

# Self-supervised Deep RL

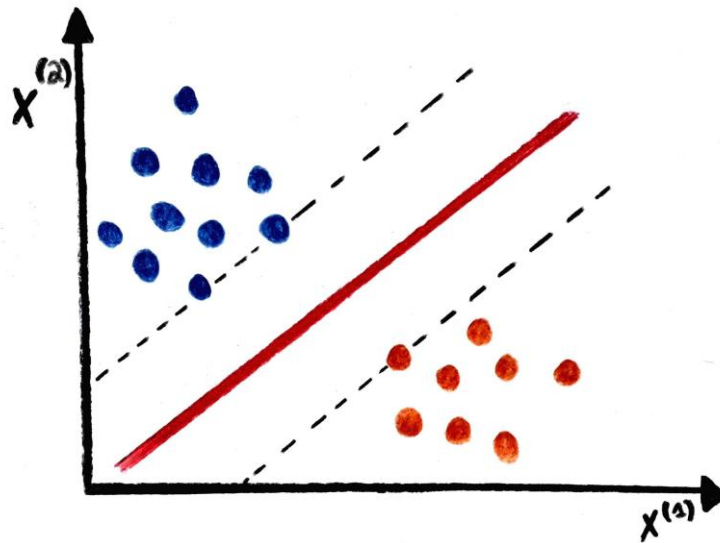
# 발표자



- **이름 : 박덕근 (deuk guen)**
  - 소속 : 한국디지털미디어 고등학교 해킹방어과 재학
- **관심분야**
  - Deep-Learning
  - GAN
  - Data pipe-line
- **연락처**
  - dkyoung2004@naver.com

# Abstract

기존 딥러닝 문제들은 Labeling이 된 다량의 데이터를 학습시키는 방식으로 모델을 구현했다.



그것이 "지도 학습"이니까..

# Abstract

더 정밀한 예측 가중치를 위해서는 많은 양의 데이터를 필요로 하는데,

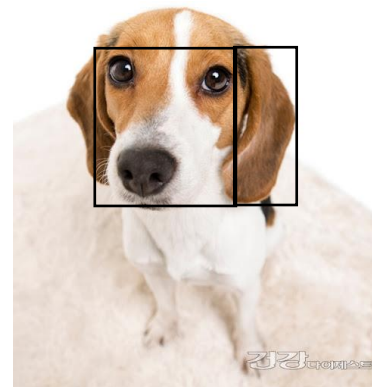
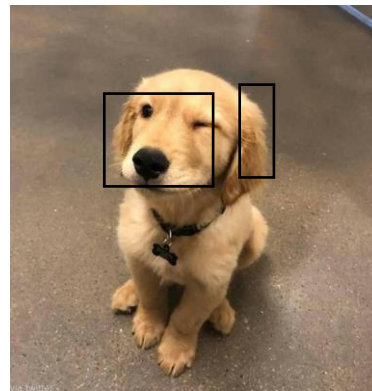
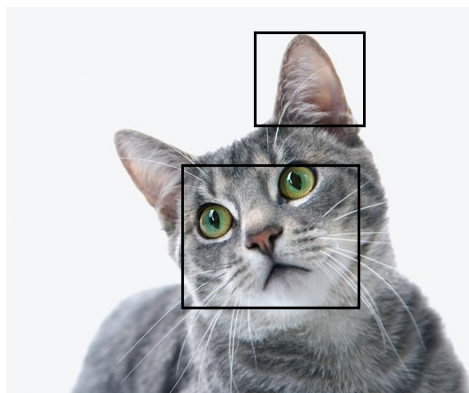
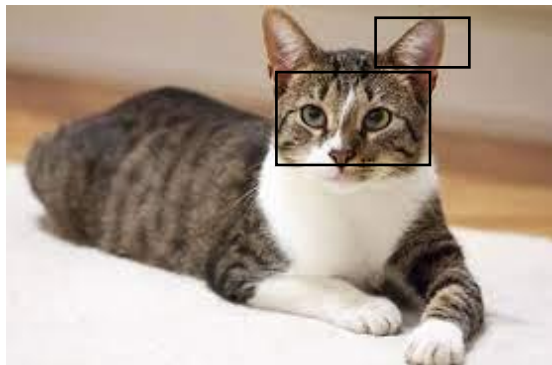
라벨링에 많은 자원이 할애되는 단점이 존재한다.

이 때문에 자체적으로 labeling 없이 가중치의 정밀도 확보를

목표로 하는 여러 알고리즘이 나오게 된다.

