심층학습 2장

송선영

2.6 특별한 종류의 행렬과 벡터

- 대각행렬 : 0이 아닌 성분들이 주대각에만 있고, 나머지 성분들은 모두 0인 행렬

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \qquad D_{i,j} = 0\ (i\ != j)$$

- 단위행렬: 주대각 성분들이 모두 1인 대각행렬

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- $diag(v)$: 벡터 v의 성분들이 주대각 성분들인 정방 대각행렬 표시

$$v = [\ a\ b\ c\ ] \qquad diag(v) \Rightarrow \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix}$$

2.6 특별한 종류의 행렬과 벡터

- 대칭행렬 : 전치행렬이 자신과 같은 행렬

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix} \qquad \boldsymbol{A} = \boldsymbol{A}^T$$

- 단위벡터 : 크기가 단위노름(unit norm)인 벡터

$$\|\boldsymbol{x}\|_2 = 1$$

2.6 특별한 종류의 행렬과 벡터

- 정규직교벡터 : 직교이면서 크기가 단위노름(1)인 벡터
- 직교행렬 : 행들이 서로 정규직교이고 열들도 서로 정규직교인 정방행렬

$$\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{I} \quad \Rightarrow \quad \boldsymbol{A}^{-1} = \boldsymbol{A}^T$$

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \boldsymbol{A}^T\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \boldsymbol{I}$$

2.7 고윳값 분해

- 12의 소인수분해: 12 = 2 X 2 X 3
- 소인수분해 표현에서 여러 유용한 성질을 이끌어낼 수 있음

- 마찬가지로 행렬을 분해하면 행렬의 특정 성질을 분석하는데 도움이 됨

- 가장 널리 쓰이는 행렬 분해 방법: **고윳값 분해, 특잇값 분해 등**

- **고윳값 분해:** 행렬을 고유벡터들과 고윳값들로 분해 (행렬이 정방행렬일 때)

$\boldsymbol{A}\boldsymbol{v} = \lambda\boldsymbol{v}$ 를 만족하는 0이 아닌 열벡터 $\boldsymbol{v}$를 고유벡터, 상수 λ를 고유값이라 정의

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} = \lambda \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} \quad \Rightarrow \quad \boldsymbol{V} = \left[\, v^{(1)}, \dots, v^{(n)} \,\right], \qquad \boldsymbol{\lambda} = \left[\, \lambda^{(1)}, \dots, \lambda^{(n)} \,\right]$$

$\boldsymbol{A}$의 고윳값 분해 식: $\boldsymbol{A} = \boldsymbol{V}\, diag(\boldsymbol{\lambda})\boldsymbol{V}^{-1}$

2.8 특잇값 분해

- 특잇값 분해: 행렬을 특이벡터들과 특잇값들로 분해

- 얻을 수 있는 정보는 고윳값 분해와 동일하지만, 고윳값 분해보다 더 일반적인 행렬들에 적용 가능함
- 모든 실수 행렬에는 특잇값 분해가 존재하지만 ,항상 고윳값 분해가 존재하는 것은 아님

$\boldsymbol{A}$의 특잇값 분해 식: $\boldsymbol{A} = \boldsymbol{UDV}^T$

$\boldsymbol{A} = m\,X\,n. \quad \Rightarrow. \quad \boldsymbol{U} = m\,X\,m, \quad \boldsymbol{D} = m\,X\,n, \quad \boldsymbol{V} = n\,X\,n$

*. $\boldsymbol{U}, \boldsymbol{V}$ 는 직교행렬, $\boldsymbol{D}$는 대각행렬

- D의 주대각 성분을 행렬 A의 특잇값이라고 부름
- U의 열을 좌특이벡터라고 부름
- V의 열을 우특이벡터라고 부름

2.9 무어-펜로즈 유사역행렬

- 정방행렬이 아닌 행렬에는 역행렬이 정의되지 않음

$$Ax = y$$

양변에 A의 역행렬인 B를 곱해주면

$$x = By$$

- 만약 A가 세로로 긴 비정방행렬이면 해가 없을 수 있고, 가로로 긴 비정방행렬이면 해가 여러 개일 수 있음
- 이런 경우에 사용하면 좋은 것이 **무어-펜로즈 유사역행렬**임

- A의 유사역행렬 : $\boldsymbol{A}^{+} = \lim_{\alpha \searrow 0} (\boldsymbol{A}^{T}\boldsymbol{A} + \alpha\boldsymbol{I})^{-1}\boldsymbol{A}^{T}$

- But, 실제로 쓰이는 알고리즘은 위가 아니라 $\boldsymbol{A}^{+} = \boldsymbol{V}\boldsymbol{D}^{+}\boldsymbol{U}^{T}$.

- 유사역행렬을 적용하면
- A가 행보다 열이 많은 행렬일 때: 여러 해가 나오는데 그 중 유클리드 노름 $\|x\|_2$ 가 최소인 $x = A^{+}y$ 를 얻게 됨
- A가 열보다 행이 많은 행렬일 때: 유클리드 노름 $\|Ax - y\|_2$ 를 측정 기준으로 해서 y와 최대한 가까운 Ax에 해당하는 x를 얻게 됨

