Intro to Deep learning (Deep neural networks)

Sang Yup Lee

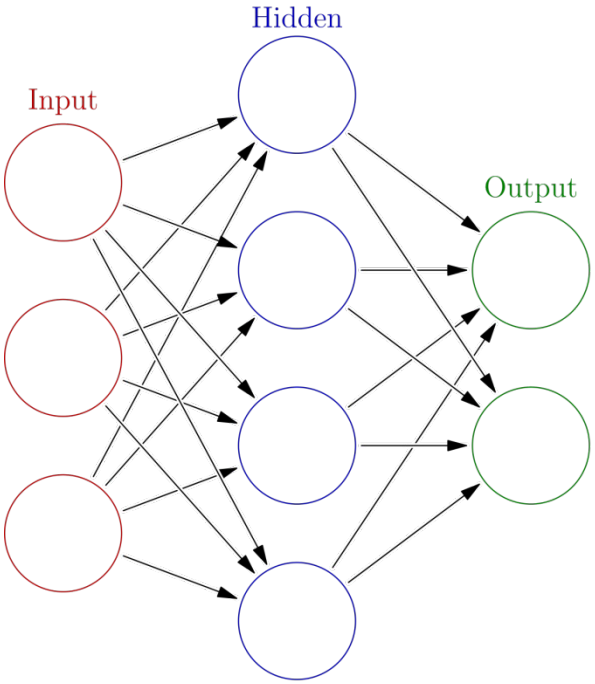
Deep learning

- A sort of machine learning
- 신경망 기반의 알고리즘
- 전통적인 ML 알고리즘들과의 비교
  - 일반적으로 성능이 더 좋다고 알려져 있음
    - 데이터의 크기와 특성 등에 따라 다름
    - 보통 비정형 데이터에 대한 성능 우수
- Then, what is a neural network? and how does it work?
  - 지도학습으로 사용되는 신경망 설명

Deep learning

- 3 basic layers
  - Input layer, Hidden layer, and Output layer

각 layer는
여러개의 노드로
구성



# of hidden layers >= 1

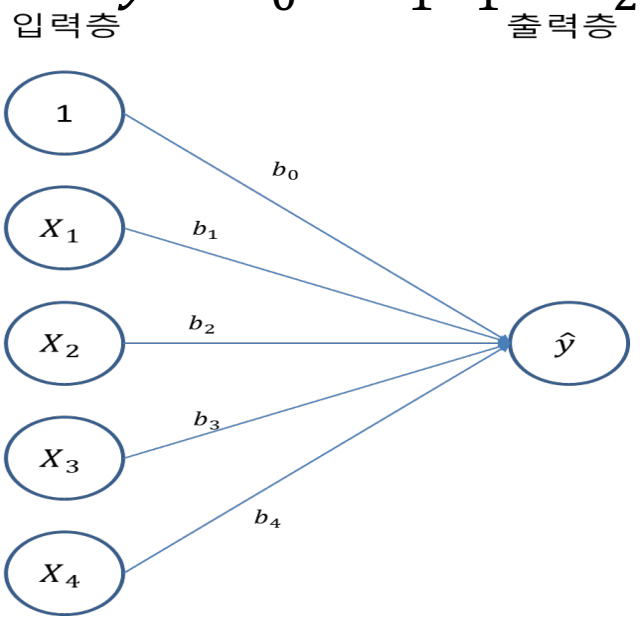
Deep learning

- 각 층(혹은 각 층의 노드들)의 주요 역할
  - 입력층: (각 관측치에 대해서) 독립변수의 값 (features 정보)을 입력 받고 다음 층으로 전달한다.
  - 은닉층: 입력받은 데이터에서 종속변수의 값을 맞히는데 중요한 특성을 추출한다.
  - 출력층: 종속변수의 예측치를 출력한다.

- 그렇다면 왜 전통적인 기계학습 알고리즘 보다 성능이 좋은가?

Deep learning

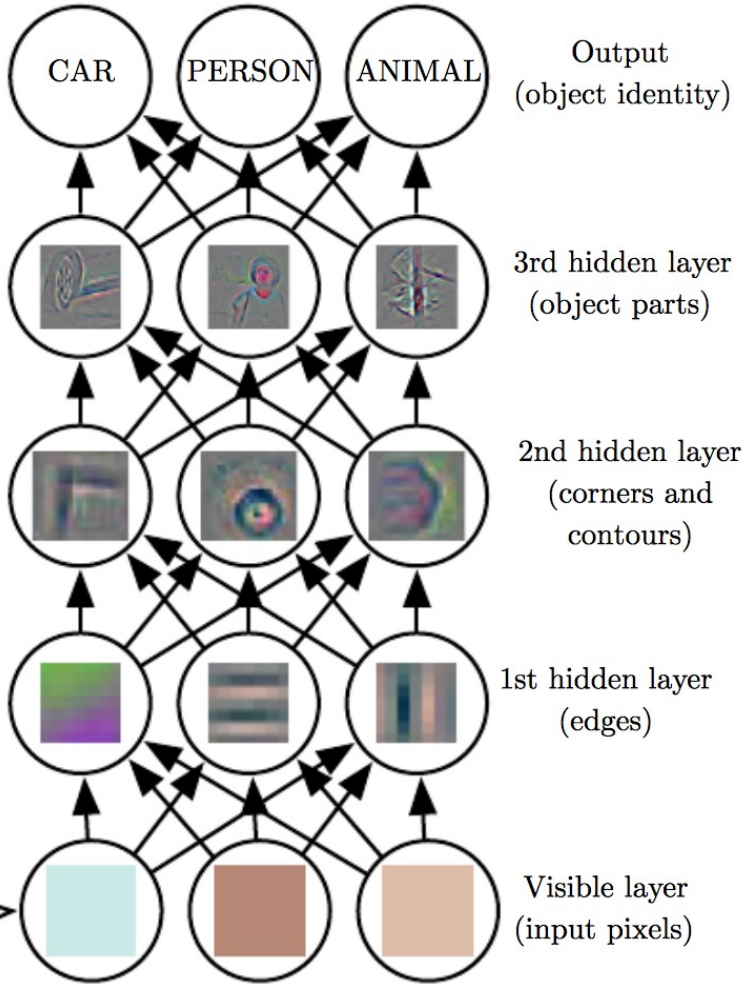
- 전통적인 기계학습 알고리즘의 신경망 표현
  - 선형회귀 모형의 예
  - $\hat{y} = b_0 + b_1 X_1 + b_2 X_2 + b_3 X_3 + b_4 X_4$



이는 은닉층이 신경망 (또는 딥러닝)에서 가장 중요한 역할을 한다라는 것을 의미함. (혹은 다른 기계학습 알고리즘과 딥러닝의 차이를 만드는 역할을 한다는 것을 의미).

Deep Learning

은닉층: 순차적으로 정답을 예측하는데 중요한 역할을 하는 정보를 추출한다.



