AlphaGo

CS221 Section
10/27

By: Pujun Bhatnagar, Michael Chen

The Past (1994)

Checkers: Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994. Used an endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of 443,748,401,247 positions. Checkers was now solved! !

The Past (1997)

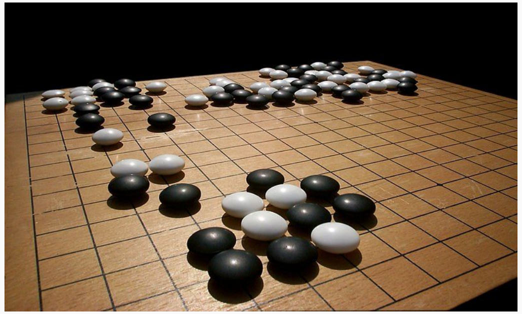
Chess: Deep Blue defeated human world champion Gary Kasparov in a six-game showdown in 1997. Deep Blue examined 200 million positions per second, used very sophisticated evaluation and undisclosed methods for extending some lines of search up to 40 ply. Current programs are even better, if less historic!

The Past

Othello: Human champions refuse to compete against computers, which are too good (lol)

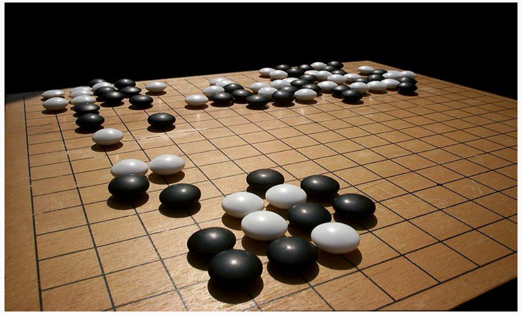
The Past (2015)

Go: Human champions are beginning to be challenged by machines, though the best humans still beat the best machines. In Go, $b > 300$, so most programs use pattern knowledge bases to suggest plausible moves, along with aggressive pruning. (2015)



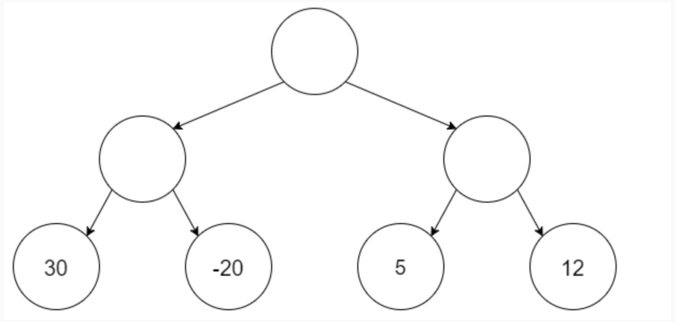
The Present (2016)

In March 2016, Google's AlphaGo defeated the world champion Lee Sedol



Game Trees (Modeling)

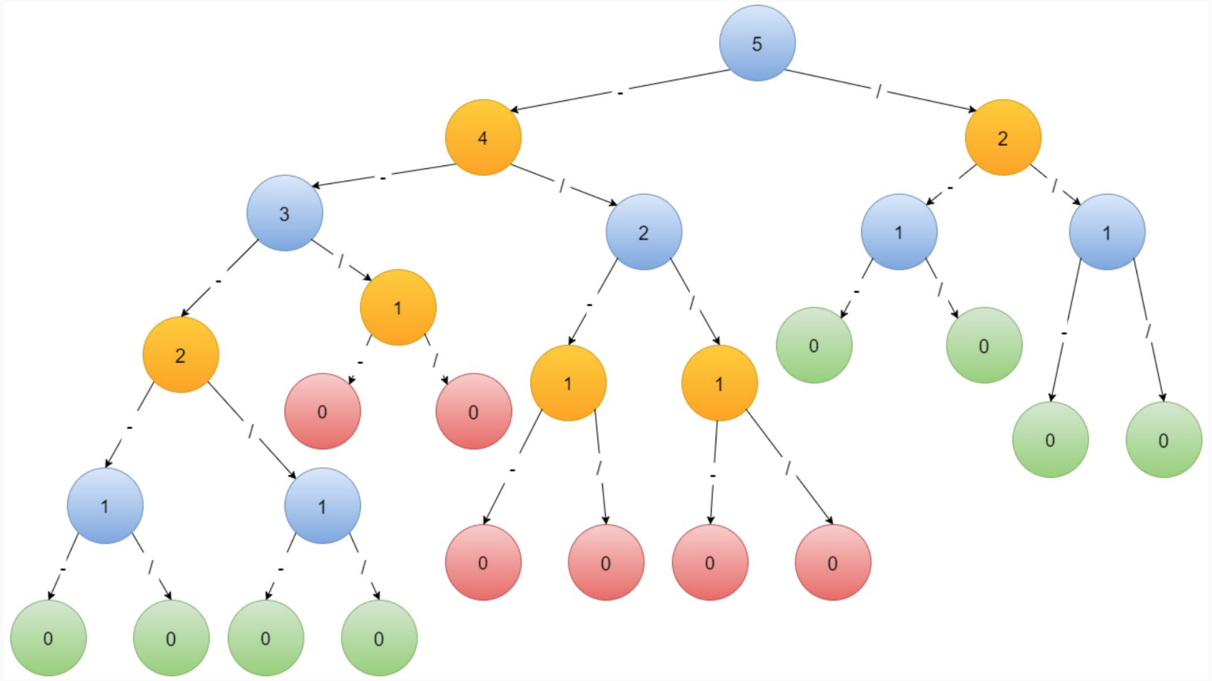
- Each node is a decision point for a player. Each root-to-leaf path is a possible outcome of the game



Example Problem

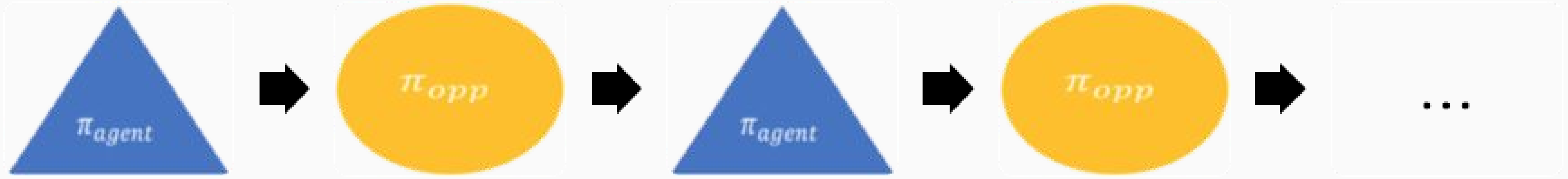
- Start with a number N
- Players take turn either decrementing N or replacing it by floor(N / 2)
- The person to reach 0 first wins!

Resulting Game Tree



Expectimax

- The agent chooses the policy that is optimal to a fixed known opp. policy



$$V_{\max,\text{opp}}(s) = \begin{cases} \text{Utility}(s) & \text{IsEnd}(s) \\ \max_{a \in \text{Actions}(s)} V_{\max,\text{opp}}(\text{Succ}(s,a)) & \text{Player}(s) = \text{agent} \\ \sum_{a \in \text{Actions}(s)} \pi_{\text{opp}}(s,a) V_{\max,\text{opp}}(\text{Succ}(s,a)) & \text{Player}(s) = \text{opp} \end{cases}$$

