SPRi Issue Report
2016. 3. 3. 제2016-002호 ver. 1.0

AlphaGo의 인공지능 알고리즘 분석

추형석 선임연구원†
안성원 선임연구원
김석원 책임연구원

SPRi 소프트웨어정책연구소

○ 본 보고서는 「미래창조과학부 정보통신진흥기금」을 지원받아 제작한 것으로 미래창조과학부의 공식의견과 다를 수 있습니다.

○ 본 보고서의 내용은 연구진의 개인 견해이며, 본 보고서와 관련한 의문사항 또는 수정・보완할 필요가 있는 경우에는 아래 연락처로 연락해 주시기 바랍니다.

- 소프트웨어정책연구소 SW융합연구실 추형석 선임연구원(hschu@spri.kr)

《 Executive Summary 》

구글 DeepMind가 바둑 인공지능 프로그램 AlphaGo를 공개한지 한 달의 시간이 흘렀다. 유럽 바둑챔피온인 판후이 프로 2단을 꺾고, 세계 바둑챔피온 이세돌 프로 9단에게 도전장을 냈다. 아직까지 많은 전문가들이 이세돌 9단의 우세를 점치고 있으나, 도전자인 AlphaGo가 챔피온이 되는 날이 머지않을 것으로 전망했다. 판후이와 대국 당시, 심판과 전문 프로기사들 역시 AlphaGo의 바둑이 사람과 크게 다르지 않다고 평했다. 인공지능 바둑프로그램이 인간을 넘어서는 초읽기가 시작된 것이다. 그렇다면 "어떻게 사람처럼 바둑을 두는 것이 가능한가?"에 대한 자연스러운 물음을 하게 된다. AlphaGo가 기존 바둑프로그램의 한계를 뛰어 넘은 핵심을 더 깊게 이해할 필요가 있다. 본 보고서는 AlphaGo의 인공지능 알고리즘을 더 이해하기 쉽게 전달하는 것을 목표로 한다. AlphaGo에 대한 막연한 궁금증을 풀기위해서 인공지능 게임 알고리즘의 핵심인 게임 트리 알고리즘부터 살펴보고자 한다.

바둑은 우주의 원자수보다 많은 경우의 수를 가지고 있다. 완벽한 탐색은 불가능에 가깝다. 실제 프로바둑기사들도 직관에 의해 경기를 진행한다. 컴퓨터가 인간의 직관을 표현할 수 있을까? 딥러닝이 그 대안이 될 수 있다. 딥러닝은 기존의 인공신경망을 확장한 개념으로 빅데이터 분석, 얼굴 인식, 이미지 분류 등 신산업의 곳곳에서 응용되고 있다. 딥러닝의 핵심은 사람처럼 학습하는 것이다. AlphaGo에서 역시 딥러닝을 활용하여 프로기사들의 기보 16만개를 학습했다. 이것은 사람이 1년 동안 공부하는 기보의 수를 1,000개라고 해도 평생 동안 학습할 수 없는 숫자이다. 더욱이 AlphaGo가 16만개의 기보를 학습하는데 걸린 시간은 고작 5주 밖에 되지 않는다. 그동안 쌓여온 바둑의 정수를 순식간에 학습했다는 것이다. AlphaGo는 이에 그치지 않고 스스로 대국하여 훈련하고 있다. 이것이 강화학습(reinforcement learning)이다.

착수를 결정하는 부분에는 몬테카를로 트리 탐색(Monte Calro Tree Search, MCTS) 기법이 사용됐다. MCTS는 바둑 인공지능에서 가장 널리 사용되는 알고리즘으로 무한대에 가까운 탐색의 폭과 깊이를 줄이는 역할을 한다. 탐색의 폭을 줄이는 것은 정책(policy)으로, 다음 수를 어디에 두는 것이 가장 좋은가에 대한 역할을 한다. 탐색의 깊이는 가치(value) 값으로 정해지고, 이것은 현재 대국에서 승산을 근사적으로 표현한다. 따라서 MCTS의 성능은 정책과 가치의 정확도에 따라 좌우된다. AlphaGo에서는 이 정책과 가치를 딥러닝으로 구현한 것이다. 바둑의 탐색 범위를 프로기사의 관점으로 좁힌 것이다. 그 결과 AlphaGo는 바둑 유럽챔피온에게 압도적인 승리를 거뒀다.

구글의 DeepMind가 이렇게 강력한 인공지능을 개발할 수 있었던 바탕은 풍부한 계산자원에 있다. 판후이와 대국을 한 분산 AlphaGo의 경우 1202개의 CPU와 176개의 GPU가 사용됐다. 바둑 인공지능 프로그램에 이정도 환경을 투자할 수 있는 기업은 전 세계적으로도 손에 꼽을 것이다. 또한 AlphaGo의 파급효과 역시 상당할 것으로 기대된다. AlphaGo 개발진은 AlphaGo를 활용하여 음성인식, 기후변화, 헬스케어 등에 접목하겠다고 밝혔다. 게임 인공지능 프로그램으로 시작했지만 그 활용분야는 무궁무진한 것이다. 국내에서도 인공지능 연구를 위한 환경조성이 시급하다. 정부차원에서의 대규모 계산자원 지원방안과 현실적인 문제해결을 위한 연구소 설립이 필요할 것이다.

《 목 차 》

1. AlphaGo 인공지능 바둑 프로그램의 등장

□ AlphaGo는 DeepMind 팀이 개발한 바둑 인공지능 프로그램

○ 바둑은 아직 인공지능 프로그램이 정복하지 못한 “그랜드 챌린지” 1)

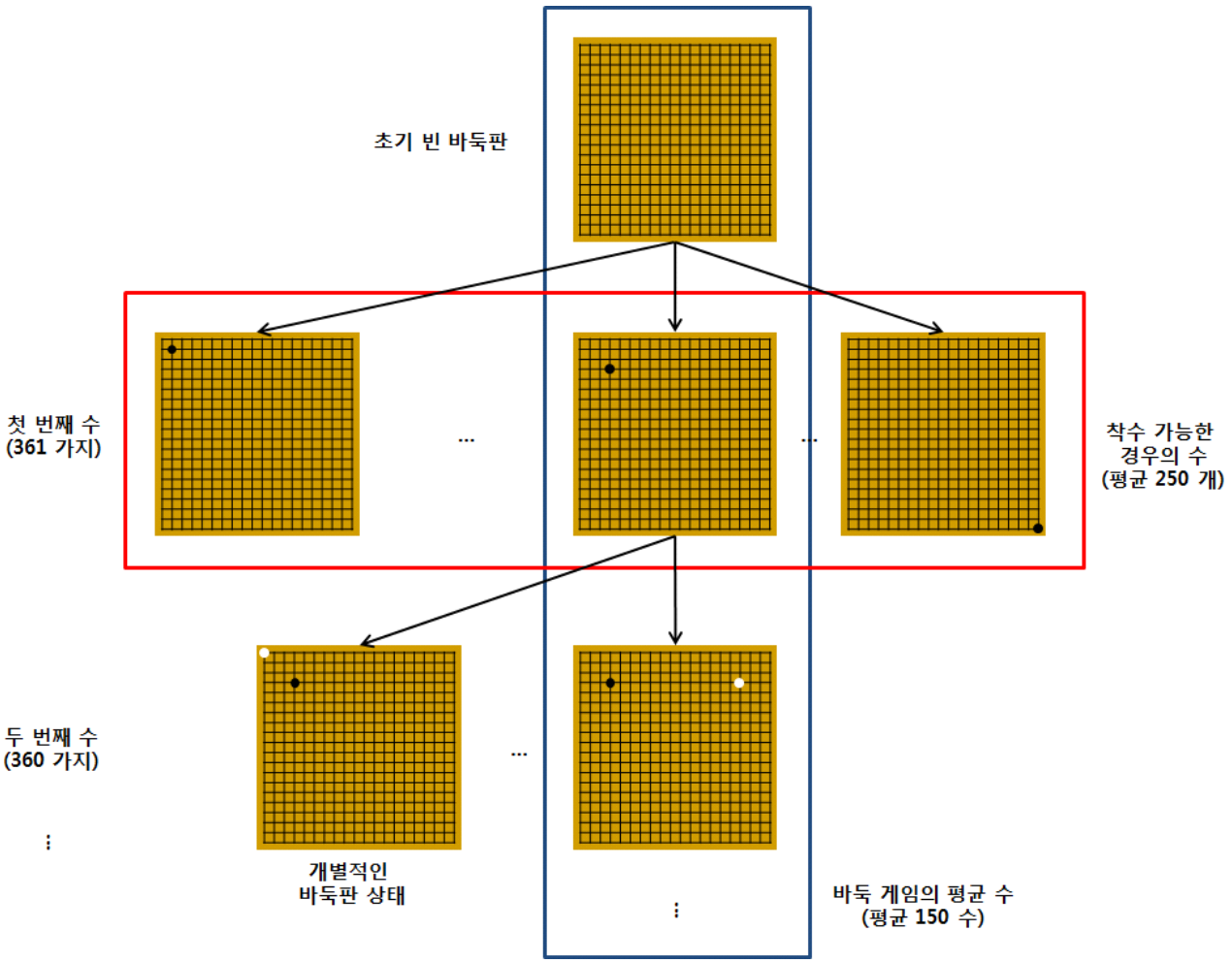
- 기존 인공지능 바둑 프로그램은 아마추어 수준이 한계 (아마추어 6단)
- AlphaGo는 프로 바둑기사와 대등하게 대국하여 승리한 최초의 바둑 인공지능 프로그램
  * 프로기사와 대국한 기존의 바둑 프로그램은 4점 접바둑에서 승리한 것이 최고의 성적
  * 2016년 3월 세계 최고의 프로 바둑기사 중 한 명인 이세돌 9단에게 도전