# 첫번째 접근, clustering

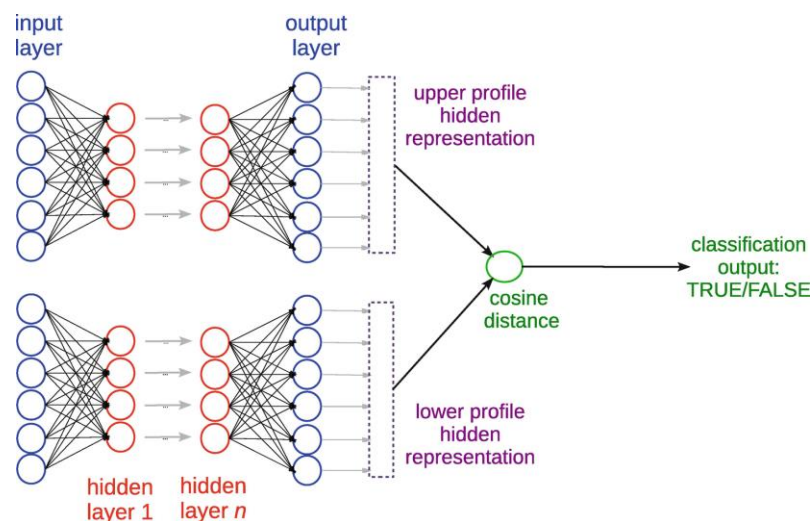
라벨링 없이 답을 찾게 하면 어떨까?  
데이터 간의 유사도를 통해서 스스로  
데이터 군집을 형성한다는 것 말이다.  
이것은 clustering 방식이며,  
최근 들어서 융합하려는 움직임이 생기고 있다.



## 2번째 접근 few-shot siam net

Deep neural networks를 사용하여 특징을 추출하는 점에서  
본 주제와 밀접하게 연관이 있다고 볼 수 있다.

라벨링 없이 CNN에 값을 넣고,  
검증 손실함수값을 최소화 하게끔 하는 CNN의 특성에 기인하여,  
각각 데이터간 범주의 코사인 유사도를 손실값으로 선정하여  
범주에 따른 객체의 특징을 얻어냄에 있다.



# Self-supervised learning

지금까지 이해한 바로는, 라벨값이 주어지지 않은 데이터셋에서 범주를 구분하는 함수를 만들어내는 알고리즘이다. 인 것 같은데,  
이를 강화학습에 적용하면 최적화를 위해  
각각의 방법이 최선인가, 아닌가,  
를 정해진 정책없이 분석하는 모양새인것 같다.  
센서를 통해 각각의 장애물이나 코너를 맞닥뜨렸을때,  
자신이 예상한 action과 reward, 그리고 실제로 일어난 것에 대한 오차를 찾으면서, 수정해 나가는 방식. 새로운 시각이다.

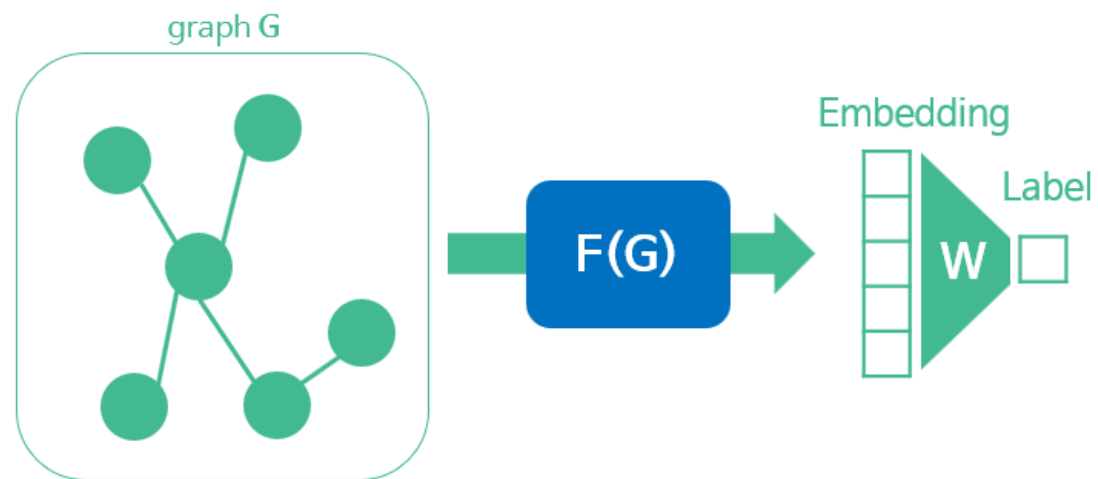
# 논문 요약

Self-supervised의 최적화 방식은 (예측상황 - 실제상황)<sup>2</sup> 에서 현재에 좀 더 집중하느냐 아니면 미래를 더 예측해야 하느냐 에서 갈린다.

