Parrot Deep Learning

Session 02.

Loss function, Back propagation

박준서(p_jun)

- 좋은 모델이란 뭘까?
- 확률과 가능도
- 어떤 파라미터의 값을 ‘얼마나’ 변경시켜야 할까?

좋은 모델

- cost나 speed는 일단 넘어가고 quality, 퍼포먼스 관점만 생각해 봅시다

- target을 정확하게 예측

- target을 잘 구분

- 비슷한거 아닌가…

좋은 모델

- 기상청 일기 예보 모델을 만들었다 생각해 봅시다

- 비 : 33% 맑음 : 34% 흐림 33%

- 비 : 2% 맑음 : 87% 흐림 11%

- 둘 모두 맑음을 예측하기 하겠죠..?

entropy

- entropy는 불확실성의 척도이다
- entropy가 높으면 정보가 많고 확률이 낮다는걸 의미한다

$$H(x) = -\sum_{i=1}^{n} p(x_i) log p(x_i)$$

- 직관적으로 어떤 데이터가 나올지 예측하기 어렵다면, 엔트로피가 높다고 이해해도 좋다
- 엔트로피가 낮을 수록 사건을 명확하게 특정지을 수 있다

entropy

- 동전 던져 앞과 뒤가 나오는 사건
- 주사위를 굴려 1~6이 나오는 사건
- 두 상황에서 불확실성은 주사위가 더 크다고 직관적으로 다가온다
  - $H(x) = -\left(\frac{1}{2}log\frac{1}{2} + \frac{1}{2}log\frac{1}{2}\right)$ = 0.693
  - $H(x) = -\left(\frac{1}{6}log\frac{1}{6} + \frac{1}{6}log\frac{1}{6} + \frac{1}{6}log\frac{1}{6} + \frac{1}{6}log\frac{1}{6} + \frac{1}{6}log\frac{1}{6} + \frac{1}{6}log\frac{1}{6}\right)$ = 1.79

Probability vs Likelihood

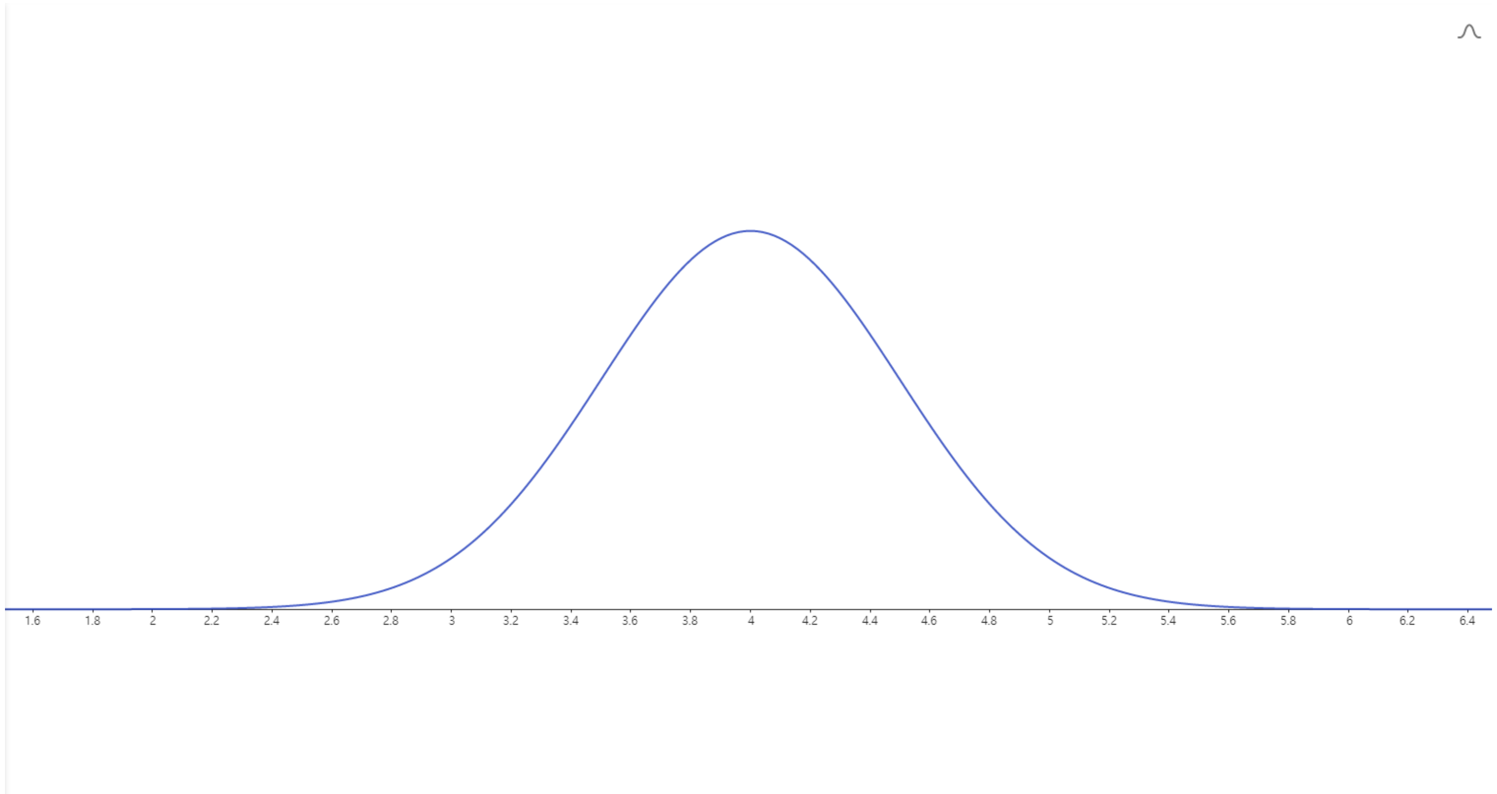
- 그래서 이걸 어떻게 우리 머신러닝에 적용 할 수 있을까?

