3주차

날짜	@2024년 3월 26일
과제	강의 요약 발표 출석 퀴즈
세부내용	[딥러닝 1단계] 3-2. 파이썬과 벡터화 (파이썬의 브로드캐스팅~로지스틱 회귀의 비용함수 설명) 4-1. 얕은 신경망 네트워크 (신경망 네트워크 개요~벡터화 구현에 대한 설명)

파이썬과 벡터화

5️⃣ 파이썬의 브로드캐스팅

브로드캐스팅 예제 (1)

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

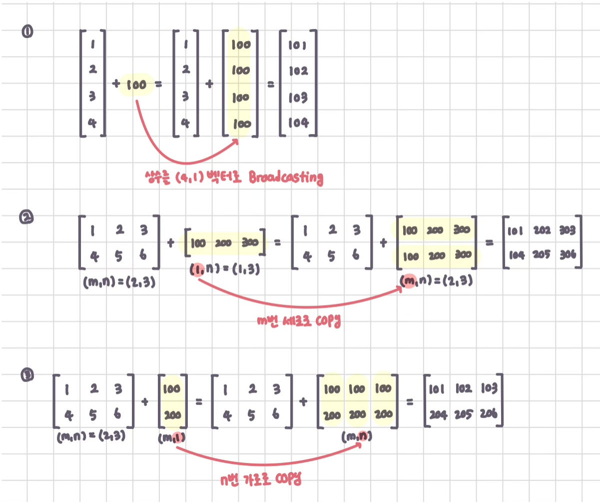
$$A = \begin{bmatrix} & \text{Apples} & \text{Beef} & \text{Eggs} & \text{Potatoes} \\ \text{Carb} & 56.0 & 0.0 & 4.4 & 68.0 \\ \text{Protein} & 1.2 & 104.0 & 52.0 & 8.0 \\ \text{Fat} & 1.8 & 135.0 & 99.0 & 0.9 \end{bmatrix}$$

- Goal : Calculate % of calories from Carb, Protein, Fat without for-loop.
- 사과 100g의 칼로리 = 56.0 + 1.2 + 1.8 = 59.0cal
    - 탄수화물의 칼로리 비율 = 56.0 / 59.0 = 94.9%
- Sol : ① 3×4의 A 행렬을 가지고 각 음식의 칼로리를 계산
  ② 각 네 열을 각 열의 합으로 나누기

```
# 행렬 A 생성
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, 135.0, 99.0, 0.9]])

# 음식 별 칼로리 계산 (= 열 별 합계 계산)
cal = A.sum(axis=0)
print(cal)
# [ 59.  239.  155.4  76.9

# 탄수화물, 단백질,  지방의 칼로리 비율 계산
percentage = 100 * A / cal.reshape(1,4)
print(percentage)
# [[94.91525424  0.          2.83140283 88.42652796]
#  [ 2.03389831 43.51464435 33.46203346 10.40312094]
#  [ 3.05084746 56.48535565 63.70656371  1.17035111]]
```

- axis = 0 : 세로 방향, axis = 1 : 가로 방향
- 3×4 행렬 A를 1x4 행렬로 나눔 → 브로드캐스팅의 예

브로드캐스팅 예제 (2)



브로드캐스팅의 일반적인 원리

- (m,n) 행렬과 (1,n) 행렬의 사칙연산 → (1,n) 행렬을 m번 복사해 (m,n) 행렬로 변환 후 요소별 계산
- (m,n) 행렬과 (m,1) 행렬의 사칙연산 → (m,1) 행렬을 n번 복사해 (m,n) 행렬로 변환 후 요소별 계산
- (m,1) 행렬/열벡터와 상수의 사칙연산 → 상수를 m번 복사해 (m,1) 행렬 생성 후 요소별 계산
- (1,m) 행렬/행벡터와 상수의 사칙연산 → 상수를 m번 복사해 (1,m) 행렬 생성 후 요소별 계산

6️⃣ 파이썬과 넘파이 벡터

Intro.

