인공지능

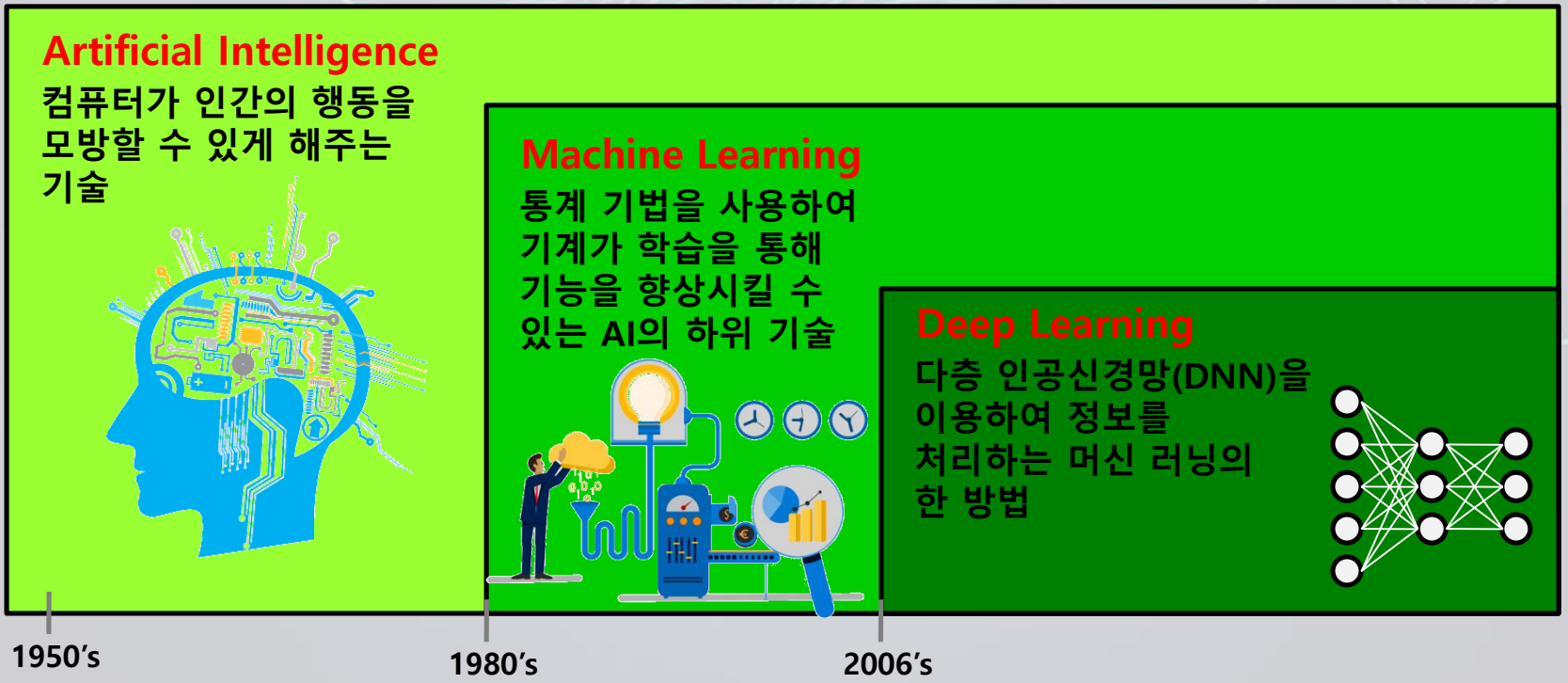
딥러닝 기초_신경망 복습

강성관 (silicon1@hanmail.net)

신경망 복습

인공지능 vs 머신러닝 vs 딥러닝

- 기계가 학습할 수 있도록 하는 분야
- 인공지능 연구의 한 분야로 최근들어 딥러닝을 통해서 빠르게 발전

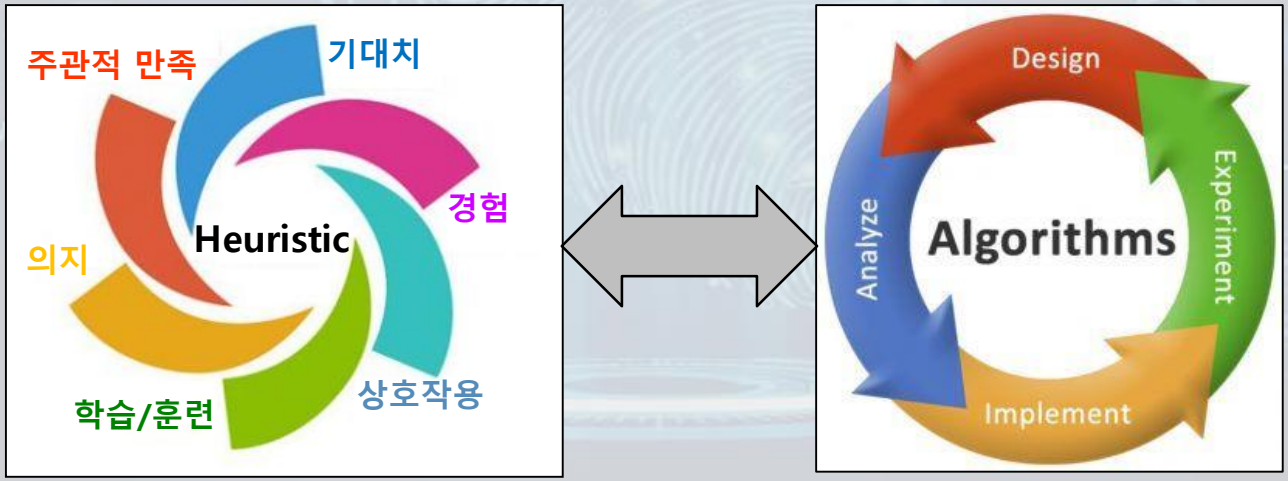


머신러닝 vs 딥러닝

구분	Machine Learning	Deep Learning
훈련 데이터 크기	작음	큼
시스템 성능	저 사양	고 사양
feature 선택	전문가 (사람)	알고리즘
feature 수	많음	적음
문제 해결 접근법	문제를 분리→각각 답을 얻음 → 결과 통합	end-to-end (결과를 바로 얻음)
실행 시간	짧음	김
해석력	해석 가능	해석 어려움

인공지능의 분야

- 인공지능이 필요한 분야
  - 최적해를 가지고 있지 않을 때
  - 휴리스틱 알고리즘을 가지고 있을 때
  - 불확실하고 불완전한 데이터를 가지고 있을 때
  - 비정형적이고 탐색형 추론을 필요로 할 때

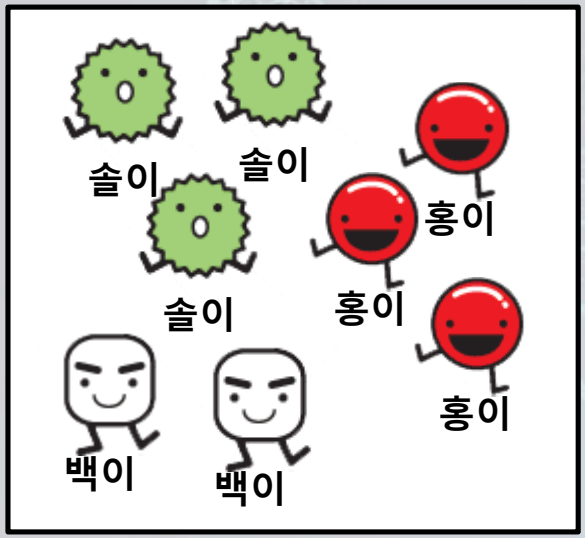


종류 – 지도의 유무

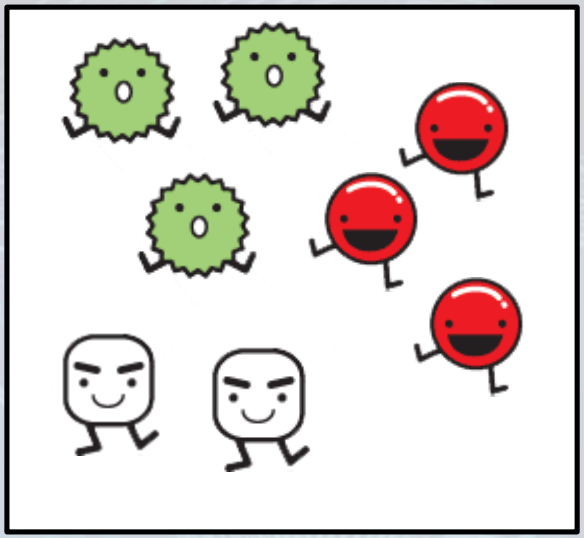
- **지도학습 (Supervised Learning)** : **속성(레이블)**과 **특정 데이터세트**(트레이닝세트, 명시적 정답이 정해진 데이터 세트)이 주어진 상태에서 컴퓨터를 학습시키는 방법 (감독학습, 교사학습)
  (예) **회귀 (Regression), 분류 (Classification)**
- **비지도 학습 (Unsupervised Learning)** : **지정된 레이블이 없는 데이터세트**(정답이 정해지지 않는 데이터 형태)에서 암시적 관계나 특징 (패턴)을 발견하는 것이 유용한 경우에 사용 (비감독학습, 비교사학습)
  (예) **군집(Clustering), 차원축소 (Dimensionality Reduction)**
- **준지도학습 (Semi-Supervised Learning)** : 학습 데이터가 **약간의 (속성)레이블**을 가지고 있음 (반지도학습) → 다수의 레이블이 없는 데이터를 약간의 레이블이 있는 데이터로 보충해서 학습
- **강화학습 (Reinforcement Learning)** : 문제의 답이 바로 주어지지 않고 **Agent**가 주어진 **환경** (Environment)에서의 **경험/학습**에 따라 **최대의 보상(reward)**을 얻는 정책(policy)을 찾는 **학습**
  → 아이는 걷기 위해 어떻게 행동할 줄 모르지만 환경과 상호작용하면서 걷는 법을 알아감
  (예) **Q-Learning, DQN (Deep-Q-Network) 등**