- *Probability vs Likelihood*

- 머신러닝을 통해 우리의 예측과 실제 데이터를 일치시키기
- Likelihood를 최대화 하기

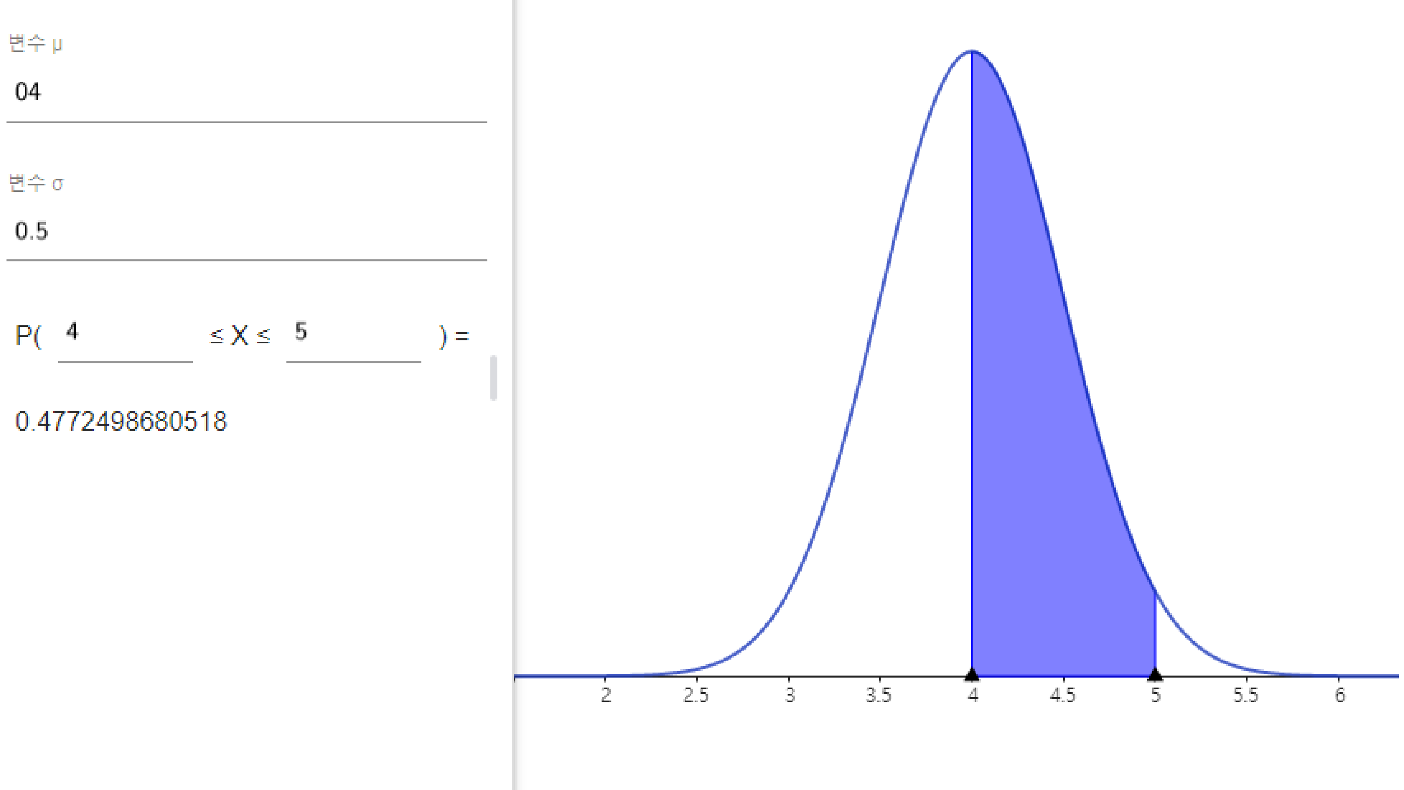
Probability vs Likelihood

- 고양이의 몸무게의 평균이 4kg이고, 표준 편차가 0.5라고 생각해보자
- 고양이의 몸무게가 4kg 에서 5kg이 될 확률은 어떻게 구할 수 있을까?