○ 그동안 인공지능에 의해 바둑이 정복되지 않은 가장 큰 이유는 바둑의 경우의 수가 매우 많기 때문

- 바둑 게임은 검은 돌과 흰 돌이 번갈아가면서 두고, 빈 바둑판으로부터 시작하여 기권이나 더 이상 둘 곳이 없을 때까지 진행
- **(바둑판의 상태는 10^{170}가지)** 19x19 격자로 구성된 바둑판의 착수할 수 있는 지점은 361개 이고, 지점마다 3가지 상태(검은 돌, 흰 돌, 빈 칸)가 있으므로 바둑판 상태는 $3^{361} \approx 10^{170}$가지가 존재
  * 10^{170}이라는 숫자는 단순히 바둑판의 상태를 나타낸 것이므로 바둑 경기를 진행하는 수순과는 다름
- **(바둑판의 경로는 10^{360}가지)** 10^{170}가지의 바둑판 상태가 수순에 따라 중복될 수 있으므로, 수순을 고려한 경로는 약 10^{360}가지가 존재
  * 빈 바둑판에 첫 수를 둘 수 있는 경우의 수는 361 개, 두 번째 수는 360 개로 마지막 361 번째 수까지 총 361!(팩토리얼)의 경우의 대국이 존재 2)
  * 착수 금지점 등 바둑의 규칙을 고려하면, 평균적으로 다음 수를 둘 수 있는 경우의 수가 250개이고 바둑 게임의 평균 수(길이)는 약 150수 정도임 (빈 바둑판으로부터 시작하여 약 $250^{150} \approx 10^{360}$ 가지의 경우의 수, 그림 1)

1) The mystery of Go, the ancient game that computers still can’t win, WIRED. (2014. 05. 12)
2) 바둑의 규칙을 적용하지 않았을 때의 경우의 수

* 250^{150}은 우주의 원자수(약 10^{80}개)[3] 보다 월등히 많은 수로 모든 경우의 수를 계산한다는 것이 사실상 불가



[그림 1] 바둑 게임의 흐름

○ 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

□ AlphaGo와 프로 바둑기사의 대결

○ AlphaGo는 딥러닝을 통해 프로 바둑기사의 패턴을 학습하여 바둑 인공지능의 성능을 획기적으로 개선

○ 2015년 10월 유럽 바둑챔피온 판후이(Fan Hui) 프로 2단을 5:0으로 완파

3) Mass, Size, and Density of the Universe, https://people.cs.umass.edu/~immerman/stanford/universe.html

AlphaGo 관련 인터뷰[4)5)]

- (판 후이, 유럽 바둑챔피온) “만약 AlphaGo를 컴퓨터 프로그램이라고 알려주지 않았다면, AlphaGo는 약간 이상하지만 매우 실력있는 바둑기사라고 생각했을 것이다. 게다가 AlphaGo는 지치지 않기 때문에, 공격적인 수를 두어도 덤덤하게 반응하는 마치 벽을 연상케 했다.”
- (토비 매닝, 심판) “누가 컴퓨터이고 사람인지 구분할 수 없을 정도로 AlphaGo는 사람과 유사하다. AlphaGo가 사람과 다른 점은 공격적인 수를 잘 두지 않는 점뿐이다. ”
- (박승철, 한국기원 프로 7단) “판후이와의 대결을 복기해본 결과, 전반적으로 판후이 2단의 기량이 AlphaGo에 미치지 못한다고 판단한다. AlphaGo는 대국 전체를 읽는 것에는 능통하나, 상대의 수를 분해하여 대세점을 찾는 능력은 부족하다.”

- 공식경기(formal) 5회, 비공식 경기(informal) 5회 대결. 비공식 경기에서는 3:2로 AlphaGo 승리, 중국식 규칙을 적용 (덤[6)] 7.5집)

경기 형태	제한시간	비 고[7)]
공식경기(formal)	1시간, 초읽기 30초 3회	장고바둑
비공식 경기(informal)	초읽기 30초 3회	속기바둑

<표 1> 공식경기와 비공식경기

○ 2016년 3월 9일부터 15일 까지 바둑 세계 챔피온인 우리나라 이세돌 9단과 대결 예정

- 전문가들은 이세돌 9단의 승리를 예측했으나, 지속적인 학습이 가능한 AlphaGo의 능력이 결국 바둑을 정복할 수도 있을 것이라 전망함[8)]

4) Go players react to computer defeat, Nature News. (2016. 01. 27)
5) 구글 ‘알파고’의 바둑실력, 기보로 가늠해보니, BLOTER (2016.02.17.)
6) 바둑을 처음 두는 순서는 검은 돌 → 흰 돌 순인데, 일반적으로 먼저 둘 경우가 유리하므로 이를 보상하는 차원에서 흰돌에 추가점을 주는 것이 덤
7) 한국바둑리그의 대국규정, Wikipedia
8) 국내교수 10명 긴급설문 “이번엔 이세돌이 알파고 이기겠지만…”, 한국경제 (2016. 02. 10)

□ AlphaGo의 구조와 성능

○ AlphaGo와 분산(distributed) AlphaGo

- AlphaGo는 단일 컴퓨터버전(single)과 분산 컴퓨터버전(distributed)으로 구현되었고 비용대비 최대의 성능을 나타낸 환경은 다음 <표 2>와 같음