- 장점 : 넓은 표현성과 유연성 → 간단한 코드로 많은 것을 구현 가능
- 단점 : 브로드캐스팅 유연성으로 이상한 결과와 오류 발생

파이썬/넘파이 벡터

- a = np.random.randn(5) → a.shape (5,) : rank 1 array ⇒ 사용하지 않기
- a = np.random.randn(5,1) → a.shape (5,1) : column vector
- a = np.random.randn(1,5) → a.shape(1,5) : row vector

```
a = np.random.randn(5)  # 가우시안 분포를 따르는 난수 5개를 배열로 저장
print(a)
print(a.shape)  # (5,) : 랭크가 1인 배열, 행/열벡터가 아님
print(a.T)            # a와 같음
print(np.dot(a,a.T))  # 행렬이 아닌 상수값이 나옴

a = np.random.randn(5,1)
print(a)
print(a.shape)  # (5,1) : 열벡터
print(a.T)      # (1,5) : 행벡터
print(np.dot(a,a.T))  # 벡터의 외적 출력
```

- 코드에서 벡터의 차원을 확실히 알지 못할 때 assert( ) 함수 사용
    - `assert(a.shape == (5,1))`
- rank 1 array를 얻게 된 경우 reshape( ) 함수 사용

- `a = a.reshape(5,1)`

7️⃣ Jupyter/iPython Notebooks 가이드

- coursera에서 사용할 수 있는 주피터 창 소개

8️⃣ 로지스틱 회귀의 비용함수 설명

Logistic Regression Cost Function

- $\hat{y} = \sigma(w^T x + b)$ where $\sigma(z) = \frac{1}{1+e^{-z}}$
- y의 예측값 $\hat{y} = P(y = 1|x)$ → x가 주어졌을 때 y가 1일 확률인 $\hat{y}$ 반환
- y=1이면 $P(y|x) = \hat{y}$ / y=0이면 $P(y|x) = 1 - \hat{y}$
- 두 식을 하나로 합치면 → $P(y|x) = \hat{y}^y(1-\hat{y})^{(1-y)}$
- 로그함수 : 강한 단조증가 함수 → $\log P(y|x)$를 최대화하는 것은 $P(y|x)$를 최대화하는 것과 같음
- $\log P(y|x) = \hat{y}^y(1-\hat{y})^{(1-y)} = y\log\hat{y} + (1-y)\log(1-\hat{y}) = -L(\hat{y}, y)$
    - 로지스틱 회귀의 손실함수의 음수가 됨
    - 보통 학습 알고리즘 훈련은 확률을 높이려고 하지만 로지스틱 회귀에서는 손실함수를 최소화하고자 함
    - 손실함수를 최소화하는 것은 확률의 로그값을 최대화하는 것과 같음

Cost on m examples

- $P(\text{labels in training set}) = \Pi_{i=1}^{m} P(y^{(i)}|x^{(i)})$ (∵ iid)
- 양변에 로그를 취하면
  $\log P(\text{labels in training set}) = \sum_{i=1}^{m} \log P(y^{(i)}|x^{(i)}) = -\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$
- 최대 우도 추정(MLE) : 훈련 세트의 타깃 확률을 최대화해주는 매개변수를 찾는 것
  → $-L(\hat{y}^{(i)}, y^{(i)})$를 최대화하는 매개변수를 찾아야 함
- Cost(minimize) : $J(w, b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$
    - 비용함수를 최소화해야 하기 때문에 음수를 제거
    - 스케일을 맞추기 위해 1/m 비례 계수 추가

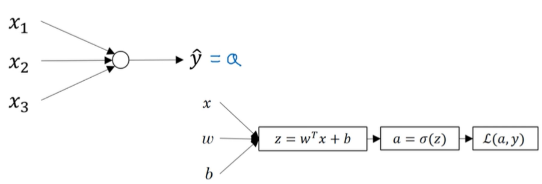
  ⇒ 비용함수를 최소화하기 위해 로지스틱 회귀 모델의 최대 우도 추정을 한 것 !

