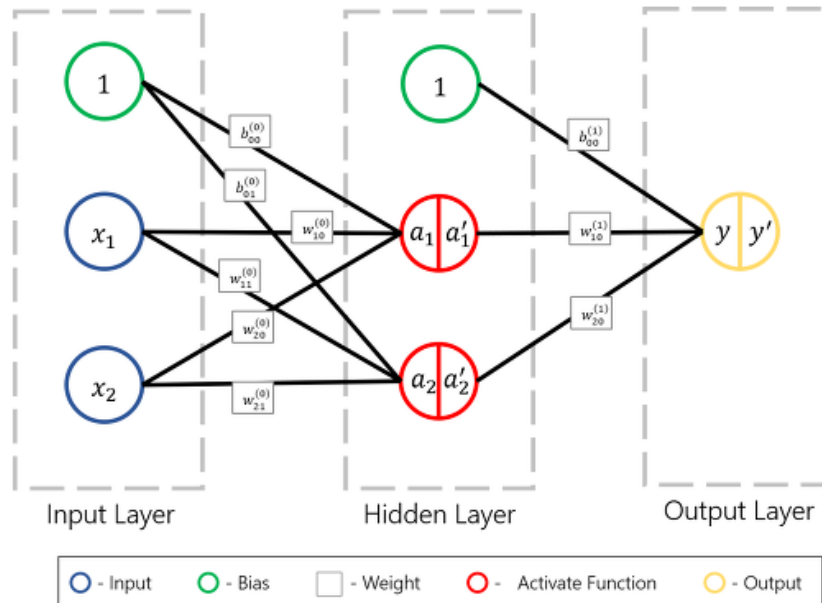


여러개의 Perceptron - Multi Layer Perceptron



- Multi Layer Perceptron (MLP) 의 그림 입니다. 두 개의 Perceptron 을 한 층에 구성하고 그 두 개의 Perceptron 을 연결하는 새로운 Perceptron 을 구성한 사례입니다. 위의 XOR 게이트의 모습과 동일하게 구성해본 모습입니다.
- 처음의 값 부분을 Input Layer, 결과가 출력되는 부분을 Output Layer, 그리고 내부의 층들을 Hidden Layer 라고 부릅니다. 그리고 Hidden Layer 가 3층 이상 되면 **Deep Neural Network (DNN)** 이라고 부릅니다. **딥러닝** 은 Hidden Layer 가 3개 이상인 인공 신경망을 말하는 것입니다.

- 자유투를 연습하는 영상입니다. 맨 앞의 분은 학습을 마치셔서 던지는 족족 샷을 성공하지만 뒤 쪽의 분들은 실패하고 다시 던지는 과정을 반복하고 있습니다.
- 던진 공이 골대의 좌측으로 가서 튕겨나오면 다시 던질 때는 조금 우측으로, 너무 낮게 던졌으면 조금 더 높게 던지면서 어떻게 던져야 샷을 성공하는지를 배워나갑니다.
- 이것을 MLP 학습과정에 적용해보면 어떻게 될까요?
- 만든 모델이 계산한 값이 정답보다 작으면 좀 더 크게 나올 수 있게 parameter 들을 수정하고, 정답보다 크면 좀 더 작게 나올 수 있게 parameter 들을 수정하는 과정. 다시말해 계산 값과 정답의 차이를 작게 만들어 나가는 과정이 학습과정이라고 할 수 있습니다. 이 과정을 수식으로 표현해보면 다음과 같습니다.

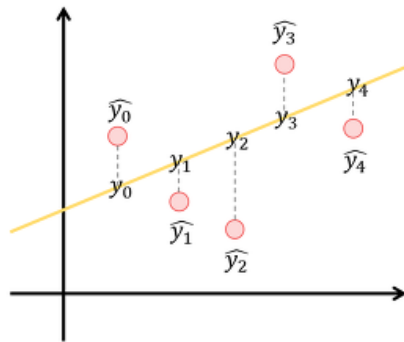
weight, bias

$$w^{new} = w - \gamma \nabla C(x)$$

차이를 계산하는 방법 - Cost Function

- 차이를 계산하는데도 방법이 있습니다. 단순히 차이를 구할수도 있고, 차이들을 제곱해서 평균내는 방법도 있고, 더 복잡한 수식을 사용하여 구하는 방법도 있습니다만 본 포스트에서는 가장 많이 사용되는 2가지 Function 에 대해 살펴보겠습니다.

