이산수학

Discrete Mathematics

알파고 알고리즘 이야기

인천대학교

INCHEON NATIONAL UNIVERSITY

꿈을 이루는 곳..

인천대학교

컴퓨터공학부

Computer Science and Engineering

Incheon National University Dept.of Computer Science & Engineering

인천대학교 컴퓨터공학과

공학시인 이숙 이철호 교수

Jullio@chol.com
zullio@inu.ac.kr
010-3957-6683

모바일컴퓨팅 연구실
07-401호

알파고에 대하여

- 알파고의 HW 사양

- 최종 버전(싱글)
  - 40개의 탐색 쓰레드
  - 48개 CPU
  - 8개 GPU를 사용
- 분산 구현 버전
  - 40개의 탐색 쓰레드
  - 1202개의 CPU
  - 176개의 GPU를 사용

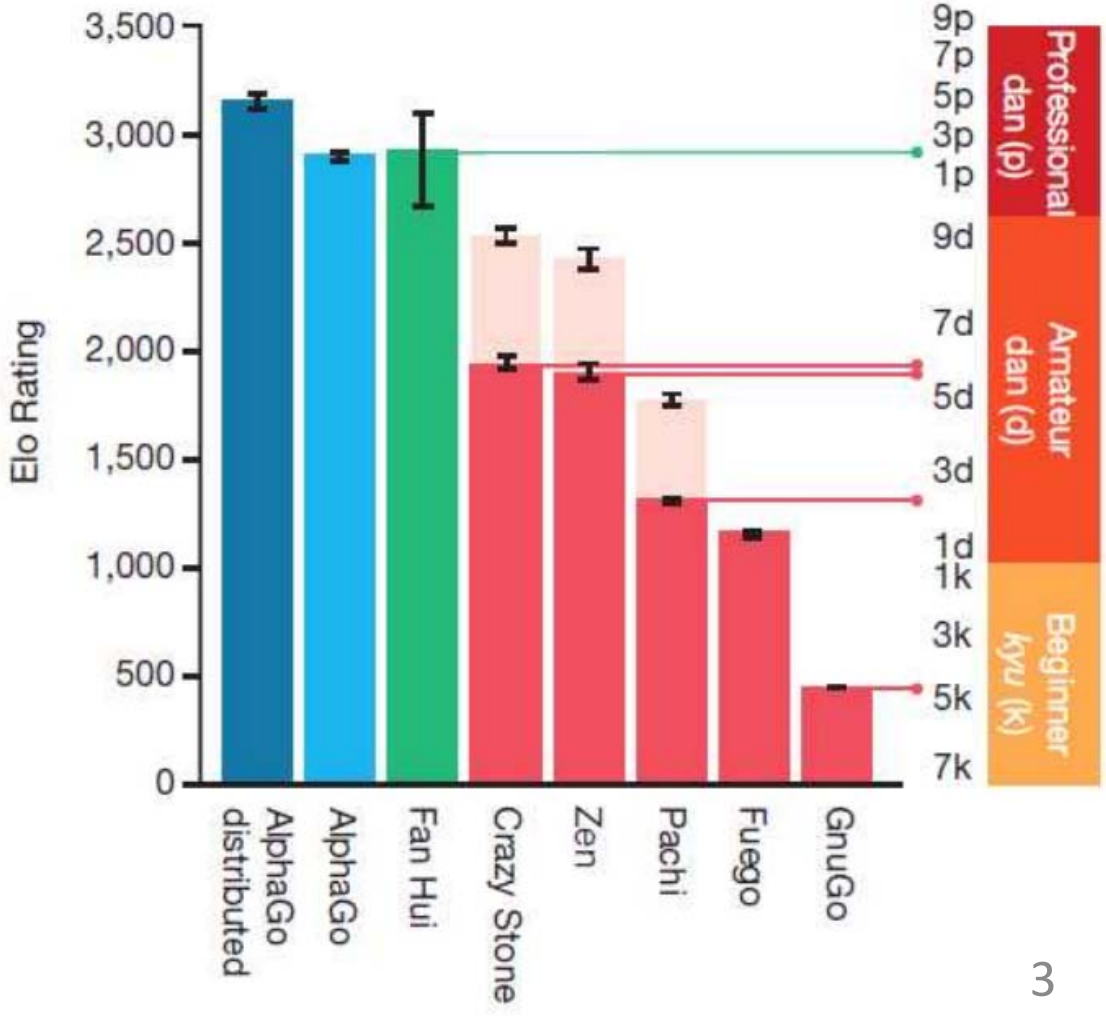
구 분	탐색 쓰레드	CPUs	GPUs	Elo rating[9)]
단일 (single)	40	48	8	2890
분산 (distributed)	40	1202	176	3140

탐색 쓰레드	쓰레드 개수만큼 바둑의 경기 경로를 탐색
CPU의 역할	CPU 한 개당 1초에 1000회 이상의 시뮬레이션 수행 ※ 이 기법은 fast rollout으로 자세한 내용은 후술
GPU의 역할	딥러닝을 사용하여 바둑판 상태의 승률과 다음 착수 예측

AlphaGo와 상용 바둑프로그램 성능 비교

ELO rating???

