인공신경망

- git 밑바닥부터 시작하는 딥러닝
  https://github.com/youbeebee/deeplearning_from_scratch

- git 딥다딥러닝
  https://github.com/rickiepark/do-it-dl

1. 인공신경망(ANN) 및 퍼셉트론

인공지능(AI)의 역사

1943년
워렌 맥클록과 월터 피츠, 전기 스위치처럼 켜고 끄는 기초기능의 인공신경을 그물망 형태로 연결하면 사람의 뇌에서 동작하는 아주 간단한 기능을 흉내낼 수 있음을 증명

1950년
앨런 튜링, 기계가 인간과 얼마나 비슷하게 대화할 수 있는지를 기준으로 기계에 지능이 있는지를 판별하는 튜링 테스트 제안

1956년

다트머스 회의에서 인공지능 용어 처음 사용. "학습의 모든 면 또는 지능의 다른 모든 특성을 기계로 정밀하게 기술할 수 있고 이를 시뮬레이션할 수 있다"

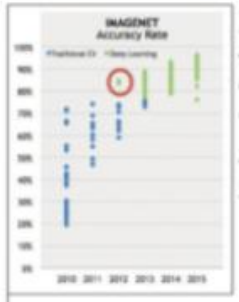
1958년
프랭크 로센블래트, 뇌신경을 모사한 인공신경 뉴런 '퍼셉트론' 제시

1970년대
AI 연구가 기대했던 결과를 보여주지 못하자 대규모 투자가 중단되며 암흑기 도래

1980년대
전문가들의 지식과 경험을 데이터베이스화해 의사결정 과정을 프로그래밍화한 '전문가 시스템' 도입. 그러나 관리의 비효율성과 유지·보수의 어려움으로 한계

1997년
IBM 딥블루, 체스 챔피언 개리 카스파로프와의 체스 대결에서 승리

2006년
제프리 힌튼 토론토대 교수, 딥러닝 알고리즘 발표



2012년
국제 이미지 인식 경진대회 '이미지넷'에서 딥러닝 활용한 팀이 우승하며 획기적 전환점

2014년
구글, 딥마인드 인수

2016년
구글 알파고, 이세돌에게 승리

1. 인공신경망(ANN) 및 퍼셉트론

1) ANN(Artificial Neural Networks) 개념 (https://needjarvis.tistory.com/178?category=933540)

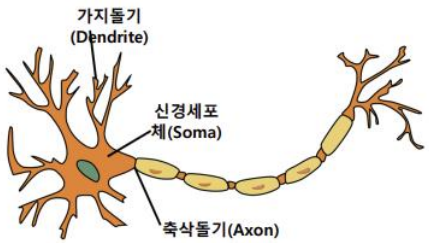
- 인간의 신경구조를 복잡한 스위치들이 연결된 네트워크로 표현가능할수 있음에 기초함.
  - 1943년, “A logical calculus of the ideas immanent in nervous activity"
  - 1958년, "The perceptron: A probabilistic model for information storage and organization in the brain"
    - ✓ 선형분류 가능한 퍼셉트론(Perceptron)
    - ✓ feed-forword 뉴럴네트워크

퍼셉트론	input과 weight들의 곱을 모두 더한뒤 활성함수를 적용

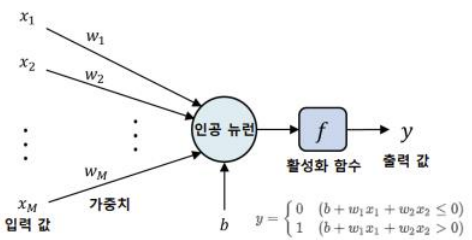
1 인공신경망(Artificial Neural Network)

핵심요약

인공신경망(ANN)은 기계학습과 인지과학 분야에서 고안한 학습 알고리즘이다. 신경세포의 신호 전달체계를 모방한 인공뉴런(노드)이 학습을 통해 결합 세기를 변화시켜 문제를 해결하는 모델 전반을 가리킨다.



〈신경세포〉



〈퍼셉트론〉

인공신경망의 시작, 퍼셉트론

퍼셉트론은 신경세포 뉴런들이 신호, 자극 등을 받아 어떠한 임계값(threshold, θ)을 넘어서면 그 결과를 전달하는 신경세포의 신호 전달 과정을 착안하여 만들어졌다. 이를 수학적인 기호로 모델링 한 것이 퍼셉트론인데, 퍼셉트론은 각각의 가중치(ω_i)의 크기를 적절히 조절하여 입력 신호(χ_i)의 크기를 정한다. 입력 신호들의 합에 활성화 함수를 거쳐 나온 결괏값을 전달함으로써 원하는 결괏값을 얻는 방식으로 동작한다.

아래 그림은 단층 퍼셉트론으로 AND, OR, NAND와 같은 비교적 간단한 문제를 해결할 수 있는 예시이다. 입력값에 대해 정해진 결괏값을 출력할 수 있는 가중치의 경우의 수는 무수히 많으며(퍼셉트론 그림에서 출력 식 참고), 주어진 연산을 만족하는 가중치의 조합을 찾음으로써 문제를 푼다고 할 수 있다.

