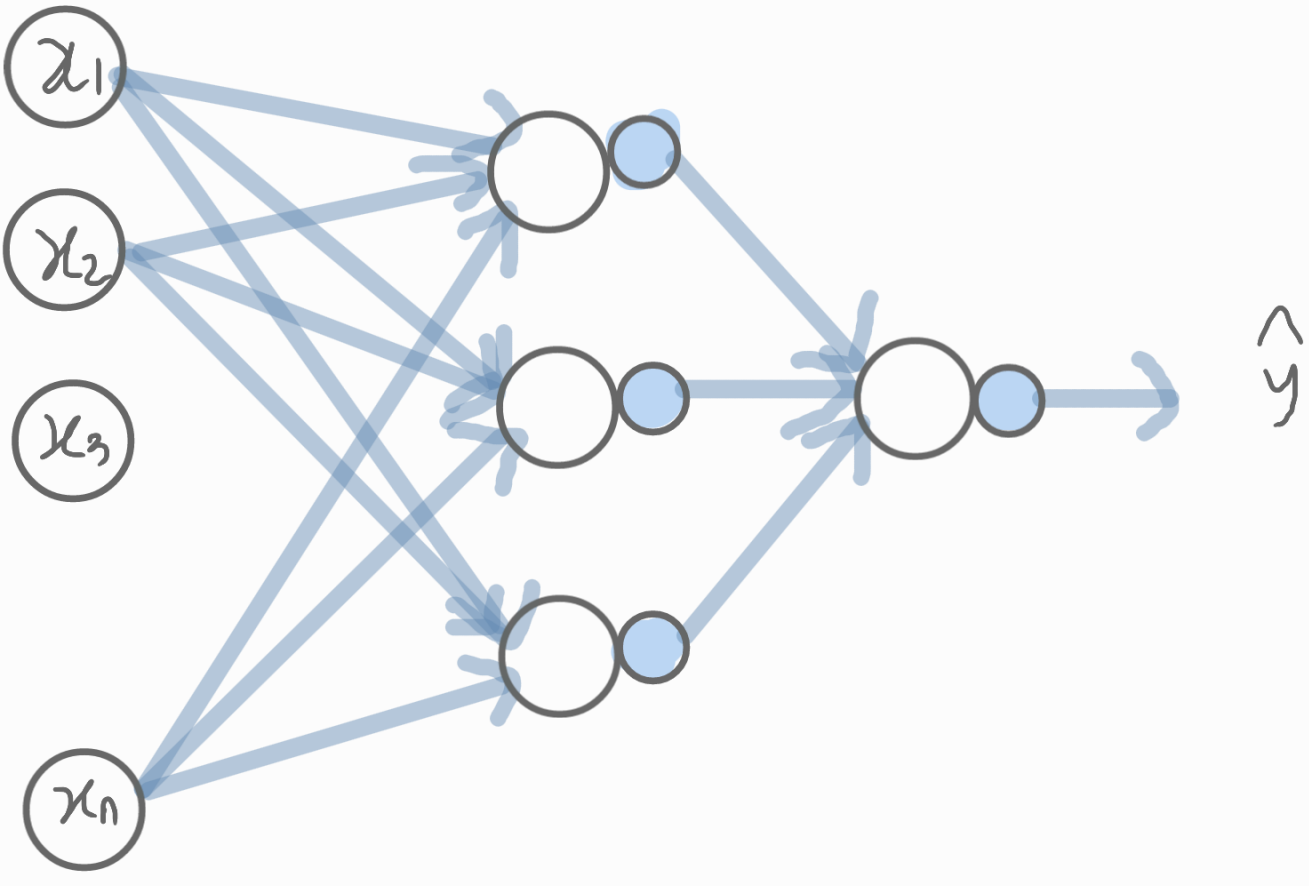
07-1
여러 개의 이미지를 분류하는 다층 신경망

다중 분류를 위한 신경망과 이진 분류를 위한 신경망의 차이점은 무엇일까?

- 이진 분류를 위한 다층 신경망의 구조

  이진 분류를 위해 출력층에는 1개의 뉴런만 두었다.