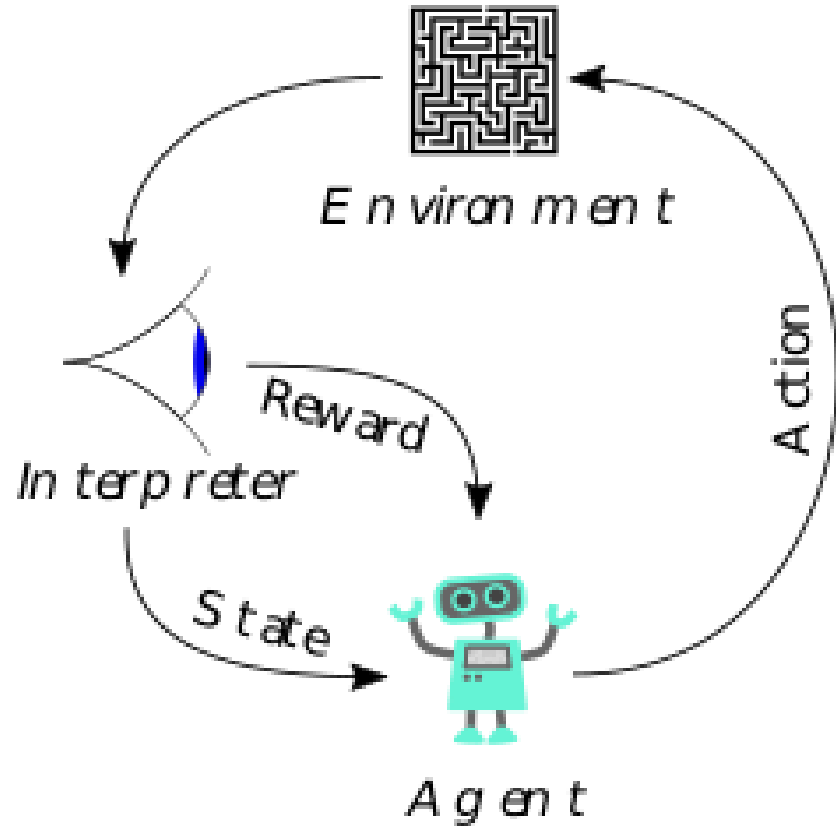
본 논문에서 역시 충돌을 최소화 하는 방향으로 환경, 정책, 행동 등을 예측하고, 실험체가 맞닥뜨린 실제상황과 얼마나 유사했는지, 그리고 나서 유기적으로 분석하는 것으로 보인다.



# 3번째 접근, graph Neural network



# What is Autonomous learning?



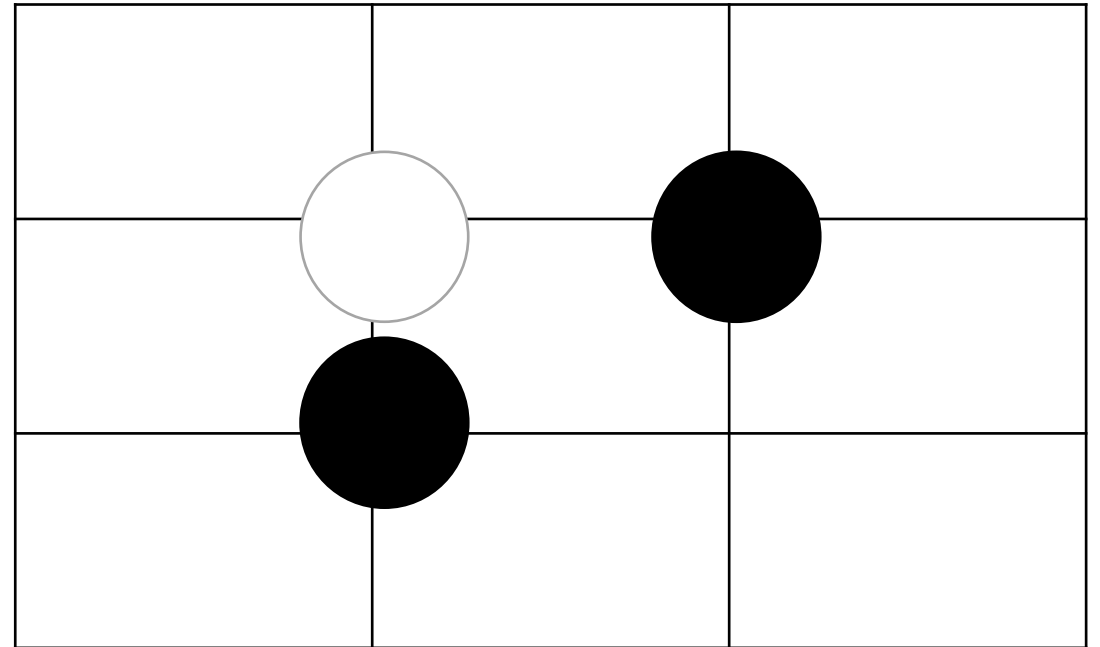
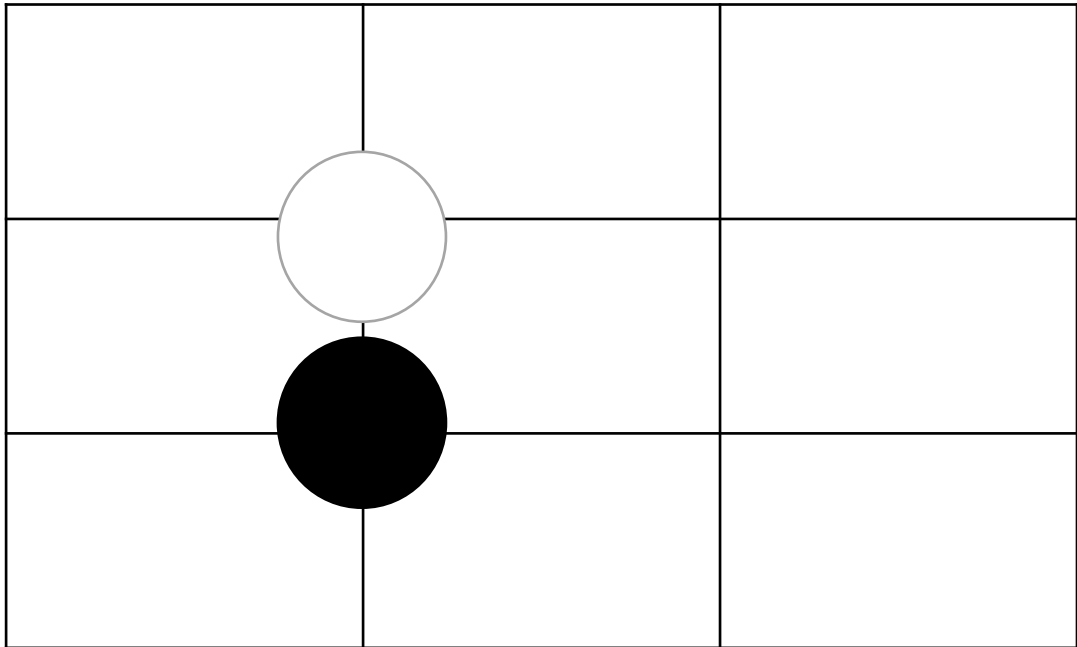
	논문명	Published in	연구 주제	특징	Dataset
1	Self-supervised Deep Reinforcement Learning with Generalized Computation Graphs for Robot Navigation	2018 IEEE International Conference on Robotics and Automation	로봇 네비게이션 계산그래프 일반화를 통한 자기-지도 강화학습	RC카와 환경의 상호작용으로 실험을 하여 검증한 점,	
2	Discriminative Unsupervised Feature Learning with Exemplar Convolutional Neural Networks	<a href="https://arxiv.org/pdf/1406.6909.pdf">https://arxiv.org/pdf/1406.6909.pdf</a>	라벨링 없는 데이터를 CNN에서 뽑아낸 특징으로 분류하였다.	CNN임에도 불구하고, 라벨링없이 분석을 진행하였다는 점.	
3	Mastering the game of Go without human knowledge				