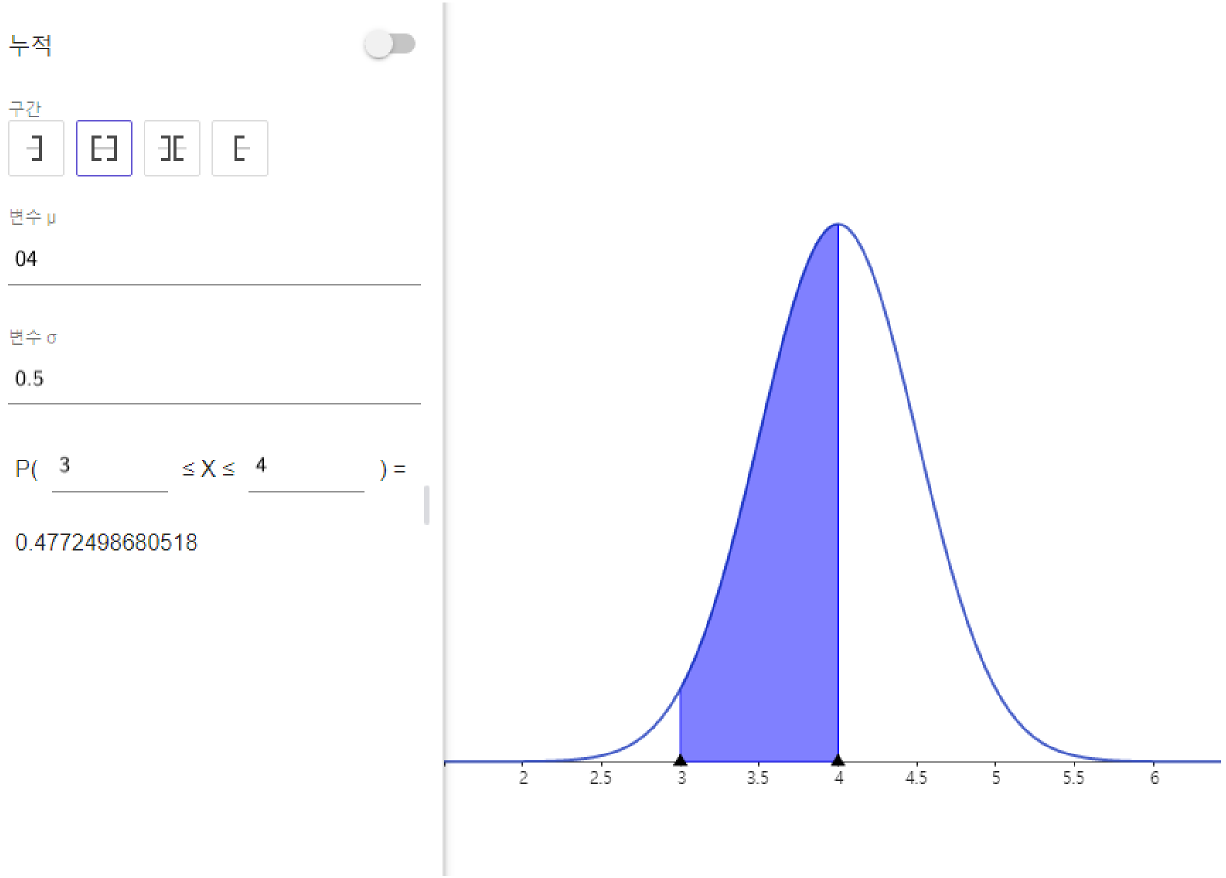
Probability vs Likelihood

- 연속 확률 변수는 확률 밀도 함수(PDF)의 넓이를 구해 확률을 구할 수 있다



Probability vs Likelihood

- 몸무게가 3kg 에서 4kg 이라면?