활성화 함수



<입력층> <은닉층> <출력층>

출력층의 활성화 값 (=확률값) 이

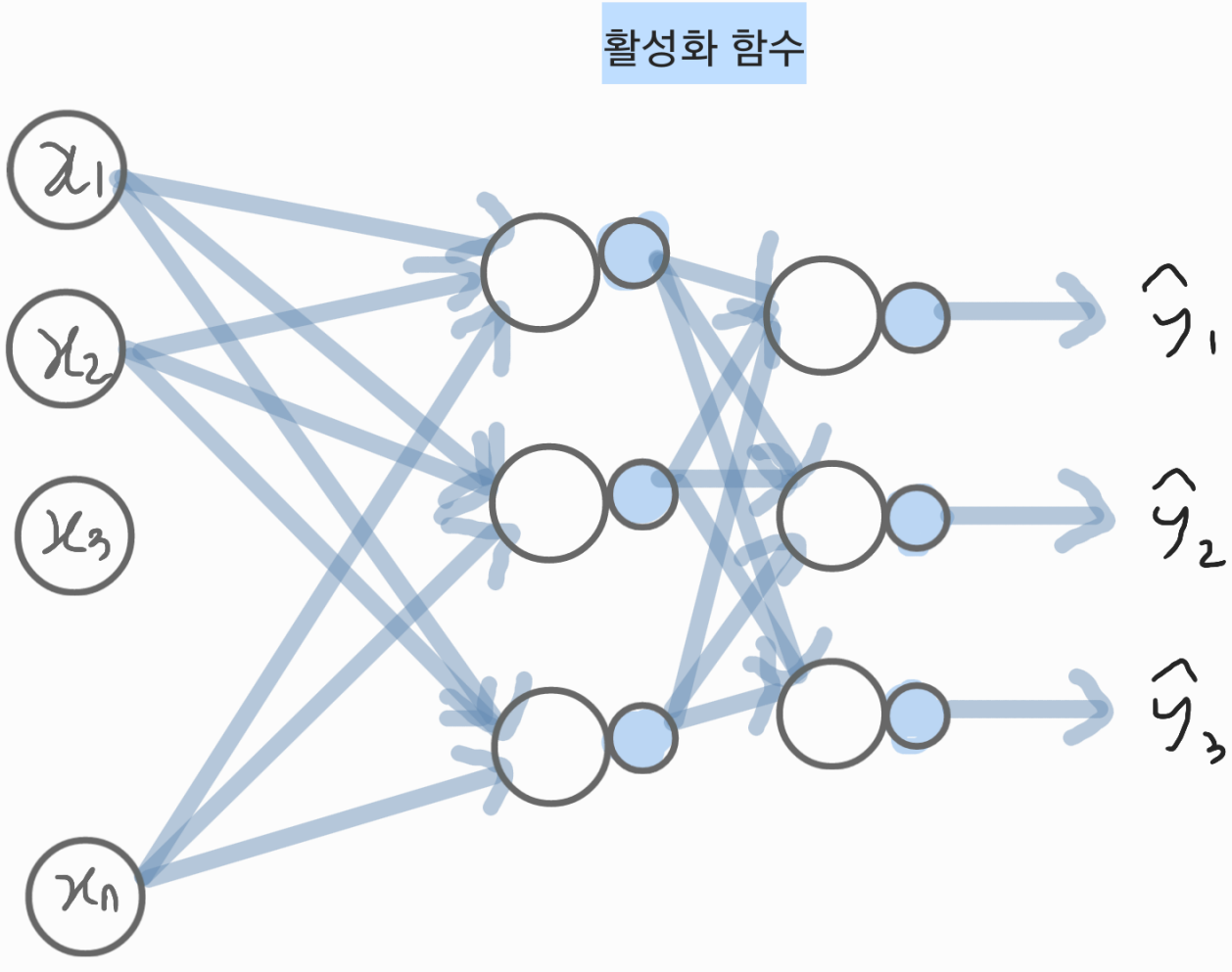
y_hat > 0.5 -> 양성 클래스

y_hat < 0.5 -> 음성 클래스

- 다중 분류를 위한 다층 신경망의 구조

<u>출력층에 뉴런이 여러 개 존재!</u>

나머지는 이진 분류의 구조와 같다.



<입력층> <은닉층> <출력층>

이진 분류는 양성 클래스에 대한 확률 y_hat 하나만 출력.
이진 분류 신경망은 출력층에 뉴런을 1개만 배치합니다.
(분류할 클래스가 True, False 의 2개 뿐이므로)

다중 분류는 각 클래스에 대한 확률값을 출력.
이를 위해 다중 분류 신경망은 출력층에 분류할 클래스 개수(=c)만큼의 뉴런을 배치합니다.

다중 분류에서 출력층의 활성화값은 주어진 입력 데이터를 그 정도의 확신을 가지고 해당 클래스라고 예측한다고 해석할 수 있습니다.

- 다중 분류의 문제점

활성화 출력의 합이 1이 아니라면 공정하게 비교하기 어렵습니다.

-> 이는 소프트맥스 함수(softmax function)을 사용하여 해결할 수 있습니다.

- 소프트맥스 함수

출력층의 출력 강도를 정규화하는 함수.
: 전체 출력값의 합을 1로 만든다 는 의미.

이렇게 하면 출력값들을 확률로 생각할 수 있게 됨.

소프트맥스 함수에서는 출력층의 각 뉴런에서 결과로 얻은 선형 출력값 zi를 계산한다.
이 z 값들을 소프트맥스 함수 공식에 대입하면
각각 그에 대응하는 정규화된 y_hat 값들을 얻을 수 있다.
(y_hat_1 + y_hat_2 + ... + y_hat_i + ... + y_hat_c = 1)

=> 소프트맥스 함수를 이용하면 확률의 합을 1로 만들어 비교가 용이해진다!
다중 분류에서 출력층을 통과한 값들은 소프트맥스 함수를 거치며 적절한 확률값으로 변한다.

- 다중 분류에서의 손실 함수

다중 분류에서는 로지스틱 손실 함수의 일반화 버전인 크로스 엔트로피(cross entropy) 손실 함수를 사용한다.