2.10 대각합 연산자

- 대각합 연산자 (Tr) : 행렬의 모든 주대각 성분의 합

$$Tr(A) = \sum_i A_{i,i}.$$

- 대각합 연산자가 유용한 이유
  - 시그마 없이는 표현하기 어려운 연산 표현 가능
  - 예를 들면, 행렬의 프로베니우스 노름을 이전과는 다른 방식으로 표현 가능

$$\|A\|_F = \sqrt{Tr(AA^T)}$$

- 대각합 연산자는 전치 연산자에 대해 불변 $Tr(A) = Tr(A^T)$

- 여러 행렬의 곱으로 이루어진 정방행렬의 대각합은 행렬곱의 위치를 변경해서 변하지 않음

$$Tr(ABC) = Tr(CAB) = Tr(BCA) \quad \Rightarrow$$

이 식을 임의의 개수의 행렬로 일반화 $\Rightarrow$ $Tr(\prod_{i=1}^{n} F^{(i)}) = Tr(F^{(n)} \prod_{i=1}^{n-1} F^{(i)})$

2.11 행렬식

- 행렬식: 행렬을 실수 스칼라로 사상하는 함수 $\det(A)$
- 행렬식은 행렬의 모든 고윳값을 곱한 것과 같음
- 행렬식의 절댓값: 주어진 행렬을 곱했을 때 공간이 얼마나 확장 또는 축소되었는지를 나타내는 척도

2.12 주성분분석 (PCA)

- $\mathbb{R}^n$에 있는 m개의 점들의 집합 $\{x^{(1)}, \dots, x^{(m)}\}$ 에 유손실 압축을 적용한다고 가정
  - 유손실 압축: 점들을 저장하는데 필요한 메모리는 줄이되 정밀도를 어느정도 잃는 것이 허용되는 압축

  - 필요한 저장공간을 줄이는 방법 중 하나는 점들을 더 낮은 차원으로 부호화하는 것

  - 만약 l이 n보다 작다면, 원래의 자료보다 더 적은 메모리로 부호점을 저장할 수 있음
    - 그러기 위해서는
      - 주어진 점에 대한 부호를 산출하는 부호화 함수 $f(x) = c$,
      - 주어진 부호로부터 원래의 점을 재구축하는 복호화 함수 $x \approx g(f(x))$ 가 필요

    - PCA를 복호화 함수로 사용할 수 있음

- 각 입력점 x에 대한 최적의 부호점c^* 를 생성하는 방법
  - 입력점 x와 그것의 재구축 결과인 $g(c^*)$ 사이의 거리를 최소화 하는 것
    - 거리는 노름으로 측정 가능
    - 주성분 분석 알고리즘에서는 L^2 노름을 사용

$$c^* = argmin\|x - g(c)\|_2$$

2.12 주성분분석 (PCA)

- L^2 노름 대신 제곱L^2 노름 써도 됨 (결과는 같을 것임)

* $g(c)$를 $g(c) = Dc$ 라고 정의

$$c^* = argmin_c \|x - g(c)\|_2^2$$

- 최소화 대상 함수를 따로 전개

$$(x - g(c))^T (x - g(c))$$

$$\Rightarrow. \quad x^T x - x^T g(c) - g(c)^T x + g(c)^T g(c)$$

$$\Rightarrow. \quad x^T x - 2x^T g(c) + g(c)^T g(c)$$

- 최소화 대상 함수에서 첫 항은 c 에 의존하지 않으므로 생략 가능

$$c^* = argmin_c \; -2x^T g(c) + g(c)^T g(c)$$

$$\Rightarrow. \quad c^* = argmin_c \; -2x^T Dc + c^T D^T Dc$$

$$\Rightarrow. \quad c^* = argmin_c \; -2x^T Dc + c^T c$$

2.12 주성분분석 (PCA)

- 미적분을 이용해서 최적화 문제를 풀 수 있음

$$\nabla_c \left(-2x^T Dc + c^T c\right) = 0$$

$$-2D^T x + 2c = 0$$

$$c = D^T x$$

$$\Rightarrow. \quad f(x) = D^T x$$

2.12 주성분분석 (PCA)

- 행렬 곱셈을 하나 더 사용하면 PCA 재구축 연산을 정의할 수 있음

$$r(x) = g\big(f(x)\big) = DD^Tx$$

$$D^* = argmin_D \sqrt{\sum_{i,j}(x_j^{(i)} - r(x^{(i)})_j)^2}$$

$$d^* = argmin_d \sum_i \left\|x^{(i)} - dd^Tx^{(i)}\right\|_2^2$$

$$d^* = argmin_d \|X - Xdd^T\|_F^2$$

* 노름 식

$$\|x\|_p = (\sum_i |x_i|^p)^{\frac{1}{p}}$$

2.12 주성분분석 (PCA)

$$argmin_d \|X - Xdd^T\|_F^2$$

* 프로베니우스 노름 식

$$\|A\|_F = \sqrt{Tr(AA^T)}$$

$$= argmin_d \, Tr((X - Xdd^T)^T(X - Xdd^T))$$

$$= argmin_d \, Tr(X^TX - X^TXdd^T - dd^TX^TX + dd^TX^TXdd^T)$$

$$= argmin_d \, Tr(X^TX) - Tr(X^TXdd^T) - Tr(dd^TX^TX) + Tr(dd^TX^TXdd^T)$$

$$= argmin_d \, -Tr(X^TXdd^T) - Tr(dd^TX^TX) + Tr(d^TX^TXdd^T)$$

$$= argmin_d \, -2Tr(X^TXdd^T) + Tr(dd^TX^TXdd^T)$$

$$= argmin_d \, -2Tr(X^TXdd^T) + Tr(X^TXdd^Tdd^T)$$

$$= argmin_d \, -Tr(X^TXdd^T)$$

$$= argmax_d \, Tr(X^TXdd^T)$$

$$= argmax_d \, Tr(d^TX^TXd^T)$$

Thank You

감사합니다.