K-Digital Training 스마트 팩토리 3기

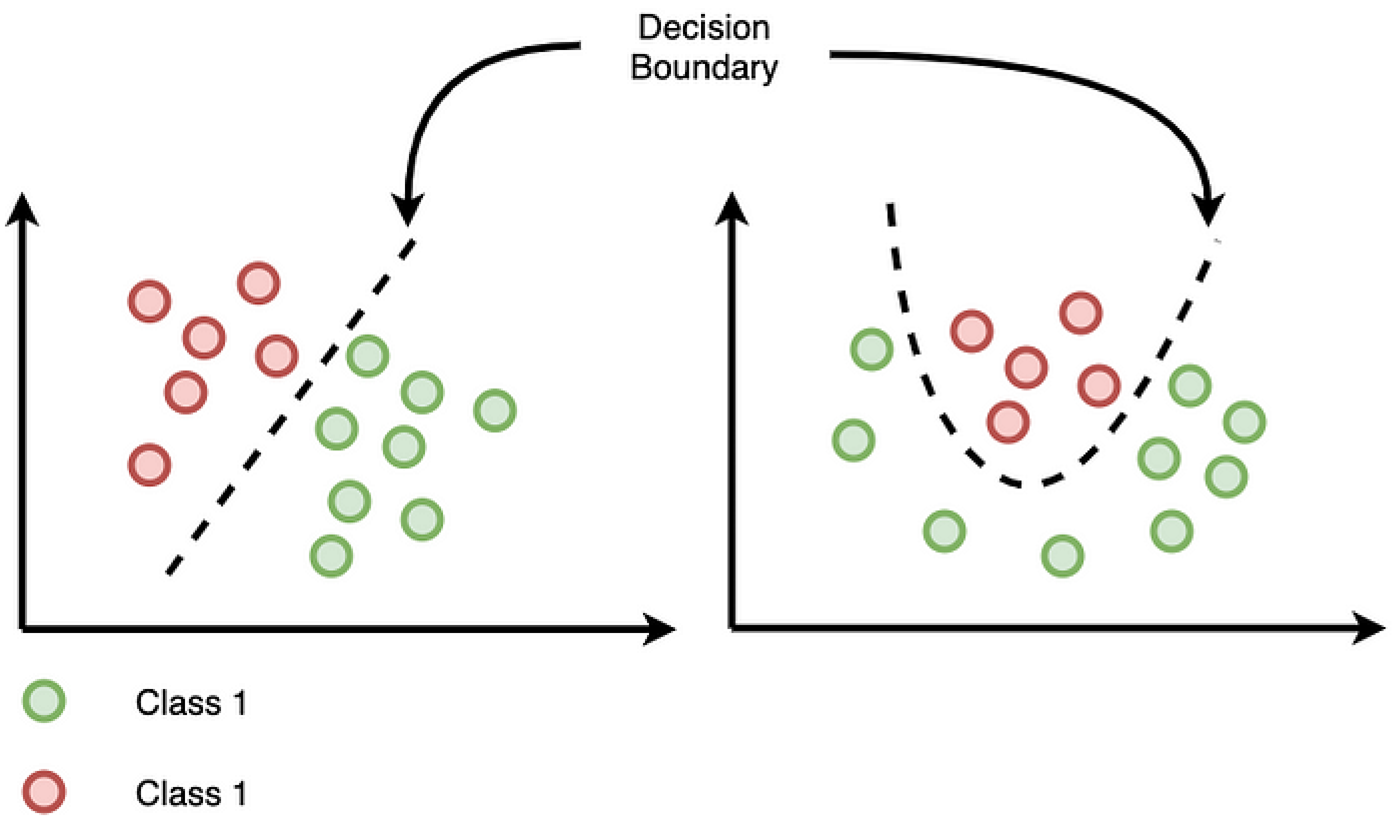
분류(Classification)

분류(Classification)

- 분류(Classification)는 머신러닝의 한 분야로, 입력 데이터를 미리 정해진 카테고리로 구분하는 알고리즘
- 분류 알고리즘의 예
  - 로지스틱 회귀, 서포트 벡터 머신, 나이브 베이즈 등

분류(Classification)