로지스틱 손실 함수는 크로스 엔트로피 손실 함수의 이진 분류 버전이다.

크로스 엔트로피 손실 함수와 로지스틱 손실 함수를 쉽게 비교할 수 있도록 로지스틱 손실 함수를 타깃이 양성 클래스인 경우와 음성 클래스인 경우 로 나누어 정리해보자.

(1-a) 를 음성 클래스의 활성화 출력이라 생각하면,
로지스틱 손실 함수의 식과 크로스 엔트로피 손실 함수의 식이 같다고 볼 수 있다.

- 경사 하강법을 사용하기 위해 크로스 엔트로피 손실 함수 미분하기

출력값 zi에 대한 크로스 엔트로피 손실 함수 L의 미분은 연쇄 법칙을 적용하여 구할 수 있다.
cf) 시그모이드 함수의 연쇄 법칙과 식이 다르다!

203 p 상단 '신경망에 각 도함수가 적용된 모습' 그림 참고
출력층의 결과 : zi -> 소프트맥스 함수 출력값 : ai

분류 문제에서 타깃의 합은
정답 타깃이 1이고 나머지 타깃은 0이므로 항상 1이다.

L의 zi 에 대한 gradient = -(yi - ai)

벡터 z에 대해 정리하면 L의 미분 결과는

-(y-a)

와 같습니다.

이것은 로지스틱 손실 함수의 미분과 정확히 같으므로,
크로스 엔트로피 손실 함수를 역전파에 사용하기 위해
코드로 따로 구현할 필요가 없다!

<다중 분류 신경망 구현하기>

다중 분류의 경사 하강법 알고리즘은
이진 분류의 경사 하강법 알고리즘과 원리는 같고
소프트맥스 함수가 추가된 점만 다르므로,
이 부분만 수정하면 된다.

1) 소프트맥스 함수 추가하기

지금까지는 활성화 함수로 시그모이드 함수만 사용했지만,
다중 분류는 마지막 출력층에 소프트맥스 함수를 사용해야 하므로 은닉층과 출력층에 서로 다른 활성화 함수를 적용한다.
기존에 activation() 메서드 였던 부분 중
은닉층의 활성화 함수는 sigmoid() 메서드 로 이름을 변경해주고, 출력층의 활성화 함수로 사용할 softmax() 메서드를 추가해줘야 한다.

2) 정방향 계산하기

3) 가중치 초기화하기

이진 분류에서는 출력층의 뉴런이 1개
-> 가중치의 크기 : (은닉층의 뉴런 개수, 1)

다중 분류에서는 출력층의 뉴런이 2개 이상
-> 가중치의 크기 : (은닉층의 뉴런 개수, 클래스 개수)
절편의 크기는 클래스 개수에 따라 지정합니다.

4) fit 메서드 수정하기

5) training() 메서드 수정하기

training() 메서드에서 사용하는 출력층의 활성화 함수를
activation() 메서드 -> softmax() 메서드
로 변경

6) predict() 메서드 수정하기

7) score() 메서드 수정하기

- 의류 이미지를 분류합니다

이번 실습부터는 패션 MNIST 데이터를 텐서플로에서 불러와 사용합니다.

- 의류 데이터를 준비합니다

패션 MNIST 데이터 세트는 이미지이고,
샘플의 양도 굉장히 많습니다.

훈련 데이터의 입력은 28x28 크기의 흑백 이미지
60,000 장 .
훈련 데이터의 타깃은 각 이미지를 분류한 타깃값이 들어 있는 크기 60,000 의 1차원 배열 .
이 배열에는 0~9까지의 정수로 이루어진 클래스 레이블이 들어 있다. : 패션 아이템의 카테고리를 나타낸다.

훈련 데이터를 적절하게 훈련 세트와 검증 세트로 나누고
각각 적절하게 정규화한다.

우리가 구현한 MultiClassNetwork 클래스는
샘플들이 1차원 배열이길 기대하지만,
훈련 세트와 검증 세트의 샘플들은 2차원 배열이다.
따라서 훈련 세트와 검증 세트를 1차원 배열로 차원 변경 해주어야 한다.

- 타깃 데이터를 준비하고 다중 분류 신경망을 훈련합니다

패션 MNIST 의 입력 데이터는 10개의 클래스로 구성되어 있으므로 출력 뉴런의 개수는 10개여야 한다.
신경망에 한 샘플이 입력으로 들어올 때마다
10개의 뉴런이 10개의 확률값을 출력해야 하는데,
타깃 세트의 값들은 0~9 사이의 정수값 1개로
10개의 출력 뉴런에 대응되지 않는다.
-> 이 타깃 데이터를 출력 뉴런의 개수에 맞게 변형해야 함.

원-핫 인코딩(one-hot encoding)
: 타깃의 값(0~9 사이의 정수)에 해당하는 원소는 1,
나머지 원소는 모두 0으로 하는 배열을 만든다.
(배열의 index=0,1,2,...,8,9)

-> 타깃값을 가지고 있는 1차원 정수 배열을 원-핫 인코딩 하면 2차원 배열이 만들어진다.