3주차 요약 정리

01. 파이썬에서의 브로드캐스팅

✅**Broadcasting example**

- for문을 사용하지 않고 각 음식에 포함된 탄수화물/단백질/지방의 비율을 계산하기

Calories from Carbs, Proteins, Fats in 100g of different foods:

	Apples	Beef	Eggs	Potatoes
Carb	56.0	0.0	4.4	68.0
Protein	1.2	104.0	52.0	8.0
Fat	1.8	135.0	99.0	0.9

1️⃣ **… 네 가지 음식에 포함된 칼로리의 합(cal)**

2️⃣ **… 네 열을 각 열의 합으로 나누기(percentage)**

```
import numpy as np

A=np.array([[56.0,0.0,4.4,68.0],[1.2,104.0,52.0,8.0],[1.8,1
cal=A.sum(axis=0) # axis=0-세로로 더하라
percentage=100*A/cal.reshape(1,4) #파이썬의 브로드캐스팅의 예시-
# 각 음식의 탄수화물/단백질/지방의 구성 비율을 (3,4) 행렬로 출력
print(percentage)
```

✅ **General Principle**

$$
\begin{array}{cccc}
(m,n) & & (1,n) & \cdots> (m,n) \\
 & +\;-\;/\;* & (m,1) & \cdots> (m,n) \\
 & & \mathbb{R} & \cdots> \mathbb{R}^{(m,n)}
\end{array}
$$

02. 파이썬과 넘파이 벡터

✅ 브로드캐스팅의 장단점

장점: 언어의 넓은 표현성과 유연성을 짧은 코드로 해결해 줌

단점: 이 유연성은 브로드캐스팅의 자세한 내용과 작동 방법을 모른다면 이상하고 찾기 어려운 오류가 생기기도 함

```
import numpy as np

a=np.random.randn(5) #가우시안 분포를 따르는 5개의 숫자가 랜덤으로
print(a.shape) #(5,): rank=1(열 벡터도, 행 벡터도 아님); a==a.T
print(np.dot(a,a.T)) # 실수 하나가 나옴

a=np.random.randn(5,1) #(5,1)인 행렬
print(np.dot(a,a.T)) #(5,5) 행렬
```

🚨 **python에서 프로그래밍 예제/신경망에서 로지스틱 회귀 코드를 작성할 때 rank=1(n,)인 이상한 배열을 사용하지 않도록 하자→대신 행 벡터/열 벡터를 사용한다면 벡터의 연산을 훨씬 잘 이해할 수 있다**

03. 로지스틱 회귀의 비용함수 설명

$$\left.\begin{array}{l} \text{If } y=1 : p(y|x)=\hat{y} \\ \text{If } y=0 : p(y|x)=1-\hat{y} \end{array}\right] \oplus p(y|x)$$

$$p(y|x)=\hat{y}^{y}(1-\hat{y})^{(1-y)}$$

If $y=1$: $p(y|x)=\hat{y}\cdot 1=\hat{y}$

If $y=0$: $p(y|x)=1\cdot(1-\hat{y})=1-\hat{y}$

* log는 단조증가 함수 ⇒ $\log p(y|x)$를 최대화하는 것은 $p(y|x)$를 최대화하는 것과 같음

$$\therefore \log p(y|x)=\log(\hat{y}^{y}(1-\hat{y})^{(1-y)})=y\log\hat{y}+(1-y)\log(1-\hat{y})$$

$$=\boxed{-\mathcal{L}(\hat{y},y)}$$

최대화

최소화

* Cost on m examples

$$\log p(\text{labels in training set})=\log\prod_{i=1}^{m}p(y^{(i)}|x^{(i)})$$

$$\log p(\cdots)=\sum_{i=1}^{m}\log\underbrace{p(y^{(i)}|x^{(i)})}_{=-\mathcal{L}(\hat{y}^{(i)},y^{(i)})}$$