- 분류란 Decision Boundary(최적 경계선)을 찾는것



분류 알고리즘 종류

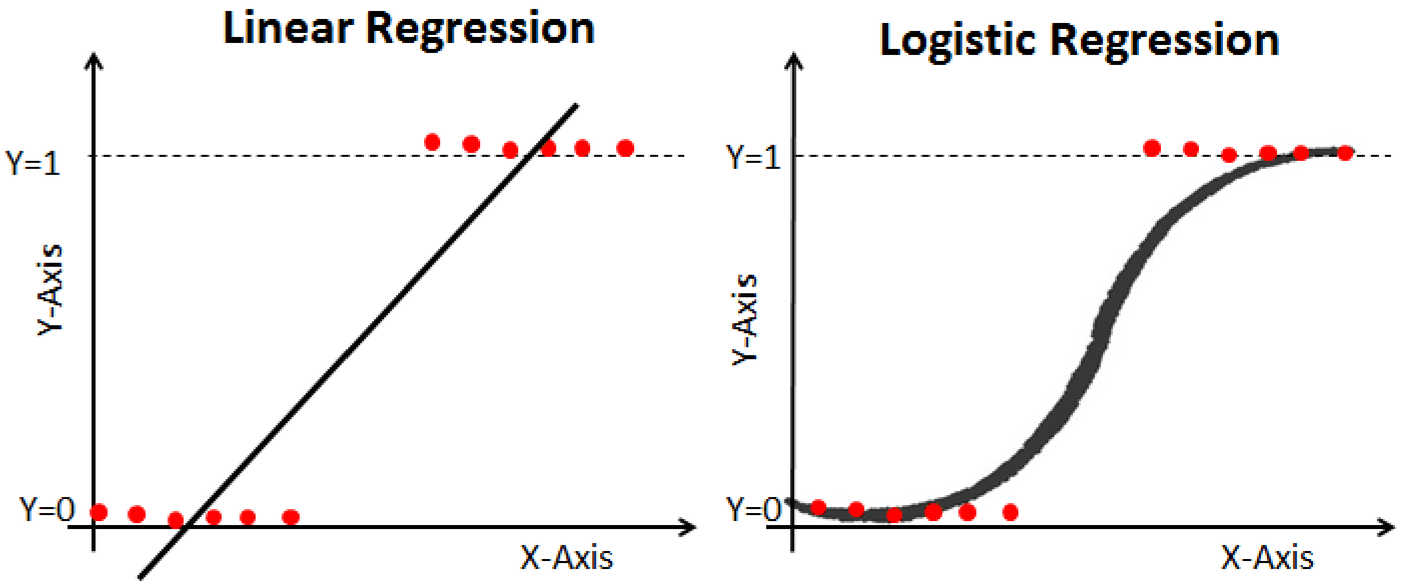
- KNN(K-nearest neighbor)
  - 다양한 레이블의 데이터 중에서, 자신과 가장 가까운 데이터를 찾아 자신의 레이블을 결정
- Decision Tree
  - 의사 결정 트리
- Random Forest
  - 의사 결정 트리를 여러 개 모은 것
- 나이브 베이즈(Naïve Bayes)
  - 확률을 이용하여 분류

분류 알고리즘 종류

- SVM(Support Vector Machine)
  - 이항분류에 많이 사용
  - 최대한 두 그룹에서 멀리 떨어져 있는 경계선을 구하는 알고리즘
- 로지스틱 회귀
  - 시그모이드 함수의 출력을 확률로 취급
  - 선형 분리 가능한 데이터에서 사용
- 신경망
  - 딥러닝의 기본 구조

선형 회귀 vs 로지스틱 회귀

- 선형회귀는 연속 출력을 제공하지만 로지스틱 회귀는 일정한 출력을 제공
- 선형회귀는 Threshold 값을 벗어날수록 오차가 커지는 문제



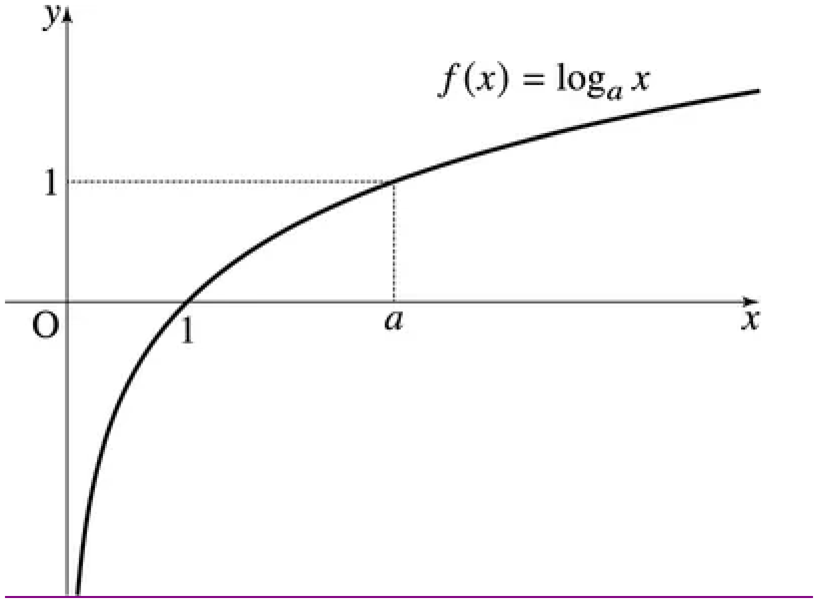
출처:https://www.datacamp.com/tutorial/understanding-logistic-regression-python

오즈(Odds)

- 성공(y=1) 확률이 실패(y=0) 확률에 비해 몇 배 더 높은가를 나타냄
- odds = $p/(1-p)$
  - p = 성공 확률 (p(y=1 | x))

로짓 변환(logit)

- 오즈에 로그를 취한 함수 형태
- 입력값 (p) 의 범위가 [0,1] 일 때, [−∞, +∞]
- $logit(p) = \log(odds) = \log(p/(1-p))$



로지스틱 함수(logistic function)

