의 basis $\{v_1, \cdots, v_n\}$에 대한 gram schmidt process는 다음과 같이 projection을 활용해 orthogonality를 유지하며 vector를 하나씩 추가한다.

$$q_1 = v_1$$
$$q_k = v_k - \pi_{\text{span}(q_1, \cdots, q_{k-1})}(b_k) \quad (k = 2, \cdots, n)$$



(a) Original non-orthogonal basis vectors $\boldsymbol{b}_1, \boldsymbol{b}_2$.

(b) First new basis vector $\boldsymbol{u}_1 = \boldsymbol{b}_1$ and projection of $\boldsymbol{b}_2$ onto the subspace spanned by $\boldsymbol{u}_1$.

(c) Orthogonal basis vectors $\boldsymbol{u}_1$ and $\boldsymbol{u}_2 = \boldsymbol{b}_2 - \pi_{\text{span}[\boldsymbol{u}_1]}(\boldsymbol{b}_2)$.

다시 말해, matrix $W = [v_1, v_2, v_3, \cdots]$을 orthogonal한 vector들을 column으로 가지는 matrix $Q = [q_1, q_2, q_3, \cdots]$로 변환하는 gram schmidt process는 다음과 같은 과정으로 수행된다.

1. v_1을 q_1으로 한다.
2. $i > 1$일 때 q_i는 $q_1, \cdots, q_{i-1}$들과 모두 orthogonal인 vector이어야 하므로, projection에 의한 error를 반복해 구하는 것으로 얻을 수 있다. 즉, v_i를 $q_1, \cdots, q_{i-1}$ 각각에 projection한 vector를 $p_1, \cdots, p_{i-1}$라 하면 $q_i = v_i - p_1 - \cdots - p_{i-1}$이다.

   예를 들어, q_3는 다음과 같이 구할 수 있다.

$$q_3 = v_3 - \frac{q_1 q_1^T}{q_1^T q_1} v_3 - \frac{q_2 q_2^T}{q_2^T q_2} v_3$$

이때 $q_1, \cdots, q_n$은 $v_1, \cdots, v_n$의 linear combination으로 표현되고, 이에 따라 W와 Q의 column space는 같다. 즉, gram schmidt process는 span을 보존한다.

혹은 단순히 basis로 column을 구성한 $\tilde{B}$를 사용해 구할 수도 있다. $[\tilde{B}\tilde{B}^T|\tilde{B}]$에 gaussian elimination을 적용하여 왼쪽 matrix를 upper triangle matrix로 만들면 왼쪽 matrix의 row는 서로 orthogonal하다. 이후 normalization하면 된다.

7.5.5. Least Square Problem

1. Least Square Problem
Least Sqaure Problem은 평면에서 여러 점들에 대해서 오차를 최소화하는 직선을 찾는 문제이다. projection을 활용해 이를 해결할 수 있다.

이런 최적의 직선을 $f(x)$라고 하자. 각 t에 대해 $f(t) = \hat{y} = C + Dt$라고 할 수 있고, 각 점의 y값과의 차이를 오차값 error로 정의할 수 있다.

$f(t)$를 Ax=b꼴로 나타내보자. 각 점을 대입하면 $C + Dx_1 = y_1, \cdots\ C + Dx_n = y_n$과 같고, C와 D를 변수로 하는 이 연립일차방정식을 다음과 같이 Ax=b꼴로 나타낼 수 있다.

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} C \\ D \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$$

물론 이때 $b \in C(A)$이면 해가 존재하지만 대체로 해가 없는 경우가 많고, 그렇다면 $err = b - Ax$, $Ax + err = b$에 대해서만 해가 존재한다. 즉, $Ax = b$로 나타냈을 때 Ax가 b에 최대한 가깝도록 하는 x를 찾는 문제이다. b를 A의 column의 span으로 생성되는 space에 orthogonal projectioin한 뒤 해를 구하는 것으로도 이해할 수 있다. 이는 C와 D에 대한 수식을 작성한 뒤 편미분해서 error가 최소가 되도록 하는 C와 D 값을 찾거나, projection을 통해 C와 D 값을 찾는 것으로 해결할 수 있다.

C와 D를 찾았으면 당연하게도 실제 예측값과 error를 구할 수 있다.

2. 편미분으로 풀기
각 점의 좌표를 (x_i, y_i)라고 했을 때 err의 각 성분은 $e_i = |f(x_i) - y_i|$이다. 즉 $L = e_1^2 + \cdots + e_n^2$을 최소로 하는 C와 D 값을 편미분해서 찾을 수 있다.

예를 들어, 위의 예시에서는 $C + D + e_1 = 1$, $C + 2D + e_1 = 2$, $C + 3D + e_1 = 2$이고 $L = (C + D - 1)^2 + (C + 2D - 2)^2 + (C + 3D - 2)^2$으로 정리할 수 있다. 이에 대해 편미분해 $\frac{\partial L}{\partial C}$, $\frac{\partial L}{\partial D}$가 0이 되는 C와 D를 찾으면 그게 최적의 직선이다.

3. Projection으로 풀기
Ax=b를 $A^T A\hat{x} = A^T b$로 변형하면 A의 column space로 b를 projection한 문제를 푸는 것으로 바꿀 수 있다. 앞에서 다룬 것처럼 A가 full column rank를 가지면 $A^T A$는 invertible하고, 항상 해가 존재한다. 즉, 단순히 $A^T A\hat{x} = A^T b$에 대해 해를 찾으면 C와 D를 알 수 있다.

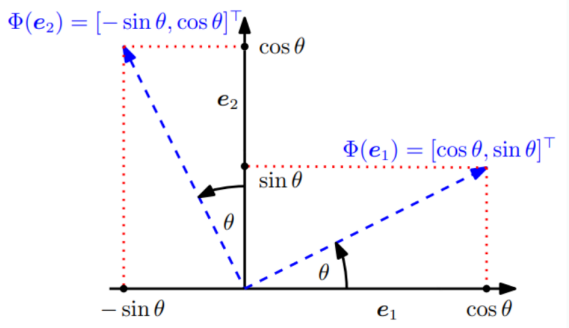
7.6. Rotation

7.6.1. Rotation

Rotation은 원래의 평면을 θ만큼 회전시키는 것을 말하며, 이때의 고정된 point를 **Origin**이라고 한다. 또한 일반적으로 positive angle θ는 반시계 방향을 의미한다. orthognal matrix는 norm과 angle을 보존하고 (orthogonality/orthonormality도 보존), 모든 rotation matrix는 orthogonal matrix이다.

$\mathbb{R}^2$에서의 rotation은 다음과 같이 정의된다. standard basis에 대한 회전을 삼각법으로 계산해 보면 당연하다.

$$\left\{ \boldsymbol{e}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \boldsymbol{e}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}$$



$\mathbb{R}^3$에서의 rotation은 다음과 같이 정의된다. $\mathbb{R}^3$에서의 rotation은 평면을 고정된 1차원 축에 대해 회전시킨다. 이때에도 positive angle은 반시계 방향인데, 축의 끝에서 원점을 바라봤을 때의 반시계 방향이다.

$$\boldsymbol{R}_1(\theta) = \begin{bmatrix} \Phi(\boldsymbol{e}_1) & \Phi(\boldsymbol{e}_2) & \Phi(\boldsymbol{e}_3) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{3.77}$$

- $\boldsymbol{e}_2$ 에 대한 rotation matrix는 다음과 같습니다.

$$\boldsymbol{R}_2(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{3.78}$$

- $\boldsymbol{e}_3$ 에 대한 rotation matrix는 다음과 같습니다.

$$\boldsymbol{R}_2(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.79}$$

n-dimension euclidean vector space에서의 rotation은 n-2개의 dimension은 고정하고, n-dimension에서 2-dimension 평면을 회전하는 것이다. 이때 $1 \leq i < j \leq n$에 대해서 i번째 dimension과 j번째 dimension으로 형성되는 평면에 대한 rotation matrix는 다음과 같다. 이때의 $R_{ij}(\theta)$를 Givens Rotation이라고 한다.

$$R_{ij}(\theta) := \begin{bmatrix} I_{i-1} & 0 & \cdots & 0 & 0 \\ 0 & \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 0 & I_{j-i-1} & 0 & 0 \\ 0 & \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & \cdots & 0 & I_{n-j} \end{bmatrix} \in \mathbb{R}^{n \times n}$$

이때 $R_{ij}(\theta)$는 identity matrix와 유사한데, 다음과 같은 원소만 다르다.

$$r_{ii} = \cos\theta, \quad r_{ij} = -\sin\theta, \quad r_{ji} = \sin\theta, \quad r_{jj} = \cos\theta$$

참고로, 3차원 이상에서는 commutative(교환법칙)가 성립하지 않으므로, rotation을 적용하는 순서가 중요하다.

8. Determinant & Trace

본 장에서는 3가지 행렬의 측면 how to summarize matrices, how matrices can be decomposed, how these decompositions can be used for matrix approximations에 대해 살펴본다.

8.1. Determinant

8.1.1. Determinant

1. Determinant
$n \times n$ matrix A의 **Determinant(행렬식)**는 A를 real number로 mapping하는 함수이다. $det(A)$ 또는 $|A|$로 표기하고, 다음과 같이 계산한다.

- A가 1×1 matrix인 경우, $det(A) = A_{11}$이다.
- A가 2×2 matrix인 경우, $det(A) = A_{11}A_{22} - A_{12}A_{21}$이다. $(ad - bc)$
- A가 $n > 2$인 n에 대해서 $n \times n$ matrix인 경우, determinant는 임의의 row 또는 column에 대한 cofactor formula로 구할 수 있다.

  모든 i에 대해서 row i에 대한 cofactor formula를 하면 determinant는 아래와 같다.

$$det(A) = \sum_{j=1}^{n} (-1)^{i+j} A_{ij} det(M_{ij})$$

  또는 모든 j에 대해서 row j에 대한 cofactor formula를 하면 determinant는 아래와 같다.

determinant가 0인 $n \times n$ matrix는 singular/non-invertible matrix이고, rank가 n이다. determinant는 singularity를 결정하므로 그 이름이 붙었다.

2. Cofactor
$n \times n$ matrix A를 생각하자. scalar $(-1)^{i+j} det(M_{ij})$는 A의 i행 j열 성분에 대한 **Cofactor(여인수)**라 한다.

cofactor를 $c_{ij} = (-1)^{i+j} det(M_{ij})$로 표기했을 때, determinant를 다음과 같이 cofactor들의 linear combination으로 나타낼 수 있다. 이를 i번째 row에 대한 **Cofactor Formula(Cofactor Expansion, 여인수 전개)** 또는 Laplace expansion(라플라스 전개)라고 한다.

$$det(A) = A_{i1}c_{i1} + A_{i2}c_{i2} + \cdots + A_{in}c_{in}$$

A의 i번째 row와 j번째 column을 지워서 얻은 $(n-1) \times (n-1)$ matrix를 A의 (i, j) **Minor(소행렬)**라고 하고, M_{ij}로 표기한다.

determinant는 그 정의상 square matrix에 대해서만 구한다.

8.1.2. Determinant의 성질

determinant는 다음과 같은 성질을 가진다. 이때 1번부터 3번까지의 성질로 나머지 성질들을 유도할 수 있다.

1. identity matrix I에 대해 $\det I = 1$이다.
2. matrix A의 두 row의 위치를 바꾼 matrix B에 대해서 $\det B = -\det A$가 성립한다.

   즉, 짝수 번 만큼 row를 교환하면 부호가 그대로, 홀수 번 만큼 row를 교환하면 부호가 바뀐다.
3. a. matrix A의 한 row에 scalar c를 곱한 matrix B에 대해 $\det B = c \det A$가 성립한다.

   이에 따라 A가 $n \times n$ matrix인 경우 scalar c에 대해 $\det cA = c^n \det A$이다.

   b. matrix A의 determinant는 한 row의 값에 따라 분할한 뒤 각각에 대한 determinant를 계산한 것과 같다. 즉, 다음이 성립한다.

$$\begin{bmatrix} a'+a & b'+b \\ c & d \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} a' & b' \\ c & d \end{bmatrix}$$

4. matrix A가 동일한 row를 가지면, $\det A = 0$이다.

   row를 바꿔서 determinant의 부호가 바뀌어도 동일하므로 당연하다. 이때 동일한 row가 있다는 것은 full rank가 아니라는 것이고, 이에 따라 non-invertible이게 된다.

5. 어떤 row의 scalar배를 다른 row에 더하는 row elementary operation은 determinant를 보존한다(바꾸지 않는다.). 즉, elemination을 해도 determinant가 보존된다.

   해당 operation이 적용된 matrix는 다음과 같이 3번 성질을 사용해 두 matrix로 나눌 수 있고, scalar배 한 것은 matrix 밖으로 뺄 수 있다. 이렇게 정리해 보면 두 row가 같으므로 determinant가 0이 된다.

$$\begin{bmatrix} a & b \\ c-la & d-lb \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} - l \begin{bmatrix} a & b \\ a & b \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

   이전에 배운 걸 생각해 봐도 만약 determinant가 보존되지 않았다면 elemination으로 invertible 여부를 판단할 수 없었을 것이고, complete soluation을 구할 수 없었을 것이다. 또한 이 성질을 이용해서 operation을 적용해 보면 2×2 matrix의 determinant가 왜 ad-bc로 계산되는지도 쉽게 증명할 수 있다.

   elementary operation의 종류에 따른 특성을 고려해 elimination을 적절히 적용하여 triangle matrix로 변환하면 determinant를 쉽게 구할 수 있다.

6. A가 0으로 이루어진 row를 가지면 A의 determinant는 0이다.

   full rank가 아니므로 당연하고, 해당 row에 scalar 0이 곱해져 있는 것으로 봐도 당연하다. 또한 cofactor formula를 해 봐도 당연하다.

7. upper diagonal matrix의 determinant는 대각성분의 곱과 같다. 또한 마찬가지로 lower diagonal matrix의 determinant도 대각성분의 곱과 같다.

   한 row의 scalar배를 다른 row에 더하는 row elementary operation을 적용해 diagonal matrix로 변형하고, 3번 성질에 의해 각 row의 scalar를 앞으로 빼 보면 당연하다.

8. $\det A = 0$인 것과 A가 singular인 것, 그리고 $\det A \neq 0$인 것과 A가 non-singular인 것은 필요충분이다.

9. $\det AB = \det A \cdot \det B$이다.

   A가 invertible이면 elementary matrix의 곱으로 나타낼 수 있고, elementary matrix와의 곱에 대해서는 $\det EA = \det E \det A$가 성립하는 게 자명하므로(operation 종류별로 증명 가능하다.) 증명할 수 있다. A가 invertible이 아니면 AB도 invertible이 아니므로 성립한다.

   이에 따라 A가 invertible인 경우 $\det A^{-1} = \frac{1}{\det A}$이다. 또한 $\det A^n = (\det A)^n$이다.

10. $\det A = \det A^T$이다. 즉, 앞서 다룬 성질들은 row에 대한 것이었는데, 이는 column에 대해서도 성립한다.

    A에 대해 LU분해를 해서 보면, $\det U^T \det L^T = \det L \det U$임을 보이면 되므로 당연하다.

11. similar matrix끼리는 determinant가 같다. 즉, linear mapping에 대한 basis change에도 determinant는 바뀌지 않는다.

1번부터 3번까지의 증명은 다루지 않는다.

이런 determinant의 성질을 이용해 cofactor formula가 왜 성립하는지 직접 계산해 볼 수 있다. 예를 들어, 3×3 matrix A의 각 원소가 $a_{ij} + 0$이므로 3번 성질을 적용하면, 각 row에 하나의 원소를 제외하고는 모두 0인 matrix들(총 27개)로 분리할 수 있다. 이때 한 column이 모두 0인 matrix은 determinant가 0이므로 row와 column에 하나의 원소만 존재하는 matrix들에 대한 determinant를 계산해 더하면 된다. 즉, 아래와 같은 matrix들에 대한 determinant를 더하면 된다.

$$\begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix}, \begin{bmatrix} a_{11} & 0 & 0 \\ 0 & 0 & a_{23} \\ 0 & a_{32} & 0 \end{bmatrix}, \begin{bmatrix} 0 & a_{12} & 0 \\ 0 & 0 & a_{23} \\ a_{31} & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & a_{12} & 0 \\ a_{21} & 0 & 0 \\ 0 & 0 & a_{33} \end{bmatrix}, \begin{bmatrix} 0 & 0 & a_{13} \\ a_{21} & 0 & 0 \\ 0 & a_{32} & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & a_{13} \\ 0 & a_{22} & 0 \\ a_{31} & 0 & 0 \end{bmatrix}$$

이 matrix들은 row를 바꾼 것이므로 0이 아닌 각 성분을 전부 곱한 뒤 부호만 맞춰주면 된다. 즉, $\det A = a_{11}a_{22}a_{33} - a_{11}a_{32}a_{23} + a_{21}a_{12}a_{33} - a_{21}a_{12}a_{33} + a_{21}a_{32}a_{13} - a_{31}a_{22}a_{13} = a_{11}C_{11} + a_{12}C_{12} + a_{13}C_{13}$가 된다.

$n \times n$ matrix에 대해서는 이렇게 나뉜 matrix가 총 $n!$개 이므로 복잡도도 그 $n!$이다. 또한 각 row에서 column이 겹치지 않게 0이 아닌 성분을 하나씩 고르고, diagoanl matrix를 만들기 위해 row를 바꾸는 횟수로 부호를 정하는 것으로도 determinant를 구할 수 있음을 알 수 있다. 즉, 다음과 같은 matrix B의 determinant는 각 row에 대해 column 4, 3, 2, 1 또는 column 3, 2, 1, 4의 성분을 사용하고, 각각 부호는 +, -이므로 determinant가 0이다. 이렇게 invertibility를 판정할 수도 있다.

$$\det B = \det(\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}) = 1 - 1 = 0$$

8.2. Determinant의 활용

8.2.1. Inverse Matrix와 Determinant

1. Cofactor Matrix와 Adjoint Matrix
$n \times n$ matrix A에 대한 Cofactor Matrix C는 다음과 같이 A에 대한 cofactor로 구성된 matrix이다.

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

Adjoint Matrix(수반 행렬)는 cofactor matrix를 transpose한 matrix이다.

$$\text{adj}(A) = C^T$$

2. Inverse Matrix와 Determinant
invertible matrix A에 대한 inverse matrix는 다음과 같이 determinant와 cofactor(adjoint) matrix로도 구할 수 있다.

$$A^{-1} = \frac{1}{\det A}\text{adj}(A) = \frac{1}{\det A}C^T$$

이는 $\det A \cdot I = AC^T$가 성립함을 보여 증명할 수 있다. 대각성분에 대해서는 cofactor formula를 생각해보면 당연하다. 대각성분 외의 성분들에 대해서는 B라는 가상의 matrix를 생각해 증명할 수 있다. B는 A와 동일한 성분을 가지는 $n \times n$ matrix인데, k번째 row만 어떤 i($i \neq k$)번째 row와 성분이 동일한 marix이다. k번째 row에 대해 cofactor formula를 적용해 B의 determinant를 구해 보면 $\det B = \sum_{j=1}^{n} a_{ij}C_{kj}$이다. 그런데 B는 두 row가 같은 matrix이므로 determiant가 0이고, 이에 따라 $\sum_{j=1}^{n} a_{ij}C_{kj} = 0$이다. 즉, 서로 다른 row의 성분과 cofactor를 사용해 계산한 값은 항상 0이 된다.

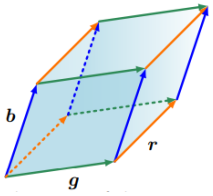
8.2.2. Volume과 Determinant

determinant는 linear transformation에 의한 영역의 변화를 나타내는 factor로도 이해할 수 있다(링크 참고). standard basis를 transform하는 것을 생각해 보면, matrix A의 determinant의 절댓값 $|\det A|$는 A의 column들이(당연히 row여도 가능하다.) 구성하는 도형의 **Volume(부피)**이다.

이는 앞서 다룬 determinant의 여러 성질을 사용해 확인할 수 있다. 우선 두 vector (1,0), (0,1)에 대해서 살

펴보면 $|\det A|$가 이 두 vector가 만드는 정사각형의 volume임은 자명하고, 이 vector 각각을 scalar배 해도 자명하다. 또한 임의의 두 vector (a, b), (c, d)가 만드는 평행사변형의 volume을 구해 봐도 자명하다.

한 column의 scalar배를 다른 column에 더하는 것은 해당 도형의 밑변과 높이를 보존한다. 이는 임의의 좌표에 있는 도형을 평행이동해서 원점으로 이동시키는 계산을 해보면 당연하다. 즉, 한 row의 scalar배를 다른 row에 더하는 것은 volume을 보존하므로 **이 연산만으로 elimination을 적용한 뒤 determinant를 구해 volume을 구할 수 있다.**



참고로 3차원 vector 3개가 이루는 삼각뿔의 부피는, 해당 vector들로 구성된 matrix의 determinant로 volume을 구하고 $\frac{1}{6}$을 곱하면 된다.

8.3. Trace

8.3.1. Trace

$n \times n$ squre matrix A에 대한 **Trace(대각합)**는 다음과 같이 대각성분의 합으로, tr()로 표기한다.

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

trace에 대해서는 다음 성질들이 성립한다.

- $n \times n$ matrix A,B에 대해 $\mathrm{tr}(A+B) = \mathrm{tr}(A) + \mathrm{tr}(B)$
- $n \times n$ matrix A와 scalar α에 대해 $\mathrm{tr}(\alpha A) = \alpha \mathrm{tr}(A)$
- $\mathrm{tr}(I_n) = n$
- $n \times k$ matrix A, $k \times n$ matrix B에 대해 $\mathrm{tr}(AB) = \mathrm{tr}(BA)$

  특히 두 vector에 대해서는 $\mathrm{tr}(xy^T) = \mathrm{tr}(y^T x) = y^T x$가 성립한다.

  similar인 두 matrix에 대해서는 $\mathrm{tr}(B) = \mathrm{tr}(S^{-1}AS) = \mathrm{tr}(ASS^{-1}) = \mathrm{tr}(A)$이므로 trace가 같다.

9. Eigen value and Eigen Vector

9.1. Eigen value and Eigen Vector

앞서 정리한 것처럼, 모든 linear mapping은 ordered basis로 정의되는 유일한 transformation matrix를 가진다. eigen value/vector를 활용한 eigen analysis는 linear mapping의 변환 동작을 이해할 수 있다.

9.1.1. Eigen value and Eigen Vector

1. Eigen value and Eigen Vector
어떤 $n \times n$ matrix A에 대해서 다음 수식(**Eigen Equation**)을 만족시키는 scalar λ가 존재하는 영벡터가 아닌 vector v를 A의 **Eigen Vector(고유벡터)**라고 하고, 이때의 scalar λ를 eigen vector v에 대응되는 A의 **Eigen Value(고윳값)**라 한다. 즉, eigen vector는 A를 곱해도 그 방향이 달라지지 않는 특별한 vector이다.

$$Av = \lambda v$$

어떤 eigen value에 대해 존재하는 모든 eigen vector들과 영벡터에 대한 집합을 **Eigen Space(고유공간)**라고 한다. 이는 아래에서 다루는 것처럼 $A - \lambda I$의 null space이다. 또한 A의 모든 eigen value의 집합을 Eigen Spectrum 또는 **Spectrum**이라고 한다.

참고로, $n \times n$ matrix A가 diagonalizable한 것과 A의 eigen space의 direct sum(직합. 합쳐서 전체 space를 구성하고, 교집합은 영벡터만 존재.)이 R^n인 것은 필요충분이다. 즉, 어떤 $n \times n$ diagonalizable matrix A에 대한 R^n(정의역)은 A의 eigen space들로 나누어 생각할 수 있고, 어떤 eigen value에 대한 eigen space에 속하는지에 따라 해당 vector가 어떻게 R^n으로(치역) 보내지는지를 알 수 있다. 더 구체적으로는, 임의의 vector x가 R^n의 eigen basis $v_1, \cdots, v_n$으로 표현될 때, 다음 수식과 같이 eigen value만큼씩이 각 eigen vector에 곱해진다.

$$Ax = A(a_1v_1 + a_2v_2 + \cdots + a_nv_n) = \lambda_1a_1v_1 + \lambda_2a_2v_2 + \cdots + \lambda_na_nv_n$$

기하학적으로, eigen vector는 해당 linear mapping이 변환(또는 확장)하는 방향을 나타내고, eigen value는 얼마만큼 변환하는지 나타내는 factor로 이해할 수 있다. eigen value가 음수이면 반대 방향으로 변환하는 것이다. 이때 eigen vector는 영벡터가 아닌 vector지만 eigen value는 0일 수 있고, eigen value가 0인 eigen space는 null space이다.

2. Eigen value/vector 구하기

$Av = \lambda v$를 정리하면 $(A - \lambda I)v = 0$이므로 eigen vector v는 $A - \lambda I$의 null space에 존재하는 vector이고, $A - \lambda I$의 special solution을 구하면 그게 eigen vector이다.

이때 eigen vector는 영벡터가 아니므로, eigen vector가 존재하려면 null space가 영공간이 아니어야 하고, 다시 말해 $\det(A - \lambda I) = 0$가 성립해야 한다. 이에 따라 $\det(A - tI) = 0$을 만족시키는 t를 찾으면 그게 λ이다. 이때 matrix A에 대해서 다음 다항식을 A의 **Characteristic Polynomial(특성다항식)**이라 하고, $det(A - tI) = 0$을 **Characteristic Equation(특성방정식)**이라 한다.

$$f(t) = det(A - tI)$$

characteristic polynomial은 다음과 같이 정의할 수도 있다. 이때 $c_0 = \det(A)$이고, $c_{n-1} = (-1)^{n-1}\mathrm{tr}(A)$

$$\begin{aligned} f(t) &= det(A - tI) \\ &= c_0 + c_1\lambda + c_2\lambda^2 + \cdots + c_{n-1}\lambda^{n-1} + (-1)^n\lambda^n \end{aligned}$$

정리하면, 다음과 같은 과정을 거쳐 eigen value/vector를 구할 수 있다.

1. characteristic equation을 만족시키는 scalar를 구하면 그게 eigen value이다.
2. eigen value를 $A - \lambda I$에 대입해 얻은 matrix의 null space의 vector를 구하면 그게 eigen vector이다.

   이는 homogeneous equation $(A - \lambda I)x = 0$에 대한 special solution을 구하는 것으로도 이해할 수 있다. 즉, $A - \lambda I$에 elimination을 적용한 뒤 free variable 중 하나에만 1, 나머지에는 0을 대입해 해당 null space(eigen space)의 basis를 찾아 구할 수 있다.

3. 대수적 중복도와 기하적 중복도

characteristic equation이 $f(t)$인 선형연산자(또는 행렬)의 고윳값 λ에 대해서, $(t-\lambda)^k$가 $f(t)$의 인수가 되도록 하는 가장 큰 자연수 k를 λ의 Multiplicity(중복도) 또는 **Algebraic Multiplicity(대수적 중복도)**라고 한다. 즉, 인수분해했을 때 해당 λ의 차수를 의미한다. diagonalization했을 때 diagonal matrix Λ에 각 λ는 algebraic multiplicity만큼 나타난다.

어떤 λ에 대한 eigen space의 dimension을 해당 λ에 대한 **Geometric Multiplicity(기하적 중복도)**라고 한다. 즉, geometric multiplicity는 eigen space에서 independant한 eigen vector들의 개수로, $A - \lambda I$의 nullity이다.

$n \times n$ matrix A의 모든 eigen value의 algebraic multiplicity를 합하면 n이다. 또한 임의의 eigen value에 대해 geometric multiplicity는 algebraic multiplicity보다 작고, 각 eigen value에 대응되는 eigen vector들의 집합이

각각 independent하면 이들을 합집합해도 independent하다(λ가 다른 두 eigen vector는 서로 independent하다.). 즉, **모든 eigen value에 대해 algebraic multiplicity만큼의 원소를 가지는 eigen space의 basis가 존재하면 diagonalization이 가능하고, 이런 basis를 모두 합치면 R^n에 대한 basis이다.** geometric multiplicity는 항상 1보다 크거나 같으므로, $n \times n$ matrix에 대해 n개의 서로 다른 eigen value가 존재한다면 해당 matirx는 diagoanlization이 가능하다.

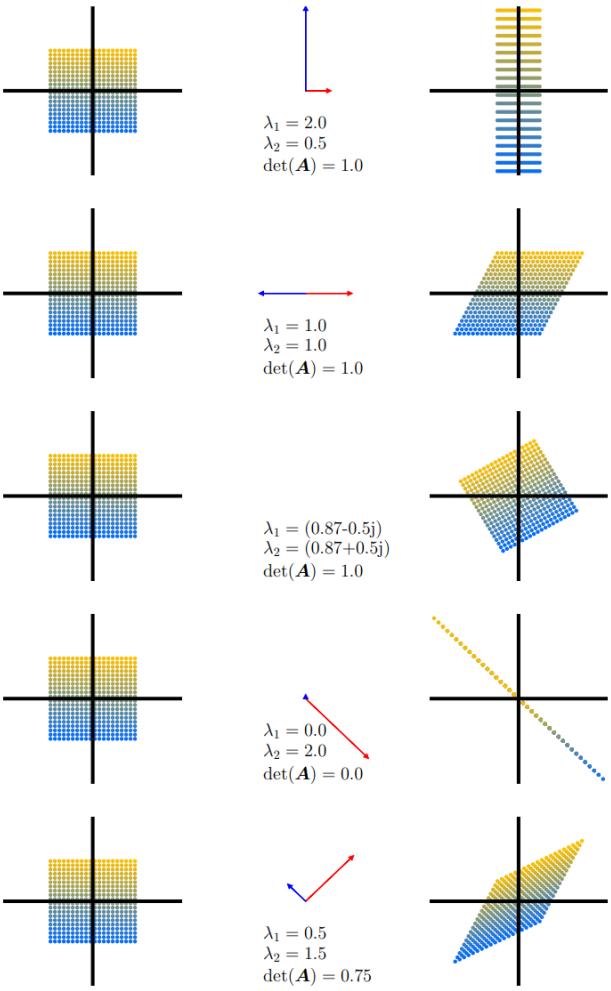
어떤 matrix가 eigen value에 대해 algebraic multiplicity보다 geometric multiplicity가 작은 경우가 있다면 (diagonalization이 불가능하다면) Defective하다고 한다.

서로 다른 eigen value들에 대한 eigen vector들은 서로 independent한데, 이는 쉽게 증명할 수 있다. $Av_1 = \lambda_1 v_1, Av_2 = \lambda_2 v_2$이고 $\lambda_1 \neq \lambda_2$일 때 scalar a에 대해 $v_1 = av_2$라고 가정하자. $aAv_2 = a\lambda_1 v_2, Av_2 = \lambda_1 v_2 = \lambda_2 v_2$이므로 $\lambda_1 = \lambda_2$으로 모순이다.

projection matrix P를 곱해 b를 어떤 공간에 projection한 Pb를 생각하자. 당연하게도 이때 b가 해당 공간에 원래 있었다면 b는 eigen vector이고, 해당 공간에 원래 있지 않았다면 b는 eigen vector가 아니다.

eigen vector는 그 정의상 영벡터가 아니다. 만약 eigen vector가 영벡터일 수 있다면 eigen value가 무한히 존재하게 된다.

다음 예시를 보면 eigen vector, eigen value, determinant에 대해 직관적으로 이해할 수 있다.



9.1.2. Eigen Value/Vector 관련 성질들

eigen value/vector 관련 성질들은 다음과 같다.

- 실수에서 정의된 $n \times n$ matrix A가 symmetric이면 A는 항상 실수 eigen value를 가지고, A의 eigen vector로 구성된 orthonormal basis(ONB)가 존재한다. 즉, A는 diagonalization이 가능하고, eigen basis를 구했으면 각 eigen space에 속하는 basis에 대해 gram schmidt process 및 normalization을 적용해 ONB를 얻을 수 있다. 이를 Spectral Theorm이라고 한다.

  symmetric matrix에 대해서는 서로 다른 eigen space에 속한 eigen vector들이 항성 서로 orthogoanl하므로, gram schmidt process를 동일한 eigen space에 속한 basis들에게만 적용하면 된다.

- matrix A의 eigen value가 λ, eigen vector가 v일 때 모든 0이 아닌 scalar에 대해 cx도 A의 eigen vector이고, x와 eigen value가 같다. 즉, x에 대해 colinear한 vector들 또한 eigen vector이다.

  두 vector가 같은 방향을 가리키는 경우 Codirected하다고 하고, 같은 방향 또는 정반대 방향인 경우 Colinear하다고 한다.

- similar한 matrix는 서로 eigen value가 같다.

  즉, determinant, trace, eigen value는 basis에 독립적이며, basis change를 적용해도 변하지 않는다. 이에 따라 linear mapping의 핵심 특성으로 활용될 수 있다.

- matrix A의 eigen value의 합은 A의 trace(대각합, 대각성분의 합.)와 같고, eigen value의 곱은 determinant와 같다.

- matrix A의 대각성분에 scalar a만큼의 값을 더하는 경우, eigen vector는 유지되고 eigen value는 a만큼 더한 값이 된다. 즉, 대각성분만 다른 경우 eigen value/vector는 한 번만 구해도 된다.

  이는 당연하게도 $Bx = (A + aI)x = Ax + ax = \lambda x + ax = (\lambda + a)x$인 것으로 보일 수 있다.

- matrix A에 대해, A^n의 eigen value는 λ^n이고 eigen vector는 A와 같다.

  x가 A의 eigen vector라고 하면 λ에 대해 $A^{100}x = \lambda^{100}x$이므로 당연하다.

- invertible matrix A에 대해, A^{-1}의 eigen value는 $\frac{1}{\lambda}$이고, eigen vector는 A와 같다.

  x가 A의 eigen vector라고 하면 $Ax = \lambda x$이므로 양번에 A^{-1}를 곱하면 $A^{-1}x = \frac{1}{\lambda}x$으로 정리할 수 있다.

- upper/lower trianlge matrix의 eigen value는 대각성분과 같다.

  matrix A가 upper/lower triangle matrix이면 $A - \lambda I$도 upper/lower triangle matrix이므로 determinant가 대각성분의 곱이 되고, 이에 따라 당연하게도 대각성분이 eigen value가 된다.

- A와 A^T는 동일한 eigen value를 가지지만, 같은 eigen vector를 가지는 것은 아니다.

- elimination은 eigen value를 보존하지 않는다.

  아무 matrix나 elimination해서 간단히 보일 수 있다.

예를 들어, 다음과 같은 2×2 matrix A의 characteristic equation은 $(3 - \lambda)^2 - 1 = \lambda^2 - 6\lambda + 8 = 0$이므로, eigen value의 합이 trace, 곱이 determinant인 것을 확인할 수 있다.

$$A = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$$

A와 B가 각각 $Ax = \lambda x$, $By = \alpha y$라고 하자. 이 경우 $C = A + B$의 eigen vector가 x 또는 y라는 것은 당연하게도 거짓이다. 실제로 대부분의 경우 eigen vector가 겹치지 않는다고 한다.

다음과 같은 matrix Q를 생각하자. 이는 vector를 90도 돌리는 rotation matrix이다. 기하적으로 생각했을 때 이런 rotation matrix는 vector를 돌리므로(방향이 바뀌므로) 실수인 eigen value가 존재할 수 없고, eigen vector는 항상 복소수이다.

$$Q = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

참고로, 원점을 기준으로 반시계 뱡향으로 θ만큼 rotation을 적용하는 rotation matrix는 다음과 같이 정의된다.

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

9.2. Diagonalization

diagonal matrix는 거듭제곱, determinant 계산, inverse matrix 계산 등에서 매우 간편하다.

9.2.1. Diagonalization

1. Diagonalization

Diagonalization(대각화)은 어떤 $n \times n$ matrix의 eigen value와 eigen vector를 사용해 diagonal matrix로 변환하는 방법이다. 서로 independent한 A의 eigen vector n개로 column을 구성한 matrix S와, S의 eigen vector의 순서와 대응되는 eigen value를 대각성분으로 가지는 diagonal matrix Λ를 생각하자. 다음 수식이 성립한다. 즉, eigen vector/value를 사용해 diagonalization이 가능하다. 또한 이렇게 A를 나누는 것을 **Eigen Decomposition** 또는 Eigen Value Decomposition이라고 한다.

$$AS = S\Lambda, \quad S^{-1}AS = \Lambda, \quad A = S\Lambda S^{-1}$$

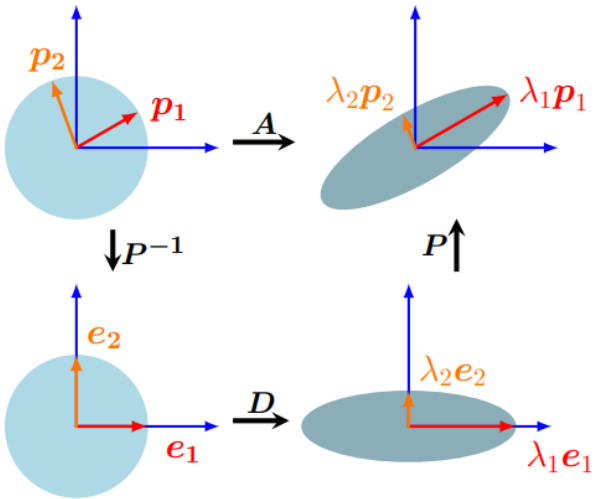
이 수식은 다음과 같이 계산되는 것을 생각해면 당연하다.

$$AS = \begin{bmatrix} \lambda_1 x_1 & \lambda_2 x_2 & \cdots & \lambda_n x_n \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} = S\Lambda$$

$S^{-1}AS = \Lambda$는 S가 invertible이어야 성립한다. 즉, eigen basis가 존재해야 정의될 수 있다.

이때 eigen basis가 ONB를 형성한다면 inverse matrix를 transpose로 간단히 구할 수 있다.

eigen decomposition에 의한 $A = S\Lambda S^{-1}$는 다음 그림과 같이 기하학적으로 이해할 수 있다. 즉, standard basis를 eigen basis로 basis change하고, 이후 eigen basis 위에서 eigen value로 scaling한 뒤, 다시 원래의 standard basis로 돌리는 과정을 통해 matrix A의 연산이 수행된다.



2. Diagonalizable 판정법

어떤 matrix A가 diagonal matrix와 similar하다면 Diagonalizable하다고 한다. 즉, diagonalization이 가능한 것을 말한다.

다음 조건을 만족할 때만 diagonalizable하다. 즉, characteristic equation을 인수분해하고, 중근을 가지는 eigen value에 대해 대수적 중복도와 기하적 중복도가 같은지 확인해야 한다.

1. T의 특성다항식이 체 위에서 완전히 인수분해됨. 실수체인 경우 실수 범위에서 인수분해되어야 한다.
2. T의 각 고윳값에 대해서 대수적 중복도와 기하적 중복도가 같다. 즉, λ의 중복도가 $nullity(T-\lambda I) = n - rank(T-\lambda I)$이다.

앞서 다룬 것처럼 모든 eigen value에 대해 대수적 중복도만큼의 원소를 가지는 eigen space의 basis가 존재하면 diagonalization이 가능하고, 이런 basis를 모두 합치면 R^n에 대한 basis이다. 이는 eigen vector로 R^n에 대한 basis를 구성할 수 있는지를 보는 것으로도 이해할 수 있다.

$S\Lambda S^{-1}$를 거듭제곱해보면 $A^n = S\Lambda^n S^{-1}$인데, 전에 다룬 것처럼 A를 거듭제곱하면 eigen vector는 그대로이고, eigen value는 동일하게 거듭제곱이 되는 것을 확인할 수 있다.

$\det A = \det(S\Lambda S^{-1}) = \det \Lambda$이므로 determinant도 쉽게 구할 수 있다.

만약 A^k에 대해서 $k \to \infty$이면 모든 eigen value가 $|\lambda_i| < 1$인 경우 $A^k \to 0$이다. 이런 성질을 이용하면 마르코프 체인 등에서 확률이 어디에 수렵하는지 알 수 있다고 한다.

9.2.2. Symmetric Matrix

Symmetric Matrix(대칭행렬)는 $A = A^T$인 matrix이다. 원소가 실수인 $n \times n$ symmetric matrix는 다음과 같은 성질을 보인다.

- eigen value가 실수이다. 즉, characteristic equation이 n개의 근을 가진다.

  $Ax = \lambda x$가 성립할 때, 양변에 conjugate(켤레)를 취하면 $\bar{A}\bar{x} = A\bar{x} = \bar{\lambda}\bar{x}$이다. transpose하고 x를 곱해서 정리해 보면 $\lambda \bar{x}^T x = \bar{\lambda}\bar{x}^T x$이므로 $\lambda = \bar{\lambda}$이다.

  물론 이때 $\bar{x}^T x \neq 0$임을 보여야 완벽하다. 해당 inner product 값을 정리하면 다음과 같다. 이때 켤레복소수와의 곱은 $\bar{x}_i x_i = (a+bi)(a-bi) = a^2 + b^2$이므로 x가 영벡터인 경우에만 $\bar{x}x = 0$가 성립하는데, eigen vector는 영벡터가 아니므로 $\bar{x}_1 x_1 \neq 0$이다.

  $$\bar{x}^T x = \bar{x}_1 x_1 + \bar{x}_2 x_2 + \cdots + \bar{x}_n x_n$$

  참고로 A의 원소가 복소수이면 해당 조건이 성립하지 않는다.

- 항상 diagonalizable하다. 즉, 항상 ONB가 존재한다.

  이는 schur decomposition이라는 방법으로 증명이 가능하다고 한다.

- symmetric matrix에 대해서 eigen vector들로 구성한 R^n의 orthogonal basis를 항상 찾을 수 있다. 즉, 각 column을 eigen vector들로 구성한 basis로 하는 matrix를 Q라 하면, $Q^T AQ = \Lambda$가 성립한다.

  하나의 eigen space에서는 gram schmidt process를 사용하여 orthogonal한 basis를 얻을 수 있다. 또한 symmetric matrix에 대해서는 서로 다른 eigen space의 basis끼리 항상 orthogonal하다. 이는 간단히 증명할 수 있다. 서로 다른 두 eigen value λ_1, λ_2에 대해 $Av_1 = \lambda_1 v_1$, $Av_2 = \lambda_2 v_2$라 했을 때, $\lambda_1 v_1^T v_2 = (Av_1)^T v_2 = v_1^T A v_2 = \lambda_2 v_1^T v_2$ 이므로 $(\lambda_1 - \lambda_2) v_1^T v_2$이고, $\lambda_1 \neq \lambda_2$이므로 $v_1^T v_2 = 0$이다.

- symmetric matrix A의 pivot의 부호 개수는 eigen value의 부호 개수와 같다. 즉, 양수의 개수, 음수의 개수, 0의 개수가 모두 같다. 또한 이에 따라 determinant가 양수/음수인지에 따라 eigen value에서 양수/음수의 개수를 알 수 있다.

  symmtric matrix A는 elimination을 적용해 $A = LDL^T$ 꼴로 decompostion할 수 있다. 이때 D는 대각성분이 pivot인 diagonal matrix이고, A와 D는 congruent(합동)이다. Sylvester's Law of Inertia(실버스터의 관성 법칙)에 의하면 congruent한 matrix끼리는 eigen value의 부호 개수가 같은데, D는 이미 diagonal matrix이므로 A의 pivot과 eigen value는 부호 개수가 같다.

  이에 따라 symmetric matrix에 대해서는 positive definite인지를 판별할 때 eigen value를 직접 구하는 대신, elimination해서 pivot의 부호만 보면 된다.

10. Matrix Decompositions

Decomposition(분해) 또는 Factorization(인수분해)는 하나의 matrix를 더 간단하거나 특정 성질의 matrix들로 나누는 기법이다.

CR, LU, spectral decomposition은 교재에서 다루지 않지만, 함께 정리했다.

10.1. SVD

eigen decomposition은 매우 편리하지만, diagonalizable한 square matrix에 대해서만 적용 가능하다. SVD는 임의의 $m \times n$ matrix에 대해 적용 가능하고, 두 vector space의 기하학적 변화를 나타내므로 의미가 있다. 이에 따라 SVD를 fundamental theorem of linear algrebra라고도 한다.

10.1.1. SVD

1. SVD

SVD(Singular Value Decomposition)는 임의의 $m \times n$ matrix A를 $A = U\Sigma V^T$ 꼴로 나누는 decomposition이다. 즉, eigen decomposition을 임의의 matrix에 대해 확장한 것이다.

$$AV = U\Sigma, \;\; A = U\Sigma V^T$$

matrix A는 R^n 중 row space에서 R^m의 column space로 vector를 보낸다. 이에 따라 U, V, Σ에 대한 정의는 다음과 같다. 이때 $r = rank(A) = rank(A^TA) = rank(AA^T)$가 성립하고, r은 0이 아닌 singular value의 개수와 같다.

- A^TA의 eigen vector로 구성한 R^n의 orthonormal eigen basis $v_1, \cdots, v_n$를 생각하자. V는 이 vector들로 column을 구성한 $n \times n$ orthogonal matrix이다. 이때 orthonormal eigen basis $v_1, \cdots, v_n$를 right singular vector라고 한다.

  이때 $v_1, \cdots, v_r$은 row space의 basis이고, $v_{r+1}, \cdots, v_n$은 null space의 basis이다.

- AA^T의 eigen vector로 구성한 R^m의 orthonormal eigen basis $u_1, \cdots, u_m$을 생각하자. U는 이 vector들로 column을 구성한 $m \times m$ orthogonal matrix이다. 이때 orthonormal eigen basis $u_1, \cdots, u_m$을 left singular vector라고 한다.

  이때 $u_1, \cdots, u_r$은 column space의 basis이고, $u_{r+1}, \cdots, u_m$은 left null space의 basis이다.

