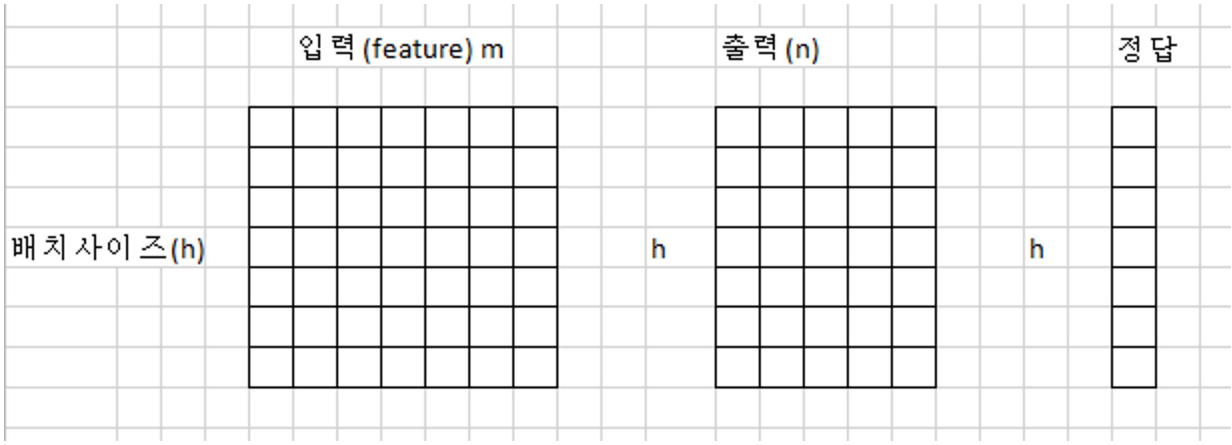
07. 행렬 연산

2020년 12월 31일 목요일 오전 11:03



1. 행렬을 이용한 순전파

● 입력 x, 가중치 w 라 했을 때

$$XW = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \square & \vdots \\ x_{h1} & x_{h2} & \cdots & x_{hm} \end{bmatrix} \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \square & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{k=1}^{m} x_{1k}w_{k1} & \sum_{k=1}^{m} x_{1k}w_{k2} & \cdots & \sum_{k=1}^{m} x_{1k}w_{kn} \\ \sum_{k=1}^{m} x_{2k}w_{k1} & \sum_{k=1}^{m} x_{2k}w_{k2} & \cdots & \sum_{k=1}^{m} x_{2k}w_{kn} \\ \vdots & \vdots & \square & \vdots \\ \sum_{k=1}^{m} x_{hk}w_{k1} & \sum_{k=1}^{m} x_{hk}w_{k2} & \cdots & \sum_{k=1}^{m} x_{hk}w_{kn} \end{bmatrix}$$

행 수가 배치 사이즈(h), 열 수가 출력 뉴런(노드)의 수(n)

● bias 는 다음과 같은 벡터인데, 계산 시 넘파이의 브로드캐스트 기능을 이용

$$\vec{b} = (b_1, b_2, \cdots, b_n)$$

● bias 까지 더한 행렬의 모습은 아래와 같음
bias = 출력 수(n)

numpy는 element-wise 사칙연산함
2개의 shape이 동일한 경우 가능

브로드캐스트 기능은,
A(3*3M)과 B(1*3M)을
A(3*3M)과 B[3, (1*3M)]으로 계산해줌

$$U = \begin{bmatrix} \sum_{k=1}^{m} x_{1k}w_{k1} + b_1 & \sum_{k=1}^{m} x_{1k}w_{k2} + b_2 & \cdots & \sum_{k=1}^{m} x_{1k}w_{kn} + b_n \\ \sum_{k=1}^{m} x_{2k}w_{k1} + b_1 & \sum_{k=1}^{m} x_{2k}w_{k2} + b_2 & \cdots & \sum_{k=1}^{m} x_{2k}w_{kn} + b_n \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{k=1}^{m} x_{hk}w_{k1} + b_1 & \sum_{k=1}^{m} x_{hk}w_{k2} + b_2 & \cdots & \sum_{k=1}^{m} x_{hk}w_{kn} + b_n \end{bmatrix}$$