- 로짓 변환의 역함수로 해석
- $\log(p/(1-p))$ = Wx + b = L
- p = $e^{-L}/(e^{-L}+1) = 1/(1+e^{-L})$

$$p(x) = \frac{1}{1+e^{-(Wx+b)}}$$

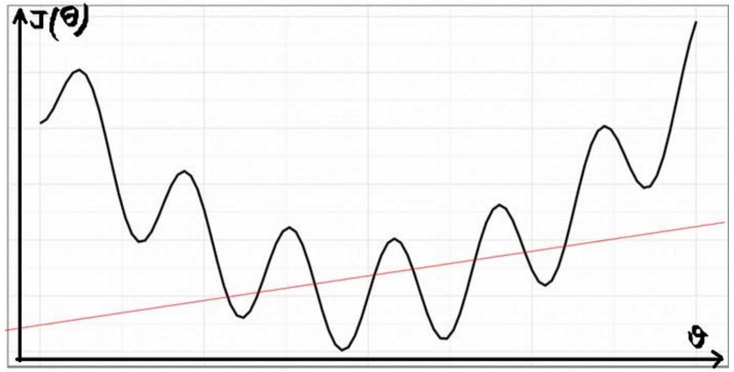
- x 데이터가 주어졌을 때 성공확률을 예측하는 Logistic Regression 은 결국 Sigmoid함수의 W와 b를 찾는 문제가 된다.

로지스틱 회귀의 cost function

- Sigmoid function 의 함수식

$$f(x) = \frac{1}{1+e^{-(x)}}$$

- 이를 cost function(MSE) 에 대입하여 그래프로 표현하면 오른쪽과 같다.(Non-convex)



- 따라서 새로운 cost function 이 필요하다

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

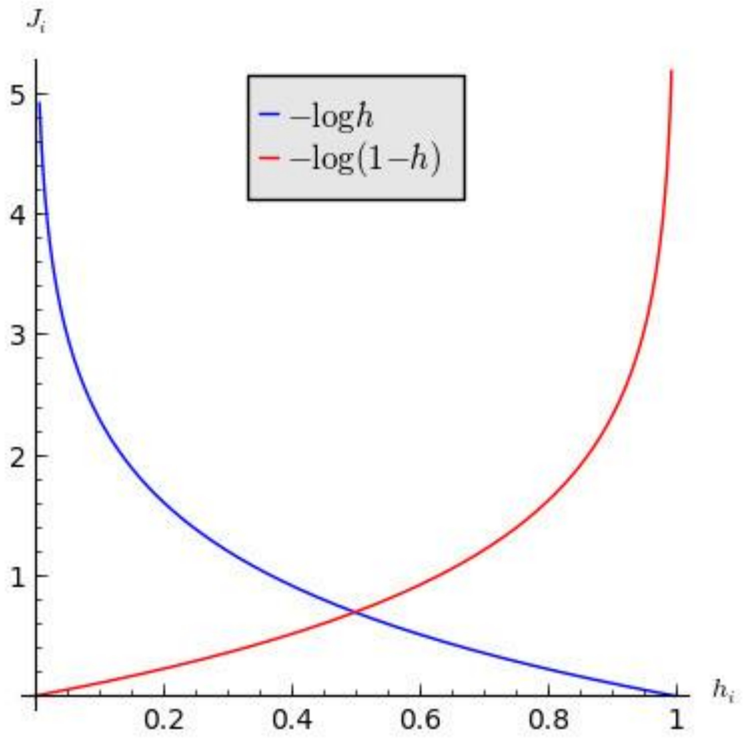
로지스틱 회귀의 cost function

- 이전 슬라이드 식의 그래프는 오른쪽과 같다

- 이를 하나의 식으로 표현하면

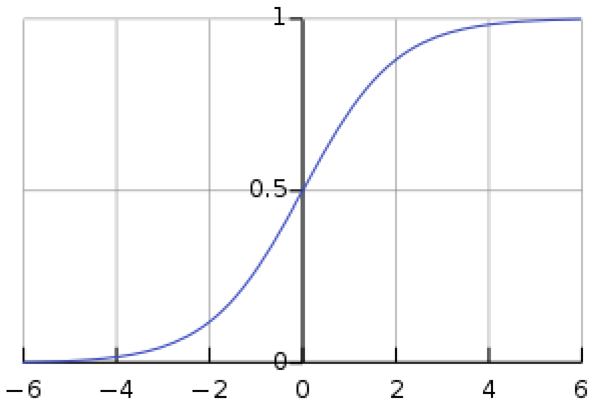
$$J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x), y)$$

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



이항 분류(Binary Classification)

- 2개의 Label을 갖는 데이터가 들어왔을 때, 0 또는 1로 분류를 하는 것
- 활성화 함수는 주로 Sigmoid(S자모양) 함수를 사용
- Sigmoid 대신 Softmax를 사용하는 것이 가능
- 오른쪽 그림은 Sigmoid 함수의 예시인 로지스틱 함수



다항 분류(Multi Classification)