# 이번 시간 피드백

- 1. 먼저 너무 추상적인 주제이다. 축소해보거라
  - 바둑으로 우선 국한 시켰습니다
- 2. 큰 주제 아래에 작은 주제들을 위치 시켜라.
  - 1. 바둑 알고리즘 논문읽기
  - 2. 실제 바둑 데이터셋, 혹은 강화학습으로 사전 데이터 셋 없이 만든 코드 확인
  - 3. self-supervised learning 적용

# 논문 briefing

알파제로는 DNN을 사용한다. (기호로는  $f_{\theta}$ )  
또한, 상태  $s$ 와 이에 대한 기존의 데이터를 입력 값으로 가집니다.



# 논문 briefing

아웃풋으로는 움직임에 대한 가능성,  
그리고  $s$ 에 대한 승리확률을 가져온다 합니다.

논문에서 제시한 용어 정리,  
 $v$ 는 스칼라, 현 상태  $s$ 에서 승리 가능성을 수치화 한 것.  
 $p$ 는 현 상태  $s$ 에서의 움직임  $a$  각각의 가능성을 수치화 한 벡터입니다.

# 논문 briefing

또한  $DNN(f_\theta)$ 은 정책 네트워크와 가치 네트워크로 이루어져 있습니다.

그리고 배치 정규화와 rectifier nonlinearities 가 있는 합성곱 층의 residual block으로 이루어져 있습니다.

\*residual block

# 논문 briefing

스스로 플레이하며 강화학습하는 알파제로


이때, MCTS(몬테카를로) 알고리즘을 각 상태  $s$ 에 적용하여 문제를 푼다고 합니다.

MCTS에서는 정책  $\pi$ 를 각 플레이마다 수정하는 방식으로 나아 갑니다.