- Σ는 대각성분이 Singular Value $\sigma_1, \cdots, \sigma_n$로 구성된 $m \times n$ diagonal matrix로, **Singular Value Matrix**라고 한다. 이때 A^TA의 eigen value를 λ라 하면 **Singular Value**는 다음과 같이 정의된다. 이에 따라 r까지는 singular value가 양수이고, r+1부터는 singular value가 0이다. 이때 A^TA와 AA^T는 서로 transpose 관계이므로 eigen value는 같다.

  $$\sigma_i^2 = \lambda_i. \;\; \sigma = \sqrt{\lambda_i}$$

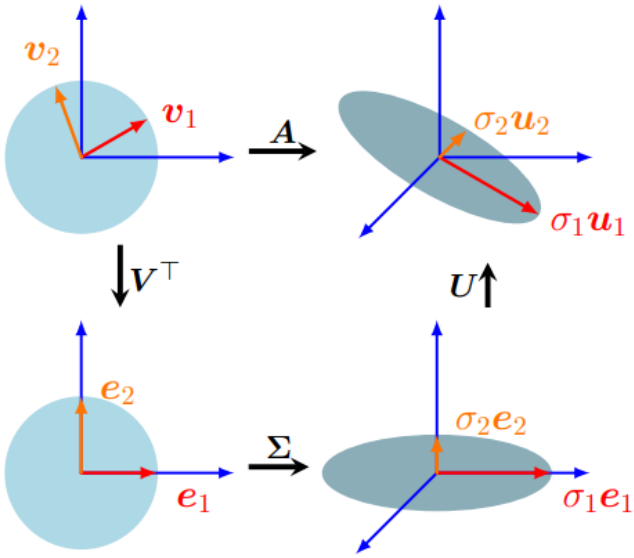
  이때 $i = 1, \cdots, r$에 대해서 $Av_i = \sigma_i u_i$가 성립한다. 즉, singular value는 R^n의 orthonormal eigen basis를 R^m의 orthonormal eigen basis로 보냈을 때 값을 scale하는(곱해서 맞추는) scalar이다. basis끼리 매핑이 존재하는 경우에는 singular value가 양의 실수값을 가지게 되고, 매핑이 존재하지 않는 경우(null space의 basis)에는 singular value가 0이 된다. 이는 앞서 다뤘던 four fundamental spaces에서 이해할 수 있다.

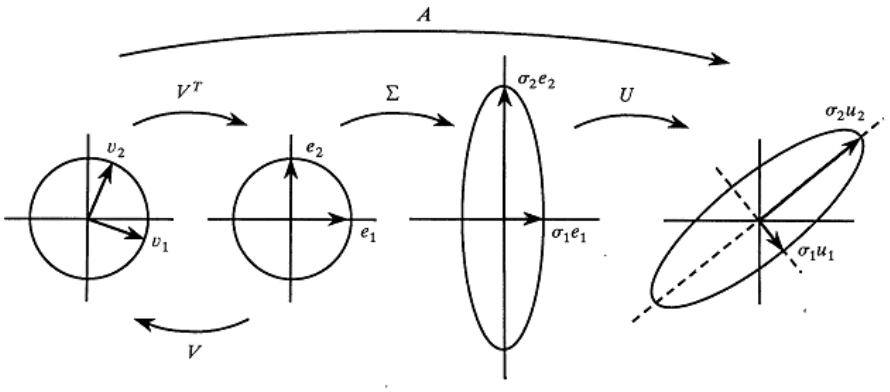
  Σ는 유일하고 A와 shape이 같다.

  SVD에서는 관례적으로 singular value를 내림차순으로 정렬해 나타낸다.

이때 U와 V는 orthogonal matrix이고, Σ는 scaling만을 수행하므로, $Ax = U\Sigma V^Tx$는 x가 V^T에 의해 회전(또는 reflection)되고, Σ에 의해 scaling되고, U에 의해 다시 회전(또는 reflection)되는 것으로 이해할 수 있다. V^T는 standard basis에서 $\{v_1, \ldots, v_n\}$으로의 basis change를 수행하고, U는 $\{u_1, \ldots, u_m\}$에서 standard basis로의 basis change를 수행한다. 즉, eigen decomposition에서와 같이 standard basis에서 orthonormal eigen

basis의 coordinate로 변환한 뒤, Σ로 scaling하며 dimension augmentation(or reduction)을 통해 반대편의 orthonormal eigen basis로 mapping하고, 이후 다시 standard basis의 coordinate로 변환하는 것으로 이해할 수 있다. eigen decomposition이 동일한 eigen basis를 사용했다면, SVD는 서로 다른 두 orthonormal eigen basis를 사용한다.





$Av_i = \sigma_i u_i$가 성립하므로, 앞서 다룬 수식이 다음과 같이 성립함을 이해할 수 있다.

$$A\begin{bmatrix} v_1 & v_2 & \cdots & v_r & \cdots & v_n \end{bmatrix} = \begin{bmatrix} u_1 & u_2 & \cdots & u_r & \cdots & u_m \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & & \cdots & & 0 \\ 0 & \sigma_2 & & \cdots & & 0 \\ & & \ddots & & & \\ \vdots & \vdots & & \sigma_r & & \vdots \\ & & & & \ddots & \\ 0 & 0 & & \cdots & & 0 \end{bmatrix}$$

2. SVD 증명

singular value는 A^TA, AA^T의 eigen value의 제곱근과 같다. 다음과 같이 SVD가 임의의 matrix에 대해 유효함을 보일 수 있다.

앞서 다룬 것처럼 임의의 $m \times n$ matrix A에 대해 A^TA를 생각하자. 이 matrix는 positive semi-definite이므로, $\lambda_i \geq 0$가 성립하여 singular value $\sigma_i = \sqrt{\lambda_i}$를 항상 정의할 수 있다. 또한 A^TA가 symmetric matrix이므로 A^TA의 eigen vector로 구성한 orthonormal eigen basis $v_1, \cdots, v_n$이 항상 존재한다.

A^TA의 eigen value를 사용해, singular value를 $\sigma = \sqrt{\lambda_i}$으로 정의하자. singular value는 r개의 양수와 $n-r$개의 0으로 구성된다.

이제 $\sigma_i u_i = Av_i$를 만족하는 orthonormal eigen basis $u_1, \cdots, u_m$이 존재함을 보이자. R^n의 basis 중 null space의 basis가 아닌 r개의 vector에 대해서는 $\sigma \neq 0$이고, $u_i = \frac{Av_i}{\sigma_i}$로 정리할 수 있다. 이때 $v_1, \cdots, v_n$은 orthonormal하므로 $u_i^T u_j = \frac{1}{\sigma_i \sigma_j} v_i^T A^T A v_j = \frac{\lambda_j}{\sigma_i \sigma_j} v_i^T v_j = 0$, $u_i^T u_i = \frac{1}{\sigma_i^2} v_i^T A^T A v_i = \frac{\lambda_i}{\sigma_i^2} v_i^T v_i = 1$이다. 즉, $u_1, \cdots, u_r$는 orthonormal하다. 또한 $AA^T u_i = \frac{AA^T A v_i}{\sigma_i} = \lambda_i \frac{Av_i}{\sigma_i} = \lambda_i u_i$이므로 AA^T의 eigen vector이다. 또한 A^TA와 AA^T의 0이 아닌 eigen value는 같다.

$u_1, \cdots, u_r$이 orthonormal eigen vector들이므로, m-r개의 orthogonal한 vector들을 gram schmidt process 등을 이용해 구하면 R^m의 orthonormal eigen basis를 찾을 수 있다.

3. SVD 값 구하기
SVD를 실제로 수행하기 위해 U, V, Σ를 구하는 방법은 다음과 같다. 이때 V와 U를 각각 A^TA와 AA^T에 대한 eigen vector를 구하는 것으로 구성할 수도 있지만, 이렇게 구하면 부호가 맞지 않는 경우가 존재하므로 둘 중에 하나에 대해 구한 뒤 $\sigma_i u_i = Av_i$임을 이용해 반대편을 구하는 게 적절하다. 이런 부호 불일치는 eigen vector는 부호가 뒤집혀도 여전히 eigen vector이기 때문에 발생한다.

1. A^TA에 대해 $v_1, \cdots, v_n$을 구하거나, AA^T에 대해 $u_1, \cdots, u_m$을 구한다. 즉, eigen value를 구한 뒤, eigen vector를 뽑고 norm으로 나눠 orthonormal eigen basis를 구성한다.
2. singular value가 0이 아닌 부분에 대해 반대편 $u_1, \cdots, u_r$ 또는 $v_1, \cdots, v_r$을 구한다. 그리고 n 또는 m에 맞춰 나머지 vector들을 적절히 구한다. AA^T, A^TA로 구하거나, 직관적으로 채우거나, gram schmidt process를 적용할 수 있다.
3. 구한 값들로 U, V, Σ를 구성한다. 이때 순서를 잘 맞춰줘야 한다.

또한 SVD의 공식에 따라 $A^TA = V\Sigma^2 V^T$, $AA^T = U\Sigma^2 U^T$와 같이 나타낼 수 있는데, 이 수식은 spectral decomposition이므로 $\sigma = \sqrt{\lambda}$이고, A^TA, AA^T의 eigen value가 같음을 확인할 수 있다.

각 space의 orthogonal eign basis는 각 eigen space에 대해 gram schmidt process를 수행해 구할 수 있지만, R^n과 R^m의 basis끼리 mapping됨을 보장하진 못한다. SVD는 singular value를 사용해 두 basis에 대한 mapping을 나타낸다.

eigen decomposition에서 matrix P는 orthogonal matrix일 필요가 없지만, SVD에서는 U와 V가 orthogonal matrix이므로 rotation(또는 reflection)이 적용되는 것으로 이해할 수 있다.

symmetric matrix $A \in R^{n \times n}$에 대한 eigen decomposition과 SVD는 동일하다.

일부 문헌에서는 $m \times n$ matrix A에 대한 SVD $A = U\Sigma V^T$에서 U를 $m \times n$, Σ를 $n \times n$, V^T를 $n \times n$으로 정의하기도 한다. 이를 Reduced SVD라고 하고, 앞에서 정의한 방식을 Full SVD라고 한다.

10.1.2. Matrix Approximation

1. Matrix Approximation
rank가 r인 $m \times n$ matrix A의 SVD를 다음과 같이 rank-1 matrix(outer product)의 합으로 나타낼 수 있다.

$$A = U\Sigma V^T = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_n u_n v_n^T = \sum_{i=1}^{n} \sigma_i u_i v_i^T$$

이때 다음과 같이 항들 중 k개만 사용해서 rank가 k인 matrix를 얻을 수 있는데, 이를 A에 대한 Low Rank Approximation 또는 Rank-r Approximation, 혹은 Truncated SVD라고 한다.

$$\hat{A}(k) = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

2. Matrix Approximation과의 Error
error(distance)를 측정하려면 matrix에 대한 norm을 정의해야 한다. 여기에서는 matrix에 대한 **Spectral**

Norm을 사용하는데, 모든 vector $x \in R^n \setminus \{0\}$에 대해 다음 수식과 같이 정의된다. 즉, 이는 어떤 vector x를 matrix A에 곱했을 때 scaling되는 최대 길이를 의미한다.

$$||A||_2 = \max_x \frac{||Ax||_2}{||x||_2}$$

이때 A의 spectral norm은 A의 가장 큰 singular value σ_1이다. (이에 대한 증명은 다루지 않는다.)

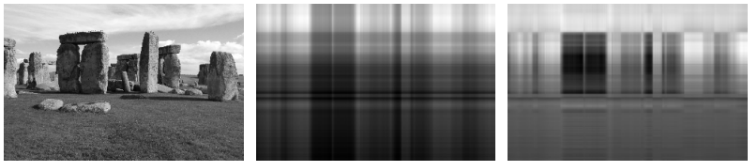
rank가 r인 $m \times n$ matrix A와 rank가 k인 $m \times n$ matrix B가 있고, $k \leq r$일 때, 다음이 성립한다. 이를 **Eckart-Young(에카르트-영) Theorem**이라고 한다. 주어진 matrix에 대한 low rank approximation을 수행할 때, 그 오차를 최소화하는 최적의 해가 SVD를 통해 도출됨을 보인 theorem이다. 이에 따르면 $\hat{A}(k)$가 error를 최소로하는 matrix라고 했을 때 이는 다음과 같고, 이때의 error가 σ_{k+1}이다. 즉, singular value가 큰 rank-1 matrix들의 합으로 나타낸 low rank approximation이 실제로 최적의 approximation이다.

$$\hat{A}(k) = argmin_{rk(B)=k}||A - B||_2 = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$$

$$||A - \hat{A}(k)||_2 = \sigma_{k+1}$$

첫 번째 수식에 대한 증명은 생략한다(필요하면 교재 등을 참고하자.). 두 번째 수식은 각 matrix를 rank-1 matrix들의 합으로 생각해 보면, spectral norm은 가장 큰 singular value이므로 성립한다.

다음과 같이 rank를 늘리면 원본에 가까워지는 것을 확인할 수 있다.



(a) Original image $\boldsymbol{A}$. (b) Rank-1 approximation $\widehat{\boldsymbol{A}}(1)$. (c) Rank-2 approximation $\widehat{\boldsymbol{A}}(2)$.

(d) Rank-3 approximation $\widehat{\boldsymbol{A}}(3)$. (e) Rank-4 approximation $\widehat{\boldsymbol{A}}(4)$. (f) Rank-5 approximation $\widehat{\boldsymbol{A}}(5)$.

10.2. 다양한 Decomposition들

10.2.1. Cholesky Decomposition

Cholesky Decomposition은 positive definite matrix A를 lower triangle matrix L과 그 전치행렬의 곱으로 나누는 decomposition이다. 이때 L을 A의 Cholesky Factor라고 하고, L은 유일하다.

$$A = LL^T$$

cholesky decomposition의 각 l_{ij}은 A의 원소와, 이전에 계산된 l_{ij}값이 있으면 계산할 수 있다. 예를 들어 다음과 같이 계산된다.

Example 4.10& (Cholesky Factorization)

Symmetric, positive definite matrix $\boldsymbol{A} \in \mathbb{R}^{3\times 3}$ 일 때, 이 행렬의 Cholesky factorization $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^\top$ 을 찾아보도록 하겠습니다. Cholesky Factorization 을 표현하면 다음과 같습니다.

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{21} & a_{22} & a_{32} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \boldsymbol{L}\boldsymbol{L}^\top = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{bmatrix} \tag{4.45}$$

위 식의 우항을 풀면, 다음과 같습니다.

$$\boldsymbol{A} = \begin{bmatrix} l_{11}^2 & l_{21}l_{11} & l_{31}l_{11} \\ l_{21}l_{11} & l_{21}^2 + l_{22}^2 & l_{31}l_{21} + l_{32}l_{22} \\ l_{31}l_{11} & l_{31}l_{21} + l_{32}l_{22} & l_{31}^2 + l_{32}^2 + l_{33}^2 \end{bmatrix} \tag{4.46}$$

이를 식 (4.45)의 좌항과 비교하면, diagonal elements l_{ii} 에서 다음의 간단한 패턴이 있다는 것을 볼 수 있습니다.

$$l_{11} = \sqrt{a_{11}}, \quad l_{22} = \sqrt{a_{22} - l_{21}^2}, \quad l_{33} = \sqrt{a_{33} - (l_{31}^2 + l_{32}^2)} \tag{4.47}$$

대각 요소 아래에 있는 요소들도 유사하게 다음과 같이 반복되는 패턴을 갖습니다.

$$l_{21} = \frac{1}{l_{11}}a_{21}, \quad l_{31} = \frac{1}{l_{11}}a_{31}, \quad l_{32} = \frac{1}{l_{22}}(a_{32} - l_{31}l_{21}) \tag{4.48}$$

이렇게 positive definit 3×3 행렬에 대해 Cholesky decomposition을 구성할 수 있습니다. Cholesky decomposition의 핵심은 a_{ij} 의 값과 이전에 계산된 l_{ij} 값을 알고 있다면 l_{ij} 를 역으로 계산할 수 있다는 것입니다.

10.2.2. CR Decomposition

CR(Colum-Row) Decomposition은 임의의 $m \times n$ matrix A를 두 행렬 C와 R로 나누는 decomposition 이다. 즉, $A = CR$ 꼴로 나눈다. 이때 C의 column은 A의 column space의 basis이고(basis of C(A)), R의 row 는 A의 row space의 basis이다(basis of $C(A^T)$). 즉, **CR Decomposition은 column space와 row space 의 basis를 찾아내는 분해**이다.

CR decomposition의 과정은 아래와 같다.

1. 첫 번째 matrix C를 구성한다. 첫번째 column은 A의 첫번째 column을 그냥 넣는다. 두번째 column 은 넣은 첫번째 column으로 만들 수 있는지 판단하고, 만들 수 없으면 그냥 넣고 만들 수 있으면 넣지 않는다.
2. 두 번째 matrix R을 구성한다. 첫번째 matrix의 column들을 활용해 기존 matrix을 생성할 수 있도록 구성한다.

   이때 A의 각 column이 C의 linear combination인 것을 고려하면 R을 채우기 쉽다. 즉, 다른 column들로 만들 수 있는 column이 아니면 하나만 1이고 나머지는 0인 column이 들어가게 되고, 다른 column들로 만들 수 있다면 해당 linear combination에 대응되는 값들이 들어가게 된다.

$$\text{Example} \quad A = \begin{bmatrix} 1 & 2 & 4 \\ 1 & 3 & 5 \end{bmatrix} \quad \begin{matrix} \text{Columns 1 and 2 of } A \text{ go directly into } C \\ \text{Column 3} = \mathbf{2}\,(\text{Column 1}) + \mathbf{1}\,(\text{Column 2}) \end{matrix}$$

$$A = \begin{bmatrix} 1 & 2 & \mathbf{4} \\ 1 & 3 & \mathbf{5} \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 1 & 0 & \mathbf{2} \\ 0 & 1 & \mathbf{1} \end{bmatrix} = CR \qquad \begin{matrix} 2 \text{ columns in } C \\ 2 \text{ rows in } R \end{matrix}$$

(↓ 2 1)

또는 아래와 같이 좀 더 명확한 방법도 존재한다.

1. elimination으로 A에 대한 RREF를 구한다.
2. 첫 번째 matrix C를 구성한다. A의 column 중 RREF에서 privot이 있는 위치의 것들을 가져와 순서대로 C를 구성한다.
3. 두 번째 matrix R을 구성한다. RREF에서 0으로만 이루어진 row를 모두 제거한 것을 R로 한다.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 7 & 10 \end{bmatrix} \to \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad A = CR = \begin{bmatrix} 1 & 2 \\ 2 & 4 \\ 3 & 7 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

10.2.3. LU Decomposition

1. LU Decomposition

LU Decomposition은 $m \times n$ matrix A를 lower triangle인 $m \times m$ matrix L과 upper triangle인 $m \times n$ matrix U로 나누는 decomposition이다. 즉, $A = LU$ 꼴로 나눈다.

한 row에 scalar배 해서 다른 row에 곱하는 elementary operation를 사용하면 A를 row echelon form으로 만들 수 있는데, 이게 U이다. 그리고 해당 elimination에 사용된 elementary matrix의 inverse matrix를 모두 곱한 것이 L이 된다. 즉, $(E_k \cdots E_1)A = U$, $A = (E_k \cdots E_1)^{-1}U$, $L = (E_k \cdots E_1)^{-1} = E_1^{-1} \cdots E_k^{-1}$이다.

row에 scalar배 해서 다른 row에 곱하는 elementary matrix의 inverse matrix는, row 값에 곱하는 scalar인 Multiplier(승수)의 부호를 바꾼 것과 같다. 당연하게도 동일한 row에 multiplier의 음수를 곱해 더해야 원래의 값이 나온다. 이때 elimination은 위쪽 row들부터 순차적으로 수행되고, $L = E_1^{-1} \cdots E_k^{-1}$에서는 그 역순으로 수행되므로 아래쪽 row들부터 연산되어 서로의 연산에 영향을 주지 않는다. 즉, L의 성분으로는 대응되는 elementary operation들에 대한 multipier의 부호만 바꾼 것들을 그냥 넣어주면 된다.

$$E = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad E^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$E_3E_2E_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 3 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad L = E_1^{-1}E_2^{-1}E_3^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 2 & -3 & 1 \end{bmatrix}$$

즉, LU decomposition은 아래와 같은 방법으로 구할 수 있다.

1. A에 row에 scalar배 해서 다른 row에 곱하는 elementary operation을 적용해 U를 만든다.
2. elementary operation에 사용된 multiplier의 부호를 바꿔 대응되는 위치에 넣어 L을 만든다.

2. PA=LU

invertible matrix에 대해 LU decomposition을 적용할 때 항상 한 row의 scalar배를 다른 row에 더하는 elementary operation으로 decomposition이 가능한 것은 아니다. 특히 pivot에 해당하는 위치의 성분이 0인 경우에 두 row를 바꾸는 elementary operation을 적용해야 할 수 있는데, permutation matrix를 곱하는 것으로 이를 적용할 수 있다. permutation matrix들은 해당 elimination 중간에 곱할 수도 있고, elimination 시작 전에 모두 적용할 수도 있는데 당연하게도 후자가 더 간결하다.

즉, 아래와 같이 작성할 수 있다.

$$P_n \cdots P_2P_1A = PA = LU$$

3. LU Decomposition의 장점

LU decomposition으로 matrix A를 L과 U로 나누는 것은 아래의 장점이 존재한다. 또한 두 개의 행렬로 나누지만 lower/upper triangle matrix이므로 A와 거의 동일한 storage를 사용할 수 있다.

1. A의 행렬식을 간단히 구할 수 있다.
2. systems of linear equation $Ax = b$를 더 쉽게 풀 수 있다. 특히 L과 U에 대한 equation은 L에 대해서는 전방 대입을, U에 대해서는 후방 대입해서 풀 수 있다. 즉, 앞/뒤부터 순차적으로 대입하기만 하면 x를 구할 수 있다.
3. Ax=b 연산에 elimination을 사용하는 것은 $x = bA^{-1}$를 구하는 것에 비해 computation을 줄일 수 있다.

$n \times n$ matrix A에 대한 elimination cost를 생각해보자. 곱하고 더하는 것을 단위로 생각한다면 각 pivot의 아래쪽을 전부 0으로 만드는 연산은 해당 row를 제외한 다른 아래쪽 row 모두와 연산하므로 $\sum_1^n k^2 - k = \frac{1}{3}n^3 + ...$이다.

또한 Ax=b에서 b에 대해 동일한 elimination을 수행하는 것을 생각해보면 이에 대한 연산은 각 원소를 scalar배 해서 그 아래 원소들에 더하는 것이므로 $(n-1) + (n-2) + \cdots + 1$이고, 후방 대입하는 것은 구하려는 변수 외의 값들을 사용해 전부 뺀 뒤에 한 번의 나누기를 하는 것이므로 $1 + \cdots + (n-1) + n$이다. 이 둘을 더하면 n^2이다.

즉, 정리하면 A와 b 각각에 대한 elimination에 $\frac{1}{3}n^3$과 n^2의 연산량이 필요하다. Ax=b는 $x = A^{-1}b$로도 구할 수 있는데, A^{-1}를 구하는 데에는 $2n^3$의 연산량이 필요하다고 한다. 즉, elimination을 활용하는 LU decomposition이 연산량 측면에서 더 효율적이다.

Lower Triangle Matrix(하삼각행렬)는 대각성분 위쪽 성분들이 값이 모두 0인 $n \times n$ matrix이고, Upper Triangle Matrix는 대각성분 아래쪽 성분들이 값이 모두 0인 $n \times n$ matrix이다. lower/upper triangle matrix의 determinant는 모든 대각성분의 값을 곱해 구할 수 있다.

LU에서 U를 diagonal matrix D로 쪼개 L 뿐만 아니라 U의 대각성분도 1로 만드는 LDU Decomposition이라는 decomposition도 존재한다. 즉, $A = LDU$로 분해한다.

10.2.4. Spectral Decomposition

1. Spectral Decomposition
Spectral Decomposition은 symmetric matrix A를 $A = Q\Lambda Q^T$꼴로 나누는 decomposition이다. 앞서 다룬 것처럼 실수 symmetric matrix에 대한 diagonalization은 항상 가능하고, $Q^TAQ = \Lambda$꼴로 나타내어진다. 즉, eigen decomposition을 symmetric matrix라는 특수 케이스에 적용한 것이다.

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

2. Rank-1 Matrix로 정리
Rank-1 matrix는 모든 column 또는 row가 하나의 vector의 상수배인 matrix로, 복잡한 matrix를 분해하는 기본 단위로 주로 사용되는 matrix이다. 이는 앞서 몇 번 다룬 것처럼 두 vector의 outer product 꼴 xy^T로 표현된다.

이에 따라 $A = Q\Lambda Q^T$는 다음과 같이 나타낼 수 있다. 이때 $q_iq_i^T$는 $n \times n$ matrix로, 이는 1) rank-1 matrix이고 2) projection matrix이고 3) symmetric matrix이다. 즉, symmetric matrix는 symmetric인 projection matrix들의 합으로 나타낼 수 있다.

$$A = Q\Lambda Q^T = \begin{bmatrix}\lambda_1 q_1 & \lambda_2 q_2 & \cdots & \lambda_n q_n\end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ \cdots \\ q_n \end{bmatrix} = \lambda_1 q_1 q_1^T + \lambda_2 q_2 q_2^T + \cdots + \lambda_n q_n q_n^T$$

λ_i가 중요도 순으로 정렬되어 있고, 일부 λ_i만 있어도 충분하다면 이렇게 rank-1 matrix들로 나타냈을 때 중요도가 낮은 뒤쪽 값들은 그냥 버릴 수 있다. 이런 식으로 데이터를 압축하거나 중요한 정보만 추출하는 것이 가능하다.

또한 Q가 orthogonal matrix이므로 이렇게 rank-1 matrix로 정리한 수식에 Q의 column을 곱하면 더 간단히 정리할 수 있다. 예를 들어, Sq_1은 다음과 같이 정리가 가능하다.

$$Aq_1 = (\lambda_1 q_1 q_1^T + \lambda_2 q_2 q_2^T + \cdots + \lambda_n q_n q_n^T)q_1 = \lambda_1 q_1 q_1^T q_1 = \lambda_1 q_1$$

11. Vector Calculus

Vector Calculus(벡터 미적분학)에서는 vector, 다변수 function에 대한 해석과 이해, 모델링등을 다룬다. vector calculus는 ML에서 사용하는 가장 기본적인 수학적 도구 중 하나로, 특히 ML의 많은 알고리즘은 objective function을 최적화하는 것을 그 목적으로 한다.

이 장의 핵심은 function이다. **function(함수)** f는 일반적으로 $x \in \mathbb{R}^D$와 function value(함숫값) $f(x) \in \mathbb{R}$에 대한 mapping이다. 이때 $\mathbb{R}^D$는 f의 **Domain(정의역)**이며, $\mathbb{R}$은 f의 **Codomain(공역)**, function value로 구성된 집합은 **Image/Range(치역)**이다.

이때 각 function은 미분가능하다고 가정하고, 따로 언급하지 않는다면 real number만 다룬다.

11.1. Differentiation

11.1.1. Differentiation of Univariate Functions

1. Differentiation of Univariate Functions
Difference Quotient(차분 몫)는 다음과 같이 계산되며, 이는 그래프 상 **Secant(할선)**의 **Slope(기울기, 변화율)**를 나타낸다. 이는 해당 구간에서의 **Average Slope(평균변화율)**로도 이해할 수 있다.

$$\frac{\delta y}{\delta x} = \frac{f(x+\delta x) - f(x)}{\delta x}$$

f가 미분가능할 때, $\delta x \to 0$에 대한 극한은 secant는 **Tangent(접선)**로 수렴하게 되고, 이때의 값 또한 x에서의 f의 tangent가 가지는 slope로 수렴한다. 즉, f를 **Differentiate(미분)**해 얻을 수 있는 f의 **Derivative(도함수)**는 다음과 같이 정의된다.

$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

2. Differentiation Rules
기본적인 differentiation rule로는 다음과 같은 것들이 있다.

$$\begin{aligned}
&\text{Product rule :} && (f(x)g(x))' = f'(x)g(x) + f(x)g'(x) \\
&\text{Quotient rule :} && \left(\frac{f(x)}{g(x)}\right)' = \frac{f'(x)g(x) - f(x)g'(x)}{(g(x))^2} \\
&\text{Sum rule :} && (f(x) + g(x))' = f'(x) + g'(x) \\
&\text{Chain rule :} && (g(f(x)))' = (g \circ f)'(x) = g'(f(x))f'(x)
\end{aligned}$$

x^n의 derivative는 nx^{n-1}이고, 이는 위 식과 이항정리를 활용해 증명할 수 있다.

11.1.2. Taylor Series

Taylor Series(테일러 급수)는 임의의 function f를 특정 위치 x_0에 대해서 polynomial로 근사하는 방법론으로, f를 어떤 항의 무한한 합으로 표현한다.

x_0에서 $f : \mathbb{R} \to \mathbb{R}$의 n차 **Taylor Polynormial(테일러 다항식)**은 다음과 같이 정의된다. 이때 $f^{(k)}(x_0)$는 x_0에서의 k차 derivative이고, $\frac{f^{(k)}(x)}{k!}$은 **Coefficient(계수)**이다.

$$T_n(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k$$

smooth function(연속이고 미분가능한) $f \in C^\infty$, $f : \mathbb{R} \to \mathbb{R}$에 대해, x_0에서의 taylor series는 다음과 같이 정의된다. 즉, taylor series는 무한대까지 항을 더하는 taylor polynomial이라고 할 수 있다. 이때 만약 $f(x) = T_\infty(x)$라면 f를 **Analytic(해석적)**하다고 한다. $x_0 = 0$에서의 taylor series는 Maclaurin Series라고도 한다.

$$T_\infty(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k$$

이미 polynomial인 function f에 대해, f의 차수보다 높은 차수를 사용해 구한 taylor polynomial은 f와 일치한다. 당연하게도 기존 차수보다 더 많이 differentiate하면 0이 되어 없어지므로 차수가 맞게 된다.

11.2. Partial Differentiations and Gradients

11.2.1. Partial Differentiations and Gradients

1. Partial Differentiations and Gradients
여러 개의 변수를 가지는 **Multivariate Function(다변수 함수)**의 derivative는 **Gradient**라고 한다. gradient는 **Partial Derivative(편미분)**으로 구할 수 있다. function $f : \mathbb{R}^n \to \mathbb{R}$가 있을 때, $x = \{x_1, \cdots, x_n\}$에 대한 partial derivative는 다음과 같다. 즉, 한 변수에 대해서만 differentiate한 것이다.

$$\begin{aligned}
\frac{\partial f}{\partial x_1} &= \lim_{h \to 0} \frac{f(x_1 + h, x_2, \cdots, x_n) - f(x)}{h} \\
\frac{\partial f}{\partial x_2} &= \lim_{h \to 0} \frac{f(x_1, x_2 + h, \cdots, x_n) - f(x)}{h} \\
&\vdots \\
\frac{\partial f}{\partial x_n} &= \lim_{h \to 0} \frac{f(x_1, x_2, \cdots, x_n + h) - f(x)}{h}
\end{aligned}$$

x를 column vector $x = [x_1, \cdots, x_n]^T$라 하고, 다음과 같이 각 원소에 대한 partial derivative로 각 column을 구성할 수 있다. 이렇게 multivariate function에 대해 partial derivative를 모아 놓은 것을 f에 대한 **Gradient**(또는 Jacobian)라고 한다. ∇_x는 x에 대한 gradient임을 나타낸다.

$$\nabla_x f = \text{grad} f = \frac{df}{dx} = [\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n}] \in \mathbb{R}^{1 \times n}$$

2. Gradients of Vector-Valued Functions
gradient 개념을 vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^m$으로 일반화할 수 있다. f와 $x = [x_1, \cdots, x_n]^T$에 대해 function value에 대응되는 vector와 x에 대한 partial derivative는 다음과 같다.

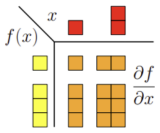
$$f(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_m(x) \end{bmatrix} \in R^m, \quad \frac{df}{dx} = \begin{bmatrix} \frac{\partial f_1(x)}{\partial x} \\ \frac{\partial f_2(x)}{\partial x} \\ \vdots \\ \frac{\partial f_m(x)}{\partial x} \end{bmatrix}$$

vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^m$의 모든 first-order partial derivatives(1차 편도함수)의 collection을 **Jacobian**이라고 한다. 이때 n dimension에서 m dimension으로 보내는 function의 jacobian J는 다음과 같은 $m \times n$ matrix이다. graident는 jacobian의 special case로 이해할 수 있다. 즉, column의 개수는 변수의 개수이고, row의 개수는 image/range/codomain의 dimension이다.

$$J = \nabla_x f = \frac{df}{dx} = \begin{bmatrix} \frac{\partial f_1(x)}{\partial x_1} & \cdots & \frac{\partial f_1(x)}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m(x)}{\partial x_1} & \cdots & \frac{\partial f_m(x)}{\partial x_n} \end{bmatrix}, \quad J(i,j) = \frac{\partial f_i}{\partial x_j}$$

jacobian은 coordinate transformation을 나타내는 matrix이다. 어떤 mapping에 대한 coordinate transformation을 나타내는 matrix는, 해당 mapping이 linear하다면 단순히 transformation matrix를 찾아 얻을 수 있다. 하지만 non-linear한 mapping에 대해서는 partial derivative를 사용하는 더 general한 방법을 사용해야 한다. coordinate transformation이 non-linear하다면 jacobian은 이 transformation을 linear한 것으로 근사한다.

Figure 5.6
Dimensionality of (partial) derivatives.



이번 장에서 함수의 derivatives(도함수)에 대해 살펴봤고, 위의 그림은 이러한 도함수의 차원에 대해 간략히 요약하고 있습니다.

함수가 $f : \mathbb{R} \to \mathbb{R}$ 라면, gradient는 단순히 스칼라 값 입니다(top-left).

$f : \mathbb{R}^D \to \mathbb{R}$ 라면, gradient는 $1 \times D$ row vector 입니다 (top-right).

$\boldsymbol{f} : \mathbb{R} \to \mathbb{R}^E$ 라면, gradient는 $E \times 1$ column vector 입니다.

$\boldsymbol{f} : \mathbb{R}^D \to \mathbb{R}^E$ 라면, gradient는 $E \times D$ matrix 입니다.

3. Partial Differentiation Rules
partial differentiation에 대해서는 다음과 같은 rule들이 성립한다. 다음 수식에서와 같이 chain rule에서는 마치 분모와 분자의 ∂f가 소거되는 것처럼 계산할 수 있다.

$$\begin{aligned}
\text{Product rule}: &\quad \frac{\partial}{\partial \boldsymbol{x}}(f(\boldsymbol{x})g(\boldsymbol{x})) = \frac{\partial f}{\partial \boldsymbol{x}}g(\boldsymbol{x}) + f(\boldsymbol{x})\frac{\partial g}{\partial \boldsymbol{x}} \\
\text{Sum rule}: &\quad \frac{\partial}{\partial \boldsymbol{x}}(f(\boldsymbol{x}) + g(\boldsymbol{x})) = \frac{\partial f}{\partial \boldsymbol{x}} + \frac{\partial g}{\partial \boldsymbol{x}} \\
\text{Chain rule}: &\quad \frac{\partial}{\partial \boldsymbol{x}}(g \circ f)(\boldsymbol{x}) = \frac{\partial}{\partial \boldsymbol{x}}(g(f(\boldsymbol{x}))) = \frac{\partial g}{\partial f}\frac{\partial f}{\partial \boldsymbol{x}}
\end{aligned}$$

이때 multivariate function에 대한 jacobian이 matrix일 수 있으므로, chain rule은 matrix multiplication의 형태로 작성될 수 있다.

gradient(jacobian)을 구할 때는 우선 shape을 확인한 뒤 계산하는 것이 편리하다.

gradient를 이렇게 column에 각 값을 넣어 row vector처럼 표현하는 것은 연산의 간편함 때문이라고 한다.

다음은 chain rule을 matrix multiplication의 형태로 나타내는 예시이다.

두 변수 x_1, x_2 에 대한 함수 $f : \mathbb{R}^2 \to \mathbb{R}$ 을 살펴보겠습니다. 추가로 x_1, x_2 는 t 에 대한 함수 $x_1(t), x_2(t)$ 로 정의합니다. t 에 대해 f 의 gradient를 계산하려면 다음과 같이 multivariate 함수에 대한 chain rule (5.48)을 적용해야 합니다.

$$\frac{\mathrm{d}f}{\mathrm{d}t} = \begin{bmatrix}\frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2}\end{bmatrix}\begin{bmatrix}\frac{\partial x_1(t)}{\partial t} \\ \frac{\partial x_2(t)}{\partial t}\end{bmatrix} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial t} \tag{5.49}$$

여기서 d 는 gradient를, ∂ 은 partial derivatives를 나타냅니다.

x_1, x_2 가 두 개의 변수 s, t 에 대한 함수 $x_1(s,t), x_2(s,t)$ 이고 $f(x_1, x_2)$ 가 있을 때, chain rule을 사용하여 다음과 같이 편미분 계산 결과를 얻을 수 있습니다.

$$\frac{\partial f}{\partial s} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial s} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial s} \tag{5.51}$$

$$\frac{\partial f}{\partial t} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial t} \tag{5.52}$$

그리고 행렬 곱에 의해서 gradient는 다음과 같이 얻을 수 있습니다.

$$\frac{\mathrm{d}f}{\mathrm{d}(s,t)} = \frac{\partial f}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}}{\partial (s,t)} = \underbrace{\begin{bmatrix}\frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2}\end{bmatrix}}_{=\frac{\partial f}{\partial \boldsymbol{x}}}\underbrace{\begin{bmatrix}\frac{\partial x_1}{\partial s} & \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial s} & \frac{\partial x_2}{\partial t}\end{bmatrix}}_{=\frac{\partial \boldsymbol{x}}{\partial (s,t)}} \tag{5.53}$$

예를 들어, vector y가 m개의 원소를 가지고 vector x가 n개의 원소를 가진다면, jacobian은 $m \times n$ matrix가 되고, 이 경우 다음과 같이 변수의 순서를 잘 맞춰 jacobian matrix를 작성할 수 있다.

$$\frac{dy}{dx} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

다음과 같이 chain rule을 사용해 합성함수에 대한 jacobian을 계산할 때에는, $x = g(t)$와 같이 function value와 변수를 맞춰주면 이해가 빠르다. **즉, 각 function의 domain과 docomain을 파악하고, 이후 x=g(t) 꼴로 각 function을 정리하면 chain rule로 나타내어 쉽게 풀 수 있다.**

Example 5.10 (Chain Rule)

다음과 같이 주어지는 함수 $h : \mathbb{R} \to \mathbb{R}, h(t) = (f \circ g)(t)$ 에서 t 에 대한 h 의 gradient를 계산해보도록 하겠습니다.

$$f : \mathbb{R}^2 \to \mathbb{R} \tag{5.69}$$
$$g : \mathbb{R} \to \mathbb{R}^2 \tag{5.70}$$
$$f(\boldsymbol{x}) = \exp(x_1 x_2^2) \tag{5.71}$$
$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = g(t) = \begin{bmatrix} t\cos t \\ t\sin t \end{bmatrix} \tag{5.72}$$

$f : \mathbb{R}^2 \to \mathbb{R}$ 이고, $g : \mathbb{R} \to \mathbb{R}^2$ 이므로, 다음과 같이 편도함수의 차원을 결정할 수 있습니다.

$$\frac{\partial f}{\partial \boldsymbol{x}} \in \mathbb{R}^{1\times 2}, \quad \frac{\partial g}{\partial t} \in \mathbb{R}^{2\times 1} \tag{5.73}$$

구하고자 하는 gradient는 다음과 같이 chain rule을 적용하여 계산할 수 있습니다.

$$\frac{\mathrm{d}h}{\mathrm{d}t} = \frac{\partial f}{\partial \boldsymbol{x}} \frac{\partial \boldsymbol{x}}{\partial t} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix} \begin{bmatrix} \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial t} \end{bmatrix} \tag{5.74a}$$
$$= \begin{bmatrix} \exp(x_1 x_2^2) x_2^2 & 2\exp(x_1 x_2^2) x_1 x_2 \end{bmatrix} \begin{bmatrix} \cos t - t\sin t \\ \sin t + t\cos t \end{bmatrix} \tag{5.74b}$$
$$= \exp(x_1 x_2^2)(x_2^2(\cos t - t\sin t) + 2x_1 x_2(\sin t + t\cos t)) \tag{5.74c}$$

$(x_1 = t\cos t, \quad x_2 = t\sin t)$

loss에 대한 graident로 이런 식으로 구할 수 있다.

Example 5.11 (Gradient of a Least-Squares Loss in a Linear Model)

$\boldsymbol{\theta} \in \mathbb{R}^D$ 가 parameter vector, $\boldsymbol{\Phi} \in \mathbb{R}^{N \times D}$ 가 input features, 그리고 $\boldsymbol{y} \in \mathbb{R}^N$ 이 대응되는 observations(관측치) 일 때, 다음의 linear model에 대해 살펴보도록 하겠습니다 (9장에서 조금 더 자세히 다룹니다).

$$\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{\theta} \tag{5.75}$$

여기서 다음의 함수들을 정의합니다.

$$L(\boldsymbol{e}) := \|\boldsymbol{e}\|^2 \tag{5.76}$$
$$\boldsymbol{e}(\boldsymbol{\theta}) := \boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta} \tag{5.77}$$

이때, $\frac{\partial L}{\partial \boldsymbol{\theta}}$ 를 찾는데, 이를 위해서 chain rule을 사용합니다. 여기서 L 을 *least-squares loss* function이라고 부릅니다.

미분을 하기 전에, 먼저 gradient의 차원을 결정합니다.

$$\frac{\partial L}{\partial \boldsymbol{\theta}} \in \mathbb{R}^{1 \times D} \tag{5.78}$$

Chain rule을 적용하면, gradient를 다음과 같이 표현할 수 있습니다.

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \frac{\partial L}{\partial \boldsymbol{e}} \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{\theta}} \tag{5.79}$$

여기서 d번째 element는 다음과 같이 주어집니다.

$$\frac{\partial L}{\partial \boldsymbol{\theta}}[1, d] = \sum_{n=1}^{N} \frac{\partial L}{\partial \boldsymbol{e}}[n] \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{\theta}}[n, d] \tag{5.80}$$

$\|\boldsymbol{e}\|^2 = \boldsymbol{e}^\top \boldsymbol{e}$ 이므로,

$$\frac{\partial L}{\partial \boldsymbol{e}} = 2\boldsymbol{e}^\top \in \mathbb{R}^{1 \times N} \tag{5.81}$$

이고, 다음의 결과도 얻을 수 있습니다.

$$\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{\theta}} = -\boldsymbol{\Phi} \in \mathbb{R}^{N \times D} \tag{5.82}$$

따라서, 구하고자 하는 gradient는 다음과 같이 계산됩니다.

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = -2\boldsymbol{e}^\top \boldsymbol{\Phi} \overset{(5.77)}{=} \underbrace{-2(\boldsymbol{y}^\top - \boldsymbol{\theta}^\top \boldsymbol{\Phi}^\top)}_{1 \times N} \underbrace{\boldsymbol{\Phi}}_{N \times D} \in \mathbb{R}^{1 \times D} \tag{5.83}$$

Remark. Chain rule을 사용하지 않고도 다음의 식으로 동일한 결과를 바로 얻을 수 있습니다.

$$L_2(\boldsymbol{\theta}) := \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta}\|^2 = (\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta})^\top (\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta}) \tag{5.84}$$

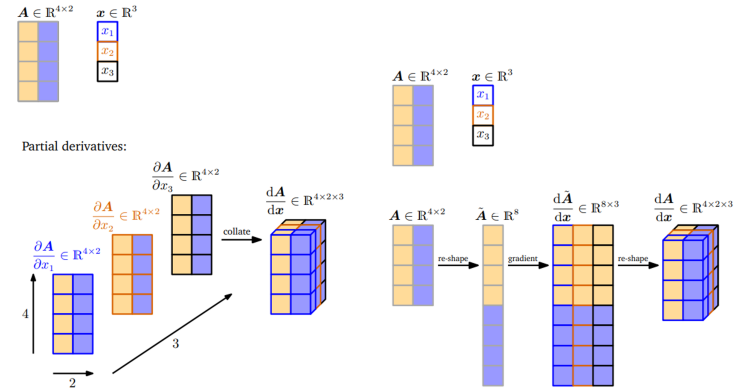
다만, 이러한 접근 방식은 L_2 와 같이 간단한 함수에서는 효과적이지만, deep function compositions에서는 실용성이 없습니다.

11.2.2. Gradients of Matrices

$p \times q$ matrix B에 대한 $m \times n$ matrix A의 gradient(jacobian)는 다음과 같이 계산된다.

$$J_{ijkl} = \frac{\partial A_{ij}}{\partial B_{kl}}$$

이런 matrix A와 B를 각각 길이가 mn인 vector와 길이가 pq인 vector로 reshape해서 gradient를 계산한 뒤 다시 shape을 맞춰주는 것으로도 계산이 가능하다.



(a) Approach 1: We compute the partial derivative $\frac{\partial \boldsymbol{A}}{\partial x_1}, \frac{\partial \boldsymbol{A}}{\partial x_2}, \frac{\partial \boldsymbol{A}}{\partial x_3}$, each of which is a 4×2 matrix, and collate them in a $4 \times 2 \times 3$ tensor.

(b) Approach 2: We re-shape (flatten) $\boldsymbol{A} \in \mathbb{R}^{4\times2}$ into a vector $\tilde{\boldsymbol{A}} \in \mathbb{R}^8$. Then, we compute the gradient $\frac{\mathrm{d}\tilde{\boldsymbol{A}}}{\mathrm{d}\boldsymbol{x}} \in \mathbb{R}^{8\times3}$. We obtain the gradient tensor by re-shaping this gradient as illustrated above.

gradient of vectors respect to matrices, gradient of matrices respect to matrices에 대한 예시도 나와 있으니, 필요하면 참고하자. 둘 다 각 원소에 대한 partial derivative를 수식적으로 나타낸 뒤 differentiate하거나, 혹은 여러 derivative를 묶여서 더 간단히 나타내는 approach를 사용하고 있다.