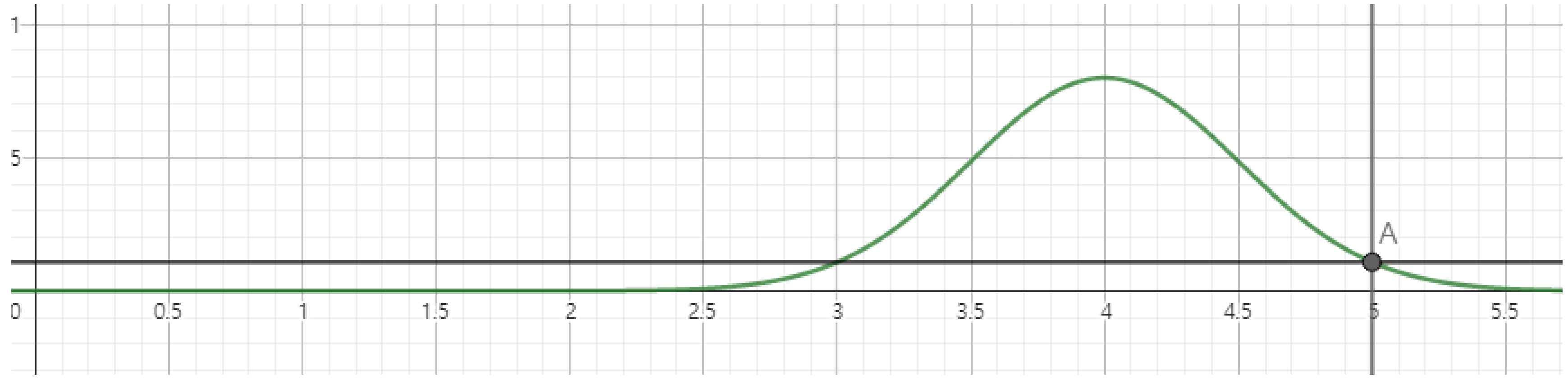
Probability vs Likelihood

- 입력 데이터(사건의 범위)는 변하지만 분포는 고정되어 있는 상황
- 데이터 : 4<=x<=5, 3<=x<=4

- $P(data \mid distribution) = probability$

Probability vs Likelihood

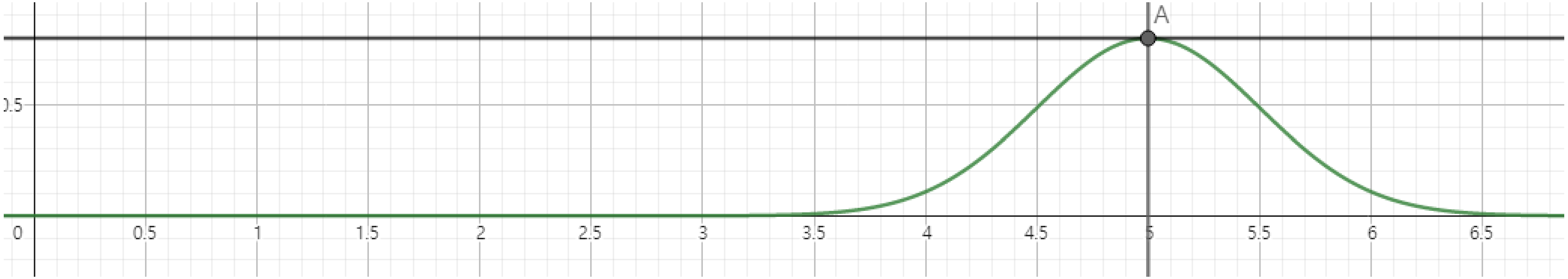
- Likelihood는 사건, 데이터가 정해졌을때, 어떤 분포로 부터 왔는지를 의미한다
- 우리 집 고양이가 5kg라고 해보자
- f(N(4, 0.5) | x = 5) = 0.108



Probability vs Likelihood

- 평균이 5, 표준편차가 0.5인 분포에 5kg의 고양이가 있다면?

- f(N(5, 0.5) | x = 5) = 0.7979



Probability vs Likelihood

- $L(distribution|data) = likelihood$
- 입력 데이터가 고정되어 있고, 분포가 변화는 상황
- 데이터가 주어졌을 때, 분포가 데이터 얼마나 잘 설명하는가

Probability vs Likelihood

- 평균이 4kg인 분포보다, 5kg인 분포가 주어진 데이터들을 더 잘 설명한다
- 즉, 발생가능성이 더 높다

- 데이터들이 주어졌을 때 더 나은 설명을 하는 분포 찾기 -> 머신러닝의 문제

$$X \sim P_\theta(X) \cdots \text{확률변수} X \text{가 모수 } \theta \text{에 대해 가지는 분포}$$