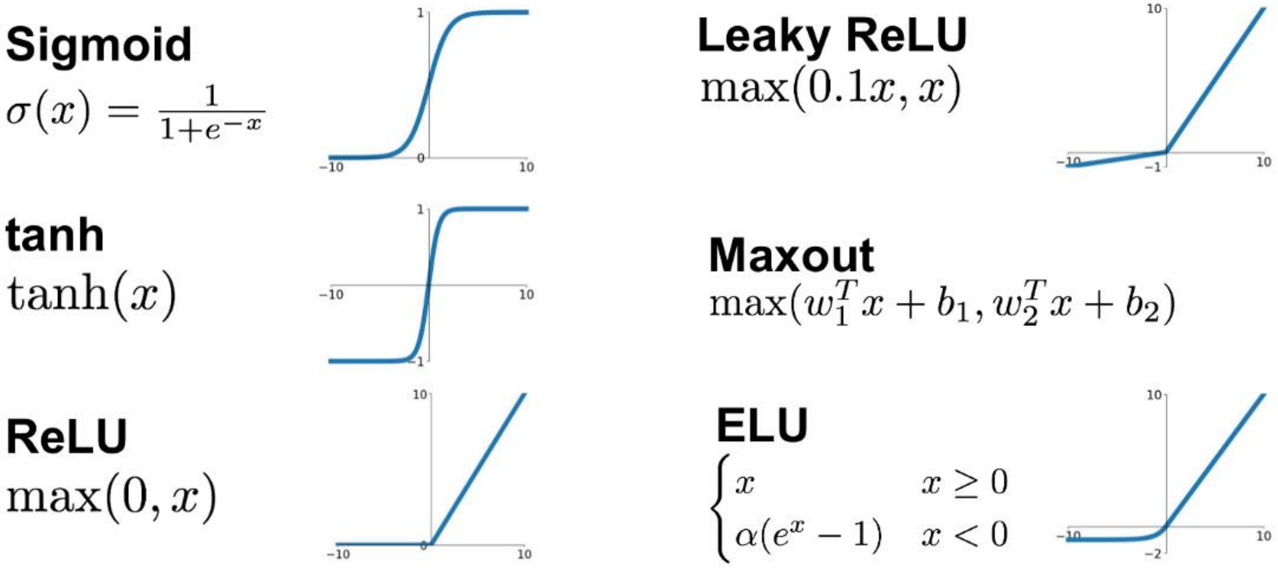
- 3개 이상의 Label을 갖는 데이터에 대한 분류 작업
- 활성화 함수는 주로 Softmax 함수를 사용

다항 분류(Multi Classification)

- 다중 분류 손실함수
  - categorical_crossentropy
    - One-hot encoding 데이터에 사용
    - 0으로 이루어진 백터에 하나만 1의 값으로 구별
    - [0,1,0,0,⋯,0]]
    - 가장 높은 값만 1, 나머지는 0
  - sparse_categorical_crossentropy
    - Integer type 클래스
    - [0,1,2]
    - 각 샘플이 여러 개의 클래스에 속할 수 있을때 사용

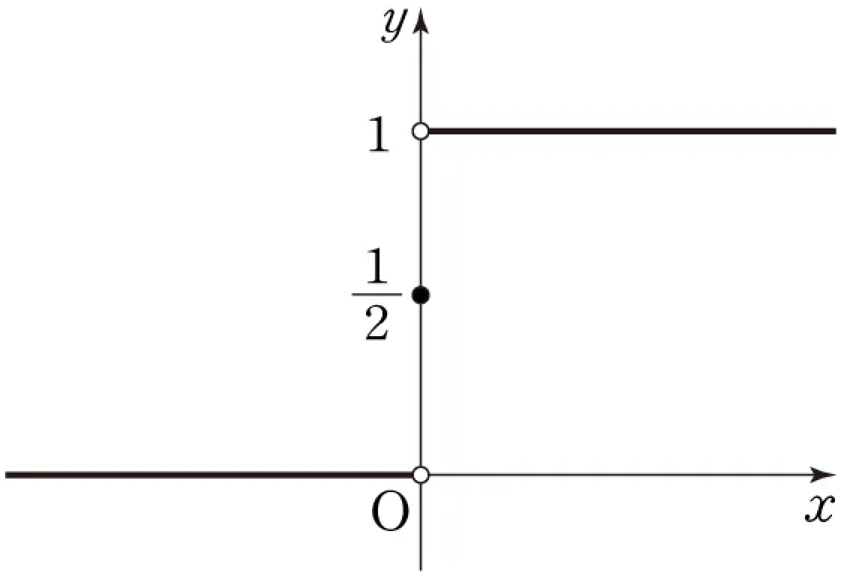
활성화 함수(Activation Function)

- 뉴런을 활성화 할 필요가 있는지 없는 지 결정하는 데 도움을 주는 함수
- 비선형 문제를 해결하는 데 중요한 역할을 한다
- 선형분류기를 비선형분류기로 만들 수 있다.