11.2.3. Useful Identities for Computing Gradients

다음 수식을 참고해 graident를 더 간단히 계산할 수 있다. 이때 대문자는 matrix, 소문자는 vector(column vector), $tr()$은 trace, det()은 determinant이다.

$$\frac{\partial}{\partial \boldsymbol{X}} \boldsymbol{f}(\boldsymbol{X})^\top = \left(\frac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial \boldsymbol{X}}\right)^\top$$
$$\frac{\partial}{\partial \boldsymbol{X}} \text{tr}(\boldsymbol{f}(\boldsymbol{X})) = \text{tr}\left(\frac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial \boldsymbol{X}}\right)$$
$$\frac{\partial}{\partial \boldsymbol{X}} \det(\boldsymbol{f}(\boldsymbol{X})) = \det(\boldsymbol{f}(\boldsymbol{X}))\text{tr}\left(\boldsymbol{f}(\boldsymbol{X})^{-1}\frac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial \boldsymbol{X}}\right)$$
$$\frac{\partial}{\partial \boldsymbol{X}} \boldsymbol{f}(\boldsymbol{X})^{-1} = -\boldsymbol{f}(\boldsymbol{X})^{-1}\frac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial \boldsymbol{X}}\boldsymbol{f}(\boldsymbol{X})^{-1}$$
$$\frac{\partial \boldsymbol{a}^\top \boldsymbol{X}^{-1}\boldsymbol{b}}{\partial \boldsymbol{X}} = -(\boldsymbol{X}^{-1})^\top \boldsymbol{a}\boldsymbol{b}^\top (\boldsymbol{X}^{-1})^\top$$
$$\frac{\partial \boldsymbol{x}^\top \boldsymbol{a}}{\partial \boldsymbol{x}} = \boldsymbol{a}^\top$$
$$\frac{\partial \boldsymbol{a}^\top \boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{a}^\top$$
$$\frac{\partial \boldsymbol{a}^\top \boldsymbol{X}\boldsymbol{b}}{\partial \boldsymbol{X}} = \boldsymbol{a}\boldsymbol{b}^\top$$
$$\frac{\partial \boldsymbol{x}^\top \boldsymbol{B}\boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{x}^\top(\boldsymbol{B} + \boldsymbol{B}^\top)$$
$$\frac{\partial}{\partial \boldsymbol{s}}(\boldsymbol{x} - \boldsymbol{A}\boldsymbol{s})^\top \boldsymbol{W}(\boldsymbol{x} - \boldsymbol{A}\boldsymbol{s}) = -2(\boldsymbol{x} - \boldsymbol{A}\boldsymbol{s})^\top \boldsymbol{W}\boldsymbol{A} \quad \text{for symmetric } \boldsymbol{W}$$

$x^T Bx$에 대한 derivative는 자주 활용하므로 특히 기억하자.

11.2.4. Backpropagation

1. Backpropagation

deep network는 다음과 같이 많은 function들의 합성으로 이해할 수 있고, 이에 따라 chain rule을 활용해 deep network에서 graident를 **Backpropagation(역전파)**으로 구할 수 있다. 직접 수식을 확인하고 명시적으로 differntiate하는 것은 그 cost가 많이 들 수 있는데, backpropagation을 활용하면 효율적으로 gradient를 계산할 수 있다. 다음 수식에서 x는 input, y는 observation(ex. class label)이고, 각 function $f_1, \cdots, f_K$는 각각의 parameter A와 b를 가진다.

$$\hat{y} = (f_K \circ f_{k-1} \circ \cdots \circ f_1)(x)$$

deep network의 각 layer는 다음과 같이 나타낼 수 있다. 이때 x_{i-1}은 $i-1$번째 layer의 output이고, σ는 activation function(ex. ReLU, sigmoid), θ는 parameter 집합 $\{A_0, b_0, \cdots, A_{K-1}, b_{K-1}\}$이다. input x와 observation y가 주어졌을 때 이 network를 학습시키려면 모든 parameter A_j, b_j $(j = 1, \cdots, K)$에 대한 loss function L의 graident가 필요하다.

$$\begin{aligned} f_0 &= x \\ f_i &= \sigma(A_{i-1}x_{i-1} + b_{i-1}) \quad i = 1, \cdots, K \\ L(\theta) &= ||y - f_K(\theta, x)||^2 \end{aligned}$$

backpropagation에 의해 다음과 같이 계산할 수 있다. 이때 주황색 부분은 각 layer에서 input에 대한 output의

derivative이고, 파란색 부분은 각 layer에서 parameter에 대한 output의 derivative이다. 또한 이전 layer에서 연산했다면 현재 layer에서는 박스 안에 있는 부분만 새로 계산하면 된다. (세 번째 수식은 마지막 notation이 잘못 표기되어 있음을 유의하자.)

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-1}} = \frac{\partial L}{\partial \boldsymbol{f}_K}\frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{\theta}_{K-1}}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-2}} = \frac{\partial L}{\partial \boldsymbol{f}_K}\boxed{\frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}}\frac{\partial \boldsymbol{f}_{K-1}}{\partial \boldsymbol{\theta}_{K-2}}}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-3}} = \frac{\partial L}{\partial \boldsymbol{f}_K}\frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}}\boxed{\frac{\partial \boldsymbol{f}_{K-1}}{\partial \boldsymbol{f}_{K-2}}\frac{\partial \boldsymbol{f}_{K-2}}{\partial \boldsymbol{f}_{K-3}}}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{i}} = \frac{\partial L}{\partial \boldsymbol{f}_K}\frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}}\cdots\boxed{\frac{\partial \boldsymbol{f}_{i+2}}{\partial \boldsymbol{f}_{i+1}}\frac{\partial \boldsymbol{f}_{i+1}}{\partial \boldsymbol{\theta}_{i}}}$$

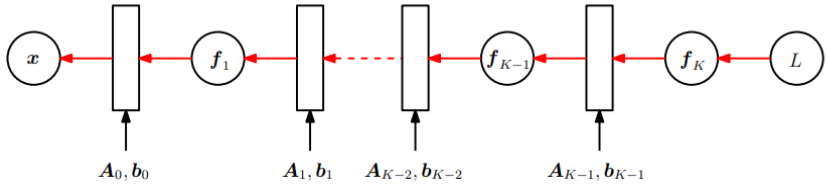
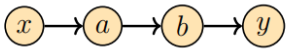


Figure 5.9 Backward pass in a multi-layer neural network to compute the gradients of the loss function.

실제로 backprop을 계산할 때에는 위의 figure와 같이 $f_0(=x), f_1, \cdots, f_k$는 function value로 생각하고, 각 function value 사이에 연산(parameter를 활용하는 연산 포함)이 존재하는 것으로 생각하자. 이는 backpropagation이 automatic differientiation의 speical case인 것을 고려하면 쉽게 받아들일 수 있다. 즉, intermediate variable에 대해 partial differentiate하는 것이다.

2. Automatic Differientiation

Automatic Differientiation(자동 미분)은 컴퓨터 프로그램을 통해 intermediate variable들을 가지고 chain rule을 적용해 정확한 gradient를 계산하는 기법이다. 이때 각 변수에는 기본 산술 연산(addition/multiplication)과 elementary function(sin, cos, exp, log)를 적용한다. 다음 그림과 같이 input x와 output y, 그리고 intermediate variable a, b가 있을 때 그 아래 수식과 같이 x에 대한 y의 gradient를 chain rule로 구할 수 있다.



$$\frac{dy}{dx} = \frac{dy}{db}\frac{db}{da}\frac{da}{dx}$$

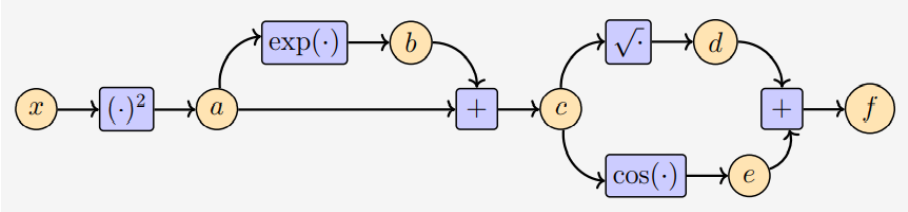
automatic differientiation에는 forward mode와 reverse mode가 있는데, reverse mode는 backpropagation을 구현한다. $x_1, \cdots, x_d$를 input variable, $x_{d+1}, \cdots, x_{D-1}$을 intermediate variables, x_D를 output variable, g_i를 elementary function, $P_a(x_i)$를 graph에서 x_i의 parent node로 구성된 집합이라고 하자. forward propagation은 다음 수식과 같이 정의된다. 즉, 모든 parent node를 elementary function에 넣어 얻은 값이다.

$$x_i = g_i(x_{P_a(x_i)}) \quad i = d+1, \cdots, D$$

$x_D = f$라고 하면, backpropagation은 다음 수식과 같이 정의된다. 즉, 모든 child node의 derivative를 활용해 derivative를 구할 수 있다.

$$\frac{\partial f}{\partial x_i} = \sum_{x_j : x_i \in P_a(x_j)} \frac{\partial f}{\partial x_j}\frac{\partial x_j}{\partial x_i} = \sum_{x_j : x_i \in P_a(x_j)} \frac{\partial f}{\partial x_j}\frac{\partial g_j}{\partial x_i}$$

예를 들어, 다음과 같은 computation graph를 생각할 수 있다.



연산 그래프를 살펴보고, output에서부터 역방향으로 계산하여 $\partial f/\partial x$ 를 아래와 같이 계산할 수 있습니다.

$$\frac{\partial f}{\partial c} = \frac{\partial f}{\partial d}\frac{\partial d}{\partial c} + \frac{\partial f}{\partial e}\frac{\partial e}{\partial c}$$
$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial c}\frac{\partial c}{\partial b}$$
$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial b}\frac{\partial b}{\partial a} + \frac{\partial f}{\partial c}\frac{\partial c}{\partial a}$$
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial a}\frac{\partial a}{\partial x}$$

11.3. Higher-Order Derivatives

11.3.1. Higher-Order Derivatives

두 변수 x,y를 가지는 function $f : \mathbb{R}^2 \to \mathbb{R}$이 있을 때, higher order derivative는 다음과 같이 나타낸다.

- $\frac{\partial^n f}{\partial x^n}$: x에 대한 n-th partial derivative.
- $\frac{\partial^2 f}{\partial y \partial x} = \frac{\partial}{\partial y}(\frac{\partial f}{\partial x})$: x에 대한 first derivative에서 y에 대해 구한 derivative.

Hessian은 모든 second-order partial derivative의 collection으로, function $f(x, y)$에 대한 hessian은 다음과 같이 표기한다. 일반적으로 $f : \mathbb{R}^n \to \mathbb{R}$에 대한 hessian matrix는 $n \times n$이다. hessian은 function의 곡률을 나타낸다.

$$\nabla^2_{x,y} f(x, y) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

만약 $f(x, y)$가 두 번 differentiate이 가능한 function이라면 다음 수식이 성립한다. 즉, differentiate의 순서는 중요하지 않다. 이에 따라 hessian matrix는 symmetric이다.

$$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial^2 f}{\partial y \partial x}$$

11.3.2. Multivariate Taylor Series

Multivariate Taylor Series는 임의의 multivariate function f를 특정 위치 x_0에 대해서 polynomial로 근사하는 방법론으로, f를 어떤 항의 무한한 합으로 표현한다.

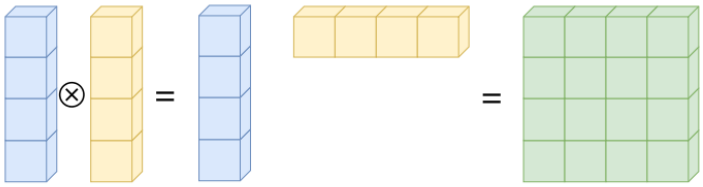
x_0에서 $f : \mathbb{R}^D \to \mathbb{R}$의 n차 Multivariate Taylor Polynormial은 다음과 같이 정의된다. 이때 $D^k_x(x_0)$는 x_0에서의 k차 derivative이고, $\delta = x - x_0$이다.

$$T_n(x) = \sum_{k=0}^{n} \frac{D^k_x(x_0)}{k!}\delta^k$$

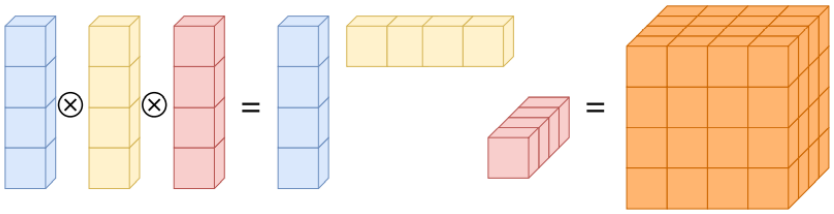
smooth function(연속이고 미분가능한) $f \in C^\infty$, $f : \mathbb{R} \to \mathbb{R}$에 대해, x_0에서의 taylor series는 다음과 같이 정의된다. 즉, taylor series는 무한대까지 항을 더하는 taylor polynomial이라고 할 수 있다. 이때 만약 $f(x) = T_\infty(x)$라면 f를 Analytic(해석적)하다고 한다. $x_0 = 0$에서의 taylor series는 Maclaurin Series라고도 한다.

$$T_\infty(x) = \sum_{k=0}^{\infty} \frac{D_x^k(x_0)}{k!} \delta^k$$

이때 δ^k는 δ에 대한 outer product를 k번 한 것이다.



(a) Given a vector $\boldsymbol{\delta} \in \mathbb{R}^4$, we obtain the outer product $\boldsymbol{\delta}^2 := \boldsymbol{\delta} \otimes \boldsymbol{\delta} = \boldsymbol{\delta}\boldsymbol{\delta}^\top \in \mathbb{R}^{4\times4}$ as a matrix.



(b) An outer product $\boldsymbol{\delta}^3 := \boldsymbol{\delta} \otimes \boldsymbol{\delta} \otimes \boldsymbol{\delta} \in \mathbb{R}^{4\times4\times4}$ results in a third-order tensor ("three-dimensional matrix"), i.e., an array with three indexes.

더 정확하고 복잡한 계산 과정은 교재를 참고하자.

계산 연습은 교재에 나와 있는 예시인 $x^2 + 2xy + y^3$에 대한 multivariate taylor polynomial을 구하는 걸로 하자. 이 예시에서만 해도 3차 이상의 differentiate가 필요하므로 추후 시간이 나면 해보자.

12. Probability and Distributions

ML model과, 그 model이 생성한 결과에 대한 uncertainty(불확실성)을 정량화하는 것이 중요한데, 이때 사용되는 것이 관심 있는 속성의 집합에 random한 실험의 결과를 매핑하는 개념인 random variable이다. random variable과 관련해서 특정 결과가 발생할 확률을 측정하는 function을 probability distribution이라고 한다.

"Probability theory은 실험에서 무작위로 발생하는 결과를 설명하는 수학적 구조를 정의하는데 그 목적을 둡니다. 예를 들어, 하나의 동전을 던질 때, 우리는 결과를 확정할 수는 없습니다. 하지만 여러 번 동전을 던졌을 때, 평균적인 결과에서 regularity를 관측할 수 있습니다. 이러한 확률의 수학적 구조를 사용하여 automated reasoning(자동화된 추론)을 수행하는 것이 목표이며, 이러한 의미에서 확률은 logical reasoning(논리적 추론)을 일반화합니다."

12.1. Probability and Distributions

12.1.1. Philosophical Issues

Plausible reasoning(개연적 추론)은 절대적인 증명이 아니라, plausibility(그럴듯함)을 근거로 결론을 도출하는 사고 방식을 말한다. 반면 **Logical Reasoning(논리적 추론)**은 사실이나 전제를 바탕으로 논리 규칙을 적용하여 결론을 도출하는 사고방식을 말한다. automated reasoning system을 구축할 때, 고전적인 boolean logic은 plausible reasoning을 표현하지 못했다. probability theory는 plausible reasoning까지 포함하는 boolean logic의 일반화라고 생각할 수 있고, ML에서는 automated reasoning system의 설계를 위해 이를 활용한다.

Cox-Jaynes theorem은 plausibility가 수학적 규칙을 정의하는 데에 사용될 수 있음을 보인다. 이 theorem에 의하면 plausibility는 다음과 같은 성질을 만족시켜야 한다.

1. Degrees of plausibility는 실수(real numbers)로 표현된다.
2. 이러한 수치는 common sense에 근거해야 한다.

3. 추론 결과는 아래에서 설명하는 'consistent'라는 단어의 세 가지 의미와 일치해야 한다.
   (a) Consistency or non-contradiction : 다른 수단을 통해 동일한 결과에 도달할 수 있을 때, 동일한 plausibility가 모든 경우에 대해 발견되어야 한다.
   (b) Honesty : 이용가능한 모든 데이터가 반드시 고려되어야 한다.
   (c) Reproducibility : 두 가지 문제에 대해 우리 지식의 상태(state of knowledge)가 같다면, 그 두 가지 문제에 대해 동일한 정도의 plausibility가 할당되어야 한다.

probability에 대해서는 2가지 주요 접근 방식이 있다. Frequentist(빈도주의적) 해석은 사건의 발생 빈도에 기반해 객관적으로 분석하고, 어떤 사건에 대한 probability는 무한한 데이터를 가지고 있을 때 극한에서 사건이 발생하는 상대적 빈도로 나타낸다. Bayesian(베이지안) 해석은 사건의 빈도가 아니라 주관적 믿음의 정도를 probability로 나타내고, 새로운 사건이 추가될 때마다 그 믿음을 수정하는 접근이다.

probability theory는 모집단의 특성을 모두 알고 있다고 가정하고, 미래에 발생할 event에 대해 예측한다. 반면 Statistic(통계학)에서는 모집단의 실제 특성을 알지 못하는 상태에서, 관측된 결과를 바탕으로 모집단의 특성을 추정하고 검증한다.

12.1.2. Probability and Random Variables

1. Probability Space
Probability Space는 무작위 사건을 수학적으로 모델링하기 위한 구조로, $(\Omega, \mathcal{A}, P)$로 구성된다.

- **Sample Space(표본 공간)** Ω: 실험에서 발생할 수 있는 모든 단일 결과들의 집합. 관측 가능한 데이터에 대한 전체 범위를 정의한다. sample space의 모든 결과들이 가지는 probability의 합은 1이다.
- **Event Space(사건 공간)** $\mathcal{A}$: sample space Ω 의 부분집합들을 원소로 가지는 집합족(집합을 원소로 하는 집합). 단일 결과들만으로는 복합적인 상황을 다루기 어렵기 때문에, sample space의 결과들을 묶어 proabability를 측정할 수 있는 대상인 event(사건)을 구성한다.
- **Probabiltiy(확률)** P: event space에 속하는 개별 event $A \in \mathcal{A}$에 대하여 0 이상 1 이하의 실수 값을 할당하는 function으로, event A에 대한 probability는 $P(A)$로 표기한다.

예를 들어, 주사위를 굴리는 sample space는 $\Omega = \{1, 2, 3, 4, 5, 6\}$, 짝수가 나오는 event는 $A = \{2, 4, 6\}$이고 그 probability는 $\frac{1}{2}$이다. 이런 event들의 집합이 event space이다.

2. Random Variable
어떤 real-world phenomenon을 모델링할 때에는, probabilities on quantities of interest인 subset를 포함하는 **Target Space** $\mathcal{T}$를 활용한다. 이때 target space의 요소를 **State**라고 한다. 이를 위해 sample space의 원소를 target space의 원소로 mapping하는 function $X : \Omega \to \mathcal{T}$를 사용하는데, 이 function을 **Random Variable(확률변수)**이라고 한다. random variable은 특정 **Probability Distribution(확률분포)**을 따라 분포한다.

random variable은 그 자체로 고정된 값을 가진다기보단, 무한히 다양하게 변할 수 있는 값들의 묶음을 나타낸다. 이에 따라 random variable에 대한 distribution에 대해서도 이야기할 수 있다.

이때 특정 event $S \in \mathcal{T}$에 대해 $P_X(S)$는 다음과 같이 random variable X에 대응되어 발생하는 sample space의 특정 사건과 대응된다.

$$P_X(S) = P(X \in S) = P(X^{-1}(S)) = P(\{w \in \Omega : X(w) \in S\})$$

하나의 random variable에 대한 분포를 **Univariate Distribution**이라 하고, 둘 이상의 random variable들에 대한 분포는 **Multivariate Distribution**이라고 하며 이때 state는 random variable들로 구성된 vector이다. 기본적으로 univariate distribution과 multivariate distribution 각각에서 값 x는 각각 x와 $\boldsymbol{x}$로 나타낸다. (하지만 본 요약본에서는 작성자의 편의를 위해 혼용한다. 지키는 경우도 있지만 지키지 않는 경우도 있다.)

예를 들어, 두 개의 동전을 돌려서 앞면이 나오는 횟수를 셀 때, random variable X는 앞면이 나오는 횟수라고 할 수 있다. 이때 가능한 결과는 $X(hh) = 2, X(ht) = 1, X(th) = 1, X(tt) = 0$이고, $\mathcal{T} = \{0, 1, 2\}$이다. $X = 2$에 대한 확률은 $P(X = 2) = \frac{1}{4}$이다.

12.1.3. Discrete Probabilities and Continuous Probabilities

1. Discrete Probabilities
target space $\mathcal{T}$가 discrete할 때, random variable X가 특정 값 $x \in \mathcal{T}$를 가질 probability를 지정할 수 있다. 이때의 X를 **Discrete Random Variable(이산확률변수)**이라 하고, $P(X = x)$를 **Probability Mass Function(PMF, 확률질량함수)**, 이때의 distribution을 **Discrete Probability Distribution(이산확률분포)**라 한다.

2. Continuous Probabilities
target space $\mathcal{T}$가 continuous할 때, $\mathcal{T}$는 real line이 된다. 또한 이런 경우 random variable X에 대한 probability는 $P(a \leq X \leq b)$와 같이 구간으로 표현하는 것이 자연스럽다. 이때의 X를 **Continuous Random Variable(연속확률변수)**이라 하고, 다음의 두 가지 조건을 만족하는 function $f : \mathbb{R}^D \to \mathbb{R}$를 **Probability Density Function(PDF, 확률밀도함수)**이라 한다.

1. $\forall x \in \mathbb{R}^D : f(x) \geq 0$. 즉, non-negative이다.
2. 다음을 만족하는 integral이 존재한다. 즉, $\mathbb{R}^D$ 공간 전체에 대해 integral(다변수적분)하면 1이 된다.

$$\int_{\mathbb{R}^D} f(x)dx = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f(x_1, \cdots, x_D)dx_1 \cdots dx_D = 1$$

random variable X에 대한 probability는 다음과 같이 나타낼 수 있다. 이때 특정 값에 대한 probability는 $P(X = x) = 0$이다.

$$P(a \leq X \leq b) = \int_a^b f(x)dx$$

이때의 distribution을 **Continuous Probability Distribution(연속확률분포)**라 한다.

또한 multivariate real-valued random variable $X = [X_1, \cdots, X_D]^T$와 state $x = [x_1, \cdots, x_D]^T \in \mathbb{R}^D$에 대한 **Cumulative Distribution Function(CDF, 누적분포함수)**는 다음과 같이 정의된다. CDF는 관습적으로 많이 사용한다고 한다.

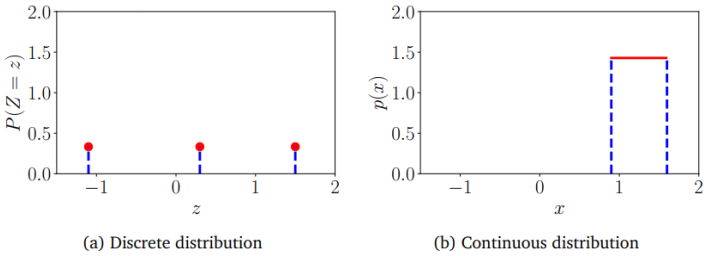
$$\begin{aligned} P_X(x) &= P(X_1 \leq x_1, \cdots, X_D \leq x_D) \\ &= \int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_D} f(z_1, \cdots, z_D)dz_1 \cdots dz_D \end{aligned}$$

3. ML에서의 Distrete/Continuous Probabilities
ML 문헌에서는 PMF를 distribution, PDF를 density, CDF를 distribution이라고 칭한다. 또한 다음과 같은 notation을 사용한다.

Type	"Point probability"	"Interval probability"
Discrete	$P(X = x)$ Probability mass function	Not applicable
Continuous	$p(x)$ Probability density function	$P(X \leq x)$ Cumulative distribution function

Cartesian Product(데카르트 곱, 곱집합)은 두 집합 A와 B에서 각각 원소를 하나씩 뽑아 만들 수 있는 모든 가능한 순서쌍의 집합을 말한다.

당연하게도 discrete/continuous distribution은 그 양상에서 차이가 있다.



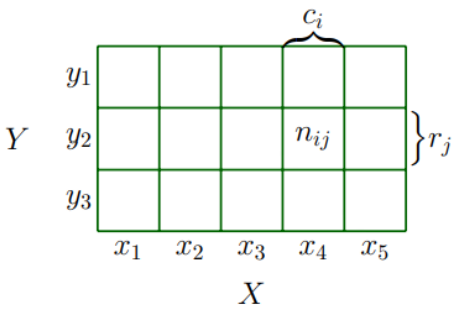
(a) Discrete distribution (b) Continuous distribution

Figure 6.3 Examples of (a) discrete and (b) continuous uniform distributions. See Example 6.3 for details of the distributions.

12.1.4. Joint Probability and Marginal Probability

1. Joint Probability

target space가 discrete할 때, 여러 random variable들에 대한 probability distribution을 다음 그림과 같이 이해할 수 있다.



Joint Probability는 두 event의 교집합에 대한 확률이며 다음과 같이 각 random variable에 대한 cartesian product로 나타낸다. 이때 n_{ij}는 state가 x_i, y_j인 event의 수, N은 모든 event의 수이다. 이때 p는 probability mass function으로, state를 받아서 probability를 출력하는 function으로 이해할 수 있다. 어떤 random variable이 probability distribution $p(x)$에 따라 분포한다는 것을 $X \sim p(x)$로 표기한다.

$$p(x_i, y_j) = P(X = x_i, Y = y_j) = P(X = x_i \cap Y = y_j) = \frac{n_{ij}}{N}$$

target space가 continuous한 경우, probability distribution은 연속적인 공간에서 정의되므로 PMF 대신 PDF $p(x, y)$ 를 사용한다. 이때 두 random variable에 대한 joint probability는 특정 영역 면적에 대한 이중 적분으로 계산된다.

$$P(a \leq X \leq b, c \leq Y \leq d) = \int_c^d \int_a^b p(x, y) dx dy$$

2. Marginal Probability

당연하게도 random variable이 여러 개 정의되어 있어도, 다른 random variable에 관계없이 특정 random variable에 대한 probability distribution을 표기할 수 있다. 이를 **Marginal Distribution(주변 분포)**이라고 한다. 특정 random variable의 영향을 모두 더하거나 적분하여 소거하고, 남은 variable만의 순수한 probability distribution를 구하는 과정을 **Marginalization(주변화)**라고 하며, 이때의 proabability를 **Marginal Probability(주변 확률)**라고 한다.

위의 그림과 같은 discrete probabiltiy distributaion에서는 각 column의 합을 c_i라고 했을 때 $P(X = x_i) = \frac{c_i}{N}$으로 X에 대한 marginal distribution을 나타낼 수 있다. 또한 이 경우에도 해당 random variable에 대한 probability의 합은 1이다.

continuous probability distribution에서의 marginal distribution 역시 관심 없는 다른 random variable에 대해 적분해 얻을 수 있다. 예를 들어, X 에 대한 marginal PDF $p(x)$는 joint PDF $p(x, y)$를 Y가 가질 수 있는 모든 가능한 y의 구간에 대해 적분하여 얻는다.

$$p(x) = \int p(x,y)dy = \int p(x)p(y|x)dy = \int p(y)p(x|y)dy$$

이는 discrete space에서 각 column이나 row의 합을 구하던 연산이 continuous space에서 적분으로 전환된 것이다. 도출된 marginal PDF $p(x)$ 를 모든 x 의 구간에 대해 적분한 값은 항상 1이다.

12.1.5. Conditional Probability and Likelihood

$X = x_0$일 때, $Y = y$인 경우의 probability를 **Conditional Proabability(조건부 확률)**라고 하며 $p(y|x_0)$로 표기한다. 위의 그림에서는 $P(Y = y_j|X = x_i) = \frac{n_{ij}}{c_j}$로 이해할 수 있다.

뒤에서 다루겠지만 $p(y|x)$는 likelihood로도 이해할 수 있다. **Likelihood(우도)**는 $p(y|x)$에 대해 y가 이미 관측된 상태에서, y를 발생시켰을 x가 얼마나 타당한지(plausible)를 나타내는 값이다. 즉, $p(y|x)$ 크기는 likelihood이고, 이는 y에 대해 x가 통계적으로 얼마나 관련이 있는지를 나타낸다. $p(y|x)$는 likelihood of x (given y) 또는 probability of y (given x)로 표기한다. likelihood는 probability와 동일한 수식 $p(y|x)$를 기반으로 하지만, probability는 x가 알려진 상태에서 y가 관측될 가능성을 말하므로 고정되는 변수가 다르다. ML에서는 y가 학습 데이터, x가 가중치에 해당되며, likelihood를 최대로 하는 x를 찾는 것을 목적으로 한다.

conditional probability와 likelihood는 continous probability distribution에서도 성립한다.

어떤 probability에 대한 수식의 모든 항에 conditional을 추가(**Conditionalization**)할 수 있다. 예를 들어, bayes' theorem에는 다음과 같이 적용 가능하다.

$$p(Z \mid X, \theta) = \frac{p(X, Z \mid \theta)}{p(X \mid \theta)}$$

12.2. Bayes' Theorem

12.2.1. Sum/Product Rule

probability distribution이 정해지면, 두 가지 기본적인 규칙인 sum/product rule만이 존재하고, 이 rule을 사용해 joint distribution을 marginal distribution과 연결할 수 있다.

1. Sum Rule
Sum Rule은 다음과 같다. 이때 $\mathcal{Y}$는 random variable Y의 target space이다. 즉, 이 식은 random variable Y의 state 집합을 모두 더하는 것이다. 즉, 이는 marginalization이다.

$$p(x) = \begin{cases} \sum_{y \in \mathcal{Y}} p(x,y) & \text{(if y is decrete)} \\ \int_{y \in \mathcal{Y}} p(x,y)dy & \text{(if y is continuous)} \end{cases}$$

이에 따라 어떤 multivariate distribution의 값 $\boldsymbol{x} = [x_1, \cdots, x_D]^T$에 대한 marginal은 다음과 같이 계산할 수 있다. 즉, x_i에 해당하는 것을 제외한 모든 random variable에 대해 sum rule을 적용하는 것이다.

$$p(x) = \int p(x_1, \cdots, x_D)dx_{\setminus i}$$

2. Product Rule
Product Rule은 다음과 같다. 즉, x, y가 동시에 일어날 probability는 x가 일어나면서 x가 일어났을 때 y가 일어날 probability이다. 이와 같이 joint distribution을 marginal distribution과 conditional distribution의 곱으로 인수분해할 수 있다.

$$p(x,y) = p(y|x)p(x)$$

12.2.2. Bayes' Theorem

Bayesian Statistics(베이지안 통계)는 새로운 데이터를 통해 기존의 믿음(probability)을 업데이트하며 확률을 계산하는 통계적 방법론이다. **Bayes' Theorem(베이즈 정리)**는 random variable $\boldsymbol{x}$와 그 사전 지식 $p(\boldsymbol{x})$가 있다고 했을 때, $\boldsymbol{y}$를 관측한 이후 $\boldsymbol{x}$에 대한 probability를 계산하는 방법을 정의한다. 그 수식은 다음과 같고, 이는 product rule에서 유도되었다.

$$\underbrace{p(\boldsymbol{x}|\boldsymbol{y})}_{\text{posterior}} = \frac{\overbrace{p(\boldsymbol{y}|\boldsymbol{x})}^{\text{likelihood}}\overbrace{p(\boldsymbol{x})}^{\text{prior}}}{\underbrace{p(\boldsymbol{y})}_{\text{evidence}}}$$

- **Prior(사전 확률)**는 아직 관측되지 않은 random variable $\boldsymbol{x}$에 대한 주관적인 사전 지식이다.
- **Likelihood(우도)**는 주어진 $\boldsymbol{x}$ 하에서 $\boldsymbol{y}$가 발생할 probability로, $\boldsymbol{x}$와 $\boldsymbol{y}$가 얼마나 관련이 있는지로도 이해할 수 있다.
- **Posterior(사후 확률)**는 $\boldsymbol{y}$를 관측한 이후에 $\boldsymbol{x}$에 대한 probability로, bayesian statistics에서 quantity of interest이다. y 관측 이후의 x의 probability distribution을 의미한다.
- **Evidence** 또는 **Marginal Likelihood**는 likelihood를 prior에 대해 가중 평균한 기댓값으로, $\boldsymbol{y}$가 발생할 수 있는 전체 probability를 의미한다. 이는 다음과 같이 계산된다. 이는 분모에서 normalization을 수행하는 것으로도 이해할 수 있다.

$$p(\boldsymbol{y}) = \int p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})dx = \mathbb{E}_X[p(\boldsymbol{y}|\boldsymbol{x})]$$

정리하면, posterior는 prior와 likelihood의 곱에 비례하며, 이를 marginal likelihood로 나눠 전체 probability distribution의 합이 1이 되도록 normalization한 것으로 이해할 수 있다. 등식 자체는 product rule에 의해 성립한다.

bayes' theorem은 그 수식 상 $\boldsymbol{x}$와 $\boldsymbol{y}$ 사이의 관계를 뒤집을 수 있도록 하기 때문에 Probabilisic Inverse라고도 부른다.

이때 likelihood $p(x|\theta)$는 x에 대한 probability distribution이지, θ에 대한 probability distribution이 아니다. 이는 θ에 대한 probability가 아니라, θ가 x를 얼마나 잘 설명하는지를 나타내는 수치이다. θ에 대해 적분해도 1이 되지 않는다.

ML에서 bayes's theorem은 데이터 X가 관측되었을 때 parameter θ의 probability distribution을 도출하는 도구로 사용한다. 즉, 다음과 같다. likelihood와 prior의 곱은 어떤 parameter가 존재할 때 관측값이 도출될 probability이고, 이는 $p(X|\theta)p(\theta) = p(X,\theta)$로 해당 값이 θ와 X의 joint probability인 것으로도 이해할 수 있다. 또한 evidence는 해당 값들의 총합이 1이 되도록 하기 위한 regularization을 수행한다. posterior는 실제 데이터 X와 대조해본 likelihood 결과를 바탕으로, 초기의 prior를 수학적으로 보정하여 새롭게 정한 parameter θ의 proabability distribution이다. 즉, posterior는 관측된 데이터를 고려한 새로운 distribution인 것이다.

$$p(\theta|X) = \frac{p(X|\theta)p(\theta)}{p(X)}$$

"posterior는 decision-making system 내에서 사용될 수 있고, full posterior를 갖는 것은 매우 유용하고 disturbances에 견고한 결정을 내릴 수 있습니다. 예를 들어, 모델 기반의 강화학습에서 plausible transition functions의 full posterior를 사용하여 아주 빠른 학습이 가능한데, posterior의 최댓값에 초점을 맞추면 일관된 실패를 일으킵니다. 따라서, full posterior를 갖는 것은 downstream task에서 매우 유용합니다."

12.3. Summary Statistics and Independence

random variable의 집합을 요약하고, random variable 쌍을 비교하는 방법을 알아보자. mean과 (co)variance는 확률 분포의 특성(expected value and spread)를 묘사하는데 유용하다.

12.3.1. Expected Value

1. Expected Value

Expected Value(기댓값)은 특정 probability distribution을 따르는 random variable X가 가질 수 있는 모든 가능한 값에 대해, 각 값이 발생할 probability를 가중치로 반영하여 산출한 이론적인 중심값이다. 이는 population mean(모집단의 평균) μ와 수학적으로 동일하다. univariate random variable $X \sim p(x)$의 function $g : \mathbb{R} \to \mathbb{R}$에 대한 expected value는 다음과 같이 정의된다. 이는 random variable X를 따를 때, 값 $g(x)$들을 평균낸다는 것을 의미한다.

$$\mathbb{E}_X[g(x)] = \begin{cases} \displaystyle\int_X g(x)p(x)dx & \text{if } X \text{ is a continuous random variable} \\ \displaystyle\sum_{x \in X} g(x)p(x) & \text{if } X \text{ is a discrete random variable} \end{cases}$$

이때 random variable X에 대한 expected value는 $g(x) = x$일 때 정의된다. 즉, 다음과 같다.

$$\mathbb{E}[X] = \begin{cases} \displaystyle\int_X xp(x)dx & \text{if } X \text{ is a continuous random variable} \\ \displaystyle\sum_{x \in X} xp(x) & \text{if } X \text{ is a discrete random variable} \end{cases}$$

multivariate random variable $X = [X_1, \cdots, X_D]^T$의 function $g : \mathbb{R}^D \to \mathbb{R}^D$에 대한 expected value는 다음과 같이 정의되고, 각 요소 X_i에 대한 계산 방법은 univariate와 동일하다. 즉, 각 요소에 대해 expected value를 계산하면 된다. 또한 아래에서 설명하는 것처럼 expected value는 linearity를 가지므로, vector나 matrix에 대한 expected value 계산은 개별 요소에 대한 계산하는 것으로 생각할 수 있다.

$$\mathbb{E}_X[g(x)] = \begin{bmatrix} \mathbb{E}_{X_1}[g(x_1)] \\ \vdots \\ \mathbb{E}_{X_D}[g(x_D)] \end{bmatrix} \in R^D$$

expected value는 linear operator이다. 즉, scalar $a, b, c \in \mathbb{R}$, vector $x \in \mathbb{R}^D$와 $f(x) = ag(x) + bh(x) + c$에 대해 다음이 성립한다. 이는 수식을 전개해 보면 당연하다. 이때 random variable이 vector에 해당하고, random variable이 아닌 vector나 matrix도 scalar처럼 적용된다.

$$\mathbb{E}_X[f(x)] = a\mathbb{E}_X[g(x)] + b\mathbb{E}_X[h(x)] + c$$

2. Median/Mode

Median(중앙값)은 중간에 있는 값으로, discrete probability distribution에서는 값이 정렬되어 있을 때 중간에 있는 값을 의미하고, continous probability distribution에서는 CDF가 0.5인 값으로 정의된다. 비대칭이거나 long tail distribution의 경우 median은 mean보다 더 직관에 가까운 값을 나타낸다. median은 univariate random variable에 대해서만 정의된다.

Mode(최빈값)은 가장 자주 등장하는 값을 말하며, discrete probability distribution에서는 가장 많이 나타난 값이고, continous probability distribution에서는 PDF가 peak인 지점이다.

사용하는 random variable이 명확하면 아래 첨자의 X 등은 생략하기도 한다.

12.3.2. Covariance

1. Covariance

두 random variable에 대한 **Convariance(공분산)**는 두 random variable이 얼마나 의존적인지를 나타내며, 각 deviation(편차)의 곱을 평균낸 값을 갖는다. univariate random variable X, Y에 대한 convariance는 다음과 같이 정의된다. 또한 첨자를 생략하고 expect value의 linearity를 고려하면 더 정리할 수 있다.

$$\mathrm{Cov}_{X,Y}[x,y] = \mathbb{E}_{X,Y}[(x - \mathbb{E}_X[x])(y - \mathbb{E}_Y[y])]$$
$$\mathrm{Cov}[x,y] = \mathbb{E}[xy] - \mathbb{E}[x]\mathbb{E}[y]$$

state $x \in \mathbb{R}^m$, $y \in \mathbb{R}^n$ multivariate random variable X, Y에 대한 convariance는 다음과 같이 정의된다. 이때의 $m \times n$ matrix는 **Cross-Covariance Matrix(교차 공분산 행렬)**라고 한다.

$$\begin{aligned}\mathrm{Cov}_{X,Y}[x,y] &= \mathbb{E}_{X,Y}[(x - \mathbb{E}_X[x])(y - \mathbb{E}_Y[y])^T] \\ \mathrm{Cov}[x,y] &= \mathbb{E}[xy^T] - \mathbb{E}[x]\mathbb{E}[y]^T \\ &= \begin{bmatrix} \mathrm{Cov}[x_1,y_1] & \mathrm{Cov}[x_1,y_2] & \dots & \mathrm{Cov}[x_1,y_n] \\ \mathrm{Cov}[x_2,y_1] & \mathrm{Cov}[x_2,y_2] & \dots & \mathrm{Cov}[x_2,y_n] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{Cov}[x_m,y_1] & \mathrm{Cov}[x_m,y_2] & \dots & \mathrm{Cov}[x_m,y_n] \end{bmatrix}\end{aligned}$$

2. Variance
한 random variable에 대한 covariance는 **Variance(분산)**라고 하며, $\mathbb{V}[x]$로 표기한다. variance의 제곱근은 **Standard Deviation(표준편차)**이라고 하며, $\sigma(x)$로 표기한다.

univariate random variable X에 대한 variance는 다음과 같이 계산할 수 있다. 이를 **Raw-Score Formula for Variance**라고 한다.

$$\mathbb{V}[x] = \mathbb{E}[x^2] - \mathbb{E}[x]^2$$

state $x \in \mathbb{R}^D$를 가지는 multivariate random variable X에 대한 variance는 다음과 같이 계산할 수 있다.

$$\begin{aligned}\mathbb{V}_X[x] = \mathrm{Cov}[x,x] &= \mathbb{E}[(x - \mathbb{E}(x))(x - \mathbb{E}(x))^T] \\ &= \mathbb{E}[xx^T] - \mathbb{E}[x]\mathbb{E}[x^T] \\ &= \begin{bmatrix} \mathrm{Cov}[x_1,x_1] & \mathrm{Cov}[x_1,x_2] & \dots & \mathrm{Cov}[x_1,x_D] \\ \mathrm{Cov}[x_2,x_1] & \mathrm{Cov}[x_2,x_2] & \dots & \mathrm{Cov}[x_2,x_D] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{Cov}[x_D,x_1] & \mathrm{Cov}[x_D,x_2] & \dots & \mathrm{Cov}[x_D,x_D] \end{bmatrix}\end{aligned}$$

이때의 variance는 개별 dimension 간의 관계를 나타내며, 이때의 $D \times D$ matrix를 **Covariance Matrix(공분산 행렬)**라고 한다. covariance matrix는 symmetric, positive semi-definite이며, spread를 나타낸다. 이때 대각 성분은 각 좌표축 방향으로 데이터가 얼마나 넓게 분포하는지를 나타내고, 비대각 성분은 두 variable 간의 상관관계에 의해 데이터가 어떤 방향으로 쏠려 있는지를 나타낸다. 또한 대각성분의 값들은 X의 각 dimension에 대한 marginal distribution 각각의 variance이다.

3. Correlation
서로 다른 random variable 쌍의 covariance를 비교할 때, 각 random variable 자체의 variance가 covariance 값에 영향을 미친다. 이에 따라 다음 수식과 같이 covariance를 정규화한 것을 **Correlation(상관계수)**이라고 한다. correlation은 두 distribution 간의 선형적 관계의 강도/방향을 측정한다.

$$\mathrm{corr}[x,y] = \frac{\mathrm{Cov}[x,y]}{\sqrt{\mathbb{V}[x]\mathbb{V}[y]}} \in [-1,1]$$

covariance matrix에서 각 요소를 standard deviation으로 나눈 matrix를 **Correlation Matrix**라고 한다. 즉, 이는 random variable $\frac{x}{\sigma(x)}$에 대한 covariance matrix이다.

두 변수 간의 correlation이 양수이면 **Positive Correlation**, 음수이면 **Negative Correlation**이 있다고 한다. 0이면 correlation이 없다고 한다.

4. Inner Product와의 관계

random variable은 vector space에서 vector로 취급될 수 있다. 이 space에서 covariance를 inner product로 정의할 수 있다. 즉, covariance는 linear, symmetric, positive definite이다.

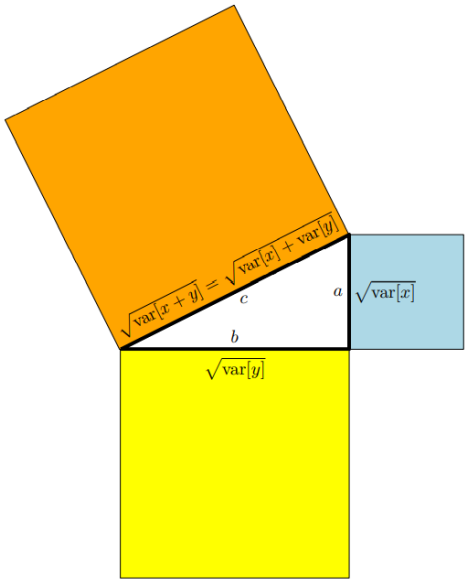
$$\langle X, Y \rangle = \mathrm{Cov}[x, y]$$

또한 다음과 같은 norm을 정의할 수 있는데, 이에 따라 random variable의 standard deviation가 norm인 것을 알 수 있다. norm이 클수록 불확실성이 크고, norm이 0이면 deterministic하다.

$$||X|| = \sqrt{\mathrm{Cov}[x, x]} = \sqrt{\mathbb{V}[x]} = \sigma[x]$$

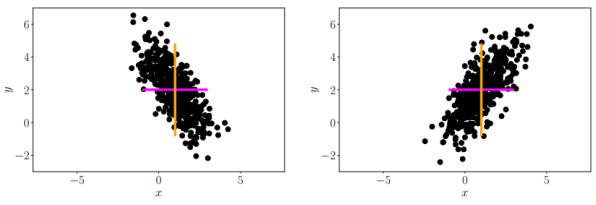
이 space에서 두 vector인 random variable 사이의 angle은 다음과 같이 계산할 수 있다. 즉, correlation을 기하학적으로 두 random variable 간의 angle로 생각할 수 있다. 두 random variable의 correlation이 0이라면 서로 orthogonal하다.

$$\cos\theta = \frac{\langle X, Y \rangle}{||X||\,||Y||} = \frac{\mathrm{Cov}[x, y]}{\sqrt{\mathbb{V}[x]\mathbb{V}[y]}}$$



본 요약본에서는 better intuition을 위해 covariance matrix를 positive definite이라고 가정한다.

다음 예시에서 선형적 관계가 강한지 약한지에 따른 correlation을 확인할 수 있다.



(a) x and y are negatively correlated.

(b) x and y are positively correlated.

Figure 6.5 Two-dimensional datasets with identical means and variances along each axis (colored lines) but with different covariances.

12.3.3. Mean & Covariance of Joint Distribution

joint distribution $p(x, y)$에 대한 mean과 covariance는 다음과 같이 x, y를 수직으로 연결한 vector에 대해 계산해 얻는다.