- 파이썬을 활용한 데이터.AI분석(건강보험심사평가원) p114

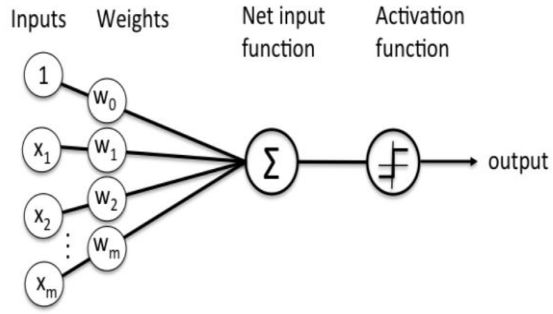
1. 인공신경망(ANN) 및 퍼셉트론

1) ANN(Artificial Neural Networks) 개념

2012년 딥러닝 기술의 적용으로 심층신경망 발전

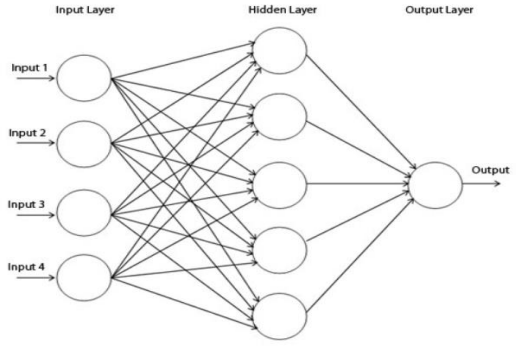
1세대 (1943 ~1986)

- 인공신경망의 개념과 퍼셉트론등장
- 논리와 규칙 기반의 전문가 시스템



2세대 (1986 ~2006)

- MLP와 Backpropagatior Algorithm
- 사람의 신경세포구조를 본떠 만든 뉴럴렛 (인공신경망)



▪ 구글검색

' 이미지넷 프로젝트 '

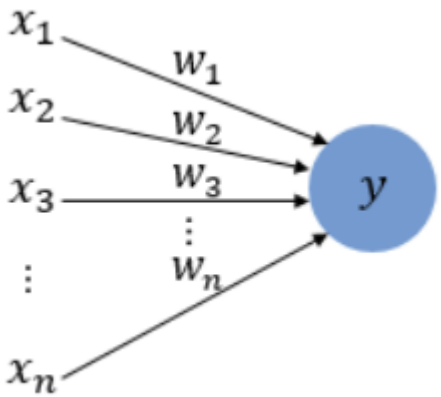
1969년 Minsky의『Perceptrons』에 의해 단층 퍼셉트론 모델의 한계점이 분석되며, 신경망 관련 연구는 15년 동안 침체기를 맞이했다.

1. 인공신경망(ANN) 및 퍼셉트론

2) 퍼셉트론: 퍼셉트론은 인공지능의 초반 기계학습 방법 (출처: https://wikidocs.net/24958)

퍼셉트론(Perceptron)은 프랑크 로젠블라트(Frank Rosenblatt)가 1957년에 제안한 초기 형태의 인공 신경망으로 다수의 입력으로부터 하나의 결과를 내보내는 알고리즘으로 퍼셉트론은 w0, x0 바이어스와 w1,w2의 가중치를 계속 변경하는 심층신경망의 기본이 됨.

퍼셉트론은 실제 뇌를 구성하는 신경 세포 뉴런의 동작과 유사한데, 신경 세포 뉴런의 그림을 먼저 보도록 하겠습니다. 뉴런은 가지돌기에서 신호를 받아들이고, 이 신호가 일정치 이상의 크기를 가지면 축삭돌기를 통해서 신호를 전달합니다.



x는 입력값을 의미하며, w는 가중치(Weight), y는 출력값입니다. 그림 안의 원은 인공 뉴런에 해당됩니다. 실제 신경 세포 뉴런에서의 신호를 전달하는 축삭돌기의 역할을 퍼셉트론에서는 가중치가 대신합니다. 각각의 인공 뉴런에서 보내진 입력값 x는 각각의 가중치 w와 함께 종착지인 인공 뉴런에 전달되고 있습니다.

각각의 입력값에는 각각의 가중치가 존재하는데, 이때 가중치의 값이 크면 클수록 해당 입력 값이 중요하다는 것을 의미합니다.

각 입력값이 가중치와 곱해져서 인공 뉴런에 보내지고, 각 입력값과 그에 해당되는 가중치의 곱의 전체 합이 임계치(threshold)를 넘으면 종착지에 있는 인공 뉴런은 출력 신호로서 1을 출력하고, 그렇지 않을 경우에는 0을 출력합니다. 이러한 함수를 계단 함수(Step function)라고 하며, 아래는 그래프는 계단 함수의 하나의 예를 보여줍니다.

이때 계단 함수에 사용된 이 임계치값을 수식으로 표현할 때는 보통 세타(Θ)로 표현합니다. 식으로 표현하면 다음과 같습니다.

$$if \sum_i^n w_i x_i \ \geq \theta \rightarrow y = 1$$

$$if \sum_i^n w_i x_i \ < \theta \rightarrow y = 0$$

퍼셉트론의 활성화 함수는 계단 함수

3) 단층퍼셉트론(논리게이트) : 매컬리-피츠의 신경회로망

단층 퍼셉트론은 값을 보내는 단계과 값을 받아서 출력하는 두 단계로만 이루어집니다. 이때 이 각 단계를 보통 층(layer)이라고 부르며, 이 두 개의 층을 입력층(input layer)과 출력층(output layer)이라고 합니다.

- 논리연산 : 거짓과 참이라는 두 상태를 지닌 x1과 x2가 주어지면, 그 둘의 상태에 따라 출력값이 결정되는 것.

AND 게이트는, x1과 x2가 모두 True(1) 이어야 결과값 y가 1이됨.

x_1	x_2	y
0	0	0
1	0	0
0	1	0
1	1	1

$(w_1, w_2, \theta) = (0.5, 0.5, 0.7)$

OR 게이트는, x1과 x2 중 한개라도 True(1) 이면 결과값 y가 1이됨.