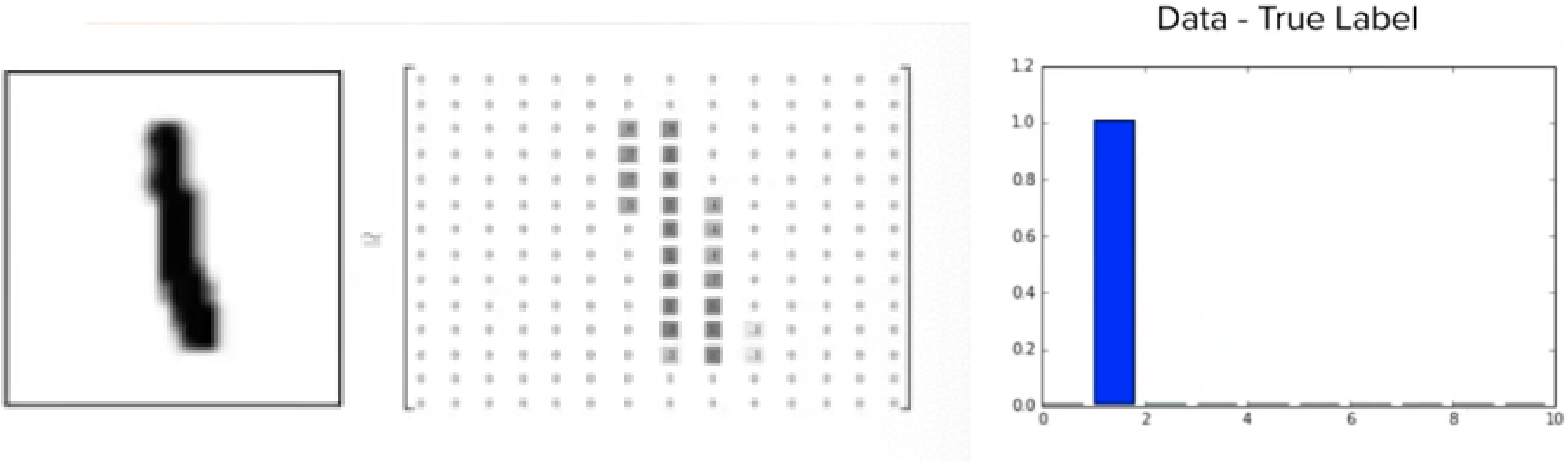
$$\mathcal{L}(\theta|x) = Pr(X = x|\theta)$$
$$\mathcal{L}(\theta|x) = \text{가능도 함수}$$

$$Pr(X = x|\theta) = Pr(x_1, x_2, x_3 \cdots |\theta) = Pr(x_1|\theta) \times Pr(x_2|\theta) \times \cdots Pr(x_n|\theta)$$
$$\mathcal{L}(\theta|x) = Pr(x_1|\theta) \times Pr(x_2|\theta) \times \cdots Pr(x_n|\theta)$$

- theta는 모델의 파라미터

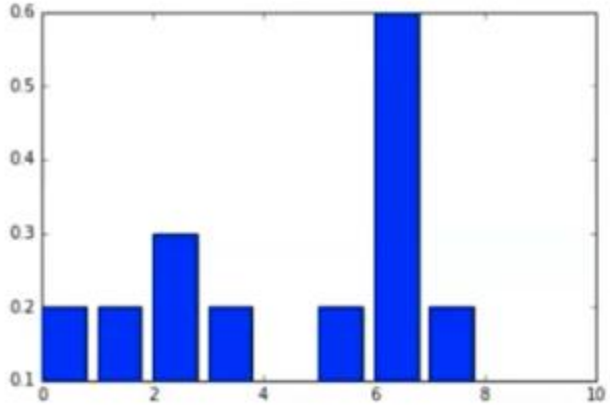
머신러닝에서 *likelihood*

- 오른쪽 손글씨가 어떤 숫자인지 알아내야하는 문제를 생각해보자
- 아래 정답은 1이다

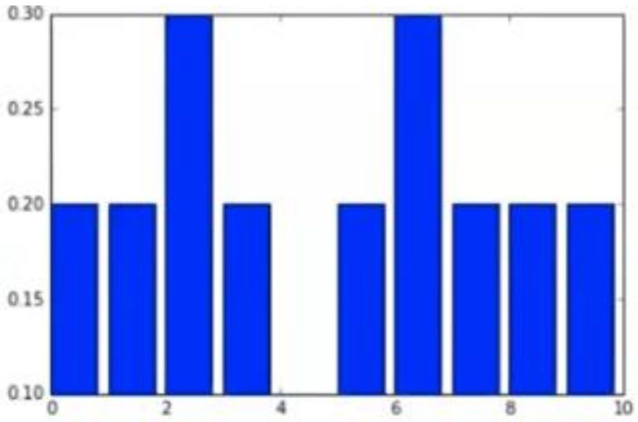


머신러닝에서 *likelihood*

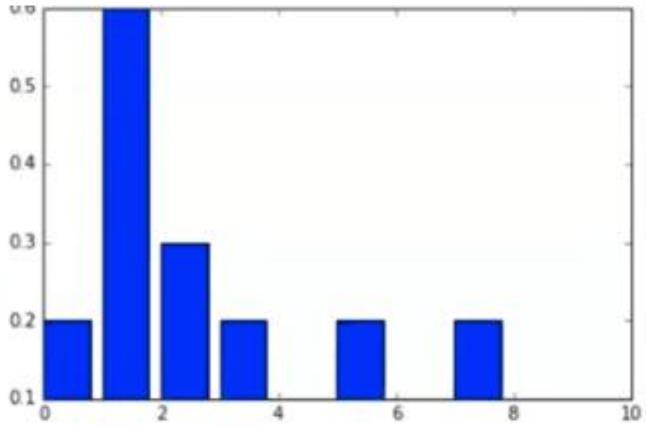
- 우린 3개의 모델을 만들었고 각각 아래와 같은 출력이 나왔다
- 가장 정답에 가까운 분포는 무엇일까?