계단 함수

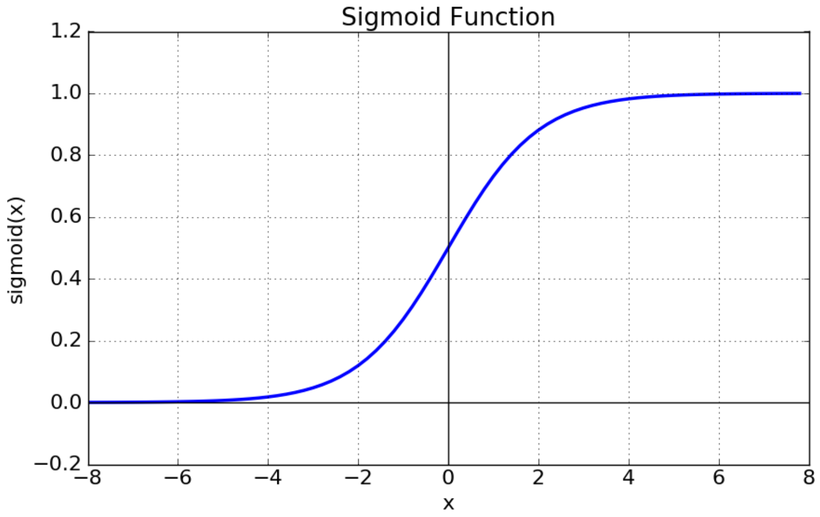
- 임계값(0)을 경계로 출력이 바뀌는 함수

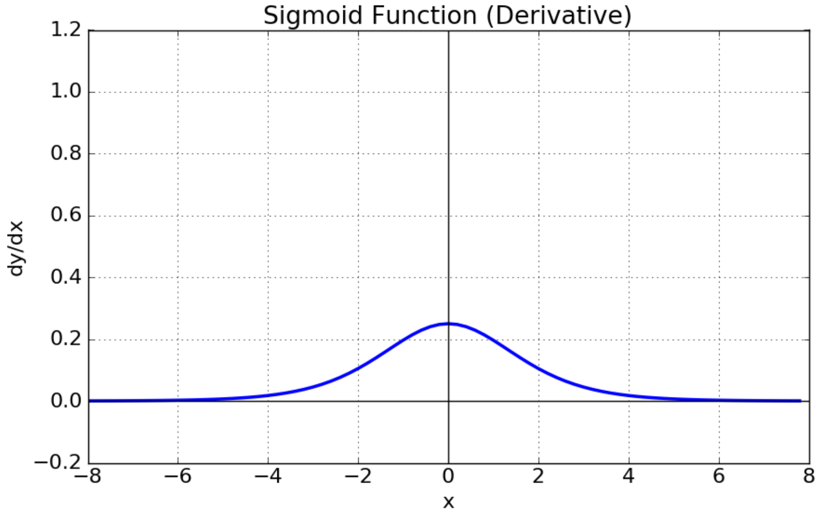


$y=u(x)$

시그모이드(Sigmoid) 함수

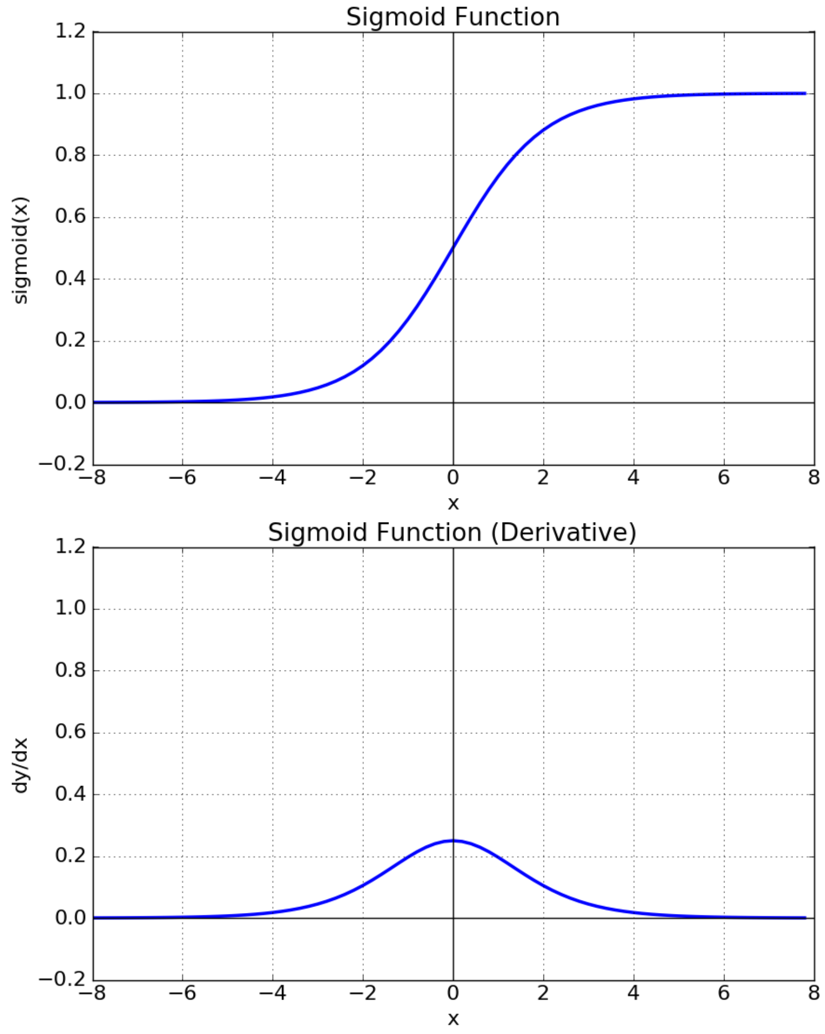
- 계단함수는 미분이 불가능하고 극단적인 값을 전달하기 때문에 데이터 정보가 손실됨
- 계단함수를 곡선의 형태로 변형시킨 형태
- 로지스틱 함수
- 특징
  - 함수값이 (0,1)로 제한
  - 중간값은 0.5
  - 매우 큰 값을 가지면 함수값은 거의 1
  - 반대는 거의 0