① 가장 기본적인 오차함수 : MSE

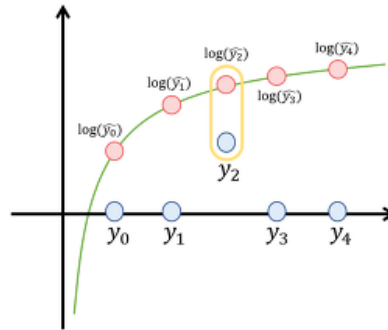


$$\text{Minimum Squared Error} = \frac{1}{n} \sum_i (\hat{y}_i - y_i)^2$$

각 정답과 예상값의 차이를 제곱해서 평균내는 방법


- 가장 기본적인 오차함수 입니다. 최소제곱법이라고도 불리는 Minimum Squared Error (MSE) 는 가장 직관적이며 벌써 하고 계신 방법일 수도 있습니다. 각 값들의 차이를 제곱해서 평균을 내는 방법입니다. 제곱하지 않으면 부호가 존재하기 때문에 차이가 상쇄될 수 있습니다. 오차를 제곱해서 합하면 이런 왜곡을 방지할 수 있고 더 큰 차이일 수록 더 크게 반영되는 장점 또한 가지고 있습니다.

② 가장 기본적인 오차함수 : ACE



$$\text{Averaged Cross Entropy Error} = -\frac{1}{n} \sum_i y_i \log(\hat{y}_i)$$

정답이 One-hot 인코딩 되어 있는 경우에만 사용가능

- 두 번째로 알아 볼 Cost Function 은 Averaged Cross Entropy Error 입니다. 이 함수는 분류 문제에 많이 사용하는 방법입니다. 분류 문제의 경우 정답이 One-hot 인코딩 되어 있는 경우가 대부분입니다.
-  One-hot 인코딩이란 하나의 값만 1 이고 나머지는 0 인 표기방법인데, 예를 들어 개와 고양이를 표기할 때 개는 [1, 0], 고양이는 [0, 1] 로 표기하는 방법입니다.
- 따라서 정답은 0 과 1 로 구성되어 있고 계산값은 log 함수에 넣습니다. 만약 정답이 0 이라면 $y \cdot \log(\hat{y})$ 값은 0 일 것입니다. 또한 정답이 1 이라면 오차는 $\log(\hat{y})$ 이고, 계산값인 \hat{y} 이 정답인 1에 가까워 질 수록 $\log(1) = 0$ 에 가까워 지기 때문에 전체 오차는 0으로 근접하는 것을 알 수 있습니다. $\log(0.5)$ 처럼 1보다 작은 수에 log 함수를 취하면 음수가 나오기 때문에 전체 식에 -1 을 곱해주는 것 또한 알 수 있습니다.

그래서 이와 같이 오차 함수가 이항 분류를 위해서 가지는 경우의 수 두가지에 해당하는 식을 표현 하자면,

$$Cost(h_{\theta}(x), y) = \begin{cases} -\log(h_{\theta}(x)) & \text{if } y = 1 \\ -\log(1 - h_{\theta}(x)) & \text{if } y = 0 \end{cases}$$

아래와 같이 나타낼 수 있는데 오차 함수를 계산하는데 있어서 조건을 항상 계산하는 것은 문제가 있다.
따라서 식을 변형하면, 다음과 같이 나타낼 수 있다.

$$Cost(h_{\theta}(x), y) = -y\log(h_{\theta}(x)) - (1 - y)\log(1 - h_{\theta}(x))$$

이를 코드로 표현하자면, 이와 같이 오차함수를 정의할 수 있고 이에 따라 cost를 최소화 하도록 경사 하강법을 적용한다.
그리고 그렇게 시그모이드 함수를 통과 시킨 우리의 가설 즉, 예측 값을 0 또는 1로 캐스팅하고 실제 데이터(Y)와 일치하는지 정확도를 계산한다.