Deep learning

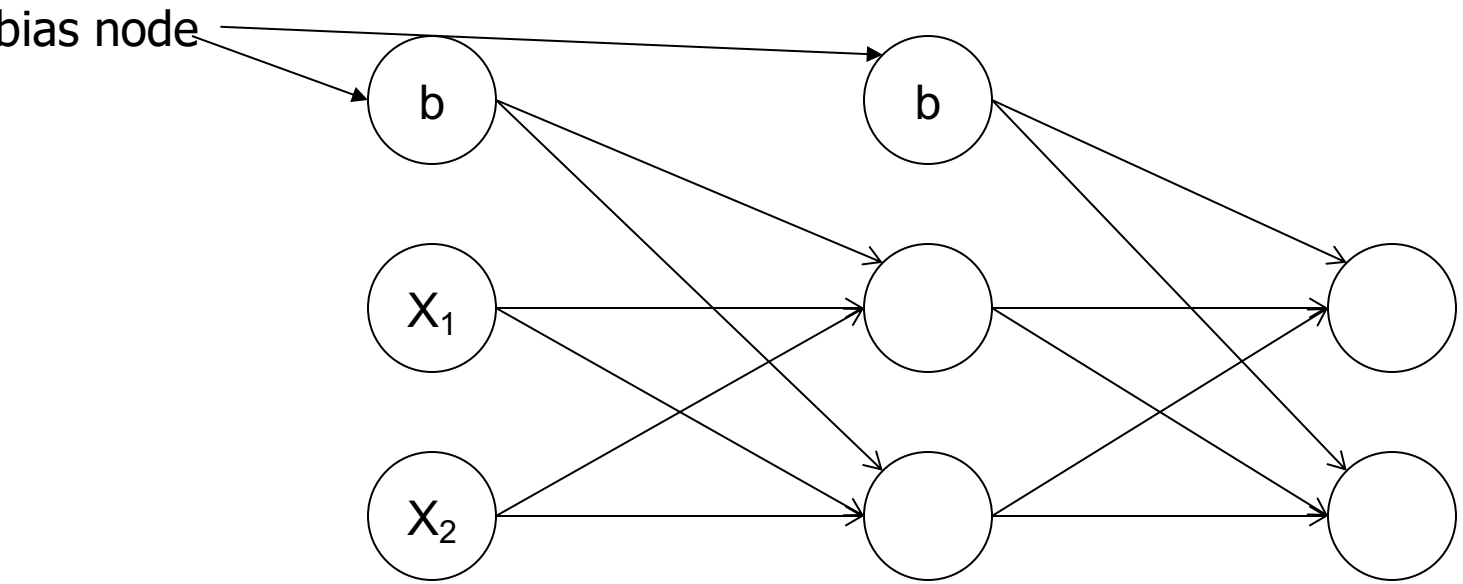
- 신경망의 구조
  - 입력층과 출력층은 언제나 1개
  - 은닉층의 수는 사용자가 결정
  - 은닉층의 수가 1 인 경우, 얕은 신경망 (shallow NN)
  - 은닉층의 수가 2개 이상인 경우 (Deep NN) =>
    보통 이를 딥러닝 알고리즘 이라고 함

Deep learning

- 신경망의 구조 (cont'd)
  - 각 층은 노드들로 구성
  - 노드의 역할: 어떠한 값을 입력받고, 출력하는 역할
    - 입력노드: 입력된 값을 그대로 출력
    - 은닉노드: 입력된 값을 특정 함수를 사용해서 변환해서 출력, 이러한 함수를 활성화함수(Activation function)이라고 함
    - 출력노드: 문제의 종류에 따라 달라짐
      - 회귀문제: 입력된 값을 그대로 출력
      - 분류문제: 종속변수가 각 값을 취할 확률을 출력
  - 각 층에 존재하는 노드의 수
    - 입력층의 노드의 수 = 독립변수의 수
    - 은닉층의 노드의 수 ⇒ 사용자가 임의로 결정
    - 출력층의 노드의 수 ⇒ 문제의 종류에 따라 달라짐
      - 회귀문제 ⇒ 출력 노드의 수 = 1
      - 분류문제 ⇒ 출력 노드의 수 = 종속변수가 취할 수 있는 값의 수

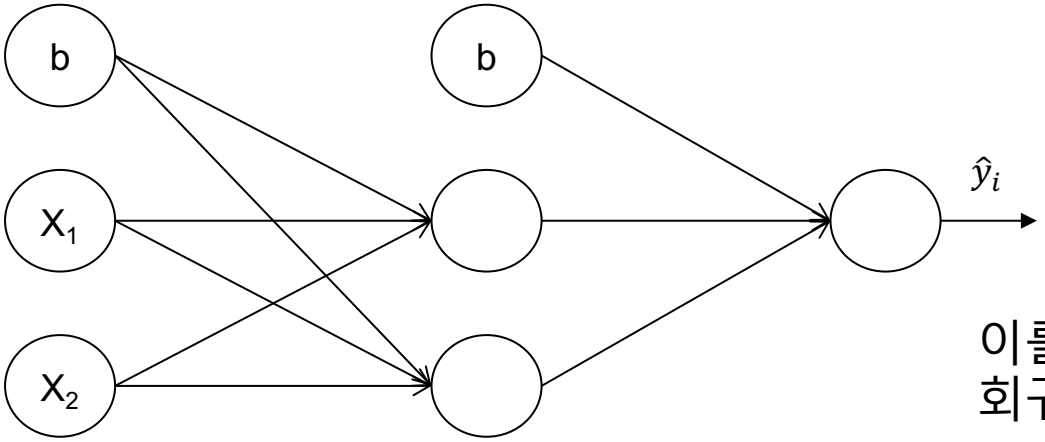
Deep learning

- bias node
  - 입력층과 은닉층에 존재
  - 선형 회귀 모형의 intercept 와 비슷한 역할
  - bias node에서 출력되는 값 ⇒ 1의 값을 출력한다고 생각



Deep Learning

- 출력노드의 수: 문제의 종류에 따라 다름
- 회귀문제
  - 예) 종속변수: 아파트 가격, 독립변수: 평수와 연식
  - 출력 노드 1개
  - 출력노드가 출력하는 값 = 종속변수의 예측치, 즉, $\hat{y}_i$