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	1

$(w_1, w_2, \theta) = (0.5, 0.5, 0.2)$

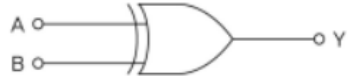
Not은 y결과의 반대 NAND는 AND의 반대 값출력(0은1로, 1은 0으로)

x_1	x_2	y
0	0	1
1	0	1
0	1	1
1	1	0

$(w_1, w_2, \theta) = (-0.5, -0.5, -0.7)$

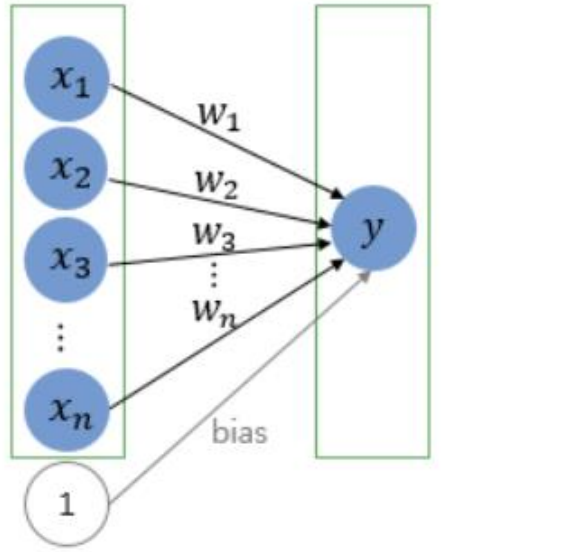
〈단층 퍼셉트론에서 AND, OR, NAND 문제 및 가중치 예시 〉

XOR은 두값이 서로 다르면(0,1 또는 1,0) 일때 y출력값이 1



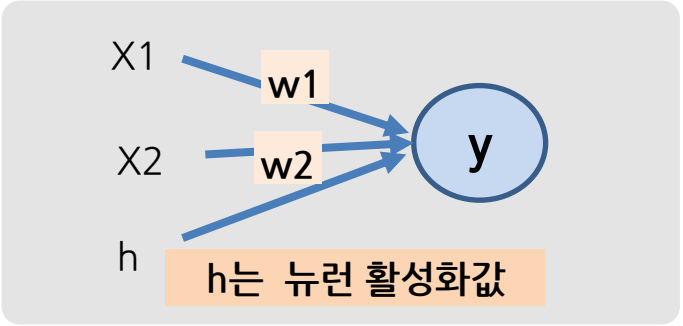
입력		출력
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

XOR 게이트의 진리표



3) 단층퍼셉트론(논리게이트) : 매컬리-피츠의 신경회로망

- **신경세포를 모방한 최초의 신경모형(사람이 미리 파라미터 정함)**
  - 여러개의 입력에 대해 하나의 값을 출력
  - 입력 x에 결합 가중치 Wi를 곱한 값의 합을 출력 y로 정함
    이 합이 임계치(파라미터) h보다 크면 1출력, 작으면 0 출력
  - And나 OR와 같은 논리 연산 표현 가능
  - 입력 1개에 출력1개, 혹은 입력 2개에 출력 1개가 되는 모형



▶AND연산

w1=1, w2=1, h=1.5

입력 x1	입력 x2	AND 출력	y	
0	0	0	0	(0*1+0*1)-1.5=-1.5
0	1	0	0	(0*1+1*1)-1.5=-0.5
1	0	0	0	(1*1+0*1)-1.5=-0.5
1	1	1	1	(1*1+1*1)-1.5=0.5

$$y = f\left(\sum_{y=1}^{n} \omega_i x_i - h\right)$$

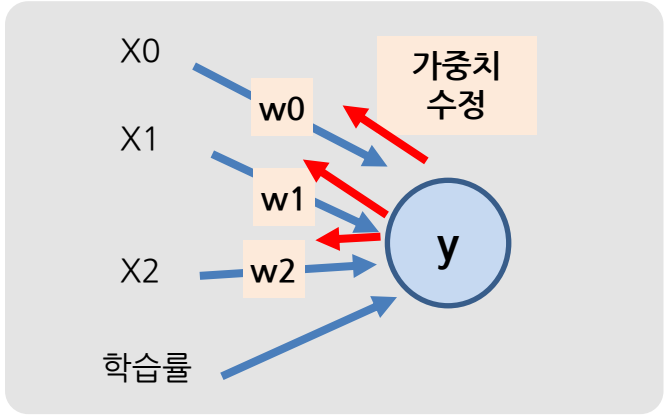
▶OR 연산

w1=1, w2=1, h=0.5

입력 x1	입력 x2	OR 출력	y	
0	0	0	0	(0*1+0*1) > h(0.5) False
0	1	1	1	(0*1+1*1) > h(0.5) True
1	0	1	1	(1*1+0*1) > h(0.5) True
1	1	1	1	(1*1+1*1) > h(0.5) True

$$y = f\left(\sum_{y=1}^{n} \omega_i x_i - h\right)$$

3) 단층퍼셉트론(논리게이트) : 매컬리-피츠의 신경회로망



- **파라미터를 학습을 통해 결정함**
  - 학습 표본과 그에 대한 정답 신호를 세트로 하여 학습(지도학습)
  - 출력값과 정답 신호 사이의 오차를 수정해 나가는 학습방법 (오차 수정학습)을 취함

[오차수정학습식] $w_i < -w_i + \alpha(r-y)x_i$
또는 $wi = wi +$ 학습률(target-f(net)) xi

α는 업데이트를 결정하는 파라미터(학습률) α를 크게설정하면 오차가 크게 반영되고 작게 설정하면 오차의 반영값이 작아짐.
출력 y와 정답신호 r이 일치하지 않는 경우에는 w값과 h를 수정

▶ 오차수정1회->AND연산

x1	x2	AND (r)	x0=-1, w0=0.3, w1=0.4, w2=0.1	y값: 결과(0이하 0, 0초과1)		
0	0	0	-1*0.3+0*0.4+0*0.1	-0.3+0+0=-0.3	0	성공
0	1	0	-1*0.3+0*0.4+1*0.1	-0.3+0+0.1=-0.2	0	성공
1	0	0	-1*0.3+1*0.4+0*0.1	-0.3+0.4+0=0.1	1	실패
1	1	1	-1*0.3+1*0.4+1*0.1	-0.3+0.4+0.1=0.2	1	성공

y값이 맞지않는 오차가 생긴 x1=1, x2=0 값으로 w값 재계산
학습률=0.05 으로 정했을때

w0: 0.3+0.05(0-1)*-1 =0.35
w1: 0.4+0.05(0-1)*1 =0.35
w2: 0.1+0.05(0-1)*0 =0.1

1. 인공신경망(ANN) 및 퍼셉트론

3) 단층퍼셉트론(논리게이트) : 매컬리-피츠의 신경회로망

[오차수정학습식]