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

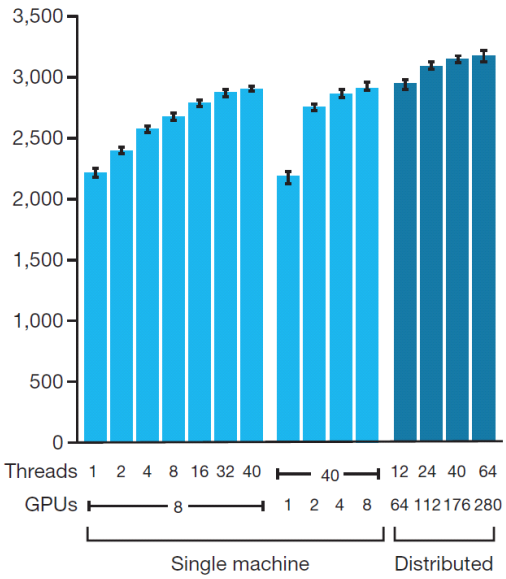
탐색 쓰레드	쓰레드 개수만큼 바둑의 경기 경로를 탐색
CPU의 역할	CPU 한 개당 1초에 1000회 이상의 시뮬레이션 수행 ※ 이 기법은 fast rollout으로 자세한 내용은 후술
GPU의 역할	딥러닝을 사용하여 바둑판 상태의 승률과 다음 착수 예측

<표 2> AlphaGo의 컴퓨팅 환경과 역할

- 분산 AlphaGo 환경은 다수의 컴퓨터를 활용하여 제한된 시간 안에 최대한의 경로를 탐색하기 위해 개발

* 유럽 바둑챔피온 판후이와의 대결에서는 분산 AlphaGo가 사용

- AlphaGo의 확장성(scalability)은 다음 [그림 2]와 같음



- 단일/분산 AlphaGo가 서로 다른 HW 구성으로 2초의 착수 제한시간을 두고 서로 대결하여 산출한 결과
- 탐색 쓰레드가 많을수록, 계산자원이 많을수록 Elo rating이 높아지나 선형적인 비례관계는 아님
- Elo Rating 바의 상단에 표현된 검정색 세로선은 바둑 실력의 편차를 나타냄
- 그림의 편차를 고려해보면 탐색 쓰레드나 GPU를 많이 사용한다고 해서 반드시 큰 폭의 성능향상을 보장하는 것은 아님

[그림 2] AlphaGo의 컴퓨팅 환경과 확장성

9) Elo rating은 게임 플레이어 간의 실력을 상대적으로 나타낸 것으로, 단일/분산 AlphaGo의 Elo rating을 측정하기 위하여 기존 인공지능 프로그램과 대결하여 산출

AlphaGo의 확장성

- [그림 2]에서 확인할 수 있듯이 계산자원의 수와 탐색 쓰레드가 증가할수록 성능이 점진적으로 개선되나, 성능향상의 폭은 점차 감소하므로 현재 AlphaGo의 바둑 실력은 한계가 존재
- AlphaGo의 성능은 더 많은 바둑 데이터와 더 강력한 계산 자원을 확보함으로써 개선될 가능성이 있으므로, 바둑 프로 9단의 실력을 따라잡는 것은 현실적으로 타당하다고 볼 수 있음

○ AlphaGo의 성능은 KGS 기준 프로 2~5단 수준

- AlphaGo와 분산 AlphaGo의 성능을 추정하기 위해서 5개의 기존 바둑 인공지능 프로그램과 495 경기 수행
- 경기결과 AlphaGo가 총 495게임 중 494번 승리 (승률 99.8%, <표 3> 참조)

* 분산 AlphaGo의 승률은 100%

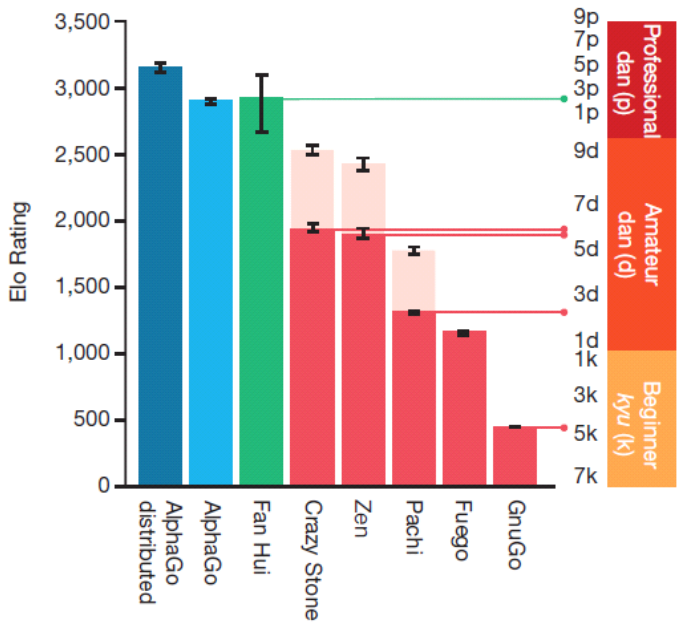
* 돌을 놓는데 걸리는 계산시간(착수시간)은 최대 5초로 설정

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

<표 3> 토너먼트 결과

- 기존의 바둑게임 프로그램과의 성능 비교. [그림 3] 참조

10) 프로(p), 아마추어 단(d), 아마추어 급(k)



- 유럽 바둑챔피온 판후이의 Elo rating 값은 실제 프로 바둑기사들의 지표를 사용[11)]

- 판후이를 제외한 나머지 인공지능 프로그램은 상호 대결을 통해 Elo rating을 계산

- Crazy Stone, Zen, Fuego의 rating의 경우 연한 색으로 표현된 부분은 4점 접바둑 기준으로 산출
  ※ 4점 접바둑 기준 AlphaGo의 승률은 77%(Crazy Stone), 86%(Zen), 99%(Pachi)

[그림 3] AlphaGo 와 상용 바둑 프로그램 성능 비교

기존 인공지능 바둑 프로그램 비교

명칭	개발자	출시 년도	최신 버전	전적	수준
Crazy Stone	Coulom R (프랑스)	2005	2015	2007, 2008 UEC컵 우승, 2014년 제2회 전성전에서 요다 노리모토 9단에게 4점 접바둑 승	6d
Zen	요지 오지마 (일본)	2009	5	2009, 2011 컴퓨터 올림피아드 우승, 2012년 다케미야 마사키 9단 에게 4점 접바둑 승	6d
Pachi	Baudiš P (체코)	2012	11	2011년 인간vs.컴퓨터 바둑대결(파리)에서 주준훈(중국) 9단 7점 접바둑 승	2d
Fuego	Muller (캐나다)	2010	svn 1989	2009년 프로그램 최초로 9x9 바둑에서 주준훈(중국) 9단 프로를 이김	1d
MoGo	Sylvain G (프랑스)	2006	TW	2008년 US바둑콩그레스 이벤트 김명완 8단에 9점 접바둑 승	2k
Gnu Go	GNU-FSF	1989	3.8		5k

11) Go Ratings(http://www.goratings.org) : 판후이의 rating은 AlphaGo와 대국 시점

2. 게임 트리 탐색 알고리즘

□ 게임 트리 탐색은 인공지능 게임 프로그램을 구현하기 위해 사용

○ 체스나 장기, tic-tac-toe 와 같이 두 플레이어가 번갈아가며 수를 두는 게임은 일반적으로 트리 형태로 표현

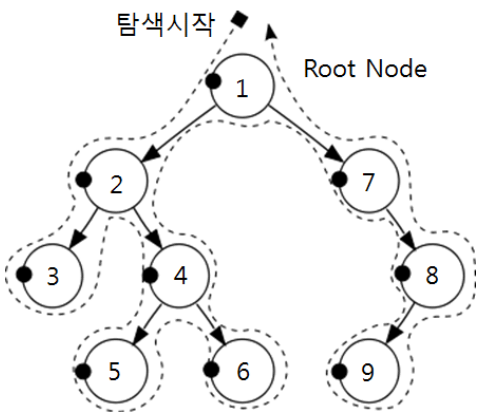
○ 바둑에서의 게임 트리는 [그림 1]과 같이 표현되고, 두 플레이어가 번갈아 가면서 수를 두면 트리가 확장