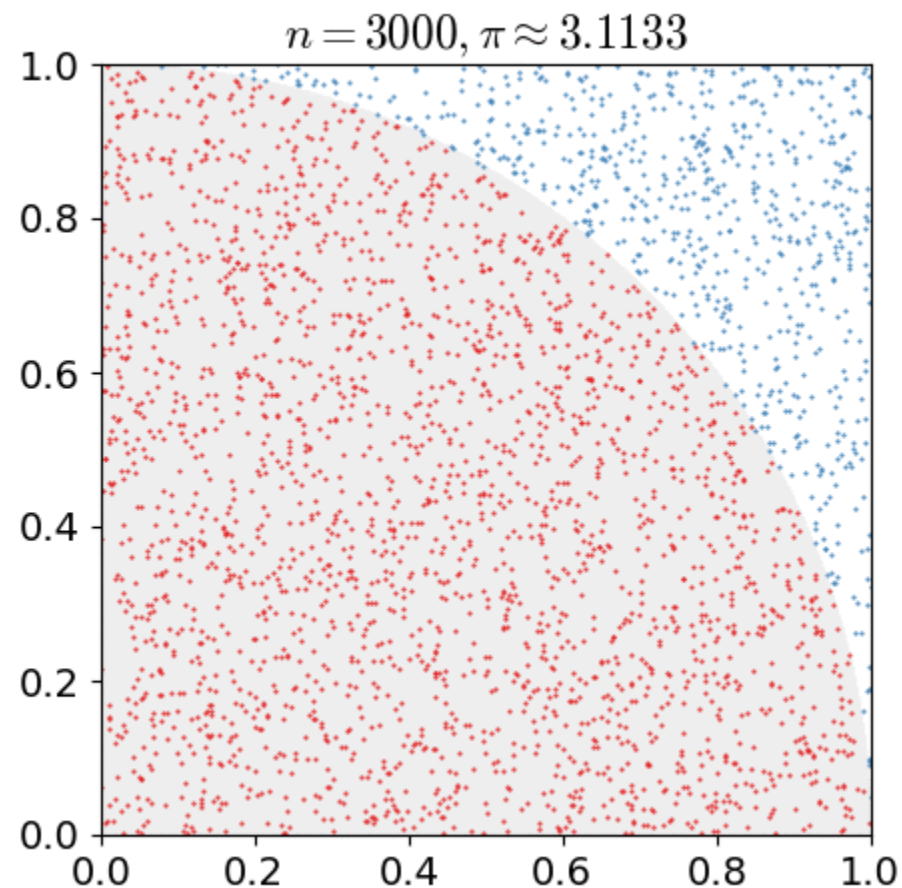


# MCTS

이것은 기존의 몬테카를로 근사입니다.

ITLAB >  mtcs.py > {} random

```
1 import random
2 n = 10000
3 count = 0
4 for i in range(n):
5     x = random.uniform(0,1.0)
6     y = random.uniform(0,1.0)
7     if (x**2+y**2)<=1:
8         count+=1
9 print(4*count/n)
10
11
```



# MCTS

답을 찾아내기 어려울 때에, 난수를 이용하여 함수의 값을 확률적으로 계산한다.

-> 어디까지나 "근사"다.

-> 횟수가 늘어날 수록 함수의 값에 "유사"해진다.

# MCTS

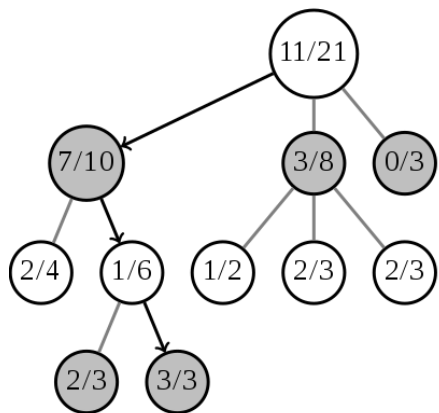
**인사 결정을 위한 "체험적 탐색 알고리즘" 으로,  
주로 게임을 할 때에 자주 사용됩니다.**

개념 자체에 대한 이해를 쉽게 하자면, 무작위로 수(움직임)을 뿌리고, 그 중에서 플레이어를 승리로 이끌었던 수를 따라가면서 왕도를 탐색하는 방식입니다.

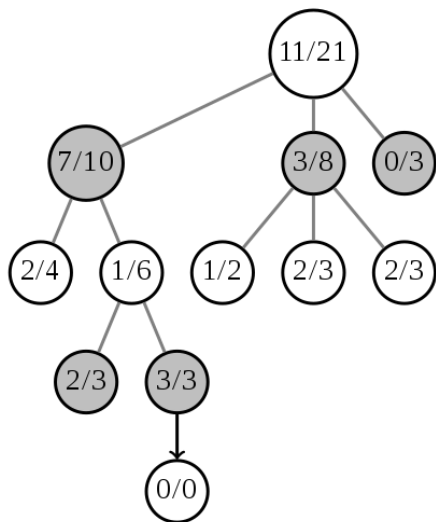
- > 승리 하는 방향의 전략으로 편향이 된다는 점.
- > 그러므로 의외성이 돋보이는 조커픽은 보기가 힘들다는것이 함정.

# MCTS

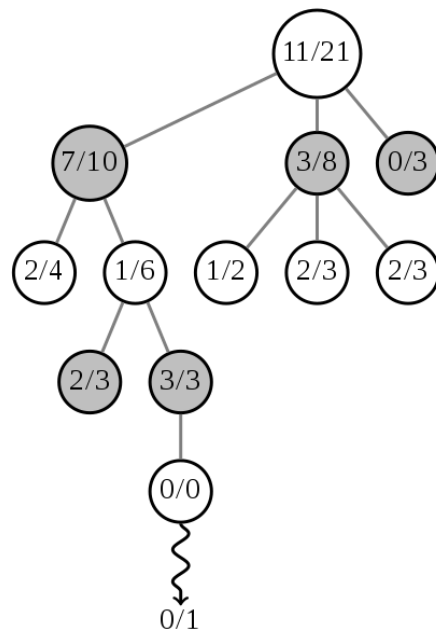
Selection



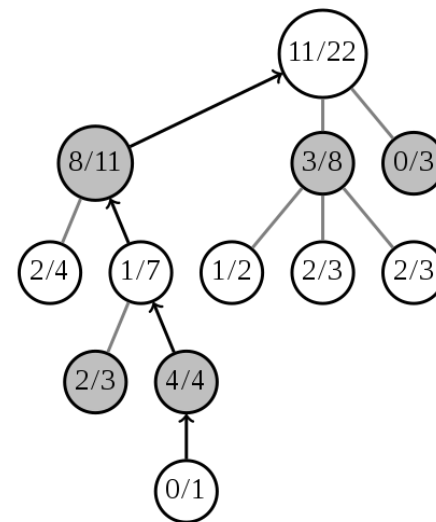
Expansion



Simulation



Backpropagation



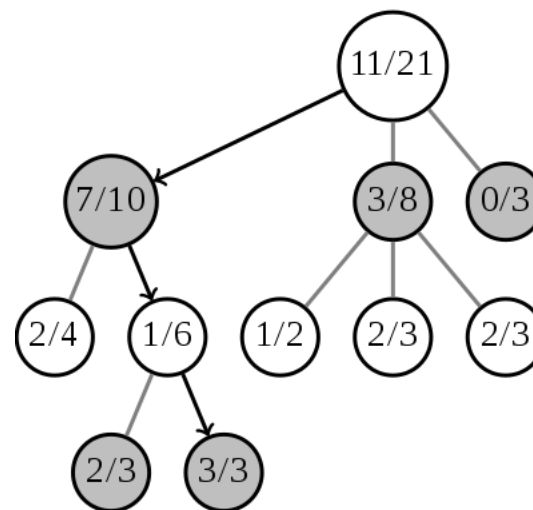
탐색트리의 형태를 띄고 있습니다.