종류

지도학습

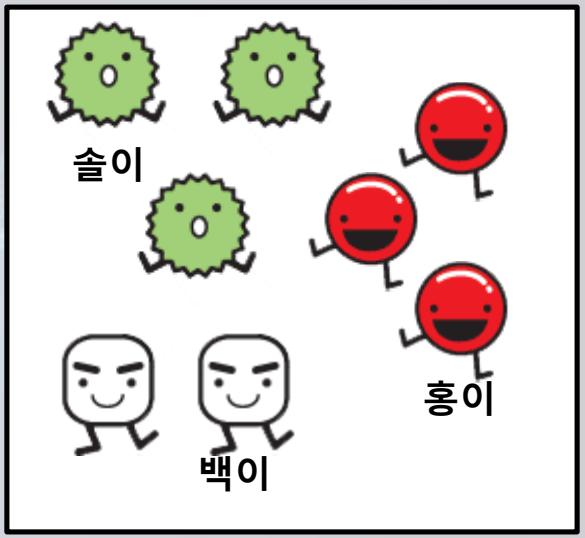


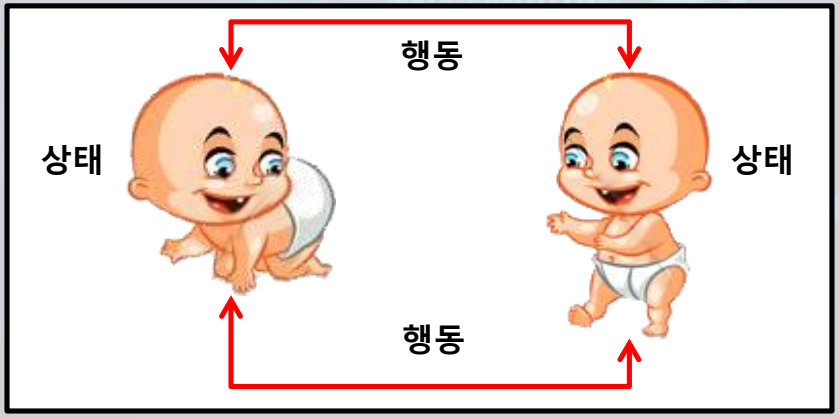
비지도학습



준지도학습

레이블이 있는 데이터를 이용하여 레이블이 없는 데이터를 추론

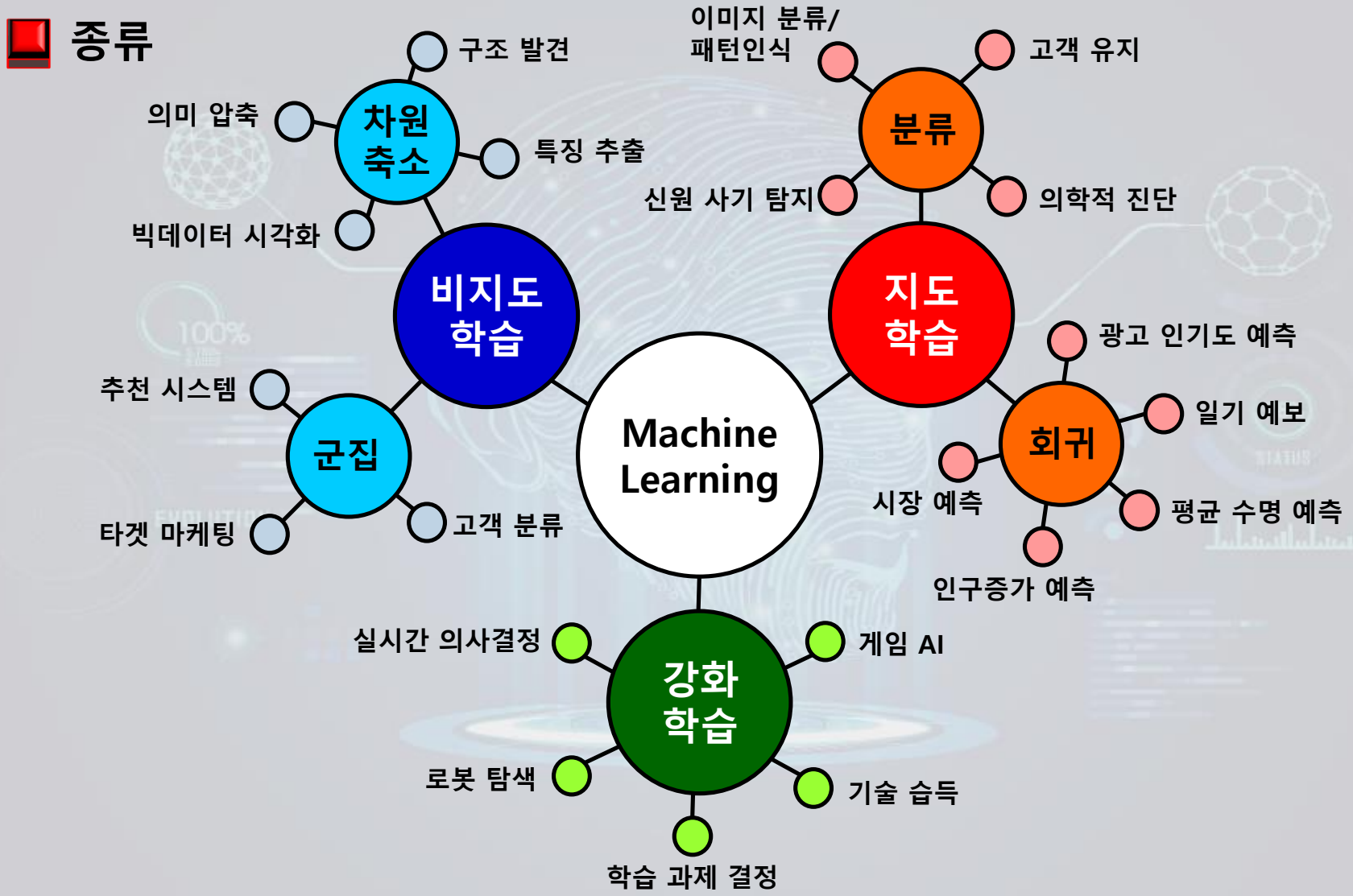




강화 학습

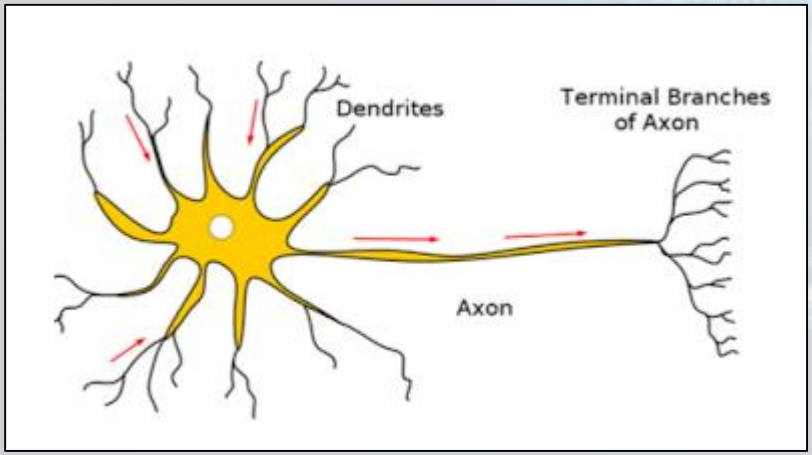
지속적인 행동을 통해 경험을 얻고 경험을 통한 학습으로 걷는 방법을 습득

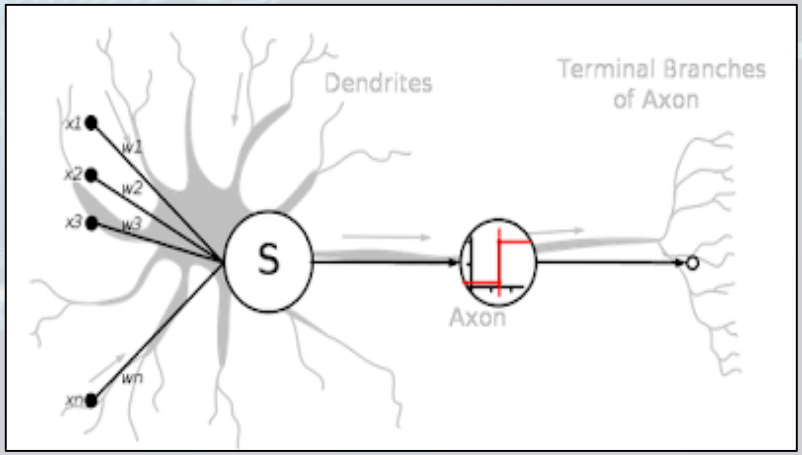
종류


Machine Learning
비지도 학습
차원 축소
구조 발견
의미 압축
특징 추출
빅데이터 시각화
군집
추천 시스템
타겟 마케팅
고객 분류
지도 학습
분류
이미지 분류/ 패턴인식
고객 유지
신원 사기 탐지
의학적 진단
회귀
광고 인기도 예측
일기 예보
시장 예측
평균 수명 예측
인구증가 예측
강화 학습
실시간 의사결정
게임 AI
로봇 탐색
기술 습득
학습 과제 결정

분류 - 퍼셉트론 (Perceptron)