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix}$$

mean과 covariance는 다음과 같이 계산된다. 이때 Σ_{xx}는 covariance matrix이고, Σ_{xy}는 cross-covariance matrix이다.

$$\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{z}] = \begin{bmatrix} \mathbb{E}[\boldsymbol{x}] \\ \mathbb{E}[\boldsymbol{y}] \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}$$
$$\boldsymbol{\Sigma} = \mathrm{Cov}(\boldsymbol{z}) = \mathbb{E}[(\boldsymbol{z} - \boldsymbol{\mu})(\boldsymbol{z} - \boldsymbol{\mu})^T] = \begin{bmatrix} \boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_{yy} \end{bmatrix}$$

12.3.4. Population vs. Empirical

지금까지 살펴본 expected value는 Population **Mean(모평균)**이라고도 하고, covariance는 **Population Covariance(모공분산)**라고도 한다. 하지만 실제 데이터셋 또는 관측값들을 사용하는 경우, 이런 Population Statistics(모집단 통계) 대신 Empirical Statistics(경험적 통계)를 사용해야 할 수 있다. 이때의 mean/covariacne를 **Empirical(또는 Sample) Mean/Covariacne**라고 한다.

expected value 또는 population mean은 random variable이 가지는 이론적인 중심값인 반면, sample mean은 실제 관측된 데이터들의 집합이 가지는 중심값이다. 관측을 수행하는 표본의 크기 n 이 무한히 커질수록, 관측된 sample mean은 해당 random variable이 가지는 이론적 expected value $\mathbb{E}[X]$ 에 확률적으로 수렴하게 된다. 즉, expected value는 무한히 많은 관측을 수행했을 때 sample mean이 도달하게 될 수학적 한계치이다.

$x_i \in R^D$, $i = 1, \cdots, N$인 multivariate 관측값들에 대한 empirical mean μ와 empirical covariance matrix Σ는 다음과 같이 구할 수 있다. empircial covariance matrix 또한 symmetric, positive definite하다.

$$\mu = \frac{1}{N}\sum_{n=1}^{N} x_n, \quad \Sigma = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu)(x_n - \mu)^T$$

당연하게도 univariate 관측값들에 대한 variance는 다음과 같이 구할 수 있다.

$$\mathbb{V}_X[x] = \mathbb{E}_X[(x - \mu)^2] = \mathbb{E}_X[x^2] - (\mathbb{E}_X[x])^2$$

혹은 다음과 같이 실제 관측값들에 대한 pairwise difference를 계산해 variance(정확히는 variance에 2를 곱한 값)를 구할 수도 있다.

$$\frac{1}{N^2}\sum_{i,j=1}^{N}(x_i - x_j)^2 = 2[\frac{1}{N}\sum_{i=1}^{N} x_i^2 - (\frac{1}{N}\sum_{i=1}^{N} x_i)^2]$$

12.3.5. Statistical Independence

1. Statistical Independence
만약 다음 식이 성립한다면 두 random variable X, Y는 **Statistically Independent**하다고 한다. 이는 y의 값이 x에 의한 추가 정보와 관련이 없는 경우를 의미한다(반대의 경우도 성립).

$$p(x, y) = p(x)p(y)$$

만약 X, Y가 independent하다면, 다음이 성립한다. 이 중 위의 2가지는 역 또한 성립한다. 아래의 2개는 역이 성립하지 않는데, 이 두 수식은 correlation이 없음을 나타내는 것으로, independent하면 correlation이 없지만 corelation이 없다고 항상 independent한 것은 아니기 때문이다. 예를 들어, x와 x^2은 correlation이 없지만, 서로 dependent하다.

- $p(x|y) = p(x)$
- $p(y|x) = p(y)$
- $\mathbb{V}[x + y] = \mathbb{V}[x] + \mathbb{V}[y]$
- $\text{Cov}[x, y] = 0$

2. Conditional Independence
만약 다음 식이 성립한다면 두 random variable X, Y는 **Conditionally Independent**하다고 하고, $X \perp\!\!\!\perp Y \mid Z$로 표기한다. $p(x, y|z)$는 z가 관측되었을 때 x, y가 동시에 관측될 probability를 의미한다. 즉, conditional independence는 z를 알고 있으면 x와 y의 distribution이 분해됨을 의미한다. 이는 product rule을 적용해 분해할 수 있다. **statistically independent하면서 조건에 들어갈 때도 빼버릴 수 있는 것으로 이해할 수 있다.**

$$p(x, y|z) = p(x|z)p(y|z)$$
$$p(x, y|z) = p(x|y, z)p(y|z)$$

위의 식을 정리하면 다음과 같다. 즉, z에 대한 정보를 알고 있을 때, y에 대한 정보를 알게 되어도 x의 정보가 변경되지 않음을 의미한다.

$$p(x|z) = p(x|y, z)$$

앞서 언급했었는데, $p(x, y) = p(x|y)p(y)$와 같은 probability equation(확률 항등식)에서는 모든 항에 동일한 조건 z 등을 추가해도 항상 성립한다. $p(x, y|z) = p(x|y, z)p(y|z)$가 product rule을 적용한 결과는 모든 항에 조건 z를 추가한 수식으로 이해할 수 있다.

3. I.I.D.
I.I.D.(Independent and Identically Distributed)는 random variable의 집합이 independent하고, 모든 random variable들이 동일한 distribution을 가지는 경우를 말한다. 이 경우 해당 random variable들에 대한 joint distribution은 개별 random variable의 distribution의 단순 곱이 된다.

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i)$$

12.4. Gaussian Distribution

gaussian distribution은 continuous-valued random variables에 대해 가장 연구가 잘 되어 있는 distribution이다. 또한 inference에 필요한 여러 편리한 특성들을 가지고 있다(sum, product, transformation).

12.4.1. Gaussian Distribution

1. Gaussian Distribution
Gaussian Distribution(가우시안 분포) 또는 Normal Distribution(정규분포)은 continuous probability distribution의 하나이다. gaussian distribution은 mean과 variance만으로 형태가 완전히 결정되며, 평균을 중심으로 좌우가 완벽하게 대칭인 PDF를 가진다. gaussian distribution의 PDF를 **Gaussian Density**라고도 한다.

univariate random variable의 경우, gaussian distribution의 PDF는 다음과 같이 정의된다. μ는 mean, σ^2는 variance이다.

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

multivariate random variable의 경우에는 PDF가 다음과 같이 정의된다. μ는 mean vector, Σ는 covariance matrix이다. $p(x) = \mathcal{N}(x|\mu, \Sigma)$ 또는 $X \sim \mathcal{N}(\mu, \Sigma)$으로 표기한다. 참고로 여기에서 $|\boldsymbol{\Sigma}|$는 determinant를 의미한다.

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

mean이 0이고 standard deviation이 1인 gaussian distribution은 **Standrad Normal Distribution(표준정규분포)**이라고 한다.

2. Joint/Conditional/Marginal

gaussian distribution을 가지는 multivariate random variable에서도 conditional/marginal distribution을 정의할 수 있고, 둘 다 여전히 gaussian distribution이다. 다음과 같이 random variable x, y에 대한 joint distribution인 $p(x, y)$가 gaussian distribution을 따른다고 하자.

$$p(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_{yy}\end{bmatrix}\right)$$

이때 $p(x), p(y), p(x|y), p(y|x)$ 또한 gaussian distribution이다. 즉, $p(\boldsymbol{x})$는 다음과 같이 $\boldsymbol{x}$에 해당하는 부분만 가져온 것이다.

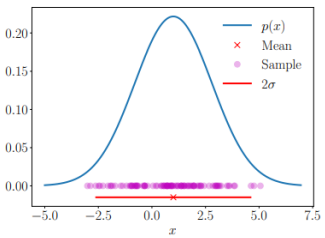
$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \boldsymbol{y})\mathrm{d}\boldsymbol{y} = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_{xx})$$

또한 $p(\boldsymbol{x}|\boldsymbol{y})$는 다음과 같다.

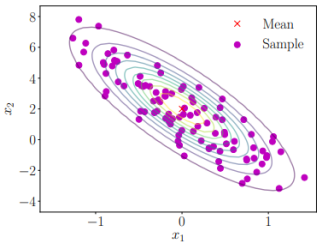
$$\begin{aligned}
p(\boldsymbol{x}|\boldsymbol{y}) &= \mathcal{N}(\boldsymbol{\mu}_{x|y}, \boldsymbol{\Sigma}_{x|y}) \\
\boldsymbol{\mu}_{x|y} &= \boldsymbol{\mu}_x + \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}(\boldsymbol{y}-\boldsymbol{\mu}_y) \\
\boldsymbol{\Sigma}_{x|y} &= \boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}\boldsymbol{\Sigma}_{yx}
\end{aligned}$$

다음은 univariate/bivariate gaussian distribution의 예시이다.

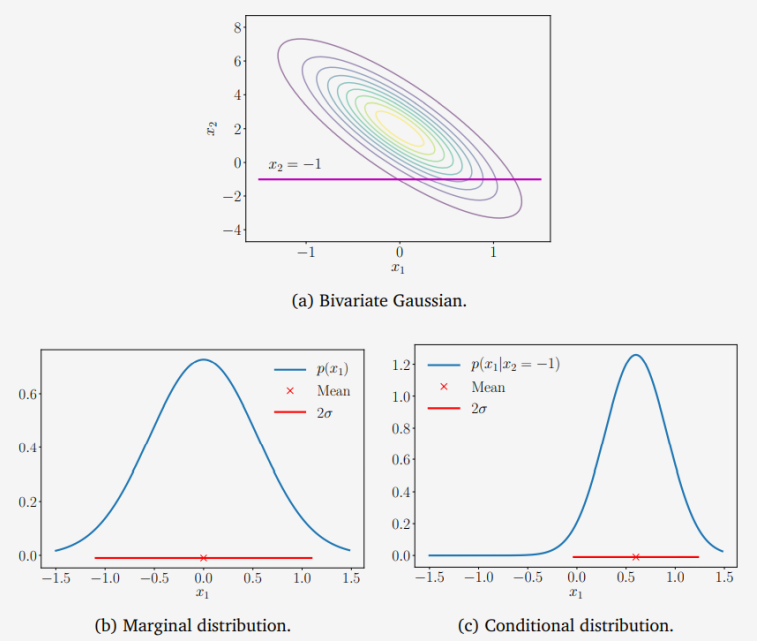
Figure 6.8 Gaussian distributions overlaid with 100 samples. (a) One-dimensional case; (b) two-dimensional case.



(a) Univariate (one-dimensional) Gaussian; The red cross shows the mean and the red line shows the extent of the variance.



(b) Multivariate (two-dimensional) Gaussian, viewed from top. The red cross shows the mean and the colored lines show the contour lines of the density.



(a) Bivariate Gaussian.

(b) Marginal distribution.

(c) Conditional distribution.

위 그림과 같은 bivariate gaussian distribution에 대해 다음과 같이 계산할 수 있다.

위 그림 Figure 6.9에서 설명하는 bivariate Gaussian distribution에 대해 살펴보겠습니다.

$$p(x_1, x_2) = \mathcal{N}\left(\begin{bmatrix}0\\2\end{bmatrix}, \begin{bmatrix}0.3 & -1\\-1 & 5\end{bmatrix}\right) \tag{6.69}$$

$x_2 = -1$ 라는 조건에 대해 Univariate Gaussian의 파라미터, 즉, 평균과 분산을 구하기 위해 식 (6.66)과 (6.67)을 적용합니다.

$$\mu_{x_1|x_2=-1} = 0 + (-1) \cdot 0.2 \cdot (-1 - 2) = 0.6 \tag{6.70}$$
$$\sigma^2_{x_1|x_2=-1} = 0.3 - (-1) \cdot 0.2 \cdot (-1) = 0.1 \tag{6.71}$$

따라서, conditional gaussian은 다음과 같이 주어집니다.

$$p(x_1|x_2 = -1) = \mathcal{N}(0.6, 0.1) \tag{6.72}$$

이와 대조적으로 marginal distribution $p(x_1)$ 은 (6.68)을 적용하여 얻을 수 있는데, 근본적으로 확률 변수 x_1 의 평균과 분산을 사용한 결과입니다.

$$p(x_1) = \mathcal{N}(0, 0.3) \tag{6.73}$$

12.4.2. Product & Sum of Gaussian Densities

1. Product of Gaussian Densities

두 gaussian density에 대한 product는 다음과 같으며, 여전히 gaussian distribution이다. 이는 두 gaussian distribution에 대한 bayes' theorem 등에서 사용될 수 있다.

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{a}, \boldsymbol{A})\mathcal{N}(\boldsymbol{x}|\boldsymbol{b}, \boldsymbol{B}) = c\mathcal{N}(\boldsymbol{x}|\boldsymbol{c}, \boldsymbol{C})$$

이때 $\boldsymbol{C}$와 $\boldsymbol{c}$, c는 다음과 같다.

$$\boldsymbol{C} = (\boldsymbol{A}^{-1} + \boldsymbol{B}^{-1})^{-1}$$
$$\boldsymbol{c} = \boldsymbol{C}(\boldsymbol{A}^{-1}\boldsymbol{a} + \boldsymbol{B}^{-1}\boldsymbol{b})$$
$$c = (2\pi)^{-\frac{D}{2}}|\boldsymbol{A} + \boldsymbol{B}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{a} - \boldsymbol{b})^\top(\boldsymbol{A} + \boldsymbol{B})^{-1}(\boldsymbol{a} - \boldsymbol{b})\right)$$

$\boldsymbol{a}, \boldsymbol{b}$는 random variable이 아니지만, c는 gaussian distribution의 형태이므로 편의상 다음과 같이 표기할 수 있다.

$$c = \mathcal{N}(\boldsymbol{a}|\boldsymbol{b}, \boldsymbol{A}+\boldsymbol{B}) = \mathcal{N}(\boldsymbol{b}|\boldsymbol{a}, \boldsymbol{A}+\boldsymbol{B})$$

2. Mixture Density
다음과 같은 두 density의 mixture를 생각하자. $p_1(x)$와 $p_2(x)$는 서로 다른 mean/variance를 가지는 univariate gaussian distribution이다(multivariate에 대해서도 x^2를 xx^T인 것으로 바꿔 동일한 방식으로 정의할 수 있다.). 이때 α는 $0 < \alpha < 1$인 scalar로, mixture weight이다.

$$p(x) = \alpha p_1(x) + (1-\alpha)p_2(x)$$

mean과 variance는 다음과 같다.

$$\begin{aligned}
\mathbb{E}[x] &= \alpha\mu_1 + (1-\alpha)\mu_2 \\
\mathbb{V}[x] &= [\alpha\sigma_1^2 + (1-\alpha)\sigma_2^2] + ([\alpha\mu_1^2 + (1-\alpha)\mu_2^2] - [\alpha\mu_1 + (1-\alpha)\mu_2]^2)
\end{aligned}$$

이는 다음과 같이 정의할 수 있다.

$$\begin{aligned}
\mathbb{E}[x] &= \int_{-\infty}^{\infty} xp(x)dx \\
&= \int_{-\infty}^{\infty} (\alpha x p_1(x) + (1-\alpha)xp_2(x))dx \\
&= \alpha\int_{-\infty}^{\infty} xp_1(x)dx + (1-\alpha)\int_{-\infty}^{\infty} xp_2(x)dx \\
&= \alpha\mu_1 + (1-\alpha)\mu_2
\end{aligned}$$

$$\begin{aligned}
\mathbb{E}[x^2] &= \int_{-\infty}^{\infty} x^2 p(x)dx \\
&= \int_{-\infty}^{\infty} (\alpha x^2 p_1(x) + (1-\alpha)x^2 p_2(x))dx \\
&= \alpha\int_{-\infty}^{\infty} x^2 p_1(x)dx + (1-\alpha)\int_{-\infty}^{\infty} x^2 p_2(x)dx \\
&= \alpha(\mu_1^2 + \sigma_1^2) + (1-\alpha)(\mu_2^2 + \sigma_2^2)
\end{aligned}$$

$$\begin{aligned}
\mathbb{V}[x] &= \mathbb{E}[x^2] - (\mathbb{E}[x])^2 \\
&= \alpha(\mu_1^2 + \sigma_1^2) + (1-\alpha)(\mu_2^2 + \sigma_2^2) - (\alpha\mu_1 + (1-\alpha)\mu_2)^2 \\
&= [\alpha\sigma_1^2 + (1-\alpha)\sigma_2^2] + ([\alpha\mu_1^2 + (1-\alpha)\mu_2^2] - [\alpha\mu_1 + (1-\alpha)\mu_2]^2)
\end{aligned}$$

"Mixture density 에서 각 컴포넌트는 조건부 분포 (conditional distributions) 로 볼 수 있습니다. mixture density 의 variacne 식은 조건부 분산 공식의 한 예이며, 이는 law of total variance 로 알려져 있습니다. 이는 일반적으로 두 개의 확률 변수 X, Y 에 대해 다음과 같은 관계를 만족합니다. $\mathbb{V}_X[x] = \mathbb{E}_Y[\mathbb{V}_X[x \mid y]] + \mathbb{V}_Y[\mathbb{E}_X[x \mid y]]$ 즉, X 의 (total) variance 는 expected conditional variance + variance of a conditional mean 입니다."

12.4.3. Sums of Random Variables

1. Sums of Random Variables

두 random variable X, Y가 각각 state $x, y \in R^D$를 가질 때, 다음이 성립한다. 이는 expected value의 linearity를 활용해 증명할 수 있다.

$$\begin{aligned}
\mathbb{E}[\boldsymbol{x}+\boldsymbol{y}] &= \mathbb{E}[\boldsymbol{x}] + \mathbb{E}[\boldsymbol{y}] \\
\mathbb{E}[\boldsymbol{x}-\boldsymbol{y}] &= \mathbb{E}[\boldsymbol{x}] - \mathbb{E}[\boldsymbol{y}] \\
\mathbb{V}[\boldsymbol{x}+\boldsymbol{y}] &= \mathbb{V}[\boldsymbol{x}] + \mathbb{V}[\boldsymbol{y}] + \mathrm{Cov}[\boldsymbol{x},\boldsymbol{y}] + \mathrm{Cov}[\boldsymbol{y},\boldsymbol{x}] \\
\mathbb{V}[\boldsymbol{x}-\boldsymbol{y}] &= \mathbb{V}[\boldsymbol{x}] + \mathbb{V}[\boldsymbol{y}] - \mathrm{Cov}[\boldsymbol{x},\boldsymbol{y}] - \mathrm{Cov}[\boldsymbol{y},\boldsymbol{x}]
\end{aligned}$$

또한 scalar를 더하는 것은 random variable의 distribution을 바꾸지 않으므로, $\mathbb{V}[x+a] = \mathbb{V}[x]$와 같이 variance에서 scalar는 제거할 수 있다.

2. Sum of Gaussian Random Variables
gaussian random variable X, Y가 independent라면 $\boldsymbol{x}+\boldsymbol{y}$ 또한 다음과 같은 gaussian distribution을 따른다. 이는 sums of random variable 수식을 사용해 증명할 수 있다.

$$\begin{aligned}
p(\boldsymbol{x}+\boldsymbol{y}) &= \mathcal{N}(\boldsymbol{\mu}_x + \boldsymbol{\mu}_y, \boldsymbol{\Sigma}_x + \boldsymbol{\Sigma}_y) \\
p(a\boldsymbol{x}+b\boldsymbol{y}) &= \mathcal{N}(a\boldsymbol{\mu}_x + b\boldsymbol{\mu}_y, a^2\boldsymbol{\Sigma}_x + b^2\boldsymbol{\Sigma}_y)
\end{aligned}$$

density에 대한 sum $p(\boldsymbol{x}) + p(\boldsymbol{y})$와 random variable에 대한 sum $p(\boldsymbol{x}+\boldsymbol{y})$를 구분하자.

12.4.4. Affine Transformation으로의 적용

1. Affine Transformation으로의 적용
random variable X가 $x \in R^D$를 가질 때, affine transformation $y = Ax + b$를 생각하자. y 또한 random variable이 되고, x에 대한 exepected value를 μ, covariance matrix를 Σ라 했을 때, expected value와 variance는 다음과 같이 정리된다. 이는 직접 수식을 전개해 정리해 보면 당연하다.

$$\begin{aligned}
\mathbb{E}[y] &= A\mathbb{E}[x] + b = A\mu + b \\
\mathbb{V}[y] &= \mathbb{V}[Ax] = A\mathbb{V}[Ax]A^T = A\Sigma A^T
\end{aligned}$$

또한 x, y의 covariance는 다음과 같다.

$$\mathrm{Cov}[x,y] = \mathbb{E}[x(Ax+b)^T] - \mathbb{E}[x]\mathbb{E}[Ax+b]^T = \Sigma A^T$$

2. Gaussian Distribution의 경우
만약 X가 gaussian distribution을 따른다면, affine transformation을 적용한 $y = Ax + b$도 gaussian distribution을 따른다. 즉, $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{A\mu} + \boldsymbol{b}, \boldsymbol{A\Sigma A}^T)$이다. 물론 이는 $b = 0$인 $y = Ax$에 대해서도 성립하고, 이때 $\boldsymbol{b}$가 바뀌어도 variance는 동일하고 $\boldsymbol{\mu}$만 이동하는 것을 알 수 있다.

$y = Ax$에 대한 reverse transformation도 구할 수 있다. A가 invertible이라면 위의 수식에서와 같이 $x = A^{-1}y$의 distribution을 구하는 것으로 생각할 수 있고, A가 non-invertible이라면 pseudo inverse에서와 같이 $x = (A^TA)^{-1}A^Ty$인 것으로 distribution을 구할 수 있다.

컴퓨터에서의 random sampling을 할 때 복잡한 분포를 바로 sampling하기는 어려우므로, uniform distribution과 같은 간단한 분포를 사용해 난수를 뽑은 후, transformation을 통해 원하는 분포의 샘플링을 얻을 수 있다고 한다.

"Multivariate normal $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ 로부터 샘플을 얻기 위해서, 가우시안 확률 변수의 linear transformation 속성을 이용할 수 있습니다. 만약 $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ 라면 $\boldsymbol{y} = \boldsymbol{Ax} + \boldsymbol{\mu}$ (where $\boldsymbol{AA}^T = \boldsymbol{\Sigma}$) 는 평균이 $\boldsymbol{\mu}$ 이고 공분산 행렬이 $\boldsymbol{\Sigma}$ 인 가우시안 분포입니다. $\boldsymbol{A}$ 를 선택하는 한 가지 편리한 방법은 공분산 행렬 $\boldsymbol{\Sigma} = \boldsymbol{AA}^T$ 에 Cholesky decomposition 을 사용하는 것입니다. Cholesky decomposition 에서는 $\boldsymbol{A}$ 가 triangular 이기 때문에 연산 측면에서 효율적입니다."

12.5. Conjugacy and the Exponential Family

ML의 맥락에서 probability distribution 조작을 할 때, closure property(연산 후 동일한 타입의 객체 반환)가 존재하고, 추가 데이터 추가에 따른 parameter 수정이 존재하지 않고, parameter에 대한 추정이 편리한 것이 좋다. 그런 의미에서 exponential family의 효용을 살펴보자. statistics, ML에서 사용되는 대부분의 distribution이 exponential family에 포함된다.

12.5.1. Conjugacy

posterior가 prior와 form/type이 같다면, 이 prior를 likelihood function에 대한 **Conjugate(켤레)**라고 한다. 이는 prior 와 posterior 가 동일한 분포족(Family of distributions)에 속하며, prior 가 likelihood 와 대수적으로 호환되는 수식 구조를 공유하여 parameter 값만 업데이트된다는 것을 의미한다.

bayes's theorem에 의하면 posterior은 prior와 likelihood의 곱에 비례한다. 하지만 posterior를 분석적으로 계산하려면 evidence를 구하기 위해 모든 공간에 대해 적분을 해야 하는데 이는 계산이 어려우며, 데이터 관측 전에 해당 문제에 대한 지식을 가지고 있지 않을 수 있으므로 prior를 설정하는 것 역시 까다로운 문제다.

하지만 exponential family의 경우 conjugate prior를 사용하면, 복잡한 적분 과정을 거치지 않고 parameter에 대한 단순 덧셈을 하는 것만으로 posterior를 도출할 수 있다.

Likelihood	Conjugate prior	Posterior
Bernoulli	Beta	Beta
Binomial	Beta	Beta
Gaussian	Gaussian/inverse Gamma	Gaussian/inverse Gamma
Gaussian	Gaussian/inverse Wishart	Gaussian/inverse Wishart
Multinomial	Dirichlet	Dirichlet

12.5.2. Bernoulli/Binomial/Beta Distribution

1. Bernoulli Distribution

Bernoulli Distribution(베르누이 분포)은 $x \in \{0, 1\}$의 state를 가진 single binary random variable X에 대한 분포이다. 베르누이 분포 $\text{Ber}(\mu)$는 $X = 1$ 의 probability를 표현하는 single continuous parameter $\mu \in [0, 1]$에 의해 제어되고, 다음과 같이 정의된다.

$$p(x|\mu) = \mu^x(1-\mu)^{1-x}, \quad x \in \{0, 1\}$$
$$\mathbb{E}[x] = \mu$$
$$\mathbb{V}[x] = \mu(1-\mu)$$

2. Binomial Distribution

Binomial Distribution(이항 분포)은 bernoulli distribution으로부터 추출한 N개의 샘플 중 $X = 1$ 이 m번 발생할 probability를 설명하는데 사용될 수 있고, binomial distribution $\text{Bin}(N, \mu)$는 다음과 같이 정의된다.

$$p(m|N, \mu) = \binom{N}{m}\mu^m(1-\mu)^{N-m}$$
$$\mathbb{E}[m] = N\mu$$
$$\mathbb{V}[m] = N\mu(1-\mu)$$

3. Beta Distribution

Beta Distribution(베타 분포)로 continual random variable $\mu \in [0, 1]$ 에 대한 분포로, 일부 binary event(ex. bernoulli distribution의 parameter)의 probability를 나타낼 때 주로 사용된다. beta distribution $\text{Beta}(\alpha, \beta)$는 두 parameter $\alpha > 0, \beta > 0$에 의해 제어되고, 이는 다음과 같이 정의된다.

$$p(\mu|\alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\mu^{\alpha-1}(1-\mu)^{\beta-1}$$

$$\mathbb{E}[\mu] = \frac{\alpha}{\alpha+\beta}, \quad \mathbb{V}[\mu] = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$$

이때, $\Gamma(\cdot)$ 은 Gamma Function으로, 다음과 같이 정의된다. 이는 beta distribution을 normalization한다.

$$\Gamma(t) := \int_0^\infty x^{t-1}\exp(-x)dx, \quad t > 0$$
$$\Gamma(t+1) = t\Gamma(t)$$

bernoulli/binomial distribution들에 대한 conjugate prior를 살펴보자.

Example 6.11 (Beta-Binormal Conjugacy)

아래와 같이 정의되는 binormal random variable $x \sim \mathrm{Bin}(N, \mu)$ 를 고려해봅시다 (N번 동전을 던졌을 때, 앞면이 x번 나올 확률).

$$p(x|N,\mu) = \binom{N}{x}\mu^x(1-\mu)^{N-x}, \quad x = 0, 1, \ldots, N \tag{6.102}$$

여기서 μ 는 앞면이 나올 확률입니다.

그리고, 파라미터 μ 에 대한 Beta prior, 즉, $\mu \sim \mathrm{Beta}(\alpha, \beta)$ 를 다음과 같이 정의합니다.

$$p(\mu|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\mu^{\alpha-1}(1-\mu)^{\beta-1} \tag{6.103}$$

이제 어떤 결과 $x = h$, 즉, 동전을 N번 던졌을 때 h번 앞면이 나오는 결과를 관측하면, μ 에 대한 사후 분포(posterior distribution)을 다음과 같이 계산할 수 있습니다.

$$\begin{aligned}
p(\mu|x=h,N,\alpha,\beta) &\propto p(x|N,\mu)p(\mu|\alpha,\beta) && (6.104a)\\
&\propto \mu^h(1-\mu)^{N-h}\mu^{\alpha-1}(1-\mu)^{\beta-1} && (6.104b)\\
&= \mu^{h+\alpha-1}(1-\mu)^{(N-h)+\beta-1} && (6.104c)\\
&\propto \mathrm{Beta}(h+\alpha, N-h+\beta) && (6.104d)
\end{aligned}$$

즉, 사후 분포(posterior distribution)는 prior와 같은 Beta distribution 입니다. 따라서, Beta prior는 Binomial likelihood function의 파라미터 μ 에 대한 conjugate 입니다.

Example 6.12 (Beta-Bernoulli Conjugacy)

$x \in \{0, 1\}$ 가 파라미터 $\theta \in [0, 1]$ 인 베르누이 분포를 따른다고 가정하겠습니다. 즉, $p(x = 1|\theta) = \theta$ 입니다. 이는 다음과 같이 표현될 수도 있습니다.

$$p(x|\theta) = \theta^x(1-\theta)^{1-x}$$

이제 θ 가 파라미터 α, β 를 갖는 베타 분포를 따른다고 가정합니다. 즉, 다음과 같습니다.

$$p(\theta|\alpha,\beta) \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}$$

베타 분포와 베르누이 분포를 곱하면, 다음의 식을 얻을 수 있습니다.

$$\begin{aligned}
p(\theta|x,\alpha,\beta) &= p(x|\theta)p(\theta|\alpha,\beta) && (6.105a)\\
&\propto \theta^x(1-\theta)^{1-x}\theta^{\alpha-1}(1-\theta)^{\beta-1} && (6.105b)\\
&= \theta^{\alpha+x-1}(1-\theta)^{\beta+(1-x)-1} && (6.105c)\\
&\propto p(\theta|\alpha+x, \beta+(1-x)) && (6.105d)
\end{aligned}$$

위 식에서 마지막 식은 파라미터 $(\alpha + x, \beta + (1 - x))$ 를 갖는 베타 분포라는 것을 보여줍니다.

12.5.3. Sufficient Statistics

어떤 random variable의 **Statistic(통계량)**은 mean, variance 등 해당 sample 집합에 대한 deterministic function이다. **Sufficient Statistic(충분통계량)**은 어떤 distribution에서 얻을 수 있는 모든 사용 가능한 정보를 포함하는 statistic이다. 즉, population(모집단)에 대해 추론하는데 필요한 모든 필요한 정보를 포함하며, distribution을 표현하는 데에 충분한 statistic이다.

Fisher-Neyman Factorization Theorem에서는, random variable X가 PDF $p(x|\theta)$를 가진다고 가정했을 때, 다음과 같이 작성될 수 있다면 statistic $\phi(x)$는 θ에 대해 Sufficient하다고 한다. 이때 $h(x)$는 θ에 independent한 distribution이고, g_θ는 sufficient statistics $\phi(x)$를 통해 θ에 의존적인 모든 정보를 포함한다.

$$p(x|\theta) = h(x)g_\theta(\phi(x))$$

모든 의존적인(가능한) 정보를 포함한다는 것은 주어진 θ에 대한 x의 probability가 θ에 의존하지 않는 부분과, $\phi(x)$만을 통해서 θ에 의존하는 부분으로 분해될 수 있다는 것을 말한다. 즉, θ에 의존적인 모든 정보는 $\phi(x)$가 가지고 있는 것이다.

"동전 던지기와 같이 성공(1) 또는 실패(0)의 결과만 가지는 독립적인 Sample $X = (X_1, X_2, \dots, X_n)$ 이 Parameter θ 를 따르는 Bernoulli distribution에서 추출되었다고 가정했을 때, Joint PMF $P(X|\theta) = \prod_{i=1}^{n} \theta^{x_i}(1-\theta)^{1-x_i} = \theta^{\sum_{i=1}^{n} x_i}(1-\theta)^{n-\sum_{i=1}^{n} x_i}$ 는 Fisher-Neyman Factorization Theorem에 따라 θ 에 독립적인 함수 $h(X) = 1$ 과 Statistic $\phi(X) = \sum_{i=1}^{n} x_i$ 로 정의된 $g_\theta(\phi(X)) = \theta^{\phi(X)}(1-\theta)^{n-\phi(X)}$ 를 통해 $P(X|\theta) = h(X)g_\theta(\phi(X))$ 로 분해될 수 있다. 이는 개별 데이터의 관측 순서와 같은 불필요한 세부 사항은 배제하고 오직 총 성공 횟수에 해당하는 $\sum_{i=1}^{n} x_i$ 만으로 Parameter θ 를 추정하는 데 필요한 모든 정보를 완전하게 보존할 수 있음을 대수적으로 증명하므로, 표본의 합 $\sum_{i=1}^{n} x_i$ 는 Bernoulli distribution의 Parameter θ 에 대한 Sufficient Statistic 입니다."

12.5.4. Exponential Family

1. Exponential Family

Exponential Family(지수족)은 paraemter $\theta \in \mathbb{R}^D$에 의해 정의되는 probability distribution의 family이다. 이는 Fisher-Neyman theorem을 exponential 형태로 나타낸 것으로 이해할 수 있다. 이때 임의의 inner product를 사용 가능한데, 여기에서는 dot product로 가정한다. $A(\theta)$는 log-partition function이라고 하며, normalization constant로, 분포의 합이 1이 되도록 한다. $\phi(x)$는 sufficient statistic이다.

$$\begin{aligned} p(x|\theta) &= h(x)\exp(\langle\theta, \phi(x)\rangle - A(\theta)) \\ &= h(x)\exp(\theta^T\phi(x) - A(\theta)) \end{aligned}$$

만약 exponential family에 속하는 어떤 distribution이 존재할 때, exponential의 형태로 정리했을 때, x만 존재하는 항은 $h(x)$, θ만 존재하는 항은 $A(\theta)$, x와 θ가 섞여 있는 항을 $\theta^T\phi(x)$로 식별할 수 있다. $\theta^T\phi(x)$에 대응되는 항을 dot product의 형태로 분리하면 θ와 $\phi(x)$를 식별할 수 있다.

exponential family에 대한 직관적인 개념은 다음과 같이 단순화해서 받아들일 수 있다. 이때 θ를 Natural Parameters라고 하고, 여기에는 추정하려는 매개변수들을 포함시킨다. natural parameter는 해당 probability distribution의 수학적 양상을 결정하는 값이다.

$$p(x|\theta) \propto \exp(\theta^T\phi(x))$$

2. Exponential Family의 Conjugate Prior

bayes's theorem에서 likelihood가 exponential family인 경우, 다음과 같은 conjugate prior를 갖는다. 이렇게 정의되는 것은 그 구조를 동일하게 맞추기 위해서라고 한다. 이때의 conjugate prior 또한 exponential family이다.

$$p(\boldsymbol{\theta}|\boldsymbol{\gamma}) = h_c(\boldsymbol{\theta})\exp\left(\left\langle\begin{bmatrix}\boldsymbol{\gamma}_1\\ \gamma_2\end{bmatrix}, \begin{bmatrix}\boldsymbol{\theta}\\ -A(\boldsymbol{\theta})\end{bmatrix}\right\rangle - A_c(\boldsymbol{\gamma})\right)$$

이때의 γ는 prior distribution을 정하는 natural parameter이다. γ_1은 vector, γ_2는 scalar이다. 이 값을 정하는 것은 실무적 문제이다.

$A_c(\gamma)$는 probability의 총합을 1로 만드는 normalization costant로, 이에 따라 evidence를 고려하지 않고 likelihood와 prior의 곱으로 posterior를 정의할 수 있다. 이때 동일한 구조의 두 distribution의 곱이 다음과 같이 단순 덧셈이 되는 것으로 이해할 수 있다. 즉, natural parameter에 대한 덧셈만이 수행된다.

$$p(\boldsymbol{\theta}|X) \propto \exp\left(\boldsymbol{\theta}^T \sum_{i=1}^{n} \boldsymbol{\phi}(x_i) - nA(\boldsymbol{\theta})\right) \cdot \exp(\boldsymbol{\gamma}_1^T \boldsymbol{\theta} - \gamma_2 A(\boldsymbol{\theta}))$$

$$p(\boldsymbol{\theta}|X) \propto \exp\left(\left(\boldsymbol{\gamma}_1 + \sum_{i=1}^{n} \boldsymbol{\phi}(x_i)\right)^T \boldsymbol{\theta} - (\gamma_2 + n)A(\boldsymbol{\theta})\right)$$

어떤 수학적 대상을 하나 이상의 parameter를 사용해 표현하는 것을 Parameterization(매개변수화)이라고 하고, 그 결과물을 Parametric Form(매개변수 형태)이라고 한다. exponential family 수식은 parametric form이다.

다음은 exponential family에 대해 각 항을 식별하는 예시이다.

Example 6.13 (Gaussian as Exponential Family)

Univariate Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ 를 고려해보도록 하겠습니다. 이때, $\phi(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix}$ 입니다. 그러면 지수족의 정의를 사용하여 아래의 식을 얻을 수 있습니다.

$$p(x|\boldsymbol{\theta}) \propto \exp(\theta_1 x + \theta_2 x^2) \tag{6.109}$$

파라미터 $\boldsymbol{\theta}$ 를 다음과 같이 설정하고

$$\boldsymbol{\theta} = \begin{bmatrix} \frac{\mu}{\sigma^2} & -\frac{1}{2\sigma^2} \end{bmatrix}^\top \tag{6.110}$$

이를 식 (6.109)에 대입하면 다음의 식을 얻을 수 있습니다.

$$p(x|\boldsymbol{\theta}) \propto \exp\left(\frac{\mu x}{\sigma^2} - \frac{x^2}{2\sigma^2}\right) \propto \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right) \tag{6.111}$$

따라서, univariate Gaussian distribution은 sufficient statistic $\phi(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix}$ 와 $\boldsymbol{\theta}$ 로 주어지는 natural parameter 갖는 지수족의 멤버입니다.

Example 6.14 (Bernoulli as Exponential Family)

Example 6.8에서 살펴본 아래의 베르누이 분포를 다시 살펴봅시다.

$$p(x|\mu) = \mu^x (1-\mu)^{1-x}, \quad x \in \{0, 1\} \tag{6.112}$$

이는 지수족 형태로 다음과 같이 작성될 수 있습니다.

$$\begin{aligned} p(x|\mu) &= \exp[\log(\mu^x (1-\mu)^{1-x})] && \text{(6.113a)} \\ &= \exp[x \log \mu + (1-x)\log(1-\mu)] && \text{(6.113b)} \\ &= \exp[x \log \mu - x \log(1-\mu) + log(1-\mu)] && \text{(6.113c)} \\ &= \exp\left[x \log \frac{\mu}{1-\mu} + \log(1-\mu)\right] && \text{(6.113d)} \end{aligned}$$

마지막 식 (6.113d) 는 (6.107)의 지수족 형태와 동일하며

$$\begin{aligned} h(x) &= 1 && \text{(6.114)} \\ \theta &= \log \frac{\mu}{1-\mu} && \text{(6.115)} \\ \phi(x) &= x && \text{(6.116)} \\ A(\theta) &= -\log(1-\mu) = \log(1+\exp(\theta)) && \text{(6.117)} \end{aligned}$$

이라는 것을 관찰할 수 있습니다.

θ 와 μ 의 관계는 invertible 하며, 다음을 만족합니다.

$$\mu = \frac{1}{1+\exp(-\theta)} \tag{6.118}$$

다음은 likelihood가 exponential family일 때 conjugate prior를 구하는 예시이다.

Example 6.15

(6.113d)의 베르누이 분포의 지수족 형태를 다시 살펴봅시다.

$$p(x|\mu) = \exp\left[x \log \frac{\mu}{1-\mu} + \log(1-\mu)\right] \tag{6.121}$$

이 분포의 canonical conjugate prior는 다음의 형태를 갖습니다.

$$p(\mu|\alpha,\beta) = \frac{\mu}{1-\mu} \exp\left[\alpha \log \frac{\mu}{1-\mu} + (\beta+\alpha)\log(1-\mu) - A_c(\boldsymbol{\gamma})\right] \tag{6.122}$$

위 식에서 $\boldsymbol{\gamma} := [\alpha, \beta+\alpha]^\top$ 이고, $h_c(\mu) := \mu/(1-\mu)$ 입니다. (6.122) 식은 다음와 같이 간단하게 정리할 수 있습니다.

$$p(\mu|\alpha,\beta) = \exp[(\alpha-1)\log\mu + (\beta-1)\log(1-\mu) - A_c(\alpha,\beta)] \tag{6.123}$$

위 식을 non-exponential family form에 대입하면 다음과 같으며,

$$p(\mu|\alpha,\beta) \propto \mu^{\alpha-1}(1-\mu)^{\beta-1} \tag{6.124}$$

이는 Beta distribution (6.98)과 동일합니다. Example 6.12에서 우리는 Beta distribution이 베르누이 분포의 conjugate prior라고 가정했었고, 실제로 conjugate prior라는 것을 봤습니다. 이번 예제에서는 지수족 형태의 베르누이 분포의 conjugate prior를 살펴봄으로써 베타 분포를 유도했습니다.

12.6. Transformation of Random Variable

알려져 있는 distribution들이 있지만, 이는 매우 제한적이다. 이에 따라 transformation이 적용된 random variable의 distribution을 확인해보자. 이때 앞서 다룬 것처럼 affine transformation을 적용할 수 있지만, 이런 적용이 어려운 non-linear transformation을 사용하거나 functional form을 얻지 못할(어떤 distribution인지 알 수 없을) 수 있다.

random variable을 변환한 distribution을 얻는 두 가지 접근 방법을 살펴보자. 하나는 CDF를 사용하는 직접적인 방식이고, 다른 하나는 chain rule을 사용하는 방식이다.

12.6.1. Distribution Function Technique

random variable X, Y에 대해 function U에 대해서 $Y = U(X)$가 성립한다고 하자. 다음과 같이 CDF $F_Y(y)$가 정의되고, CDF를 differentiate하면 PDF $f(y)$가 된다. 즉, X에 transformation을 적용했을 때의 distribution을 구할 수 있다. 이때 이런 transformation 이후에는 domain이 바뀔 수 있다.

$$F_Y(y) = P(Y \le y)$$
$$f(y) = \frac{d}{dy} F_Y(y)$$

다음은 distribution function technique의 예시이다.

Example 6.16

확률 변수 X 가 $0 \leq x \leq 1$ 에서 아래의 PDF를 갖는 연속 확률 변수라고 가정해봅시다.

$$f(x) = 3x^2 \tag{6.128}$$

이때, $Y = X^2$ 의 PDF를 찾아봅시다.

함수 f 는 x 에 대한 증가하는 함수이므로, y 의 값은 $[0, 1]$ 구간에 놓여 있습니다. 위에서 언급한 방법을 통해서 Y 의 PDF는 다음과 같이 계산할 수 있습니다.

$$\begin{aligned}
F_Y(y) &= P(Y \leq y) && \text{definition of cdf} && (6.129a)\\
&= P(X^2 \leq y) && \text{transformation of interest} && (6.129b)\\
&= P(X \leq y^{\frac{1}{2}}) && \text{inverse} && (6.129c)\\
&= F_X(y^{\frac{1}{2}}) && \text{definition of CDF} && (6.129d)\\
&= \int_0^{y^{\frac{1}{2}}} 3t^2 \mathrm{d}t && \text{CDF as a definite integral} && (6.129e)\\
&= [t^3]_{y=0}^{t=y^{\frac{1}{2}}} && \text{result of integration} && (6.129f)\\
&= y^{\frac{3}{2}}, \quad 0 \leq y \leq 1 && && (6.129g)
\end{aligned}$$

따라서, Y 의 CDF는 아래와 같습니다.

$$F_Y(y) = y^{\frac{3}{2}}, \quad 0 \leq y \leq 1 \tag{6.130}$$

PDF는 CDF를 미분하여 얻을 수 있습니다.

$$f(y) = \frac{\mathrm{d}}{\mathrm{d}y} F_Y(y) = \frac{3}{2} y^{\frac{1}{2}}, \quad 0 \leq y \leq 1 \tag{6.131}$$

또한 random variable X에 대해 monotonically increasing function CDF $F_X(x)$가 존재할 때, $Y = F_X(x)$로 정의되는 random variable Y는 uniform distribution이다. 이는 uniform distribution에서 데이터를 추출한 뒤, 이 CDF의 inverse를 활용해 데이터를 얻는 식으로 활용할 수 있다.

12.6.2. Change of Variable Technique

1. Univariate Random Variable에 대한 적용

univariate random variable X와 invertible function U, random variable $Y = U(X)$를 생각하자. X가 state $x \in [a, b]$를 가질 때 Y의 PDF는 다음과 같다.

$$f(y) = f_x(U^{-1}(y)) \cdot \left| \frac{\mathrm{d}}{\mathrm{d}y} U^{-1}(y) \right| \tag{6.143}$$

Y에 대한 CDF를 다음과 같이 유도할 수 있다. U가 invertible이므로 strictly increasing 또는 strictly decreasing하고, 이에 따라 inverse를 양변에 적용할 수 있다. strictly decreasing일 경우에는 증명 과정이 약간 다른데, inverse를 취할 때 부등호를 반대로 바꾸고, 이후 1에서 빼는 수식으로 수정하는 것으로 비슷하게 증명 가능하다.

$$\begin{aligned}
&F_Y(y) = P(Y \leq y)\\
&P(Y \leq y) = P(U(X) \leq y)\\
&P(U(X) \leq y) = P(U^{-1}(U(X)) \leq U^{-1}(y)) = P(X \leq U^{-1}(y))\\
&P(X \leq U^{-1}(y)) = \int_a^{U^{-1}(y)} f(x)\mathrm{d}x\\
&F_Y(y) = \int_a^{U^{-1}(y)} f(x)\mathrm{d}x
\end{aligned}$$

이를 y에 대해 differentiate하면 다음과 같다. 이때 PDF는 항상 0보다 크거나 같아야 하는데, U가 strictly decreasing일 경우 음수가 되므로 절댓값을 씌운다.

$$f(y) = \frac{\mathrm{d}}{\mathrm{d}y} F_Y(y) = \frac{\mathrm{d}}{\mathrm{d}y} \int_a^{U^{-1}(y)} f(x)\mathrm{d}x$$
$$= f_x(U^{-1}(y)) \cdot \left| \frac{\mathrm{d}}{\mathrm{d}y} U^{-1}(y) \right|$$

2. Multivariate Random Variable에 대한 적용
$f_x(\boldsymbol{x})$ 를 multivariate random variable X의 PDF라고 하자. 만약 vector-valued function $\boldsymbol{y} = U(\boldsymbol{x})$가 $\boldsymbol{x}$의 domain 내의 모든 값에서 differentiable/invertible이면, $\boldsymbol{y}$ 에 대응되는 값에 대해 Y의 PDF는 다음과 같다.

$$f(\boldsymbol{y}) = f_x(U^{-1}(\boldsymbol{y})) \cdot \left| \det \left(\frac{\partial}{\partial \boldsymbol{y}} U^{-1}(\boldsymbol{y}) \right) \right|$$

다음은 bivariate random variable에 대한 예시이다.