Model A



Model B

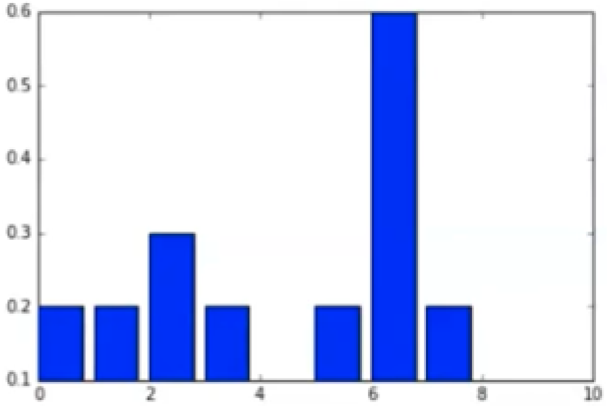


Model C

- 모델 C가 데이터를 가장 잘 설명하는 분포이다
- 즉, likelihood가 가장 높은 분포이다

머신러닝에서 *likelihood*

- 따라서 머신러닝의 목적은 likelihood를 최대화 하는 것이라 볼 수 있다



- 고양이 이미지의 성분(귀, 수염 등등)에 맞는 고양이 모델 제작

머신러닝에서 *likelihood*

- MLE(maximum log likelihood) : log가 있어도 최대화 되는 파라미터는 동일하니까~
- 추후 gradient vanishing 등 문제 해결에도 도움이 된다