- 1957년 프랑크 로젠블라트에 의해 고안
- 대뇌피질의 신경세포는 **수상돌기(dendrite)**를 통해 다른 신경세포로부터 입력 신호를 받음 → **세포체(cell body)**에서 정보 연산을 처리 → **축색(axon)**을 통해 처리된 정보를 다른 신경세포로 전달
- 이러한 신경세포는 **시냅스**라는 가중 연결자(weighted connectior)로 여러 층의 신경망을 구성
- 세포체에서의 연산 : 입력 신호들의 합을 구함 → 그것이 일정한 임계치가 되면 축색이 활성화되어 스파이크를 일으킴 → 다른 신경세포로 전기 신호를 전달
- 이러한 특성을 모방하여 만든 것이 **인공 신경망(ANN : Artificial Neural Network)**





분류 - 로지스틱 회귀 (Logistic Regression)

- 로지스틱 회귀와 퍼셉트론의 차이점

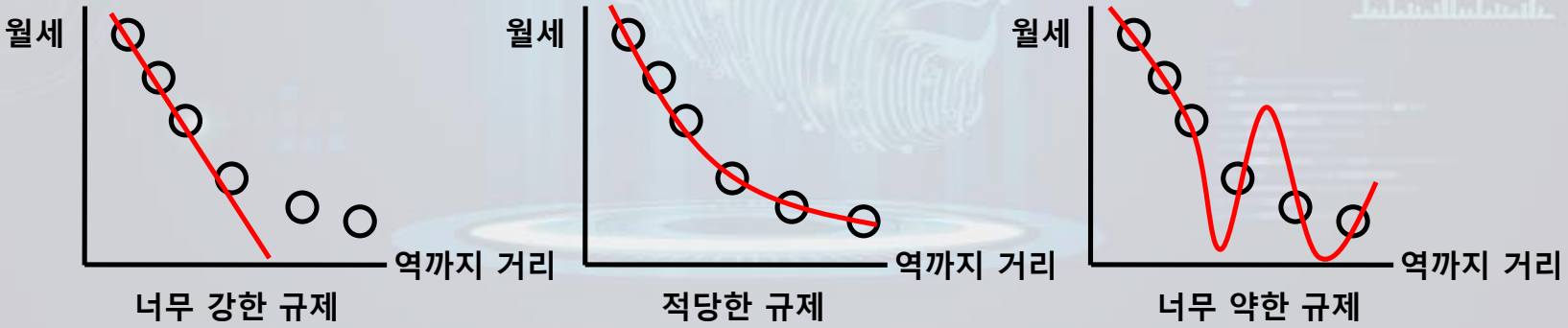
(1) **시그모이드 함수**의 사용

(2) 손실함수가 **교차 엔트로피 오차 함수** (Cross entropy error function)인 점

$$E = -\sum_{n=1}^{N} t_n \log y_n + (1-t_n)\log(1-y_n)$$

(N : 데이터 수, t (정답 1, 오답 0), y : 입력 데이터에 대한 출력)

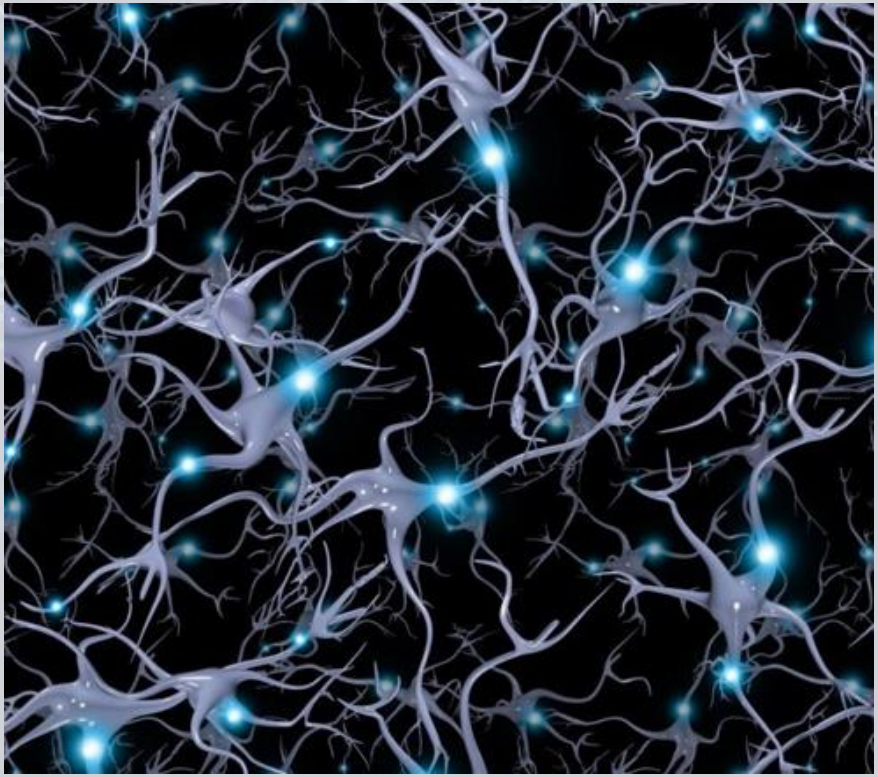
(3) **규제항 (Regularization term)**이 추가되어 과적합을 방지할 수 있다는 점 (손실함수 + 규제항)



(4) **온라인 학습과 배치학습**에 모두 적용할 수 있다는 점

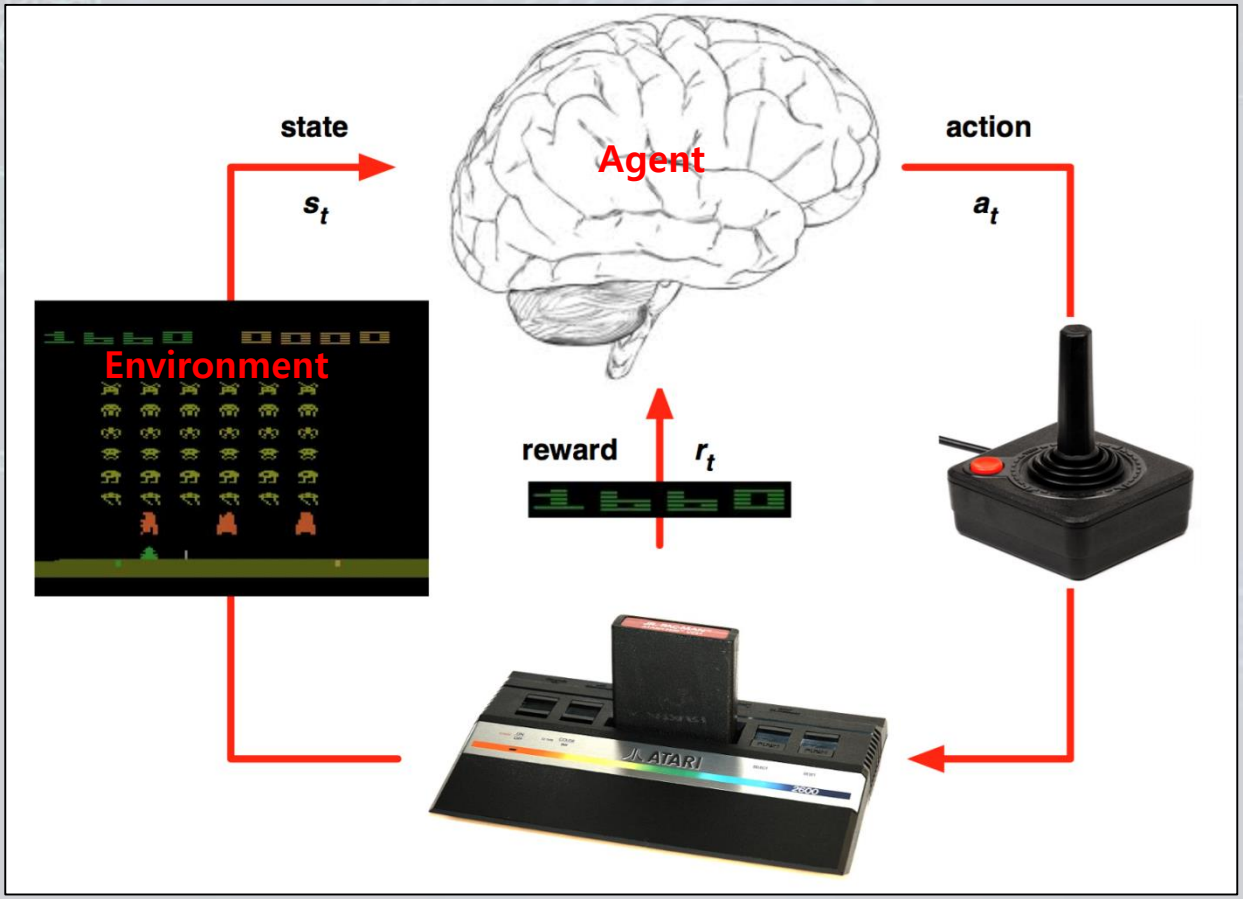
분류 - 신경망 (Neural Network)

- 신경망은 뇌의 신경세포 (뉴런)들이 시냅스로 연결되어 전기 신호를 통해 정보를 주고 받는 모습에서 착안한 것으로 **다층 퍼셉트론** (Multi-layer Perceptron)으로 부름