- 특정 게임 상태에서 다음 수를 예측하기 위해서는 수읽기를 통해 가장 승리할 확률이 높은 곳을 결정
- 이 과정이 트리의 탐색이고, 이론적으로는 현재 상태에서 가능한 모든 결과를 미리 알 경우 가장 승률이 높은 수를 선택

* 세계 체스챔피온을 꺾은 IBM의 딥블루는 가능한 모든 결과를 탐색한 예

○ 트리탐색기법(트리순회 : Tree traversal)은 트리 구조에서 각각의 노드를 한번씩, 체계적인 방법으로 방문하는 과정

- 트리탐색 기법에는 탐색 순서에 따라 전·중·후위 및 레벨 순서 순회기법이 있음
- 이중 전위순회(preorder)는 깊이우선의 탐색(depth-first search: DFS) 이라고도 하며, 이후 설명할 MinMax 알고리즘의 탐색방법
- 트리의 깊이우선 탐색 과정 [그림 4]



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

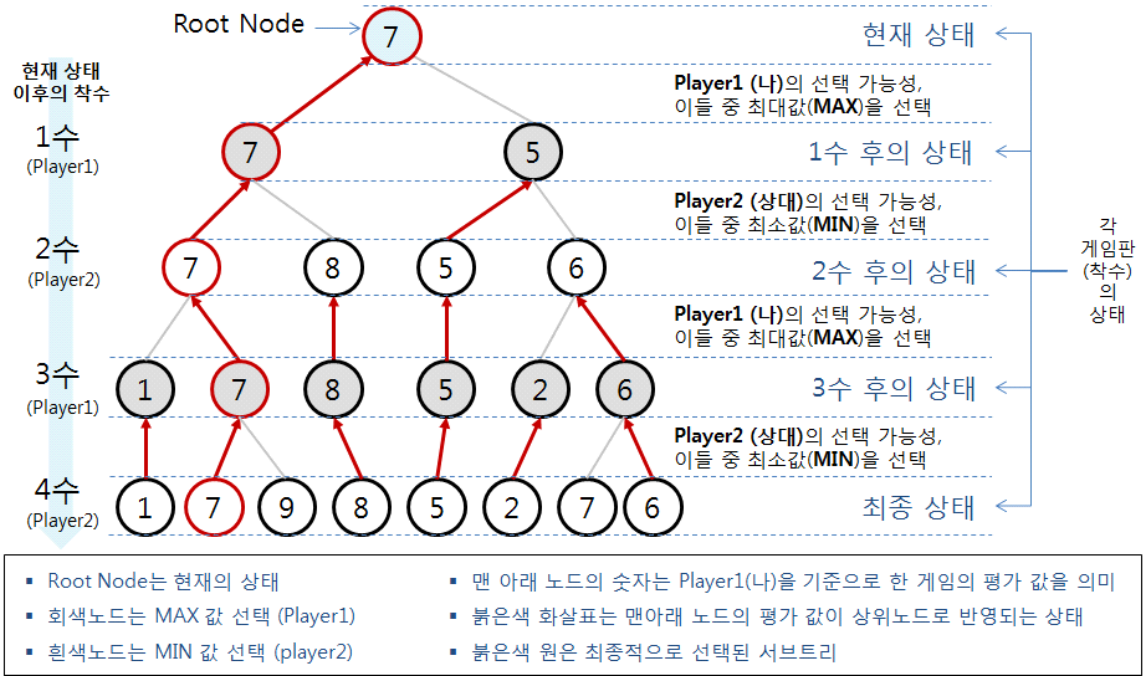
<탐색 노드의 순서>
1→2→3→4→5→6→7→8→9

[그림 4] 깊이우선 탐색(전위순회)의 과정

□ 최소-최대(Minimax, MM) 알고리즘 : 게임에서 한 플레이어에게 유리한 수는 상대 플레이어에게는 불리한 수

○ 상대의 승산을 최소화하고 자신의 승산을 최대화 하는 것이 게임에서 승리하는 방법이기 때문에, 이 경로를 찾는 것이 인공지능 게임 프로그램의 핵심

○ [그림 5]는 깊이가 4수로 제한된 게임의 MinMax 알고리즘의 탐색 결과 예시



[그림 5] MinMax 알고리즘의 탐색 결과

- [그림 5]의 MinMax 알고리즘의 트리 탐색 과정은 [그림 4]와 동일한 깊이 우선 탐색을 수행

- 각 서브트리의 탐색이 끝나면 기존 탐색된 노드들을 역으로 거슬러 올라가면서 상위노드로 평가 값을 반영

* 이때, 최대값과 최소값을 교대로 비교하여 최종 서브트리를 선택

최소-최대(MinMax) 알고리즘의 세부과정

- MinMax 알고리즘은 예상되는 최대의 손실(maximum loss)를 최소화시키기(minimize) 위해 사용
  - 깊이(depth)가 제한[12]된 깊이우선의 탐색 과정 : 트리의 깊이는 내다보는 수의 정도
- [그림 5]의 예시에서 평가 값의 반영 과정 (Player1 = 나, Player2 = 상대)
  - 게임의 순서는 현재 상태에서 1수(나) → 2수(상대) → 3수(나) → 4수(상대)
  - [그림 5]에서 4수를 둔 최종 상태의 노드들의 숫자는 Player1(나)를 기준으로 한 게임의 평가 값으로 숫자가 높을수록 승률이 높음
  - 상대의 기준으로는 평가 값이 낮을수록 나(Player1)에게 불리하고, 이는 곧 상대에게는 유리한 수
  - 평가 값이 자식노드에서 상위노드로 전파될 때 마다, 해당 상위노드의 자식 노드들 간의 비교
    - ※ **나(Player1)의 수**에서는 형제 노드들과 비교하여 **가장 큰 값(Max)**을 선택
    - ※ **상대(Player2)의 수**에서는 형제 노드들과 비교하여 **가장 작은 값(Min)**을 선택 (상대는 상대자신에게 가장 유리한 수를 선택할 것이기 때문, 이는 곧 나에게 가장 불리한 평가값을 갖는 작은 수)
- 최종적으로 선택된 서브트리 ([그림 5]의 붉은색 원형테두리 노드들)는 나(Player1)와 상대(Player2)가 모두 최선의 선택을 한 결과물
  - 계산 이후 착수에서 player1은 왼쪽 노드를 첫 번째 수로 선택
- MinMax 알고리즘의 트리탐색은 모든 노드에 걸쳐 이루어지므로, 내다보는 수(트리의 깊이)가 많아질수록 계산시간은 늘어나게 됨

□ 게임 트리 탐색의 효율성이 인공지능 프로그램의 성능을 좌우

○ 최소-최대 알고리즘의 경우 트리의 모든 노드를 탐색해야하기 때문에, 게임의 복잡도가 클 경우 트리를 전부 확장하여 탐색하는 것이 현실적으로 불가능

○ 알파-베타 가지치기 알고리즘은 최소-최대 알고리즘에서 더 이상 계산할 필요가 없는 경로를 제거하는 기법으로 계산량을 대폭 절감

12) 탐색 과정이 시작노드로부터 자식노드로 한없이 깊이 진행되는 것을 막기 위한 깊이 제한(depth bound)

- 상대가 현재 위치 이후의 어떤 지점에서 더 유리한 결과를 얻을 경우, 현재위치를 만드는 수는 나에게 무조건 불리

  * 나에게 불리한 수가 만들어지는 지점부터 트리의 가지를 침(Pluning)

- 가지치기 판단을 위해서는 게임트리 검색의 매 순간마다 Player1(나)에게 유리한 현재 최선의 값(Alpha)과 Player2(상대)에게 유리한 현재 최선의 값(Beta)을 기억