이를 이용해서 비용함수 계산
회귀문제는 MSE

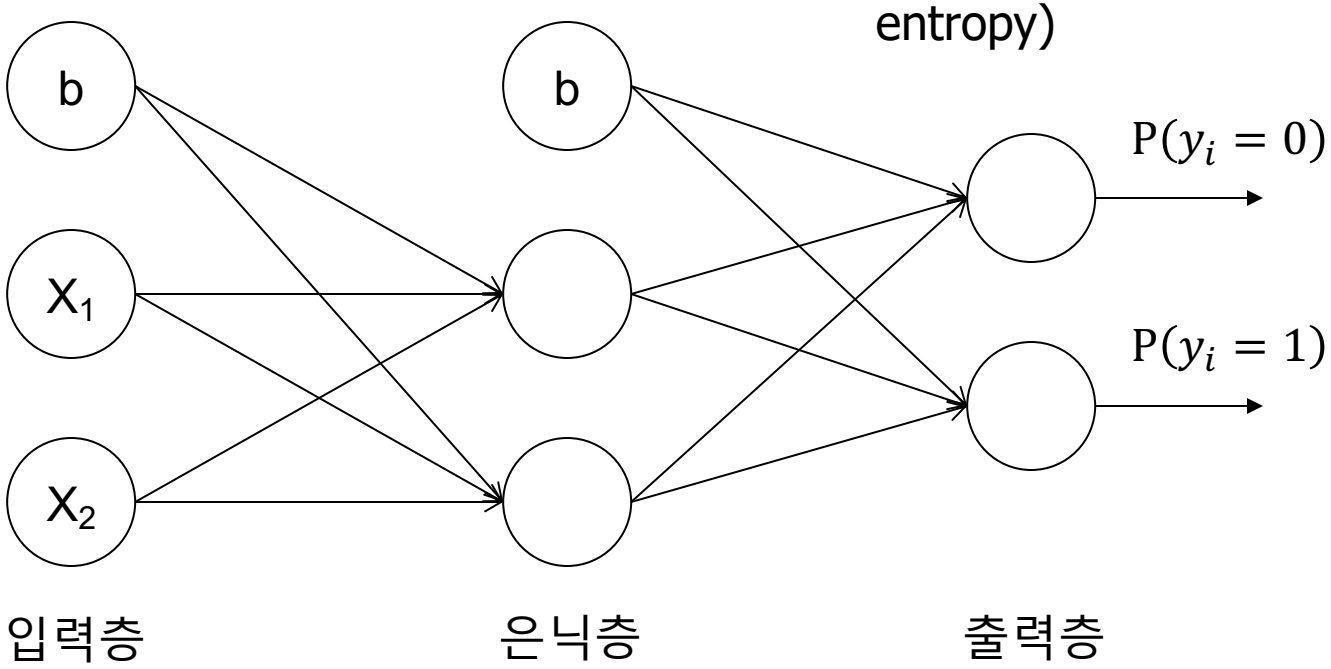
Deep Learning

- 분류문제
  - 출력 노드의 수 = 종속변수가 취할 수 있는 값의 수
  - 예) 폐암에 걸렸는지 여부, then $y_i \in \{0, 1\}$
    - 즉, 출력 노드의 수 = 2
    - 독리변수: 흡연 여부, 성별
  - 각 출력 노드가 출력하는 값은?
    - 종속변수가 각 값을 취할 확률
    - 첫번째 출력 노드의 출력값 = $\mathrm{P}(y_i = 0)$
    - 두번째 출력 노드의 출력값 = $\mathrm{P}(y_i = 1)$

Deep Learning

- 분류문제 (cont'd)

이를 이용해서 비용함수 계산
분류문제는 교차엔트로피 (cross entropy)



신경망 작동 원리

- 분류 문제의 비용함수: 교차 엔트로피
  - $E = -\left[\sum_{i=1}^{N} y_i \ln p(y_i = 1) + (1 - y_i) \ln p(y_i = 0)\right]$
  - $y_i \rightarrow$ 각 관측치의 실제 종속변수 값
  - $y_i \in \{0,1\}$
  - $p(y_i = 1), p(y_i = 0) \rightarrow$ 모형을 통해서 예측되는 값

Deep learning

- 신경망 작동 순서
  - ① 정답이 있는 데이터를 준비한다.
  - ② 정답이 있는 데이터를 학습 데이터와 평가 데이터로 분리한다.
    - 경우에 따라서는 validation set을 사용할 수도 있다.
  - ③ 신경망 (즉, 딥러닝 모형)을 이용해 학습 데이터에 존재하는 독립변수들과 종속변수의 관계를 파악한다.
  - ④ 학습 데이터에 대해서 비용함수를 최소화하는 파라미터의 (최적) 값을 찾는다.
  - ⑤ 학습을 통해 도출된 구체적인 파라미터 값을 갖는 모형의 성능을 평가 데이터를 이용해서 평가한다.
  - ⑥ 평가의 결과가 괜찮은 경우, 해당 모형을 우리가 풀고자 하는 문제에 대한 데이터에 적용해서 종속변수의 값을 예측한다.

Deep learning

- 전통적인 기계학습과 신경망 기반 딥러닝의 주된 차이
  - 수학적 모형
- 그렇다면 비용함수는?
  - 비용함수의 종류는 모형에 따라 달라지지 않는다.
  - 비용함수의 종류는 문제의 종류에 따라 달라진다.
    - 회귀문제: MSE
    - 분류문제: 교차엔트로피

Deep learning

- 신경망 모형의 작동 원리 (종속변수값 예측)
  - 예: 회귀문제
    - 종속변수 (y, 연속변수)
      - 예: 아파트 가격 예측
    - 2개의 독립변수: 평수 (X1), 연식 (X2)
  - 학습하기
    - 비용함수를 최소화하는 모형의 파라미터값을 찾는다!
    - 비용함수 $\rightarrow \frac{1}{N}\sum_i (y_i - \hat{y}_i)^2$
    - 각 관측치의 예측치 ($\hat{y}_i$)는 (신경망) 모형을 통해 도출
    - **그렇다면 $\hat{y}_i$ 는 어떻게 계산?**
  - 사용하고자 하는 신경망 모형
    - 예) 은닉층의 수 = 1, 은닉 노드의 수 =2 인 모형

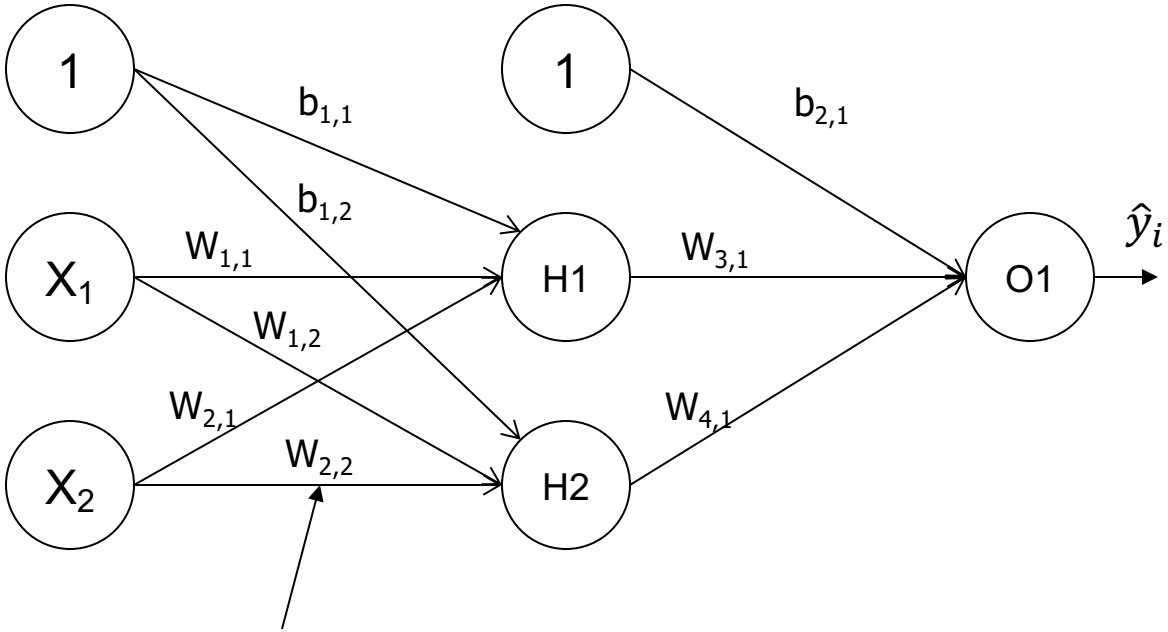
Deep learning

- 신경망 모형의 학습 (cont'd)
  - 사용하고자 하는 신경망 모형 구조 결정하기
    - 입력노드의 수 = ?
    - 출력노드의 수 = ?
    - 은닉층의 수와 은닉노드의 수는 사용자가 결정
    - 은닉층의 수 = 1, 은닉 노드의 수 =2 인 모형
      - 모형의 형태: 다음 페이지 참고

Deep learning

- Example (cont'd)

신경망에서는 파라미터는
가중치 (weight)와 편향으로 구성



이러한 화살표를 weight connection이라고 함

신경망 작동 원리

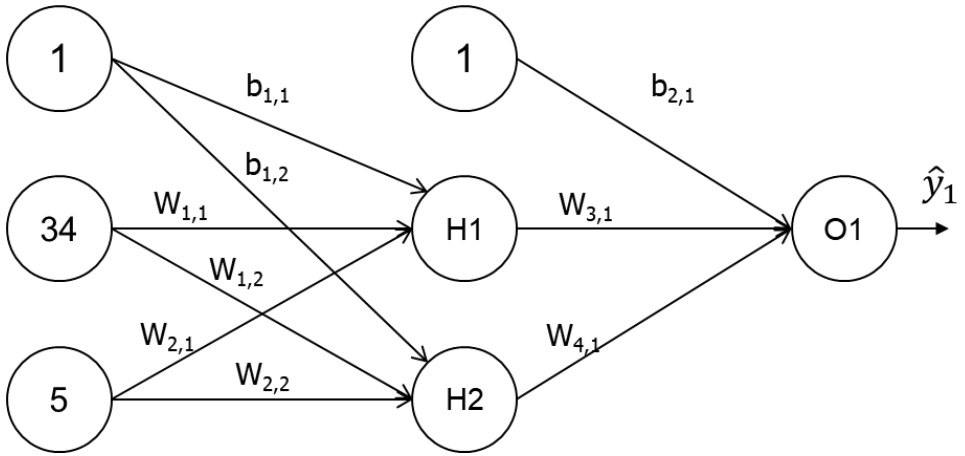
- Example
  - 문제: 아파트의 가격 예측 하기
  - 사용 독립변수: 아파트의 크기 (평형)와 연식
  - Toy training data (10개의 data points)
    - 다음 학습 데이터에 대해서 신경망이 어떻게 작동하는가?

