4주차

날짜	@2024년 4월 2일
과제	강의 요약 출석 퀴즈
세부내용	[딥러닝 1단계] 4-2. 얕은 신경망 네트워크 (활성화 함수~랜덤 초기화)

얕은 신경망 네트워크

6️⃣ 활성화 함수

Activation Function

- 은닉층과 출력층에서 어떤 활성화 함수를 사용할지 결정해야 함
- 활성값 계산 : $\mathrm{a}^{[1]} = \sigma(\mathrm{z}^{[1]})$ → 이 때 σ 대신 다른 함수 $\mathrm{g}(\mathrm{z}^{[1]})$ 사용 가능
- g는 시그모이드 함수가 아닌 다른 비선형 함수
- $\mathrm{g}^{[1]}(\mathrm{z}^{[1]})$: 층마다 다른 활성화 함수를 사용함을 나타냄

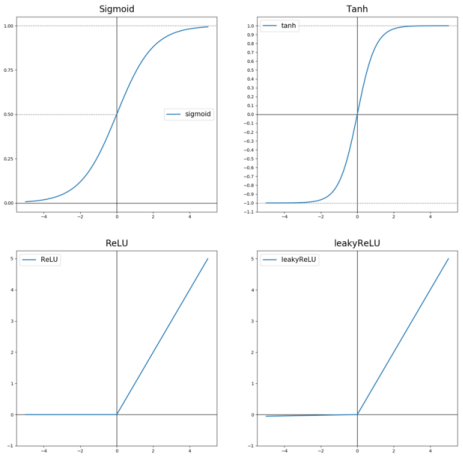
활성화 함수 종류

- 시그모이드 함수 : $\frac{1}{1+\mathrm{e}^{-\mathrm{z}}}$ → 0~1 사이 값
  - 이진분류의 출력층에서는 tanh 보다 좋을 수 있음

    → y가 0 또는 1이라면 -1~1 보다 0~1 사이로 출력하는 것이 더 좋기 때문
  - z가 굉장히 크거나 작으면 함수의 도함수가 0에 수렴 → 기울기 소실 문제
- 하이퍼볼릭 탄젠트 함수(tanh) : $\frac{\mathrm{e}^{\mathrm{z}}-\mathrm{e}^{-\mathrm{z}}}{\mathrm{e}^{\mathrm{z}}+\mathrm{e}^{-\mathrm{z}}}$ → -1~1 사이 값
  - 시그모이드 함수와 모양은 비슷하지만 원점을 지나고 비율이 달라짐
  - 시그모이드 함수보다 좋음 → 평균값이 0에 더 가깝기 때문
  - 평균값이 0에 더 가깝기 때문에 시그모이드 함수보다 좋음

    → 학습 알고리즘 훈련 시 평균값의 중심을 0으로 할 때가 있음

    → 데이터 중심을 0.5 대신 0으로 하여 다음 층의 학습을 더 쉽게 해줌
  - 기울기 소실 문제
- ReLU 함수 : $\max(0, \mathrm{z})$ → 기울기 소실 문제 해결
  - z가 음수일 때 도함수가 0인 문제
- Leaky ReLU : $\max(0.01\mathrm{z}, \mathrm{z})$
  - z가 음수일 때 도함수 0 대신 약간의 기울기를 줌
  - 0.01과 같은 값은 지정해줄 수 있음



활성화 함수 선택

- 이진 분류의 출력층 → 시그모이드 함수
- 대부분 은닉층 → ReLU가 활성화 함수 기본값으로 많이 사용됨

7️⃣ 왜 비선형 활성화 함수를 써야 할까요?

활성화 함수가 선형인 경우

- 입력값과 출력값이 같은 항등 함수를 (선형) 활성화 함수로 사용

$$\mathbf{a}^{[1]} = \mathbf{z}^{[1]} = \mathbf{W}^{[1]}\mathbf{x} + \mathbf{b}^{[1]}$$
$$\mathbf{a}^{[2]} = \mathbf{z}^{[2]} = \mathbf{W}^{[2]}\mathbf{a}^{[1]} + \mathbf{b}^{[2]}$$