  * Alpha 는 maximizing player(Player1)가 보장하는 최소의 값(초기값은 -∞) (나에게 유리한 상황의 최소값)

  * Beta 는 minimizing player(Player2)가 보장하는 최대의 값(초기값은 +∞) (나에게 불리한 상황의 최대값)

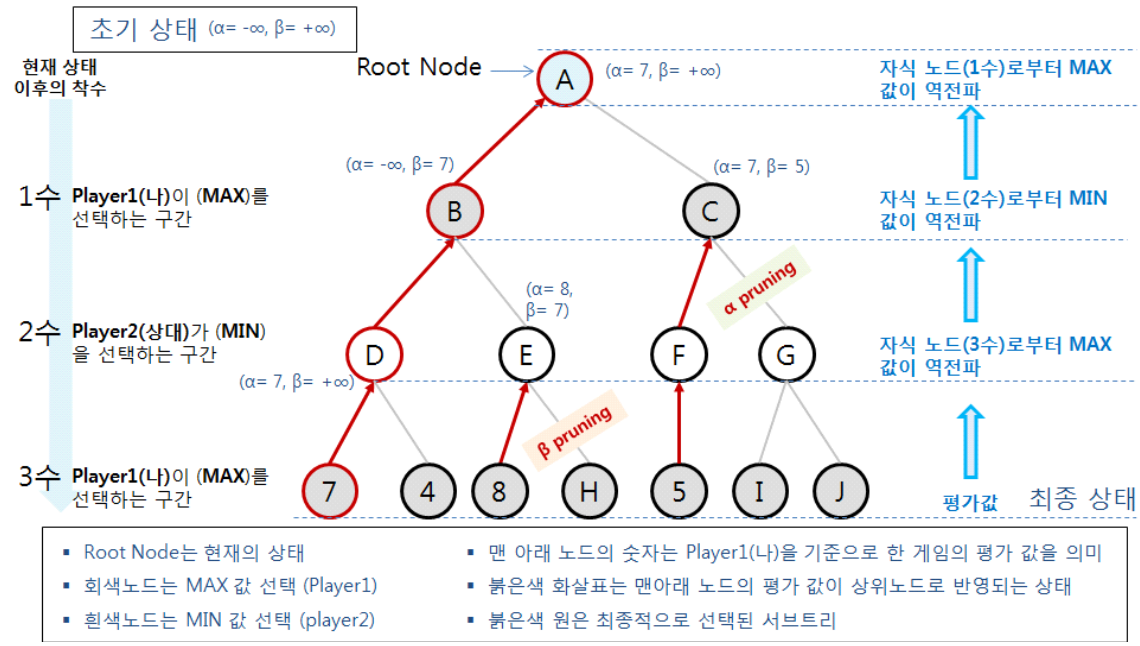
- 가지치기는 Alpha cut-off(α Pruning) 와 Beta cut-off(β Pruning) 가 있음

  * Alpha cut-off는 어떤 경우가 Player1(나)이 Player2(상대)보다 불리하여 그 경우를 선택하지 않을 때 수행

  * Beta cut-off는 어떤 경우가 Player1(나)이 Player2(상대)보다 유리하여 상대가 그 경우를 선택하지 않을 때 수행

알파-베타 가지치기(Alpha-Beta Pruning) 세부과정

- 알파-베타 가지치기 알고리즘의 탐색 과정[그림 6]



[그림 6] Alpha-Beta Pruning 알고리즘의 탐색 결과

• Alpha Beta 가지치기의 세부 과정

- 기본적인 탐색 방식은 MinMax 와 동일한 깊이 우선 탐색
- 탐색이 진행되는 동안 [그림 4]와 같은 순서대로 상위노드의 α β상태값이 자식노드로 상속

※ 노드 A에서 (α= -∞, β= +∞) 의 값을 자식노드에 상속해가며 탐색

- 서브트리의 탐색이 종료되어 상위노드로 값을 전파 할 때 변경된 α β값을 반영

※ 예를들어, 3수 구간에서 최종 평가값인 7이 상위노드인 D로 전파되고, 형제노드의 값 4와 비교하여 최종적으로 7이 최대값으로 선택

※ 이때의 D의 α값은 7로 갱신되고, 다시 B로 전파될 때는 최소값을 전파해야 하므로 B의 β값이 7로 갱신

※ B에서의 α값은 E의 값을 참조해야 알 수 있으므로 탐색전까지는 다시 -∞

※ **Alpha ≥ Beta** 가 되는 지점이 있다면 그 지점 이하의 서브트리 즉, 가지들은 탐색하지 않음 → **가지치기 수행**

- 현재 노드 B에 임시 저장된 D의 리턴값 7은 Min구간(2수)에서 최소값을 선택해서 올라온 값이므로, 또 다른 자식노드 E의 값이 7보다 작다면 갱신될 것이고, 그렇지 않다면 처음값을 유지

※ E의 왼쪽 자식노드로부터 리턴받은 값은 8인데, 이 값은 최대값으로 판단되어 전파된 값
① E의 오른쪽 자식노드 H의 값이 8보다 작다면 E의 값은 8이되고,
② H의 값이 8보다 크다면 그 값이 E의 값으로 갱신

※ ①과② 어느쪽의 값이 리턴 되더라도 E는 D와의 경합(B로 min값 전파)에서 선택되어질 수 없음 ☞ **H노드는 방문할 필요가 없음 = β Pruning**
(E의 α=8, β=7 이므로, $\alpha \geq \beta$)

- 이제 루트노드인 A의 왼쪽 자식노드 B에는 값7이 결정되어 있고, 다음으로 오른쪽 자식노드인 C를 탐색

※ C가 B와의 경합(A로 Max 값 전파)에서 선택받기 위해서는 반드시 7보다 큰 값을 가져야 함

※ C는 F와 G중의 최소값을 전파받게 되는데, 첫 번째 자식노드인 F쪽 탐색을 통해 5의 값을 전파 받음

※ C가 G의 값을 전파받기 위해서는 G의 값이 반드시 5보다 작아야 하고, 이는 C가 5보다 작은 값을 갖기에 B와의 경합에서 선택받을 수 없음

※ 그러므로 C의 자식노드인 G가 어떤 값을 갖더라도, C는 절대로 7보다 큰 값을 가질 수 없으며, 선택되지 않음 ☞ **G노드 이하는 방문할 필요가 없음 = α Pruning** (G의 α=7, β=5 이므로, $\alpha \geq \beta$)

○ 바둑은 앞서 기술한바와 같이 게임의 복잡도가 매우 크기 때문에, 인공지능 바둑 프로그램은 막대한 계산량을 최소화하고 게임 예측성능을 높이는 것이 목표

- 바둑을 트리로 표현하면 250^{150}의 노드를 갖게 되므로, 알파-베타 가지치기는 바둑 인공지능 구현에 매우 비효율적[13)]
- 모든 노드를 탐색하는 것이 사실상 불가능하므로, 트리 탐색의 폭과 깊이를 적절하게 제한하는 방법이 필요함

* 이 방법이 AlphaGo에서 사용된 것으로 자세한 내용은 다음 장에서 다룸

13) Silver, D. et al., "Mastering the game of Go with Deep neural networks and tree search," Nature vol 529, pp. 484-489, 28 Jan 2016.