신경망을 통해서 계산

ID	평수 (X1)	연식 (X2)	가격 (y)	$\hat{y}$
1	34	5	5	
2	25	5	2.5	
3	30	2	4	
4	38	20	3	
5	44	12	3.3	
6	48	18	4.2	
7	52	22	4.6	
8	60	19	6	
9	34	18	3	
10	34	22	2.9	

신경망 작동 원리

- What to do?
  - 각 관측치에 대해 종속변수의 예측치를 계산해야 함.
  - 그렇다면 어떻게 계산되는가?
- 첫번째 관측치의 경우: X1 = 34, X2 = 5
  - 첫번째 입력노드가 X1의 값을 입력받고, 두번째 입력노드가 X2의 값을 입력 받는다.



신경망 작동 원리

- 일반적인 작동 방식
  - 편향노드를 제외한 각 노드는 이전 층으로 전달받은 값들을 입력받고 그 값을 출력하여 다음 층의 노드로 전달하는 역할을 한다.
  - 입력노드는 입력된 (독립변수 or 피처)값을 그대로 출력한다.
  - 은닉노드는 입력받은 값을 그대로 출력하지 않고 그 값을 변환하여 출력한다. 이때 특정 함수가 사용된다. 이러한 함수를 **활성화 함수**라고 한다.
  - 출력노드는 문제의 종류에 따라 활성화 함수를 사용하기도하고 사용하지 않기도 한다.
    - 회귀문제: 활성화함수 없음
    - 분류의문제: 많은 경우 소프트맥스 함수 사용

신경망 작동 원리

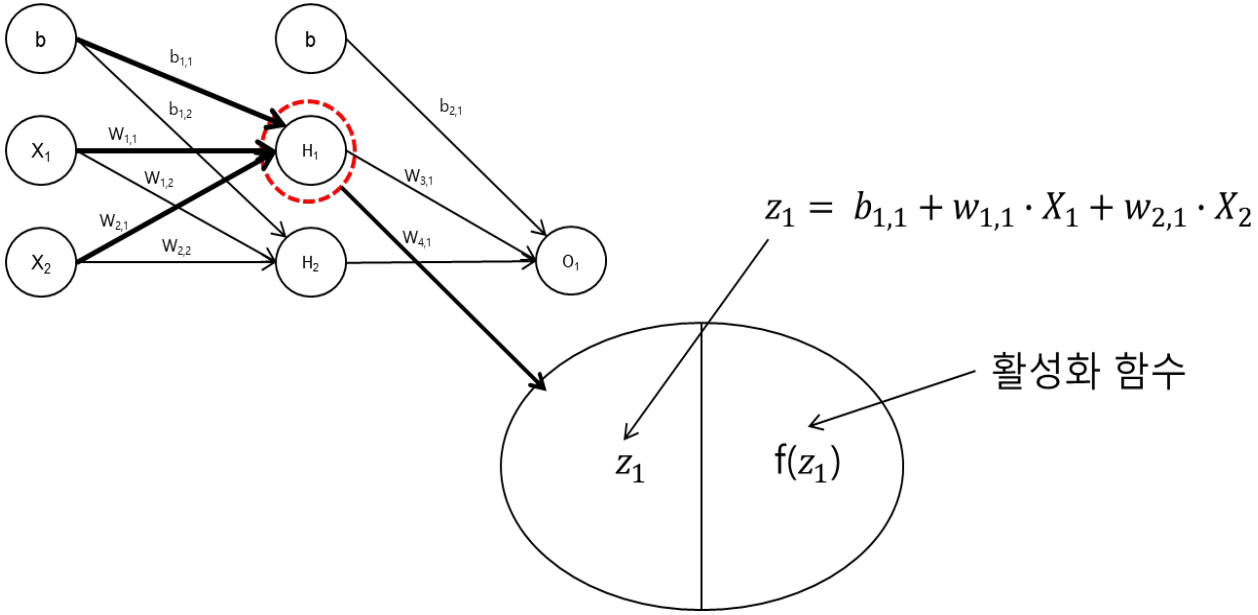
- 각 은닉노드 (즉, H1과 H2)에 입력되는 값은 무엇인가?
  - **입력 노드가 출력하는 값과 각 가중치의 곱, 그리고 이들의 합 + 편향**
  - H1에 입력되는 값 (편의상 z_1으로 표현)
    - $z_1 = b_{1,1} + w_{1,1} \cdot 34 + w_{2,1} \cdot 5$
  - H2에 입력되는 값은?
    - z_2 =?
- 은닉 노드의 출력값
  - 은닉 노드(그리고 출력 노드)는 대부분의 경우 입력 받은 값을 그대로 출력하지 않는다!
  - 입력된 값을 특정한 형태로 변환 시킴 => 이러한 목적으로 사용되는 함수를 활성화 함수 (activation function)이라고 함

신경망 작동 원리

- 활성화 함수 (activation function)
  - 보통 은닉노드와 출력 노드에 존재하며, 해당 노드에 입력된 값을 변환하여 출력하는 역할을 함
  - 출력되는 값은 해당 노드가 정답을 맞히는데 기여하는 정도를 반영
  - 여기서는 $f(z)$ 라고 표현
    - z는 해당 노드에 입력되는 값
    - 즉, z를 입력받아서 $f(z)$를 출력
  - 보통 f는 비선형 함수 => **독립변수와 종속변수 간에 존재할 수 있는 비선형 관계를 파악하기 위해서**
    - **선형함수를 여러개 사용하는 것은 별 의미가 없음 (즉, 하나의 선형함수를 사용한 것과 같은 효과)**
  - 앞 예제에서 H1과 H2 노드의 경우
    - H1: z_1을 입력받고 f(z_1)을 출력
    - H2: z_2를 입력받고 f(z_2)을 출력
    - 다음 페이지의 그림 처럼 표현될 수 있음

신경망 작동 원리

- 활성화 함수 (cont'd)



신경망 작동 원리

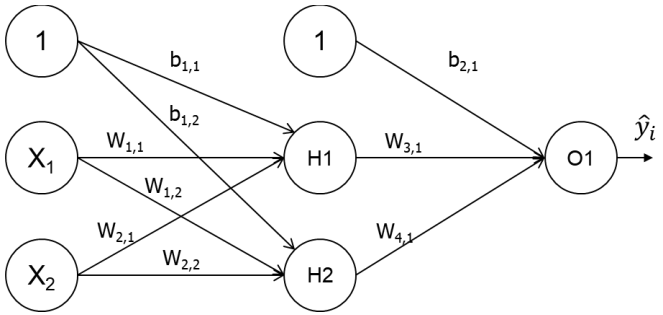
- 출력노드 (O1)에 입력되는 값
  - $z_3 = b_{2,1} + w_{3,1} \cdot f(z_1) + b_{2,1} + w_{4,1} \cdot f(z_2)$
- 출력노드의 활성화 함수
  - 출력 노드는 종속변수의 형태에 따라서 활성화 함수가 있을 수도 있고, 없을 수도 있음
  - 회귀문제: 없음 (또는 항등함수 즉, $z_3 = f(z_3)$)
  - 분류문제
    - 보통 소프트맥스 함수 사용
    - 소프트맥스는 확률값을 리턴
- 아파트 가격 예측의 문제 (회귀문제)
  - $\hat{y}_i = b_{2,1} + w_{3,1} \cdot f(z_1) + w_{4,1} \cdot f(z_2) = z_3$