# MCTS

## Selection

- Root에서 child로 내려가면서 leaf까지 도착합니다.
- 두가지 목표를 달성해야 하는데.
  - 새로운 방식을 찾아본다
  - 기존의 방식에서 가장 최적화된 방식을 찾는다.

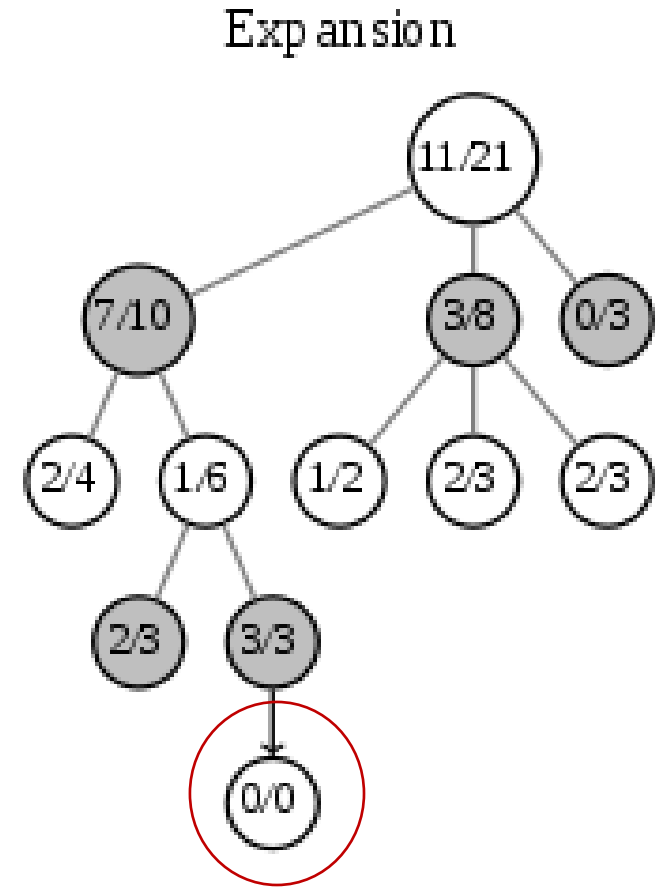
Selection



# MCTS

## Expansion(확장)

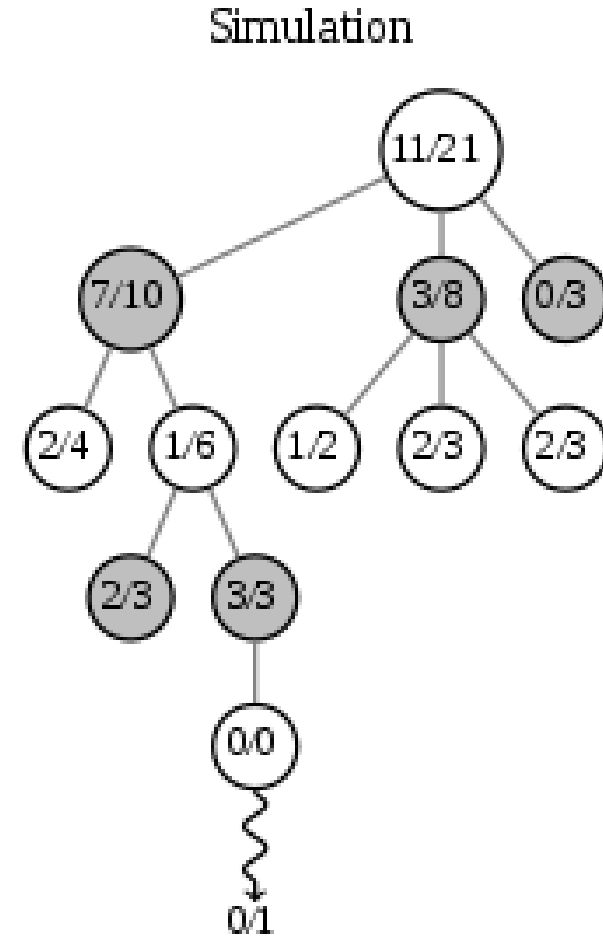
- 아무런 수확이 없이 종료가 될 경우에는 새로운 방식이 필요합니다.
- 이때, 완전히 랜덤인 자식노드로 빈도와 승률이 0인 노드를 추가하여 선택해보는 확장 단계를 거칩니다.