뒤에서 …



상용 바둑 게임 프로그램 비교

명칭	개발자 대표	출시 년도	최신 버전	사용 알고리즘	전적	수준
Crazy Stone	Coulom R (프랑스)	2005	2015	MCTS + Pattern Learning (Bradly-Terry 모델 적용)	2007, 2008 UEC 컵 우승 2013년 제 1회 전성전에서 이시다 9단에게 4점 접바둑 승리	6d
Zen	요지 오지마 (일본)	2009	5	MCTS	2009, 2011 컴퓨터 올림피아드 우승, 2012년 다케미야 아사키 9단에게 4점 접바둑 승리	6d
Pachi	Baudis P (체코)	2012	11	수정 MCTS + UCT + RAVE (오픈소스)	2011년 인간 vs 컴퓨터 바둑 대결(파리)에서 주준훈(중국) 9단 7점 접바둑 승리	2d
Fuego	Baudis P (캐나다)	2010	SVN 1989	MCTS + UCT (오픈소스)	2009년 프로그램 최초로 9x9 바둑에서 주준훈(중국) 9단 프로에게 이김	1d
GnuGo	GNU-FSF	1989	3.8	MCTS(3.8 버전) Bouzy's 5/21(2.6 버전) (오픈소스)		5k

바둑 인공지능 프로그램	버전	착수시간 설정	CPU	GPU	KGS 단수[10]	Elo Rating
분산 AlphaGo		5초	1202	176	–	3140
AlphaGo		5초	48	8	–	2890
CrazyStone	2015	5초	32	–	6d	1929
Zen	5	5초	8	–	6d	1888
Pachi	10.99	5초	16	–	2d	1298
Fuego	svn1989	5초	16	–	1d	1148
GnuGo	3.8	5초	1	–	5k	431

ELO rating???

참고 : http://egloos.zum.com/webpd77/v/4102040

- 실제 실력으로 순위를 정하자'라는 취지에 부합되는 시스템
- 전적 누계 방식
  - '승이든 패든 많은 게임을 해서 전적을 많이 쌓으면 순위가 올라가는' 방식
- 각 플레이어를 A, B라고 할 때, 그에 대응되는 승률 E

R은 현재 플레이어의 레이팅 점수

$$E_A = \frac{1}{1 + 10^{(R_B - R_A)/400}}.$$

$$E_B = \frac{1}{1 + 10^{(R_A - R_B)/400}}.$$

ELO rating???

참고 : http://egloos.zum.com/webpd77/v/4102040

$$E_A = \frac{1}{1 + 10^{(R_B - R_A)/400}}. \qquad E_B = \frac{1}{1 + 10^{(R_A - R_B)/400}}.$$

- R은 현재 플레이어의 레이팅 점수
- 레이팅 점수
  - 각 플레이어의 '실력'을 나타낸다고 생각하면 됨
  - 처음 시작하는 사람의 수준은 임의의 한 숫자로 나타냄.
    이를테면 ' 1500 '

만약 A플레이어가 B플레이어보다 레이팅 400점이 높다면,

EA는 약0.9090...,

EB는 약 0.0909...