▶ 오차수정1회->AND연산 $w_i < -w_i + \alpha(r-y)x_i$ 또는 $wi = wi +$ 학습률(target−f(net)) xi

x1	x2	AND r또는 target	x0=-1, w0=0.3, w1=0.4, w2=0.1	결과(0미만 0, 0이상1) y 또는 f(net)		
0	0	0	-1*0.3+0*0.4+0*0.1	-0.3+0+0=-0.3	0	성공
0	1	0	-1*0.3+0*0.4+1*0.1	-0.3+0+0.1=-0.2	0	성공
1	0	0	-1*0.3+1*0.4+0*0.1	-0.3+0.4+0=0.1	1	실패
1	1	1	-1*0.3+1*0.4+1*0.1	-0.3+0.4+0.1=0.2	1	성공

y값이 맞지않는 오차가 생긴 x1=1, x2=0 값으로 w값 재계산

학습률=0.05 으로 정했을때

w0: 0.3+0.05(0-1)*-1 =0.35
w1: 0.4+0.05(0-1)*1 =0.35
w2: 0.1+0.05(0-1)*0 =0.1

▶ 오차수정2회-> AND연산

x1	x2	AND r또는 target	x0=-1, w0=0.35, w1=0.35, w2=0.1	결과(0미만 0, 0이상1) y 또는 f(net)		
0	0	0	-1*0.35+0*0.35+0*0.1	-0.35+0+0=-0.35	0	성공
0	1	0	-1*0.35+0*0.35+1*0.1	-0.35+0+0.1=-0.25	0	성공
1	0	0	-1*0.35+1*0.35+0*0.1	-0.35+0.35+0=0	1	실패
1	1	1	-1*0.35+1*0.35+1*0.1	-0.35+0.35+0.1=0.1	1	성공

오차수정(2회)

학습률=0.05, x1=1, x2=0

w0: 0.35+0.05(0-1)*-1 =0.4
w1: 0.35+0.05(0-1)*1 =0.3
w2: 0.1+0.05(0-1)*0 =0.1

3) 단층퍼셉트론(논리게이트) : 매컬리-피츠의 신경회로망

[오차수정학습식]

▶ 오차수정2회-> AND연산 $w_i < -w_i + \alpha(r-y)x_i$ 또는 $wi = wi +$ 학습률(target−f(net)) xi

x1	x2	AND r또는 target	x0=-1, w0=0.35, w1=0.35, w2=0.1	결과(0미만 0, 0이상1) y 또는 f(net)		
0	0	0	-1*0.35+0*0.35+0*0.1	-0.35+0+0=-0.35	0	성공
0	1	0	-1*0.35+0*0.35+1*0.1	-0.35+0+0.1=-0.25	0	성공
1	0	0	-1*0.35+1*0.35+0*0.1	-0.35+0.35+0=0	1	실패
1	1	1	-1*0.35+1*0.35+1*0.1	-0.35+0.35+0.1=0.1	1	성공

오차수정(2회)
학습률=0.05, x1=1, x2=0

w0: 0.35+0.05(0-1)*-1 =0.4
w1: 0.35+0.05(0-1)*1 =0.3
w2: 0.1+0.05(0-1)*0 =0.1

▶ 오차수정3회-> AND연산

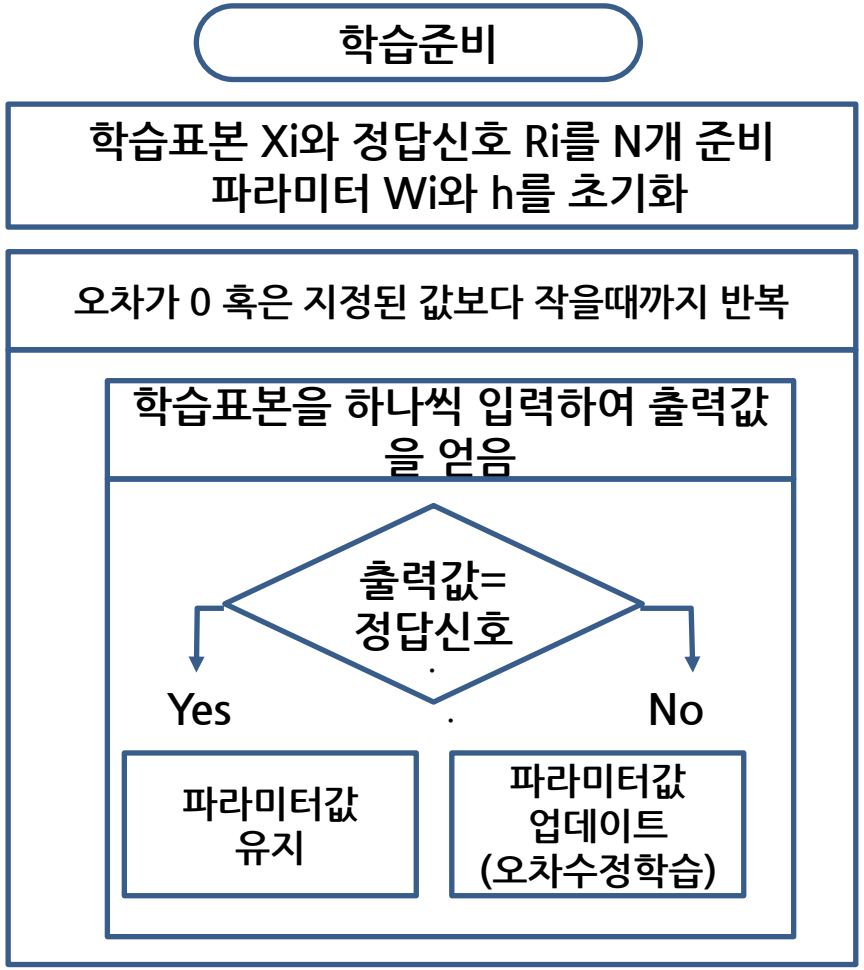
x1	x2	AND r또는 target	x0=-1, w0=0.4, w1=0.3, w2=0.1	결과(0미만 0, 0이상1) y 또는 f(net)		
0	0	0	-1*0.4+0*0.3+0*0.1	-0.4+0+0=-0.4	0	성공
0	1	0	-1*0.4+0*0.3+1*0.1	-0.4+0+0.1=-0.3	0	성공
1	0	0	-1*0.4+1*0.3+0*0.1	-0.4+0.3+0=-0.1	0	성공
1	1	1	-1*0.4+1*0.3+1*0.1	-0.4+0.3+0.1=0	1	성공