# MCTS

## Simulation

- 위에서 생성된 랜덤 노드로 위에서 말한 전체과정을 원하는 만큼 반복합니다.
- 또한 위에 과정에서는 복사본 트리를 사용하여 원본 트리에는 영향을 끼치지 않는 방향으로 진행합니다.

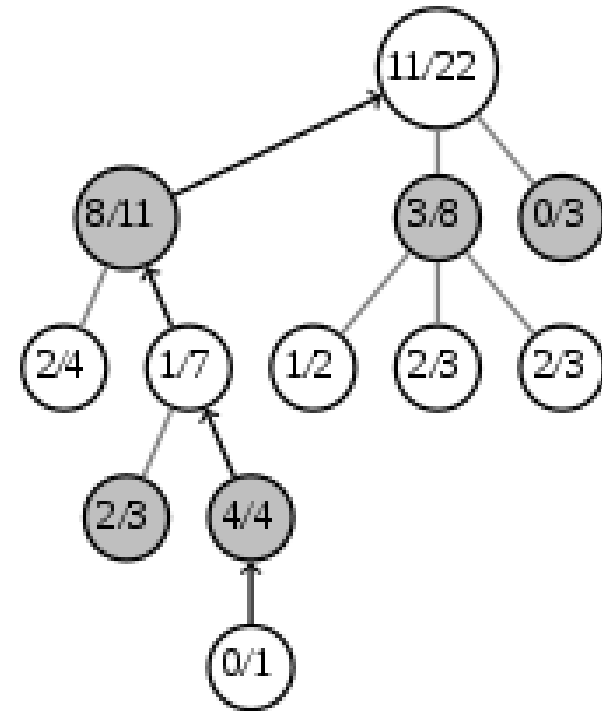


# MCTS

## Backpropagation

- 시뮬레이션의 결과로
- 승,패
- 빈도수
- 그리고 총 점수를 부모노드에 반영 시킵니다. 1대1로.
- 그리고 최종적으로 바뀐 점수들은 트리에 반영됩니다.

Backpropagation





# Figure 1

- 우선 스스로 게임을 합니다.
- 그리고 가장 마지막에 업데이트된  $f_\theta$ 를 사용합니다.
- 또한 MCTS는 각각의 수마다 작동됩니다.
- MCTS로 부터 가장 높은 승률의 수를 택합니다..
- 또한 마지막 상태 S에서 승자를 계산합니다.

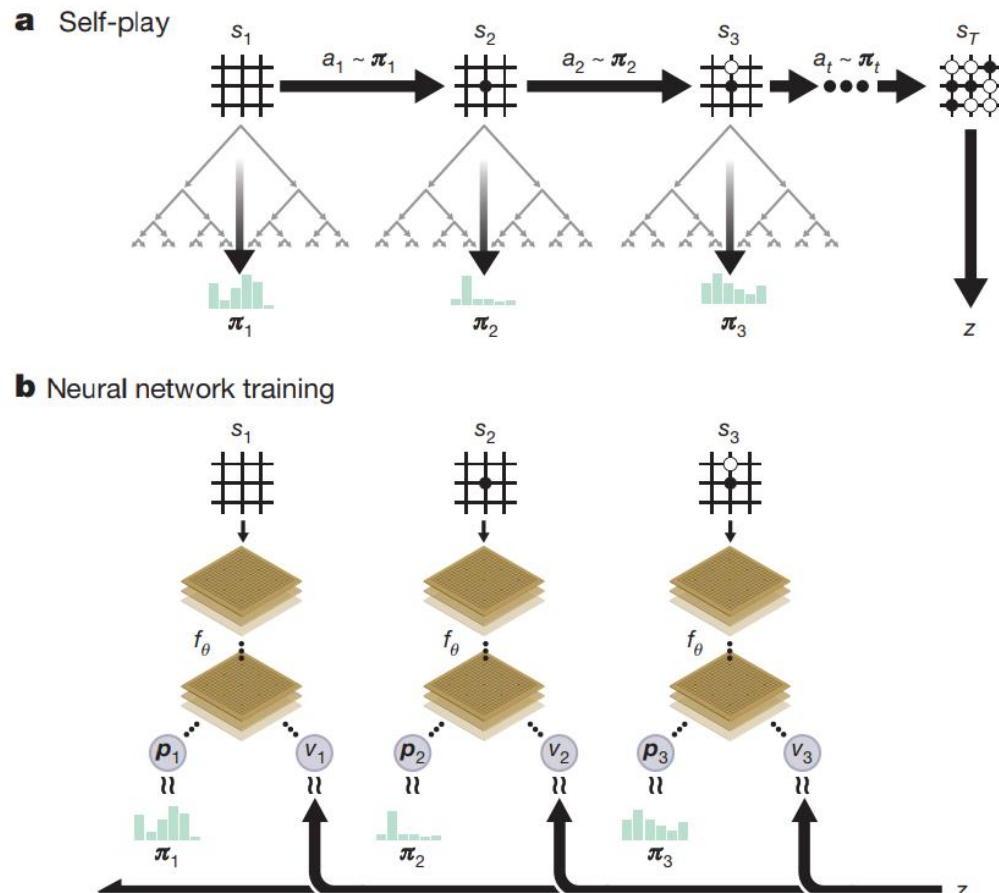
## Figure 1 -b-

- Input : raw position  $s(t)$
- Output : move probability,  
value( 현재의 상태에서 승자의 확률)
- 합성곱 층을 지난다고 합니다.  
(CNN)
- 다음 착수에 대한 예상의 유사도를 최대화 하는 것,  
승자에 대한 오차를 최소화 하는 것에 주안점을 둔다.