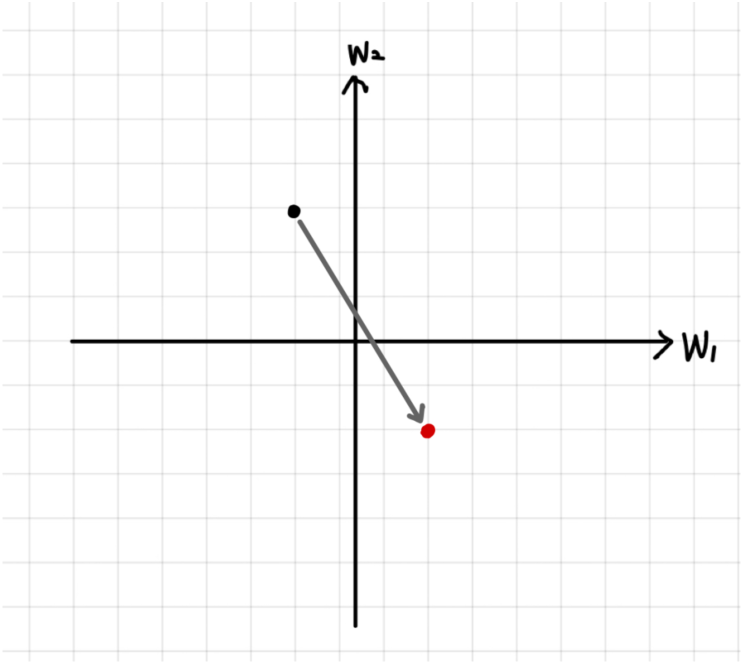
시그모이드(Sigmoid) 함수

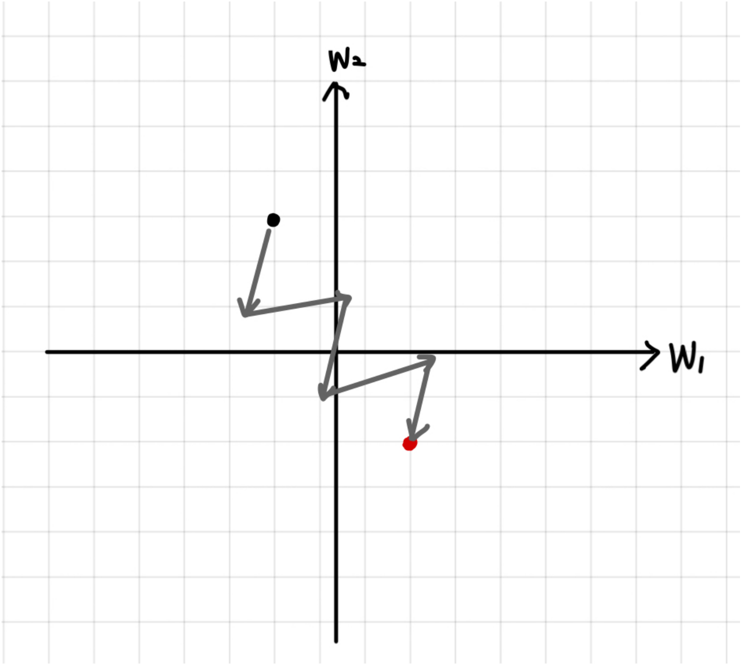
- 단점
  - **Gradient Vanishing** 현상 : x의 절대값이 커질수록 미분값이 소실
  - 함수의 중심이 0이 아니다. : 항상 같은 부호를 갖게 되므로 학습이 느려짐
- 이러한 이유로 최근에는 자주 사용하지 않음



시그모이드(Sigmoid) 함수

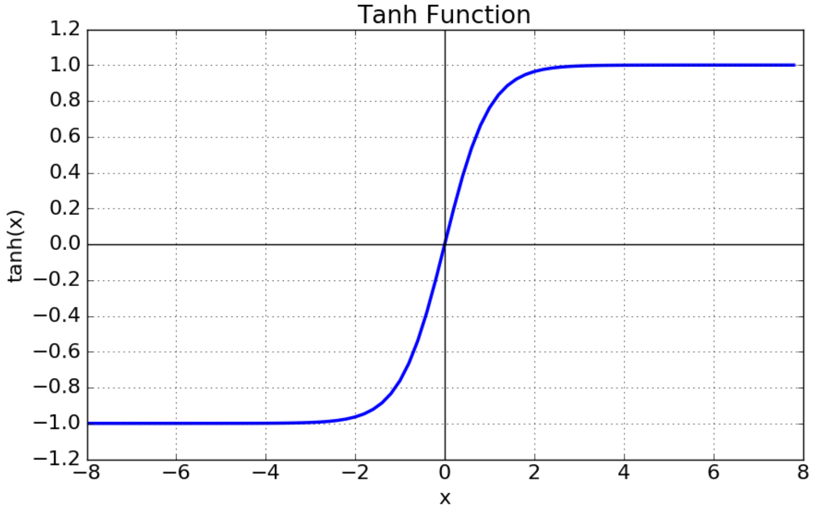
- 왼쪽은 최적의 이동(w2감소 w1증가)
- 둘의 부호가 같을 때의 이동 방법