최종
학습률=0.05,
w0=0.4, w1=0.3, w2=0.1

4) 단층퍼셉트론 정리

각 노드의 가중치와 입력치를 곱한것을 모두 합한 값이
활성함수에 의해 판단되고 그 값이
임계치(보통 0)보다 크면
뉴런이 활성화되고 결과값으로 1을 출력
뉴런이 활성화되지 않으면 결과값으로 -1 출력

장점	• 오차 수정 학습을 사용 • 자동으로 파라미터 값을 얻음 • 퍼셉트론이 가져온 가장 큰 변화
단점	• 선형 분리(직선한개로만 나누어야함) 가능한 문제만 해결가능 • XOR과같은비선형 불가능



4) 단층퍼셉트론 정리

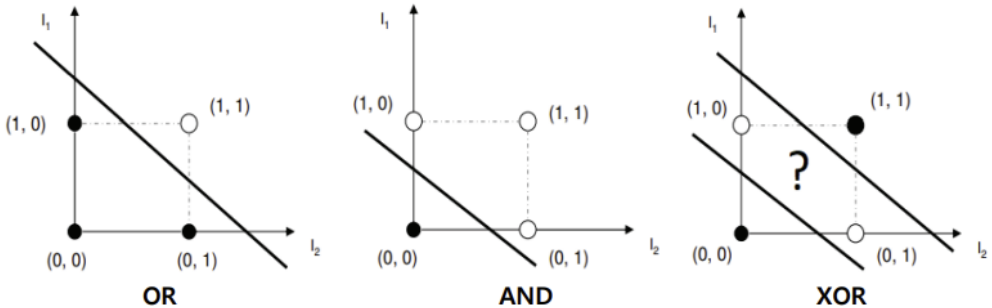
- 단층 퍼셉트론은 델타규칙을 이용해 w1, w2값을 학습하고 결정
- 델타규칙은 Gradient Descent 원리를 이용해 손실함수(Mean Square Error)를 최소화하는 w1, w2 값을 찾는 학습임
  - **w의 변화량의 정도를 결정하는 것은 학습률임(Learning rate)**
  - **학습률 값이 작을수록 w의 변화량이 작아지므로 더 정확한 결과값 도출 가능**

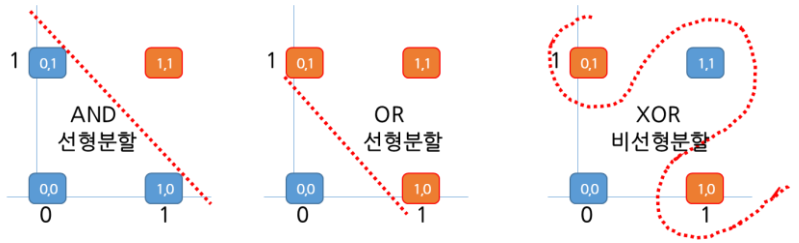
순입력 함수	• **w0*x0+w1*x1+w2+x2**
활성 함수	• $wi = wi +$ **학습률*xi(target-f(net))** • $w_i < -w_i + \alpha(r - y)x_i$

5) 단층퍼셉트론의 한계

단층 퍼셉트론의 한계

앞서 살펴본 단층 퍼셉트론으로 AND, OR, NAND(=NOT AND)와 같은 비교적 단순한 논리 게이트는 구현할 수 있었지만 XOR같은 문제를 풀 수 없는 한계가 있었다. 이는 좌표평면상에 결정 경계(Decision Boundary)를 그어보면 좀 더 명확해지는데, 단층 퍼셉트론으로는 비선형적인 결정 경계를 표현하는 데 한계가 있다는 것을 알 수 있다.





퍼셉트론 등장 (1958년) → 퍼셉트론 한계증명 (1969년) → 다층 퍼셉트론(MLP) (1986년)

- 파이썬을 활용한 데이터.AI분석(건강보험심사평가원) p115

6) 아달라인(Adaline)와 경사하강법 - 1960년에 발표된 Adaptive Linear Neuron

출처: https://m.blog.naver.com/samsjang/220959562205

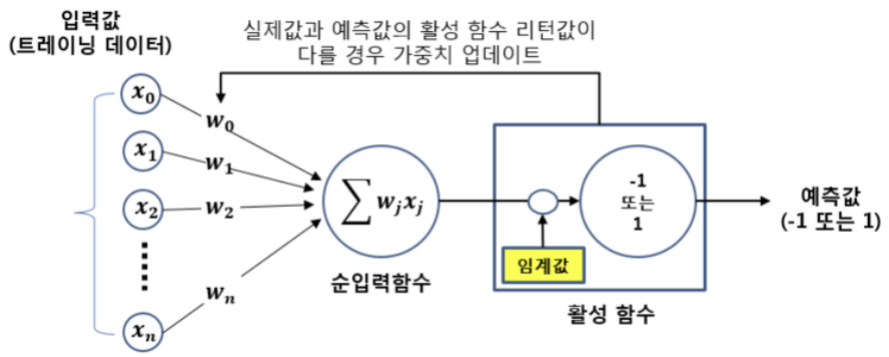
퍼셉트론에서는 활성 함수가 순입력 함수의 리턴값을 임계값과 비교하여 그 결과에 따라 1 또는 -1로 리턴합니다. 퍼셉트론에서 가중치의 업데이트는 트레이닝 데이터에 대한 활성 함수의 리턴값(-1 또는 1)과, 트레이닝 데이터의 실제 결과값(-1 또는 1)이 같은지 다른지에 따라 이루어집니다.