- 학습이란 모델을 정답에 가까운 분포로 만드는 과정이다
- 과정의 방향성은 MLE이다

- 이를 loss 함수를 통해 얼마나 분포가 차이 나는지 알아낸다
- Cross entropy

entropy와 cross entropy

- 크로스 엔트로피는 아래 식으로 나타난다
- q(x)는 실제 세상의 확률, p(x)는 모델을 통해 구한 확률이다

$$H(x) = -\sum_{i=1}^{n} p(x_i) log p(x_i)$$

entropy

$$H_p(q) = -\sum_{i=1}^{n} q(x_i) log p(x_i)$$

Cross-entropy

- 분포 간 차이
- 실제 값과 예측 값이 다른 정도

- 로스를 줄이는 방향이 곧 가능도를 최대화 하는 방향

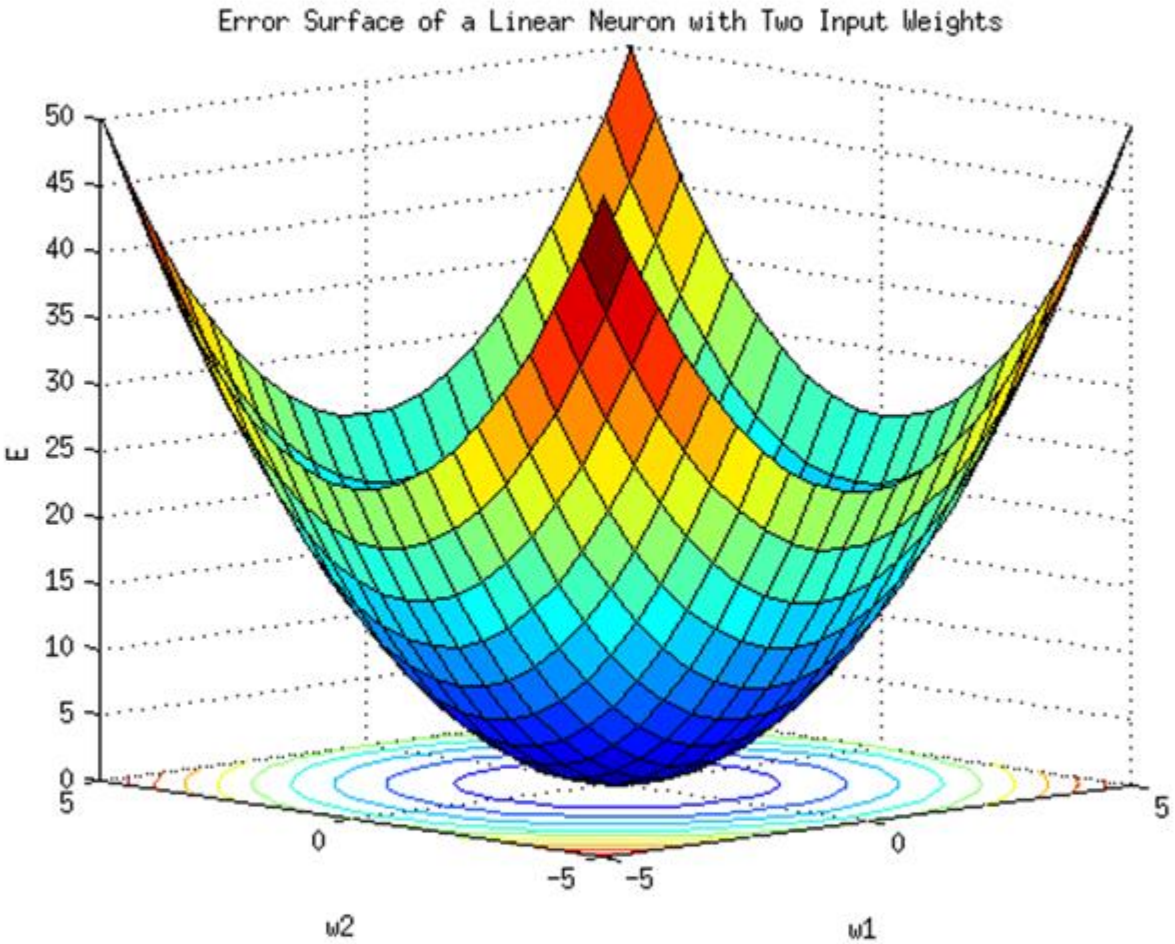
Back propagation

- 그렇다면 어떻게 cross entropy, loss 함수를 줄일 수 있을까?

- 학습에 이용되는 알고리즘 Back propagation을 알아보자
- 실제 target과 모델의 output이 얼마나 차이가 나는지 구한 후 그 오차를 다시 전파

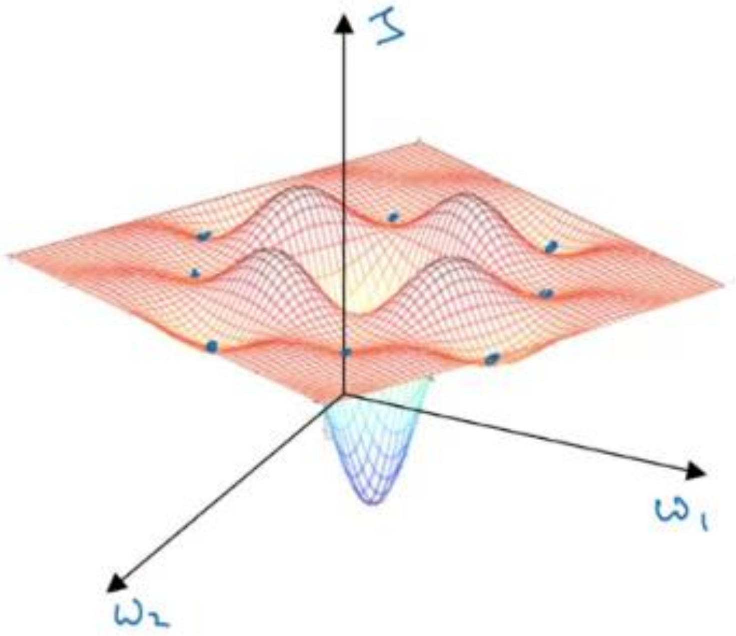
- 편미분과 chain rule은 다들 알고 계시겠죠..?

Back propagation



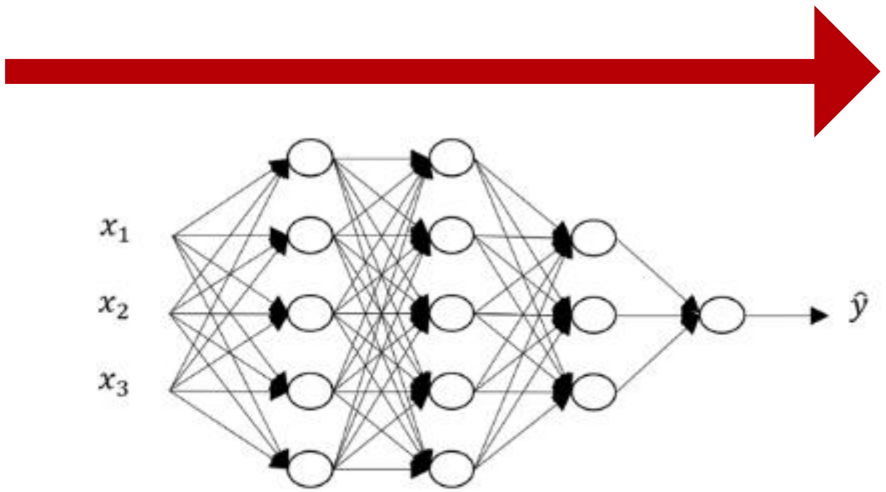
- 선형 모델일 경우 위와 같이 오목한 형태의 loss 함수 형태가 되어 gradient descent를 사용하면 됩니다

Back propagation



- Loss 함수의 형태, 모델에 따라 위와 같이 local minimum이 발생 할 수도 있습니다
- 수학적인 이유로 주로 안장점에서 발생하는데… 그냥 마법이라 생각해도 무방합니다
- 그런 경우 추후에 배울 다양한 optimizer를 사용하여 문제 해결이 가능하니 일단 넘어갑시다