얕은 신경망 네트워크

1️⃣ 신경망 네트워크 개요

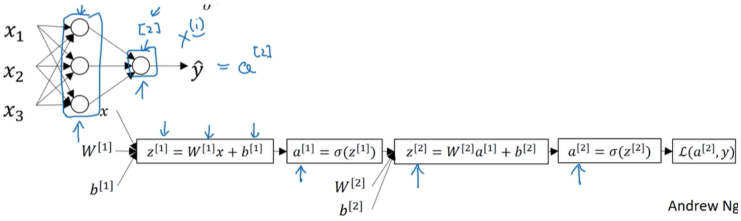
What is a Neural Network?

- 로지스틱 회귀 모델



    - 1 step) z 계산 2 step) a 계산

- 신경망 모델

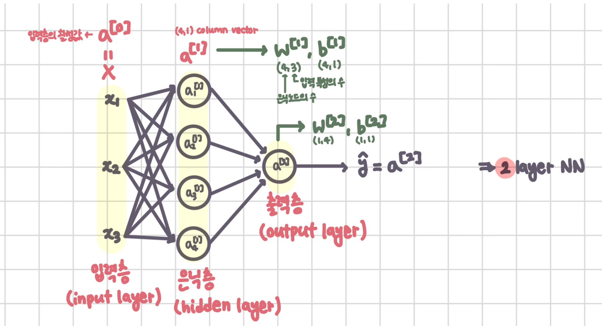


  - 위첨자 [1] : 층이라고 불리는 일련의 노드 값을 표현
  - z와 a를 여러 층에서 여러번 계산 → 마지막으로 손실 계산
  - 역방향 계산 수행

2️⃣ 신경망 네트워크의 구성 알아보기

Neural Network Representation

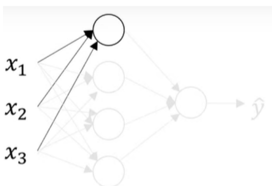
- 은닉층이 하나인 신경망 구조



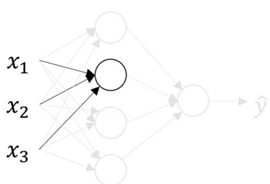
- 지도학습 : 훈련셋 입력값 x와 출력값 y로 구성
- 은닉층 실제 값은 훈련셋에 기록되어 있지 않음
- a : 활성값(activation), 신경망의 층들이 다음 층으로 전달해주는 값
- 신경망 층을 셀 때 입력층은 세지 않음
- 은닉층과 출력층은 연관된 매개변수 존재

3️⃣ 신경망 네트워크 출력의 계산

은닉층이 하는 계산



- 1 step) $z_1^{[1]} = w_1^{[1]T}x + b_1^{[1]}$
- 2 step) $a_1^{[1]} = \sigma(z_1^{[1]})$
  - 위첨자 : layer



- 1 step) $z_2^{[1]} = w_2^{[1]T}x + b_2^{[1]}$
- 2 step) $a_2^{[1]} = \sigma(z_2^{[1]})$

- 아래첨자 : node in layer

수식 벡터화

$$z^{[1]} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix} = \begin{bmatrix} w_1^{[1]T} \\ w_2^{[1]T} \\ w_3^{[1]T} \\ w_4^{[1]T} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix} = \begin{bmatrix} w_1^{[1]T}x + b_1^{[1]} \\ w_2^{[1]T}x + b_2^{[1]} \\ w_3^{[1]T}x + b_3^{[1]} \\ w_4^{[1]T}x + b_4^{[1]} \end{bmatrix}$$
$$z^{[1]} = W^{[1]}X + b^{[1]}$$

$$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix} = \sigma(z^{[1]}) = \sigma(W^{[1]}X + b^{[1]})$$

- layer 1 : $z^{[1]} = W^{[1]}a^{[0]} + b^{[1]},\ a^{[1]} = \sigma(z^{[1]})$
- layer 2 : $z^{[2]} = W^{[2]}a^{[1]} + b^{[2]},\ a^{[2]} = \sigma(z^{[2]})$