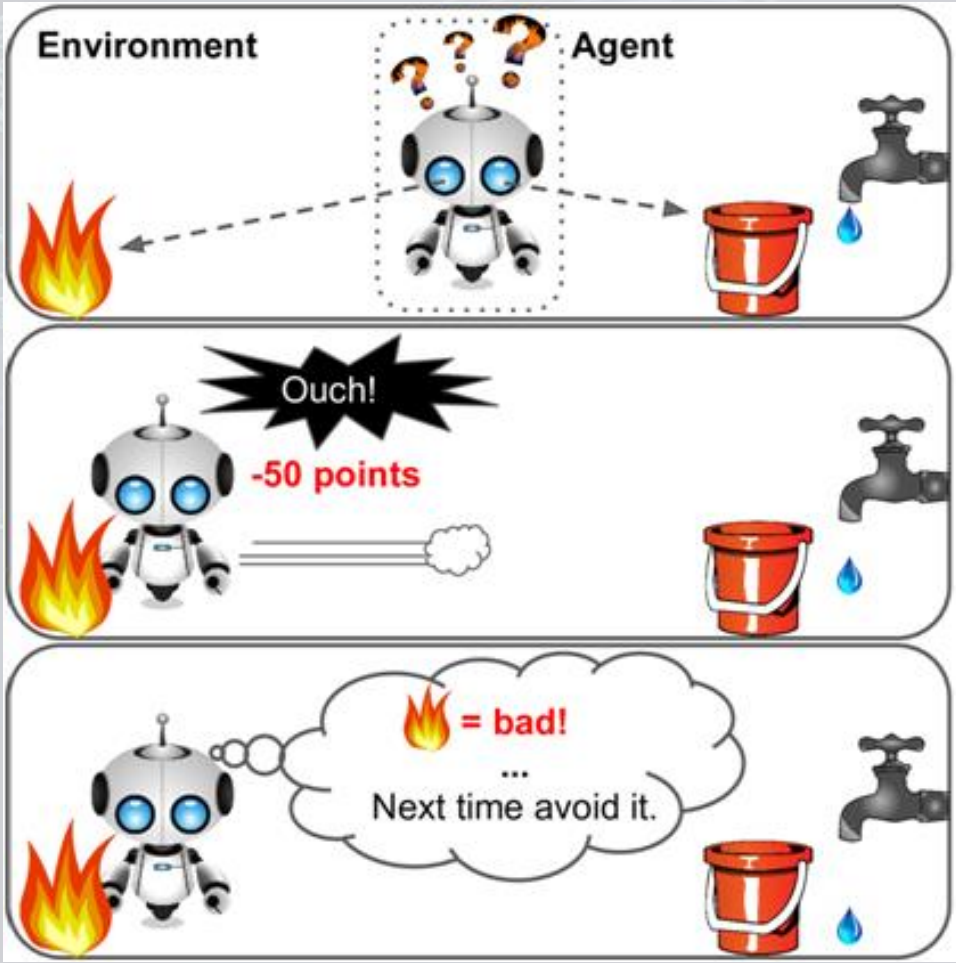
강화학습 (Reinforcement Learning)

- 개념



강화학습 (Reinforcement Learning)

- 개념



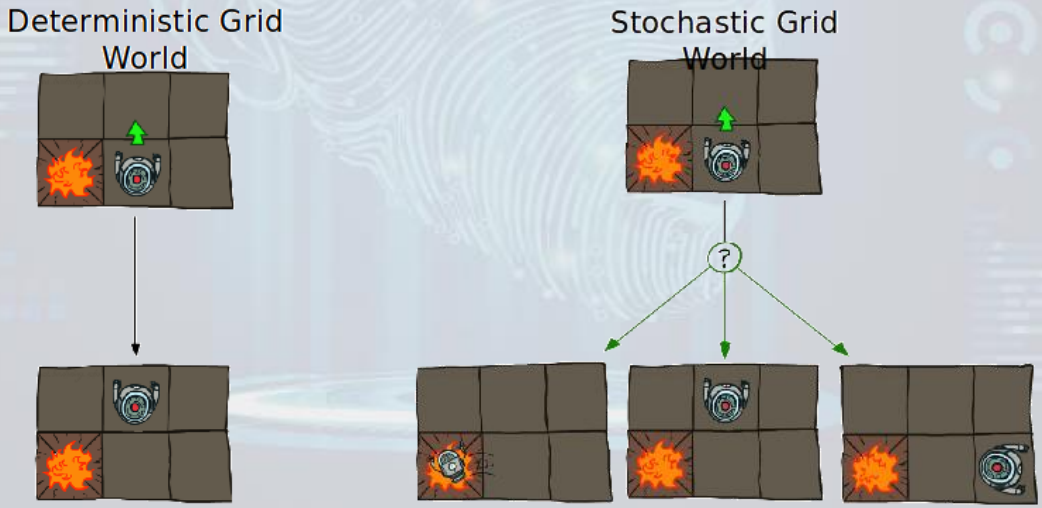
1. 관찰
2. 정책에 따라 action을 선택
3. action
4. 보상 또는 형벌
5. 정책 업데이트 (학습 단계)
6. 최적의 정책이 발견될 때까지 반복

Markov Decision Process (MDP)

- state, action, state transition probability matrix, reward, discount factor로 구성
- 로봇이 있는 위치가 **state**, 앞뒤좌우로 이동하는 것이 **action**, 저 멀리 보이는 빛나는 보석이 **reward**
  → 문제의 정의 → 이제 이 로봇은 보석을 얻기 위해 어떻게 해야 할지를 학습

Markov Decision Process (MDP)

- **State** : agent가 인식하는 자신의 상태
- **Action** : environment에서 특정 state에 갔을 때 해야 할 것을 지시하는 것 → agent는 action을 취함으로써 자신의 state를 변화 시킬 수 있음 (**Controller**)
- **State transition probability matrix** : 어떠한 action을 취할 경우 state가 결정적으로 정해진 것이 아니고 확률적으로 정해지게 되는 것



Markov Decision Process (MDP)

- **Reward** : agent가 action을 취하면 그에 따른 보상을 environment가 agent에게 알려줌
- **Discount Factor** : 시간에 따른 reward의 가치가 달라지는 것 (조삼모사) → [0, 1] 값을 가짐

Monte-Carlo Learning

- 현재의 policy를 바탕으로 움직여보면서 sampling을 통해 value function을 update하는 것을 **model-free prediction**이라 하고 policy를 update까지 하게 된다면 **model-free control**이라고 함
- Sampling을 통해서 학습하는 model-free 방법
  (1) Monte-Carlo : episode마다 update하는 방법
  (2) Temporal Difference : time step마다 update하는 방법

- **episode를 끝까지 가본 후에 받은 reward들로 각 state의 value function들을 거꾸로 계산해보는 것**
- MC(Monte-Carlo)는 끝나지 않는 episode에서는 사용할 수 없는 방법

시간차 학습 (Temporal Difference Learning)

- **time step마다 update하는 방법**

- 대학생활을 예로 들면

  - **Monte-Carlo (MC)** : 대학교에 들어와서 졸업을 한 다음에 그 동안을 돌아보며 "이건 더 했어야했고 술은 덜 마셔야했다"라고 생각하며 다시 대학교를 들어가서 대학생활을 하면서 졸업할 때까지 똑같이 살다가 다시 졸업하고 자신을 돌아봄

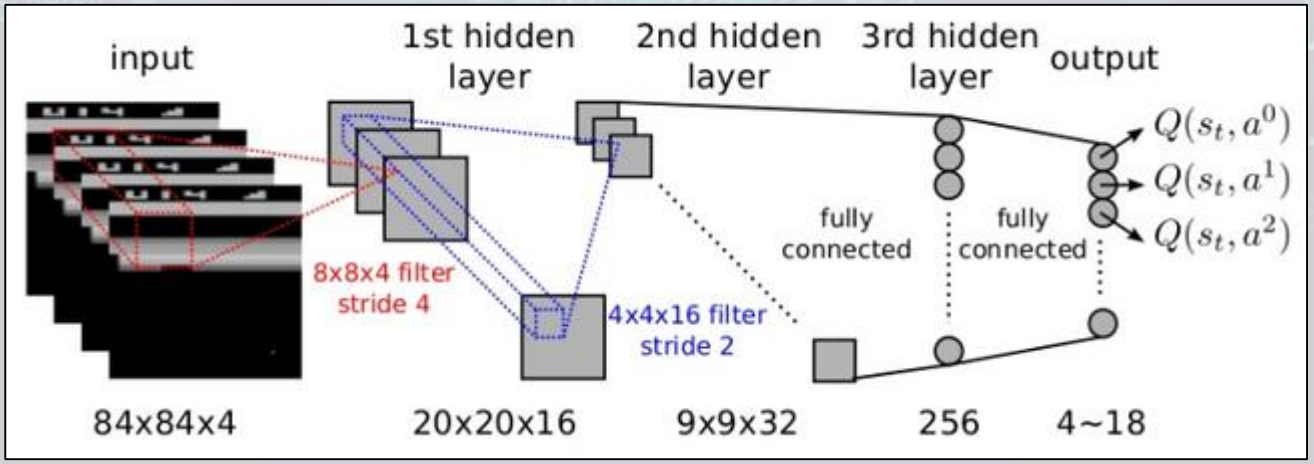
  - **Temporal Difference (TD)** : 같은 대학교를 다니고 있는 2학년 선배가 1학년선배를 이끌어주는 것

Q-Learning

- 주어진 유한 마르코프 결정 과정의 최적의 정책을 찾기 위해 사용
- 현재 state S에서 action을 선택하는 것은 behaviour policy를 따라서 선택
- TD에서 update할 때는 one-step을 bootstrap하는데 이 때 다음 state의 action을 선택하는 데는 behaviour policy와는 다른 policy(alternative policy)를 사용하면 Importance Sampling이 필요하지 않음
- 이전의 Off-Policy에서는 Value function을 사용했었는데 여기서는 action-value function을 사용함으로서 다음 action까지 선택을 해야하는데 그 때 다른 policy를 사용한다는 것

DQN (Deep Q-Network)