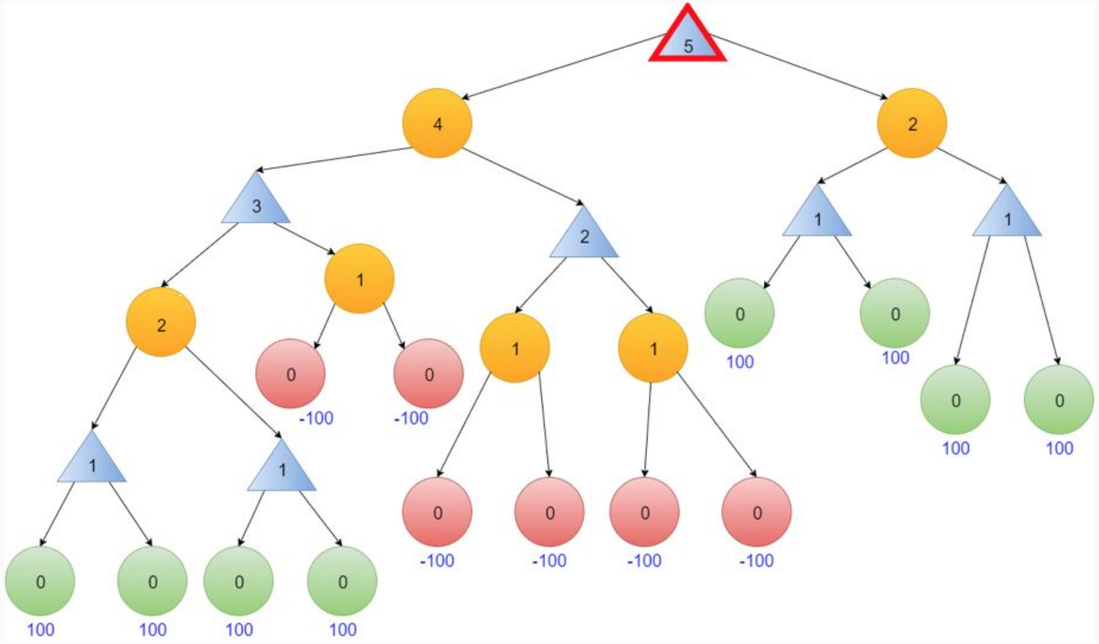
Expectimax

- Let's assume that the known policy is the following:

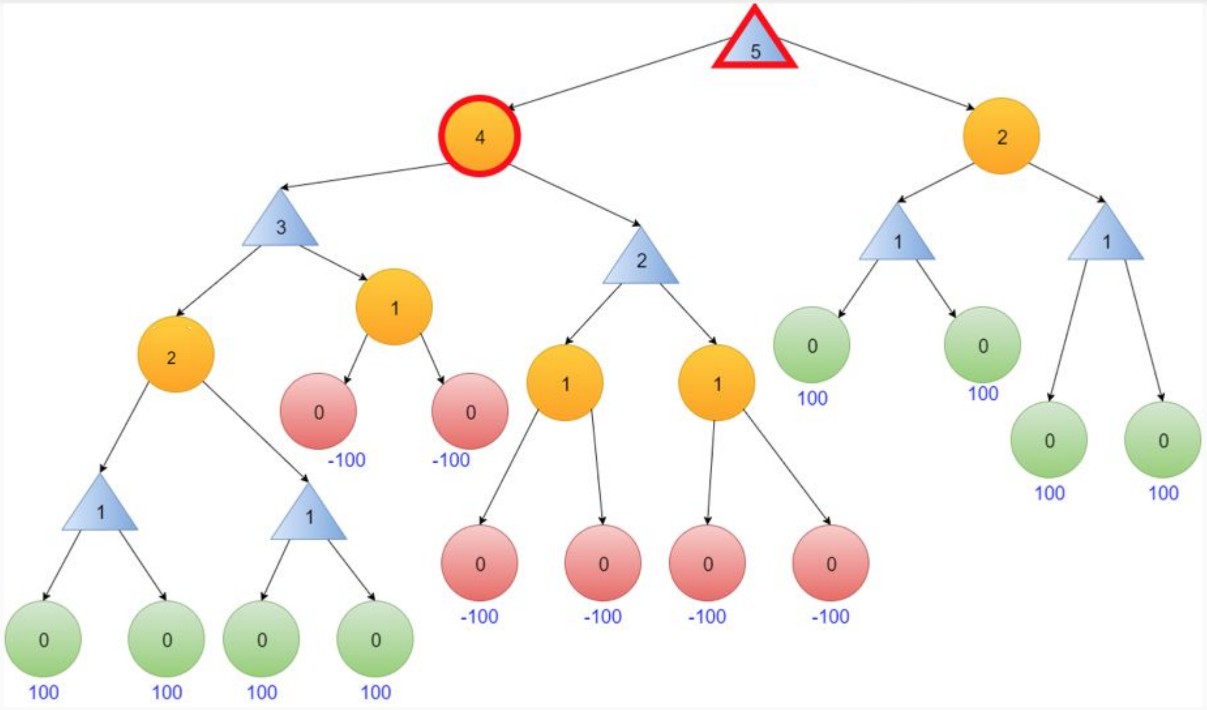
$$\pi_{opp}(s, SUB) = \begin{cases} 1, & \text{if N \% 2 == 1} \\ 0.5, & \text{otherwise} \end{cases}$$

$$\pi_{opp}(s, DIV) = \begin{cases} 0, & \text{if N \% 2 == 1} \\ 0.5, & \text{otherwise} \end{cases}$$

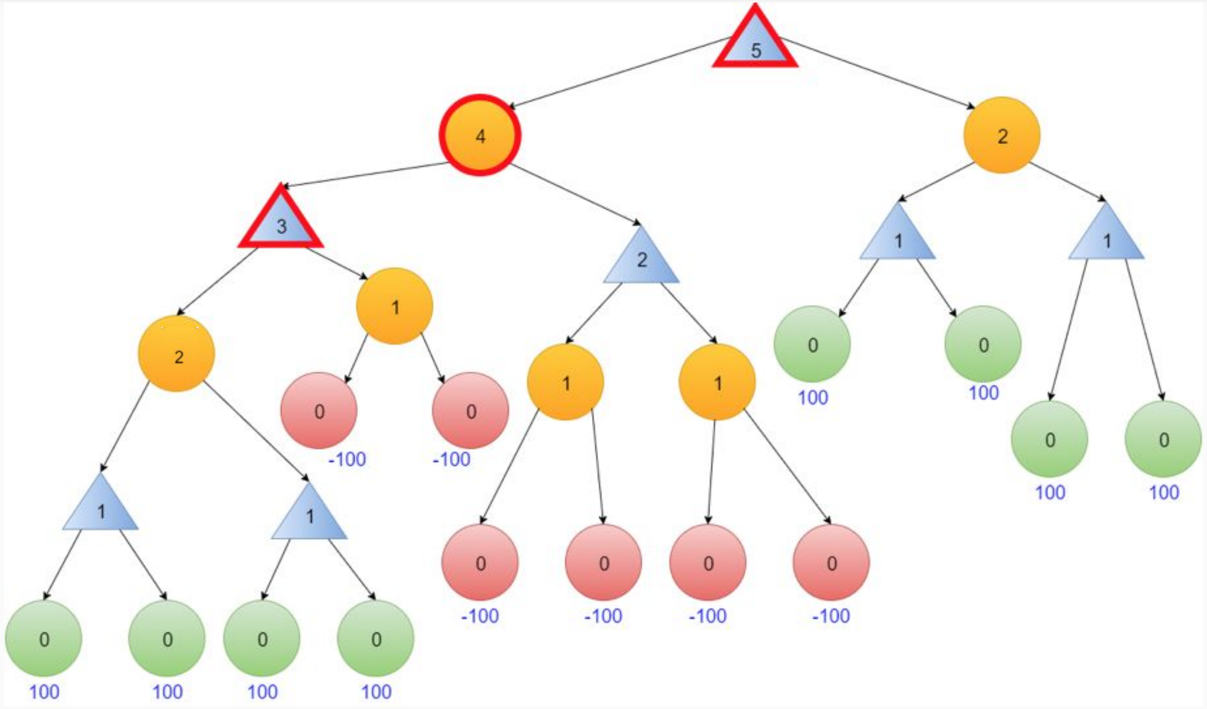
Expectimax


5
4
2
3
2
1
1
2
1
1
1
0
0
1
1
0
0
0
0
1
1
0
0
0
0
0
0
0
0
0
0
100
100
100
100
-100
-100
-100
-100
-100
-100
100
100
100
100