두 승률을 합치면 1, 즉 100%

A플레이어가 이길 확률이 약 90%,

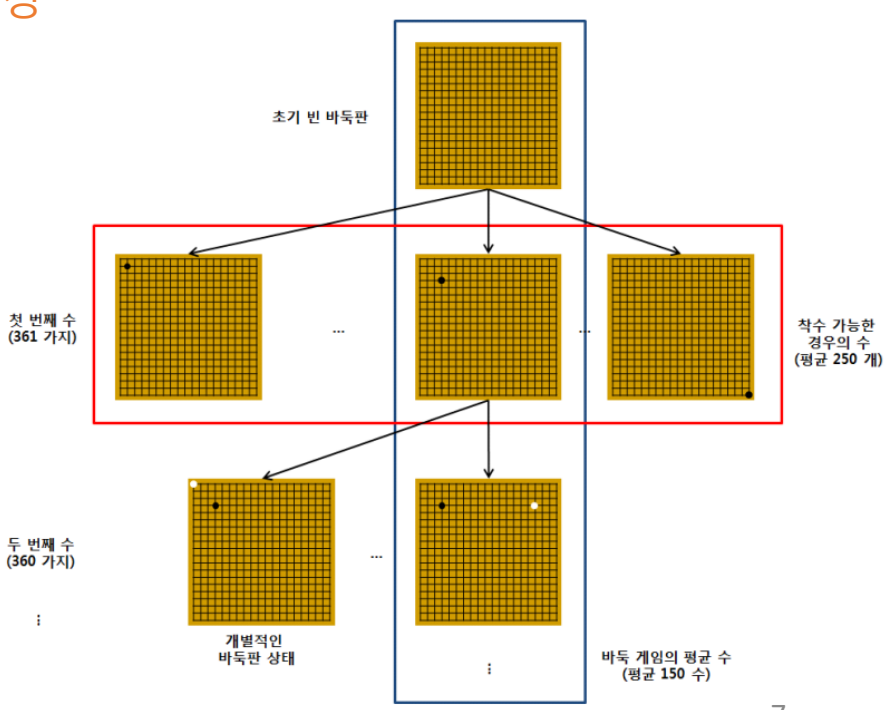
B플레이어가 이길 확률이 약 10%

게임 트리 탐색 알고리즘

- 게임에서 장기, 바둑과 같이 두 플레이어가 번갈아가면서 한번씩 게임을 하는 방식

특정 게임 상태에서
다음 수를 예측하기 위해서는 수 읽기를 통해 가장
승리할 확률이 높은 곳을 결정하는 알고리즘.
이 과정이 트리의 탐색이고,
이론적으로는 현재 상태에서
가능한 모든 결과를 미리 알 경우
가장 승률이 높은 수를 선택하는 방법

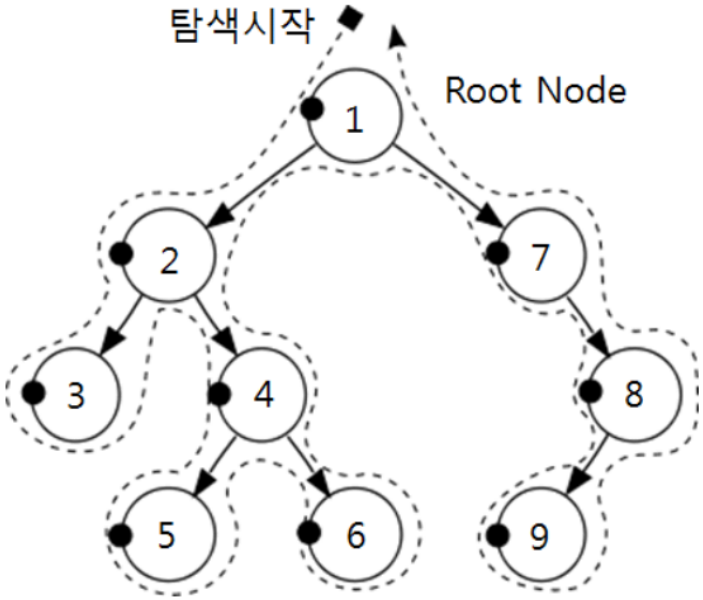
알파고의 인공지능 바둑 프로그램은
250^{150}의 경우의 수를 모두 탐색하지 않고,
제한된 시간 안에
가장 승리할 가능성이 높은 경로를 탐색.
탐색의 전략이 인공지능 바둑 프로그램의 성능을 좌우함



[그림 1] 바둑 게임의 흐름

게임 트리의 탐색 알고리즘

트리 탐색 기법(트리 순회 : Tree traversal)은 트리 구조에서 각각의 노드를 한번씩, 체계적인 방법으로 방문하는 과정.
트리 탐색 기법에는 탐색 순서에 따라 전·중·후위 및 레벨 순서 순회 기법이 있음.
이중 **전위 순회(preorder)**는 **깊이 우선의 탐색(depth-first search: DFS)** 이라고도 하며, MinMax 알고리즘의 탐색 방법임.