- $\mathbf{a}^{[1]}$을 $\mathbf{a}^{[2]}$에 대입

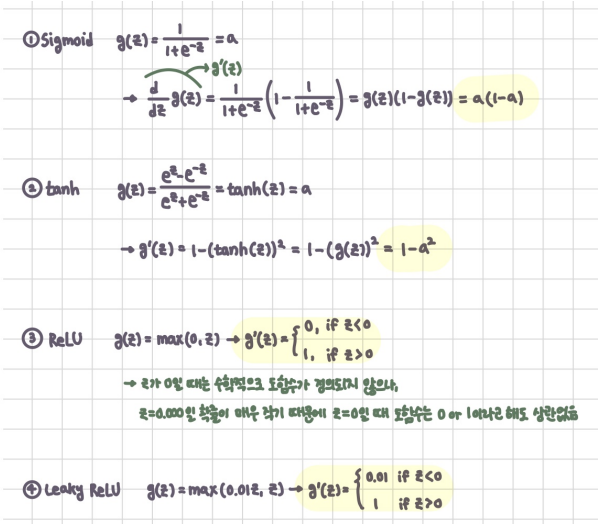
$$\mathbf{a}^{[2]} = \mathbf{W}^{[2]}(\mathbf{W}^{[1]}\mathbf{x} + \mathbf{b}^{[1]}) + \mathbf{b}^{[2]}$$
$$= (\mathbf{W}^{[2]}\mathbf{W}^{[1]})\mathbf{x} + (\mathbf{W}^{[2]} + \mathbf{b}^{[1]} + \mathbf{b}^{[2]})$$
$$= \mathbf{W}'\mathbf{x} + \mathbf{b}'$$

→ 선형 or 항등 함수를 활성화 함수로 사용하면 신경망은 입력의 선형식만을 출력하게 됨

→ 층이 얼마나 많든 간에 선형식만을 출력하기 때문에 은닉층이 없는 것과 다름 없음

⇒ 신경망의 은닉층에서는 비선형 활성화 함수를 사용 !

8️⃣ 활성화 함수의 미분



9️⃣ 신경망 네트워크와 경사 하강법

단일 신경망에서의 경사 하강법 (이진분류)

- 파라미터 : $\mathrm{W}^{[1]}(\mathrm{n}^{[1]}, \mathrm{n}^{[0]}),\ \mathrm{b}^{[1]}(\mathrm{n}^{[1]}, 1),\ \mathrm{W}^{[2]}(\mathrm{n}^{[2]}, \mathrm{n}^{[1]}),\ \mathrm{b}^{[2]}(n^{[2]}, 1)$
- $\mathrm{n_x} = \mathrm{n}^{[0]}(\text{input unit}),\ \mathrm{n}^{[1]}(\text{hidden unit}),\ \mathrm{n}^{[2]}(\text{output unit}) = 1$
- 비용 함수 : $\mathrm{J}(\mathrm{W}, \mathrm{b}) = \frac{1}{\mathrm{m}} \sum \mathrm{L}(\mathrm{a}^{[2]}, \mathrm{y})$
- 경사하강법 :

  ① 변수를 초기화

  ② 예측값 계산($\hat{\mathrm{y}}^{(\mathrm{i})}$)

  ③ 도함수 계산 : $\mathrm{dW}^{[1]} = \frac{\mathrm{dJ}}{\mathrm{dW}^{[1]}},\ \mathrm{db}^{[1]} = \frac{\mathrm{dJ}}{\mathrm{db}^{[1]}}, \mathrm{dW}^{[2]} = \frac{\mathrm{dJ}}{\mathrm{dW}^{[2]}}, \mathrm{db}^{[2]} = \frac{\mathrm{dJ}}{\mathrm{db}^{[2]}}$

  ④ 업데이트

  $$\mathrm{W}^{[1]} = \mathrm{W}^{[1]} - \alpha \cdot \mathrm{dW}^{[1]}$$
  $$\mathrm{b}^{[1]} = \mathrm{b}^{[1]} - \alpha \cdot \mathrm{db}^{[1]}$$
  $$\mathrm{W}^{[2]} = \mathrm{W}^{[2]} - \alpha \cdot \mathrm{dW}^{[2]}$$
  $$\mathrm{b}^{[2]} = \mathrm{b}^{[2]} - \alpha \cdot \mathrm{db}^{[2]}$$