# 구동 원리

뉴럴넷은 다음 착수에 대한 예상과 승자에 대한 예측의 정밀도를 올리기 위해 업데이트 한다.

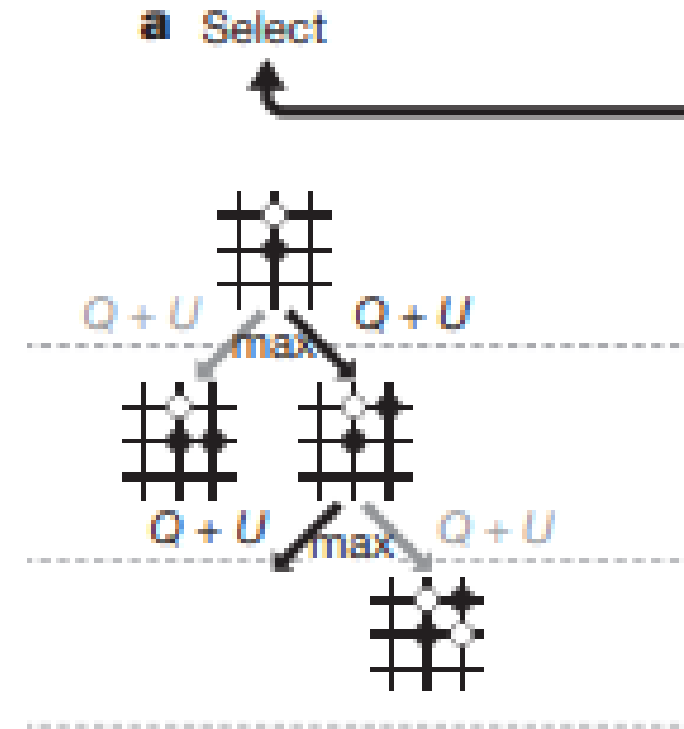
이렇게 최적화된 알고리즘으로 MCTS에서 사용하는 승리확률과 다음 착수에 대한 예상을 합니다.



# MCTS in Alpha zero

각 시뮬에서 최대의 이익을 가져다 주는 Q함수 구동

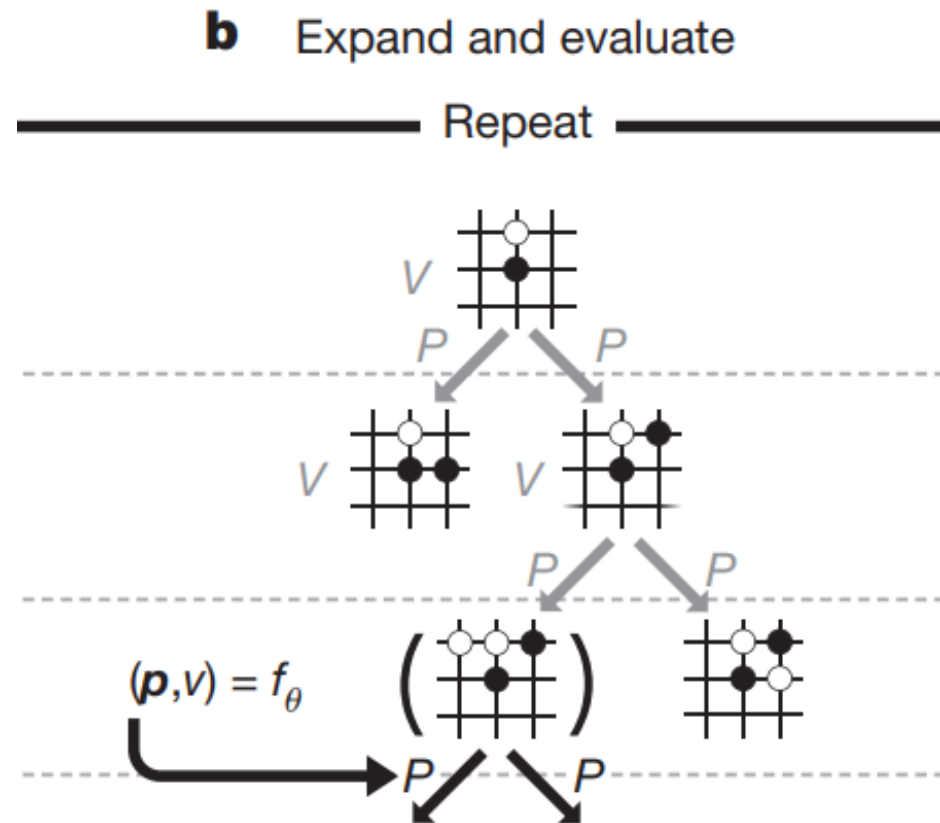
근데 여기서, 이전의 가능성  $P$ 와 방문 횟수로 산정하는 신뢰도  $U$ 를 만들어 반영한다.



# MCTS in Alpha zero

종단노드는 확장됨과  
동시에 뉴럴넷으로 관련있는  
상태  $s$ 를 계산합니다.

Probability인  $P$ 는 큐함수와 신  
뢰도  $U$ 의 합산 결과입니다.



# NN의 적용