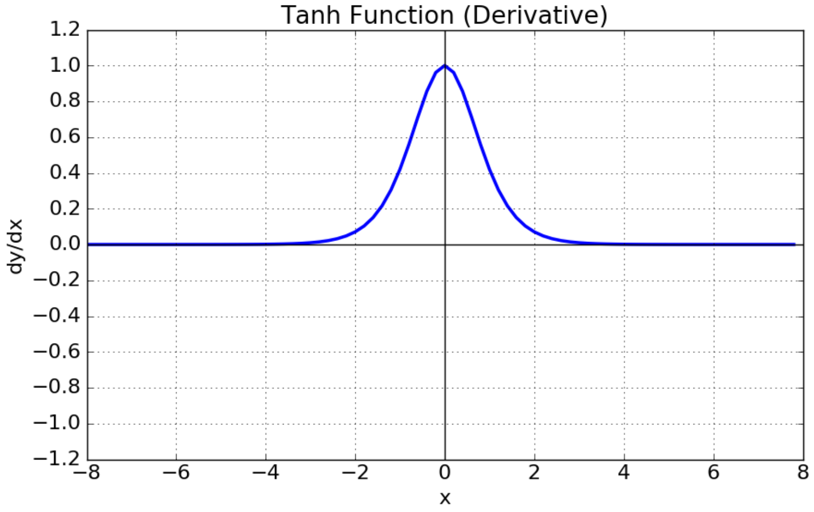




하이퍼볼릭 탄젠트(tanh) 함수

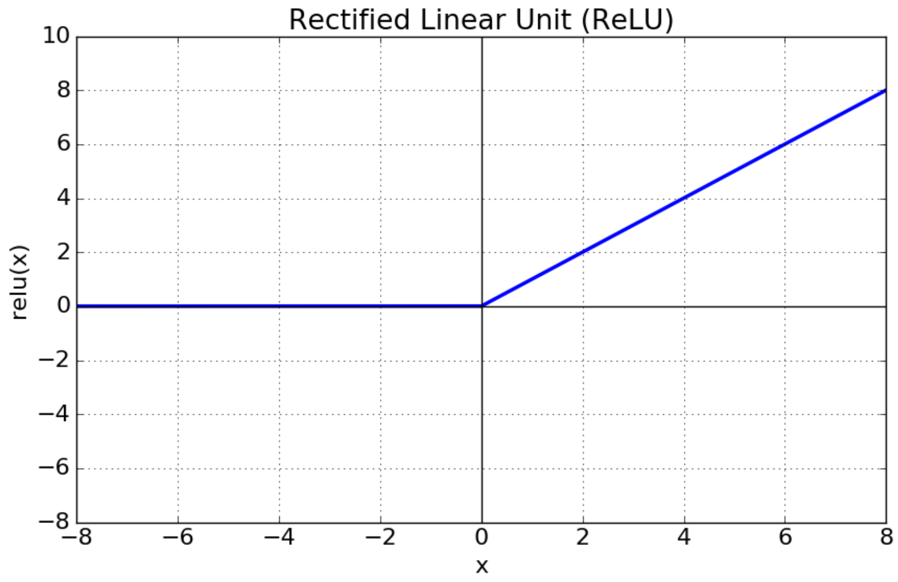
- 시그모이드 함수를 transformation해서 얻을 수 있다
- 중심값을 0으로 옮겨 최적화가 느린 문제 해결
- Gradient vanishing 문제는 여전





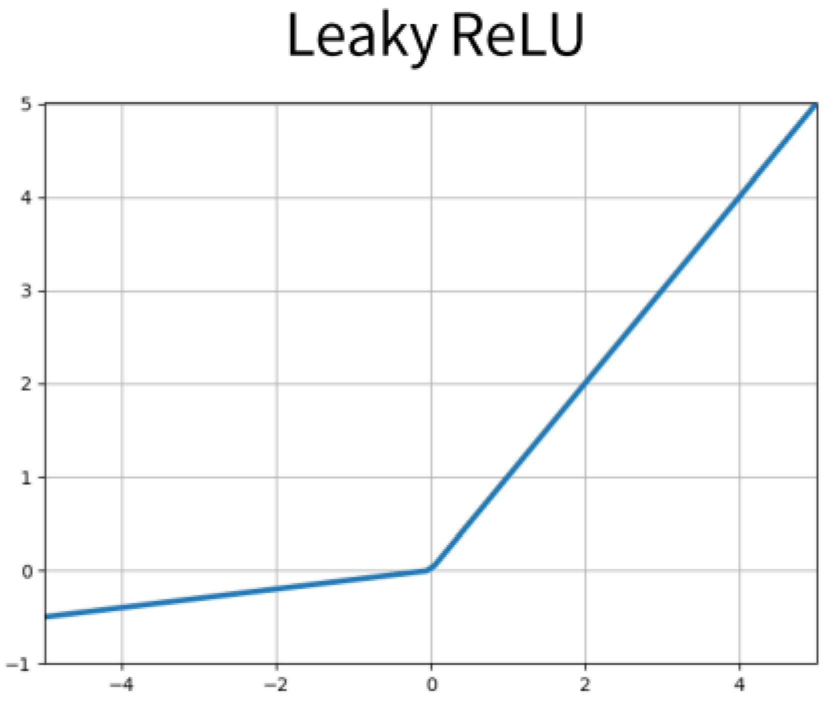
ReLU 함수 (Rectified Linear Unit)