Example 6.17

State $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ 를 갖는 bivariate random variable X 를 고려해보겠습니다. 이 확률 변수의 PDF는 아래와 같다고 가정합니다.

$$f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) = \frac{1}{2\pi} \exp\left(-\frac{1}{2} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) \tag{6.145}$$

여기서 Theorem 6.16의 change-of-variable technique를 사용하여 확률 변수의 linear transformation의 effect를 유도해보도록 하겠습니다.

Transformation matrix $\boldsymbol{A} \in \mathbb{R}^{2\times 2}$ 는 다음과 같이 정의되었다고 간주합니다.

$$\boldsymbol{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{6.146}$$

이때, $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ 의 state를 갖는 변형된 bivariate random variable Y 의 PDF를 찾아보겠습니다.

Change of variables를 위해서 $\boldsymbol{x}$ 의 inverse transformation이 y 에 대한 함수로 필요합니다. 여기서는 linear transformation을 간주하므로, inverse transformation은 역행렬로 주어집니다. 2×2 행렬에서, 공식을 사용하면 명시적으로 다음과 같이 얻을 수 있습니다.

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \boldsymbol{A}^{-1} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \tag{6.147}$$

대응되는 PDF는 다음과 같이 주어집니다.

$$f(\boldsymbol{x}) = f(\boldsymbol{A}^{-1}\boldsymbol{y}) = \frac{1}{2\pi} \exp\left(-\frac{1}{2} \boldsymbol{y}^\top \boldsymbol{A}^{-\top} \boldsymbol{A}^{-1} \boldsymbol{y} \right) \tag{6.148}$$

행렬과 벡터 곱에서 벡터에 대한 편도함수는 행렬 그 자체 이므로, 다음과 같습니다.

$$\frac{\partial}{\partial \boldsymbol{y}} \boldsymbol{A}^{-1}\boldsymbol{y} = \boldsymbol{A}^{-1} \tag{6.149}$$

역행렬의 determinant는 determinant의 역수이므로, Jacobian matrix의 determinant는 다음과 같습니다.

$$\det\left(\frac{\partial}{\partial \boldsymbol{y}} \boldsymbol{A}^{-1}\boldsymbol{y} \right) = \frac{1}{ad - bc} \tag{6.150}$$

이제 식 (6.148)과 (6.150)을 곱하여 Theorem 6.16의 change-of-variable 공식을 적용하면 다음의 식을 얻을 수 있습니다.

$$f(\boldsymbol{y}) = f(\boldsymbol{x}) \left| \det\left(\frac{\partial}{\partial \boldsymbol{y}} \boldsymbol{A}^{-1}\boldsymbol{y} \right) \right| \tag{6.151a}$$
$$= \frac{1}{2\pi} \exp\left(-\frac{1}{2} \boldsymbol{y}^\top \boldsymbol{A}^{-\top} \boldsymbol{A}^{-1} \boldsymbol{y} \right) |ad - bc|^{-1} \tag{6.151b}$$

13. Continuous Optimization

본 장에서는 ML 모델들을 training하기 위한 수치적 방법들을 다룬다. 즉, objective function이 주어졌을 때, optimization 알고리즘을 적용해 최적의 parameter 집합을 찾아본다. 특히 두 가지 main branch인 Unconstrained Optimization(제약이 없는 최적화)과 Constrained Optimization(제약이 있는 최적화)을 다룬다.

이때 objective function이 항상 미분가능하고, 각 위치에서 gradient를 계산할 수 있다고 가정한다. ML에서 대부분의 objective function은 최솟값이 최적의 값이다. 직관적으로 최적의 값을 찾는 것은 목적 함수의 valleys를 찾는

것과 같으며, gradient는 이 valleys에서 오르막길로 이끌어주는 역할을 합니다. 이러한 아이디어는 (gradient의 반대 방향) 내리막길로 이동하여 가장 깊은 지점을 찾는 것입니다.

차수가 낮은 polynomial에 대해서는 직접 differientate해서 최적의 지점을 찾을 수 있지만, 이런 analytic solution을 찾기 어려운 많은 경우에 대해서는 gradient를 활용해 최적의 지점을 찾아야 한다.

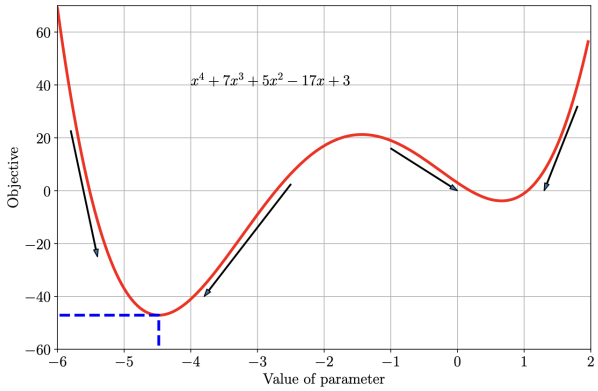


Figure 7.2 Example objective function. Negative gradients are indicated by arrows, and the global minimum is indicated by the dashed blue line.

13.1. Gradient Descent

13.1.1. Gradient Descent

Gradient Descent(경사하강법)은 first-order optimization 알고리즘으로, function $f : \mathbb{R}^n \to \mathbb{R}$의 local minimum을 찾기 위해, 초기 추측값 x_0에서 시작해 다음 수식을 iterate한다. 이때 f에 대한 gradient는 row vector로 정의했다.

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \gamma_i((\nabla f)(x_i))^T$$

적절한 scalar γ_i를 사용했을 때, $f(x_0) \geq f(x_1) \geq \cdots$은 local minimum으로 수렴한다. 이때의 γ는 Step Size 또는 **Learning Rate**라고 한다. learning rate가 너무 작으면 수렴하는 데에 너무 오래 걸리고, learning rate가 너무 크면 수렴하지 못하거나, overshoot되거나, 발산하게 된다.

더 구체적으로는, standard gradient descent는 batch optimization method로, 이때의 batch는 데이터셋 전체를 말한다. 즉, $n = 1, \cdots, N$으로 data point들이 주어졌을 때, 다음과 같이 매번 데이터셋 전체에 대한 gradient를 활용한다.

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \gamma_i \sum_{n=1}^{N}((\nabla f_n)(x_i))^T$$

univariate function에 대해 gradient는 해당 지점에서 접선의 기울기로 생각할 수 있다. multivariate function에서는 contour line에 orthogonal한 방향으로 생각할 수 있다. **Contour line(등고선)**은 3차원 이상의 다변수 함수에서 함수의 결괏값이 동일한 지점들을 연결하여 2차원 평면에 투영한 선을 의미한다.

Adaptive Gradient Descent에서는 iteration마다 step size를 rescale한다. 다음과 같은 heuristic으로 동작할 수 있다.

- Gradient step 이후 함수 값이 증가하면 step-size가 너무 크다는 것이므로, 이 step을 다시 되돌리고 step-size를 감소시킨다.
- 함수값이 감소한다는 것은 step-size가 더 커져도 된다는 것을 의미하므로, step-size를 증가시킨다.

다음은 gradient descent의 예시이다.

Example 7.1

2차원의 quadratic function 과 이 함수의 gradient가 다음과 같습니다.

$$f\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}\right) = \frac{1}{2}\begin{bmatrix}x_1\\x_2\end{bmatrix}^\top \begin{bmatrix}2 & 1\\1 & 20\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} - \begin{bmatrix}5\\3\end{bmatrix}^\top \begin{bmatrix}x_1\\x_2\end{bmatrix} \tag{7.7}$$

$$\nabla f\left(\begin{bmatrix}x_1\\x_2\end{bmatrix}\right) = \begin{bmatrix}x_1\\x_2\end{bmatrix}^\top \begin{bmatrix}2 & 1\\1 & 20\end{bmatrix} - \begin{bmatrix}5\\3\end{bmatrix}^\top \tag{7.8}$$

초기 위치(initial location) $\boldsymbol{x}_0 = [-3, -1]^\top$ 에서 시작해서, (7.6) 식을 반복하면 minimum value로 수렴하는 sequence of estimates를 얻을 수 있습니다 (아래의 Figure 7.3 참조).

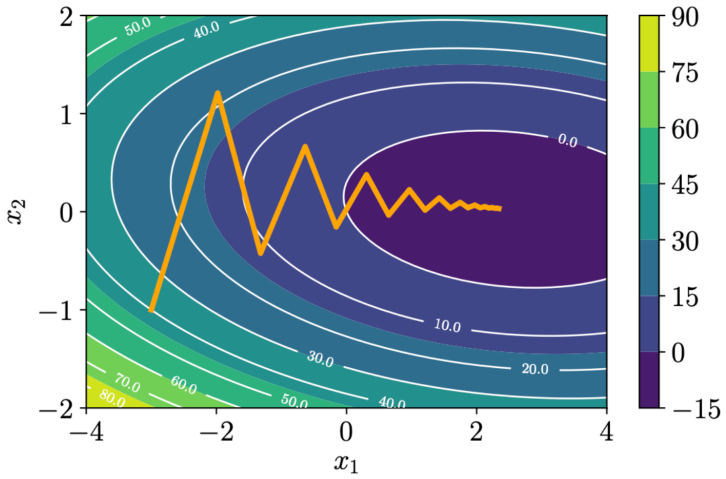


Figure 7.3 Gradient descent on a two-dimensional quadratic surface (shown as a heatmap). See Example 7.1 for a description.

위 그림에서 (7.6)의 gradient descent를 수행하여, $\boldsymbol{x}_0$ 에서 북동쪽 방향으로 $\boldsymbol{x}_1 = [-1.98, 1.21]^\top$ 에 이어지는 것을 볼 수 있습니다. Gradient descent를 반복 수행하면서 $\boldsymbol{x}_2 = [-1.32, -0.42]^\top$ 등을 얻을 수 있습니다.

gradient descent는 수렴이 비교적 느린 optimization으로, local minimum에 근접할수록 수렴이 느려진다.

$Ax = b$에 대해서도 $||Ax - b|| = (Ax + b)^T(Ax + b)$에 gradient descent를 적용해 해를 구할 수 있지만, analytic solution이 이미 존재하고, 매우 느리게 수렴할 수 있다.

13.1.2. Gradient Descent with Momentum

Gradient descent with momentum은 이전 iteration에서 발생한 것을 기억하는 additional term을 도입하는 기법이다. 이는 다음과 같이 정의된다. 즉, 직전 값을 구하기 위해 사용한 gradient와 momentum 값을, scalar α로 그 비율을 지정해 함께 사용한다. 이때 $\alpha \in [0, 1]$이다.

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \gamma_i((\nabla f)(\boldsymbol{x}_i))^\top + \alpha\nabla\boldsymbol{x}_i$$
$$\nabla\boldsymbol{x}_i = \boldsymbol{x}_i - \boldsymbol{x}_{i+1} = \alpha\nabla\boldsymbol{x}_{i-1} - \gamma_{i-1}((\nabla f)(\boldsymbol{x}_{i-1}))^\top$$

직전 값만을 저장해 두고 활용하지만, 해당 값에는 그 이전 값들이 반영되어 있다.

gradient descent는 curvature(곡률)에 따라서 매우 느릴 수 있으므로, 약간의 memory를 사용하는 것이다. 이때 추가된 term은 진동을 감쇠시키고 gradient updates가 부드러워지도록 한다. 공으로 비유하자면, momentum term은 방향을 바꾸지 않으려는 무거운 공의 특징이라고 생각할 수 있다.

13.1.3. Stochastic Gradient Descent

Stochastic Gradient Descent(SGD, 확률적 경사하강법) 또는 Mini-batch Gradient Descent는 $n = 1, \cdots, N$인 data point들이 존재하고 각각에 대한 loss L_n이 있을 때, 모든 L_n을 사용하는 대신 L_n의 subset을 선택해 gradient descent를 수행하는 기법을 말한다. 즉, 모든 sample에 대한 gradient를 계산하는 것은 비용(VRAM, latency 등)이 크므로, gradient에 대한 noisy approximation을 사용하는 것이다.

큰 mini-batch size는 더 정확한 gradient approximation을 가능하게 하고, parameter 업데이트의 variacne를

감소시킨다. 이는 좀 더 안정적인 수렴을 이끌어내지만, 각 gradient 계산 비용이 크다. 반면 더 작은 mini-batche size를 활용하면 빠른 연산이 가능하고, gradient approximation의 noise로 인해 bad local optima를 벗어날 수 있다. 또한 이런 optimization의 최종적인 목적은 generalization performance를 향상시키는 것이고, objective function의 최소값을 정확하게 추정하는 것이 아니기 때문에 그 의의가 더 크다.

또한 subset만을 활용해 optimization해도 거의 확실하게 local minimum으로 수렴한다고 한다.

13.2. Lagrange Multiplier Method

13.2.1. Lagrange Multiplier Method

1. Lagrange Multiplier Method

Lagrange Multiplier Method(라그랑주 승수법)는 보조 변수인 lagrange multiplier λ 를 추가로 사용하는 constrained optimization 기법으로, objective function의 contour line과 **Constraint(제약 조건)** 곡선이 최적점에서 서로 접한다는 기하학적 성질을 활용한다. objective function $f : \mathbb{R}^D \to \mathbb{R}$ 와 $i = 1, \dots, m$ 에 대해 $g_i : \mathbb{R}^D \to \mathbb{R}$ 인 constraints $g_i(\boldsymbol{x}) \leq 0$ 이 존재하는 경우, optimization problem은 다음과 같다.

$$\begin{aligned} &\min_{\boldsymbol{x}} f(\boldsymbol{x}) \\ &\text{subject to } g_i(\boldsymbol{x}) \leq 0 \quad i = 1, \dots, m \end{aligned}$$

lagrange multiplier method에서는 다음과 같이 lagrange multiplier λ 를 추가한 **Lagrangian**에 대한 optimization을 적용하는 것으로 constrained optimization problem을 푼다. 이때 λ_i 는 $\lambda_i \geq 0$ 인 scalar이고, λ 는 이들로 구성한 column vector이다. $\boldsymbol{g}(\boldsymbol{x})$ 또한 $g_i(\boldsymbol{x})$ 로 구성한 column vector이다. 즉, $g(x)$가 vector 또는 matrix이면 λ도 column vector가 된다.

$$\begin{aligned} \mathcal{L}(\boldsymbol{x}, \lambda) &= f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i g_i(\boldsymbol{x}) \\ \mathcal{L}(\boldsymbol{x}, \lambda) &= f(\boldsymbol{x}) + \lambda^\top \boldsymbol{g}(\boldsymbol{x}) \end{aligned}$$

$\lambda \geq 0$ 제약이 추가되는 이유는, constraints의 경계점 $\boldsymbol{x}^*$ 에서 objective function이 감소하는 방향 $-\nabla f(\boldsymbol{x}^*)$ 이 feasible region을 벗어나는 방향인($g(\boldsymbol{x}) \leq 0$ 이 조건이므로 gradient를 생각해 보면 당연하다.) $\nabla g_i(\boldsymbol{x}^*)$ 와 동일한 방향을 가리켜야 하기 때문이다. $f(\boldsymbol{x})$ 가 최솟값에 도달했다는 것은, constraints를 위반하지 않는 선에서는 더 이상 $f(\boldsymbol{x})$ 를 감소시킬 수 있는 방향이 존재하지 않는다는 의미이다. 이는 $f(\boldsymbol{x})$ 를 감소시키는 방향인 $-\nabla f(\boldsymbol{x}^*)$ 가 feasible region을 벗어나는 방향인 $\nabla g(\boldsymbol{x}^*)$ 와 일직선상에서 같은 방향을 가리키기 때문에 이동이 불가능해진 상태가 됨을 의미한다.

최적점에서 objective function $f(\boldsymbol{x})$ 의 contour line은 constraints 곡면 $\boldsymbol{g}(\boldsymbol{x}) = 0$ 에 접해야 하고, 이때의 두 gradient는 서로 평행하다(부등식 제약 조건 $\boldsymbol{g}(\boldsymbol{x}) \leq 0$ 이 활성화된 경우 방향이 반대이고 상수배이다.). 이때 contour line과 constraints가 접하는 이유는, 만약 constraints가 contour line을 교차해 지나간다면, 이는 constraints를 따라 더 이동했을 때 더 크거나 작은 다른 contour line으로 진입할 수 있음을 의미하기 때문이다. 다중 제약 조건의 경우 objective function의 gradient는 constraints gradient들의 선형 결합으로 표현된다.

만약 constraints가 $h(\boldsymbol{x}) = 0$ 로 등식이라면, 이는 두 개의 부등식 constraints $h(\boldsymbol{x}) \leq 0$ 과 $h(\boldsymbol{x}) \geq 0$ 으로 분리할 수 있다. 즉, $h(\boldsymbol{x}) \leq 0$ 과 $-h(\boldsymbol{x}) \leq 0$ 이 있고, 각각의 lagrange multiplier를 λ_1, λ_2 라 하면 $\mathcal{L}(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) + \lambda h(\boldsymbol{x})$ 이고 $\lambda = \lambda_1 - \lambda_2$ 가 된다. 즉, 등호에 대한 λ 는 부호 제약이 없게 된다.

2. Lagrangian duality

다음 수식과 같이 Lagrangian duality를 적용해 이를 λ 에 대한 문제로 바꿀 수 있다. optimization에서 **Duality(쌍대)**는 기존의 변수인 **Primal Variable** $\boldsymbol{x}$ 에 대한 optimization problem(Primal Problem)을 또 다른 변수인 **Dual Variables** λ 에 대한 optimization problem(Dual Problem)로 변환하는 것을 말한다.

$$\max_{\lambda \in \mathbb{R}^m} D(\lambda)$$
$$\text{subject to } \lambda \geq 0$$

이때 dual objective function $D(\lambda)$ 는 $D(\lambda) = \min_{\boldsymbol{x} \in \mathbb{R}^D} \mathcal{L}(\boldsymbol{x}, \lambda)$ 로 정의된다. 이때 다음과 같은 Minimax Inequality(Weak Duality)에 의해 $\lambda \geq 0$ 일 때 Lagrangian $\mathcal{L}(\boldsymbol{x}, \lambda)$ 는 $\boldsymbol{x}$ 에 대한 기존 문제의 lower bound가 된다. 즉, λ 에 대한 문제인 $\max_{\lambda \geq 0} \min_{\boldsymbol{x} \in \mathbb{R}^D} \mathcal{L}(\boldsymbol{x}, \lambda)$ 를 푸는 것이 기존 문제 $\min_{\boldsymbol{x} \in \mathbb{R}^D} \max_{\lambda \geq 0} \mathcal{L}(\boldsymbol{x}, \lambda)$ 에 대한 lower bound를 구하는 것과 같다.

$$\min_{\boldsymbol{x} \in \mathbb{R}^D} \max_{\lambda \geq 0} \mathcal{L}(\boldsymbol{x}, \lambda) \geq \max_{\lambda \geq 0} \min_{\boldsymbol{x} \in \mathbb{R}^D} \mathcal{L}(\boldsymbol{x}, \lambda)$$

이때 inner product에 대해서 $\lambda^\top \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{x})^\top \lambda$ 가 성립하므로, $D(\lambda)$ 는 다음과 같다. 즉, 이는 λ 에 대한 affine function들의 pointwise minimum이므로 항상 concave하고, global maximum이 존재한다.

$$D(\lambda) = \min_{\boldsymbol{x} \in \mathbb{R}^D} \left(f(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})^\top \lambda \right)$$

objective function $f(\boldsymbol{x})$ 나 constraints $g(\boldsymbol{x})$ 가 non-convex인 경우, local minimum에 빠지기 쉬워 global minimum을 찾는 것이 수학적으로 어렵다. 반면 $D(\lambda)$ 는 affine function의 pointwise minimum으로, $f(\boldsymbol{x})$ 나 $g(\boldsymbol{x})$ 와 관계없이 항상 concave가 된다(concave임이 증명되어 있다.).

하지만 일반적인 경우에 대한 lagrange multiplier method는 weak duality에 의해 **lower bound를 구하는 것일 뿐 항상 최적해가 도출되지는 않고, 이때 KKT는 최적해에 대한 필요조건이다**. 즉, primal problem의 해와 dual problem의 해의 차이를 의미하는 Duality Gap이 존재할 수 있다. 반면 해당 problem이 **convex optimization problem**인 경우 strong duality이므로 **항상 동일한 최적해가 도출되고, KKT는 최적해에 대한 충분조건이다**.

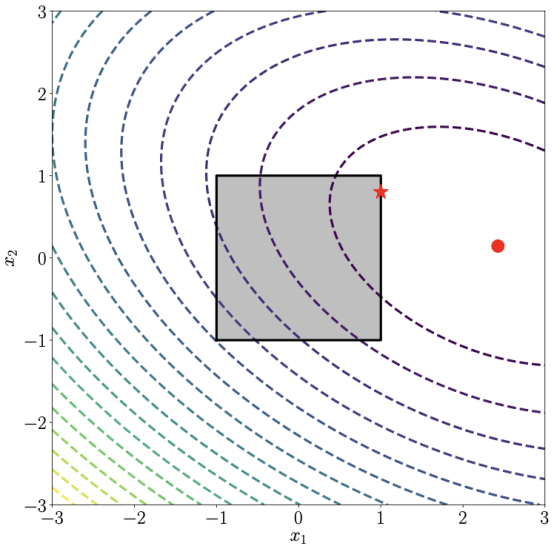


Figure 7.4 Illustration of constrained optimization. The unconstrained problem (indicated by the contour lines) has a minimum on the right side (indicated by the circle). The box constraints ($-1 \leqslant x \leqslant 1$ and $-1 \leqslant y \leqslant 1$) require that the optimal solution is within the box, resulting in an optimal value indicated by the star.

optimization problem에서 subject to는 " 라는 제약 조건 하에"라는 뜻으로, 목적 함수를 최소화하거나 최대화할 때 지켜야 할 constraints를 명시할 때 사용한다.

만약 여러 개의 constraints가 존재한다면 다음 수식과 같이 각각의 항을 추가할 수 있다.

$$\mathcal{L}(x, \lambda, \nu) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{j=1}^{n} \nu_j h_j(x)$$

13.2.2. KKT Conditions

KKT(Karush-Kuhn-Tucker) Conditions는 constrained optimization problem에서 어떤 해가 최적해인지에 대한 필요조건이다. 즉, KKT conditions가 성립한다고 반드시 최적해인 것은 아니지만, 최적해의 경우 이 조건들이 성립한다. 이는 다음과 같은 optimization problem에 적용 가능하다.

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & g_i(\boldsymbol{x}) \leq 0, \quad i = 1, \dots, m \\ & h_j(\boldsymbol{x}) = 0, \quad j = 1, \dots, p \end{aligned}$$

KKT conditions는 다음과 같은 4가지 조건으로 구성된다.

- Stationary (정상성): Lagrangian $\mathfrak{L}$ 의 미분값이 최적점에서 0이 되어야 한다.

$$\nabla f(\boldsymbol{x}^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(\boldsymbol{x}^*) + \sum_{j=1}^{p} \nu_j^* \nabla h_j(\boldsymbol{x}^*) = 0$$

- Primal Feasibility (primal problem의 실행 가능성): 최적해 $\boldsymbol{x}^*$는 primal problem의 모든 constraints를 만족해야 한다.

$$g_i(\boldsymbol{x}^*) \leq 0, \quad h_j(\boldsymbol{x}^*) = 0$$

- Dual Feasibility (dual problem의 실행 가능성): lagrange multiplier는 항상 0보다 크거나 같아야 한다.

$$\lambda_i^* \geq 0$$

- Complementary Slackness (상보 여유성): constraints g_i 와 그에 대응하는 lagrange multiplier λ_i 중 하나는 반드시 0이어야 한다. 즉, constraints가 활성화(Active)되어 $g_i(\boldsymbol{x}^*) = 0$이거나, 비활성화되어 $\lambda_i^* = 0$이어야 한다는 것을 의미한다.

$$\lambda_i^* g_i(\boldsymbol{x}^*) = 0$$

13.3. Convex Optimization

convex optimization problem에 대해 알아보고, 그 예시를 살펴보자.

13.3.1. Convex Optimization

1. Convex Set and Convex Function
집합 $\mathcal{C}$의 vector $x, y \in \mathcal{C}$와, 임의의 scalar $0 \leq \theta \leq 1$에 대해 다음 식이 성립할 때, $\mathcal{C}$를 **Convex Set**이라고 한다. 집합의 두 요소를 연결하는 직선이 그 집합 내에 존재하는 집합으로도 이해할 수 있다.

$$\theta x + (1 - \theta) y \in \mathcal{C}$$

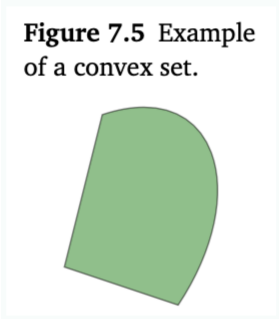


Figure 7.5 Example of a convex set.

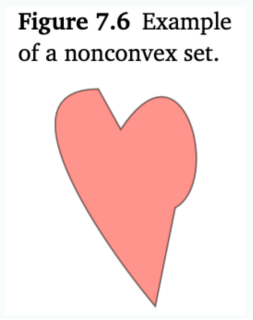


Figure 7.6 Example of a nonconvex set.

function $f : \mathbb{R}^D \to \mathbb{R}$의 domain이 convex set일 때, f의 domain에 있는 임의의 두 vector x, y와 임의의 scalar $0 \leq \theta \leq 1$에 대해 다음이 성립하면 f를 **Convex Function(볼록 함수)**이라고 한다. 또한 convex function의 negative를 **Concave Function(오목 함수)**이라고 한다.

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

function 위 어떤 두 점 간의 직선이 그 function value들보다 항상 위에 있는 함수로도 이해할 수 있다. 즉, f가 미분가능할 때 임의의 두 점 x, y에 대해 다음이 성립하면 convex이다. 또한 f가 두 번 미분가능하다면 hessian $\nabla_x^2 f(x)$가 positive semidefinite이면 f는 convex이다.

$$f(y) \geq \nabla_x f(x)^T (y - x) + f(x)$$

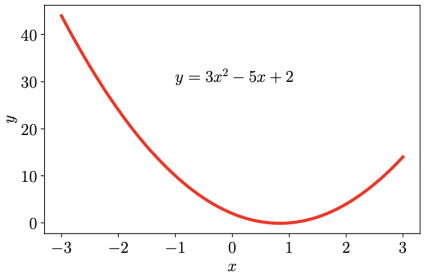


Figure 7.7 Example of a convex function.

convex function들의 nonnegative weighted sum도 convex function이다. 즉, convex function f_1, f_2가 있고, nonnegative scalar $\alpha, \beta \geq 0$이 있을 때 다음 $\alpha f_1(x) + \beta f_2(x)$도 convex function이다. 즉, function 간 덧셈과 nonnegative scalar 곱은 convexity에 대해 닫혀 있으므로, 이를 활용해 convexity를 확인하는 것이 좋다. 이런 nonnegative weighted sum을 취하는 모든 부등식을 Jensen's Inequality라고 한다.

2. Convex Optimization Problem

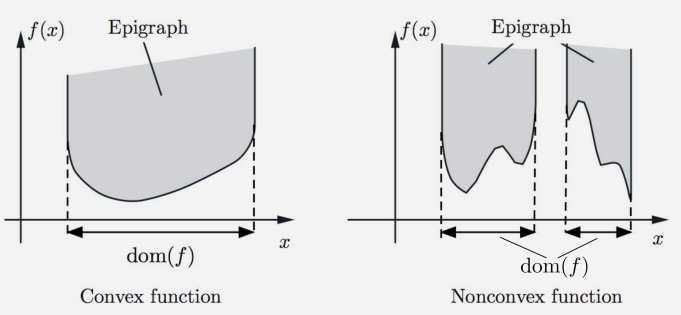
function f가 convex function이고, g, h를 포함하는 constraints가 convex일 때, 이 optimization을 **Convex Optimization Problem**이라고 한다. 즉, 다음 조건을 만족하는 constrained optimization problem을 convex optimization problem이라고 한다.

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & g_i(\boldsymbol{x}) \leq 0 \quad \text{for all} \quad i = 1, \ldots, m \\ & h_j(\boldsymbol{x}) = 0 \quad \text{for all} \quad j = 1, \ldots, n \\ \text{where all functions } & f(\boldsymbol{x}) \text{ and } g_i(\boldsymbol{x}) \text{ are convex functions,} \\ \text{and all } & h_j(\boldsymbol{x}) = 0 \text{ are convex sets.} \end{aligned}$$

convex optimization problem의 경우 global optimality가 보장된다. 또한 Strong Duality를 가지는데, 이는 dual problem의 최적해가 primal problem의 최적해와 같다는 것을 의미한다. 이때 KKT conditions는 해가 최적해이기 위한 충분조건이다. 즉, KKT conditions을 만족하면 그 해는 최적해이다.

$D(\lambda) = \min_{\boldsymbol{x} \in \mathbb{R}^D} \mathcal{L}(\boldsymbol{x}, \lambda)$ 를 구하는 것은 constraints가 존재하지 않는 optimization problem이고, $\mathcal{L}(\boldsymbol{x}, \lambda)$ **가 미분 가능한 convex function인 경우 differentiate해서 0인 지점을 찾아 활용할 수 있다(KKT의 stationary). 또한 이때 기존의 constraint를 만족하는지도 확인해야 한다.**

다음 그림과 같이 convex set은 convex function을 filling in하는 집합으로도 생각할 수 있다. 이렇게 채워서 만든 집합을 convex function의 Epigraph라고 한다.



다음은 convex optimization problem의 예시이다. quadratic programming에서 Q는 positive definite이고, x^TQx는 convex이다. 즉, convex optimization problem의 경우 이렇게 $\mathcal{L}(x, \lambda)$를 구하고, 미분해서 0이 되는 식을 얻은 뒤, x를 제거해 λ에 대한 식으로 정리할 수 있다.

Linear Programming

위에서 살펴봤던 함수들이 모두 linear라고 간주한다면, 다음과 같이 표현할 수 있습니다.

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^d} \quad & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \end{aligned} \tag{7.39}$$

위 식에서 $\boldsymbol{A} \in \mathbb{R}^{m\times d}$ 그리고 $\boldsymbol{b} \in \mathbb{R}^m$ 입니다. 이를 *linear program*(선형계획법) 이라고 합니다. 여기서는 d 개의 변수와 m 개의 선형 제약(linear constaints)를 가지고 있습니다.

위 문제에 Lagrangian은 다음과 같이 주어집니다.

$$\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{\lambda}^\top (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \tag{7.40}$$

여기서 $\boldsymbol{\lambda} \in \mathbb{R}^m$ 은 non-negative인 라그랑주 승수의 벡터입니다. $\boldsymbol{x}$ 에 대응되는 항들로 정리하면 다음과 같습니다.

$$\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = (\boldsymbol{c} + \boldsymbol{A}^\top \boldsymbol{\lambda})^\top \boldsymbol{x} - \boldsymbol{\lambda}^\top \boldsymbol{b} \tag{7.41}$$

위 식을 통해 $\boldsymbol{x}$ 에 대한 $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ 의 도함수를 구하고, 0과 같다고 설정하면 다음의 식을 얻을 수 있습니다.

$$\boldsymbol{c} + \boldsymbol{A}^\top \boldsymbol{\lambda} = \boldsymbol{0} \tag{7.42}$$

따라서, dual Lagrangian 은 $\mathfrak{D}(\boldsymbol{\lambda}) = -\boldsymbol{\lambda}^\top \boldsymbol{b}$ 입니다.

$\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ 의 도함수가 0이기 때문에 제약이 있을 뿐만 아니라, $\boldsymbol{\lambda} \geq \boldsymbol{0}$ 이므로 결국 다음과 같은 dual optimization problem 이 만들어집니다.

$$\begin{aligned} \max_{\boldsymbol{\lambda}\in\mathbb{R}^m} \quad & -\boldsymbol{b}^\top \boldsymbol{\lambda} \\ \text{subject to} \quad & \boldsymbol{c} + \boldsymbol{A}^\top \boldsymbol{\lambda} = \boldsymbol{0} \\ & \boldsymbol{\lambda} \geq \boldsymbol{0} \end{aligned} \tag{7.32}$$

이 또한 m 개의 변수를 갖는 선형계획법입니다. 여기서 m 또는 d 중에 어느 것이 더 큰지에 따라 (7.39)의 primal 이나 (7.43)의 dual 중 선택하여 풀면 됩니다. 여기서 d 는 primal program 에서 변수의 갯수이며 m 은 primal program에서 제약의 갯수입니다.

Quadratic Programming

이번에는 제약 조건이 affine인 convex quadratic objective function 케이스를 살펴보도록 하겠습니다. 즉, 다음과 같습니다.

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^d} \quad & \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^\top \boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \end{aligned} \tag{7.45}$$

여기서 $\boldsymbol{A} \in \mathbb{R}^{m\times d}, \boldsymbol{b} \in \mathbb{R}m$ 그리고 $\boldsymbol{c} \in \mathbb{R}^d$ 입니다.

Square symmetric matrix $\boldsymbol{Q} \in \mathbb{R}^{d\times d}$ 는 positive definite 이며, 따라서 목적 함수는 convex 입니다. 이를 *quadratic program* (2차계획법) 이라고 합니다. 여기에는 d 개의 변수와 m 개의 선형 제약이 있습니다.

Lagrangian은 다음과 같이 주어집니다.

$$\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{\lambda}^\top(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) \tag{7.48a}$$

$$= \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + (\boldsymbol{c} + \boldsymbol{A}^\top\boldsymbol{\lambda})^\top \boldsymbol{x} - \boldsymbol{\lambda}^\top \boldsymbol{b} \tag{7.48b}$$

$\boldsymbol{x}$ 에 대한 $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ 의 도함수를 구해 0과 같다고 설정하면, 다음의 식을 얻을 수 있습니다.

$$\boldsymbol{Q}\boldsymbol{x} + (\boldsymbol{c} + \boldsymbol{A}^\top\boldsymbol{\lambda}) = 0 \tag{7.49}$$

$\boldsymbol{Q}$ 가 invertible 하다고 가정한다면, 다음의 식으로 $\boldsymbol{x}$ 를 계산할 수 있습니다.

$$\boldsymbol{x} = -\boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^\top\boldsymbol{\lambda}) \tag{7.50}$$

식 (7.50)을 primal Lagrangian $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ 에 치환하면, 다음의 dual Lagrangian을 얻을 수 있습니다.

$$\mathfrak{D}(\boldsymbol{\lambda}) = -\frac{1}{2}(\boldsymbol{c} + \boldsymbol{A}^\top\boldsymbol{\lambda})^\top \boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^\top\boldsymbol{\lambda}) - \boldsymbol{\lambda}^\top \boldsymbol{b} \tag{7.51}$$

따라서, dual optimization problem은 다음과 같이 주어집니다.

$$\begin{aligned} \max_{\boldsymbol{\lambda}\in\mathbb{R}^m} \quad & -\frac{1}{2}(\boldsymbol{c} + \boldsymbol{A}^\top\boldsymbol{\lambda})^\top \boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^\top\boldsymbol{\lambda}) - \boldsymbol{\lambda}^\top \boldsymbol{b} \\ \text{subject to} \quad & \boldsymbol{\lambda} \geq \boldsymbol{0} \end{aligned} \tag{7.52}$$

12장에서 quadratic programming을 머신러닝에 적용한 사례를 살펴봅니다.

13.3.2. Legendre-Fenchel Transform and Convex Conjugate

1. Hyperplane

Hyperplane(초평면)은 geometry 및 linear algebra에서, 주어진 n 차원 공간보다 한 차원 낮은 $(n-1)$ 차원의 affine subspace 또는 subspace(intercept가 영벡터인 경우)를 의미한다. $\mathbb{R}^n$에서의 hyperplane은 다음과 같은 집합으로 정의된다. 이때 $\boldsymbol{w} \in \mathbb{R}^n$은 영벡터가 아닌 vector로, hyperplane과 orthogonal하다. b는 affine subspace를 나타내는 scalar이다.

$$H = \{\boldsymbol{x} \in \mathbb{R}^n \mid \langle \boldsymbol{w}, \boldsymbol{x}\rangle + b = \boldsymbol{w}^T\boldsymbol{x} + b = 0\}$$

즉, hyperplane을 inner product를 사용해 정의할 수 있다. 풀어서 쓰면 $w_1x_1 + w_2x_2 + \cdots + w_nx_n + b = 0$ 이고, 이를 만족하는 $\boldsymbol{x}$의 집합은 최대 dimension이 $n-1$인 affine subspace이다. 이때 $\boldsymbol{w}$를 해당 affine space에 대한 Normal Vector(법선 벡터), b를 Intercept(절편) 또는 Bias라고 한다. 특히 normal vector는 해당 affine space와 orthogonal한데, 이는 affine space 위의 임의의 두 vector $\boldsymbol{x}_1, \boldsymbol{x}_2$에 대해 정리하면 $\boldsymbol{w}^T(\boldsymbol{x}_1 - \boldsymbol{x}_2) = 0$ 이 성립하는 것으로 증명할 수 있다.

또한 intercept는 $\boldsymbol{w}$의 방향으로 평행이동한 정도를 나타낸다. 어떤 원점을 지나는 hyperplane $\boldsymbol{w}^T\boldsymbol{x} = 0$와 해당 hyperplane 위의 임의의 점 $\boldsymbol{x}_0$을 생각하자. $\boldsymbol{x}_0$를 임의의 vector $\boldsymbol{u}$만큼 점을 평행이동시키면(모든 점이 이동한다.) $\boldsymbol{x}_0' = \boldsymbol{x}_0 + \boldsymbol{u}$이 되고, 기존 hyperplane 수식을 활용하면 $\boldsymbol{w}^T\boldsymbol{x}_0 = \boldsymbol{w}^T(\boldsymbol{x}_0' - \boldsymbol{u}) = \boldsymbol{w}^T\boldsymbol{x}_0' + b = 0$이 된다. 즉, 해당 hyperplane은 기존 hyperplane을 $\boldsymbol{u}$만큼 평행이동한 hyperplane이다. 또한 $\boldsymbol{w}^T(\boldsymbol{x}_0' - \boldsymbol{u})$에서 $\boldsymbol{u}$는 $\boldsymbol{w}$에 orthogonal한 원소가 사라지므로, $\boldsymbol{w}$의 방향으로 평행이동이 적용된다.

1차원 공간에서 hyperplane은 0차원인 점, 2차원 공간에서 hyperplane은 1차원인 직선, 3차원 공간에서 hy-

perplane은 2차원인 일반적인 평면이다.

기계 학습 및 데이터 분석에서의 모델, 특히 뒤에서 살펴볼 SVM이나 퍼셉트론과 같은 선형 분류기에서 hyperplane은 매우 중요한 개념이다. 입력 데이터가 고차원의 특징 공간(Feature space)에 매핑될 때, hyperplane은 데이터를 서로 다른 클래스로 양분하는 decision boundary로 기능한다. 즉, 새로운 데이터 포인트 $\boldsymbol{x}$ 가 입력되었을 때 $\boldsymbol{w}^T\boldsymbol{x}+b$ 의 부호(양수 또는 음수)에 따라 해당 데이터의 클래스를 분류할 수 있다.

2. Legendre-Fenchel Transform and Convex Conjugate
어떤 집합 S가 존재할 때, 해당 S의 경계에 있는 특정 점 x_0를 지나는 hyperplane을 생각하자. 이 hyperplane은 전체 space를 이등분하며, S가 두 반공간 중 한쪽 반공간에만 완전히 포함된다면, 해당 hyperplane을 점 x_0에서의 **Supporting Hyperplane**이라 한다. 이때 만약 S가 convex set이라면 supporting hyperplane이 적어도 하나 이상 존재하며, supporting hyperplane들이 생성하는 반공간들의 부분집합을 구하면 S가 된다. 즉, convex set은 hyperplane의 집합으로 나타낼 수 있다. 앞서 언급한 epigraph도 경계의 점을 지나는 hyperplane들을 생각해 이해할 수 있다. 또한 이 hyperplane들은 convex set에 대응되는 convex function과 접하므로, convex function은 이들의 gradient에 대한 function으로 나타낼 수 있다.

Legendre-Fenchel Transformation(르장드르 펜첼 변환)은 differentiable(미분 가능)한 convex function $f(x)$를 그 tangent(접선)의 gradient $\nabla_x f(x)$와 연관된 function으로 변환하는 transformation이다. 참고로, 이는 x가 아니라 function f에 대한 transformation이다. 이 transformation의 결과를 **Convex Conjugate Function(볼록 켤레 함수)**이라고 한다. 즉, function을 점들의 집합이 아니라 gradient의 집합으로 보는 duality가 존재한다.

function $f:\mathbb{R}^D\to\mathbb{R}$의 conjugate function f^*은 다음과 같이 정의된다. 이때 sup은 supremum으로, 상한(최댓값)을 뜻한다. 즉, s를 포함하는 vector에 대한 hyperplane에 대해, $f(x)$와의 차이가 최대인 지점의 거리를 출력하는 function이다. 참고로 이때 $f(x)$가 differentiable이거나 convex일 필요는 없다.

$$f^*(s)=\sup_{x\in\mathbb{R}^n}(\langle s,x\rangle-f(x))$$

직관적으로 생각해 보면 conjugate function의 function value는 각 s에 의해 정의되는 hyperplane이 $f(x)$의 아래쪽에 접하는 지점까지 평행이동시키는 거리이고, 이를 통해 $f(x)$가 어떻게 생겼는지를 알 수 있다.

convex function의 경우, Legendre transform을 2번 적용하면 원래의 function이 도출된다. 또한 convex differentiable function은 hyperplane과 접하는 지점이 가장 아래쪽에 항상 존재하며, 접하는 지점을 x_0라고 하면 다음과 같이 작성할 수 있다.

$$f^*(s)=\langle s,x_0\rangle-f(x_0)$$

다음은 convex conjugate를 계산하는 예시들이다.

Example 7.7 (Convex Conjugates)

Convex conjugates의 응용을 설명하기 위해서, 다음의 quadratic function 을 살펴봅시다.

$$f(\boldsymbol{y})=\frac{\lambda}{2}\boldsymbol{y}^\top\boldsymbol{K}^{-1}\boldsymbol{y} \tag{7.59}$$

$\boldsymbol{K}\in\mathbb{R}^{n\times n}$ 은 positive definite matrix 입니다.

여기서 primal variable은 $\boldsymbol{y}\in\mathbb{R}^n$ 으로 표기하고, dual variable은 $\boldsymbol{\alpha}\in\mathbb{R}^n$ 으로 표기합니다.

Definition 7.4를 적용하면 다음의 함수를 얻을 수 있습니다.

$$f^*(\boldsymbol{\alpha})=\sup_{\boldsymbol{y}\in\mathbb{R}^n}\langle\boldsymbol{y},\boldsymbol{\alpha}\rangle-\frac{\lambda}{2}\boldsymbol{y}^\top\boldsymbol{K}^{-1}\boldsymbol{y} \tag{7.60}$$

함수가 미분 가능하기 때문에, $\boldsymbol{y}$ 에 대한 도함수를 구하고 0과 같다라고 설정하면 maximum을 찾을 수 있습니다.

$$\frac{\partial[\langle\boldsymbol{y},\boldsymbol{\alpha}-\frac{\lambda}{2}\boldsymbol{y}^\top\boldsymbol{K}^{-1}\boldsymbol{y}]}{\partial\boldsymbol{y}}=(\boldsymbol{\alpha}-\lambda\boldsymbol{K}^{-1}\boldsymbol{y})^\top \tag{7.61}$$

Gradient가 0일 때, $\boldsymbol{y}=\frac{1}{\lambda}\boldsymbol{K}\boldsymbol{\alpha}$ 입니다. 이를 (7.60)에 대입하면, 다음의 식을 얻을 수 있습니다.

$$f^*(\boldsymbol{\alpha})=\frac{1}{\lambda}\boldsymbol{\alpha}^\top\boldsymbol{K}\boldsymbol{\alpha}-\frac{\lambda}{2}\left(\frac{1}{\lambda}\boldsymbol{K}\boldsymbol{\alpha}\right)^\top\boldsymbol{K}^{-1}\left(\frac{1}{\lambda}\boldsymbol{K}\boldsymbol{\alpha}\right)=\frac{1}{2\lambda}\boldsymbol{\alpha}^\top\boldsymbol{K}\boldsymbol{\alpha} \tag{7.62}$$

이렇게 각 sample에 대한 loss 값의 합은 개별 sample의 loss 값의 conjugate function을 생각할 수 있다.

Example 7.8

머신러닝에서는 보통 함수들의 합을 사용합니다. 예를 들어, training set의 objective function으로 training set의 각 example들의 losses 합을 사용합니다. 이번에는 $l : \mathbb{R} \to \mathbb{R}$ 인 losses $l(t)$ 의 합의 convex conjugate를 유도해보도록 하겠습니다. 이 예제는 벡터의 경우에 convex conjugate를 어떻게 적용하는지도 보여줍니다.

여기서 $\mathcal{L}(\boldsymbol{t}) = \sum_{i=1}^{n} l_i(t_i)$ 입니다.

그러면,

$$\mathcal{L}^*(\boldsymbol{z}) = \sup_{\boldsymbol{t}\in\mathbb{R}^n} \langle \boldsymbol{z}, \boldsymbol{t} \rangle - \sum_{i=1}^{n} l_i(t_i) \tag{7.63a}$$

$$= \sup_{\boldsymbol{t}\in\mathbb{R}^n} \sum_{i=1}^{n} z_i t_i - l_i(t_i) \quad \text{definition of dot product} \tag{7.63b}$$

$$= \sum_{i=1}^{n} \sup_{\boldsymbol{t}\in\mathbb{R}^n} z_i t_i - l_i(t_i) \tag{7.63c}$$

$$= \sum_{i=1}^{n} l_i^*(z_i) \qquad \text{definition of conjugate} \tag{7.63d}$$

입니다.

다음은 convex optimizaton problem을 convex conjugate를 활용해 접근하는 예시이다.