깊이 우선 탐색(전위 순회)의 과정

<탐색 순서>
1. 루트 노드에서 시작
2. 왼쪽 자식 노드를 방문
   - 왼쪽 서브트리의 전위 순회
3. 오른쪽 자식 노드를 방문
   - 오른쪽 서브트리의 전위 순회

<탐색 노드의 순서>
1→2→3→4→5→6→7→8→9

바둑에서의 탐색 알고리즘

- 바둑(19 X 19)
- 보다 탐색 정도가 낮은 체스(8 x 8)의 경우
  완전한 게임 트리에는 약 1040개의 노드가 존재
  (약 35^{80} 가지의 경우의 수)
- 효율적인 탐색을 위해 휴리스틱(Heuristic)기법
  깊이 또는 너비 우선 탐색 기법이 사용.
- 바둑과 같은 복잡한 게임에서는
  충분한 도움이 되지 않음.
- 바둑은 게임 중에서도 극단적으로 계산 량이 많음.
  가장 어려운 문제로 알려져 있음.
  (약 250^{150} 가지의 경우의 수)

몬테카를로 방법
(Monte Carlo method)

- 난수를 이용하여 함수의 값을 확률적으로 계산하는 알고리즘.

- 수학이나 물리학 등에 자주 사용.

- 계산하려는 값이 닫힌 형식으로 표현되지 않거나 복잡한 경우에 근사적으로 계산할 때 사용.

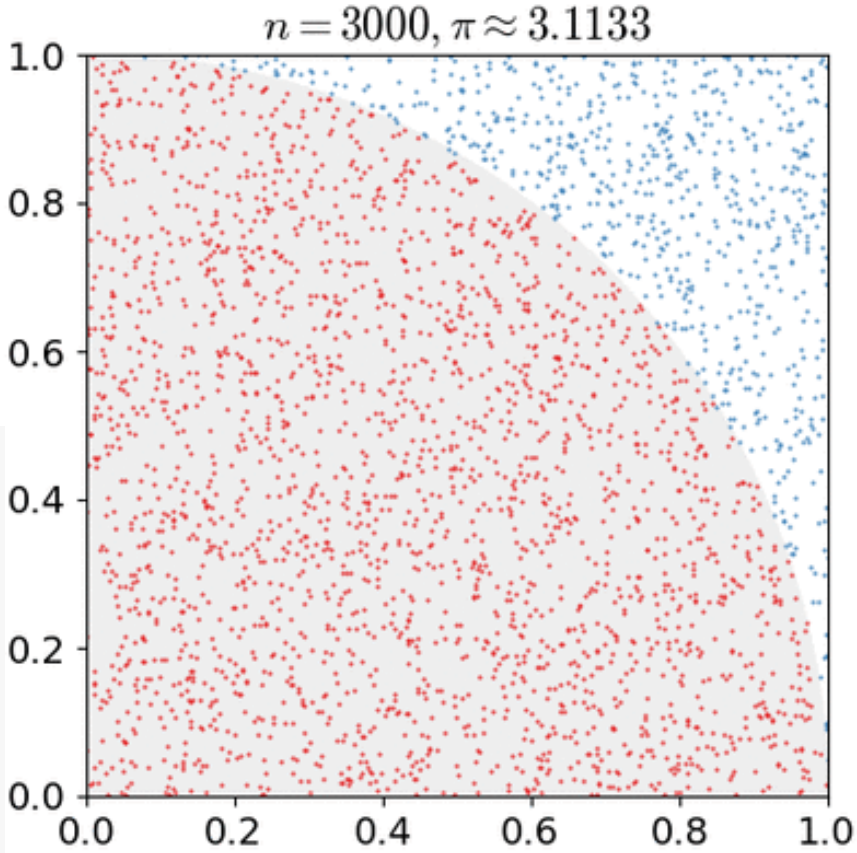
- 모나코의 유명한 도박의 도시 몬테카를로의 이름에서 명명.