4️⃣ 많은 샘플에 대한 벡터화

Vectorizing across multiple examples

- $\mathrm{x}^{(1)} \rightarrow \hat{\mathrm{y}}^{(1)} = \mathrm{a}^{[2](1)}, \cdots, \mathrm{x}^{(\mathrm{m})} \rightarrow \hat{\mathrm{y}}^{(\mathrm{m})} = \mathrm{a}^{[2](\mathrm{m})}$
- $a^{[j](i)}$ → i번째 훈련 샘플, j번째 층을 의미

  for i=1 to m:

  $\mathrm{z}^{[1](\mathrm{i})} = \mathrm{W}^{[1]}\mathrm{x}^{(\mathrm{i})} + \mathrm{b}^{[1]}$

  $\mathrm{a}^{[1](\mathrm{i})} = \sigma(\mathrm{z}^{[1](\mathrm{i})})$

  $\mathrm{z}^{[2](\mathrm{i})} = \mathrm{W}^{[2]}\mathrm{a}^{[1](\mathrm{i})} + \mathrm{b}^{[2]}$

  $\mathrm{a}^{[2](\mathrm{i})} = \sigma(\mathrm{z}^{[2](\mathrm{i})})$

- $X = \begin{bmatrix} \mathrm{x}^{(1)} & \mathrm{x}^{(2)} & \cdots & \mathrm{x}^{(\mathrm{m})} \end{bmatrix}, (\mathrm{n_x}, \mathrm{m})$
- $Z^{[1]} = \begin{bmatrix} \mathrm{z}^{[1](1)} & \mathrm{z}^{[1](2)} & \cdots & \mathrm{z}^{[1](\mathrm{m})} \end{bmatrix}$
- $A^{[1]} = \begin{bmatrix} \mathrm{a}^{[1](1)} & \mathrm{a}^{[1](2)} & \cdots & \mathrm{a}^{[1](\mathrm{m})} \end{bmatrix}$
- 행렬 A의 행은 각 노드의 활성값, 열은 각 훈련샘플을 의미

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$
$$A^{[1]} = \sigma(Z^{[1]})$$
$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$
$$A^{[2]} = \sigma(Z^{[2]})$$

5️⃣ 벡터화 구현에 대한 설명

Justication for vectorized implementation

$$z^{[1](1)} = W^{[1]}x^{(1)} + \cancelto{0}{b^{[1]}},\quad z^{[1](2)} = W^{[1]}x^{(2)} + \cancelto{0}{b^{[1]}},\quad z^{[1](3)} = W^{[1]}x^{(3)} + \cancelto{0}{b^{[1]}}$$ 0으로 가정

$$W^{[1]} = \begin{bmatrix} — \\ — \\ \vdots \\ — \end{bmatrix},\quad W^{[1]}x^{(1)} = \underbrace{\begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}}_{\text{column vector}},\quad W^{[1]}x^{(2)} = \underbrace{\begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}}_{\text{column vector}},\quad W^{[1]}x^{(3)} = \underbrace{\begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}}_{\text{column vector}}$$

$$W^{[1]}\underbrace{\begin{bmatrix} x^{(1)} & x^{(2)} & x^{(3)} \end{bmatrix}}_{X} = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} = \begin{bmatrix} z^{[1](1)} & z^{[1](2)} & z^{[1](3)} \end{bmatrix} = z^{[1]} \quad \Rightarrow z^{[1]} = W^{[1]}X + b^{[1]}$$

↳ 훈련샘플을 열로 쌓으면

↳ 결과값도 열로 쌓임

$$A^{[1]} = \begin{bmatrix} a^{[1](1)} & a^{[1](2)} & a^{[1](3)} \end{bmatrix} = \begin{bmatrix} \sigma(z^{[1](1)}) & \sigma(z^{[1](2)}) & \sigma(z^{[1](3)}) \end{bmatrix} = \sigma(z^{[1]})$$