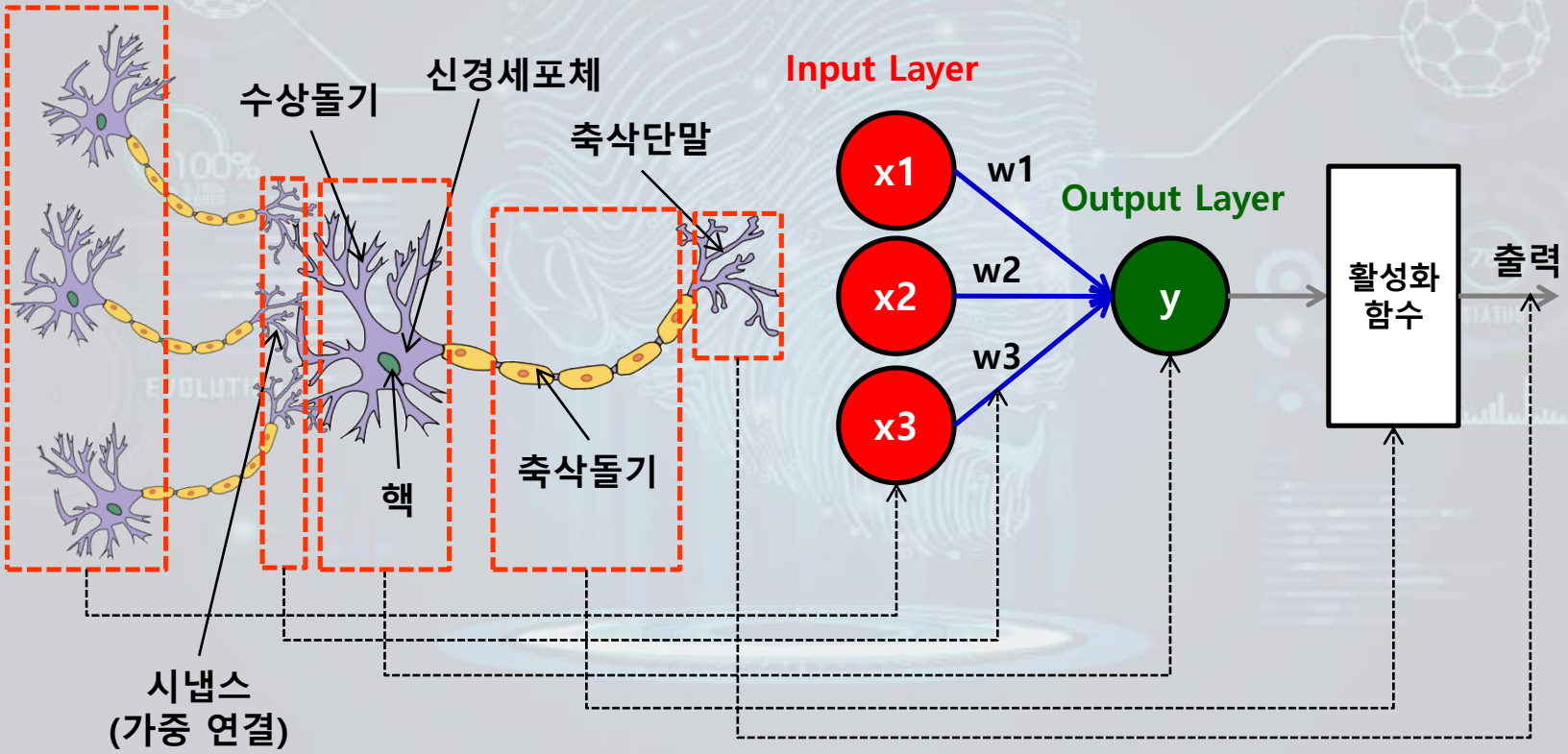
- 강화학습은 현재 상태 S에서 행동 a를 결정하고 그에 따른 보상을 받아 행동을 수정
- Q테이블은 각 상태집합에서 행동에 따른 우선순위가 있는 테이블
- Q(S, a)는 상태 S에서 a라는 행동을 했을때의 이득값으로 현재 상태의 모든 행동 a에 대해서 Q값을 구해 가장 높은 우선순위의 행동을 수행
- 보통 Q테이블을 배열로 설정하는데 이를 딥러닝 CNN으로 변경한 것이 DQN이므로 그렇기 때문에 상태나 행동이 큰 집합도 학습이 잘됨 → 이세돌과의 바둑 경기로 유명한 알파고가 바로 이 방법을 사용



DQN (Deep Q Network)

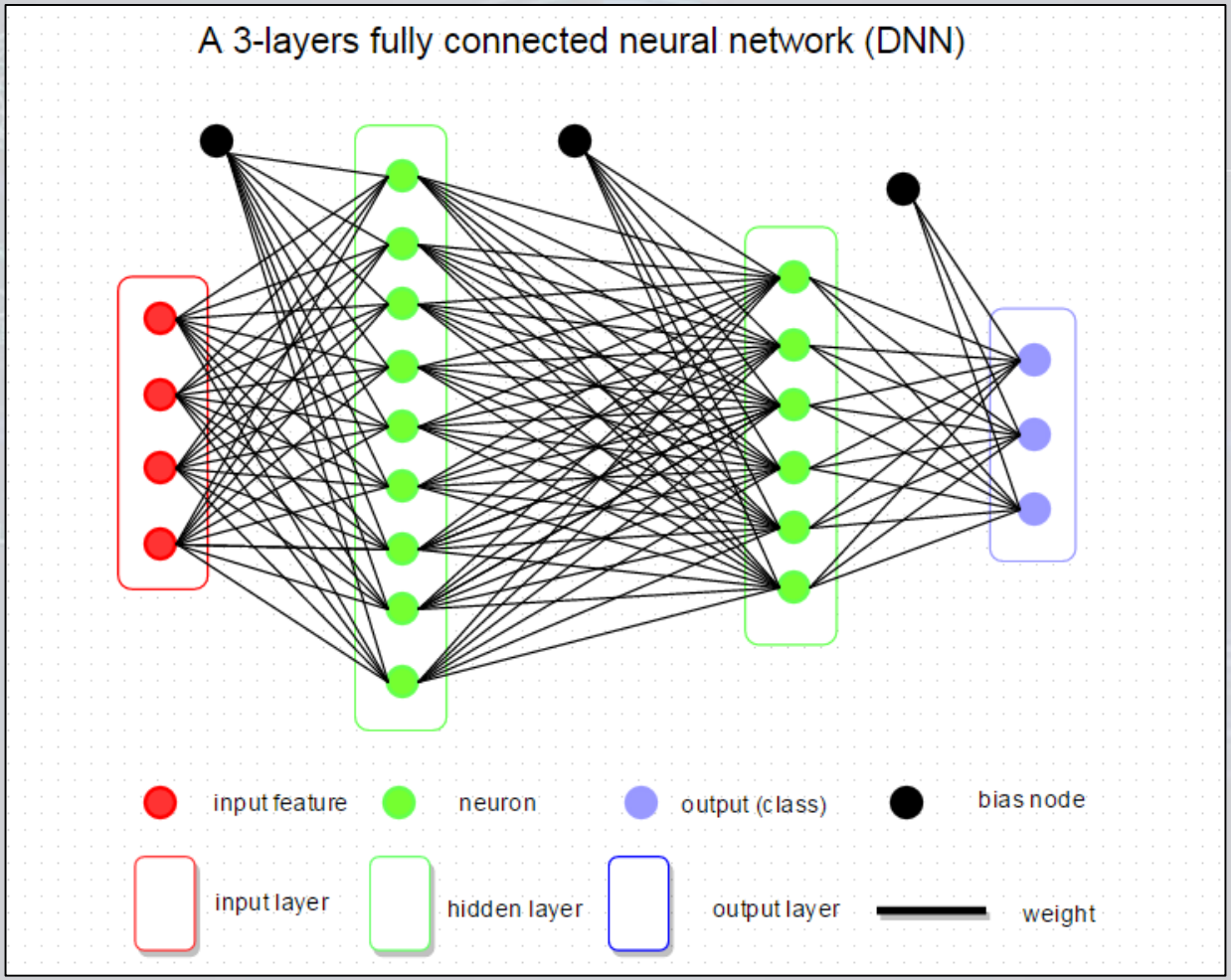
- 구글 딥마인드가 개발한 알고리즘으로 강화 학습 가능한 심층 신경망을 이용하여 초창기 아타리 2600 게임들을 인간 수준으로 플레이할 수 있는 인공지능
- 어떤 상태에서 어떤 행동을 취하는 것이 **가장 큰 보상을 받을 수 있는지를 학습**하는데 이를 위해 Q 함수라는 것을 사용하여 상태와 행동을 입력으로 주면 기대값을 출력해줌
- **게임을 할 때 Q라는 사람을 옆에 두고 매번 Q의 의견을 물어서 플레이에 참고하는 것**
  - 학습을 처음 시작할 때는 Q의 실력은 인간 이하지만 학습을 통해 Q의 의견대로 게임을 한 뒤 졌으면 혼내고 이기면 칭찬해서 Q의 판단력을 프로게이머 수준까지 키우는 것
- 문제는 현재까지 방법으로는 Q를 제대로 학습시키는 것이 매우 어려웠다는 것 → 깊은 신경망을 통해 Q함수를 구성 → DQN이란 이름은 Q 함수를 'Deep Neural Network로 구성했기 때문
- Deep CNN (Convolutional Neural Network)를 도입해서 network를 훈련시키는 것
- CNN은 최근의 딥러닝 열풍을 몰고 온 장본인으로서 이미지를 학습시키는 데 최적화된 Neural Network모델

신경망


수상돌기
신경세포체
축삭단말
핵
축삭돌기
시냅스
(가중 연결)
Input Layer
x1
x2
x3
w1
w2
w3
Output Layer
y
활성화
함수
출력

다층 퍼셉트론 (MLP : Multi Layer Perceptron)

DNN (Deep Neural Network)



인공신경망의 문제

덜하거나

Underfitting 학습이 잘 안되요 !!

느리거나

Slow 오늘 안에는 끝나겠죠 ?