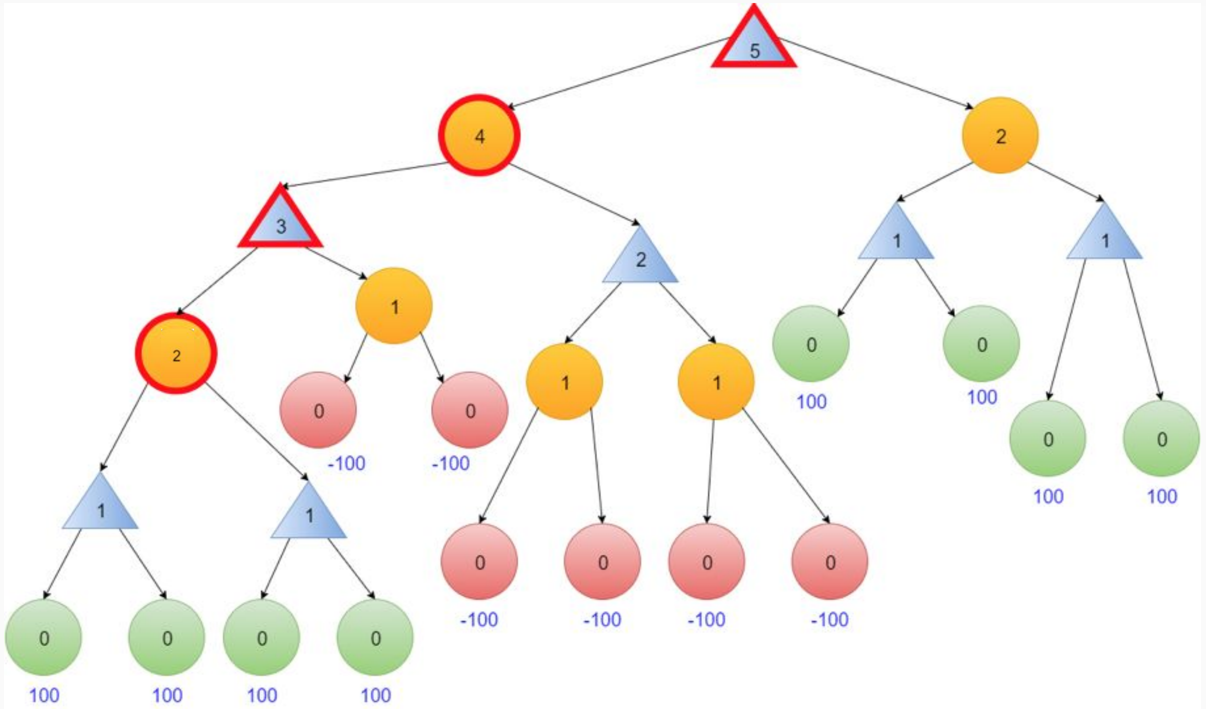
Expectimax


5
4
2
3
2
1
1
2
1
1
1
1
1
1
0
-100
0
-100
0
-100
0
-100
0
-100
0
-100
0
100
0
100
0
100
0
100
0
100
0
100
0
100
0
100

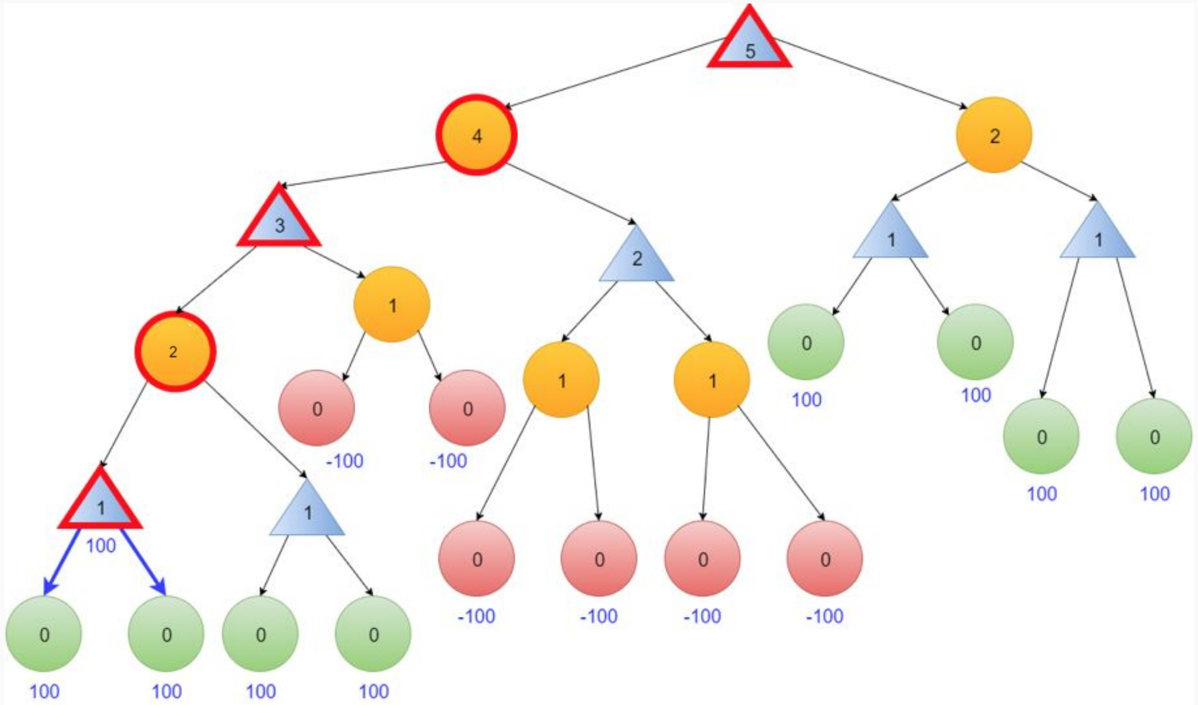
Expectimax


5
4
2
3
2
1
1
1
2
1
1
0
0
0
0
100
100
100
100
0
0
-100
-100
1
1
0
0
0
0
-100
-100
-100
-100
0
0
0
0
100
100
100
100
100
100
100
100

Expectimax

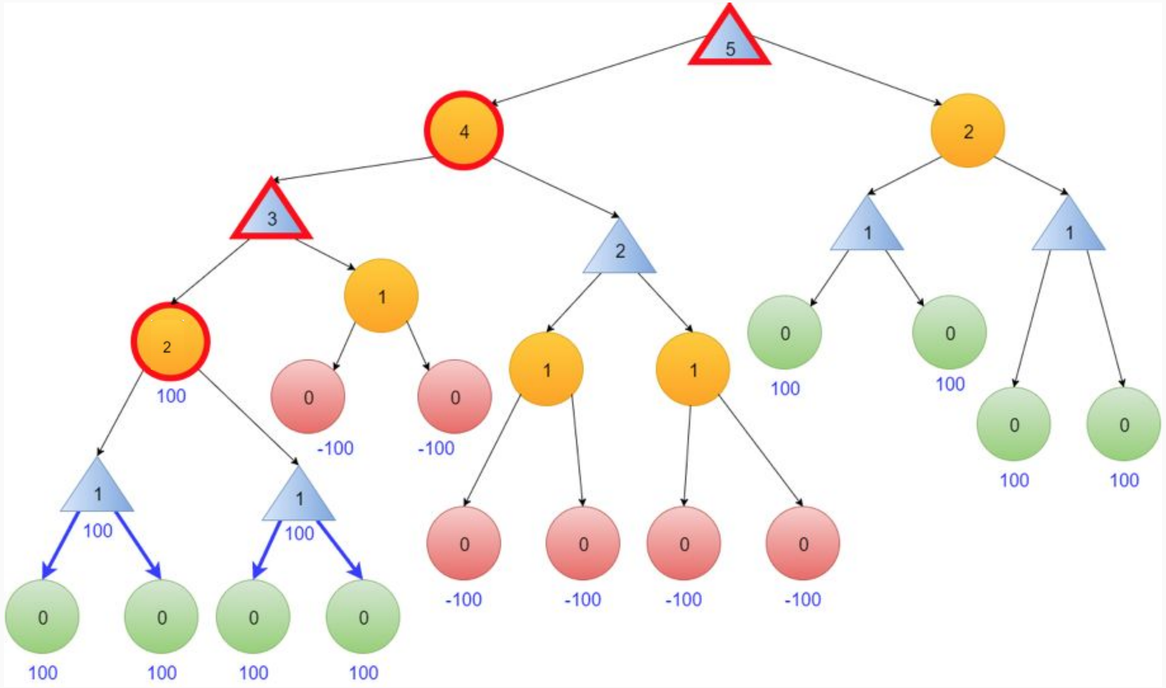

5
4
2
3
2
1
1
1
2
1
1
1
1
1
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
100
100
100
100
100
100
100
100
-100
-100
-100
-100
-100
-100