몬테카를로 알고리즘

출처 : https://ko.wikipedia.org/wiki/

- **원주율 계산 알고리즘**

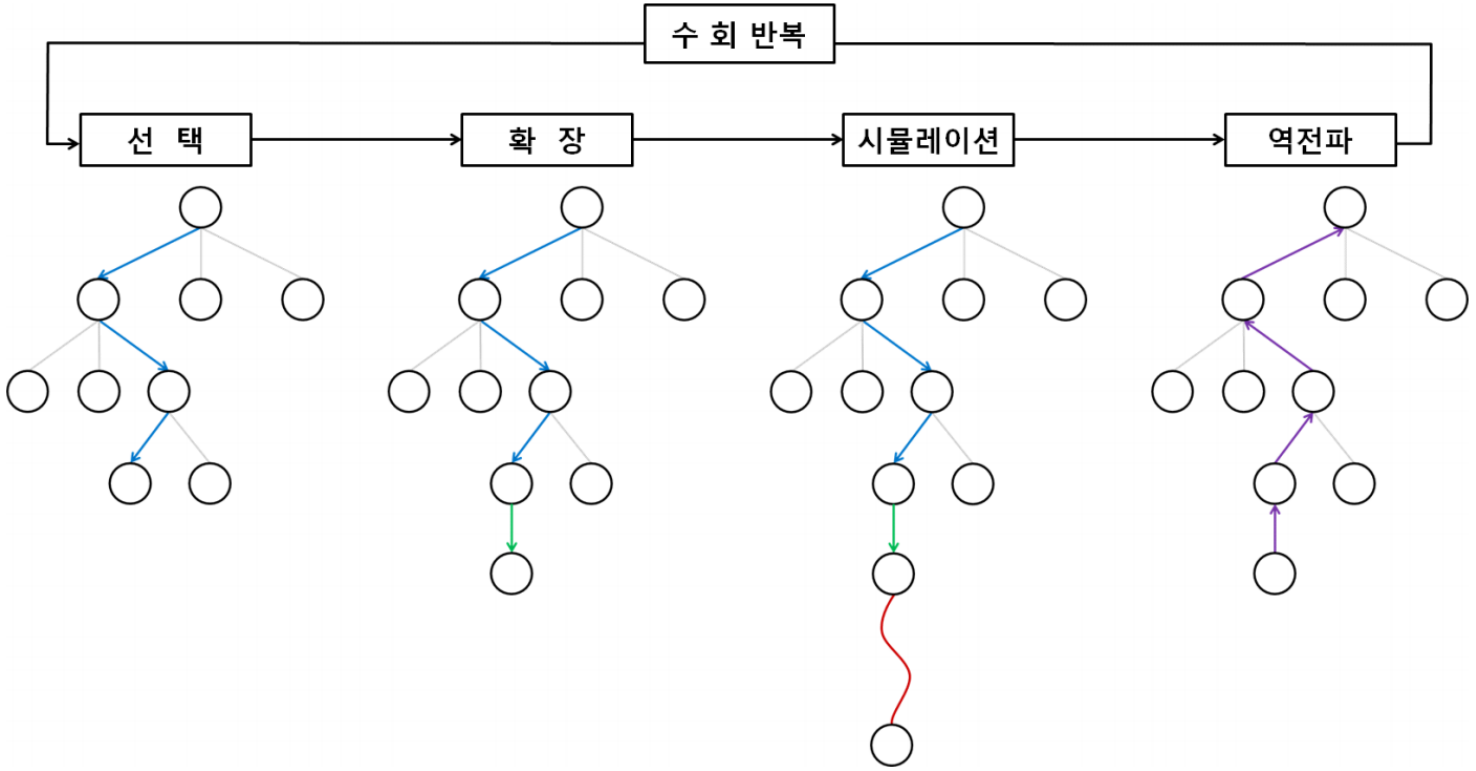
$[0,1] \times [0,1]$에서 점 (x,y)를 표집.
표집한 점의 중심이 $(0,0)$에 있고,
반지름이 1인 원에 속하는지 계산.
원의 정의에 따라 $x^2 + y^2$와 1을
비교함으로써 계산.
위의 두 과정을 충분히 반복하여,
원에 속한 점들의 개수를 계산.



표집 영역과 원의 공통 영역은 $\pi/4$ 의넓이를 가지며,
전체 점 갯수를 원에 속한 점 갯수로 나눈 비율은 이 값을 근사화

몬테카를로 트리 탐색 알고리즘
MCTS(Monte Carlo Tree Search)

- 바둑에서 가장 널리 사용되는 인공지능 알고리즘
- MCTS는 최소-최대 알고리즘의 성능을 개선한 것
- 모든 경로를 탐색하는 것이 불가능 할 때 효율적



MCTS의 4단계 과정

①선택

현재 바둑판 상태에서 특정 경로로 수읽기를 진행

②확장

일정 수 이상 수읽기가 진행되면 그 지점에서
한 단계 더 착수 지점을 예측(게임 트리의 확장)

③시뮬레이션

②에서 선택한 노드에서 바둑이 종료될 때까지
무작위(random) 방법으로 진행.
속도가 빠르기 때문에 여러 번 수행할 수 있으나
착수의 적정성은 떨어짐