3. AlphaGo의 게임 탐색 알고리즘 - 몬테카를로 트리 탐색

□ 바둑 게임의 특성과 인공지능

○ 바둑은 상대방 바둑돌을 많이 포위할수록 승리한다는 간단한 규칙 때문에 자유도가 매우 높으나 “정석”이 존재

- 고대 중국에서부터 시작된 바둑은 상대에게 승리할 수 있는 여러 가지 방법이 축적되어 왔음

  * 바둑의 첫 수는 귀의 화점에 두는 것이 일반적[14)]

- 바둑기사처럼 바둑을 두는 인공지능 프로그램을 개발하기 위해서는 바둑기사들의 패턴을 학습하는 것이 중요한 요소

  * 예를 들어, 빈 바둑판에 둘 수 있는 경우의 수는 361가지이나 바둑기사들이 착수하는 지점은 한정되어 있음

○ 바둑기사들은 “수읽기”를 통해 경기의 진행이 유리하도록 착수를 결정

- 이 과정은 앞서 설명한 게임 트리의 탐색과 유사

  * 착수 가능한 여러 지점에서 다양한 전개과정이 도출되므로, 제한된 시간 안에 최대한 많은 수읽기를 하는 것이 필요

- 지역적인 형세판단을 통해 전체적인 게임의 승산을 판단

○ 프로 바둑기사들은 직관적인 판단을 통해 착수를 결정

- 바둑의 경우의 수가 거의 무한대에 가까운 점은 사람에게도 마찬가지이므로 경험을 바탕으로 한 착수가 가능

- 프로 바둑기사는 묘수를 개발하여 다른 기사들이 사용하지 않은 새로운 전략을 제시

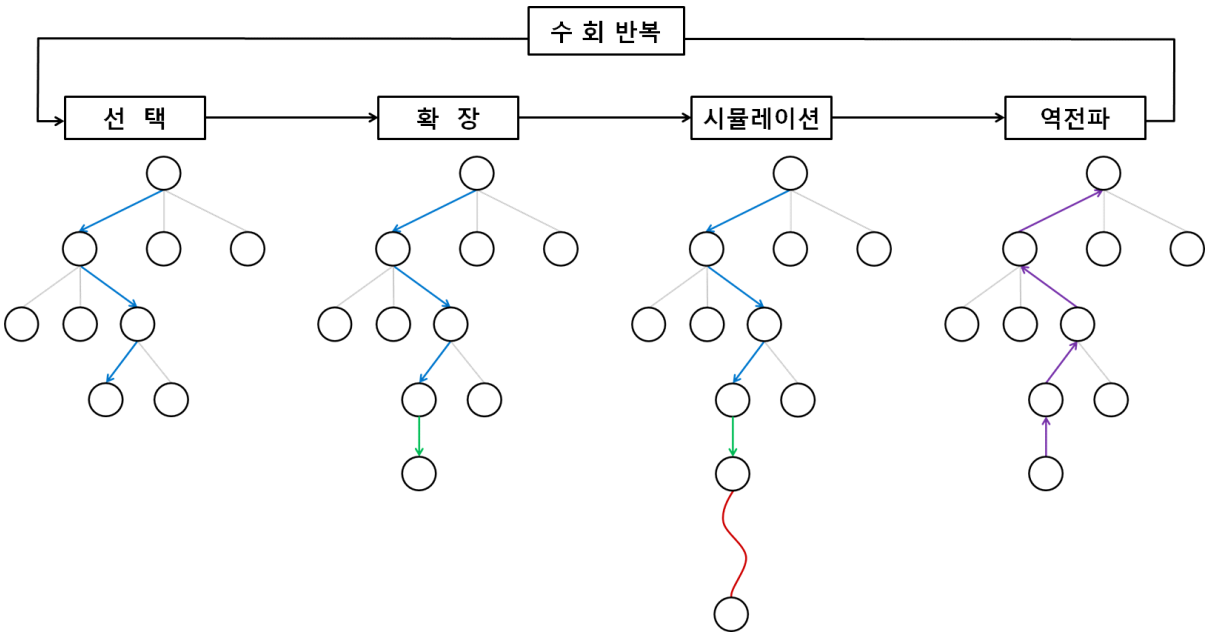
○ 바둑은 초읽기[15)]라는 규칙이 있기 때문에 인공지능 프로그램은 제한된 시간 안에 프로 바둑기사와 같이 수읽기와 형세판단을 하여 최적의 수를 예측해야 함

14) 4선(바둑판의 네 번째 줄)과 4선이 만나는 지점이 귀
15) 속기바둑의 경우 정해진 횟수의 초읽기 시간 안에 착수해야 함

□ MCTS(Monte Calro Tree Search - 몬테카를로 트리 탐색) : 바둑에서 가장 널리 사용되는 인공지능 알고리즘

○ MCTS는 최소-최대 알고리즘의 성능을 개선한 것으로, 모든 경로를 탐색하는 것이 불가능 할 때 효율적

○ [그림 7]는 MCTS의 일반적인 구조를 나타내고, 이를 바둑 인공지능 관점에서 해석해보면 <표 4>와 같음[16)]



[그림 7] MCTS 알고리즘

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.[17)] 속도가 빠르기 때문에 여러 번 수행할 수 있으나 착수의 적정성은 떨어짐[18)]
④역전파 : ③의 결과를 종합하여 확장한 노드의 가치(②에서 한 단계 더 착수한 것의 승산)를 역전파하여 해당경로의 승산 가능성을 갱신

<표 4> MCTS의 4단계 과정

16) Chaslot, Guillaume, et al. “Monte-Carlo Tree Search: A New Framework for Game AI.“ AIIDE. 2008

○ MCTS는 게임 트리 탐색에서 두 가지 핵심요소가 있음. <표 5>

정책	**트리의 폭을 제한하는 역할** ※ MCTS의 두 번째 단계인 확장에서 주로 사용되는 것으로, 특정 시점에서 가능한 모든 수 중 가장 승률이 높은 것을 예측
가치	**트리의 깊이를 제한하는 역할** ※ 가치는 현재 대국상황의 승산을 나타낸 것으로, 승산이 정확할수록 많은 수(더 깊은 노드)를 볼 필요가 없음

<표 5> MCTS의 핵심요소

- 바둑에서 **정책**은 전문가의 전략이 될 수도 있고, 과거 기보 데이터에서 많은 프로 기사들이 선호한 패턴이 될 수도 있음 (선호되는 방향을 정의)
- **가치**는 바둑 국면의 형세판단을 근사하여 수치화한 것으로 볼 수 있음. 정확한 값은 모든 경우의 수를 계산하여 게임 종료 시점까지 가봐야 하기 때문에 계산이 어려움

○ MCTS에서 정책은 정확한 가치를 추정하는데 중요한 역할을 함. 정책은 게임의 미래 상태를 예측하기 때문에 게임의 특성과 경험을 반영

- 만약 정책이 최적화(optimize)하는 방향으로 수렴(converge)한다면, 가치 역시 최적의 값으로 수렴한다고 볼 수 있음 (i.e. 다음 착수가 가장 최적화된 방향이라면 경기에 승리할 확률이 높아짐)

□ AlphaGo의 MCTS는 더 효율적인 정책과 가치를 구현하여 성능을 프로수준으로 끌어올림

○ 프로기사의 기보를 바탕으로 정책과 가치를 제안

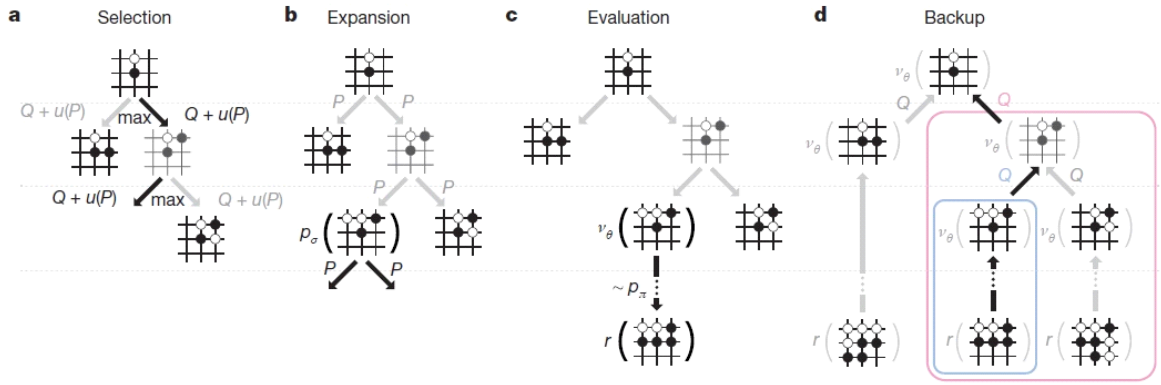
- 전문 바둑기사의 관점으로 탐색의 폭과 깊이를 결정하기 때문에, AlphaGo의 기보를 분석한 많은 전문가들은 사람과 매우 유사한 수를 둔다고 평함