Expectimax



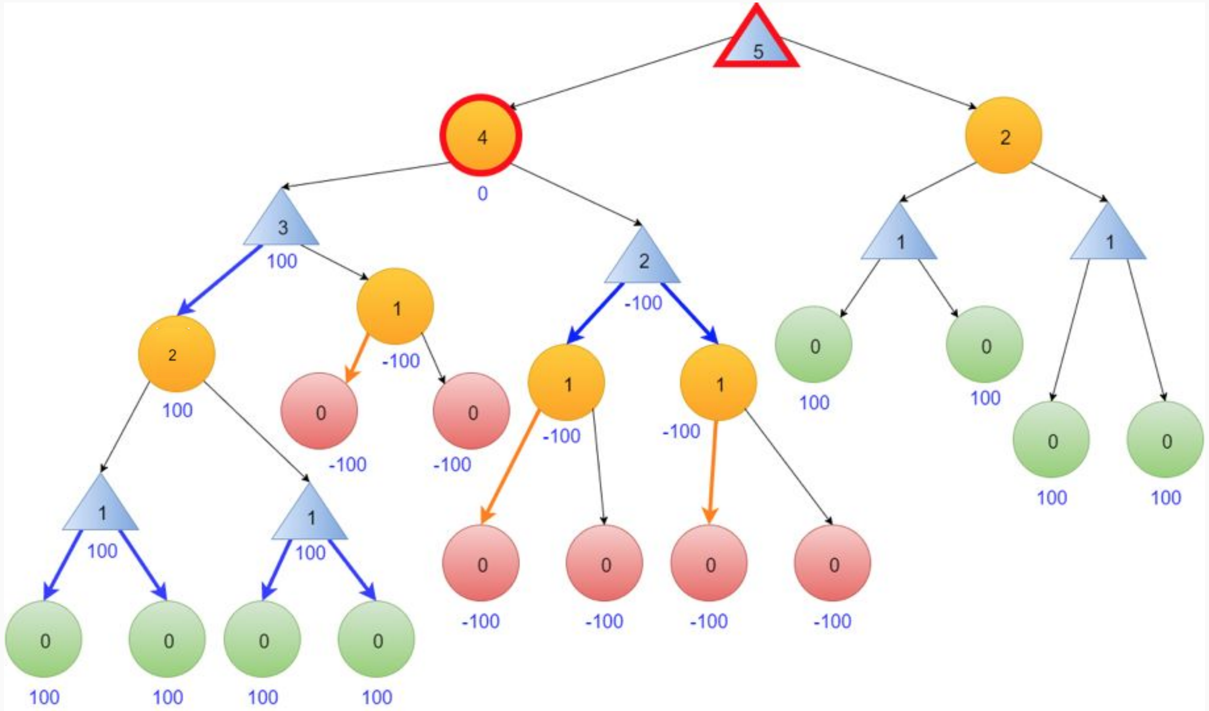
5
4
2
3
2
1
1
1
2
1
1
1
1
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
100
100
100
100
100
100
100
100
100
-100
-100
-100
-100
-100
-100

Expectimax

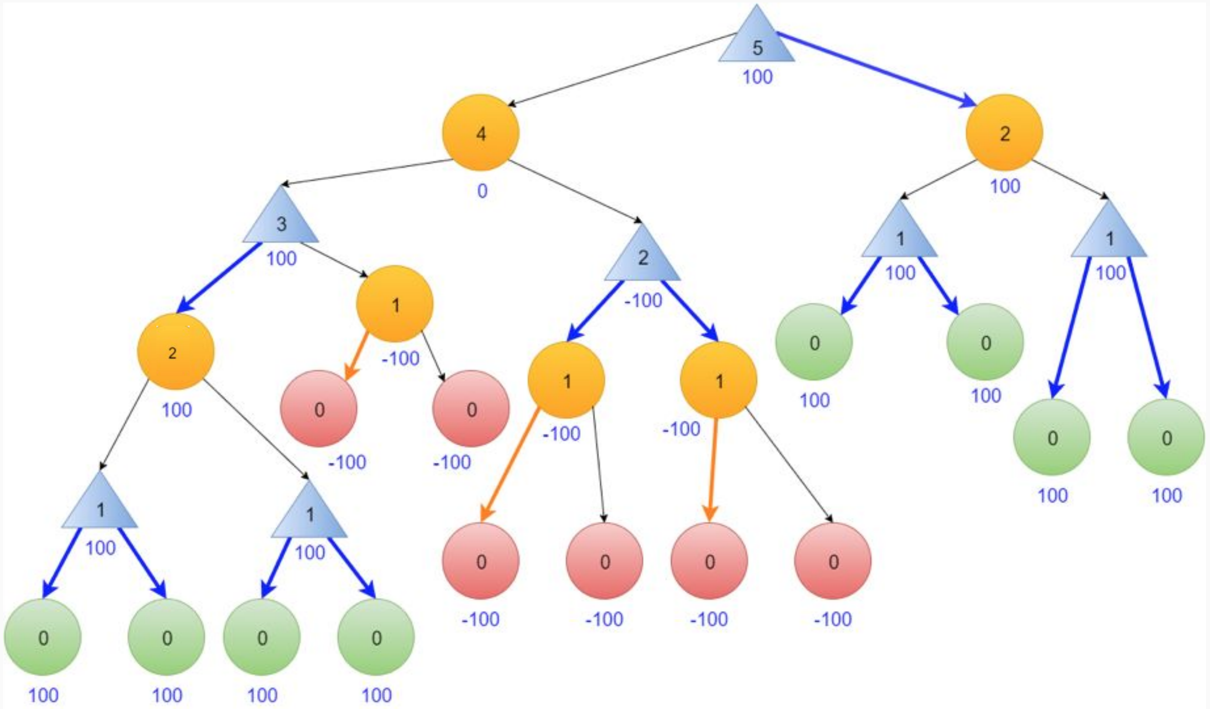


5
4
2
3
2
1
1
2
1
1
1
0
0
100
100
0
0
100
100
0
0
-100
-100
1
1
100
100
0
0
0
0
-100
-100
-100
-100
0
0
0
0
100
100
100
100

Expectimax

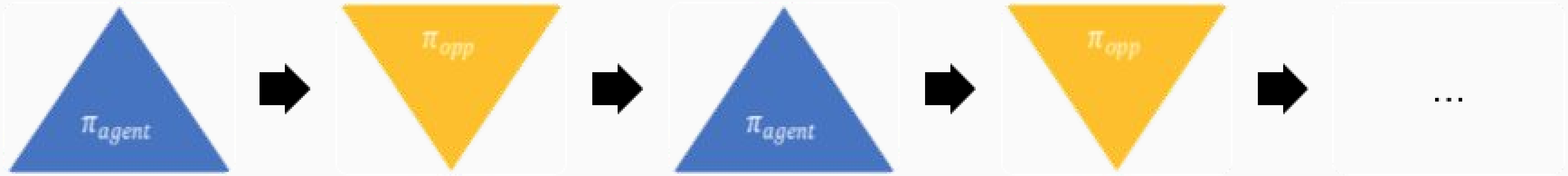

5
4
0
2
3
100
2
-100
1
1
1
0
0
2
1
-100
1
1
100
100
100
0
0
0
0
0
0
0
-100
-100
-100
-100
100
100
1
1
100
100
0
0
0
0
0
0
0
0
-100
-100
-100
-100
100
100
100
100

Expectimax


5
100
4
0
2
100
3
100
2
-100
1
100
1
100
2
100
1
-100
1
-100
1
-100
0
100
0
100
0
100
0
100
1
100
1
100
0
-100
0
-100
0
-100
0
-100
0
-100
0
-100
0
100
0
100
0
100
0
100