④역전파

③의 결과를 종합하여 확장한 노드의 가치
(②에서 한 단계 더 착수한 것의 승산)를 역전파하여
해당 경로의 승산 가능성을 갱신

MCTS의 핵심 요소

- 정책

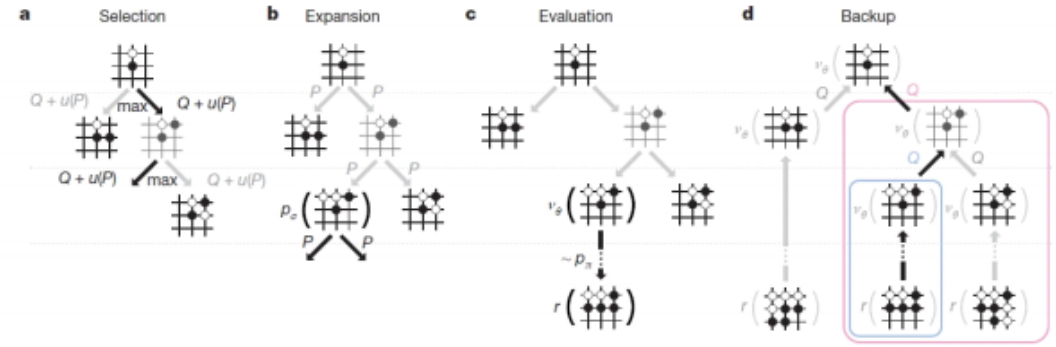
  트리의 폭을 제한하는 역할
  MCTS의 두 번째 단계인 확장에서 주로 사용
  특정 시점에서 가능한 모든 수 중 가장 승률이 높은 것을 예측

- 가치

  트리의 깊이를 제한하는 역할
  가치는 현재 대국상황의 승산을 나타낸 것
  승산이 정확할수록 많은 수
  (더 깊은 노드)를 볼 필요가 없음

스스로 대국하는 학습기법을 통해 정책과 가치의 성능을 향상시킴

AlphaGo의 MCTS



[그림 8][19] AlphaGo의 몬테카를로 트리 탐색 (MCTS)

- **Step a. 선택 (selection) : 현재 바둑판(t)에서 특정 시점(L)까지 착수 선택**

  현재 바둑판 상태에서 $Q+u$값이 최대가 되는 지점을 선택. 여기서 Q값은 MCTS의 가치값 등으로 정해진 것으로 높을수록 승리할 확률이 높음. u값은 바둑판 탐색의 폭을 넓히기 위해서 고안된 변수 (노드의 방문횟수에 반비례)

- **Step b. 확장 (expansion) : 탐색 경로의 마지막 노드(L)를 확장시킴**

  <표 4>의 두 번째 단계인 확장과 동일하게 특정 시점까지 선택이 된 노드로부터 확장(child node 생성)을 수행. AlphaGo에서 확장을 하는 기준은 마지막 노드(L)의 방문 횟수가 40회 이상인 경우

- **Step c : 평가 (evaluation) : 마지막 노드(L)의 승산 평가**

  마지막 노드(확장이 된 경우 확장된 노드, L+1)의 가치를 평가하기 위해서 마지막 노드 시점부터 게임 종료까지 고속 시뮬레이션(fast rollout[20])을 수행. 시뮬레이션의 평균값(r)과 딥러닝으로 추정한 가치값(v_θ)을 통해서 마지막 노드의 가치를 평가함

  ※ 시뮬레이션 평균값(r)과 딥러닝을 사용한 가치값(v_θ)의 비율은 같음

- **Step d : 갱신 (backup) : 바둑판 상태의 가치값 갱신**

  시작 지점(t)에서 마지막 노드(L 또는 L+1)까지의 경로에 있는 노드의 Q값 갱신

- **착수는 가장 많이 방문한 노드로 결정**

  ※ 가치가 가장 높은 노드를 결정할 경우 과적합 문제가 발생

AlphaGo의 차별성 – 딥러닝(Deep Learning)

- 딥러닝을 활용하여 전문 바둑기사들의 패턴을 학습함.
  - 바둑 기보를 19x19 픽셀을 갖는 이미지로 입력받아 전문 바둑기사의 다음 착수를 학습하는 과정.
    * 바둑 입문자가 기보를 공부하여 바둑기사들의 패턴을 습득하는 것과 유사함.
    * AlphaGo 개발자인 데이비드 실버는 "AlphaGo는 16만개의 기보를 5주만에 학습했다" 라고 밝힘.

  - AlphaGo는 딥러닝 기법 중 특히 이미지 처리에 강한 컨볼루션 신경망을 기반으로 학습하기 때문에 국지적인 패턴인식에도 강점을 가짐
    * 바둑에서 지역적인 대국이 전체적인 형세 판단에 매우 중요한 역할을 함
- 바둑기사의 착수를 학습한 것은 정책 네트워크임
- 국지적인 패턴 인식을 통한 승산 판단은 가치 네트워크로 구현
- 정책과 가치 네트워크는 MCTS에서 게임 트리를 탐색할 때 적용됨

AlphaGo의 정책과 가치 네트워크

- **정책 네트워크(Policy Network)**
  - **정책 계산을 위한 딥러닝 신경망**

  - 정책 네트워크에서 사용된 딥러닝 기법은 컨볼루션 신경망(Convolution Neural Network, CNN)으로 19x19 바둑판 상태를 입력하여 바둑판 모든 자리의 다음 수 선택 가능성 확률 분포를 출력.

