딥 러닝

벡터와 행렬

▼ 토글

- **벡터**

  스칼라를 원소로 가지는 1차원 배열

  n차원 공간 상의 한 점, 원점으로부터의 상대적 위치

- **norm**

  원점에서부터의 거리

  임의의 d차원에 대해 성립 (어떤 차원에서든 norm을 정의할 수 있음)

  - L1 norm

    각 성분의 변화량의 절대값을 모두 합산

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

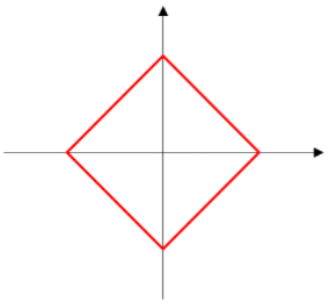
  - L2 norm

    피타고라스 정리를 이용해 유클리드 거리를 계산

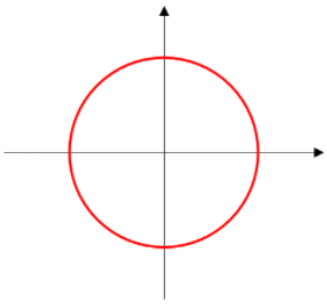
$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{d} |x_i|^2}$$

  norm은 d차원 공간에서 거리를 정의하는 방법이기 때문에 사용하는 norm의 종류에 따라서 도형의 기하학적 성질이 바뀌기도 하며, 딥 러닝 분야에서는 필요에 따라 L1 norm이 사용되는 경우가 있음

  ex) 원의 정의 : 중심으로부터 거리가 동일한 점들의 집합



L1 norm 공간 상의 원



L2 norm 공간 상의 원

- **행렬**

  벡터를 원소로 갖는 2차원 배열

  n차원 공간 상의 점들의 집합 (행렬의 i번째 행/열벡터가 i번째 데이터라고 볼 수 있음)

  벡터/행렬의 차원을 이동시키는 연산자

  > 벡터에 어떤 행렬을 곱했을 때 행렬의 형태에 따라 벡터의 차원이 변하기 때문에 행렬을 곱하는 것은 벡터나 행렬을 다른 차원 공간으로 보내는 연산자로 해석하는 것도 가능
  >
  > *모든 선형 변환은 행렬곱으로 나타낼 수 있다

- **역행렬 (A^{-1})**

  행렬곱을 되돌리는 연산

  행렬에 역행렬을 곱하면 단위행렬 I가 되며, 이 때 교환법칙 성립

  행렬 A가 정칙 행렬일 때만 역행렬이 존재 (n x n이며 determinant를 구할 수 있어야 함)

- **의사 역행렬** (A^{+}, pseudo inverse, 무어-펜로즈 역행렬)

  정칙 행렬이 아닌 행렬의 역행렬로, 일반 역행렬과 구별하기 위해 +기호로 표현

  m x n 행렬 A가 존재할 때 n과 m의 차이에 따라 곱해지는 위치가 다름 (행렬의 모양 때문)

  > m > n (좌 역행렬) $A^{+} = (A^{T}A)^{-1}A^{T},\ A^{+}A = I$

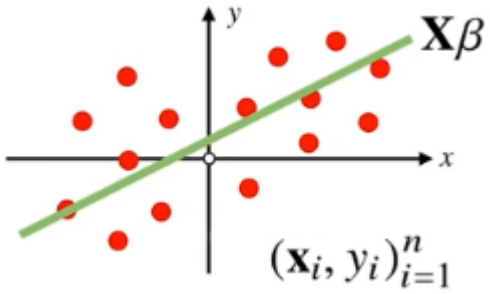
m < n (우 역행렬) $A^{+} = A^{T}(AA^{T})^{-1},\ AA^{+} = I$