○ 또한, 스스로 대국하는 학습기법을 통해 정책과 가치의 성능을 향상시킴

17) 이 부분이 몬테카를로 시뮬레이션으로 이것은 다수의 랜덤 시뮬레이션으로부터 목표 값을 근사하는 기법. 예를 들어, 중심이 (0,0)이고 반지름이 1인 원의 넓이를 구하고자 한다면, -1에서 1사이 좌표를 랜덤하게 생성하여 그 좌표가 원 안에 들어간 수를 통해 원의 넓이를 근사할 수 있음

18) AlphaGo에서는 무작위 방법 대신 fast rollout policy를 학습하여 적용함

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

19) Silver, D. et al., "Mastering the game of Go with Deep neural networks and tree search," Nature vol 529, pp. 484-489, 28 Jan 2016.

20) fast rollout 기법은 빠르게 다음 수를 예측하여 게임 종료시점까지 시뮬레이션 하는 정책으로, 딥러닝 기반의 정책보다 정확도는 떨어지나 계산에서 약 150배정도 빠름. 롤아웃은 전체적인 바둑상태가 아닌 3x3의 국소적인 위치로부터 다음 수를 예측해 나감

4. AlphaGo의 차별성 - 딥러닝

□ AlphaGo의 딥러닝

○ 딥러닝을 활용하여 전문 바둑기사들의 패턴을 학습함

- 바둑 기보를 19x19 픽셀을 갖는 이미지로 입력받아 전문 바둑기사의 다음 착수를 학습하는 과정

  * 이 과정은 바둑 입문자가 기보를 공부하여 바둑기사들의 패턴을 습득하는 것과 유사함

  * AlphaGo 개발자인 데이비드 실버는 “AlphaGo는 16만개의 기보를 5주만에 학습했다” 라고 밝힘[21)]

- AlphaGo는 딥러닝 기법 중 특히 이미지 처리에 강한 컨볼루션신경망을 기반으로 학습하기 때문에 국지적인 패턴인식에도 강점을 가짐

  * 바둑에서 지역적인 대국이 전체적인 형세판단에 매우 중요한 역할을 함

○ 바둑기사의 착수를 학습한 것은 **정책네트워크**이고 국지적인 패턴인식을 통한 승산판단은 **가치네트워크**로 구현

- 정책과 가치네트워크는 MCTS에서 게임 트리를 탐색할 때 적용됨

□ AlphaGo의 정책과 가치네트워크

○ **정책 네트워크(Policy Network, p) : 정책 계산을 위한 딥러닝 신경망**

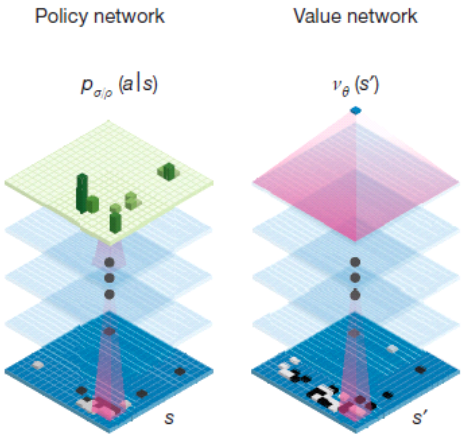
- 정책 네트워크에서 사용된 딥러닝 기법은 컨볼루션 신경망(Convolution Neural Network, CNN)으로 19x19 바둑판 상태를 입력하여 바둑판 모든 자리의 다음 수 선택 가능성 확률 분포를 출력 ([그림 9] 참조)[22)]

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

21) 프로기사가 1년에 1000번 정도 대국한다면, 사람이 1000년에 걸쳐서 학습해야할 데이터를 AlphaGo는 단 5주만에 학습한 것

22) Silver, D. et al., “Mastering the game of Go with Deep neural networks and tree search,” Nature vol 529, pp. 484-489, 28 Jan 2016.

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



[그림 9] 정책 네트워크와 가치 네트워크의 구성

○ 정책 네트워크 학습

- **지도학습(supervised learning, p_σ) : 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, p_ρ) : 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[23]. 약 128번의 자체대결을 수행

  * 이로부터 도출된 경기결과(reward)를 바탕으로 이기는 방향으로 가도록 네트워크의 가중치를 강화(개선). 강화학습 후의 정책 네트워크로도 기존 바둑 프로그램인 Pachi와 대결하여 85%의 승률[24]

23) 처음에는 지도학습의 결과를 그대로 이용하여 경기를 진행하지만 학습이 진행되면서 여러 버전의 네트워크가 생성되며 이들 간의 강화학습을 통해 실제로 승리하는 빈도가 높은 쪽으로 가중치가 학습됨

* 50개의 GPU를 사용하여 학습 (기간 : 1일)

○ **가치 네트워크(Value Network, v_θ) : 바둑의 전체적인 형세를 파악**

- AlphaGo에서는 가치(value)를 계산하기 위해 딥러닝을 이용한 가치 네트워크(value network) 사용

  * 기존 프로그램의 가치함수는 비교적 간단한 선형결합으로 표현했기 때문에 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[25)]

  * 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)

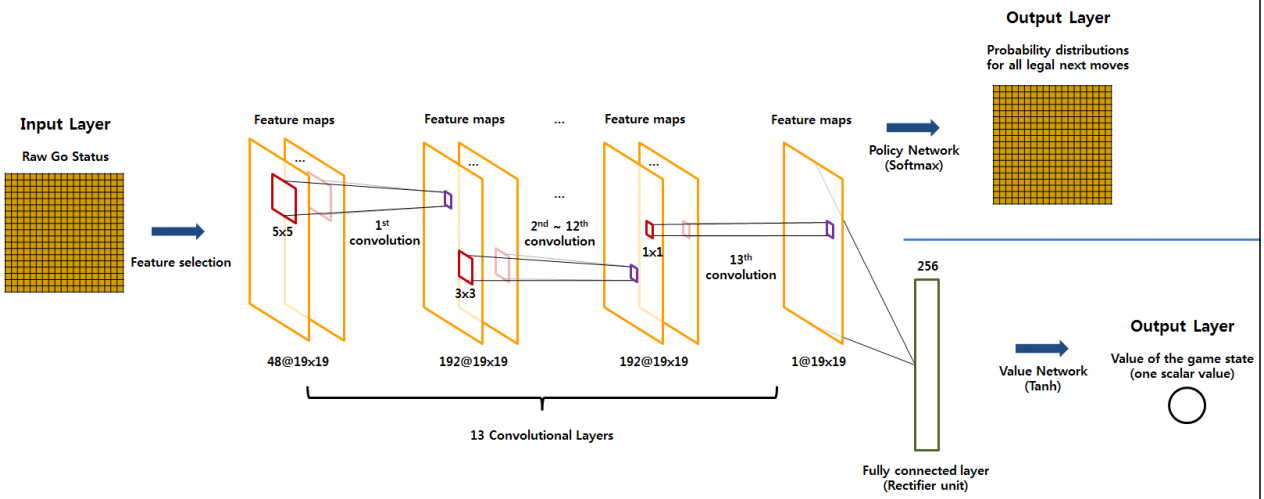
AlphaGo의 컨볼루션신경망

- 컨볼루션신경망 개요
  - 컨볼루션신경망은 이미지나 비디오에서 객체의 분류에 특화된 방법
  - 이미지의 객체분류는 전통적인 인공신경망인 다층퍼셉트론으로도 충분히 가능했으나, 노드간 링크가 모두 연결되어있는 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
- AlphaGo에서 사용된 컨볼루션신경망 구조