과하거나

Overfitting 융통성이 없어요!!

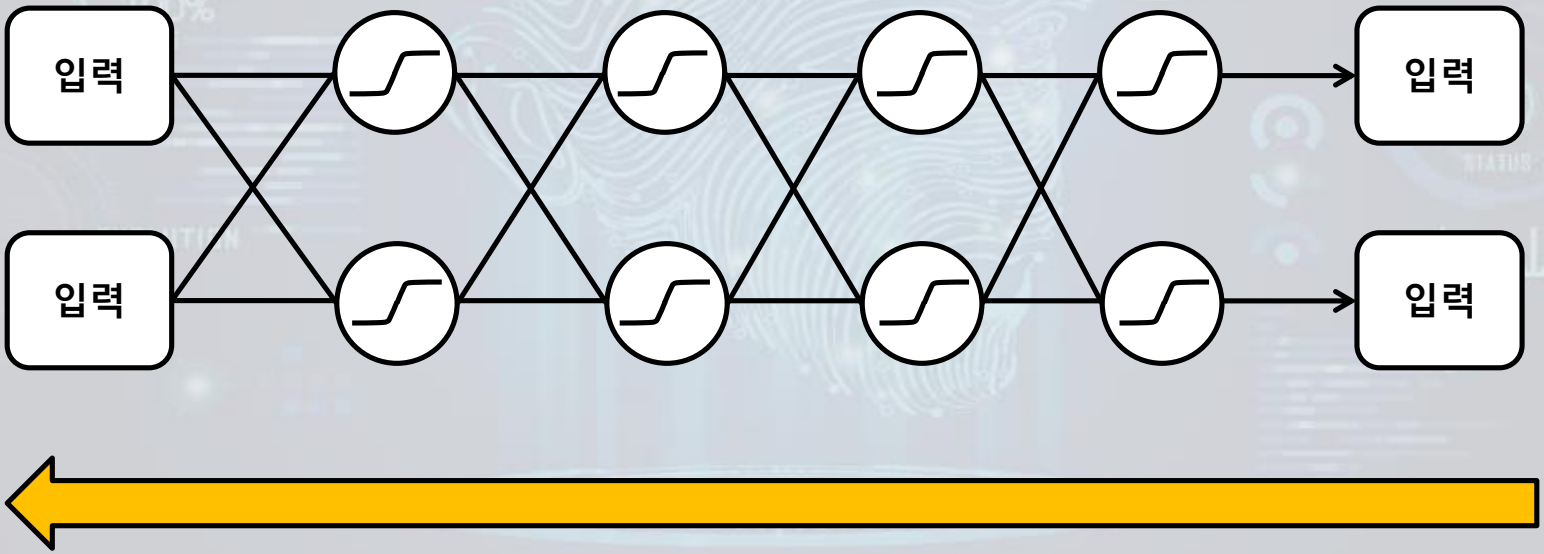
사라지거나

Gradient Vanishing 일을 하다가 말아요 ㅠㅠ

역전파 (Backpropagation)

뭐가 전달되는가 ?

현재 내가 틀린 정도를 미분(기울기) 한 값을 전달



미분하고, 곱하고, 더하고를 역방향으로 반복하며 업데이트

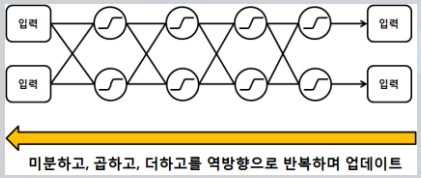
역전파 (Backpropagation)

그런데 문제는 ?

활성화 함수로 sigmoid 함수를 썼다는 것

여기의 미분(기울기)는 있으니 그래도 다행

여기는 기울기가 0이라 뒤로 전달할 게 없는데 ...

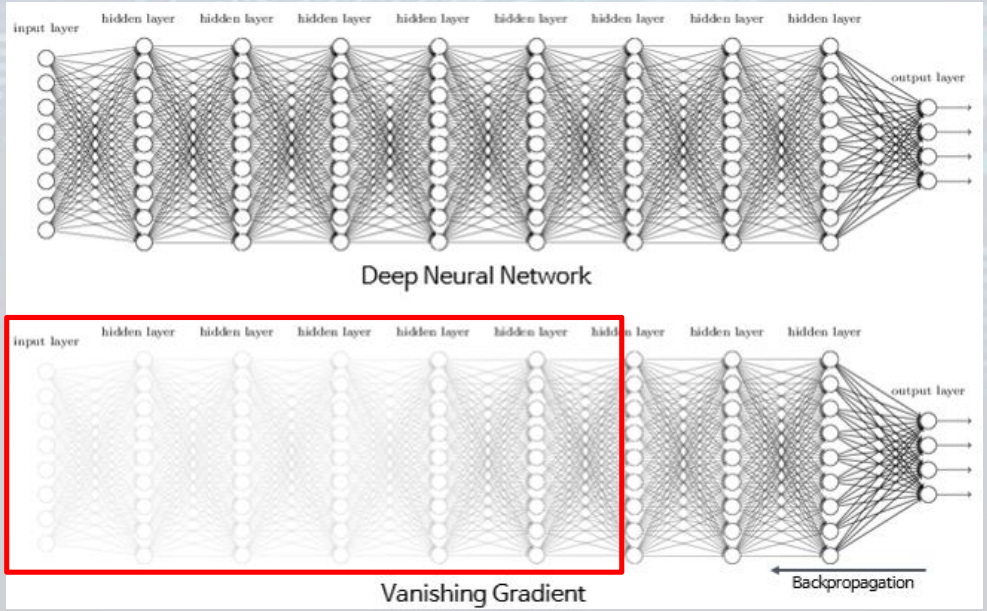


이런 상황에서 역전파를 수행하면 ?

역전파 (Backpropagation)

- Vanishing Gradient : 층이 깊어질 수록 업데이트가 사라지는 현상

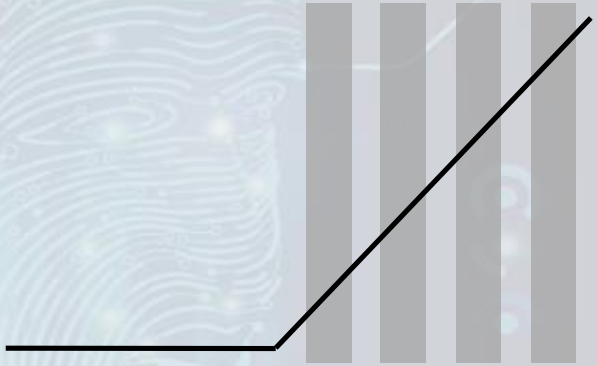
학습이 잘 안됨



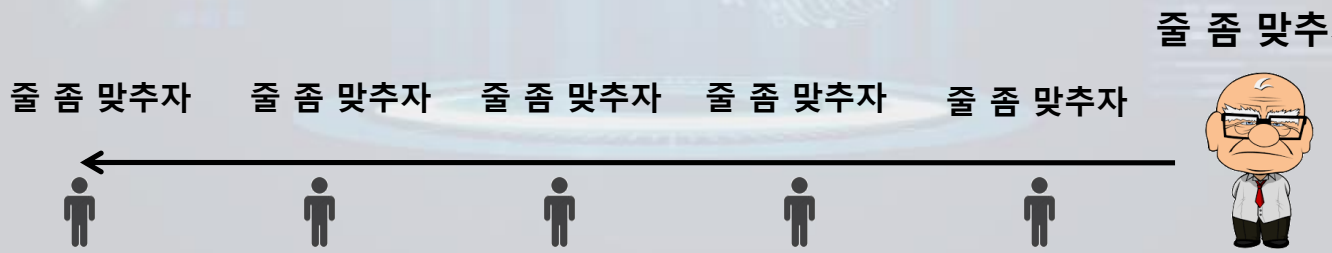
역전파 (Backpropagation)

사라지는 Sigmoid 대신에

ReLU (Rectified Linear Units)를 써보자

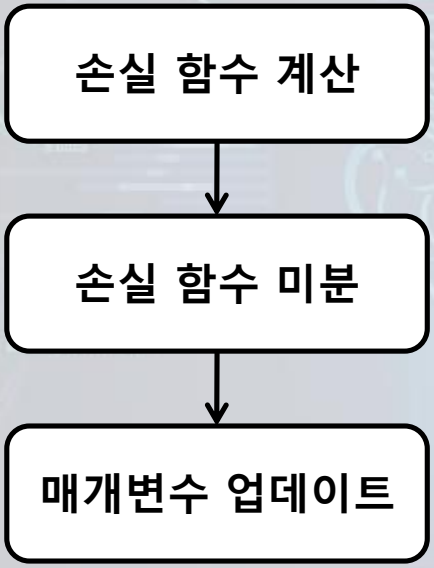


양의 구간에서 전부 미분값이 1이다



경사하강법 (Gradient Decent)

- 신경망의 가중치 매개변수들을 최적화 하는 방법으로 손실함수의 현 가중치의 기울기를 구해서 손실을 줄이는 방향으로 업데이트



현재 가중치에 가지고 있는 모든 데이터를 이용하여 실제 값과 예측 값의 차이를 계산 (RMSE, CEE 등)

에러를 줄이는 방향 (내 기울기 * 반대방향) 계산

해당 방향으로 학습율을 곱해서 매개변수를 업데이트 (반복)

가중치 업데이트 = 에러를 낮추는 방향 (decent) X 한 걸음 크기 (learning rate) X 현 지점의 기울기 (gradient)

경사하강법 (Gradient Decent)