신경망 작동 원리

- 비용함수
  - 회귀문제: MSE 등
    - $\frac{1}{N}\sum_i(y_i - \hat{y}_i)^2$
    - 아파트 가격 문제의 경우
      - y_i: i 번째 관측치의 실제 y값
      - $\hat{y}_i$: i 번째 관측치에 대한 (모형을 통한) 예측치
        - $\hat{y}_i = b_{2,1} + w_{3,1} \cdot f(z_{i,1}) + b_{2,1} + w_{4,1} \cdot f(z_{i,2})$
  - 비용함수는 파라미터에 대한 함수
  - 학습을 위해 경사하강법 사용

신경망 작동 원리

- Recap: 신경망에서의 학습
  - 전통적인 기계학습에서의 학습의 의미와 동일
  - 즉, 비용함수를 최소화하는 파라미터의 값을 찾는 것
  - 아파트 가격 문제의 경우
    - $\frac{1}{N}\sum_i(y_i - \hat{y}_i)^2$를 최소화하는 모형의 파라미터 (b, w 등)의 값을 찾는 것



신경망 작동 원리

- 분류 문제의 경우
  - 예: 폐암 여부
    - 폐암에 걸렸으면 y = 1, 그렇지 않으면 y = 0
  - 학습 데이터 (예)

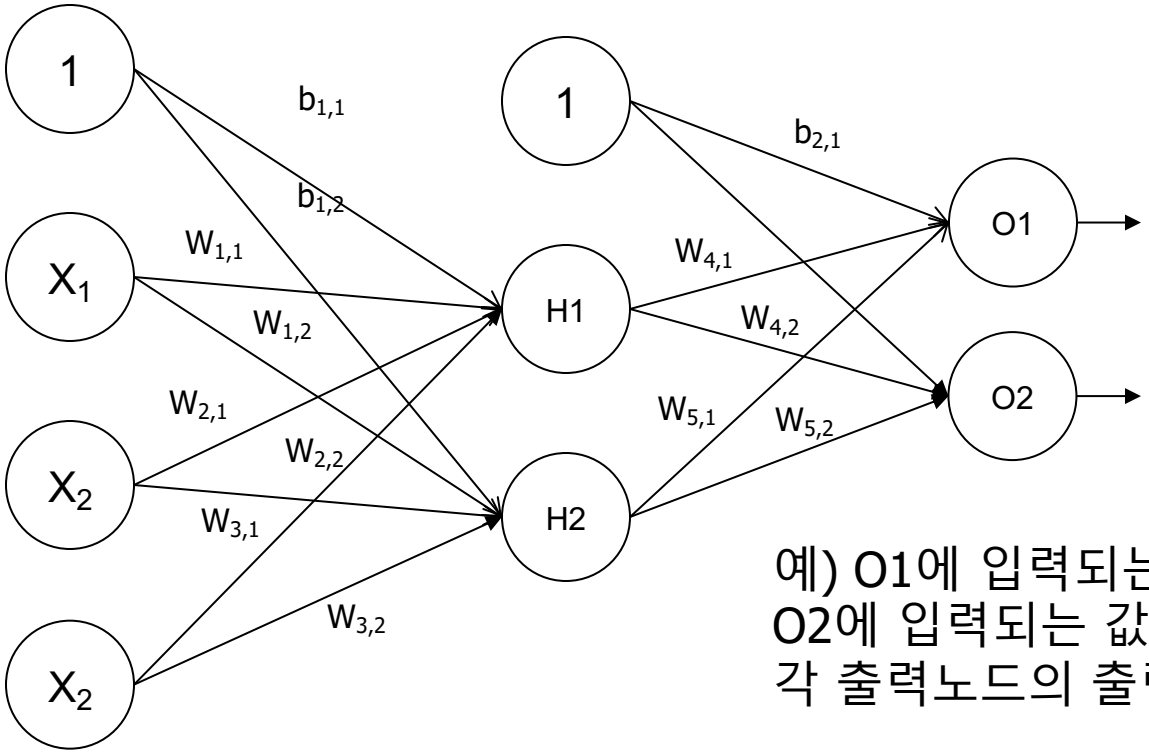
ID	나이 (X1)	흡연여부 (X2)	도시거주여부 (X3)	폐암 여부 (y)
1	34	1	0	1
2	60	0	1	1
3	55	0	0	0

신경망 작동 원리

- 분류 문제의 경우 (cont'd)
  - 비용함수: 교차 엔트로피
    - $E = -\left[\sum_{i=1}^{N} y_i \ln p(y_i = 1) + (1 - y_i) \ln p(y_i = 0)\right]$
    - $y_i \rightarrow$ 각 관측치의 실제 종속변수 값
    - $y_i \in \{0,1\}$
    - $p(y_i = 1), p(y_i = 0) \rightarrow$ 모형을 통해서 예측되는 값

신경망 작동 원리

- 분류문제의 예 (cont'd)



- i번째 노드에서 출력되는 값

$$S(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

- z_i는 i번째출력노드에 입력되는 값
- $S(z_i)$은 확률값을 의미

예) O1에 입력되는 값이 10이고,
O2에 입력되는 값이 20 이라면,
각 출력노드의 출력값은 얼마인가?