  ⑤ 변수들이 수렴할 때까지 ②~④를 반복

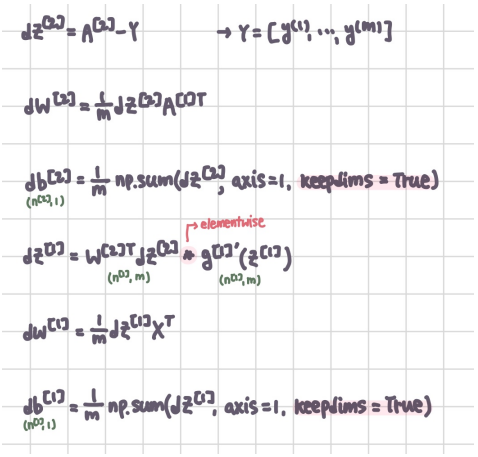
Formulas for Computing Derivatives

- Forward Propogation

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

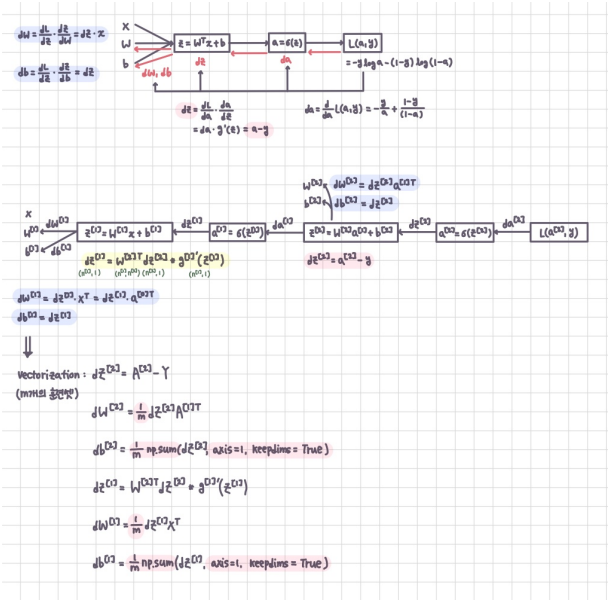
  - 이진분류이므로 활성화함수로 시그모이드 함수 사용

- Back Propogation



- `keepdims=True` : 1차원 배열을 출력하지 않도록 함 → (n, 1)

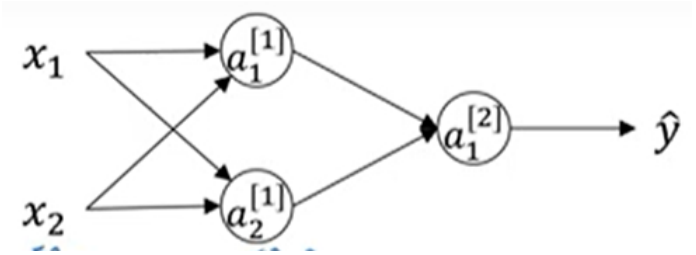
🔟 역전파에 대한 이해



1️ 랜덤 초기화

- 로지스틱 회귀 : 파라미터를 모두 0으로 초기화해도 괜찮음
- 신경망 : 파라미터를 잘 초기화해야 함

What happens if you initialize weight to zero?



- $\mathrm{n}^{[0]} = 2,\ \mathrm{n}^{[1]} = 2$인 신경망
- 가중치를 모두 0으로 초기화하면? $W^{[1]} = [[0,0],[0,0]]$
  - $\mathrm{a}_1^{[1]} = \mathrm{a}_2^{[1]}$: 활성값이 모두 같아짐
  - $\mathrm{dz}_1^{[1]} = \mathrm{dz}_2^{[1]}$: 역전파할 때 도함수 값이 모두 같아짐
  - $W^{[2]} = [0,0]$

    ⇒ a_1 유닛과 a_2 유닛이 완전히 같아짐 → 완전 대칭

  → 두 은닉 유닛이 항상 출력 유닛에 같은 영향을 주게 됨

  → 신경망이 얼마나 많은 훈련하는 지와 상관없이 항상 같은 유닛이 됨

  → 은닉층이 하나인 신경망과 같음 ⇒ 랜덤 초기화 필요

Random Initialization

- `w1 = np.random.randn((2,2)) * 0.01`
- `b1 = np.zeros((2,1))` → b는 0으로 초기화해도 괜찮음 ⇒ 대칭 회피
- `w2 = np.random.randn((1,2)) * 0.01`
- `b2 = np.zeros((1,1))`
- 가중치 초기값을 매우 작은 값으로 설정하는 것이 좋음
  - 가중치가 너무 큰 경우 활성값을 계산하면 기울기 소실 문제 발생 (학습 속도 느려짐)