그런데 아달라인에서는 활성 함수가 퍼셉트론에서 처럼 -1, 1의 단순 비교가 아닌 순입력 함수의 리턴값과 실제 결과값 자체의 오차가 최소가 되도록 하는 함수가 됩니다. 아달라인을 발표한 논문에서는 활성함수를 위해 최소제곱법을 이용한 비용함수(cost function) ***J(w)***를 아래와 같이 정의하였고, ***J(w)*** 값이 최소가 되도록 가중치를 업데이트 하는 것이 활성 함수의 핵심입니다.

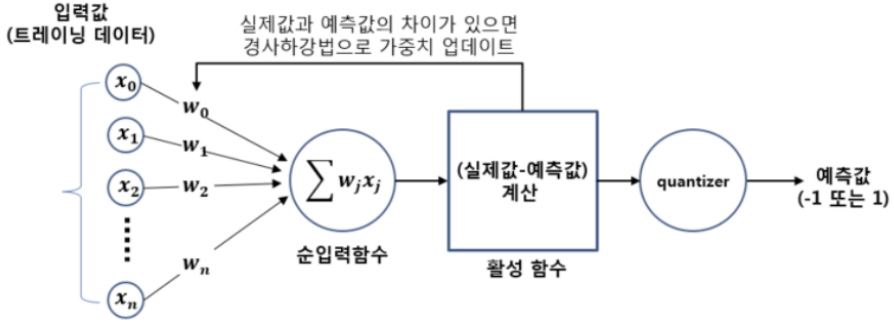
$$J(w) = \frac{1}{2}\sum_{i}\left(y^{(i)} - y^{\wedge(i)}\right)^2$$

https://brunch.co.kr/@hvnpoet/64

기존의 퍼셉트론



아달라인



1. 인공신경망(ANN) 및 퍼셉트론

1 파이썬 구현, 퍼셉트론

https://blog.naver.com/samsjang/220955881668

2 퍼셉트론을 이용한 iris 데이터 머신러닝

https://blog.naver.com/samsjang/220956787180

3 아달라인을 이용한 iris 데이터 머신러닝

https://blog.naver.com/samsjang/220959562205

4 아달라인과 학습률

https://blog.naver.com/samsjang/220960393669

5 표준화 적용후 아달라인

https://blog.naver.com/samsjang/220962195250

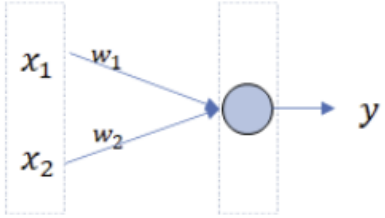
6 확률적 경사하강법을 적용한 아달라인

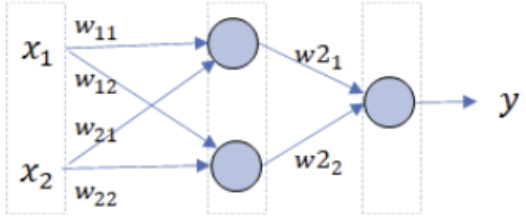
https://blog.naver.com/samsjang/220963580290

2. 다층퍼셉트론

다층 퍼셉트론의 등장

이와 같은 문제를 해결하기 위해 다층 퍼셉트론이 등장하였다. NAND와 OR연산의 결괏값을 다시 AND하여 XOR연산을 표현할 수 있다는 점에서, 입력층과 출력층 사이에 은닉층을 두어 다층 퍼셉트론을 만들면 좀 더 복잡한 문제를 해결할 수 있다는 것을 발견하였다. 현재 다양한 종류의 인공신경망은 모두 입력층, 은닉층, 출력층으로 나누어 구성된다.





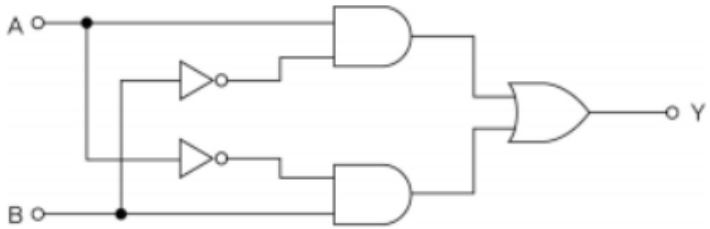
AND
NAND OR
XOR

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0

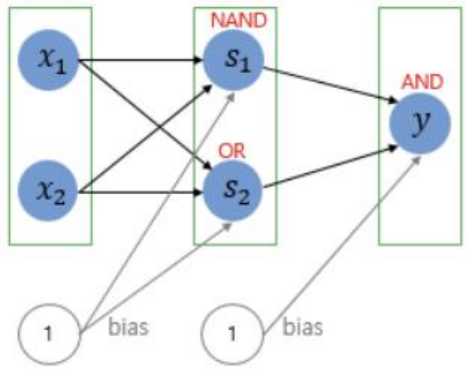
다층으로 해결

x_1	x_2	s_1 (NAND)	s_2 (OR)	y
0	0	1	0	0
1	0	1	1	1
0	1	1	1	1
1	1	0	1	0

- 파이썬을 활용한 데이터.AI분석(건강보험심사평가원) p115



XOR 게이트의 논리회로 구성(참고문헌: '디지털 논리회로 이해', 오창환 저, 한국학술정보(주))



https://leedakyeong.tistory.com/entry/%EB%B0%91%EB%B0%94%EB%8B%A5%EB%B6%80%ED%84%B0-%EC%8B%9C%EC%9E%91%ED%95%98%EB%8A%94-%EB%94%A5%EB%9F%AC%EB%8B%9D-%ED%8D%BC%EC%85%89%ED%8A%B8%EB%A1%A0%EC%9C%BC%EB%A1%9C-XOR-%EA%B2%8C%EC%9D%B4%ED%8A%B8-%EA%B5%AC%ED%98%84%ED%95%98%EA%B8%B0-in-python-%ED%8C%8C%EC%9D%B4%EC%8D%AC