Activation functions

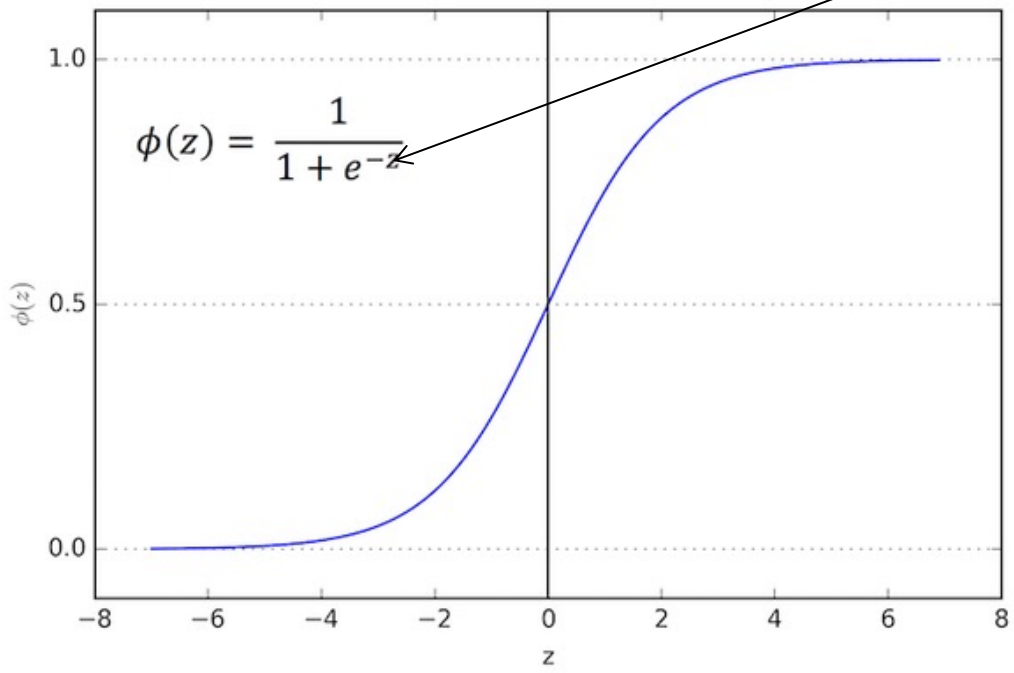
Deep learning

- Activation function (활성화함수)
  - f(z)
    - z 값을 받아서 특정 구간의 값으로 변환, 이렇게 변환된 값이 다음 layer node의 input 값으로 사용됨.
  - 역할
    - **특정한 노드가 종속변수의 값을 예측하는데 기여하는 정도 반영**
      - **보통 기여를 많이 하면 더 큰 값을 출력**
    - **비선형 함수 사용: IVs와 DV간 비선형 관계 파악**
    - **미분이 쉬어야 함**
  - 주요 활성화 함수
    - Logistic 함수 (also known as Sigmoid function)
    - Hyperbolic Tangent (tanh) 함수
    - Rectified Linear Unit (Relu) 함수
    - Leaky Relu 함수

Deep learning

- Activation function (활성화함수)
  - Sigmoid function: 0 ~ 1

$$z = b_{1,1} + w_{1,1} \cdot X_1 + w_{2,1} \cdot X_2$$



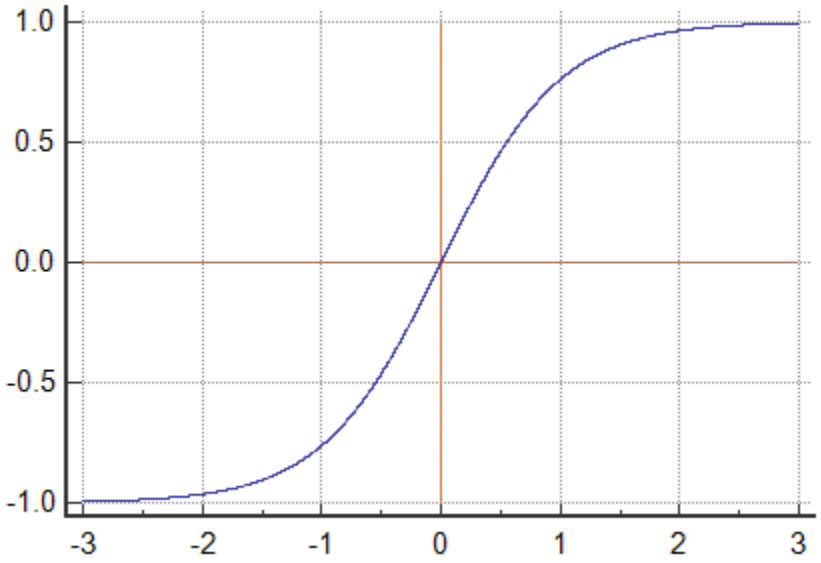
- What is the derivative of the sigmoid function?
- What is the value of the derivative at z=0?

Deep learning

- Activation function (활성화함수)
  - Hyperbolic tangent (tanh): -1 ~ 1

$$\tanh(z) = \frac{\sinh(z)}{\cosh(z)} = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$z = b_{1,1} + w_{1,1} \cdot X_1 + w_{2,1} \cdot X_2$$

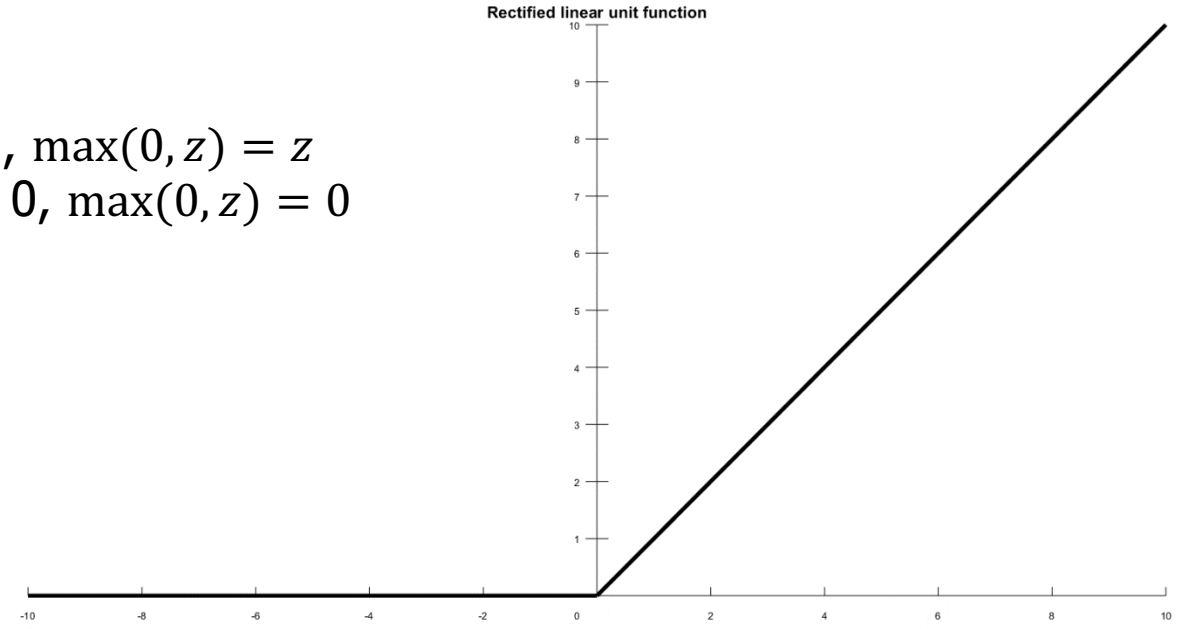


Deep learning

- Activation function (활성화함수)
  - Rectified linear unit (Relu): 0 ~

$$f(z) = \max(0, z) \qquad z = b_{1,1} + w_{1,1} \cdot X_1 + w_{2,1} \cdot X_2$$

when z > 0, $\max(0, z) = z$
when z <= 0, $\max(0, z) = 0$

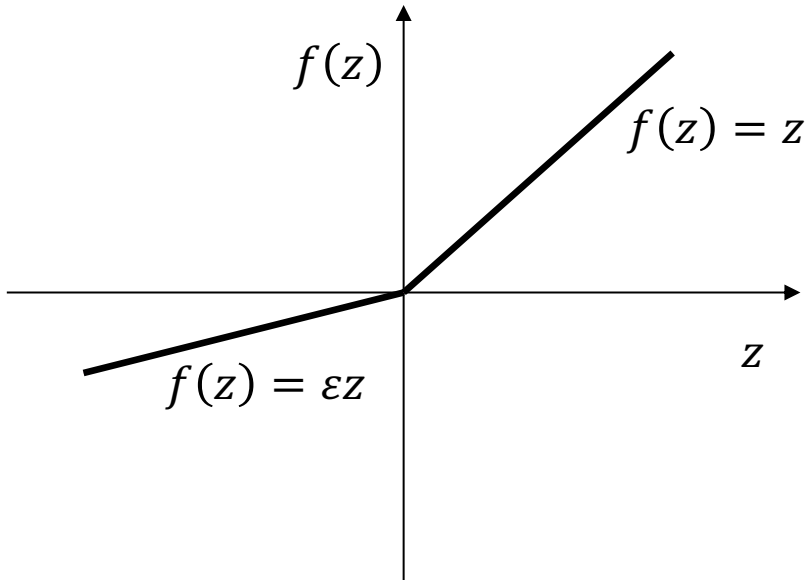


Deep learning

- Activation function (활성화함수)
  - Leaky Relu

$$\varepsilon = 0.01\ in\ general$$

$$f(z) = \max(\varepsilon z, z)$$

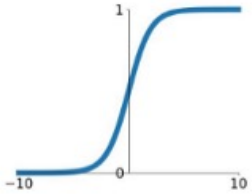


Deep learning

- Activation functions

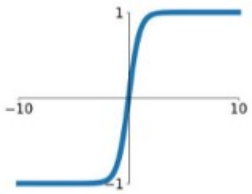
Sigmoid

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



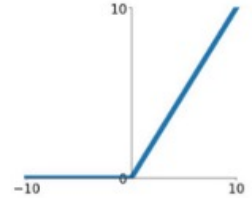
tanh

$\tanh(x)$



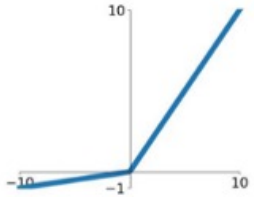
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