Example 7.9

$f(\boldsymbol{y})$ 와 $g(\boldsymbol{x})$ 가 convec functions 이고, 행렬 $\boldsymbol{A}$ 가 $\boldsymbol{Ax} = \boldsymbol{y}$ 를 만족하는 적절한 차원의 real matrix 라고 가정합시다. 그러면, 다음의 식이 성립합니다.

$$\min_{\boldsymbol{x}} f(\boldsymbol{Ax}) + g(\boldsymbol{x}) = \min_{\boldsymbol{Ax}=\boldsymbol{y}} f(\boldsymbol{y}) + g(\boldsymbol{x}) \tag{7.64}$$

제약 조건 $\boldsymbol{Ax} = \boldsymbol{y}$ 에 대한 라그랑주 승수 $\boldsymbol{u}$ 를 도입하면,

$$\min_{\boldsymbol{Ax}=\boldsymbol{y}} f(\boldsymbol{y}) + g(\boldsymbol{x}) = \min_{\boldsymbol{x},\boldsymbol{y}} \max_{\boldsymbol{u}} f(\boldsymbol{y}) + g(\boldsymbol{x}) + (\boldsymbol{Ax} - \boldsymbol{y})^\top \boldsymbol{u} \tag{7.65a}$$

$$= \max_{\boldsymbol{u}} \min_{\boldsymbol{x},\boldsymbol{y}} f(\boldsymbol{y}) + g(\boldsymbol{x}) + (\boldsymbol{Ax} - \boldsymbol{y})^\top \boldsymbol{u} \tag{7.65b}$$

이고, 여기서 $f(\boldsymbol{y})$ 와 $g(\boldsymbol{x})$ 가 convex function이기 때문에 마지막 줄의 max와 min의 위치를 바꾸었습니다.

Dot product term을 분리하고, $\boldsymbol{x}$ 와 $\boldsymbol{y}$ 에 대해 정리하면,

$$\max_{\boldsymbol{u}} \min_{\boldsymbol{x},\boldsymbol{y}} f(\boldsymbol{y}) + g(\boldsymbol{x}) + (\boldsymbol{Ax} - \boldsymbol{y})^\top \boldsymbol{u} \tag{7.66a}$$

$$= \max_{\boldsymbol{u}} \left[\min_{\boldsymbol{y}} -\boldsymbol{y}^\top \boldsymbol{u} + f(\boldsymbol{y}) \right] + \left[\min_{\boldsymbol{x}} (\boldsymbol{Ax})^\top \boldsymbol{u} + g(\boldsymbol{x}) \right] \tag{7.66b}$$

$$= \max_{\boldsymbol{u}} \left[\min_{\boldsymbol{y}} -\boldsymbol{y}^\top \boldsymbol{u} + f(\boldsymbol{y}) \right] + \left[\min_{\boldsymbol{x}} \boldsymbol{x}^\top \boldsymbol{A}^\top \boldsymbol{u} + g(\boldsymbol{x}) \right] \tag{7.66c}$$

Definition 7.4의 convex conjugate와 dot product가 symmetric 하다는 사실을 통해 다음의 식이 도출됩니다.

$$\max_{\boldsymbol{u}} \left[\min_{\boldsymbol{y}} -\boldsymbol{y}^\top \boldsymbol{u} + f(\boldsymbol{y}) \right] + \left[\min_{\boldsymbol{x}} \boldsymbol{x}^\top \boldsymbol{A}^\top \boldsymbol{u} + g(\boldsymbol{x}) \right] \tag{7.67a}$$

$$= \max_{\boldsymbol{u}} -f^*(\boldsymbol{u}) - g^*(-\boldsymbol{A}^\top \boldsymbol{u}) \tag{7.67b}$$

따라서, 다음의 식이 성립합니다.

$$\min_{\boldsymbol{x}} f(\boldsymbol{Ax}) + g(\boldsymbol{x}) = \max_{\boldsymbol{u}} -f^*(\boldsymbol{u}) - g^*(-\boldsymbol{A}^\top \boldsymbol{u}) \tag{7.68}$$

14. When Models Meet Data

여기에서부터는 지금까지 다룬 내용들을 ML에서 어떻게 활용하고 있는지를 다룬다. 우선 그 전에, 본 장에서는 우선 모델이란 무엇인지, 어떤 모델이 있는지, 좋은 모델은 무엇인지, 어떻게 모델을 선택하는지 등을 알아보자.

특히 세 가지 학습 유형인 ERM, MLE/MAP, bayesian inference를 중점적으로 알아본다.

14.1. Data, Models, and Learning

14.1.1. Data

여기에서는 데이터가 컴퓨터로 읽을 수 있고, 적절히 숫자 형태로 표현될 수 있다고 가정한다.

데이터셋에서 데이터의 개수는 N으로 표기하고, 데이터셋 내의 각 데이터는 $n = 1, \cdots, N$으로 인덱싱한다. 다음 표와 같이 데이터는 vector들의 배열로 주어진다고 하고, 이때의 각 row x_n는 개별 데이터를 나타내며 example 또는 data point라고 지칭한다. column은 관심있는 특정 feature를 나타내며, feature는 $1, \cdots, D$로 인덱싱한다. 이에 따라 x_n는 D-dimensional vector이다. 즉, example은 matrix $X \in \mathbb{R}^{N \times D}$로 나타낸다. 물론 ML 알고리즘에 따라 example을 column vector로 나타내기도 한다.

Gender ID	Degree	Latitude (in degrees)	Longitude (in degrees)	Age	Annual Salary (in thousands)
-1	2	51.5073	0.1290	36	89.563
-1	3	51.5074	0.1275	47	123.543
+1	1	51.5071	0.1278	26	23.989
-1	1	51.5075	0.1281	68	138.769
+1	2	51.5074	0.1278	33	113.888

Table 8.2 Example data from a fictitious human resource database (see Table 8.1), converted to a numerical format.

supervised learning problem에서는 example x_n이 있을 때 각 example에 대한 label y_n이 존재한다. label은 target, response variable, annotation이라고도 한다. 이 경우 데이터셋은 $\{(x_1, y_1), \cdots, (x_N, y_N)\}$과 같이 example-label 쌍의 집합으로 정의된다.

ML에서는 example 집합이 i.i.d.임을 가정한다. 즉, example 간의 statistically indepdent하고, 동일한 distribution을 가진다고 한다.

"이제 책의 첫 번째 파트에서 살펴본 수학적 개념들을 사용하여 위에서 살펴본 것과 같은 머신러닝 문제를 공식화합니다. 데이터를 벡터로 표현하면 2장에서 살펴본 선형대수(linear algebra) 개념을 사용할 수 있습니다. 많은 머신러닝 알고리즘에서는 추가적으로 두 벡터를 비교할 수 있어야 합니다. 뒤에서 살펴볼텐데, 두 example 간의 similarity나 distance를 계산하여 유사한 features를 가진 examples가 유사한 label을 갖는다는 것을 공식화할 수 있습니다."

"데이터를 벡터로 표현하기 때문에, 데이터를 조작하여 잠재적으로 더 좋은 표현(better representations)을 찾을 수 있습니다. Good representations를 찾는 두 가지 방법을 논의하는데, 한 가지 방법은 original feature vector의 lower-demensional approximations를 찾는 것이고, 다른 하나는 original feature vector의 non-linear higher-dimensional combinations를 사용하는 것입니다."

14.1.2. Models

model/predictor가 어떤 것인지는 다음과 같은 두 가지 관점으로 이해할 수 있다.

1. Models as Functions
predictor를 input sample에 대해 output을 생성하는 linear function으로 정의한다.

$$f(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{x} + \theta_0$$

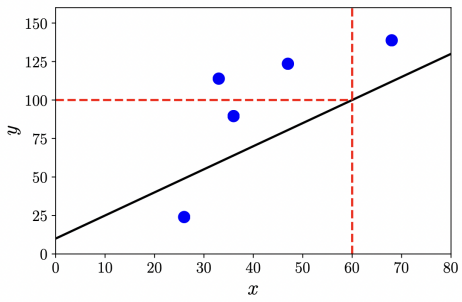


Figure 8.2 Example function (black solid diagonal line) and its prediction at $x = 60$, i.e., $f(60) = 100$.

2. Models as Probability Distributions
predictor를 single function으로 간주하는 대신, function이 가지는 distribution을 설명하는 probabilistic model로 정의할 수 있다. 이런 probabilistic model은 포함하는 모든 random variable의 joint distribution으로 정의된다.

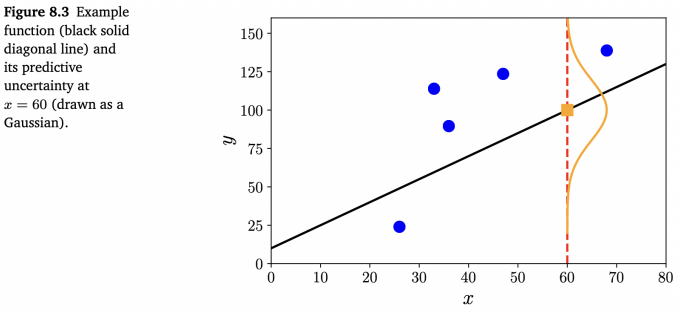


Figure 8.3 Example function (black solid diagonal line) and its predictive uncertainty at $x = 60$ (drawn as a Gaussian).

14.1.3. Learning

learning 또는 training의 목표는 unseen data에 대해서도 predictor가 잘 예측하도록 좋은 parameters를 찾는 것이다.

ML 알고리즘은 3가지 구분되는 단계로 이해할 수 있다.

1. Prediction/Inference: unseen data에 대해 학습된 predictor를 사용하는 단계.
2. Training/Parameter Estimation: training data를 기반으로 predictor를 조정하는 단계. measure of qualtiy를 기반으로 조정하는 방식과(loss/objective function 사용), bayesian inference를 활용해 조정하는 방식이 있다.
3. Hyperparameter Tuning/Model Selection

14.2. Empirical Risk Minimization

predictor가 function인 경우에 대해 알아보자.

14.2.1. Empirical Risk Minimization

1. Empirical Risk Minimization
Empirical Risk Minimization(ERM)은 ML에서 모델을 학습할 때, 관측 가능한 학습 데이터셋에 대한 오차(Risk)를 최소화하여 최적의 모델을 찾는 기본적인 방법론이다. 이때의 design choice는 다음과 같은 것들이 있다.

- What is the set of functions we allow the predictor to take?
- How do we measure how well the predictor performs on the training data?
- How do we construct predictors from only training data that performs well on unseen test data?
- What is the procedure for searching over the space of models?

2. ERM에서의 Function
N개의 example $\boldsymbol{x_n} \in \mathbb{R}^D$와, 각 example에 대응되는 $y_n \in \mathbb{R}$이 주어진 $\{(\boldsymbol{x_1}, y_1), \cdots, (\boldsymbol{x_N}, y_N)\}$ 쌍에 대한 supervised learning을 가정하자. parameter가 $\boldsymbol{\theta}$로 표기되는 function(predictor) $f : \mathbb{R}^D \to \mathbb{R}$에 대해 다음과 같이 좋은 parameter $\boldsymbol{\theta}^*$을 찾는 것이 ERM의 목적이다.

$$f(\boldsymbol{x_n}, \boldsymbol{\theta}^*) \approx y_n \quad \text{for all} \quad n = 1, \cdots, N$$

이때의 predictor의 output(예측값)은 $\hat{y} = f(\boldsymbol{x_n}, \boldsymbol{\theta}^*)$로 표기한다.

또한 다음과 같이 linear function으로 나타내거나, $\boldsymbol{x_n}$에 1을 추가해서 $\boldsymbol{x_n} = [1, x_1, \cdots, x_D]^T$, $\boldsymbol{\theta} = [\theta_0, \cdots, \theta_D]$인 경우에 대해 affine model로도 표기할 수 있다. 물론 알고리즘에 따라 non-linear function으로도 정의할 수 있다.

$$f(\boldsymbol{x_n}, \boldsymbol{\theta}^*) = \boldsymbol{\theta}^T \boldsymbol{x_n}$$
$$f(\boldsymbol{x_n}, \boldsymbol{\theta}^*) = \theta_0 + \sum_{d=1}^{D} \theta_d x_d$$

3. Loss Function
예측이 데이터에 잘 맞는다는 것을 이야기하기 위해서는 ground truth(label) y_n과, prediction $\hat{y}_n$ 사이의 non-negative 값인 **Loss**를 출력하는 **Loss Function** $\ell(y_n, \hat{y}_n)$을 정의해야 한다. 좋은 parameter $\boldsymbol{\theta}^*$를 찾는다는 목표는 N개의 training example 집합에 대한 평균 loss를 최소화하는 것이다. 이때 example 집합은 i.i.d.이므로, 이런 empirical mean이 모집단 평균(population mean)의 good estimate라고 생각할 수 있다.

즉, example matrix $X = [x_1, \cdots, x_N]^T \in \mathbb{R}^{N \times D}$와 label vector $\boldsymbol{y} = [y_1, \cdots, y_N]^T \in \mathbb{R}^N$에 대해 평균 loss는 다음과 같이 작성할 수 있다. 이때 $\hat{y}_n = f(\boldsymbol{x}_n, \boldsymbol{\theta})$이다. 이를 **Empirical Risk**라고 하며, ERM은 이를 최소화하는 problem이다.

$$\boldsymbol{R}_{emp}(f, \boldsymbol{X}, \boldsymbol{y}) = \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, \hat{y}_n)$$

또한 무한한 개수의 데이터를 가정한 경우에 대해 True Risk를 다음과 같이 작성할 수 있다. 즉, 모든 데이터에 대한 loss의 평균을 최소로 하는 function f를 찾는 것이 목표이다.

$$\boldsymbol{R}_{\text{true}}(f) = \mathbb{E}_{\boldsymbol{x},y}[\ell(y, f(\boldsymbol{x}))]$$

예를 들어, least square loss의 경우 다음과 같이 작성할 수 있다.

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^D} \frac{1}{N} \sum_{n=1}^{N} (y_n - f(\boldsymbol{x}_n, \boldsymbol{\theta}))^2 = \min_{\boldsymbol{\theta} \in \mathbb{R}^D} \frac{1}{N} \sum_{i=1}^{N} (y_n - \boldsymbol{\theta}^\top \boldsymbol{x})^2 = \min_{\boldsymbol{\theta} \in \mathbb{R}^D} \frac{1}{N} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}\|^2$$

14.2.2. Generalization Performance

1. Generalization
predictor는 unseen data에 대해 좋은 성능을 보여야 한다. 즉, generalization 능력을 가지고 있어야 한다.

학습 시에 유한한 데이터 집합이 있을 때, 이는 training set과 test set으로 나눈다. training set은 모델을 학습할 때 사용되고, test set은 unseen data로서 generalization performance를 평가할 때 사용한다. training set 중 일부는 그 subset인 validation set으로 분리해 학습 과정에서의 performance를 측정하고 hyperparameter 튜닝을 하는 데에 사용한다. 반면 test set은 모든 학습 및 hyperparameter 튜닝이 완료된 이후 모델의 성능을 확인할 때 사용된다.

ERM은 predictor가 training set에만 너무 잘 맞고, test set에는 잘 맞지 않는 Overfitting(과적합)에 빠질 수 있다. 이는 주로 데이터가 적고 hypothesis class가 복잡한 경우에 주로 발생한다. empirical risk가 true risk를 과소평가하고 있는 것으로도 이해할 수 있다.

2. Regularization
Regularization(정규화)은 training optimizer가 지나치가 flexible한 predictor를 반환하기 어렵도록 하는 penalty term을 추가하는 기법이다. 이때 추가하는 항을 Regularizer라고 하고, λ를 Regularization Parameter라고 한다.

대표적인 regularation으로는 L1 regularization과 L2 regularization이 있다. overfitting이 일어나면 일반적으로 parameter 값의 크기가 커지는 경향이 있는데, 이 두 방식 모두 optimizaiton 시에 parameter가 0에 가까워지도록 한다. L1/L2 regularization은 아래의 수식과 같이 각각 loss function에 $\lambda|\theta|$와 $\lambda||\theta||^2$를 패널티로 더한다. λ는 패널티의 크기를 지정하는 양의 실수로, hyperparameter이다.

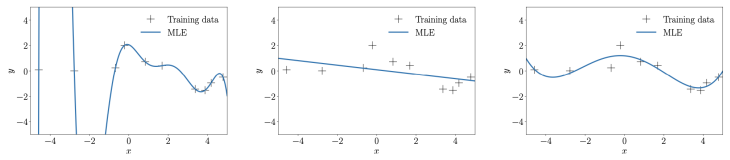
$$\ell' = \ell + \lambda|\theta|$$
$$\ell' = \ell + \lambda||\theta||^2$$

이때 L1은 λ가 충분히 작아지면 θ는 0으로 최적화되고, L2는 λ가 충분히 작아져도 0으로 최적화되지는 않는다. 각 loss를 미분한 것을 생각해 보면 L1에서는 graient가 $\lambda(sign(\theta))$이고, L2에서는 gradient가 $2\lambda\theta$이므로 쉽게 이해할 수 있다. 또한 L2는 θ의 크기에 따라 패널티 값이 정해지기 때문에 θ 값이 부드럽게 0에 근접하게 된다. 어떨 때는 L1, 어떨 때는 L2가 더 나아서, empirical하게 판단해야 한다.

14.2.3. Model Fitting

Fitting은 모델의 parameter를 최적화해 어떤 loss function을 최소화하는 것을 의미한다. 최적화 이후 가능한 best parameter를 찾았을 때, 3가지 case 중 하나일 수 있다.

1. **Overfitting**: 모델이 해당 데이터셋을 모델링하기에 너무 과한 상황. predictor가 training set에만 너무 잘 맞고, test set에는 잘 맞지 않는 경우를 말한다. 주로 parameter 수가 너무 많은 경우에 발생한다.
2. **Underfitting**: 모델이 해당 데이터셋을 모델링하기에 충분하지 않은 상황. 주로 parameter 수가 너무 적은 경우에 발생한다.
3. Fitting well: 모델이 해당 데이터셋을 모델링하기에 적절한 상황.



(a) Overfitting (b) Underfitting. (c) Fitting well.

Figure 8.8 Fitting (by maximum likelihood) of different model classes to a regression dataset.

14.3. Parameter Estimation

predictor가 probability distribution인 경우에 대해 알아보자. **Parameter Estimation**은 데이터를 바탕으로 해당 데이터를 생성할 probability distribution의 parameter를 수학적으로 추론하는 과정을 의미한다. 특히 아래에서 알아볼 MLE와 MAP는 모델을 가장 잘 설명하는 단 하나의 최적의 θ 값을 찾는 point estimate를 수행한다. 즉, 그 결과물은 probability distribution이 아니라 상수 값인 parameter다. 여기에서 살펴볼 likelihood는 loss function과 유사하고, prior는 regularization과 유사하다.

14.3.1. Maximum Likelihood Estimation

Maximum Likelihood Estimation(MLE)는 데이터에 대한 likelihood를 최대로 하는 parameter를 찾는 기법이다. 즉, 데이터에 잘 맞는 모델을 찾기 위해 parameter에 대한 likelihood function을 정의해 활용하는 것이다.

N개의 example $\boldsymbol{x_n} \in \mathbb{R}^D$와, 각 example에 대응되는 $y_n \in \mathbb{R}$이 주어진 $\{(\boldsymbol{x_1}, y_1), \cdots, (\boldsymbol{x_N}, y_N)\}$ 쌍에 대한 supervised learning을 가정하자. 이때 example의 집합과 label의 집합이 i.i.d.이므로, $X = \{\boldsymbol{x_1}, \cdots, \boldsymbol{x_N}\}, Y = \{y_1, \cdots, y_N\}$에 대한 likelihood는 다음과 같이 개별 example에 대한 likelihood의 곱으로 분해할 수 있다.

$$p(Y|X, \boldsymbol{\theta}) = \prod_{n=1}^{N} p(y_n|\boldsymbol{x_n}, \boldsymbol{\theta})$$

random variable $\boldsymbol{x}$로 표현되는 데이터와, parameter $\boldsymbol{\theta}$를 가지는 continous probability distribution $p(\boldsymbol{x}|\boldsymbol{\theta})$에 대해 Negative Log-likelihood $\mathcal{L}_x(\theta)$는 다음과 같이 정의된다. random variable이 명확할 때는 $\mathcal{L}(\theta)$로도 작성한다. $p(\boldsymbol{x}|\boldsymbol{\theta})$는 관측된 데이터 $\boldsymbol{x}$가 도출되기에 parameter $\boldsymbol{\theta}$가 얼마나 적합한지를 나타내고, negative log likelihood 값이 작을수록 적합한 것을 알 수 있다.

$$\mathcal{L}_x(\theta) = -\log p(\boldsymbol{x}|\boldsymbol{\theta})$$

negative log likelihood를 사용하면 각 distribution에 대한 곱을 덧셈으로 풀 수 있고, MLE는 이 negative log likelihood가 최소로 하는 parameter를 찾는 것을 그 목적으로 한다.

$$\mathcal{L}(\boldsymbol{\theta}) = -\log p(Y|X,\boldsymbol{\theta}) = -\sum_{n=1}^{N} \log p(y_n|\boldsymbol{x}_n,\boldsymbol{\theta})$$
$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$$

MLE는 무한히 많은 데이터가 존재할 때 예측값이 true value로 수렴하고, 예측값과 true value 간의 오차는 mean이 0인 gaussian distribution으로 수렴한다. 또한 N개의 데이터를 사용했을 때 variance는 $\frac{1}{N}$으로, 더 많은 데이터를 사용할수록 variance가 줄어든다. 반면 적은 데이터를 사용하는 경우 overfitting이 발생한다.

MLE에 대해서도 closed-form analytic solution이 없을 수 있는데, 이 경우 앞서 다룬 optimization 알고리즘을 활용한다.

다음은 gaussian distribution의 negative log liklihood를 구하는 예시이다.

Example 8.5

Gaussian likelihood (8.15)의 예제를 이어서 살펴보겠습니다. Negative log-likelihood는 아래와 같이 다시 작성됩니다.

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{n=1}^{N} \log p(y_n|\boldsymbol{x}_n,\boldsymbol{\theta}) = -\sum_{n=1}^{N} \log \mathcal{N}(y_n|\boldsymbol{x}_n^\top\boldsymbol{\theta},\sigma^2) \tag{8.18a}$$
$$= -\sum_{n=1}^{N} \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_n-\boldsymbol{x}_n^\top\boldsymbol{\theta})^2}{2\sigma^2}\right) \tag{8.18b}$$
$$= -\sum_{n=1}^{N} \log \exp\left(-\frac{(y_n-\boldsymbol{x}_n^\top\boldsymbol{\theta})^2}{2\sigma^2}\right) - \sum_{n=1}^{N} \log \frac{1}{\sqrt{2\pi\sigma^2}} \tag{8.18c}$$
$$= \frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n-\boldsymbol{x}_n^\top\boldsymbol{\theta})^2 - \sum_{n=1}^{N} \log \frac{1}{\sqrt{2\pi\sigma^2}} \tag{8.18d}$$

σ 가 주어지면, (8.18c)의 두 번째 항은 상수이므로, $\mathcal{L}(\boldsymbol{\theta})$ 를 최소화하는 것은 least-squares problem(첫 번째 항)을 푸는 것과 같습니다 (식 (8.8)과 비교).

14.3.2. Maximum A Posteriori Estimation

Maximum A Posteriori(MAP) Estimation은 관측된 데이터와 parameter에 대한 prior(사전 지식)를 모두 고려하여, posterior를 최대로 하는 parameter를 찾는 통계적 기반의 추론 기법이다. 즉, negative log-likelihood의 최솟값을 추정하는 대신, negative log-posterior의 최솟값을 추정하는 problem을 풀게 된다. 이를 위해 bayes' theorem을 적용하면 다음과 같은 수식을 얻을 수 있다. 이때 parameter와 데이터셋은 independent 하므로 $p(\boldsymbol{\theta}|\mathcal{X}) = p(\boldsymbol{\theta})$이다.

$$p(\boldsymbol{\theta}|X,Y) = \frac{p(Y|X,\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(Y|X)}$$

parameter의 distribution에 prior(사전 지식)가 존재한다면, 다음과 같이 bayes' theorem을 활용해 반영할 수 있다. 이때 evidence는 parameter와 관련이 없으므로 제외한다. 즉, 이 경우 posterior가 큰 모델일수록 prior/likelihood가 크므로 더 좋은 모델이라고 할 수 있다. posterior의 정의에 의해서도, $p(\boldsymbol{\theta}|\boldsymbol{x})$는 데이터를 고려했을 때 해당 parameter가 선택될 proabability를 나타낸다.

$$p(\boldsymbol{\theta}|Y,X) \propto p(Y|X,\boldsymbol{\theta})p(\boldsymbol{\theta})$$

이 경우 likelihood와 prior의 곱을 최대화하는 parameter $\hat{\boldsymbol{\theta}}_{MAP}$를 찾는 optimization problem으로 정의된다.

$$\hat{\boldsymbol{\theta}}_{MAP} = \arg\max_{\boldsymbol{\theta}} p(Y|X,\boldsymbol{\theta})p(\boldsymbol{\theta})$$

계산의 편의를 위해 negative log를 적용하면, 다음과 같이 정리된다. 이때 추가된 $-\log p(\boldsymbol{\theta})$는 regularization의 penalty term의 역할을 수행한다.

$$\hat{\boldsymbol{\theta}}_{MAP} = \arg\min_{\boldsymbol{\theta}} \left(-\log p(Y|X,\boldsymbol{\theta}) - \log p(\boldsymbol{\theta})\right)$$

14.4. Probabilistic Modeling and Inference

MLE와 MAP에서는 최적화를 통해 parameter의 단일 추정치를 찾았는데, 여기에서는 parameter에 대한 posterior distribution 전체를 구하는 방법을 알아본다. 이렇게 distribution 자체를 구하고 활용하는 것을 통해 모델이 가지는 불확실성을 정량화할 수 있고, overfitting을 피할 수 있다.

"단일 추정치 θ^* 는 사후 분포에서 확률이 가장 높은 하나의 점(Mode)을 의미합니다. 이는 해당 파라미터가 얼마나 신뢰할 수 있는지에 대한 정보를 제공하지 않습니다. 반면, 전체 사후 분포 $p(\theta \mid X)$ 를 계산하면 파라미터 공간의 분산과 형태를 파악할 수 있습니다. 데이터가 적거나 노이즈가 많아 분포가 넓게 퍼져 있다면 모델은 자신의 추론에 확신이 없음을 나타내며, 이는 신뢰도 기반의 제어 시스템에서 필수적인 지표입니다."

"전체 사후 분포 $p(\theta \mid X)$ 를 가중치로 삼아 모든 모델의 예측을 적분하는 과정은 사실상 무한한 수의 모델을 결합하는 앙상블(Ensemble) 기법과 동일한 효과를 지닙니다. 이를 통해 새로운 테스트 데이터에 대한 예측의 견고성을 향상시킬 수 있습니다."

14.4.1. Bayesian Inference

Bayesian Inference는 parameter가 가지는 posterior distribution 전체를 활용해 예측하는 방식이다. MLE와 MAP과 같이 point estimate를 해서 상수 parameter 값을 구하는 경우 true value와의 오차(정보의 손실)가 존재한다. 이에 따라 full posterior distribution을 찾아 활용하는 것은 더 견고하게 예측하는 데에 도움이 된다. 즉, 이는 모델의 parameter를 고정된 상수가 아닌, random variable로 취급하는 방식이다.

"빈도주의(Frequentist) 통계학에서는 모델의 파라미터 $\boldsymbol{\theta}$ 를 '알려지지 않았지만 고정되어 있는 단일한 상수'로 간주합니다. 반면, 베이즈 통계학에서는 파라미터 자체를 불확실성을 내포한 확률 변수 로 취급합니다. 아직 데이터를 관측하기 전이므로 파라미터가 어떤 값을 가질지에 대한 불확실성을 확률 분포인 사전 확률 로 나타내며, 모델에 적용될 구체적인 파라미터 $\boldsymbol{\theta}_k$ 는 이 확률 분포에서 도출되는 하나의 결과값(샘플)이라고 가정합니다."

데이터셋 X 와 parameter $\boldsymbol{\theta}$ 를 생각했을 때, bayes' theorem에 의해 다음과 같이 posterior distribution을 얻을 수 있다.

$$P(\boldsymbol{\theta}|X) = \frac{P(X|\boldsymbol{\theta})P(\boldsymbol{\theta})}{P(X)}$$

데이터셋 X 로 학습된 parameter $\boldsymbol{\theta}$ 의 경우, 어떤 데이터 $\boldsymbol{x}$ 에 대해 해당 distribution을 활용한 inference는 다음과 같은 수식으로 나타낼 수 있다. 훈련 데이터셋 X가 주어졌을 때 새로운 데이터 $\boldsymbol{x}$에 대해 $p(\boldsymbol{x} \mid X)$를 계산하는 것이 목적이므로 이렇게 정의된다.

이때 conditional probability의 정의에 의해 $p(\boldsymbol{x}, \boldsymbol{\theta} \mid X) = p(\boldsymbol{x} \mid \boldsymbol{\theta}, X)p(\boldsymbol{\theta} \mid X)$ 이고, $\boldsymbol{\theta}$ 가 주어졌을 때 $\boldsymbol{x}$

와 X 는 conditional independent하므로 $p(\boldsymbol{x} \mid \boldsymbol{\theta}, X) = p(\boldsymbol{x} \mid \boldsymbol{\theta})$ 가 성립한다. 즉, 다음 수식은 모든 그럴듯한 parameter 값들 각각의 예측값을 평균낸 것이다. 이 경우 풀어야 하는 problem은 optimization이 아니라 적분이 된다.

$$\begin{aligned} p(\boldsymbol{x} \mid X) &= \int p(\boldsymbol{x}, \boldsymbol{\theta} \mid X) d\boldsymbol{\theta} \\ &= \int p(\boldsymbol{x} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid X) d\boldsymbol{\theta} \end{aligned}$$

이는 posterior $p(\boldsymbol{\theta} \mid X)$에 대한 likelihood $p(\boldsymbol{x} \mid \boldsymbol{\theta})$의 기댓값을 연산하는 것과 동일하다. 다시 말해, 모든 가능한 parameter들에 대해서, likelihood를 posterior로 가중평균한 것으로 이해할 수 있다.

$$\mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid X)}[p(\boldsymbol{x} \mid \boldsymbol{\theta})]$$

"Maximum likelihood 또는 MAP 추정을 통한 파라미터 추정은 파라미터의 일관된 point estimates θ^*를 추출하며, 풀어야 하는 핵심 문제는 최적화입니다. 이와 대조적으로, 베이지안 추론은 사후 분포를 추출하며, 풀어야 하는 핵심 문제는 적분입니다. Point estimate를 사용한 예측은 간단하지만, 베이지안 프레임워크의 예측은 다른 적분 문제를 풀어야 합니다 (8.23 식 참조). 그러나 베이지안 추론은 파라미터 추정 문맥에서는 쉽게 수행되지 않는 사전 지식을 통합하고, 무가적인 정보들을 설명하고, 구조적 지식을 통합하는 원칙적인 방법을 제공합니다. 또한, 예측에 대한 매개변수 불확실성의 전파는 데이터의 효율적인 학습의 맥락에서 risk 평가 및 탐색에 대한 의사결정 시스템에서 유용할 수 있습니다."

14.4.2. Latent Variable Models

Latent Variable Model(잠재 변수 모델)은 데이터 생성 과정을 더 명시적으로 설명하는 latent variable을 추가로 도입한 모델이다. latent variable을 사용해 parameter에서 데이터를 생성하는 프로세스를 정의할 수 있다. 데이터를 $\boldsymbol{x}$, parameter를 $\boldsymbol{\theta}$, latent variable을 $\boldsymbol{z}$로 정의하면 다음과 같은 conditional probability를 얻을 수 있다.

$$p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{\theta})$$

다음과 같이 latent variable을 고려하여 모델의 likelihood를 계산할 수 있다. 이를 활용해 latent variable model에 대해서도 MLE, MAP를 적용해 parameter 학습과 inference를 할 수 있고, bayesian inference도 가능하다.

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \int p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{\theta}) p(\boldsymbol{z}) d\boldsymbol{z}$$

이때 latent variable에 대한 marginalization이 필요하다는 문제가 있는데, conjugate prior를 활용하는 경우를 제외하면 이는 계산이 어렵거나 근사에 의존해야 한다.

다음 수식과 같이 생성된 $\boldsymbol{x}$를 사용해 역으로 latent variable에 대한 posterior도 구할 수 있다. latent variable은 직접 관측되지는 않지만 데이터 생성에 영향을 미치는 근본적인 요인이므로, 이에 대한 posterior를 구해 생성된 결과를 기반으로 그 원인의 확률 분포를 역으로 추론할 수 있다. 이를 통해 데이터의 특성, 저차원 표현(representation)을 추정하는 등의 작업을 진행할 수 있다.

$$p(\boldsymbol{z}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{p(\boldsymbol{x})}, \quad p(\boldsymbol{x}|\boldsymbol{z}) = \int p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta}, \quad p(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta}) = \frac{p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{\theta})p(\boldsymbol{z})}{p(\boldsymbol{x}|\boldsymbol{\theta})}$$

variational autoencoder 등에서 이런 latent variable을 사용한다. 또한 본 요약본에 정리하진 않았지만 PCA, gaussian mixture model 등도 latent variable model 관점에서 접근이 가능하다.

14.5. Directed Graphical Models

14.5.1. Directed Graphical Models

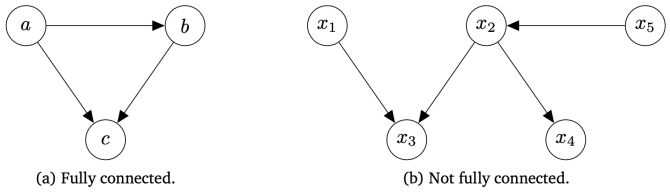
Directed Graphical Models 또는 Bayesian Networks는 여러 random variable로 구성된 joint distribution을 인수분해하여, 각 random variable 간의 conditional dependence를 directed acyclic graph로 표현하는 방법론이다. 특히 $p(x, y, z)$ 꼴의 joint distribution은 그 자체로 random variable 간의 관계를 보여주진 않는데, 이를 directed graphical model로 나타내 probability model을 구체화하고, random variable 간의 dependency를 시각적으로 확인할 수 있다.

directed graphical model에서 각 node(정점)는 random variable이고, edge(간선)는 random variable 간의 probabilistic relation(확률적 관계)인 conditional probability를 나타낸다. edge가 없는 경우 두 random variable 간에 conditional independence가 성립한다. 예를 들어, 다음의 왼쪽 그림에서 a와 b 사이의 edge는 $p(b|a)$를 나타낸다.

어떤 joint distribution에 대한 directed graphical model을 구하는 과정은, 1) random variable들에 대응되는 node를 그리고, 2) joint distribution을 인수분해한 뒤 directed link(arrow)를 그리는 것이다. 이에 따라 어떻게 인수분해하느냐에 따라 그 구조가 달라질 수 있다. 예를 들어, 아래의 왼쪽 그림은 $p(a, b, c) = p(c\#a, b)p(b\#a)p(a)$의 directed graphical model은 인 것으로 이해할 수 있다.

또한 이 반대 과정을 통해 directed graphical model로부터 distribution을 얻을 수 있다. 예를 들어, 아래의 오른쪽 그림의 distribution은 $p(x_1, x_2, x_3, x_4, x_5) = p(x_1)p(x_5)p(x_2 \mid x_5)p(x_3 \mid x_1, x_2)p(x_4 \mid x_2)$이다. 혹은 다음과 같은 수식으로도 나타낼 수 있는데, $Pa(X_i)$는 parent node 집합을 의미한다.

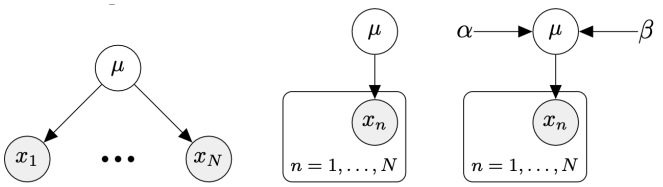
$$P(X_1, X_2, \ldots, X_N) = \prod_{i=1}^{N} P(X_i \mid \mathrm{Pa}(X_i))$$



(a) Fully connected. (b) Not fully connected.

Figure 8.9 Examples of directed graphical models.

또한 다음과 같이 plate notation을 사용해 반복을 나타낼 수 있고, deterministic parameter의 경우 원을 그리지 않는다.



(a) Version with x_n explicit. (b) Version with plate notation. (c) Hyperparameters α and β on the latent μ.

Figure 8.10 Graphical models for a repeated Bernoulli experiment.

14.5.2. d-Separation

non-intersection(교집합이 없는 경우)인 node 집합 A, B, C를 생각하자, **d-Separation**은 다음과 같은 A와 $B|C$의 independency를 판단하는 방법이다.

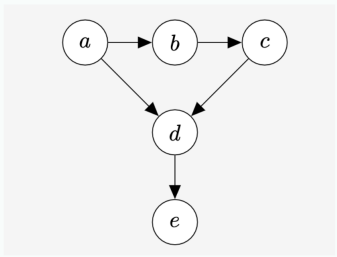
$$A \perp\!\!\!\perp B|C$$

방향을 무시하고 A에서 B로 가는 모든 경로를 directed graphical model에서 확인했을 때, 다음 중 하나에 해당되는 node가 관측된다면 경로가 차단된 것으로 판단한다. 모든 경로가 차단되었다면 A는 $B|C$로부터 d-separate되었다고 하고, 위의 수식인 independency가 성립한다.

- 경로의 화살표가 node에서 head to tail 또는 tail to tail 만족하고, 그 node가 C의 집합 내의 node
- node에서 화살표가 head to head를 만족하고, 그 node나 node의 자손이 C 집합에 없는 node

head-to-head는 두 개의 node가 하나의 공통된 child node를 가리키는 형태로, 이러한 child node가 관측되지 않으면 두 node는 independent하다. tail-to-tail은 두 개의 node가 하나의 공통된 parent node를 가지는 형태로, 이러한 parent node가 관측되지 않으면 두 node는 dependent하다. head-to-tail은 한 node에서 다른 node로, 그리고 그 다음 node로 연결된 형태이다.

Example 8.9 (Conditional Independece)



위 그림과 같은 그래프 모델을 봅시다. 이를 살펴보면 다음을 찾아낼 수 있습니다.

$$b \perp\!\!\!\perp d|a,c \tag{8.35}$$
$$a \perp\!\!\!\perp c|b \tag{8.36}$$
$$b \not\perp\!\!\!\perp d|c \tag{8.37}$$
$$a \not\perp\!\!\!\perp c|b,e \tag{8.38}$$

14.6. Model Selection

어떤 문제에 대해 모델을 선택하는 경우에는, 모델의 parameter 개수 등과 같은 모델 자체의 복잡성만 고려하는 것은 적절하지 않다. 앞서 다룬 것처럼 dataset의 크기와 모델의 복잡성이 비등한 경우에 test set에서도 좋은 성능을 보일 수 있다. 이에 따라 모델을 선택할 때에는 그 generalization 능력을 평가하는 것이 중요할 수 있다. 본 절에서는 generalization 능력의 평가와, 모델의 복잡성과 데이터 사이의 적합성을 고려한 모델 선택에 대해 다룬다. 이런 선택은 Occam's Razor로도 이해할 수 있다.

14.6.1. Cross-validation

1. Cross-validation
데이터셋의 크기는 고정되어 있고, 이를 training/validation/test set으로 분리해 사용한다. 모델을 학습시킬 때에는 training set을 최대한 크게 하는 것이 performance에 유리하지만, training set의 크기를 키우면 validation set의 크기가 줄어들어 결과적으로 예측이 noisy해질 수 있다. 이런 tradeoff를 완화하는 한 가지 방법은 **Cross-validation(교차검증)**이다. cross-validation을 통해, 특히 데이터셋 크기 작은 경우에 모델의 generalization 성능을 더 robust하게 평가할 수 있다.

K-fold Cross-validation은 다음 그림과 같이 데이터를 K개의 chunk로 분리하여, $K-1$개는 training set R로, 나머지 하나는 validation set V로 활용하는 기법이다. 이때 모든 R과 V의 조합을 활용한다.

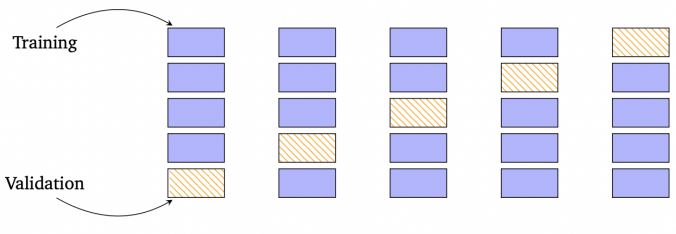


Figure 8.4 K-fold cross-validation. The dataset is divided into $K = 5$ chunks, $K - 1$ of which serve as the training set (blue) and one as the validation set (orange hatch).

k-fold cross-validation을 적용한 경우, training data $R^{(i)}$으로 학습시켜 얻은 predictor $f^{(i)}$가 도출된다. $f^{(i)}$가 $V^{(i)}$에서 보이는 성능 R이라고 하면(ex. root mean square error) 이를 모든 chunk에 대해 평균낸 값을 expected generalization error라 할 수 있다. 이때 k번의 순차적인 학습이 수행되는 것이 아니라, k개의 독립적인 학습 프로세스를 수행한 뒤 그 성능만 평균내는 것이다. 즉, 이는 더 robust하게 성능을 측정하는 방법이다.

$$\mathbb{E}_{\mathcal{V}}[R(f,V)] \approx \frac{1}{K}\sum_{k=1}^{K} R(f^{(k)},V^{(k)})$$

cross-validation은 computation이 많이 필요할 수 있지만, embarrassingly parallel problem이므로 computing resource만 충분하다면 병렬적으로 수행할 수 있다. Embarrassingly Parallel Problem은 전체 작업을 여러 개의 하위 작업으로 분할할 때, 하위 작업들 간에 data dependency이나 communication이 전혀 또는 거의 필요하지 않은 문제를 말한다.

2. Nested Cross-validation
Nested Cross-validation은 training set과 validation set에 대한 분리만을 다루는 k-fold cross-validation에서 더 나아가, test set에 대한 분리도 추가로 다루는 기법이다. test set에 대한 분리를 Outer Loop, validation set에 대한 분리를 Inner Loop라고 한다.

이는 다음 그림과 같이 test set을 분리한 뒤, validation set을 분리한다. test set으로는 generalization 능력을 평가하고, validation set으로는 모델 선택과 hyperparameter tuning을 수행한다. 정리하면 다음과 같은 과정을 통해 수행된다.

1. 전체 데이터를 K_{outer} 개의 Chunk로 분할하고, K_{outer} 번의 반복 루프(Outer Loop)를 시작합니다.
2. 매 반복마다 1개의 Chunk를 독립된 Test Set으로 떼어두고, 나머지 $K_{outer}-1$ 개의 Chunk를 결합하여 임시 학습 데이터로 구성합니다.
3. 임시 학습 데이터를 다시 K_{inner} 개의 Chunk로 분할하여 Inner Loop 를 시작합니다.
4. 지정된 각 하이퍼파라미터 조합에 대해 K_{inner} 번의 교차 검증을 수행하여 성능을 평균 냅니다.
5. 가장 낮은 오차를 보인 최적의 하이퍼파라미터 조합을 도출합니다.
6. 도출된 최적의 하이퍼파라미터를 적용하여, 단계 2에서 결합했던 $K_{outer}-1$ 개의 임시 학습 데이터 전체를 사용하여 모델을 1회 재학습시킵니다.
7. 학습이 완료된 모델을 단계 2에서 분리해 두었던 Test Set에 입력하여 성능 지표를 산출합니다.
8. 위 과정을 반복하여 모든 K_{outer} 개의 Chunk가 한 번씩 Test Set으로 사용되게 합니다.
9. 루프가 종료된 후, 도출된 K_{outer} 개의 성능 지표들을 평균 내어 해당 머신러닝 방법론의 최종적인 일반화 성능과 표준 오차(Standard Error)를 확정합니다.

14.6.2. Bayesian Model Selection

1. Bayesian Model Selection
Bayesian model selection은 주어진 데이터 D를 바탕으로 여러 후보 모델 $M_1,\dots,M_K$ 중 가장 확률적으로 타당한 모델을 찾는 과정이다. bayes' theorem에 의하면 데이터 D가 주어졌을 때, 모델 M_k 가 정답일 posterior probability는 다음과 같다. 이에 대해 MAP을 적용해 최적의 모델 M^*을 선정할 수 있다.

$$p(M_k \mid D) = \frac{p(D \mid M_k)p(M_k)}{p(D)}$$
$$M^* = \arg\max_{M_k} p(M_k \mid D)$$

모든 모델이 선택될 확률이 동일한 사전 확률 uniform prior $p(M_k)=\frac{1}{K}$에서, 모든 모델에 대한 MAP estimate은 model evidence를 최대화하는 모델을 선택하는 것과 같다. 예를 들어, 다음 그림과 같이 y축이 evidence, x축이 가능한 모든 데이터셋 D인 경우을 생각할 수 있다. 모델이 생성할 수 있는 데이터에 대한 probability의 총합은 1인데, parameter가 많은 모델은 다양한 데이터를 생성할 수 있으므로 evidence는 전반적으로 낮아지게 된다. bayesian model selection에서는 이에 따라 S에 대해 evidence가 높은 M_1을 선택하게 된다. 이를 automatic occam's razor로 이해할 수 있다.

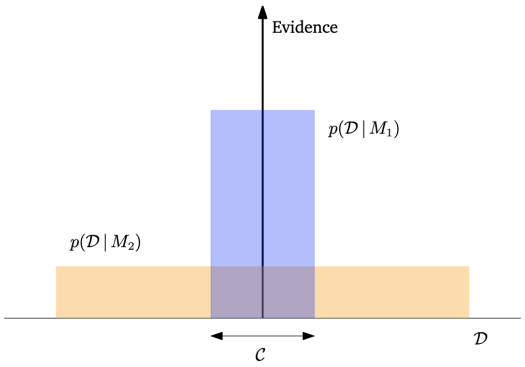
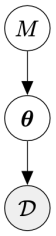


Figure 8.14
Bayesian inference embodies Occam's razor. The horizontal axis describes the space of all possible datasets $\mathcal{D}$. The evidence (vertical axis) evaluates how well a model predicts available data. Since $p(\mathcal{D} \mid M_i)$ needs to integrate to 1, we should choose the model with the greatest evidence.

2. Using Marginal Likelihood
이때 likelihood가 어떻게 계산되는지 알아보자. 앞서 다룬 것처럼 bayesian에서는 모델의 parameter를 고정된 상수 값이 아닌 random variable로 취급한다. 이에 따라 유한한 수의 모델 $M = \{M_1, \cdots, M_K\}$가 존재하고 그 parameter를 θ_k라 하자. 모델과 parameter, 모델이 생성한 데이터 D에 대해서는 다음이 성립한다.

$$M_k \sim p(M), \quad \boldsymbol{\theta}_k \sim p(\boldsymbol{\theta} \mid M_k), \quad D \sim p(D \mid \boldsymbol{\theta}_k)$$

이를 directed graphical model로도 나타낼 수 있다.