- 최근 가장 많이 사용되는 활성화 함수
  - F(x) = max(0,x)
- 특징
  - X 〉 0 이면 기울기가 1인 직선
  - X 〈 0 이면 0
  - 학습 속도가 빠름
  - 구현이 간단
  - X 〈 0 인 값들에 대해서는 기울기가 0이라서 학습에 사용되지 않는(뉴런이 죽는) 단점



Leakly ReLU 함수

- ReLU의 뉴런이 죽는(Dying ReLU) 문제를 해결하기 위해 나온 함수
  - F(x) = max(0.01x, x)
- X < 0 값에 대해 미분값이 0이 아님



Softmax 함수

- 입력되는 모든 값을 0과 1사이의 값으로 normalize
- 모든 입력값의 합이 1이 되게 함
- argmax ([1, 3, 0, 2]) = [0, 1, 0, 0]
- soft arg max([1, 3, 0, 2]) ≈ [0.087 0.644 0.032 0.24]
- 보통 출력층에 많이 사용됨

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n} \exp(a_i)}$$

Cross entropy

- Entropy
  - 정보량
  - 최적의 전략 하에서 그 사건을 예측하는 데에 필요한 질문 개수
  - 최적의 전략 하에서 필요한 질문개수에 대한 기댓값
  - Entropy가 감소한다는 것은 우리가 그 사건을 맞히기 위해서 필요한 질문의 개수가 줄어드는 것 (= 정보량이 줄어든다)

Cross entropy

- Entropy
  - 모든 사건이 같은 확률로 일어날 때 최댓값을 갖는다
  - 예를 들어 1~16사이의 숫자 중 하나를 맞춘다고 하자. 상대방은 yes/no 만 대답 가능하다.
    - 총 4번의 질문이 필요하다.(binary search 이용했을 때) **4** = $\log_2(16)$
    - 확률 p=1/16 이라고 하면 정보량 I는

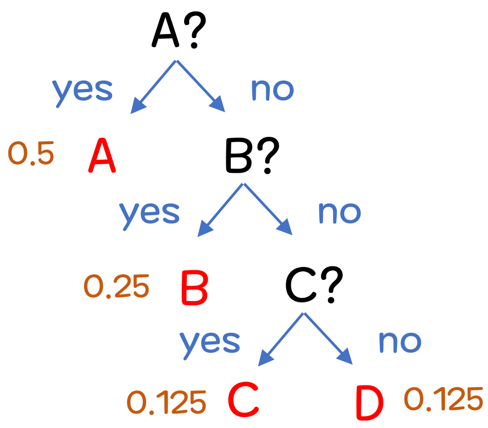
$$I = \log_2\left(\frac{1}{p}\right) = -log_2(p)$$

Cross entropy

- Entropy
  - 4가지 경우의 수의 엔트로피
    - A,B,C,D가 나올 확률이 각각 $^1/_4$ 일때
    - $^1/_4 * -\log_2(^1/_4) + {}^1/_4 * -\log_2(^1/_4) + {}^1/_4 * -\log_2(^1/_4) + {}^1/_4 * -\log_2(^1/_4) =$ **2**

Cross entropy

- Entropy
  - A가 나올 확률이 $^1/_2$, B가 나올 확률이 $^1/_4$, C가 나올 확률이 $^1/_8$, D가 나올 확률이 $^1/_8$일때
    - 최적의 질문 전략은



    - $^1/_2 * -\log_2(^1/_2) + {}^1/_4 * -\log_2(^1/_4) + {}^1/_8 * -\log_2(^1/_8) + {}^1/_8 * -\log_2(^1/_8) =$ **1.75**
  - **확률에 따라 엔트로피 값은 달라진다**

Cross entropy

- Cross entropy
  - 어떤 문제에 대해 **특정 전략을 쓸 때** 예상되는 질문 개수에 대한 기댓값
  - 확률분포로 된 어떤 문제 p 에 대해 확률 분포로 된 어떤 전략 q를 사용할 때의 질문 개수에 대한 기댓값
  - 앞선 문제에서 마찬가지로 binary search 전략으로 물어본다면?
    - $\frac{1}{2} * -\log_2(\frac{1}{4}) + \frac{1}{4} * -\log_2(\frac{1}{4}) + \frac{1}{8} * -\log_2(\frac{1}{4}) + \frac{1}{8} * -\log_2(\frac{1}{4})$ = **2**

Cross entropy

- Cross entropy
  - 최적의 전략을 쓸 때 entropy가 최소가 된다.
  - Cross entropy를 최소화 하는 것이 가장 loss를 적게 하는 것
  - 분류의 주된 loss function 으로 사용

KL-divergence(Kullback-Leibler divergence)

- 쿨백-라이블러 발산(상대 엔트로피 라고도 함)
- 두 확률분포의 차이를 계산하는 데에 사용하는 함수
- 어떤 이상적인 분포에 대해, 그 분포를 근사하는 다른 분포를 사용해 샘플링을 한다면 발생할 수 있는 정보 엔트로피의 차이를 계산
- Cross entropy를 minimize하는것은 KL-divergence를 minimize하는것과 동일
- 항상 0 이상인 값