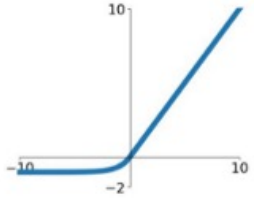


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

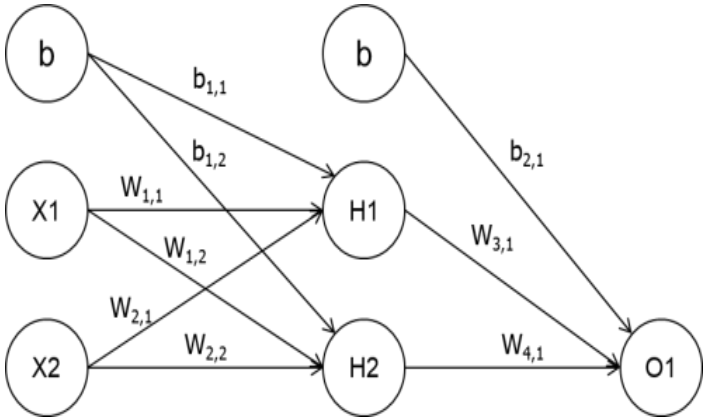
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



신경망 작동 원리

- 아파트 가격 예측 문제 Revisit
  - 신경망 구조



  - 은닉 노드의 활성화 함수
    - sigmoid function: $f(z) = \frac{1}{1+e^{-z}}$

신경망 작동 원리

- 아파트 가격 예측 문제 Revisit
  - Then what happens?
  - 각 은닉 노드에 입력되는 값은 동일 즉, for the first data point (i.e., X1=34, X2=5)
    - H1에 입력되는 값
      - $z_1 = b_{1,1} + w_{1,1} \cdot 34 + w_{2,1} \cdot 5$
    - H2에 입력되는 값
      - $z_2 = b_{1,2} + w_{1,2} \cdot 34 + w_{2,2} \cdot 5$
  - 각 은닉 노드에서 출력되는 값은?
    - H1에서 출력되는 값
      - $f(z_1)$ $(=\frac{1}{1+e^{-z_1}})$
    - H2에서 출력되는 값
      - $f(z_2)$ $(=\frac{1}{1+e^{-z_2}})$

신경망 작동 원리

- 아파트 가격 예측 문제 Revisit
  - 각 관측치의 종속변수 예측치
    - 오차 계산
  - MSE 비용함수 계산
- Optimization problem
  - 비용함수를 minimize 하는 파라미터의 값 찾기
    - 경사하강법

신경망에서의 경사하강법

신경망 작동 원리

- 학습
  - 즉, 비용함수를 최소화하는 가중치의 값 찾기
  - Optimization problem
    - 주요 방법
      - Normal equation: appropriate when the cost function is convex
        - But, usually the cost function of DL is not convex (a lot more complex)
        - or too many parameters
      - Gradient descent
        - 경사하강법은 다음과 같은 식을 통해서 가중치들의 값을 여러번 업데이트 하면서 비용함수의 값을 최소화하는 가중치 값을 찾는 방법 ($w_{1,1}$의 경우)
        - $w_{1,1,new} = w_{1,1,current} - \eta \frac{\partial \mathrm{E}}{\partial w_{1,1}}$

Review

- 신경망
  - 딥러닝은 신경망 모형을 기반으로 함
  - 신경망의 구조
    - 입력층, 은닉층, 출력층
    - 은닉층의 존재 → 다른 기계학습 알고리즘과의 가장 큰 차이
    - 각 층 혹은 각 층의 노드의 역할
  - 은닉 노드
    - 활성화 함수 존재
    - 같은 층에 있는 노드들은 같은 활성화함수 사용
    - 활성화함수 보통 비선형 함수를 사용
    - 각 은닉노드가 종속변수를 예측하는데 얼마만큼의 기여를 하는지를 반영

Review

- 신경망의 작동원리 (지도학습의 경우)
  - 여느 기계학습 알고리즘과 거의 동일
    - 참고: 신경망에서는 파라미터는 가중치와 편향으로 구분
- 비용함수
  - 문제의 종류에 따라 구분
  - 실제의 종속변수 값과 모형을 통한 예측치로 구성
  - 회귀문제의 경우
    - 종속변수 값의 예측치 출력
  - 분류문제의 경우
    - 각 노드에서 출력되는 값 → 종속변수가 특정한 값을 갖을 확률
    - 이는 출력노드에 입력된 값이 확률 값으로 변환되어 출력된다는 것을 의미
    - 그렇다면 확률값은 어떻게 계산되는가?

신경망 작동 원리

- 경사하강법의 종류
  - 경사하강법은 한번 업데이트 할 때 사용되는 데이터 포인트의 양에 따라 크게 3가지로 구분
    - Batch Gradient Descent
      - 업데이트를 한번 할 때 마다 (비용함수의 값을 계산하기 위해서) 전체 데이터를 모두 사용하는 방법
      - 많은 메모리 공간 필요, 계산하는데 시간이 오래 걸림
    - Stochastic Gradient Descent
      - 업데이트를 한번 할때, 랜덤하게 선택된 하나의 데이터 포인트만을 사용
      - 업데이트 되는 방향이 일정하지 않음, 수렴하는데 시간이 오래 걸림
    - Mini-batch Gradient Descent
      - 한번 업데이트할 때 하나의 전체 데이터의 일부 (이를 mini-batch 라고 함)를 사용하고, 그 다음 업데이트를 하기 위해서 또 다른 mini-batch를 이용해서 비용함수 (혹은 기울기)의 값을 구하는 방법
    - https://hyunw.kim/blog/2017/11/01/Optimization.html 참고

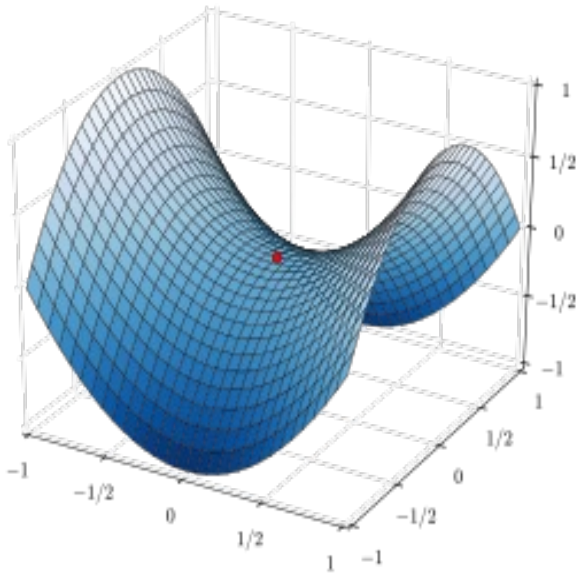
신경망 작동 원리

- 경사하강법
  - 비용함수: MSE
  - 비용함수의 형태
    - For the total data points (in the training dataset, batch data, # of data points = N)
      - $\mathrm{E} = \frac{1}{N}\sum_{i=1}^{N} E_i = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$
    - For a single data point
      - $E_i = (y_i - \hat{y}_i)^2$
    - For a mini batch (i.e., # of data points = h, where 1<h<N)
      - $\mathrm{E} = \frac{1}{h}\sum_{i=k+1}^{k+h} E_i = \frac{1}{h}\sum_{i=k+1}^{k+h}(y_i - \hat{y}_i)^2$

OPTIMIZER의 종류

신경망 작동 원리

- 기본적 방법의 문제
  - 1) 비용함수의 saddle point를 잘 벗어나지 못한다.
    - 딥러닝에서는 파라미터가 많아서 local optima는 많이 존재하지 않는다. 대신 오른쪽 그림과 같은 saddle point가 많이 존재한다. 즉, saddle point를 어떻게 벗어나느냐가 중요한 문제이다.
  - 2) 속도가 느리다.