위 수식의 posterior $p(M_k \mid D)$는 parameter에 의존하지 않는다. 모델의 최적 parameter $\boldsymbol{\theta}_k$를 찾는 bayes' theorem은 다음과 같고, 이에 따라 $p(D \mid M_k)$도 유도될 수 있는데, 이 경우 parameter가 적분에 의해 사라지기 때문이다. 이런 independency는 d-separation으로도 확인할 수 있다. 즉, 이는 marginal likelihood로 어떤 모델이 해당 데이터셋을 잘 설명하는지를 나타낸 것이다.

$$p(\boldsymbol{\theta}_k \mid D, M_k) = \frac{p(D \mid \boldsymbol{\theta}_k, M_k) p(\boldsymbol{\theta}_k \mid M_k)}{p(D \mid M_k)}$$
$$p(D \mid M_k) = \int p(D \mid \boldsymbol{\theta}_k, M_k) p(\boldsymbol{\theta}_k \mid M_k) d\boldsymbol{\theta}_k$$

앞서 다룬 bayesian inference $p(\boldsymbol{x} \mid X)$는 기존 데이터를 활용해 새로운 데이터를 예측했지만, 이 likelihood는 기존 데이터를 모델이 얼마나 잘 예측하는지를 나타낸다.

3. Bayes Factors
주어진 데이터셋 $\mathcal{D}$에서 두 probability model M_1, M_2를 비교하는 상황을 생각하자. 각각의 posterior의 비를 계산하면 다음과 같다.

$$\underbrace{\frac{p(M_1 \mid \mathcal{D})}{p(M_2 \mid \mathcal{D})}}_{\text{posterior odds}} = \frac{\frac{p(\mathcal{D} \mid M_1) p(M_1)}{p(\mathcal{D})}}{\frac{p(\mathcal{D} \mid M_2) p(M_2)}{p(\mathcal{D})}} = \underbrace{\frac{p(M_1)}{p(M_2)}}_{\text{prior odds}} \underbrace{\frac{p(\mathcal{D} \mid M_1)}{p(\mathcal{D} \mid M_2)}}_{\text{Bayes factor}}$$

posterior의 비는 **Posterior Odds**라고 한다. prior의 비는 **Prior Odds**라고 하며, prior belief(사전 믿음)이 M_2보다 M_1을 얼마나 더 선호하는지를 나타낸다. likelihood의 비는 **Bayes Factor**라고 하며, M_2보다 M_1이 $\mathcal{D}$를 얼마나 더 잘 예측하는지를 나타낸다.

만약 uniform prior를 사용하는 경우 prior odd는 1이 되고, posterior odd는 bayes factor와 같아진다.

15. Linear Regression

본 장에서는 linear regression 문제를 풀어본다. 특히 MLE, MAP, bayesian linear regression 등의 측면에서 살펴본다. 참고로 bayesian linear regression은 모델 parameter에 대한 더 높은 수준의 추론을 가능하게 하므로, MLE 또는 MAP 에서 발생하는 문제 중 일부를 해결할 수 있다.

15.1. Linear Regression

15.1.1. Problem Formulation

Regression(회귀)은 input $x \in \mathbb{R}^D$를 대응되는 function value $f(x) \in \mathbb{R}$에 매핑하는 function f를 찾는 것을 그 목적으로 한다. 이는 ML, DL, classification 등의 핵심 아이디어로 활용된다. 좋은 regression function을 찾으려면 model selection, finding good parameter, overfitting 등을 고려해야 한다.

Linear Regression(선형 회귀)은 주어진 독립 변수와 종속 변수 간의 선형 상관관계를 모델링하는 알고리즘이다. 즉, 주어진 데이터들과의 오차를 최소화하는 최적의 linear function(hyperplane) f를 찾는 것을 그 목적으로 한다.

linear regression에서는 function value(예측값) $f(\boldsymbol{x})$와 실제 데이터 간의 오차(error, noise)가 존재한다. 이에 따라 probabilistic approach를 사용하고, 그 likelihood function은 다음과 같다. $x \in \mathbb{R}^D$는 input이고, $y \in \mathbb{R}$는 noisy function value(target)이다. 이때 noise의 variance는 알고 있다고 가정한다.

$$p(y|\boldsymbol{x}) = \mathcal{N}(y|f(\boldsymbol{x}), \sigma^2)$$

이때의 $\boldsymbol{x}$와 y의 function relationship은 다음과 같이 정의된다. 이때 $\epsilon \sim \mathcal{N}(0, \sigma^2)$는 i.i.d.(각 데이터의 error에 대해.) gaussian noise이다.

$$y = f(\boldsymbol{x}) + \epsilon$$

또한 linear regression에서는 다음과 같이 parameter $\boldsymbol{\theta} \in \mathbb{R}^D$가 선형으로 존재한다고 가정한다. 즉, 예측값 $f(\boldsymbol{x})$은 입력 변수 $\boldsymbol{x}$와 파라미터 $\boldsymbol{\theta}$의 linear combination으로 표현된다. 본 장에서는 좋은 parameter $\boldsymbol{\theta}$를 찾는 데에 집중한다. (θ_0 항은 편의를 위해 생략했다.)

$$f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{\theta} = \theta_1 x_1 + \cdots + \theta_D x_D$$

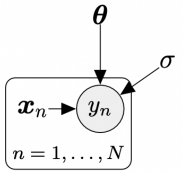
즉, N개의 $\boldsymbol{x}_n \in \mathbb{R}^D$와 이에 대응되는 observation/target $y_n \in \mathbb{R}$을 포함하는 training set $\mathcal{D}$가 $\{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_N, y_N)\}$라고 하면, linear regression은 다음과 같이 정의된다. ϵ이 gaussian noise이므로, y는 mean이 $\boldsymbol{x}^T\boldsymbol{\theta}$이고 variance가 σ^2인 gaussian distribution이 된다. 이때 σ^2는 free model parameter가 아닌 것으로 취급하고, optimization을 통해 parameter가 고정된 이후 계산한다.

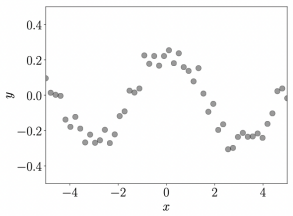
$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(y|\boldsymbol{x}^T\boldsymbol{\theta}, \sigma^2) \Longleftrightarrow y = \boldsymbol{x}^T\boldsymbol{\theta} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

또한 각 y_i는 $i \neq j$인 y_j, x_j에 대해 independent하므로 다음과 같이 정리할 수 있다.

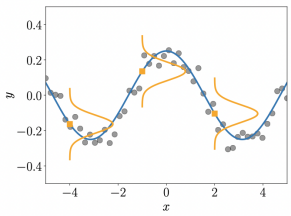
$$\begin{aligned} p(Y|X, \boldsymbol{\theta}) &= p(y_1, \ldots, y_N|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N, \boldsymbol{\theta}) \\ &= \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta}) = \prod_{n=1}^{N} \mathcal{N}(y_n|\boldsymbol{x}_n^\top\boldsymbol{\theta}, \sigma^2) \end{aligned}$$

이에 대한 graphical model은 다음과 같다.





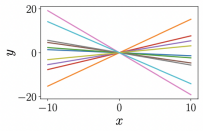
(a) Regression problem: observed noisy function values from which we wish to infer the underlying function that generated the data.



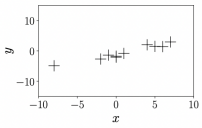
(b) Regression solution: possible function that could have generated the data (blue) with indication of the measurement noise of the function value at the corresponding inputs (orange distributions).

Figure 9.1
(a) Dataset;
(b) possible solution to the regression problem.

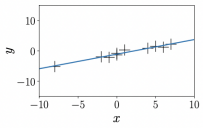
Figure 9.2 Linear regression example. (a) Example functions that fall into this category; (b) training set; (c) maximum likelihood estimate.



(a) Example functions (straight lines) that can be described using the linear model in (9.4).



(b) Training set.



(c) Maximum likelihood estimate.

왜 noise가 gaussian을 따르는가? "실제 데이터에서 발생하는 오차 ϵ 은 모델에 포함되지 않은 수많은 독립적인 관측 불가능한 변수들과 측정 노이즈들의 합으로 간주할 수 있습니다. 중심 극한 정리에 따르면, 서로 독립적인 수많은 무작위 변수들의 합은 그 개별 변수들의 분포 형태와 무관하게 가우시안 분포에 수렴합니다. 따라서 현실 세계의 복합적인 노이즈를 모델링할 때 가우시안 분포를 가정하는 것은 통계적으로 가장 합리적입니다."

"자유 모델 파라미터(Free Model Parameter)는 수학적 모델이나 기계 학습 모델에서 사전에 값이 고정되지 않고, 주어진 데이터를 통해 학습되거나 추정되어야 하는 변수를 의미합니다."

15.2. Parameter Estimation on Linear Regression

15.2.1. MLE

1. MLE
linear regression에서 원하는 parameter $\theta_{\text{ML}} \in \mathbb{R}^D$를 찾는 데에 MLE를 사용할 수 있다. 즉, 다음과 같이 likelihood를 최대화하는 optimization 문제로 생각할 수 있다.

$$\boldsymbol{\theta}_{\text{ML}} = \arg\max_{\theta} p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\theta})$$

이에 대한 optimization은 negative log를 씌워 수행한다. 이때 각 데이터들은 i.i.d.이므로 각 probability의 합으로 나타낼 수 있다.

$$-\log p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\theta}) = -\sum_{n=1}^{N} \log p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta})$$

y_i는 mean이 $\boldsymbol{x}_n^T\boldsymbol{\theta}$, variance가 σ^2인 gaussian distribution이므로 정리하면 다음과 같다. 또한 이 수식을 최소화하는 θ를 찾는 것이므로, loss function을 정의할 수 있다. 이때 $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^\top \in \mathbb{R}^{N \times D}$, $\boldsymbol{Y} =$

$[y_1, \cdots, y_N]^T \in \mathbb{R}^N$이라 한다.

$$-\log p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta}) = \frac{1}{2\sigma^2}(y_n - \boldsymbol{x}_n^\top \boldsymbol{\theta})^2 + \text{const}$$

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= \frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n - \boldsymbol{x}_n^\top \boldsymbol{\theta})^2 \\ &= \frac{1}{2\sigma^2}(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\theta})^\top(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\theta}) = \frac{1}{2\sigma^2}\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\theta}\|^2\end{aligned}$$

이때 $\mathcal{L}(\boldsymbol{\theta})$가 convex이므로, gradient descent를 하는 대신 다음과 같이 gradient를 계산하고 0이 되는 지점의 θ_{ML}를 얻을 수 있다. 이때 $\boldsymbol{X}$의 rank가 D인 것을 가정하면 $\boldsymbol{X}^T\boldsymbol{X}$가 positive definite이므로 inverse matrix가 존재하므로 정리할 수 있다.

$$\begin{aligned}\frac{d\mathcal{L}}{d\boldsymbol{\theta}} &= \frac{d}{d\boldsymbol{\theta}}\left(\frac{1}{2\sigma^2}(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\theta})^\top(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\theta})\right) \\ &= \frac{1}{2\sigma^2}\frac{d}{d\boldsymbol{\theta}}\left(\boldsymbol{Y}^\top\boldsymbol{Y} - 2\boldsymbol{Y}^\top\boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{\theta}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\theta}\right) \\ &= \frac{1}{\sigma^2}(-\boldsymbol{Y}^\top\boldsymbol{X} + \boldsymbol{\theta}^\top\boldsymbol{X}^\top\boldsymbol{X}) \in \mathbb{R}^{1\times D}\end{aligned}$$

$$\begin{aligned}\frac{d\mathcal{L}}{d\boldsymbol{\theta}} = \boldsymbol{0}^\top &\iff \boldsymbol{\theta}_{\text{ML}}^\top\boldsymbol{X}^\top\boldsymbol{X} = \boldsymbol{y}^\top\boldsymbol{X} \\ &\iff \boldsymbol{\theta}_{\text{ML}}^\top = \boldsymbol{Y}^\top\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X})^{-1} \\ &\iff \boldsymbol{\theta}_{\text{ML}} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{Y}\end{aligned}$$

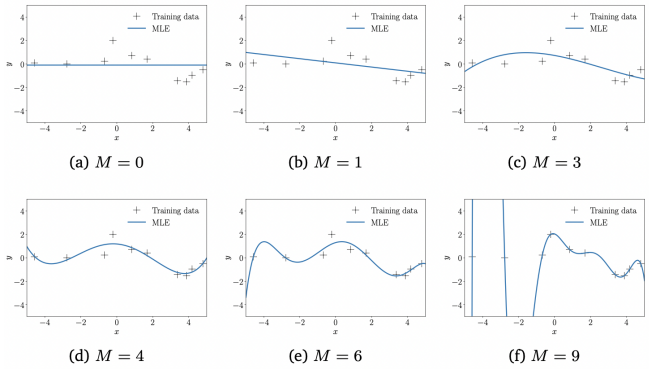
2. MLE as Orthogonal Projection
앞서 다룬 것처럼 linear regression model은 $y = \boldsymbol{x}^T\boldsymbol{\theta} + \epsilon$와 같이 정의되고, MLE로 찾은 최적의 parameter는 $\boldsymbol{\theta}_{\text{ML}} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{Y}$이다. 이 parameter로 생성한 noise가 없는 예측값 $\hat{\boldsymbol{y}}$는 다음과 같다.

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\theta}_{ML} = \boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top Y$$

즉, MLE에 의해 찾은 parameter로 얻는 prediction은 $\boldsymbol{X}$가 생성하는 column space에 $\boldsymbol{Y}$를 orthogonal projection한 것과 같다.

이때 뒤에서 다룰 것처럼 feature matrix를 사용하는 경우 $\boldsymbol{X}$ 대신 $\boldsymbol{\Phi}$로 작성이 가능하다. 또한 feature matrix가 orthonormal할 경우 $\boldsymbol{\Phi}\boldsymbol{\Phi}^T y$로 더 간단히 나타낼 수 있다.

당연하게도 다음 그림과 같이 overfitting이 발생할 수 있다.



(a) $M = 0$ (b) $M = 1$ (c) $M = 3$

(d) $M = 4$ (e) $M = 6$ (f) $M = 9$

Figure 9.5 Maximum likelihood fits for different polynomial degrees M.

15.2.2. MLE with Features

앞서 다룬 MLE에서는 데이터를 linear function으로 예측하지만, 실제로는 linear function만으로 데이터를 충분히 예측하지 못할 수 있다. linear regression은 parameter의 linearity만을 고려하므로, 다음과 같이 $\boldsymbol{x}$에 대한 non-linear transformation $\phi : \mathbb{R}^D \to \mathbb{R}^K$을 수행한 뒤 $\boldsymbol{\theta}$와 linear combination하는 것으로 non-linearity를 활용할 수 있다.

$$p(y \mid x, \theta) = \mathcal{N}(y \mid \phi^\top(x)\theta, \sigma^2) \iff y = \phi^\top(x)\theta + \epsilon = \sum_{k=0}^{K-1} \theta_k \phi_k(x) + \epsilon$$

이때 $\phi_k : \mathbb{R}^D \to \mathbb{R}$는 feature vector ϕ의 k번째 component로, 다음과 같이 정의된다.

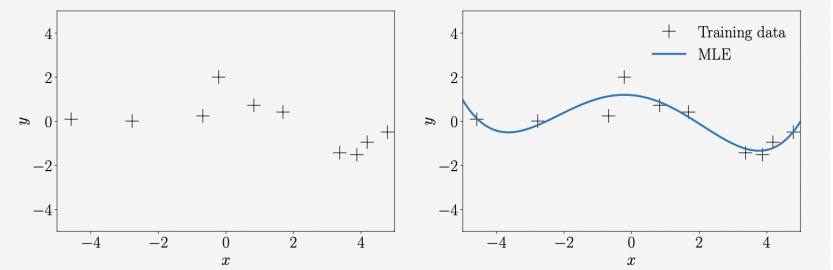
$$\phi(x) = \begin{bmatrix} \phi_0(x) \\ \phi_1(x) \\ \vdots \\ \phi_{K-1}(x) \end{bmatrix}$$

training input $x_n \in \mathbb{R}^D$와 target $y_n \in \mathbb{R}$가 존재할 때, non-linear function ϕ를 활용해 정의되는 matrix를 **Feature Matrix** 또는 Design Matrix라고 하고, 이는 다음과 같다. ϕ가 원본 데이터를 새로운 feature space로 mapping하는 것을 생각해 보면 이해할 수 있다.

$$\Phi = \begin{bmatrix} \phi^\top(x_1) \\ \vdots \\ \phi^\top(x_N) \end{bmatrix} = \begin{bmatrix} \phi_0(x_1) & \cdots & \phi_{K-1}(x_1) \\ \phi_0(x_2) & \cdots & \phi_{K-1}(x_2) \\ \vdots & \ddots & \vdots \\ \phi_0(x_N) & \cdots & \phi_{K-1}(x_N) \end{bmatrix} \in \mathbb{R}^{N \times K}$$

수식적으로는 단순히 X가 Φ로 치환되는 것으로 이해할 수 있으므로, negative log likelihood를 다음과 같은 수식으로 나타낼 수 있고, θ_{ML}도 동일하게 나타낼 수 있다. 마찬가지로 θ_{ML}은 Φ의 rank가 K일 때 이렇게 나타낼 수 있다.

$$-\log p(Y \mid X, \theta) = \frac{1}{2\sigma^2}(y - \Phi\theta)^\top(y - \Phi\theta) + \text{const}$$
$$\theta_{\text{ML}} = (\Phi^\top\Phi)^{-1}\Phi^\top y$$



(a) Regression dataset.

(b) Polynomial of degree 4 determined by maximum likelihood estimation.

예를 들어, 2차 다항식에 대한 feature matrix는 다음과 같다.

$$\Phi = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 \end{bmatrix}$$

15.2.3. Estimating Noise Variance

지금까지는 noise variance σ^2를 알고 있다고 가정했는데, parameter가 고정된 이후 σ^2에 대한 MLE를 적용해 그 값을 구할 수 있다. 이는 parameter를 구할 때와 동일한 수식에 대해 σ^2로 differentiate한 뒤, 0인 지점을 찾으면 된다.

$$\begin{aligned} \log p(\mathcal{Y}|\mathcal{X}, \boldsymbol{\theta}, \sigma^2) &= \sum_{n=1}^{N} \log \mathcal{N}(y_n | \phi^\top(\boldsymbol{x}_n)\boldsymbol{\theta}, \sigma^2) \\ &= \sum_{n=1}^{N} \left(-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log\sigma^2 - \frac{1}{2\sigma^2}(y_n - \phi^\top(\boldsymbol{x}_n)\boldsymbol{\theta})^2 \right) \\ &= -\frac{N}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\underbrace{\sum_{n=1}^{N}(y_n - \phi^\top(\boldsymbol{x}_n)\boldsymbol{\theta})^2}_{=:s} + \text{const} \end{aligned}$$

$$\frac{\partial \log p(\mathcal{Y}|\mathcal{X}, \boldsymbol{\theta}, \sigma^2)}{\partial \sigma^2} = -\frac{N}{2\sigma^2} + \frac{1}{2\sigma^4}s = 0$$

즉, variance σ^2는 다음과 같이 noise-free인 function value와 그에 대응되는 y_n 간의 squared distance에 대한 empirical mean이다. 즉, MSE(Mean Squared Error)이다.

$$\sigma^2_{\text{ML}} = \frac{s}{N} = \frac{1}{N}\sum_{n=1}^{N}(y_n - \phi^\top(\boldsymbol{x}_n)\boldsymbol{\theta})^2$$

15.2.4. Quality of Linear Regression Model

linear regression에서 MLE를 통해 얻는 모델의 quality는 다음과 같이 앞서 다룬 negative log likelihood의 값을 구해 판단할 수 있다.

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2\sigma^2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}\|^2$$

실제 평가에서는 σ^2를 제외하고 다음과 같은 **RMSE(Root Mean Square Error)**를 사용해 모델의 quality를 평가한다. 즉, MSE에 root를 씌운 것이다. σ^2는 free model parameter가 아니고, 단위 문제가 존재하므로 제외한다고 한다.

$$\sqrt{\frac{1}{N}||\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta}||^2} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(y_n - \phi^\top(\boldsymbol{x}_n)\boldsymbol{\theta})^2}$$

15.2.5. MAP

MAP를 적용해 MLE에서의 overfitting 문제를 완화할 수 있다. 즉, prior를 고려해 posterior에 대한 optimization problem을 푼다. 수식으로 나타내면 다음과 같다.

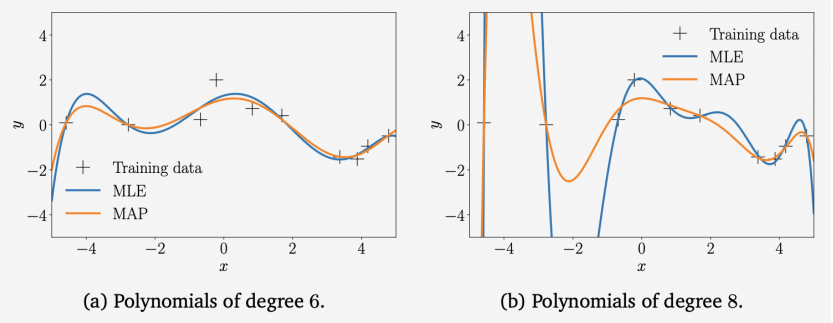
$$\begin{aligned} -\log p(\boldsymbol{\theta}|\mathcal{X}, \mathcal{Y}) &= -\log p(\mathcal{Y}|\mathcal{X}, \boldsymbol{\theta}) - \log p(\boldsymbol{\theta}) + \text{const} \\ \boldsymbol{\theta}_{\text{MAP}} &= \arg\min_{\boldsymbol{\theta}}\{-\log p(\mathcal{Y}|\mathcal{X}, \boldsymbol{\theta}) - \log p(\boldsymbol{\theta})\} \end{aligned}$$

parameter $\boldsymbol{\theta}$에 대한 conjugate gaussian prior를 $p(\boldsymbol{\theta}) = \mathcal{N}(0, b^2\boldsymbol{I})$라고 하면 다음과 같이 수식을 작성할 수 있다. 마지막 줄의 수식에서 알 수 있듯이, MAP에 의해 추가되는 항은 regularization을 수행한다. 마지막 줄의 수식에서 첫번째 항을 Data-fit Term 또는 Misfit Term이라고 하고, 두번째 항을 Regularizer라고 하며, λ를 Regularization Parameter라고 한다.

$$\begin{aligned} -\log p(\boldsymbol{\theta}|\mathcal{X}, \mathcal{Y}) &= \frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta})^\top(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta}) + \frac{1}{2b^2}\boldsymbol{\theta}^\top\boldsymbol{\theta} + \text{const} \\ \mathcal{L}(\boldsymbol{\theta}) &= \frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta})^\top(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta}) + \frac{1}{2b^2}\boldsymbol{\theta}^\top\boldsymbol{\theta} \\ &= ||\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta}||^2 + \frac{1}{2b^2}||\boldsymbol{\theta}||_2^2 \\ &= ||\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta}||^2 + \lambda||\boldsymbol{\theta}||_2^2 \end{aligned}$$

이를 differentiate하고 0과 같다고 둬서 풀 수 있다.

$$\begin{aligned} -\frac{d\log p(\boldsymbol{\theta}|\mathcal{X}, \mathcal{Y})}{d\boldsymbol{\theta}} &= \frac{1}{\sigma^2}(\boldsymbol{\theta}^\top\boldsymbol{\Phi}^\top\boldsymbol{\Phi} - \boldsymbol{y}^\top\boldsymbol{\Phi}) + \frac{1}{b^2}\boldsymbol{\theta}^\top \\ \boldsymbol{\theta}_{\text{MAP}} &= \left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \frac{\sigma^2}{b^2}\boldsymbol{I}\right)^{-1}\boldsymbol{\Phi}^\top\boldsymbol{y} \end{aligned}$$



(a) Polynomials of degree 6. (b) Polynomials of degree 8.

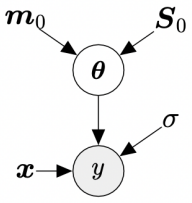
15.3. Bayesian Linear Regression

15.3.1. Bayesian Linear Regression

1. Bayesian Linear Regression
Bayesian Linear Regression에서는 linear regression을 bayesian inference로 해결한다. 즉, MLE와 MAP과 같이 point estimate를 계산하는 대신 parameter에 대한 전체 posterior distribution을 고려하고, 모든 가능성 있는 parameter에 대한 평균을 계산한다.

이때 prior와 likelihood는 다음과 같다고 가정한다.

$$p(\theta) = \mathcal{N}(m_0, S_0)$$
$$p(y \mid x, \theta) = \mathcal{N}(y \mid \phi^\top(x)\theta, \sigma^2)$$



2. Prior Prediction
학습 데이터를 관측하기 전에, prior만을 사용해 prediction해보자. 즉, 이 경우 다음과 같은 수식으로 prediction에 대한 distribution을 얻을 수 있다. x_*는 학습 데이터에 없는 새로운 데이터이고, $y_* = \phi^\top(x_*)\theta + \epsilon$는 그 예측값이다.

$$p(y_* \mid x_*) = \int p(y_* \mid x_*, \theta)p(\theta)d\theta$$

이때 likelihood와 prior가 모두 gaussian이므로 prediction 또한 gaussian이다. 이때 다음과 같이 mean의 linearity와 variance에 대해 $\mathbb{V}[Ax] = A\mathbb{V}[x]A^T$가 성립한다는 점, θ와 ϵ이 서로 independent하다는 점을 활용해 mean과 variance를 구할 수 있다. 이때 σ^2는 noise에 의한 항으로, noise-free function value를 얻으려면 해당 항을 뺀 gaussian distribution을 활용하면 된다. 이렇게 두 gaussian distribution에 대한 곱의 적분은 mean과 variance만을 구하는 것으로 구할 수 있음에 유의하자. 또한 이때 covariance matrix를 사용하지 않은 것은 y_*가 scalar이기 때문이라는 점도 상기하자.

$$\mathbb{E}[y_*] = \mathbb{E}_{\theta,\epsilon}[\phi^\top(x_*)\theta + \epsilon] = \phi^\top(x_*)\mathbb{E}[\theta] + \mathbb{E}[\epsilon]$$
$$\mathbb{V}[y_*] = \mathbb{V}_{\theta,\epsilon}[\phi^\top(x_*)\theta + \epsilon] = \mathbb{V}_\theta[\phi^\top(x_*)\theta] + \mathbb{V}_\epsilon[\epsilon]$$

이때 $\mathbb{E}[y_*]$는 x_*가 주어진 상황을 가정하므로, $\mathbb{E}[y_* \mid x_*]$와 같은 것으로 이해할 수 있다.

$$p(y_* \mid x_*) = \mathcal{N}(\phi^\top(x_*)m_0, \phi^\top(x_*)S_0\phi(x_*) + \sigma^2)$$

3. Posterior Distribution/Prediction
데이터셋 $\mathbf{x}_n \in \mathbb{R}^D$와 대응되는 $\mathbf{y}_n$이 $n = 1, \cdots, N$에 대해 주어졌을 때, bayes' theorem에 의해 다음과 같이 posterior를 구할 수 있다.

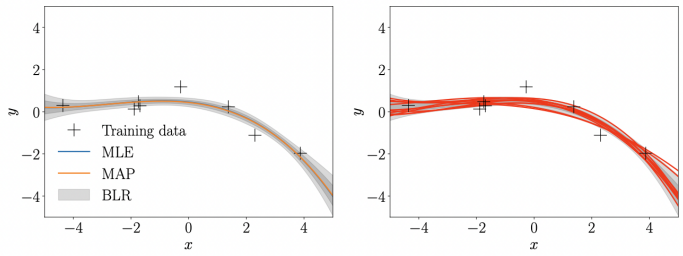
$$p(\theta|\mathcal{X}, \mathcal{Y}) = \frac{p(\mathcal{Y}|\mathcal{X}, \theta)p(\theta)}{p(\mathcal{Y}|\mathcal{X})}$$

이때 posterior는 gaussian이고, 다음과 같은 mean과 variance를 가진다. 이에 대한 증명은 꽤 복잡하므로 교재를 참고하자.

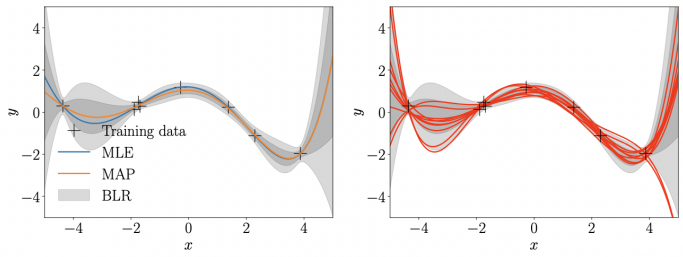
$$
\begin{aligned}
p(\theta|\mathcal{X},\mathcal{Y}) &= \mathcal{N}(\theta|\mathbf{m}_N,\mathbf{S}_N) \\
\mathbf{S}_N &= (\mathbf{S}_0^{-1}+\sigma^{-2}\mathbf{\Phi}^\top\mathbf{\Phi})^{-1} \\
\mathbf{m}_N &= \mathbf{S}_N(\mathbf{S}_0^{-1}\mathbf{m}_0+\sigma^{-2}\mathbf{\Phi}^\top\mathbf{y})
\end{aligned}
$$

이런 posterior를 활용해서, 다음과 같이 prediction의 distribution을 얻을 수 있다. 즉, mean과 variance만 달라지고 prior에서와 수식은 동일하다. 마찬가지로 σ^2 항을 빼면 noise-free function value를 얻을 수 있다.

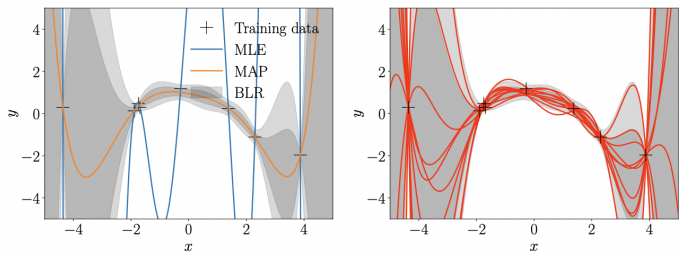
$$
\begin{aligned}
p(y_*|\mathcal{X},\mathcal{Y},\mathbf{x}_*) &= \int p(y_*|\mathbf{x}_*,\theta)p(\theta|\mathcal{X},\mathcal{Y})\mathrm{d}\theta \\
&= \int \mathcal{N}(y_*|\phi^\top(\mathbf{x}_*)\theta,\sigma^2)\mathcal{N}(\theta|\mathbf{m}_N,\mathbf{S}_N)\mathrm{d}\theta \\
&= \mathcal{N}(y_*|\phi^\top(\mathbf{x}_*)\mathbf{m}_N,\phi^\top(\mathbf{x}_*)\mathbf{S}_N\phi(\mathbf{x}_*)+\sigma^2)
\end{aligned}
$$



(a) Posterior distribution for polynomials of degree $M = 3$ (left) and samples from the posterior over functions (right).



(b) Posterior distribution for polynomials of degree $M = 5$ (left) and samples from the posterior over functions (right).



(c) Posterior distribution for polynomials of degree $M = 7$ (left) and samples from the posterior over functions (right).

Figure 9.11 Bayesian linear regression. Left panels: Shaded areas indicate the 67% (dark gray) and 95% (light gray) predictive confidence bounds. The mean of the Bayesian linear regression model coincides with the MAP estimate. The predictive uncertainty is the sum of the noise term and the posterior parameter uncertainty, which depends on the location of the test input. Right panels: sampled functions from the posterior distribution.

15.3.2. Computing the Marginal Likelihood of Linear Regression

bayesian model selection 등을 고려하면 marginal likelihood를 계산하는 것이 중요할 수 있다. marginal

likelihood는 다음과 같다. multivariate이므로 covariance를 사용하는 점 유의하자.

$$\begin{aligned} p(\mathcal{Y}|\mathcal{X}) &= \int p(\mathcal{Y}|\mathcal{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta} \\ &= \int \mathcal{N}(\boldsymbol{y}|\boldsymbol{X}\boldsymbol{\theta}, \sigma^2 \boldsymbol{I}) \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{m}_0, \boldsymbol{S}_0) d\boldsymbol{\theta} \end{aligned}$$

이때 $\boldsymbol{\theta}$와 $p(y_n|\boldsymbol{x}_n, \boldsymbol{\theta})$는 gaussian distribution이고, 다음과 같다. 이에 따라 marginal likelihood 또한 gaussian이다.

$$\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{m}_0, \boldsymbol{S}_0)$$

$$y_n|\boldsymbol{x}_n, \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{x}_n^\top \boldsymbol{\theta}, \sigma^2)$$

이에 따라 mean과 covariance를 구하면 다음과 같다. 여기에서도 independence가 활용되었다.

$$\mathbb{E}[\mathcal{Y}|\mathcal{X}] = \mathbb{E}_{\boldsymbol{\theta}, \boldsymbol{\epsilon}}[\boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{\epsilon}] = \boldsymbol{X}\mathbb{E}_{\boldsymbol{\theta}}[\boldsymbol{\theta}] = \boldsymbol{X}\boldsymbol{m}_0$$

$$\begin{aligned} \mathrm{Cov}[\mathcal{Y}|\mathcal{X}] &= \mathrm{Cov}_{\boldsymbol{\theta}, \boldsymbol{\epsilon}}[\boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{\epsilon}] = \mathrm{Cov}_{\boldsymbol{\theta}}[\boldsymbol{X}\boldsymbol{\theta}] + \sigma^2 \boldsymbol{I} \\ &= \boldsymbol{X}\mathrm{Cov}_{\boldsymbol{\theta}}[\boldsymbol{\theta}]\boldsymbol{X}^\top + \sigma^2 \boldsymbol{I} = \boldsymbol{X}\boldsymbol{S}_0\boldsymbol{X}^\top + \sigma^2 \boldsymbol{I} \end{aligned}$$

정리하면 gaussian distribution은 다음과 같다.

$$p(\mathcal{Y}|\mathcal{X}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{X}\boldsymbol{m}_0, \boldsymbol{X}\boldsymbol{S}_0\boldsymbol{X}^\top + \sigma^2 \boldsymbol{I})$$

16. Dimensionality Reduction with PCA

high dimensional data는 그 자체로 분석하거나 시각화하기 어렵고, 처리하는 데에 비용이 많이 들며, redundancy가 존재한다. 이에 따라 dimensionality reduction을 적용할 수 있다. 본 장에서는 linear dimensionality reduction 기법인 PCA에 대해서 다룬다.

16.1. Dimensionality Reduction with PCA

16.1.1. Problem Formulation

PCA(Principal Component Analysis, 주성분 분석)는 고차원의 데이터를 정보 손실을 최소화하면서 저차원으로 줄이는 기법이다.

empirical mean이 0이고 i.i.d.인 데이터셋을 나타내는 matrix $X = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N]$이 존재할 때, PCA에서는 데이터 $\boldsymbol{x}_n \in \mathbb{R}^D$에 대해 lower dimensionality를 가지는 projection $\tilde{\boldsymbol{x}}_n \in \mathbb{R}^D$를 찾는 것을 목적으로 한다. $\tilde{\boldsymbol{x}}_n$는 $\boldsymbol{x}$를 projection하여 재구성한 것으로, $\boldsymbol{x}$와 동일한 dimension을 가지지만 subspace에 존재하는 vector이다.

이때 X는 mean이 0이고, 이에 따라 variance가 다음과 같이 정의된다.

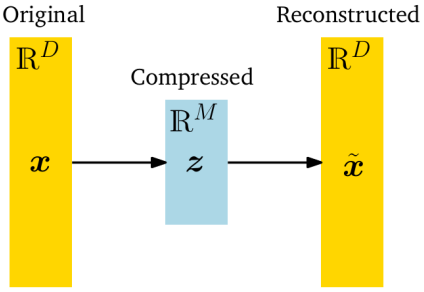
$$S = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^T = \frac{1}{N} X X^T$$

$\boldsymbol{z}_n$는 low dimension을 가지는 vector로, 다음과 같이 정의된다. $B \in \mathbb{R}^{D \times M}$는 column이 orthonormal한 projection matrix이다. $\tilde{\boldsymbol{x}}_n$는 $\boldsymbol{z}_n$을 reconstruct한 것으로 이해할 수 있다.

$$\boldsymbol{z}_n = B^T \boldsymbol{x}_n$$

즉, $\tilde{\boldsymbol{x}}_n$는 다음과 같다. 즉, B^T는 encoder, B는 decoder로 이해할 수 있다.

$$\tilde{\boldsymbol{x}}_n = B\boldsymbol{z}_n$$



16.1.2. Key Steps of PCA in Practice

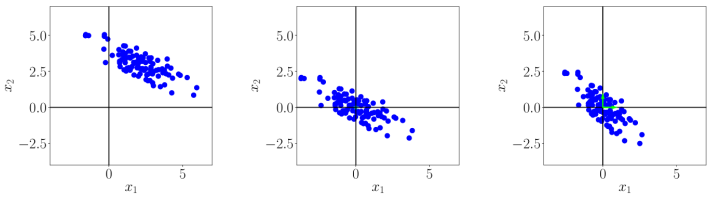
뒤에서 설명할 내용들을 고려해서 정리한, PCA를 실제로 수행하는 process는 다음과 같다.

1. Mean Subtraction: 데이터셋의 empirical mean μ를 구한 뒤, 각 데이터를 μ로 빼서 mean을 0으로 맞춘다.
2. Standardization: dimension $d = 1, \cdots, D$ 별로 standard deviation σ_d를 구한 뒤, 각 데이터의 각 dimension을 σ_d로 나눠 variance를 1로 맞춘다.
3. Eigen Decomposition of Covariance Matrix: 데이터셋의 covariance matrix를 구한 뒤 해당 matrix의 eigen value/vector를 구한다. covariance matrix는 symmetric이므로, spectral theorem에 의해 항상 ONB를 구할 수 있다. 이후 eigen value가 큰 상위 eigen vector들을 normalize해 subspace ONB를 만든 뒤, 이 ONB를 column으로 하는 matrix B를 구성한다.
4. Projection: B를 활용해 다음과 같이 데이터를 손실이 가장 적은 low-dimensonality subspace로 projection할 수 있다.

   $$\tilde{x}_n = BB^\top x_n, \quad z_n = B^\top x_n$$

   이때 기존 데이터의 original data space의 vector가 필요하다면 다음과 같이 mean subtraction, standardization을 반대로 연산한다.

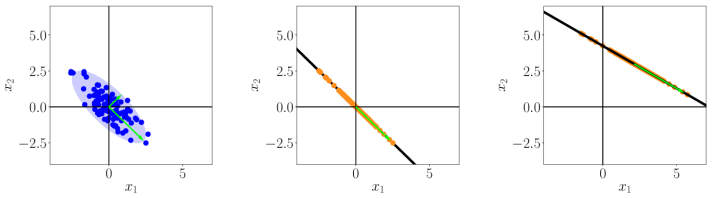
   $$\tilde{x}_n^{(d)} \leftarrow \tilde{x}_n^{(d)} \sigma_d + \mu_d$$



(a) Original dataset.

(b) Step 1: Centering by subtracting the mean from each data point.

(c) Step 2: Dividing by the standard deviation to make the data unit free. Data has variance 1 along each axis.



(d) Step 3: Compute eigenvalues and eigenvectors (arrows) of the data covariance matrix (ellipse).

(e) Step 4: Project data onto the principal subspace.

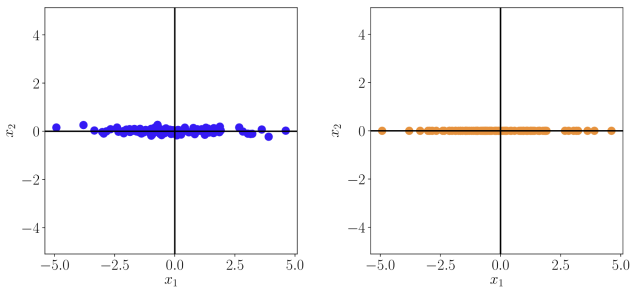
(f) Undo the standardization and move projected data back into the original data space from (a).

Figure 10.11 Steps of PCA. (a) Original dataset; (b) centering; (c) divide by standard deviation; (d) eigendecomposition; (e) projection; (f) mapping back to original data space.

16.2. Maximum Variance Perspective

16.2.1. Maximum Variance Perspective

PCA에서는 다음 그림과 같이 특정 dimensionality의 성분을 제거한다. 이 그림에서도 알 수 있듯이, 만약 잘못된 dimension을 제거할 경우 데이터 distribution이 작아지거나 아예 사라질 수 있고, 적절히 제거한다면 distribution이 유지된다. 이에 따라 PCA에서는 데이터의 low-dimensional representation의 variance를 최대로 하는 것을 목적으로 한다. $\boldsymbol{x}$의 variance는 이미 고정되어 있고, B에 의한 $\boldsymbol{z}$의 variance가 최대가 되도록 한다. 이때 variance는 mean에 영향을 받지 않으므로 mean은 모두 0으로 가정한다.



(a) Dataset with x_1 and x_2 coordinates.

(b) Compressed dataset where only the x_1 coordinate is relevant.

Figure 10.1 Illustration: dimensionality reduction. (a) The original dataset does not vary much along the x_2 direction. (b) The data from (a) can be represented using the x_1-coordinate alone with nearly no loss.

z_{in}를 z_n의 i번째 원소라고 하고 B의 column을 b_i라고 하면, $\boldsymbol{z}_n = B^T\boldsymbol{x}_n$이므로 $z_{in} = \boldsymbol{b}_i^T\boldsymbol{x}_n$이 성립한다. 또한 $\tilde{\boldsymbol{x}}_n = \boldsymbol{b}_k\boldsymbol{b}_k^T\boldsymbol{x}_n$이 성립한다. 이에 따라 $\boldsymbol{z}_k$에 대한 variance는 다음과 같다. 이때 $\boldsymbol{S}$는 X에 대한 covariance matrix이다.

$$V_k = \mathbb{V}[\boldsymbol{z}_k] = \frac{1}{N}\sum_{n=1}^{N} z_{kn}^2 = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{b}_k^\top\boldsymbol{x}_n)^2 = \boldsymbol{b}_k^\top\left(\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n\boldsymbol{x}_n^\top\right)\boldsymbol{b}_k = \boldsymbol{b}_k^\top\boldsymbol{S}\boldsymbol{b}_k$$

이때 $\boldsymbol{b}_k$의 norm이 커지면 variance의 크기도 변할 수 있으므로 $||\boldsymbol{b}_k|| = 1$로 한다. 즉, 이는 다음과 같은 conditioned optimization problem이 된다.

$$\max_{\boldsymbol{b}_k} \boldsymbol{b}_k^\top \boldsymbol{S}\boldsymbol{b}_k$$
$$\text{subject to } \|\boldsymbol{b}_k\|^2 = 1.$$

lagrange multiplier method를 적용하면 다음과 같다.

$$L(\boldsymbol{b}_k, \lambda_k) = \boldsymbol{b}_k^\top \boldsymbol{S}\boldsymbol{b}_k + \lambda_k(1 - \boldsymbol{b}_k^\top \boldsymbol{b}_k)$$

이를 $\boldsymbol{b}_k$에 대해 partial differentiate해 0인 지점을 찾으면 $\boldsymbol{S}\boldsymbol{b}_k = \lambda_k \boldsymbol{b}_k$이 된다. **즉, variance를 최대로 하는 $\boldsymbol{b}_k$는 covarance matrix $\boldsymbol{S}$의 eigen vector이고, lagrange multiplier λ_k는 eigen value이다.** 또한, 이를 활용해 variance를 정리하면 $V_k = \boldsymbol{b}_k^\top \boldsymbol{S}\boldsymbol{b}_k = \lambda_k \boldsymbol{b}_k^T \boldsymbol{b}_k = \lambda_k$으로, **variance는 eigen value와 같다.**

이에 따라, low-dimensional representation의 variance를 최대화하기 위해서는 원본 데이터의 covariance matrix가 가지는 가장 큰 eigen value들에 대한 eigen vector로 basis를 구성해야 한다. 이때의 eigen vector를 **Principal Component(주성분)**라고 한다. 이때 가장 큰 eigen value를 가지는 것부터 first principal component가 된다.

M개의 앞쪽 principal component를 사용하는 경우 variance V_M과 손실되는 variance lost J_M은 다음과 같다.

$$V_M = \sum_{m=1}^{M} \lambda_m, \quad J_M = \sum_{j=M+1}^{D} \lambda_j = V_D - V_M$$

16.2.2. M-dimensional Subspace with Maximal Variance

covariance matrix $\boldsymbol{S}$로부터 m-1개의 앞쪽 principal componet를 추출했다고 하자. $\boldsymbol{S}$는 symmetric하므로 ONB를 얻을 수 있고, dimension이 $m-1$인 subspace $U \subset \mathbb{R}^D$를 생각할 수 있다. 데이터를 column으로 가지는 matrix $X = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N] \in \mathbb{R}^{D \times N}$에 대해 $m-1$개의 principal component를 고려해 압축되지 않은 정보만 남긴 matrix를 다음과 같이 얻을 수 있다. 이때 B_{m-1}는 ONB인 $\boldsymbol{b}_1, \cdots, \boldsymbol{b}_{m-1}$이 span하는 subspace로 projection하는 projection matrix이다($P^2 = P, P^T = P$가 성립한다.).

$$\hat{X} = X - \sum_{i=1}^{m-1} \boldsymbol{b}_i \boldsymbol{b}_i^\top X = X - B_{m-1}X$$

이 $\hat{X}$에 대한 covariance matrix $\hat{\boldsymbol{S}}$가 가지는 가장 큰 eigen value에 대응되는 eigen vector를 X에 대한 m번째 principal component로 할 수 있다. 이는 S와 $\hat{S}$의 eigen vector가 같다는 사실에 기반한다(eigen value 집합은 같지 않다.). 이는 다음과 같이 $\hat{S}\boldsymbol{b}_i$를 전개해 증명할 수 있다.

$$\begin{aligned}\hat{S} &= \frac{1}{N}(X - B_{m-1}X)(X - B_{m-1}X)^\top \\ &= S - SB_{m-1} - B_{m-1}S + B_{m-1}SB_{m-1}\end{aligned}$$

$i \geq m$인 경우, $B_{m-1}\boldsymbol{b}_i = 0$이므로 대입해 보면 $\hat{S}\boldsymbol{b}_i = S\boldsymbol{b}_i = \lambda_i \boldsymbol{b}_i$가 성립하므로 eigen value와 eigen vector가 모두 같다. $i < m$인 경우 B_{m-1}이 projection matrix이므로 $B_{m-1}\boldsymbol{b}_i = \boldsymbol{b}_i$가 성립해, 정리하면 $\hat{S}\boldsymbol{b}_i = 0$가 되어 eigen value는 서로 같지 않고($\hat{S}$는 eigen value가 0), eigen vector는 같다.