● 이 행렬의 각 원소는 활성화 함수 f 로 처리하고 출력 Y를 다음과 같이 구함

$$Y = f(U) = \begin{bmatrix} f(\sum_{k=1}^{m} x_{1k}w_{k1} + b_1) & f(\sum_{k=1}^{m} x_{1k}w_{k2} + b_2) & \cdots & f(\sum_{k=1}^{m} x_{1k}w_{kn} + b_n) \\ f(\sum_{k=1}^{m} x_{2k}w_{k1} + b_1) & f(\sum_{k=1}^{m} x_{2k}w_{k2} + b_2) & \cdots & f(\sum_{k=1}^{m} x_{2k}w_{kn} + b_n) \\ \vdots & \vdots & \ddots & \vdots \\ f(\sum_{k=1}^{m} x_{hk}w_{k1} + b_1) & f(\sum_{k=1}^{m} x_{hk}w_{k2} + b_2) & \cdots & f(\sum_{k=1}^{m} x_{hk}w_{kn} + b_n) \end{bmatrix}$$

층의 출력 Y는 h X n, 즉 배치 사이즈 X 뉴런(노드) 수 행렬이 됨

위의 연산을 넘파이를 이용하여 표현하면 다음과 같이 간단히 정리됨
** 활성화 함수가 시그모이드인 경우

```
u = np.dot(x, w) + b
y = 1 / (1+np.exp(-u) )
```

자동으로 numpy 브로드캐스팅되어 u 출력

2. 행렬을 이용한 역전파

● δ 의 행렬 Δ 는 다음과 같음

$$\Delta = \begin{bmatrix} \delta_{11} & \delta_{12} & \cdots & \delta_{1n} \\ \delta_{21} & \delta_{22} & \cdots & \delta_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{h1} & \delta_{h2} & \cdots & \delta_{hn} \end{bmatrix}$$

Δ 는 층의 출력 Y와 동일하게 h X n, 즉 배치 사이즈 X 뉴런(노드) 수 행렬임

● ∂w_{ij} 는 다음과 같다.

$$\partial w_{ij} = \frac{\partial E}{\partial w_{ij}} = y_i \delta_j$$

앞 층의 출력 y_i 는 현재 층의 입력과 같으므로 이 후 행렬식에서 y 대신 x로 표기할 것임

배치 사이즈를 처리하기 위해 구한 기울기값을 모두 더해야 함, 다음과 같은 식으로 정리됨

$$\sum_{k=1}^{h} \frac{\partial E_k}{\partial w_{ij}}$$

● 이상의 과정을 거쳐 배치를 고려한 가중치의 기울기 행렬 ∂W 는 다음과 같음

$$\partial W = \mathrm{X}^T\Delta = \begin{bmatrix} x_{11} & x_{21} & \cdots & x_{h1} \\ x_{12} & x_{22} & \cdots & x_{h2} \\ \vdots & \vdots & \square & \vdots \\ x_{1m} & x_{2m} & \cdots & x_{hm} \end{bmatrix} \begin{bmatrix} \delta_{11} & \delta_{12} & \cdots & \delta_{1n} \\ \delta_{21} & \delta_{22} & \cdots & \delta_{2n} \\ \vdots & \vdots & \square & \vdots \\ \delta_{h1} & \delta_{h2} & \cdots & \delta_{hn} \end{bmatrix} \quad \delta : \text{배치수h} * \text{출력n}$$

