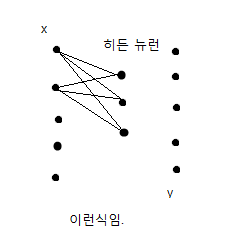
NN : 뉴럴 네트워크?, 뉴랄넷? : error를 이용

두뇌를 흉내, 기본 단위를 뉴런이라 부른다.



선은 w

겹y <- f(겹x, 겹w)

D={(겹xi, 겹yi)}i는 1부터 n까지 => w, s.t. y = f(겹x, 겹w)

1. 겹xd를 NN에 입력.

2. hidden unit 값 계산

Si = sigma from j wij xj

xi = small sigma(Si) = 1/(1+e^(-Si))

3. output unit에 값 계산

Sk = sigma wki xi

4. y(hat)k와 yk 값을 비교해서 error 계산.

Ed(겹w) = 1/2 \* sigma from k=1 to m (yk – y(hat)k)^2 + a\*C(겹w) <- 복잡도

5. w의 값을 조정.(error minimize 하도록)

wi <- wi + (learning rate) \* (겹y – 겹y(hat))xi

6. hidden ~~ weight 층에도 반복

히든 뉴런의 개수를 조정, 개수가 낮으면 복잡도가 낮아지고 에러가 많이 발생한다, 복잡한 문제를 해결 x

개수가 많으면 overfitting : 패턴을 찾는 것이 아니라..

Occam’s Razor : 어떤 이론이나 model을 만들 때 똑 같은 현상에 대해 설명하는 게 있으면 간단한 것을 선택하라.

Decision tree : entropy를 이용한다.

y <- f(겹x, DT)

Data ->(학습) DT

\* Entropy

- uncertainty

- order

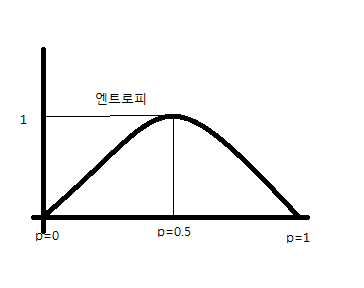
- 엔트로피 – H(p) = - sigma i pi \* log(pi)

p0 = 0.9

p1(우수 고객) = 0.1

Ii = log(1/pi) = -log(pi)

Ii 의 평균은 엔트로피. 따라서 엔트로피는 평균 정보량.



\* 데이터가 많아지면 DT는 사용 못한다.

\* 정확도는 NN과 DT 둘 다 비슷하다.

\* DT는 NN보다 이해하기 쉽다.

\* DT에서 path를 따라가면 association rule이다.

예) 강남에 살고 나이가 좀 있고 여자면 -> 우수 고객이다.