이에 따라 $m-1$번째까지 principal component를 구했으면, 그 다음 prinical component로는 $\hat{S}$의 가장 큰 eigen value를 가지는 eigen vector를 선택할 수 있다. 이런 선택의 과정을 반복하는 것으로 PCA를 수행할 수 있다.

참고로, SVD를 사용해 covariance matrix S의 eigen value와 eigen vector를 구할 수 있다. $X = U\Sigma V^\top$라 하면 covariacne matrix는 다음과 같다.

$$S = \frac{1}{N}XX^\top = \frac{1}{N}U\Sigma V^\top V\Sigma^\top U^\top = \frac{1}{N}U\Sigma\Sigma^\top U^\top$$

여기에 U의 column vector인 ONB를 대입해 보면 해당 column vector들이 eigen vector인 것을 알 수 있고, 이때 eigen value는 다음과 같다.

$$\lambda_d = \frac{\sigma_d^2}{N}$$

16.3. Projection Perspective

16.3.1. Projection Perspective

1. Projection Perspective
데이터셋 $X = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}, \boldsymbol{x}_i \in \mathbb{R}^D$와, $\mathbb{R}^D$의 ONB $\{\boldsymbol{b}_1, \cdots, \boldsymbol{b}_D\}$를 생각하자. 여기에서의 관심사는 X를 $\dim(U) = M < D$인 $\mathbb{R}^D$의 lower-dimenional subspace U로 projection하는 최적의 linear projection을 찾는 것이다. 이때의 U를 **Principal Subspace**라고 한다. principal subspace U에 존재하는 것은 $\boldsymbol{x}_n$을 projection한 dimension이 D인 vector $\tilde{\boldsymbol{x}}_n$이고, $\boldsymbol{z}_n$은 이 vector에 대한 dimension이 M인 coordinate이다(나머지 부분은 0). 즉, 다음과 같이 나타낼 수 있다. 이때 z_{mn}는 z_n의 m번째 원소이다.

$$\tilde{\boldsymbol{x}}_n = \sum_{m=1}^{M} z_{mn}\boldsymbol{b}_m = \boldsymbol{B}\boldsymbol{z}_n \in \mathbb{R}^D$$

즉, average reconstruction error인 $\boldsymbol{x}_n$와 $\tilde{\boldsymbol{x}}_n$ 사이의 distance를 최소화하는 $\tilde{\boldsymbol{x}}_n$와 principal component를 찾는 것을 목적으로 한다.

$$J_M = \frac{1}{N}\sum_{n=1}^{N}||\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n||^2$$

2. Finding Coordinate
$\mathbb{R}^D$의 ONB 중 M개로 구성한 matrix $B = [\boldsymbol{b}_1, \cdots, \boldsymbol{b}_M]$을 생각하자. BB^T는 projection matrix이고, $\tilde{\boldsymbol{x}}_n$는 다음과 같이 $\boldsymbol{x}_n$을 BB^T로 projection한 것이다.

$$\begin{aligned}\boldsymbol{z}_n &= B^T\boldsymbol{x}_n \\ \tilde{\boldsymbol{x}}_n &= B(B^TB)^{-1}B^T\boldsymbol{x}_n = BB^T\boldsymbol{x}_n\end{aligned}$$

이는 J_M를 $\boldsymbol{z}_n$에 대해 partial differentiate하는 것으로 증명할 수 있다. distance의 최솟값은 derivate가 0이 되는 지점을 찾으면 된다.

$$\begin{aligned}J_M &= \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}_n - \boldsymbol{B}\boldsymbol{z}_n)^\top(\boldsymbol{x}_n - \boldsymbol{B}\boldsymbol{z}_n) = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}_n^\top\boldsymbol{x}_n - 2\boldsymbol{x}_n^\top\boldsymbol{B}\boldsymbol{z}_n + \boldsymbol{z}_n^\top\boldsymbol{B}^\top\boldsymbol{B}\boldsymbol{z}_n) \\ \frac{\partial J_M}{\partial \boldsymbol{z}_n} &= \frac{1}{N}(-2\boldsymbol{B}^\top\boldsymbol{x}_n + 2\boldsymbol{B}^\top\boldsymbol{B}\boldsymbol{z}_n) = 0 \\ \boldsymbol{z}_n &= (\boldsymbol{B}^\top\boldsymbol{B})^{-1}\boldsymbol{B}^\top\boldsymbol{x}_n, \quad \tilde{\boldsymbol{x}}_n = \boldsymbol{B}(\boldsymbol{B}^\top\boldsymbol{B})^{-1}\boldsymbol{B}^\top\boldsymbol{x}_n = \boldsymbol{B}\boldsymbol{B}^\top\boldsymbol{x}_n\end{aligned}$$

3. Finding Basis of Principal Subspace
최적의 linear projection을 찾기 위해서는 principal subspace의 ONB를 우선 찾아야 한다. 이를 위해 앞서 다룬 결과를 활용해 J_M을 정리해보자. ONB 중 B에 사용한 D개의 basis를 제외한 나머지 $D - M$개의 basis

$\boldsymbol{b}_{M+1}, \ldots, \boldsymbol{b}_D$를 column으로 하는 matrix $\boldsymbol{C} \in \mathbb{R}^{D \times (D-M)}$를 생각하자. 원본 데이터 $\boldsymbol{x}_n$과 projection으로 dimensionality를 줄인 데이터 $\tilde{\boldsymbol{x}}_n$ 간의 차이는 다음과 같이 principal subspace의 orthogonal complement(직교여공간)의 vector이다.

$$\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n = \boldsymbol{C}\boldsymbol{C}^\top \boldsymbol{x}_n$$

정리하면 norm은 다음과 같다.

$$\|\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n\|^2 = (\boldsymbol{C}\boldsymbol{C}^\top \boldsymbol{x}_n)^\top (\boldsymbol{C}\boldsymbol{C}^\top \boldsymbol{x}_n) = \boldsymbol{x}_n^\top \boldsymbol{C}\boldsymbol{C}^\top \boldsymbol{x}_n$$

이를 활용해 J_M을 정리할 수 있다. 이때 'trace trick'과 $\boldsymbol{S}\boldsymbol{b}_j = \lambda_j \boldsymbol{b}_j$임을 활용한다.

$$\begin{aligned} J_M &= \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n^\top \boldsymbol{C}\boldsymbol{C}^\top \boldsymbol{x}_n = \frac{1}{N}\sum_{n=1}^{N} \mathrm{tr}(\boldsymbol{C}^\top \boldsymbol{x}_n \boldsymbol{x}_n^\top \boldsymbol{C}) = \mathrm{tr}(\boldsymbol{C}^\top \boldsymbol{S}\boldsymbol{C}) \\ &= \sum_{j=M+1}^{D} \boldsymbol{b}_j^\top \boldsymbol{S}\boldsymbol{b}_j = \sum_{j=M+1}^{D} \lambda_j \end{aligned}$$

즉, **원본 데이터의 covariance matrix가 가지는 eigen value 중 가장 값이 큰 것들의 eigen vector들로 ONB를 구성해야 J_M을 최소화할 수 있다.** 이는 maximum variance perspective에서의 결론과 완전히 동일하다.

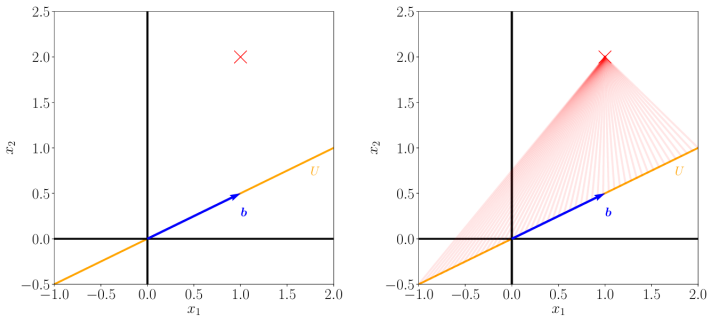


Figure 10.7 Simplified projection setting. (a) A vector $\boldsymbol{x} \in \mathbb{R}^2$ (red cross) shall be projected onto a one-dimensional subspace $U \subseteq \mathbb{R}^2$ spanned by $\boldsymbol{b}$. (b) shows the difference vectors between $\boldsymbol{x}$ and some candidates $\tilde{\boldsymbol{x}}$.

(a) Setting.

(b) Differences $\boldsymbol{x} - \tilde{\boldsymbol{x}}_i$ for 50 different $\tilde{\boldsymbol{x}}_i$ are shown by the red lines.

"'Trace Trick'에 의하면 스칼라 값(1x1 행렬)은 자기 자신의 대각합(Trace, tr)과 같으며, Trace 안에서는 행렬 곱의 순서를 순환시킬 수 있습니다. ($\mathrm{tr}(\boldsymbol{A}\boldsymbol{B}\boldsymbol{C}) = \mathrm{tr}(\boldsymbol{C}\boldsymbol{A}\boldsymbol{B})$)."

16.3.2. Practical Approaches on Calculating Eigen Value/Vectors

1. Power Iteration
eigen value/vector를 얻기 위해 characteristic polynomial을 풀어서 계산하는 것은 해당 polynomial의 degree가 5 이상이면 구하는 것이 불가능해진다. 또한 주로 관심 있는 부분은 그 값이 가장 큰 eigen value들이므로, 이렇게 모든 eigen value를 구하는 것은 비효율적이다. 이에 따라 iterative process를 통해 가장 값이 큰 eigen value만 구하는 기법들이 존재하는데, 그 중 **Power Iteration**에서는 matrix S에 대해 가장 큰 eigen value 하나만을 구한다. null space의 vector가 아닌 vector x_0로부터 시작해, 다음과 같은 수식을 반복 계산하다보면 가장 큰 eigen value를 가지는 eigen vector에 수렴한다고 한다.

$$\boldsymbol{x}_{k+1} = \frac{S\boldsymbol{x}_k}{\|S\boldsymbol{x}_k\|}, \quad k = 0, 1, \ldots$$

2. PCA in High-dimensions
실제로 데이터의 dimension D가 높은 경우, $D \times D$ matrix인 convariance matrix에 대한 eigen value/vector

를 구하는 것의 overhead가 크다. 이에 따라 데이터의 개수 N보다 dimension D가 훨씬 큰 경우에는, 다음과 같이 covariance matrix를 변형하고 $\boldsymbol{c}_m = \boldsymbol{X}^\top \boldsymbol{b}_m$으로 두면 $N \times N$ matrix에 대해 eigen value/vector를 구할 수 있다.

$$S\boldsymbol{b}_m = \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{b}_m = \lambda_m \boldsymbol{b}_m$$
$$\frac{1}{N}\boldsymbol{X}^\top \boldsymbol{X}(\boldsymbol{X}^\top \boldsymbol{b}_m) = \lambda_m(\boldsymbol{X}^\top \boldsymbol{b}_m), \quad \frac{1}{N}\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{c}_m = \lambda_m \boldsymbol{c}_m$$

교재에는 latent variable perspective도 존재하지만 정리하지 않았다.

17. Density Estimation with Gaussian Mixture Models

데이터의 값을 그 자체로 해당 데이터에 대한 representation으로 사용하는 것은 straightforward하지만, 데이터셋이 커지면 overhead가 커지고, 데이터의 특성을 분석하기에 적절하지 않다. 본 장에서는 데이터를 gaussian distribution과 같은 probability distribution의 density로 표현하는 방법을 알아본다. 이렇게 데이터를 특정 density로 표현하는 것을 **Density Estimation**이라고 한다.

17.1. Gaussian Mixture Models

17.1.1. Gaussian Mixture Models

Mixture Model은 다음 수식과 같이 여러 distribution으로 구성된 모델이다. 이때 p_k는 gaussian과 같은 basic distribution이고 각 distribution을 **Mixture Component**라고 한다. π_k는 Mixture Weight이다.

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k p_k(\boldsymbol{x})$$
$$0 \leqslant \pi_k \leqslant 1, \quad \sum_{k=1}^{K} \pi_k = 1$$

basic distribution이 gaussain인 mixture model을 **Gaussian Mixture Model(GMM)**이라고 하고, 다음과 같이 정의된다. 이때 $\boldsymbol{\theta}$는 모델이 가지는 모든 parameter의 집합으로, $\boldsymbol{\theta} = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k : k = 1, \ldots, K\}$과 같다.

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$
$$0 \leqslant \pi_k \leqslant 1, \quad \sum_{k=1}^{K} \pi_k = 1$$

GMM은 그 parameter가 가지는 likelihood를 최대로 하는 것으로 학습시킬 수 있고, 이 경우 앞서 linear regression과 PCA에서 한 것과 같이 closed form solution을 구할 수 없으므로 EM algorithm과 같은 iterative operation을 활용한다.

다음과 같이 GMM은 단일 gaussian distribution비 비해 더 높은 표현력을 가진다.

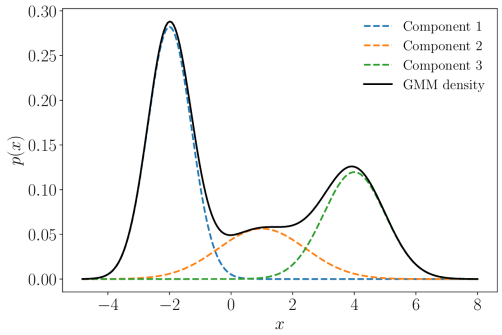


Figure 11.2 Gaussian mixture model. The Gaussian mixture distribution (black) is composed of a convex combination of Gaussian distributions and is more expressive than any individual component. Dashed lines represent the weighted Gaussian components.

mixture model의 정의에서 사용된 것처럼, 이렇게 합이 1이고 각 가중치가 음수가 아닌 결합을 Convex Combination 이라고 한다.

17.2. Parameter Learning with MLE

17.2.1. Responsibity

데이터셋 $X = \{\boldsymbol{x}_1 \cdots, \boldsymbol{x}_N\}$이 주어졌을 때, 이에 대한 good approximation인 distribution을 GMM을 활용해 찾는 것이 목적이다. 하지만 이를 위해 log likelihood를 작성하면 다음과 같은데, 단일 gaussian distribution의 경우 differentiate해서 0인 지점을 찾으면 되지만, 이렇게 log 내부에 덧셈이 존재하면 closed form solution을 구할 수 없다(실제로 differentiate해 보면 수식이 정리되지 않는다.).

$$\mathcal{L} = \sum_{n=1}^{N} \log p(\boldsymbol{x}_n|\boldsymbol{\theta}) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

이에 따라 closed form solution을 바로 구하는 대신, iterative approach인 EM algorithm을 사용해 최적의 parameter를 찾는다. 즉, 다른 parameter들은 고정해 두고 한 번에 하나의 parameter만 update한다.

이를 위해 K개의 각 mixture component에 대한 **Responsibility**를 다음과 같이 정의한다. 데이터셋의 어떤 데이터 $\boldsymbol{x}_n$에 대해 $\boldsymbol{r}_n = [r_{n1}, \cdots, r_{nK}]$를 생각할 수 있고, $\boldsymbol{r}_n$은 probability vector이다. 즉, responsibility는 총합이 1이고 0보다 크거나 같은 값이므로 probability이고, $\boldsymbol{x}_n$이 k번째 mixture component에 의해 생성되었을 probability를 나타낸다.

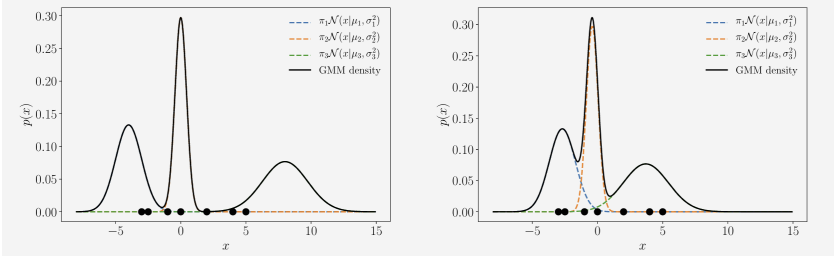
$$r_{nk} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

responsibility를 활용해 mean, covariance, mixture weight 각각을 update할 수 있다. model parameter를 update할 때 수식에 responsibility가 모두 존재하는데, 이에 따라 그냥 differentiate했을 때는 closed form solution을 얻을 수 없었다. 쉽게 말해, $\mu_k = f(\mu_k, \dots)$와 같이 구하고자 하는 변수가 좌변과 우변 양쪽에 모두 나타나고, 이를 정리해서 한쪽 변에만 존재하도록 할 수 없었다. 해당 부분은 responsibility로 정의하여 상수화하는 것으로(해당 파라미터들은 고정) closed form solution을 갖도록 하는 것이 여기에서의 parameter update의 핵심이다.

17.2.2. Parameter Learning with MLE

1. Updating the Means
mean에 대한 update는 다음 수식으로 수행할 수 있다. 이때 N_k는 k번째 mixture componet가 전체 데이터셋에 대해 가지는 responsiblity의 총합이다. 즉, 이는 단순 평균값을 계산하는 기존의 방식과는 달리 responsibility 기반으로 가중 평균을 수행하고, 각 mixture component가 가지는 responsibility(likelihood)를 고려해서 기존의 mean을 이동시킨다.

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{\sum_{n=1}^{N} r_{nk}\boldsymbol{x}_n}{\sum_{n=1}^{N} r_{nk}} = \frac{1}{N_k}\sum_{n=1}^{N} r_{nk}\boldsymbol{x}_n$$



(a) GMM density and individual components prior to updating the mean values.

(b) GMM density and individual components after updating the mean values.

이는 다음과 같이 증명 가능하다. 우선 전체 distribution에 대해 partial differnetiate하고, 이후 log를 씌운 수식을 확인하자.

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k} = \sum_{n=1}^{N} \frac{\partial \log p(\boldsymbol{x}_n|\boldsymbol{\theta})}{\partial \boldsymbol{\mu}_k} = \sum_{n=1}^{N} \frac{1}{p(\boldsymbol{x}_n|\boldsymbol{\theta})} \frac{\partial p(\boldsymbol{x}_n|\boldsymbol{\theta})}{\partial \boldsymbol{\mu}_k}$$

$$\begin{aligned}\frac{\partial p(\boldsymbol{x}_n|\boldsymbol{\theta})}{\partial \boldsymbol{\mu}_k} &= \sum_{j=1}^{K} \pi_j \frac{\partial \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\partial \boldsymbol{\mu}_k} = \pi_k \frac{\partial \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\partial \boldsymbol{\mu}_k} \\ &= \pi_k (\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k} &= \sum_{n=1}^{N} \frac{\partial \log p(\boldsymbol{x}_n|\boldsymbol{\theta})}{\partial \boldsymbol{\mu}_k} = \sum_{n=1}^{N} \frac{1}{p(\boldsymbol{x}_n|\boldsymbol{\theta})} \frac{\partial p(\boldsymbol{x}_n|\boldsymbol{\theta})}{\partial \boldsymbol{\mu}_k} \\ &= \sum_{n=1}^{N} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1} \underbrace{\frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}}_{=r_{nk}} \\ &= \sum_{n=1}^{N} r_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}.\end{aligned}$$

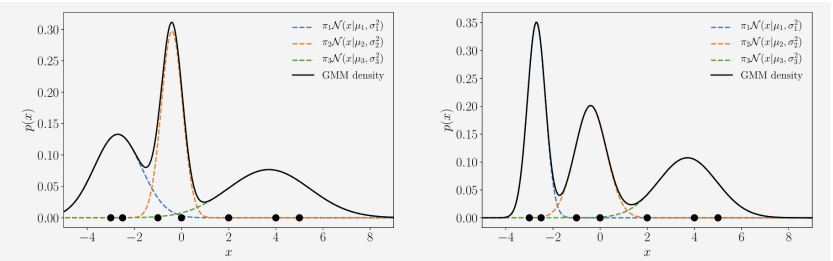
이와 같이 responsibility로 상수화하면 해당 derivate이 0이 되는 지점을 closed form solution으로 구할 수 있다. 즉, 다음과 같이 정리된다.

$$\sum_{n=1}^{N} r_{nk}\boldsymbol{x}_n = \sum_{n=1}^{N} r_{nk}\boldsymbol{\mu}_k^{\text{new}} \iff \boldsymbol{\mu}_k^{\text{new}} = \frac{\sum_{n=1}^{N} r_{nk}\boldsymbol{x}_n}{\sum_{n=1}^{N} r_{nk}} = \frac{1}{N_k}\sum_{n=1}^{N} r_{nk}\boldsymbol{x}_n$$

2. Updating the Covariances

covariance에 대한 update는 다음 수식으로 수행할 수 있다. 여기에서도 마찬가지로 이는 단순 평균값을 계산하는 기존의 방식과는 달리 responsibility 기반으로 가중 평균을 수행하고, 각 mixture component가 가지는 responsibility(likelihood)를 고려해서 기존의 covariance를 이동시킨다. 이때 r_{nk}와 N_k는 앞서 정의한 것과 같다.

$$\Sigma_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top$$



(a) GMM density and individual components prior to updating the variances.

(b) GMM density and individual components after updating the variances.

이는 다음과 같이 증명 가능하다. mean에서와 동일한 방식이다. 우선 differentiate하면 다음과 같다.

$$\frac{\partial L}{\partial \Sigma_k} = \sum_{n=1}^{N} \frac{\partial \log p(\boldsymbol{x}_n|\theta)}{\partial \Sigma_k} = \sum_{n=1}^{N} \frac{1}{p(\boldsymbol{x}_n|\theta)} \frac{\partial p(\boldsymbol{x}_n|\theta)}{\partial \Sigma_k}$$

이후 product differentiation을 하고, 기존 수식에 대입한다.

$$\begin{aligned}
\frac{\partial p(\boldsymbol{x}_n|\theta)}{\partial \Sigma_k} &= \frac{\partial}{\partial \Sigma_k} \left[\pi_k (2\pi)^{-\frac{D}{2}} \det(\Sigma_k)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} (\boldsymbol{x}_n - \boldsymbol{\mu}_k) \right) \right] \\
&= \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \Sigma_k) \cdot \left(-\frac{1}{2} \right) \left(\Sigma_k^{-1} - \Sigma_k^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} \right)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial L}{\partial \Sigma_k} &= \sum_{n=1}^{N} \frac{1}{p(\boldsymbol{x}_n|\theta)} \frac{\partial p(\boldsymbol{x}_n|\theta)}{\partial \Sigma_k} \\
&= \sum_{n=1}^{N} \underbrace{\frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \Sigma_j)}}_{=r_{nk}} \cdot \left(-\frac{1}{2} \right) \left(\Sigma_k^{-1} - \Sigma_k^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} \right) \\
&= -\frac{1}{2} \sum_{n=1}^{N} r_{nk} \left(\Sigma_k^{-1} - \Sigma_k^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} \right) \\
&= -\frac{1}{2} \Sigma_k^{-1} \underbrace{\sum_{n=1}^{N} r_{nk}}_{=N_k} + \frac{1}{2} \Sigma_k^{-1} \left(\sum_{n=1}^{N} r_{nk}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \right) \Sigma_k^{-1}
\end{aligned}$$

이후 derivate이 0이 되는 지점을 closed form solution으로 구할 수 있다.

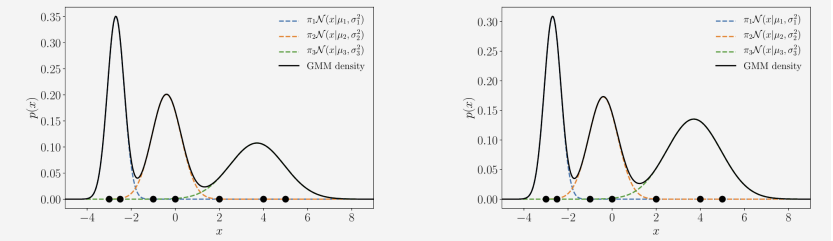
$$\Sigma_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top$$

3. Updating the Mixture Weights

mixture weight에 대한 update는 다음 수식으로 수행할 수 있다. 이때 N은 데이터의 개수이고, N_k는 앞서

정의한 것과 같다. N은 N_k의 총합과 같으므로, 이 수식은 전체 mixture component들에 대한 각 mixture component가 가지는 responsibility의 비율을 나타낸 것과 같다.

$$\pi_k^{\text{new}} = \frac{N_k}{N}$$



(a) GMM density and individual components prior to updating the mixture weights.

(b) GMM density and individual components after updating the mixture weights.

이는 다음과 같이 증명할 수 있다. responsiblity의 총합이 1이라는 constraint가 존재하므로, lagrange multiplier method를 통해 수식을 정리하면 다음과 같다.

$$\begin{aligned}\mathcal{L} &= L + \lambda\left(\sum_{k=1}^{K}\pi_k - 1\right) \\ &= \sum_{n=1}^{N}\log\sum_{k=1}^{K}\pi_k\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k) + \lambda\left(\sum_{k=1}^{K}\pi_k - 1\right)\end{aligned}$$

이를 mixture weight에 대해 differentiate하면 다음과 같다.

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\pi_k} &= \sum_{n=1}^{N}\frac{\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K}\pi_j\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)} + \lambda \\ &= \frac{1}{\pi_k}\sum_{n=1}^{N}\frac{\pi_k\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K}\pi_j\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)} + \lambda = \frac{N_k}{\pi_k} + \lambda\end{aligned}$$

이후 derivate을 0으로 두고 정리하면 다음과 같다.

$$\pi_k = -\frac{N_k}{\lambda}$$

이를 활용해 다음과 같이 정리할 수 있고, 결과적으로 mixture weight를 update할 수 있다. 이때 responsibility를 모두 더하면 1이므로 $\sum_{k=1}^{K}N_k = N$이 성립한다.

$$\begin{aligned}&\sum_{k=1}^{K}\pi_k = 1 \iff -\sum_{k=1}^{K}\frac{N_k}{\lambda} = 1 \iff -\frac{N}{\lambda} = 1 \\ &\pi_k^{\text{new}} = \frac{N_k}{N}\end{aligned}$$

17.2.3. EM algorithm

EM algorithm은 mixture model, latent variable model 등에서의 MLE/MAP를 통한 parameter learning

을 위한 iterative scheme이다. 이는 다음과 같은 과정을 반복 적용하는 알고리즘으로, 매 step에서 likelihood가 커진다고 한다. 해당 값이 수렴했는지는 직접 likelihood나 parameter를 확인해 판단한다.

1. $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k$를 초기화한다. (초기 값을 구한다.)
2. E-step: 현재의 $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k$와 전체 데이터셋을 활용해 responsibility를 구한다.

$$r_{nk} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

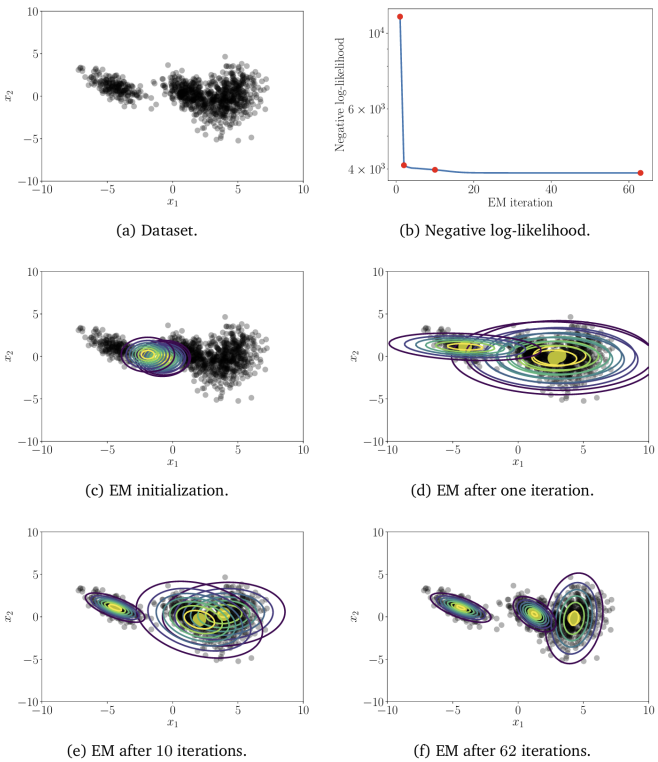
3. M-step: 현재의 responsibility를 활용해 $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k$를 update한다.

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k^{\text{new}})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{\text{new}})^\top$$

$$\pi_k = \frac{N_k}{N}$$

responsiblity에 해당되는 수식 안에 $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k$가 모두 존재해서 closed form solution을 얻을 수 없었다. EM algorithm을 적용했을 때 closed form solution을 얻을 수 있었던 것은 $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k$를 고정해서 상수 취급하고, iterative하게 접근하기 때문이다.



(a) Dataset. (b) Negative log-likelihood. (c) EM initialization. (d) EM after one iteration. (e) EM after 10 iterations. (f) EM after 62 iterations.

Figure 11.9 Illustration of the EM algorithm for fitting a Gaussian mixture model with three components to a two-dimensional dataset. (a) Dataset; (b) negative log-likelihood (lower is better) as a function of the EM iterations. The red dots indicate the iterations for which the mixture components of the corresponding GMM fits are shown in (c) through (f). The yellow discs indicate the means of the Gaussian mixture components. Figure 11.10(a) shows the final GMM fit.

교재에는 latent variable perspective도 존재하지만 정리하지 않았다.

18. Classification with SVM

본 장에서는 SVM을 활용한 binary classification 문제에 대해 알아본다. 즉, classification error를 최소화하는 predictor $f : \mathbb{R}^D \to \{-1, +1\}$를 찾는 것을 그 목적으로 한다. 이때의 $\{-1, +1\}$의 각 원소를 Negative Class와 Positive Class라고 한다.

18.1. Classification with SVM

18.1.1. Classification with SVM

1. SVM

SVM(Support Vector Machine)은 데이터에 대한 classification/regression을 수행하는 데에 사용되는 supervised learning 알고리즘으로, space에서 서로 다른 class를 잘 분리할 수 있는 최적의 hyperplane을 찾는 것을 그 목적으로 한다. SVM에서는 기하학적으로 접근하여 function predictor를 정의하고, 이를 ERM으로 최적화하는 것으로 binary classification을 풀 수 있다. 즉, binary classification에서 두 class를 잘 separate 하는 hyperplane을 찾는다.

continous optimization에서 정리한 것처럼 hyperplane은 $f : \mathbb{R}^D \to \mathbb{R}$, $f(x) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = \boldsymbol{w}^T \boldsymbol{x} + b$에 대해서 다음과 같은 affine space로 정의된다.

$$\{\boldsymbol{x} \in \mathbb{R}^D \mid f(\boldsymbol{x}) = 0\}$$

2. Positive/Negative Side

만약 임의의 vector $\boldsymbol{v}_a \in \mathbb{R}^D$가 주어졌을 때, 이를 hyperplane(affine space) 위의 점 $\boldsymbol{v}'_a$와, scaling을 수행하는 scalar α에 대해 $\boldsymbol{v}_a = \boldsymbol{v}'_a + \alpha \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$와 같이 orthogonal한 두 vector의 합으로 나타낼 수 있다.

임의의 vector $\boldsymbol{x} \in \mathbb{R}^D$에 대해 $f(\boldsymbol{x})$의 부호에 따라 해당 vector가 positive side에 있는지, negative side에 있는지가 결정된다. $\boldsymbol{v}_a$를 f에 넣어 정리하면 $f(\boldsymbol{v_a}) = \alpha \|\boldsymbol{w}\|$가 되고, α에 의해 $f(\boldsymbol{v_a})$의 부호가 결정된다. 즉, $f(\boldsymbol{x})$의 부호로 hyperplane에 대해 어떤 side에 존재하는지를 판단할 수 있다.

총 N개의 $\boldsymbol{x}_n \in \mathbb{R}^D$인 데이터와 그 label $y_n \in \{-1, +1\}$로 구성된 데이터셋 $\{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_N, y_N)\}$을 생각하자. 앞서 다룬 결론에 따라 각 데이터에 대해 다음이 성립하도록 classifier를 학습시킨다.

$$\begin{aligned} \langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b \geq 0 \quad &\text{when} \quad y_n = +1 \\ \langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b < 0 \quad &\text{when} \quad y_n = -1 \end{aligned}$$

하나의 수식으로 정리하면 다음과 같다.

$$y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \geq 0$$

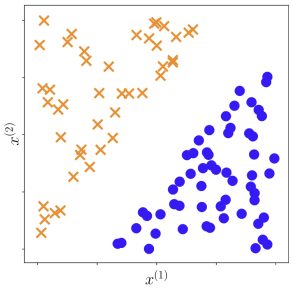
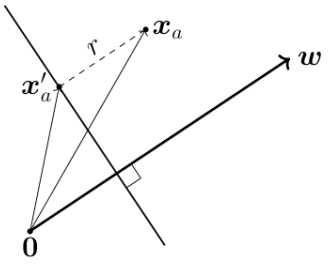


Figure 12.1 Example 2D data, illustrating the intuition of data where we can find a linear classifier that separates orange crosses from blue discs.

18.2. Primal SVM

18.2.1. Hard Margin SVM

1. Margin
앞서 다룬 것처럼 vector $\boldsymbol{v}_a \in \mathbb{R}^D$를 $\boldsymbol{v}_a = \boldsymbol{v}'_a + \alpha \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$꼴로 나타낼 수 있다. 이때 $\boldsymbol{x}_a$와 hyperplane 간의 distance를 생각할 수 있고, hyperplane과 가장 가까운 데이터 간의 distance를 **Margin**이라고 하며, r로 표기한다. margin은 distance이므로 $r > 0$이다. 참고로, 다음 그림에서 $\boldsymbol{x}'_a$을 vector로 보면 hyperplane 위에 있지 않은 것이라고 착각할 수 있는데, hyperplane은 affine space이므로 점으로 이해해야 한다.



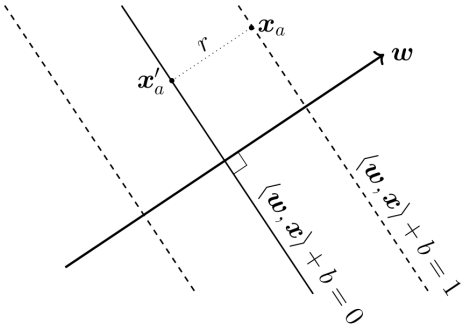
2. Scaling for Calculating Distance
hyperplane $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0$에 대해, scalar $c > 0$를 곱한 $\langle c\boldsymbol{w}, \boldsymbol{x} \rangle + bc = 0$ 또한 동일한 hyperplane을 나타낸다. 하지만 기존의 수식은 $f(\boldsymbol{x}_a) = \alpha||\boldsymbol{w}||$인데, c를 곱한 이후에는 $f(\boldsymbol{x}_a) = \alpha||c\boldsymbol{w}|| = \alpha c||\boldsymbol{w}||$으로, 실제로 distance가 달라지게 된다. 이에 따라 hyperplane과의 distance를 계산할 때에는 어떤 크기의 $\boldsymbol{w}, b$에 대해 계산할 것인지 scaling을 해줘야 한다.

$\|\boldsymbol{w}\| = 1$로 constraint를 지정해 scale을 고정할 수 있다. 하지만 이 경우 해당 수식은 $\boldsymbol{w}$에 대한 이차식이므로, lagrange multiplier method를 적용하고 differentiate할 때 계산이 어려워지는 문제가 발생한다. 이에 따라 다음과 같이 margin을 지정할 수 있다. $\|\boldsymbol{w}\| = 1$로 두는 contraint를 제거하고, $\boldsymbol{w}' = \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$으로 기존의 $\boldsymbol{w}$를 대체한 뒤, 수식을 정리해 보면 이렇게 margin을 지정하는 것은 $\|\boldsymbol{w}\| = 1$로 두는 것과 완전히 동일하다.

$$\langle \boldsymbol{w}, \boldsymbol{x}_a \rangle + b = 1$$

$\boldsymbol{v}_a = \boldsymbol{v}'_a + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$임을 이용해서 정리하면 $r\|\boldsymbol{w}\| = 1, r = \frac{1}{\|\boldsymbol{w}\|}$가 된다. 즉, 이 constraint는 다음 그림과 같이 margin을 $\frac{1}{\|\boldsymbol{w}\|}$으로 하는 것과 같다.

Figure 12.5
Derivation of the margin: $r = \frac{1}{\|\boldsymbol{w}\|}$.



3. Hard Margin SVM
여러 candidiate hyperplane 중에 데이터(positive/negative example 모두 포함)와의 margin을 최대화하는 hyperplane이 가장 좋은 hyperplane이라고 생각할 수 있다. **Hard Margin SVM**은 다음 수식과 같이 데이터 셋이 선형적으로 분리가 가능하다는 것(linearly separable)을 가정하고, scale을 고려해서 다음과 같은 training objective를 같는다. 즉, 데이터가 hyperplane에 의해 완전히 분리될 수 있을 때, 데이터에 맞는 범위 내에서 margin을 최대화한다.

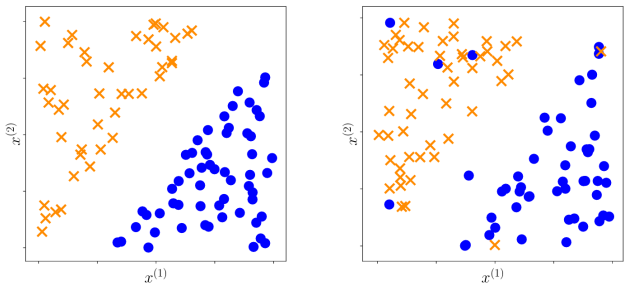
$$\max_{\boldsymbol{w},b} \quad \underbrace{\frac{1}{\|\boldsymbol{w}\|}}_{\text{margin}}$$
$$\text{subject to} \quad \underbrace{y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \geqslant 1}_{\text{data fitting}}$$

또는 이를 minimization problem으로 다음과 같이 수정할 수 있다. 이때 $\frac{1}{2}$는 계산적 편의를 위한 것으로, 수렴하는 값에는 영향이 없다.

$$\min_{\boldsymbol{w},b} \quad \underbrace{\frac{1}{2}\|\boldsymbol{w}\|^2}_{\text{margin}}$$
$$\text{subject to} \quad \underbrace{y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \geqslant 1}_{\text{data fitting}}$$

margin이 크다면 class complexity가 낮고, 학습이 용이할 수 있다.

데이터가 선형적으로 분리되지 않는 경우는 아래의 soft margin SVM에서 살펴보자.



(a) Linearly separable data, with a large margin

(b) Non-linearly separable data

Figure 12.6 (a) Linearly separable and (b) non-linearly separable data.

18.2.2. Soft Margin SVM

1. Geometric View

Soft Margin SVM은 데이터가 선형적으로 분리되지 않는 상황까지 classification을 일반화하며, hard margin SVM과 달리 classification error(margin 내부나 wrong side에 있는 경우)를 허용한다. 이를 위해 각 데이터 $(\boldsymbol{x}_n, y_n)$에 대응되는 non-negative scalar인 Slack Variable ξ을 사용한다.

이때 $C > 0$인 C는 Regularization Parameter로, margin과 전체 slack variable 간의 tradeoff를 조정한다. 또한 margin term $\|\boldsymbol{w}\|^2$는 Regularizer로, regularization을 적용하는 효과가 있다. C가 크면 regularization이 덜 적용되어 margin을 넘어가는 것들도 많이 용인하게 되고, C가 작으면 많이 적용되어 더 strict하게 최적화된다. 즉, margin maximization은 regularization으로 이해할 수 있다.

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N}\xi_n \\ \text{subject to} \quad & y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n \rangle + b) \geqslant 1 - \xi_n \\ & \xi_n \geqslant 0 \end{aligned}$$

즉, 다음 그림과 같이 ξ가 존재해서 classification error를 반영하고, 이를 최소화하도록 추가로 minimization problem에 항으로 추가한다.

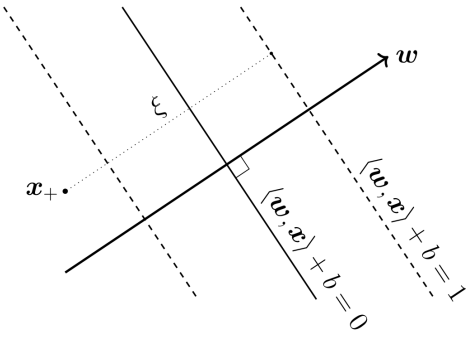


Figure 12.7 Soft margin SVM allows examples to be within the margin or on the wrong side of the hyperplane. The slack variable ξ measures the distance of a positive example $\boldsymbol{x}_+$ to the positive margin hyperplane $\langle \boldsymbol{w}, \boldsymbol{x}\rangle + b = 1$ when $\boldsymbol{x}_+$ is on the wrong side.

2. Loss Function View
앞서 다룬 것처럼 hyperplane은 다음과 같이 정의된다. loss function view에서는 다음과 같은 function에 대해 loss function을 정의하고, ERM을 적용해 최적의 function을 찾는다.

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, x\rangle + b$$

loss function으로는 Hinge Loss를 사용한다. 이는 다음과 같이 정의된다. 즉, $\langle \boldsymbol{w}, x\rangle + b$ 값이 margin에 해당하는 1을 넘어가는 순간부터 loss가 0이 아니게 되고, hyperplane을 넘어서 반대쪽 side로 갈수록 loss가 더 커진다.

$$\ell(t) = \max\{0, 1-t\} \quad \text{where} \quad t = yf(\boldsymbol{x}) = y(\langle \boldsymbol{w}, \boldsymbol{x}\rangle + b)$$

이를 더 알아보기 쉽게 표기하면 다음과 같다.

$$\ell(t) = \begin{cases} 0 & \text{if } t \geqslant 1 \\ 1-t & \text{if } t < 1 \end{cases}$$

해당 loss function을 사용해 다음과 같은 unconstrained optimization을 풀 수 있다. gradient descent 등으로 접근 가능하다. 이때 $\frac{1}{2}\|\boldsymbol{w}\|^2$은 regularization term으로 추가되었다. geometric view와 비교했을 때 사실상 동일한 의미의 수식인 것을 알 수 있다.

$$\min_{\boldsymbol{w},b} \underbrace{\frac{1}{2}\|\boldsymbol{w}\|^2}_{\text{regularizer}} + C \underbrace{\sum_{n=1}^{N} \max\{0, 1-y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n\rangle + b)\}}_{\text{error term}}$$

18.3. Dual SVM

앞에서 다룬 $\boldsymbol{w}$와 b를 사용해 정의한 SVM을 Primal SVM이라고 하는데, 이 경우 $\boldsymbol{w}$는 데이터 $\boldsymbol{x}_n$과 shape이 같고, 이에 따라 feature의 개수가 늘어나면 parameter의 개수 또한 선형적으로 늘어난다. Dual SVM은 feature의 개수에 비례하도록 하는 대신, 데이터셋의 크기에 비례하도록 한다. 이는 high dimension 데이터에 PCA를 적용할 때 다뤘던 것처럼, 데이터셋의 크기보다 feature의 개수가 훨씬 많은 경우에 유용하다. 또한 뒤에서 설명할 kernel을 적용할 수 있게 된다는 장점도 있다.

교재에는 convex hull view도 있지만 정리하지 않았다.

18.3.1. Dual Optimization Problem of SVM

다음과 같이 soft margin SVM에 대한 optimization problem을 정의했었다.

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N}\xi_n \\ \text{subject to} \quad & y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n\rangle + b) \geqslant 1 - \xi_n \\ & \xi_n \geqslant 0 \end{aligned}$$

lagrange multiplier method를 사용하면 lagrangian은 다음과 같다.

$$\mathfrak{L}(\boldsymbol{w}, b, \xi, \alpha, \gamma) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N}\alpha_n(y_n(\langle \boldsymbol{w}, \boldsymbol{x}_n\rangle + b) - 1 + \xi_n) - \sum_{n=1}^{N}\gamma_n\xi_n$$

이를 $\boldsymbol{w}, b, \xi_n$ 각각에 대해 partial differentiate하면 다음과 같다. 최적의 지점을 찾기 위해 각 partial differentiate 값을 0으로 둔다.

$$\begin{aligned} \frac{\partial \mathfrak{L}}{\partial \boldsymbol{w}} &= \boldsymbol{w}^\top - \sum_{n=1}^{N}\alpha_n y_n \boldsymbol{x}_n^\top = \mathbf{0} \\ \frac{\partial \mathfrak{L}}{\partial b} &= -\sum_{n=1}^{N}\alpha_n y_n = 0 \\ \frac{\partial \mathfrak{L}}{\partial \xi_n} &= C - \alpha_n - \gamma_n = 0 \end{aligned}$$

이에 따라 $\boldsymbol{w}$는 다음과 같이 정리할 수 있고, 이는 최적의 $\boldsymbol{w}$가 데이터에 대한 linear combination으로 구성된다는 것을 의미한다. 이때 $\alpha_n = 0$에 해당하는 데이터들은 최적의 $\boldsymbol{w}$(hyperplane)를 "support"하지 않는데, $\alpha_n > 0$에 해당하는 데이터들은 "support"하고, 이 vector들을 Support Vector라고 한다.

$$\boldsymbol{w} = \sum_{n=1}^{N}\alpha_n y_n \boldsymbol{x}_n$$

또한 정리하면 다음과 같이 soft margin SVM에 대한 dual optimization problem을 구성할 수 있다. 이렇게 데이터셋의 크기만큼의 parameter들을 구하는 문제로 변환할 수 있다. (이에 대한 유도는 정리하지 않았다.) 이 과정을 통해 최적의 Dual Parameter $\alpha_1, \cdots, \alpha_N$을 구하면 $\boldsymbol{w}, b$도 구할 수 있다.

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j\rangle - \sum_{i=1}^{N}\alpha_i \\ \text{subject to} \quad & \sum_{i=1}^{N} y_i \alpha_i = 0 \\ & 0 \leqslant \alpha_i \leqslant C \quad \text{for all} \quad i = 1, \ldots, N. \end{aligned}$$

교재에는 kernel과 numerical solution에 대한 내용도 있지만 정리하지 않았다.