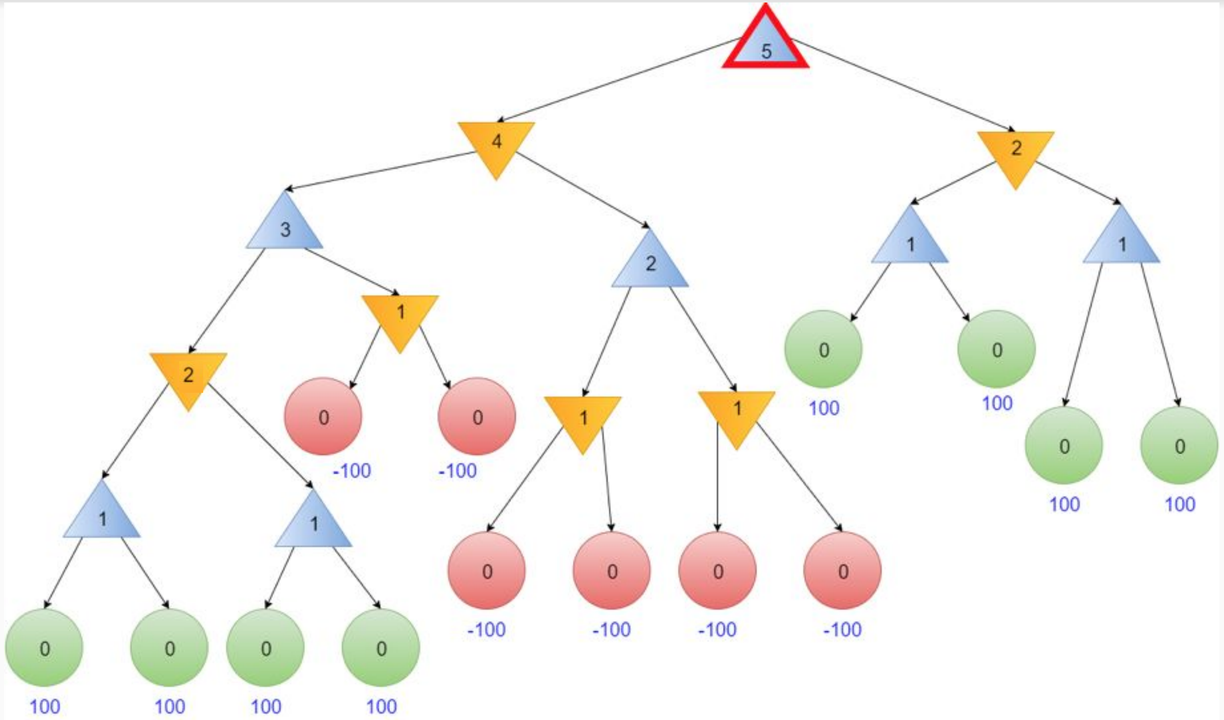
Minimax

- The agent chooses the policy knowing that the opponent is adversarial



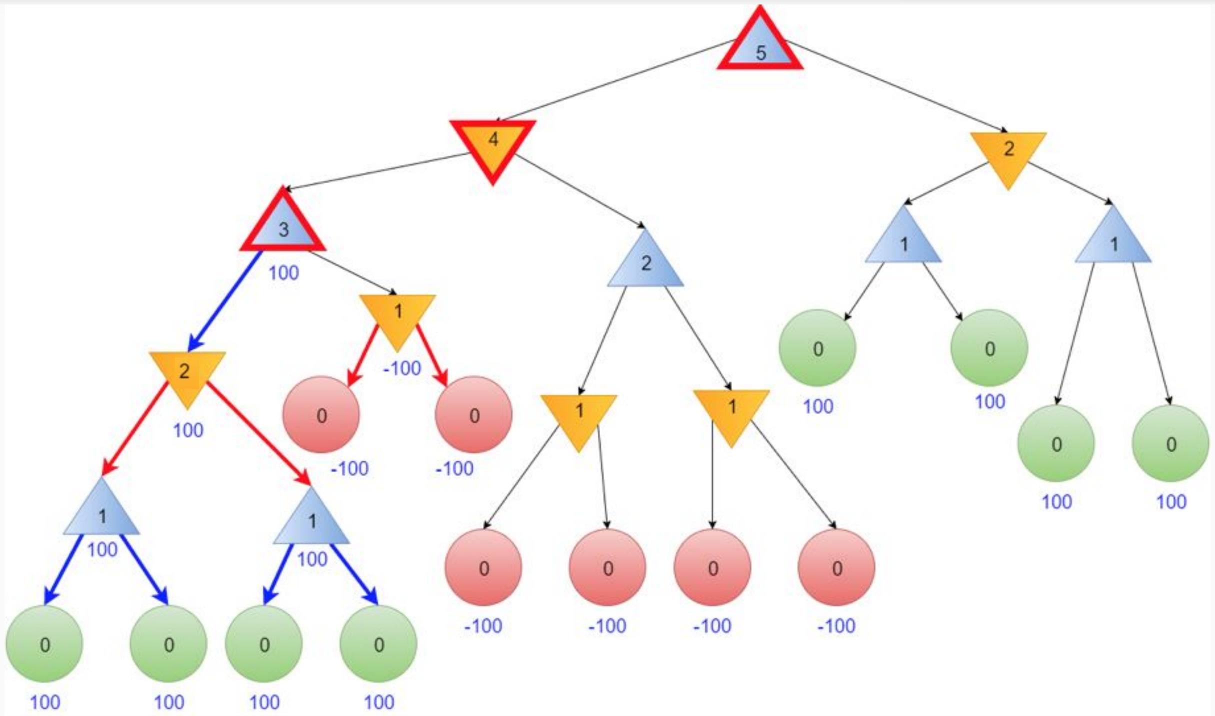
$$V_{\max,\min}(s) = \begin{cases} \text{Utility}(s) & \text{IsEnd}(s) \\ \max_{a \in \text{Actions}(s)} V_{\max,\min}(\text{Succ}(s,a)) & \text{Player}(s) = \text{agent} \\ \min_{a \in \text{Actions}(s)} V_{\max,\min}(\text{Succ}(s,a)) & \text{Player}(s) = \text{opp} \end{cases}$$

Minimax



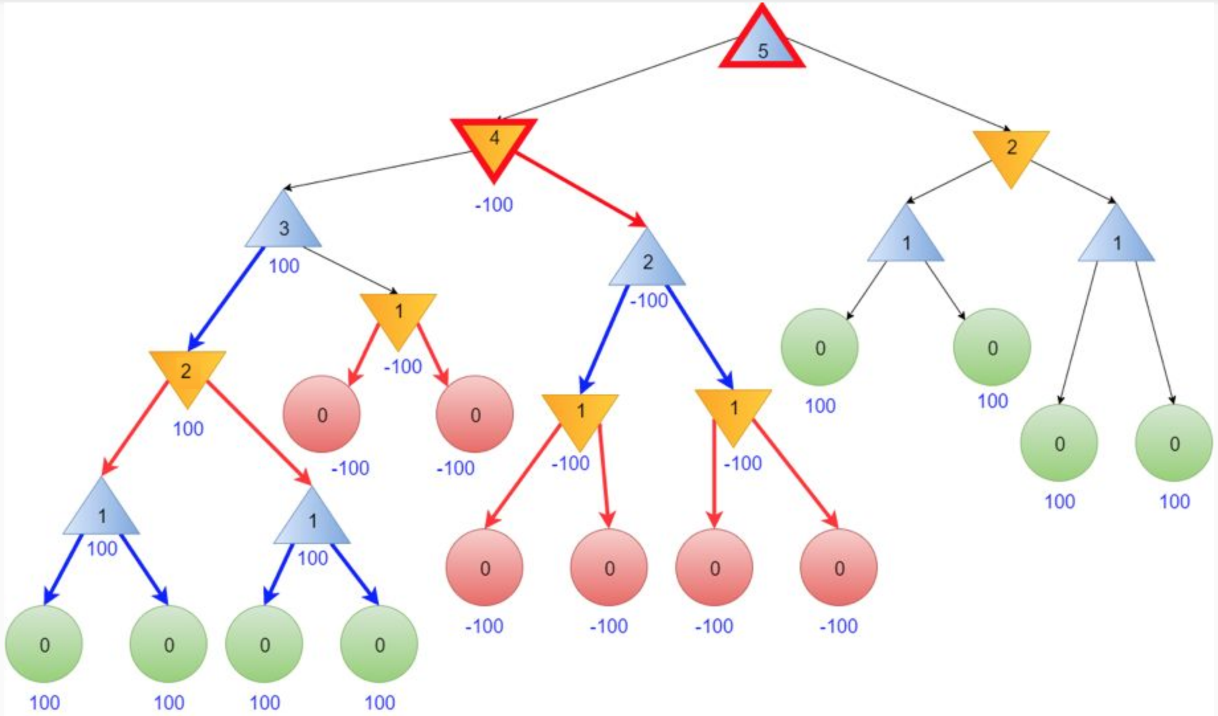
5
4
2
3
2
1
1
1
2
1
1
1
1
1
0
100
0
100
0
100
0
100
0
-100
0
-100
0
-100
0
-100
0
-100
0
-100
0
100
0
100
0
100
0
100