  * 컨볼루션 신경망은 페이스북의 얼굴 인식 기술인 DeepFace에 적용된 기술로 입력 이미지를 작은 구역으로 나누어 부분적인 특징을 인식하고, 이것을 결합하여 전체를 인식하는 특징을 가짐.

  - 바둑에서는 국지적인 패턴과 이를 바탕으로 전반적인 형세를 파악하는 것이 중요하므로 컨볼루션 신경망을 활용하는 것이 적절한 선택

정책 네트워크 학습

- 지도학습(supervised learning)

  프로 바둑기사들의 착수 전략 학습

  * KGS Go Server 프로 6단에서 9단 사이의 실제 대국 16만개 기보로부터 3000만 가지 바둑판 상태를 추출하여 데이터로 사용함
  * 이 중 약 2900만 개를 학습에 이용하고, 나머지 100만 가지 바둑판 상태를 시험에 이용 (정확도 57%). 이것은 사람이 다음 수를 두는 경향을 모델링 한 것
  * 50개의 GPU를 사용하여 학습 (기간 : 3주, 3억4천 번의 학습과정)

- 강화학습(reinforcement learning)

  스스로 경기하여 지도학습을 강화함

  * 지도학습의 결과로 구해진 정책네트워크는 사람의 착수 선호도를 표현하지만

  이 정책이 반드시 승리로 가는 최적의 선택이라고 볼 수 없음

  * 이것을 보완하기 위해 지도학습으로 구현된 정책 네트워크와 자체대결(self-play)을 통해 결과적으로 승리하는 선택을 "강화" 학습함

  약 128번의 자체대결을 수행

  * 이로부터 도출된 경기 결과(reward)를 바탕으로 이기는 방향으로 가도록 네트워크의 가중치를 강화(개선).

  강화학습 후의 정책 네트워크로도 기존 바둑 프로그램인 Pachi와 대결하여 85%의 승률

  * 50개의 GPU를 사용하여 학습 (기간 : 1일)

가치 네트워크(Value Network)
바둑의 전체적인 형세를 파악

- AlphaGo에서는 가치(value)를 계산하기 위해

  딥러닝을 이용한 가치 네트워크(value network) 사용

  * 기존 프로그램의 가치함수는 비교적 간단한 선형 결합으로 표현

    인공신경망을 활용하여 더 정확한 값을 기대할 수 있음

- 인공신경망의 입력층과 은닉층 구조는 정책네트워크와 유사한 컨볼루션 신경망이지만

  출력층은 현재의 가치(형세)를 표현하는 하나의 값(scalar)이 나오는 구조

- 특정 게임 상태에서의 승률(outcome)을 추정

  * 강화학습의 자체대결에서 생성된 3천만 개의 바둑상태로부터 가치 네트워크를 학습함
  * 게임에서 이길 경우의 승률을 1이라고 볼 때, 가치 네트워크의 오차는 약 0.234 수준

  (강화학습의 자체대결로 학습된 신경망을 시험(test)한 오차)

  * 50개의 GPU를 사용하여 학습 (기간 : 1주)

AlphaGo의 컨볼루션 신경망

- 컨볼루션 신경망 개요
  - 컨볼루션 신경망은 이미지나 비디오에서 객체의 분류에 특화된 방법
  - 이미지의 객체분류는 전통적인 인공신경망인 다층 퍼셉트론으로도 충분히 가능했으나, 노드간 링크가 모두 연결되어 있는 구조(fully-connected)가 갖는 한계 때문에 그 대안으로 컨볼루션신경망이 부상함
  - 이미지 처리(Image processing) 분야에서의 컨볼루션은 필터(커널)을 지칭하고, 이 컨볼루션 필터로 원본 이미지를 처리하여 특징을 추출해 냄
  - 바둑에서 컨볼루션 필터의 의미는 국소적, 지역적인 대국의 특징을 추출해내서 전반적인 형세를 추론하는 도구로 볼 수 있음

AlphaGo의 컨볼루션 신경망

- AlphaGo에서 사용된 컨볼루션신경망 구조
  - 특정 바둑상태는 19x19의 행렬에 대하여 48가지 특징을 추출
    - 흰 돌, 검은 돌, 빈 칸, 축, 활로, 과거기록 등
    - 각 각 48가지 특징 맵(feature map) 19x19의 이진 행렬로 구성됨
    - 컨볼루션 신경망의 미지수는 필터의 가중치 값