신경망 작동 원리

- 기본 경사 하강법의 제한점
  - 기본 업데이트 공식 => $w_{j,i,new} = w_{j,i,current} - \eta \frac{\partial \mathrm{E}}{\partial w_{j,i}}$
  - 제한점
    - 지금까지 업데이트된 정도가 반영되지 않는다.
    - 업데이트 횟수와 상관없이 learning rate가 고정되어 있다.

신경망 작동 원리

- Optimizer의 종류와 특성
  - 이러한 문제를 보완하기 위해서 다양한 형태의 optimizer 제안
  - 기본 업데이트 공식을 약간씩 수정/보완한 방법들임
  - 대표적 Optimizers
    - Momentum
    - Adagrad (Adaptive Gradient)
    - RMSprop (Root Mean Square Propagation)
    - Adam
    - 다른 optimizer들은 https://ruder.io/optimizing-gradient-descent/를 참고

Optimizer

- Momentum
  - 이전 update 정보를 기억, 현재 update에 반영하는 방법

$$v_t = \gamma v_{t-1} + \eta \frac{\partial \mathrm{E}}{\partial w_{t,i}}$$

$$w_{t+1,i} = w_{t,i} - v_t$$

이번에 계산된 gradient

이전 update 내용

$0 < \gamma < 1$

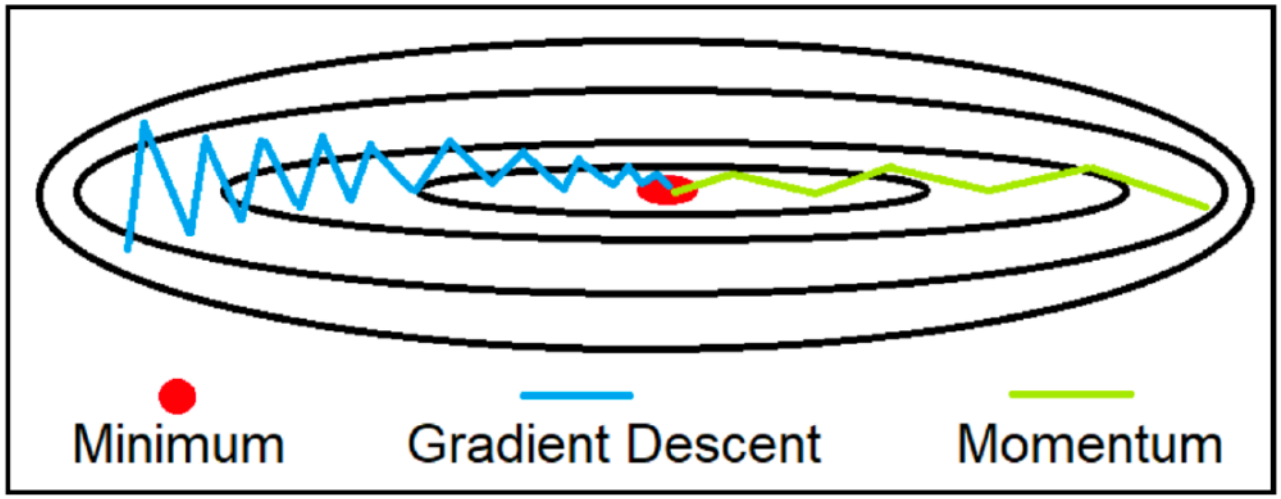
$$v_{t-1} = \gamma v_{t-2} + \eta \frac{\partial \mathrm{E}}{\partial w_{t-1,i}}$$

가장 최근 정보를 더 많이 반영

  - 장점 (GD 에 비해서)
    - local minimum/saddle point 를 잘 피한다.
    - 수렴하는 속도가 빠르다.

Optimizer

- Momentum (cont'd)
  - 효과



Optimizer

이러한 방법을 learning rate decay라고 부름

- Adagrad (Adaptive Gradient)
  - GD => 업데이트 횟수와 상관없이 learning rate를 동일하게
  - 하지만, Adagrad는 다르게
    - 지금까지 업데이트 된 정도를 반영한다.
    - 지금까지 업데이트가 많이된 parameter는 learning rate를 작게

$$w_{t+1,i} = w_{t,i} - \frac{\eta}{\sqrt{G_t + \epsilon}} \cdot g_{t,i}$$

지금까지 gradient의 합

$$\sum_{k=1}^{t} g_{k,i}^2$$

$$g_{t,i} = \frac{\partial \mathrm{E}}{\partial w_i}$$

ϵ is a smoothing term that avoids divisio

  - 주요 문제
    - Gt가 갈수록 커진다 => 업데이트가 거의 발생하지 않는다.
    - Adadelta, RMSprop

Optimizer

- RMSprop (Root Mean Square Propagation)
  - Adagrad의 확장 버전
    - 무조건적으로 줄어드는 learning rate 문제를 보완
    - 합이 아니라 평균을 사용 [과거 gradients 들의 평균]

$$w_{i,t+1} = w_{i,t} - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \frac{\partial \mathrm{E}}{\partial w_i}$$

$$E[g^2]_t = \rho E[g^2]_{t-1} + (1 - \rho) g_t^2$$

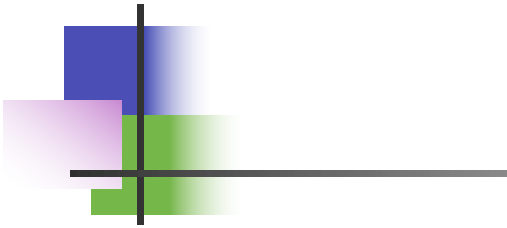
이를 moving average라고 함
ρ는 보통 0.9 정도

Optimizer

- Adam
  - RMSprop + momentum
  - 다음과 같이 표현될 수 있음
    - Momentum 방식
      - $v_t = \gamma v_{t-1} + \eta \frac{\partial \mathrm{E}}{\partial w_i}$
      - $w_{t+1,i} = w_{t,i} - v_t$
    - RMSprop 방식
      - $w_{t+1,i} = w_{t,i} - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \frac{\partial \mathrm{E}}{\partial w_{t,i}}$
    - 둘을 결합하면,
      - $w_{t+1,i} = w_{t,i} - \frac{v_t}{\sqrt{E[g^2]_t + \epsilon}}$
  - 자세한 내용은
    - Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
    - https://ruder.io/optimizing-gradient-descent/index.html#adam

Optimizer

- Optimizer들의 비교
  - http://ruder.io/optimizing-gradient-descent/index.html#visualizationofalgorithms
- Which optimizer to use?
  - https://ruder.io/optimizing-gradient-descent/index.html#whichoptimizertouse



Q & A