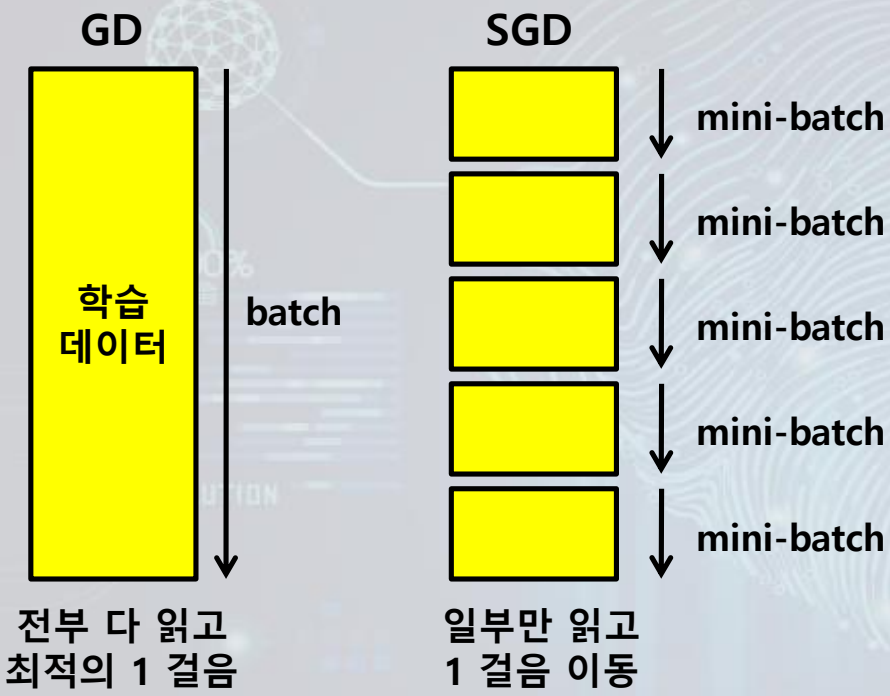
현재 가중치에 가지고 있는 모든 데이터를 이용하여
실제 값과 예측 값의 차이를 계산 (RMSE, CEE 등)

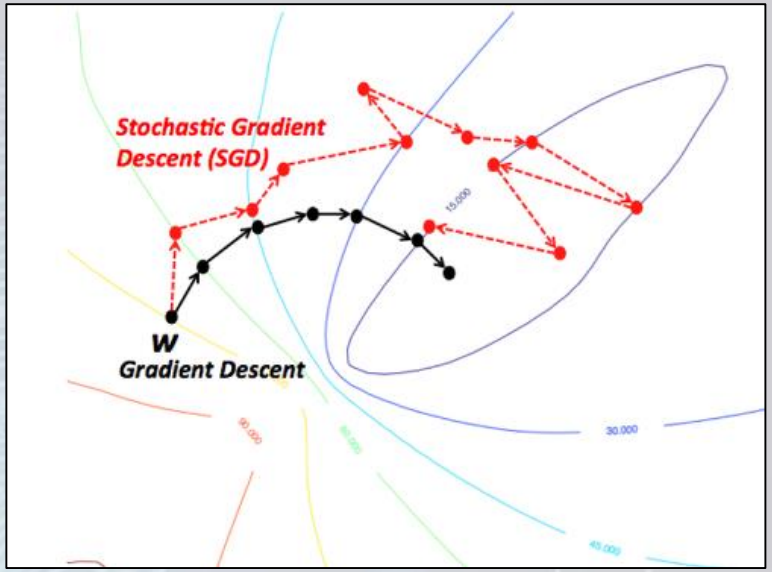
만약 가지고 있는 데이터가 몇 억 건이라면
한 걸음 걸을 때마다 몇 억 건을 다 넣으면 속도는 ?

↓

SGD (Stochastic Gradient Decent)
모든 것을 다 보면서 느리게 가기보다는
조금만 보면서 빨리 가자

경사하강법 (Gradient Decent)





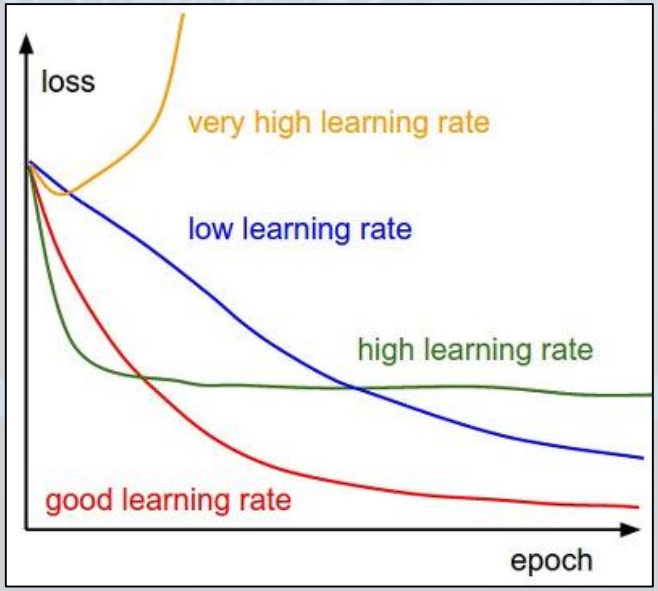
GD : 모든 걸 계산(1시간) 후 최적의 1 걸음 (6걸음 x 1시간 = 6시간)
최적인데 너무 느리다 !!

SGD : 일부만 계산(5분), 틀려도 빠르게 1걸음 (11걸음 x 5분 = 55분)
조금 헤매도 인근까지 빠르게 갔다 !!

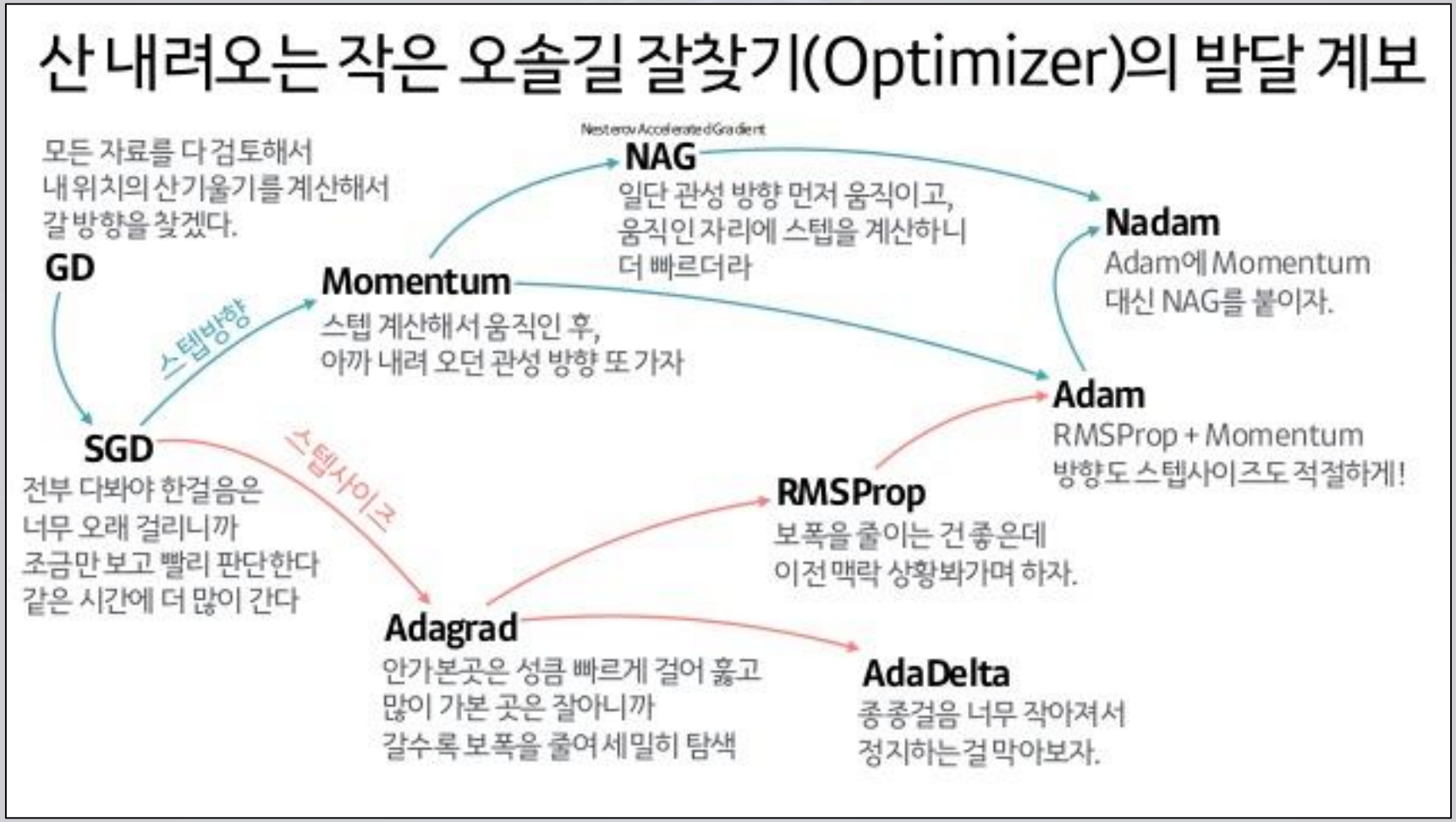
경사하강법 (Gradient Decent)

그런데 너무 헤매면서 간 것 같은데 …
덜 헤매는 방법은 없을까 ?

걸음의 크기 (learning rate)가 문제
산을 잘 내려오는 것은 방향도 중요하지만
얼마의 보폭으로 내려 올지도 중요



경사하강법 (Gradient Decent)



Overfitting

열심히 신경망에게 고양이를 가르쳤는데 ...

애가 고양이야 알았지 !!