2. 다층퍼셉트론

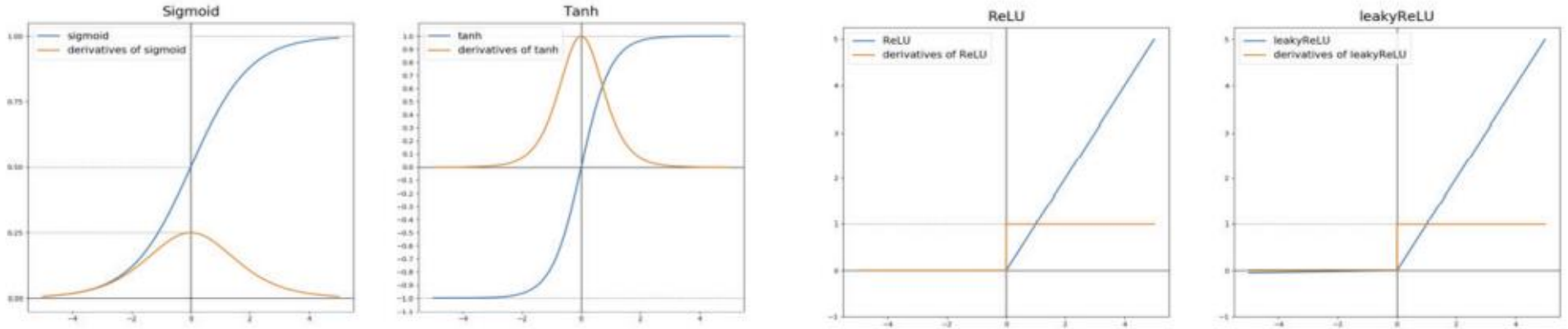
활성화 함수(Activation Function)

앞서 살펴본 것처럼 은닉층을 여러 층 두면 단순한 선형이 아니라 좀 더 복잡한 분류기를 만들 수 있는 가능성을 보았다. 하지만 단순히 은닉층(Hidden Layer)을 쌓는 것만으로는 부족하다. 왜냐하면 출력값이 다음 입력값으로 넘어갈 때 선형식(Linear)으로 전파된다면 은닉층을 여러 개 두는 방식이나 하나를 두는 것이나 차이가 없기 때문이다. 이러한 이유로 출력층의 결괏값을 비선형(Non Linear)으로 변환시켜 줄 장치인 '활성화 함수(Activation Function)'를 두게 되었다.

인공신경망에는 다양한 활성화 함수(Activation Function)가 존재한다. 대표적으로 Sigmoid, Tanh, ReLU 등이 있다. Sigmoid의 경우 양 끝에서 기울기가 거의 0에 수렴하고, 이로 인한 기울기 소실(Vanishing Gradient)문제가 생기기 때문에 대부분 인공신경망은 활성화 함수로 ReLU 함수를 사용한다.

ReLU함수는 0보다 작은 구간에서 기울기가 0이고 나머지 0보다 큰 구간에서는 항상 일정한 크기의 기울기를 가지는 특성을 갖고 있기 때문에 기울기 소실 문제를 막아 좋은 학습 결과를 기대할 수 있다.

활성화 함수는 인공신경망이 학습을 잘할 수 있도록 하는 하나의 수학적 장치이며 모델 설계 시 다양하게 실험하여 좋은 것을 선택하면 된다.



〈활성화 함수와 도함수의 그래프〉

- 파이썬을 활용한 데이터.AI분석(건강보험심사평가원) p116

3. 활성화 함수 & 손실함수 & 오차역전파

- **선형함수와 비선형함수 개념이해 및 활성화 함수**

  https://wikidocs.net/24987

- **손실함수**

- **오차역전파**

  https://ksm2853305.tistory.com/46

- **미분, 역전파, 및 실습 (chain rule:연쇄법칙에 간단히 개념잡고 아래 내용을 살펴봅니다.)**

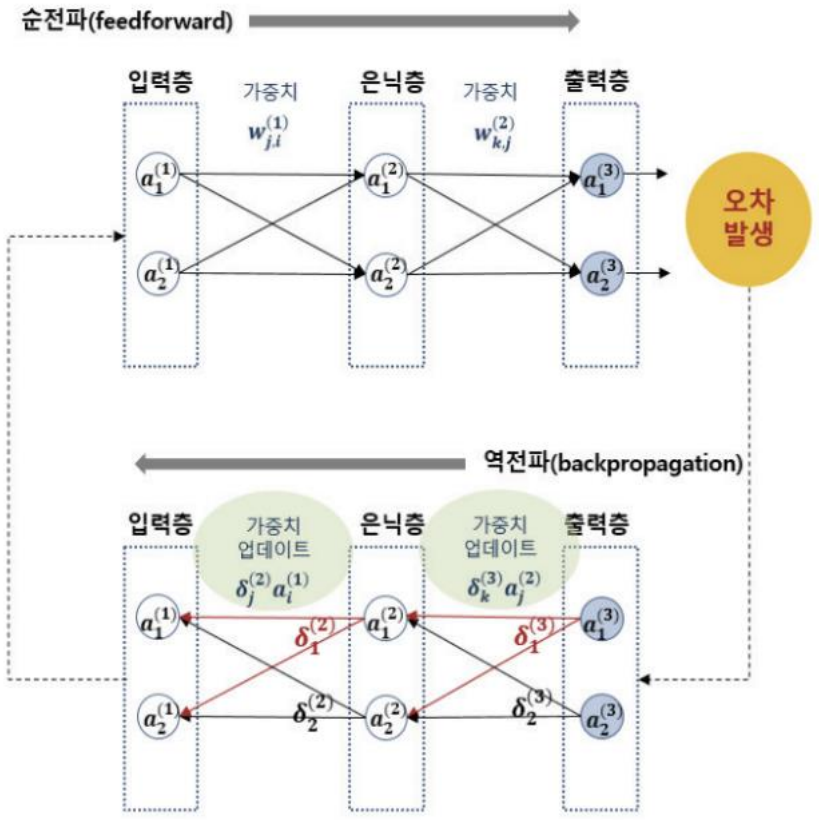
미분	https://www.youtube.com/watch?v=vS51prw_yfw
경사하강 4분35부터 봐도됨	https://www.youtube.com/watch?v=GEdLNvPIbiM
오차역전파	https://www.youtube.com/watch?v=1Q_etC_GHHk
OLS(최소제곱법,미분)	https://www.youtube.com/watch?v=-oBmMED_5rI
코드실습	https://nbviewer.org/github/rickiepark/do-it-dl/blob/master/Ch03.ipynb