- 신경망 구조
  - 입력층 : 특정 대국에 대한 48가지 특징 맵
  - 은닉층 : 13개의 컨볼루션 층
  - 결과층 : 착수 가능한 다음 수의 확률 분포 (19x19, 정책 네트워크)
    현재 대국에서의 승산 (스칼라, 가치 네트워크)

AlphaGo의 컨볼루션 신경망

- 컨볼루션 층의 상세 구조
  - 컨볼루션 필터는 k개로 AlphaGo에서는 k=128, 192, 256, 384의 경우 모두 테 스트 함

    (성능이 가장 좋은 필터 개수는 192)
  - 첫 번째 은닉층의 컨볼루션 필터는 5x5 크기로 총 k개.

    zero-padding으로 19x19를 23x23으로 표현

    (stride는 1)
  - 두 번째부터 12번째 은닉층의 컨볼루션 필터는 3x3 크기로 총 k개

    (1 zero-padding, stride는 1)
  - 13 번째 은닉층은 1x1 컨볼루션 필터 한 개로

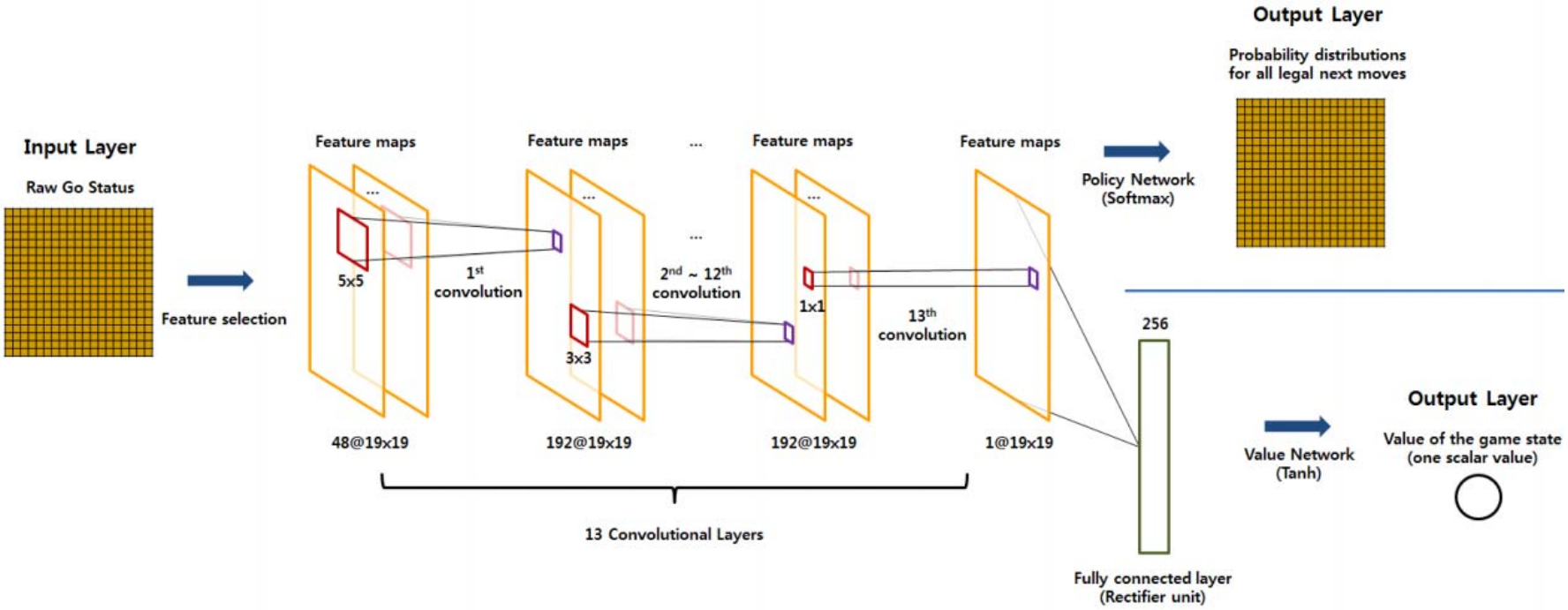
    19x19 한 개가 13번째 층의 결과 값
  - (정책네트워크 결과층)

    softmax 활성 함수를 통해 착수 가능한 지점의 확률 분 포 산출
  - (가치네트워크 결과층)

    fully-connected된 256노드의 은닉층을 지나 결과층으로 전파됨.

    마지막으로 tangent hyperbolic 활성함수를 지나 스칼라 값 산출

AlphaGo의 컨볼루션 신경망



AlphaGo의 컨볼루션 신경망 구조 (정책, 가치 네트워크)