Back propagation



- 모델의 예측 과정을 다시 살펴보자
- input : $\boldsymbol{a}^{[l-1]}$
- output : $\boldsymbol{a}^{[l]} = \boldsymbol{g}^{[l]}(\boldsymbol{z}^{[l]}),\ \boldsymbol{z}^{[l]} = \boldsymbol{w}^{[l]}\,\boldsymbol{a}^{[l-1]} + \boldsymbol{b}^{[l]}$

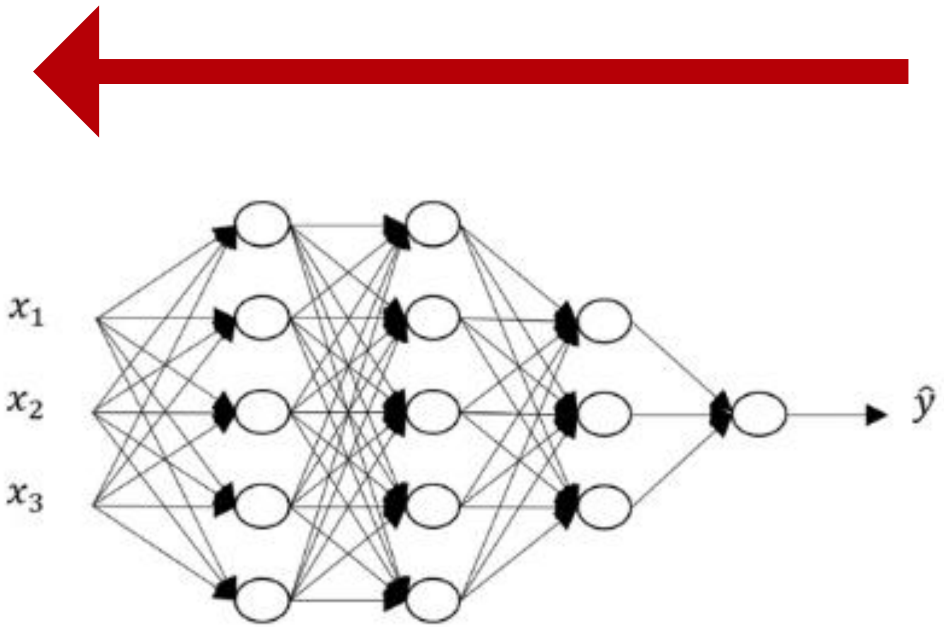
Back propagation

- 전체적인 과정은 이전에 배운 머신러닝에서의 gradient descent와 동일합니다
- 차이점은 각 레이어 단위로 dw, db를 계산하여 모든 파라미터를 동시에 업데이트 해야합니다

- 전 보통 db를 dw에 붙여서 계산합니다…만 이번엔 따로 식을 살펴보죠

- 모두 공통적으로 loss 함수를 미분하였기에 dL은 생략하겠습니다
- $\mathbf{d}\boldsymbol{w}^{[l]} := \boldsymbol{dL}/\mathbf{d}\boldsymbol{w}^{[l]}$

Back propagation



- input : $\mathbf{d}\boldsymbol{a}^{[l]}$
- output : $\mathbf{d}\boldsymbol{a}^{[l-1]}$, $\boldsymbol{dw}^{[l]}$, $\boldsymbol{db}^{[l]}$

Back propagation

- input : $\mathbf{d}\boldsymbol{a}^{[l]}$
- output : $\mathbf{d}\boldsymbol{a}^{[l-1]}$, $\boldsymbol{dw}^{[l]}$, $\boldsymbol{db}^{[l]}$

$$\boldsymbol{a}^{[l]} = \boldsymbol{g}^{[l]}(\boldsymbol{z}^{[l]})$$
$$\boldsymbol{z}^{[l]} = \boldsymbol{w}^{[l]}\, \boldsymbol{a}^{[l-1]} + \boldsymbol{b}^{[l]}$$

$$dz^{[l]} = da^{[l]} \times g^{[l]\prime}(z^{[l]})$$
$$dw^{[l]} = dz^{[l]} a^{[l-1]^T}$$
$$db^{[l]} = dz^{[l]}$$
$$da^{[l-1]} = w^{[l]^T} dz^{[l]}$$

- 사실 식 전개만 하면 되죠..?

Back propagation

- input : $\mathbf{d}\boldsymbol{a}^{[l]}$
- output : $\mathbf{d}\boldsymbol{a}^{[l-1]}$, $\boldsymbol{dw}^{[l]}$, $\boldsymbol{db}^{[l]}$

- $\boldsymbol{w}^{[l]} = \boldsymbol{w}^{[l]} - \ \boldsymbol{\alpha}\ \boldsymbol{dw}^{[l]}$
- $\boldsymbol{b}^{[l]} = \boldsymbol{b}^{[l]} - \ \boldsymbol{\alpha}\ \boldsymbol{db}^{[l]}$

- 모든 계산을 진행 후 동시에 업데이트를 합니다
- 이 과정을 통해 우린 모델을 학습 시킬 수 있습니다

QnA

- 질문 있으신가요?