24) MCTS 탐색 알고리즘을 적용하지 않고 단순히 현재 상황에서 다음 착수를 예측하는 정책 네트워크만으로 Pachi와 대결함

25) 실제 기보로 학습할 경우 바둑 기사들이 두는 수가 매우 비선형적이기 때문에 과적합 문제가 발생

- 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)
- 컨볼루션 층의 상세구조
  - 컨볼루션 필터는 k개로 AlphaGo에서는 k=128, 192, 256, 384의 경우 모두 테스트 함 (성능이 가장 좋은 필터 개수는 192)
  - 첫 번째 은닉층의 컨볼루션 필터는 5x5 크기로 총 k개. zero-padding으로 19x19를 23x23으로 표현 (stride는 1) [그림 10] 참조
  - 두 번째부터 12번째 은닉층의 컨볼루션 필터는 3x3 크기로 총 k개 (1 zero-padding, stride는 1)
  - 13번째 은닉층은 1x1 컨볼루션 필터 한 개로 19x19 한 개가 13번째 층의 결과 값
  - (정책네트워크 결과층) softmax 활성함수를 통해 착수가능한 지점의 확률 분포 산출
  - (가치네트워크 결과층) fully-connected된 256노드의 은닉층을 지나 결과층으로 전파됨. 마지막으로 tangent hyperbolic 활성함수를 지나 스칼라 값 산출



[그림 10] AlphaGo의 컨볼루션 신경망 구조 (정책, 가치 네트워크)

□ AlphaGo의 계산성능 분석

○ AlphaGo는 딥러닝으로 구현된 정책과 가치네트워크를 활용하여 MCTS 탐색 기법을 통해 착수를 결정

- 분산 AlphaGo를 기준으로 착수(decision)는 초읽기 30초 동안 약 10만 번의 수읽기를 통해 결정

* 체스 인공지능 프로그램인 딥블루의 경우 약 2억 번의 수읽기

- 바둑계의 전설적인 인물인 이창호 9단의 경우 100수 정도의 수읽기[26)]

○ AlphaGo의 MCTS 탐색에서 가장 시간이 많이 소요되는 부분은 가치, 정책 네트워크의 값을 산출하는 과정

- 13층의 컨볼루션 신경망의 값을 산출해내기 위해서는 약 300억 번의 연산수(30 GFLOP)가 필요함

- 이러한 이유로 AlphaGo의 탐색 쓰레드는 비동기식(asynchronous)으로 진행됨

○ AlphaGo의 대국을 분석해 보면 시간이 많이 주어질수록 강한 면모를 보임

- 판후이 2단과의 대결에서 초읽기 30초만 주어진 비공식(informal)경기는 AlphaGo가 3:2로 우세하였으나, 공식(formal)대결에서는 5:0으로 완승

- 이세돌 9단과의 대결은 2시간 제한시간에 60초 초읽기 3회가 주어지므로, 판후이와의 대결보다는 더 나은 기력을 보일 수 있음[27)]

* 대국규정은 구글 DeepMind와 한국기원이 협약한 사항

- AlphaGo의 시간분배는 주로 중반전에 이루어짐[28)]

26) “바둑은 9단, 연애는 9급”, 주간동아 (2000.01.06.)
27) ‘이세돌-알파고’ 세기의 대결바익에 궁금한 것 11가지, 조선비즈 (2016.2.23.)
28) Silver, D. et al., “Mastering the game of Go with Deep neural networks and tree search,” Nature vol 529, pp. 484-489, 28 Jan 2016.

5. 결론 및 시사점

□ 구글이 AlphaGo라는 바둑프로그램을 통해 인공지능 딥러닝 기술의 성능을 보여준 실증 사례

○ 바둑은 경우의 수가 매우 많고 복잡하여 인공지능 분야의 큰 도전과제였으나, 이번을 계기로 큰 의미와 성과를 거둠

□ AlphaGo의 성능은 Elo Rating으로 최대 3140으로 KGS기준으로 현재 프로 2단 ~ 5단 정도의 수준으로 파악

○ 기존 상용 바둑 프로그램들과의 경기에서 99.8%의 승률을 보임

□ AlphaGo의 인공지능 알고리즘은 기존 몬테카를로 트리 탐색(MCTS) 방식에 컨볼루션 신경망(CNN)을 통한 딥러닝 수행

○ 실제 바둑기사의 착수를 학습한 정책네트워크(Policy Network)와 국지적인 패턴인식으로 승산판단을 위한 가치네트워크(Value Network)로 구현

○ 프로바둑 기사들의 착수전략을 학습하는 지도학습(Supervised learning) 방식과 스스로 경기하여 학습된 전략을 강화하는 강화학습(Reinforcement learning) 방식을 적용

□ 현실적인 문제를 해결하는 연구개발을 위한 고성능 컴퓨팅 환경 조성과 공동 활용을 위한 국가차원의 지원이 필요

○ 구글은 AlphaGo를 학습 및 실행시키기 위한 대용량 계산 및 저장 컴퓨팅 자원 환경을 갖추고 있음

○ 아이디어를 가지고 있더라도 컴퓨팅 자원의 부족으로 이것을 구현할 수 있는 조직은 전 세계에 많지 않음

○ 이번 사례를 통해 고성능 컴퓨팅 환경이 기술발전에 중요한 요소임을 재확인

[참고문헌]

1. 국외문헌

Silver, D. et al., “Mastering the game of Go with Deep neural networks and tree search,” Nature vol 529, pp. 484-489, 28 Jan 2016.

Chaslot, Guillaume, et al. “Monte-Carlo Tree Search: A New Framework for Game AI.“ AIIDE. 2008

Korf, Richard E., and David W. Chickering. “Best-first minimax search: First results.“ Proceedings of the AAAI’93 Fall Symposium. 1993.

2. 기타(신문기사 등)

“Google AI algorithm masters ancient game of Go” Nature.com, 27 Jan. 2016. http://www.nature.com/news/google-ai-algorithm-masters-ancient-game-of-go-1.19234

“구글 ‘알파고’의 바둑실력, 기보로 가늠해보니”, BLOTER 2016.02.17. http://www.bloter.net/archives/249875

“이세돌 신의 한 수 vs 알파고 컴의 한 수” chosun.com, 2016.01.28., http://news.chosun.com/site/data/html_dir/2016/01/28/2016012800485.html

“바둑은 9단, 연애는 9급”, 주간동아, 2000.01.06., http://weekly.donga.com/Library/3/all/11/62340/1

Monte Carlo tree search, Wikipedia, https://en.wikipedia.org/wiki/Monte_Carlo_tree_search

Convolutional neural network, Wikipedia, https://en.wikipedia.org/wiki/Convolutional_neural_network

Reinforcement learning, Wikipedia, https://en.wikipedia.org/wiki/Reinforcement_learning

Elo rating system, Wikipedia, https://en.wikipedia.org/wiki/Elo_rating_system

주 의

1. 이 보고서는 소프트웨어정책연구소에서 수행한 연구보고서입니다.
2. 이 보고서의 내용을 발표할 때에는 반드시 소프트웨어정책연구소에서 수행한 연구결과임을 밝혀야 합니다.