Minimax



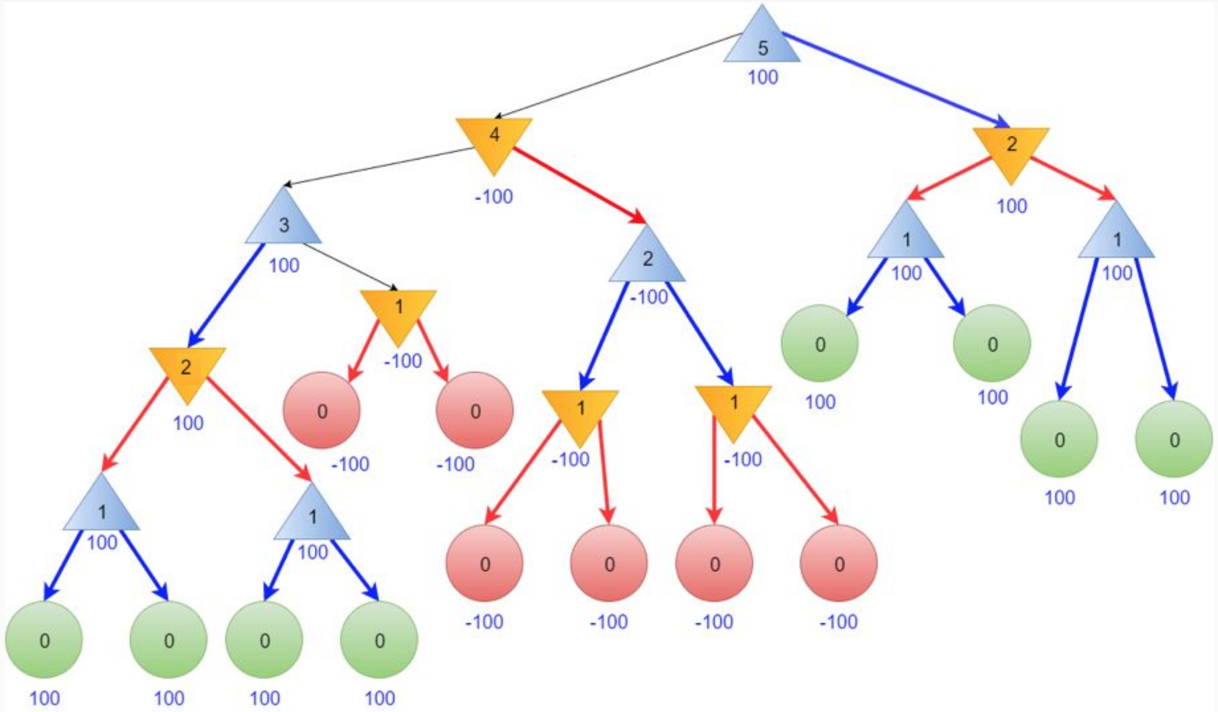
5
4
2
3
100
1
-100
2
1
1
2
100
0
0
-100
-100
1
1
0
0
100
100
1
100
1
100
0
0
0
0
-100
-100
-100
-100
0
0
100
100
0
0
0
0
100
100
100
100

Minimax



5
4
-100
2
3
100
1
-100
2
-100
1
1
2
100
0
0
-100
-100
1
-100
1
-100
0
0
100
100
0
0
100
100
1
100
1
100
0
0
0
0
-100
-100
-100
-100
0
0
0
0
100
100
100
100

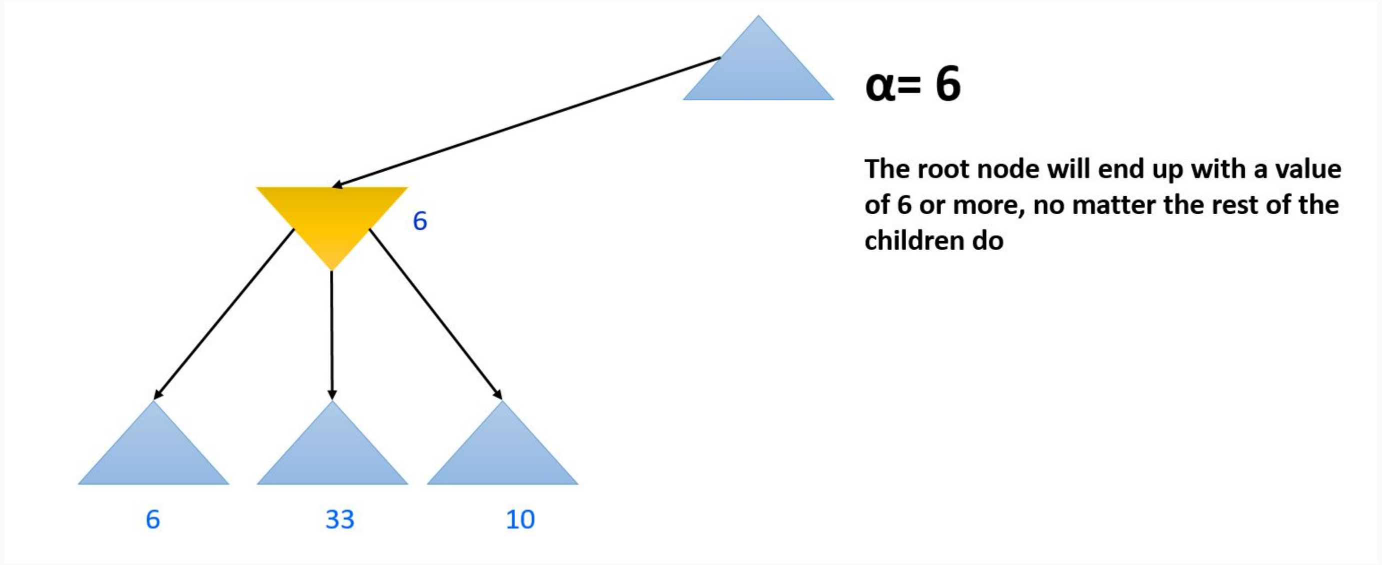
Minimax


5
100
4
-100
2
100
3
100
1
-100
2
-100
1
100
1
100
2
100
0
0
1
-100
1
-100
0
0
1
100
1
100
0
0
0
0
0
0
-100
-100
-100
-100
-100
-100
100
100
100
100
100
100
100
100

Minimax w/ alpha beta pruning

- The alpha beta procedure can speed up depth first minimax search
- Alpha: a lower bound on the value that a maximizing node may ultimately be assigned
    - $v >= \alpha$
- Beta: an upper bound on the value that a minimizing node may ultimately be assigned
    - $v <= \beta$