$-\mathcal{L}(\hat{y},y)=\log p(\cdots)$

* Maximum Likelihood Estimation (MLE) - 특정 값을 최대화하라

Cost : $J(w,b)=-\log p(\text{labels in training set})=\frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)},y^{(i)})$

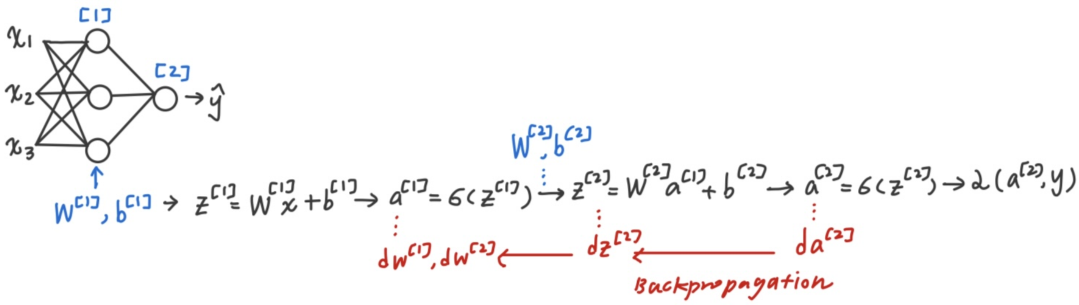
must be minimized

for scaling

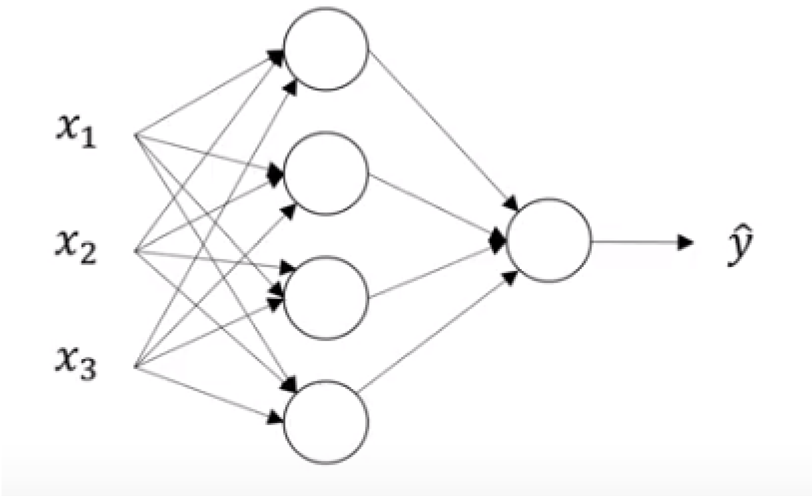
최대화

최소화

04. 신경망 네트워크 개요



05. 신경망 네트워크의 구성 알아보기



- input layer($a^{[0]}$=X)-hidden layer($a^{[1]}$)-output layer($a^{[2]}) = \hat{y}$)
- 은닉층은 입력층과 출력층 사이의 모든 층을 의미하며, 은닉층에 있는 값은 알 수 없음
- l번째 은닉층의 n번째 노드는 $a_n^{[l]}$으로 표기함
    - 예를 들어 1번째 은닉층의 첫 번째 노드는 $a_1^{[1]}$으로 표기, 1번째 은닉층의 두 번째 노드는 $a_2^{[1]}$으로 표기
- $a^{[0]}$에서 a는 활성값을 의미
- 입력층에서 X가 은닉층으로 넘어가면 $a^{[1]} = (a_1^{[1]}, a_2^{[1]}, a_3^{[1]}, a_4^{[1]})$을 내놓음. 이때 은닉층은 파라미터 $w^{[1]}, b^{[1]}$와 관련되어 있음. $w^{[1]}$은 (4,3) 행렬, $b^{[1]}$은 (4,1) 행렬임
- 출력층은 실수 $\hat{y} = a^{[2]}$를 내놓고, 파라미터 $w^{[2]}, b^{[2]}$와 관련되어 있음. $w^{[2]}$은 (1,4) 행렬, $b^{[2]}$은 (1,1) 행렬임
- n번째 층에서 파라미터의 차원은 w는 (n번째 노드의 개수, n-1번째 노드의 개수), b는 (n번째 노드의 개수, 1)