$$= \begin{bmatrix} \sum_{k=1}^{h} x_{k1}\delta_{k1} & \sum_{k=1}^{h} x_{k1}\delta_{k2} & \cdots & \sum_{k=1}^{h} x_{k1}\delta_{kn} \\ \sum_{k=1}^{h} x_{k2}\delta_{k1} & \sum_{k=1}^{h} x_{k2}\delta_{k2} & \cdots & \sum_{k=1}^{h} x_{k2}\delta_{kn} \\ \vdots & \vdots & \square & \vdots \\ \sum_{k=1}^{h} x_{km}\delta_{k1} & \sum_{k=1}^{h} x_{km}\delta_{k2} & \cdots & \sum_{k=1}^{h} x_{km}\delta_{kn} \end{bmatrix}$$

X 와 Δ의 행렬 곱을 위해 X^T 처럼 X를 전치시켜 앞쪽 행렬의 컬럼 수와 뒤쪽 행렬의 로우 수를 맞춰야 함

결과적으로 얻게 되는 행렬의 각 원소는 배치 내의 총합을 구한 것

이 행렬의 크기는 m X n 이고 W의 크기와 일치

위 행렬 곱을 파이썬 코드로 나타내면 다음과 같음

```
grad_w = np.dot(x.T, delta)
```

오차는 가중치 기울기 행렬

grad_w는 가중치 기울기 행렬 ∂W 이고 x는 입력 행렬 X, delta는 δ의 행렬 Δ를 나타냄

● bias의 기울기는 다음의 식을 이용해 구함

$$\partial \mathrm{b}_j = \delta_j$$

배치를 고려한 기울기는 다음과 같이 δ를 배치 내에서 모두 더함

$$\sum_{i=1}^{h} \frac{\partial E_k}{\partial b_j}$$

위 식을 파이썬 코드로 나타내면 다음과 같음

```
grad_b = np.sum(delta, axis=0)
```

● 앞 층의 출력 기울기(현재 층의 입력 기울기)는 다음의 식과 같음

$$\partial \mathrm{y}_i = \sum_{r=1}^{n} \delta_r w_{ir}$$

가중치 기울기와 마찬가지로 앞 층의 출력 y_i 는 현재 층의 입력과 같기 때문에 y를 x로 바꾸어 표기

∂y 를 배치에 고려한 행렬을 ∂X 로 하며 다음과 같이 정리됨

$$\partial X = \Delta W^T = \begin{bmatrix} \delta_{11} & \delta_{12} & \cdots & \delta_{1n} \\ \delta_{21} & \delta_{22} & \cdots & \delta_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{h1} & \delta_{h2} & \cdots & \delta_{hn} \end{bmatrix} \begin{bmatrix} w_{11} & w_{21} & \cdots & w_{m1} \\ w_{12} & w_{22} & \cdots & w_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ w_{1n} & w_{2n} & \cdots & w_{mn} \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{k=1}^{n} \delta_{1k} w_{1k} & \sum_{k=1}^{n} \delta_{1k} w_{2k} & \cdots & \sum_{k=1}^{n} \delta_{1k} w_{mk} \\ \sum_{k=1}^{n} \delta_{2k} w_{1k} & \sum_{k=1}^{n} \delta_{2k} w_{2k} & \cdots & \sum_{k=1}^{n} \delta_{2k} w_{mk} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{k=1}^{n} \delta_{hk} w_{1k} & \sum_{k=1}^{n} \delta_{hk} w_{2k} & \cdots & \sum_{k=1}^{n} \delta_{hk} w_{mk} \end{bmatrix}$$

Δ와 W의 행렬 곱을 위해 W^T 와 같이 W를 전치시키며, 이를 통해 Δ의 컬럼 수와 W의 로우 수가 일치되고 행렬 곱을 할 수 있음

결과적으로 얻게 되는 행렬의 각 원소는 현재 층에 있는 모든 뉴런(노드)의 결과를 합한 것이 됨

위 식을 파이썬 코드로 표현하면 다음과 같음

```
grad_x = np.dot(delta, w.T)
```