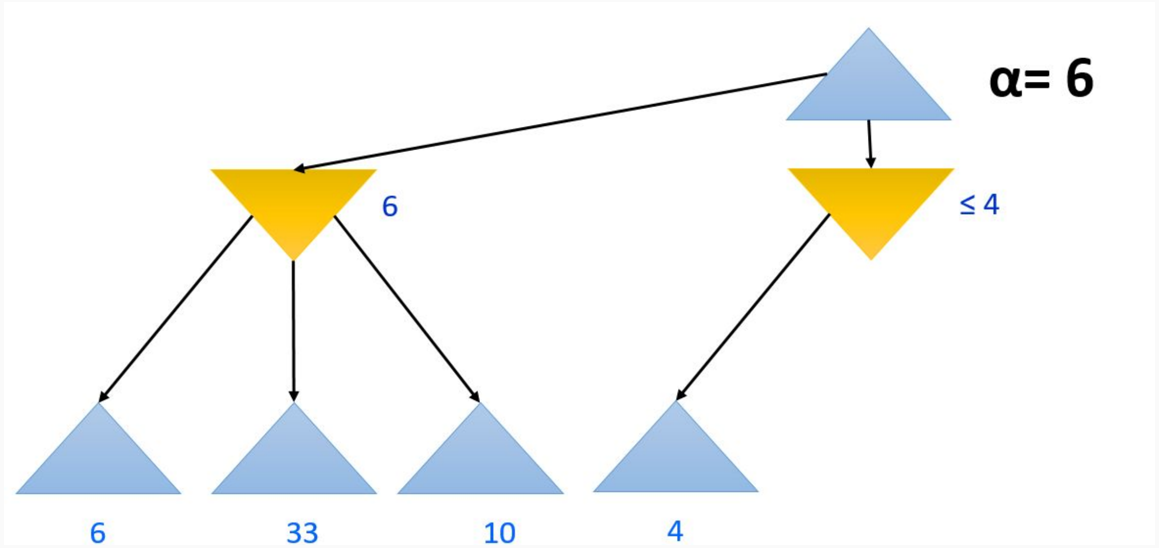
Minimax w/ alpha beta pruning



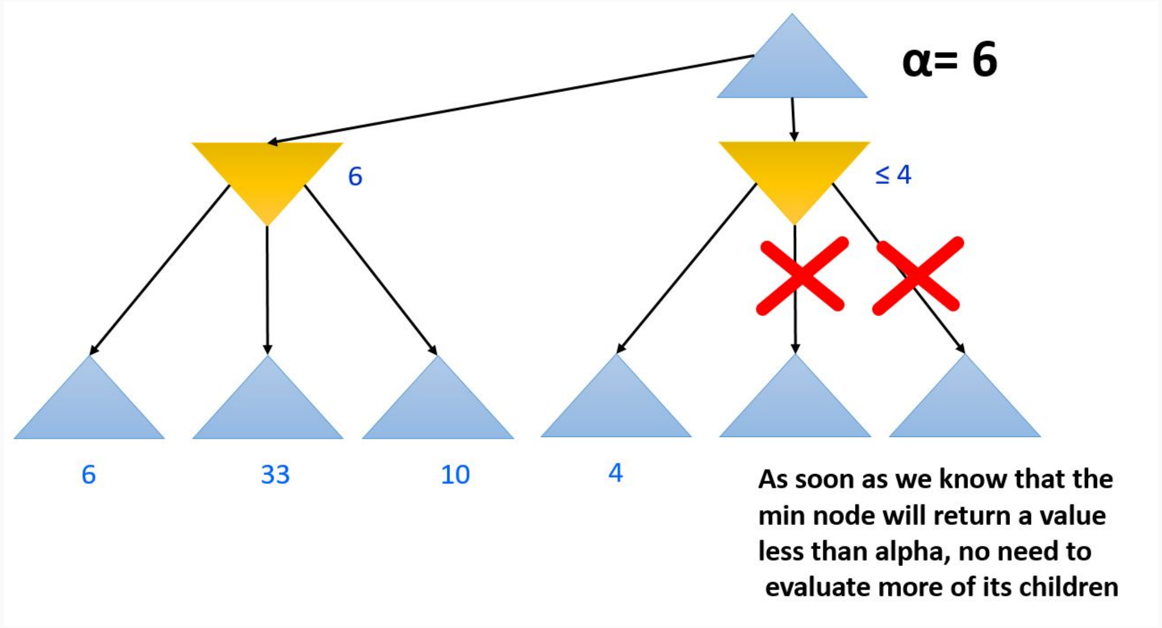
α= 6

The root node will end up with a value of 6 or more, no matter the rest of the children do

Minimax w/ alpha beta pruning

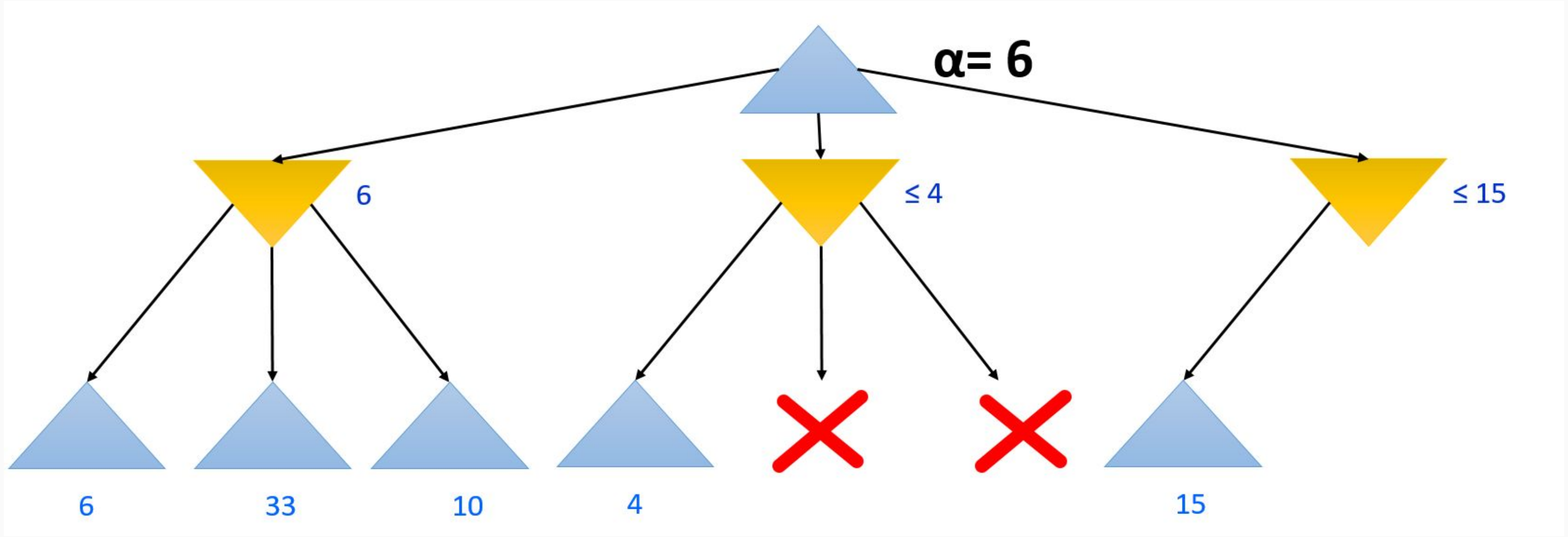


Minimax w/ alpha beta pruning

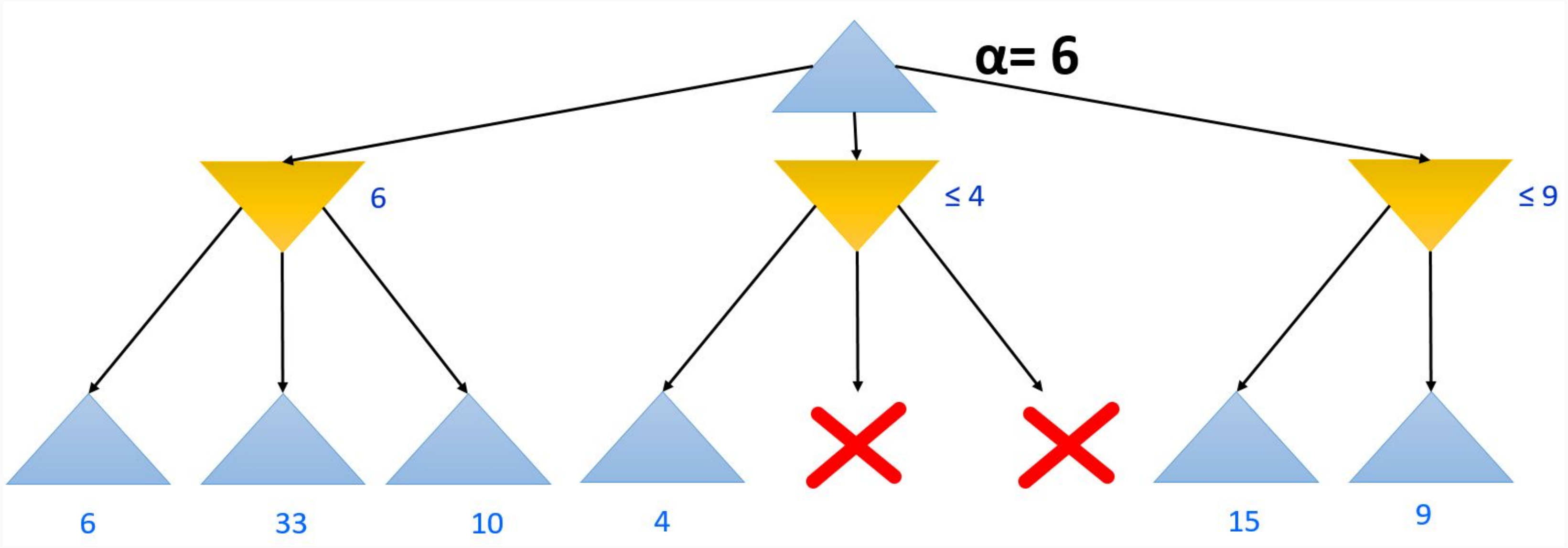


As soon as we know that the min node will return a value less than alpha, no need to evaluate more of its children