어 검정색이네요
고양이 아님

돼지인가요 ?
고양이 아님

귀는 어디갔나요 ?
고양이 아님

Overfitting

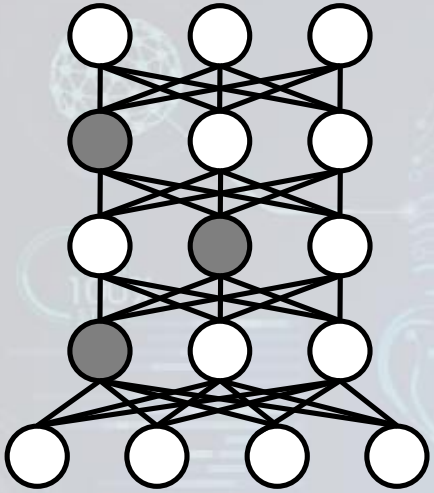
신경망에게 융통성을 가르쳐보자.
가르칠 때부터 조금씩만 가르쳐보자

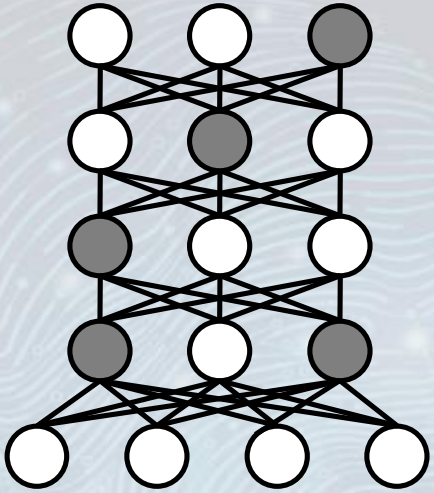
DropOut
학습시킬 때 일부러 정보를 누락시키거나
중간 중간 노드를 끊음.

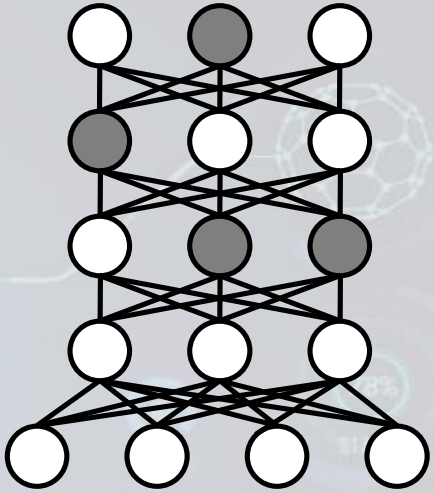
↓

일부에 집착하지 않고 중요한 요소가 무엇인지 학습

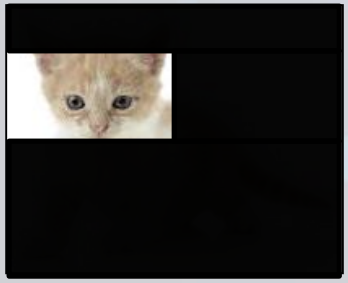
Overfitting







얼굴위주



색은 지우고

귀는 빼고

신경망

신경망이 똑똑해졌어요 ^^

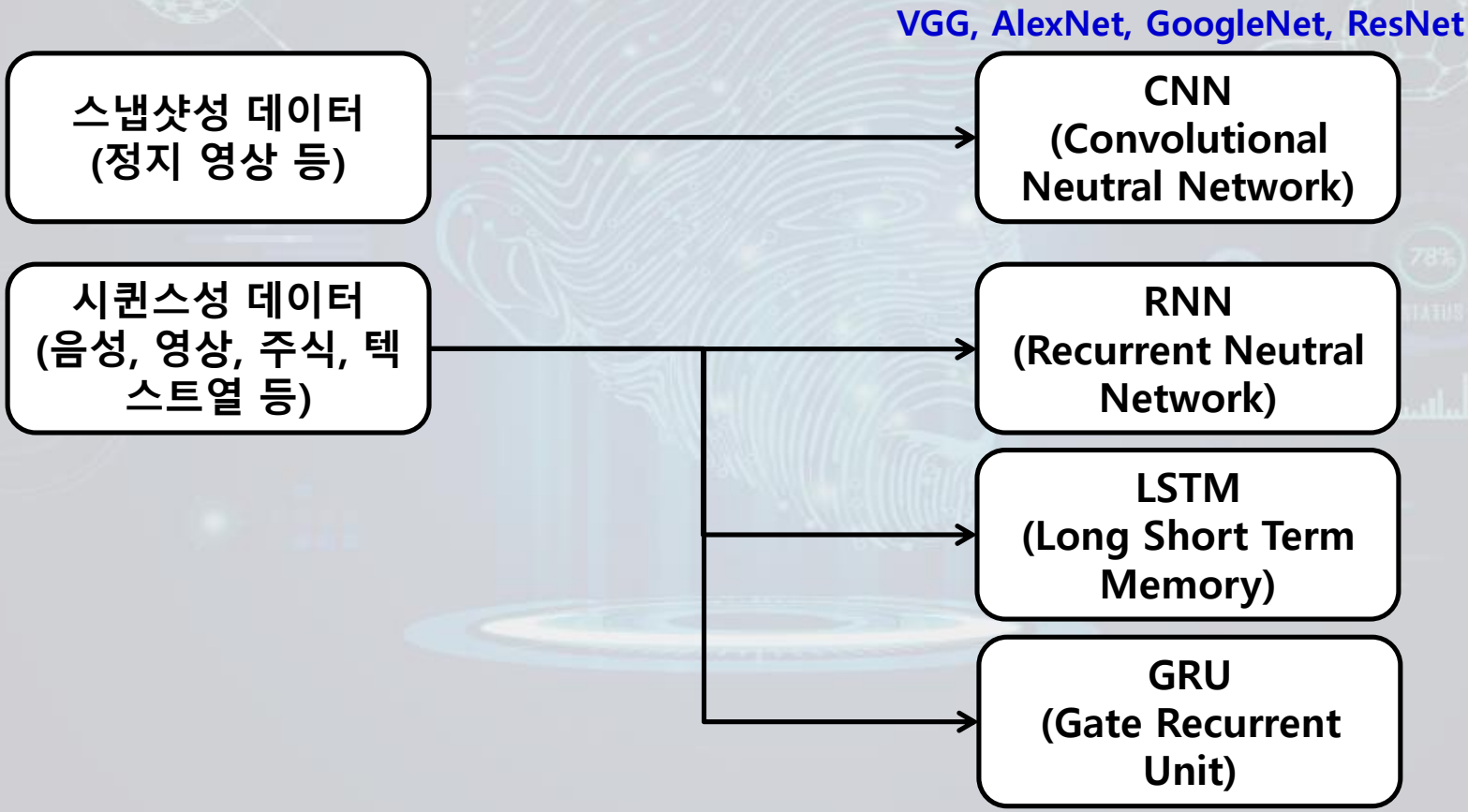
ReLU
Vanishing Gradient가 없어졌어요 !!

Adam
속도가 빨라졌어요 !!

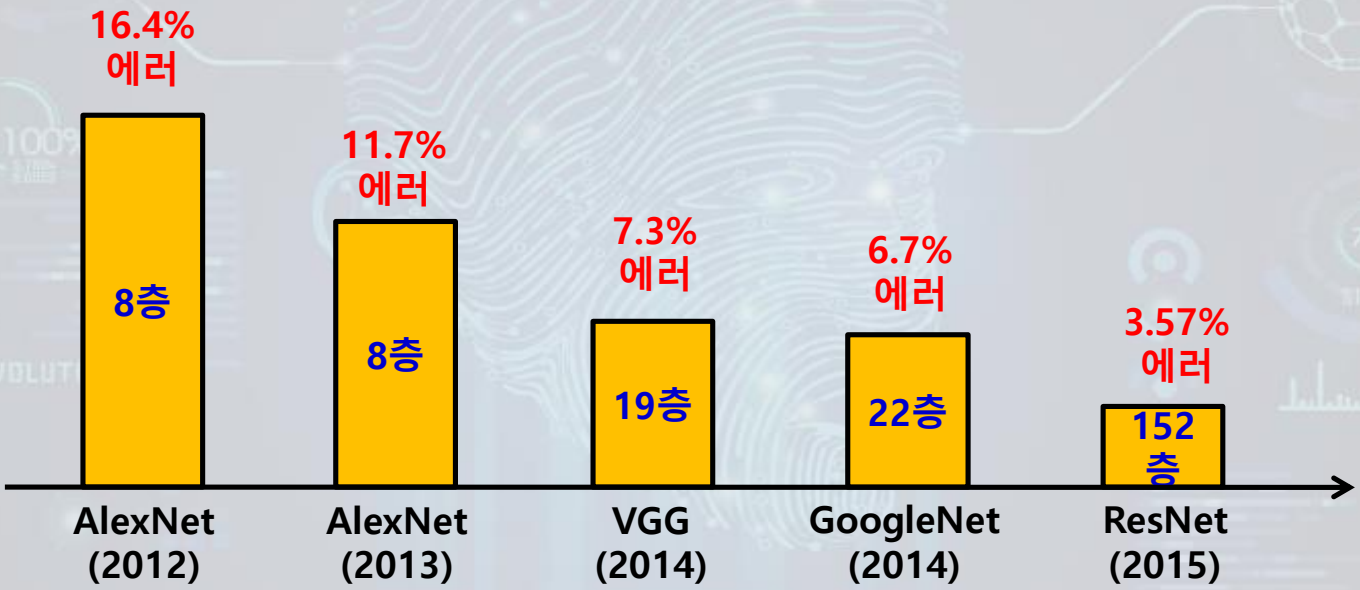
DropOut
유연성도 생겼어요 !!

신경망

이제 무얼 할 수 있을까요 ?



신경망의 층의 변화


16.4%
에러
8층
AlexNet
(2012)
11.7%
에러
8층
AlexNet
(2013)
7.3%
에러
19층
VGG
(2014)
6.7%
에러
22층
GoogleNet
(2014)
3.57%
에러
152
층
ResNet
(2015)