- 입력층은 층 수를 세는 데 포함되지 않음. 위의 그림은 2층 신경망임

06. 신경망 네트워크 출력의 계산

- 하나의 노드에서 두 개의 연산을 거침. l번째 층에서 n번째 노드라고 한다면,
    1. $z_n^{[l]} = w_n^{[l]T}x + b_n^{[l]}$
    2. $a_n^{[l]} = \sigma(z_n^{[l]})$

    ➡️ 만약 for문을 통해 이 연산을 해당 층의 모든 노드에 대해 한다면 상당히 비효율적일 것→**벡터화 필요!**

$$z^{[1]} = \underbrace{\overbrace{\begin{bmatrix} - w_1^{[1]T} - \\ - w_2^{[1]T} - \\ - w_3^{[1]T} - \\ - w_4^{[1]T} - \end{bmatrix}}^{W^{[1]}}}_{(4\times 3)} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{(3\times 1)} + \underbrace{\overbrace{\begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix}}^{b^{[1]}}}_{(4\times 1)} = \underbrace{\begin{bmatrix} w_1^{[1]T}x + b_1^{[1]} \\ w_2^{[1]T}x + b_2^{[1]} \\ w_3^{[1]T}x + b_3^{[1]} \\ w_4^{[1]T}x + b_4^{[1]} \end{bmatrix}}_{(4\times 1)} = \underbrace{\begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix}}_{(4\times 1)} = \underset{(4\times 3)}{W^{[1]}}\underset{(3,1)}{x} + \underset{(4,1)}{b^{[1]}}$$

$$\underset{(4,1)}{a^{[1]}} = \sigma(\underset{(4,1)}{z^{[1]}})$$

$$\underset{(1,1)}{z^{[2]}} = \underset{(1,4)}{W^{[2]}}\underset{(4,1)}{x} + \underset{(1,1)}{b^{[2]}}$$

$$\underset{(1,1)}{a^{[2]}} = \sigma(\underset{(1,1)}{z^{[2]}})$$

이 네 개의 방정식만으로 연산 가능!

07. 많은 샘플에 대한 벡터화

- $a^{[j](i)}$
    - j : j번째 층
    - i : i번째 훈련 데이터
- 만약 m개의 데이터셋에 대해 네 개의 연산을 반복하고 싶다면, 모든 z와 a에 첨자 $[i]$를 붙이고 for i in range(1,m+1)문을 돌려야 함→**벡터화 해야 함!**

non-vectorized	vectorized
for i in range(1, m+1):	$Z^{[1]} = W^{[1]}X + b^{[1]}$
$z^{[1](i)} = W^{[1]}x^{(i)} + b^{[1]}$	$A^{[1]} = \sigma(Z^{[1]})$
$a^{[1](i)} = \sigma(z^{[1](i)})$	$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$
$z^{[2](i)} = W^{[2]}a^{[1](i)} + b^{[2]}$	$A^{[2]} = \sigma(Z^{[2]})$
$a^{[2](i)} = \sigma\ z^{[2](i)})$	

where $Z^{[1]} = \begin{bmatrix} | & | & & | \\ z^{[1](1)} & z^{[1](2)} & \cdots & z^{[1](m)} \\ | & | & & | \end{bmatrix}$,

$$A^{[1]} = \begin{array}{c} \left[\begin{array}{cccc} | & | & & | \\ a^{[1](1)} & a^{[1](2)} & \cdots & a^{[1](m)} \\ | & | & & | \end{array}\right] \text{the number of nodes} \\ \text{— the number of examples (m) —} \end{array}$$