3. 활성화 함수 & 손실함수 & 오차역전파

순전파(Feedforward), 역전파(Backpropagation), 최적화(Optimization)

앞에서 인공신경망이 수많은 가중치를 변경하여 데이터를 잘 표현하는 가중치를 찾아 문제를 해결하며, 가중치를 변경할 때 참고하는 도구로써 손실 함수(Loss Function)를 사용한다는 것을 살펴보았다.
그렇다면 우리의 고민은 이렇게 정리할 수 있다. '어떻게 하면 손실 함수(Loss Function)의 값을 최소로 만들 수 있을까?(예측값과 정답값을 비슷하게 하는 가중치의 조합을 찾기)' 인공신경망은 최적화(Optimization)와 역전파(Backpropagation)의 방식을 이용한다. 손실 함수가 각각의 가중치를 변수로 하는 함수라는 점을 이용하여, 손실 함수의 값을 각각 가중치들로 국소적 미분하고 지역적 최솟값(Local Minimum)으로 하는 가중치들을 찾는다. 이때 가중치는 연쇄 법칙(Chain Rule)을 사용하여 역전파(Backpropagation)법으로 최솟점을 찾는 기울기 하강(Gradient Descent)을 하며 변경(Update)된다.

인공신경망은 모델이 예측 결괏값을 찾는 과정이 입력층-은닉층-출력층으로 순차적으로 흐른다고 하여 순전파(Feedforward)라 하며 가중치의 수정은 출력층-은닉층-입력층으로 흐르기 때문에 역전파(Backpropagation)라고 한다. 또한 최적의 가중치를 찾아가는 방식, 그러한 알고리즘을 최적화(Optimization)라고 한다.



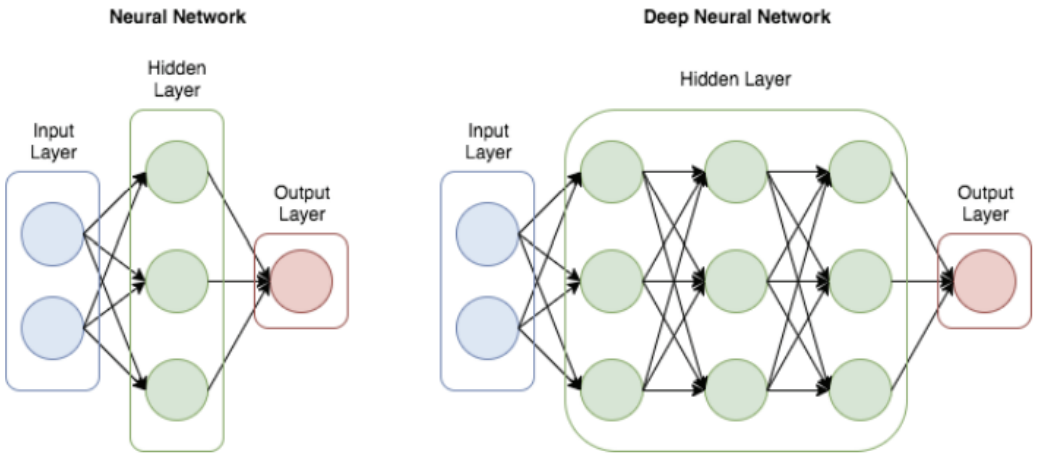
〈순전파(Feedforward)와 역전파(Backpropagation)〉

- 파이썬을 활용한 데이터.AI분석(건강보험심사평가원) p118

4. 심층신경망

핵심요약

심층신경망(Deep Neural Network)은 인공신경망(Artificial Neural Network)과 동일한 구조와 동작 방식을 갖고 있다. 심층신경망은 단지 인공신경망에서 은닉층(Hidden Layer)의 깊이가 깊어진 형태를 말한다.



신경망의 발전계기

인공신경망의 기본 아이디어가 나온 것은 1950년대이고 이후에 발전을 거듭하지만 불과 2000년대 중반까지만 해도 사람들로 하여금 그다지 알려지고 활용되던 기술이 아니었다. 여기에는 몇 가지 이유가 있는데 하나는 당시에 GPU와 같은 수많은 가중치를 빠르게 계산할 수 있는 자원(Resource)이 충분치 않았고, 사람이 원하는 수준으로 성능을 올리기 위해서 필요했던 충분한 양의 데이터(Data)를 모으기에도 인프라가 부족했다. 또한 모델의 성능향상과 효율성의 측면에서 획기적인 도움을 주었던 역전파(Backpropagation) 및 최적화(Optimization)와 같은 효율적인 알고리즘(Algorithm)이 없었기 때문이었다.

딥러닝(Deep Learning)

반도체 기술의 발전으로 GPU 자원이 상용화되고, 정보화 사회를 거치며 충분한 데이터를 모을 수 있는 인프라가 마련되었으며 학자들이 수학적 기법을 이용하여 다양한 학습 알고리즘을 개발하면서 이러한 인공신경망의 문제점을 극복할 수 있었다. 그리고 마침내 오랫동안 부정적이었던 이미지를 가진 인공신경망(Artificial Neural Network)이라는 용어를 대체할 '깊이 학습한다'는 의미를 지닌 딥러닝(Deep Learning)이라는 용어를 만들게 되었다.

- 파이썬을 활용한 데이터.AI분석(건강보험심사평가원) p121