- 선형 회귀 분석

  연립방정식의 변수가 식의 수보다 많은 경우(부정) 의사 역행렬을 이용하면 무한한 해 중 하나를 구할 수 있음 (변수 개수보다 많은 데이터가 주어졌을 때 선형 회귀식을 찾는 등)

$$\begin{bmatrix} — \mathbf{x}_1 — \\ — \mathbf{x}_2 — \\ \vdots \\ — \mathbf{x}_n — \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

$$\mathbf{X}\beta = \mathbf{y}$$



최적화

▼ 토글

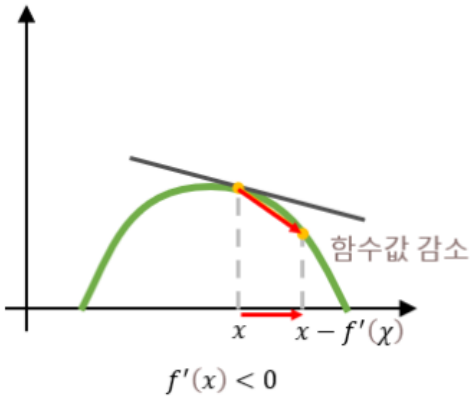
- **미분**

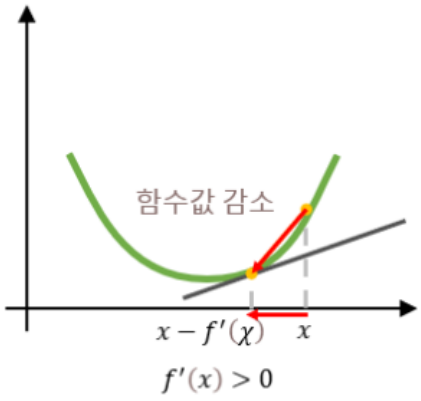
  주어진 점 $(x, f(x))$에서 함수 f의 접선의 기울기

  변수의 움직임에 따른 함수값의 변화를 측정하기 위한 도구

  미분값을 더하면 함수값 증가, 미분값을 빼면 함수값 감소

  함수의 오목/볼록한 모양, 그에 따른 x값에서의 접선의 기울기와 관련이 있음





- **경사 하강법**

  함수의 극소/극대를 구하는 데 미분을 사용 (접선의 기울기에 따라 미분값을 빼는/더하는 것)

  극소값을 구하기 위해 미분값을 조금씩 빼 나가는 방법

  학습률(learning rate)을 조정하면 경사 하강법의 수렴 속도를 조절 가능

  단, 적절한 값으로 설정하지 않았을 경우 느리게 업데이트되거나 발산할 수 있음

- **다변수 함수의 경사 하강법**

  다변수 함수에서 각 변수의 변화에 따른 함수값의 변화를 알고 싶을 때는 편미분을 사용

  기울기 벡터 (Gradient) : 각 변수에 대해 편미분한 것

$$\nabla f = (\partial_{x_1} f, \partial_{x_2} f, \cdots, \partial_{x_d} f)$$

- **다변수함수의 선형 회귀에 경사 하강법 적용**

  경사 하강법을 적용할 경우 유사 역행렬을 사용하지 않고 선형 회귀 분석 가능

  - 선형 회귀의 목적식 (손실 함수)

    입력으로 주어지는 모든 x들과 y 사이의 거리 $||y - X\beta||2$

    (=L2 norm, =Root Mean Square)

  딥 러닝의 목표는 손실 함수의 값이 최소가 되는 위치를 찾는 것으로, 손실 함수에 경사 하강법을 적용하여 최소가 되는 지점을 찾을 수 있음

  그러나 X와 y는 고정 입력/출력으로 주어지기 때문에, 손실 함수값이 줄어들도록 하기 위해서는 β를 업데이트해야 함

  각 β에 대해 편미분한 Gradient를 구하고 경사 하강법을 통해 β를 업데이트

  선형 회귀의 경우 목적식(L2 norm)이 회귀계수 β에 대해 오목한(최소=극소) 함수라서 경사 하강법을 적용할 경우 항상 수렴하는 것을 보장할 수 있음

  단, 비선형 회귀 문제는 목적식이 오목하지 않은 경우가 있어 경사 하강법을 변형해서 사용

- **확률적 경사 하강법(SGD)**

  경사 하강법을 적용할 때 모든 데이터를 다 사용하는 것이 아니라 일부(mini batch)만 사용해서 변수를 업데이트하는 방법

  목적식이 볼록이 아닌(최소≠극소) 경우에도 최적화할 수 있음

  확률적으로 선택된 미니 배치를 이용해 그래디언트 벡터를 계산하기 때문에 목적식의 모양이 바뀌게 되어 극소값에서 탈출할 가능성이 존재하기 때문

  하드웨어적 장점 : 데이터를 미니 배치로 조각내서 사용할 경우 GPU에서 연산하는 동안 CPU에서는 데이터 전처리 및 다음 미니 배치 준비를 할 수 있어 메모리를 효율적으로 사용 가능