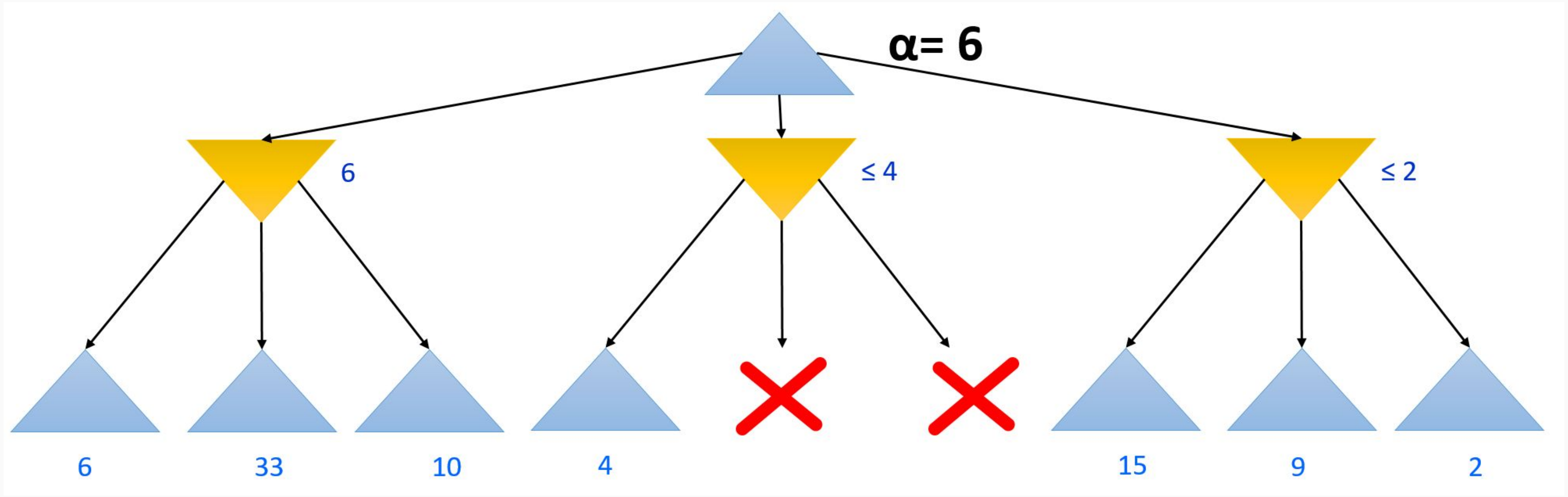
Minimax w/ alpha beta pruning



Minimax w/ alpha beta pruning



Minimax w/ alpha beta pruning



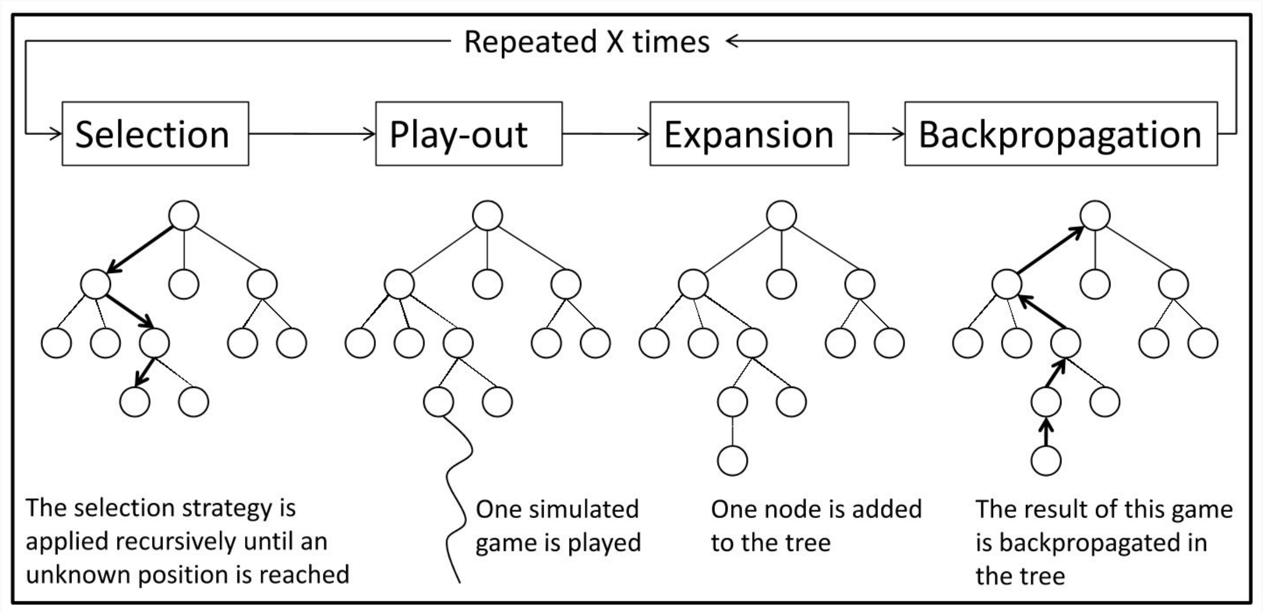
Minimax w/ alpha beta pruning

- Pruning does not affect the final result
    - Gets the same result as a full minimax search
- Good move ordering could potentially improve the effectiveness of pruning
    - With best ordering, can double the depth of the search space
    - Really helpful for problems with big branching factors

Monte Carlo Tree Search (MCTS)

- Heuristic search algorithm for decision problems
  - Can be used for playing board games, real time video games and nondeterministic games like poker
- Focuses on the analysis of the most promising moves, expanding the search tree based on random sampling of the search space
  - The game is played out to the very end by selecting moves at random.
  - The final game result of each simulation is then used to weight the nodes in the game tree so that the better nodes are more likely to be chosen in future playouts.

Monte Carlo Tree Search (MCTS)


Repeated X times
Selection
Play-out
Expansion
Backpropagation
The selection strategy is applied recursively until an unknown position is reached
One simulated game is played
One node is added to the tree
The result of this game is backpropagated in the tree

Let’s conquer Go

Armed with the methods learned so far, can we take on GO?!?!

What makes Go difficult?

- Branching factor ~ 300, vs 35 in Chess
- Games last longer (150 moves, ~2x Chess)
- Total number of possible games
    - ~10^120 for Chess
    - ~10^761 for Go
    - There’s only ~10^80 atoms in the universe!

AlphaGo’s Approach

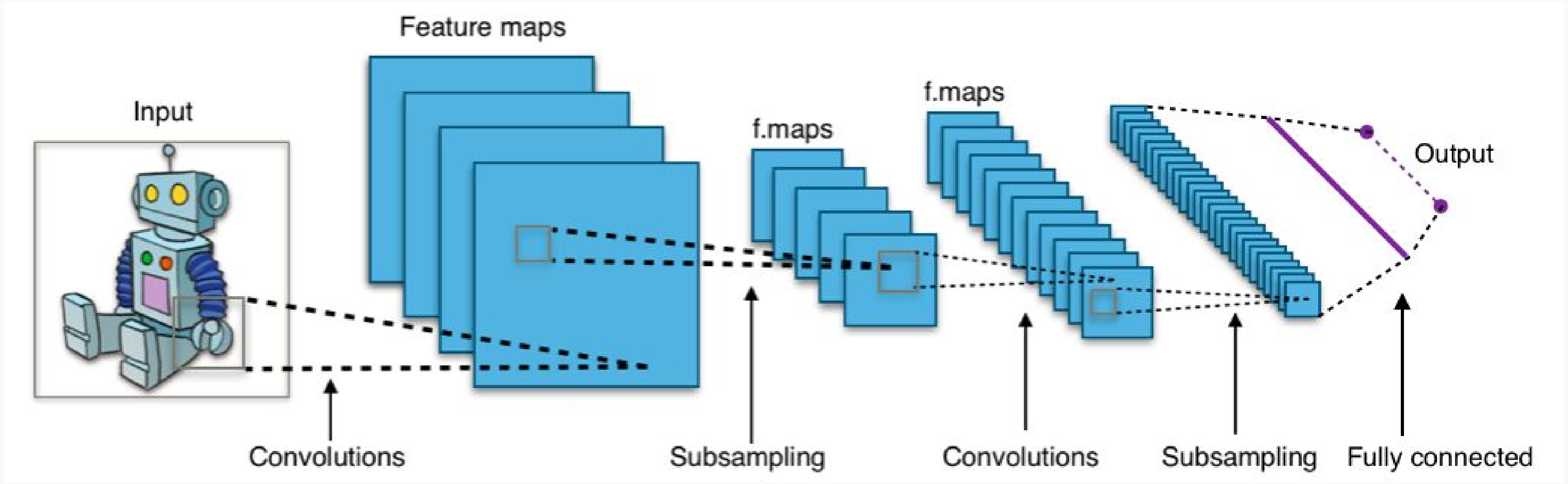
- Supervised learning from seeing many expert games
- Convolutional Neural Networks for selecting moves and predicting winner
  - Reinforcement learning to improve these networks
- MCTS

Neural Networks for Go

- Policy function: select next move to play
- Value function: predict winner
- (Both of these are ConvNets, with boards are passed in as 19x19 images)

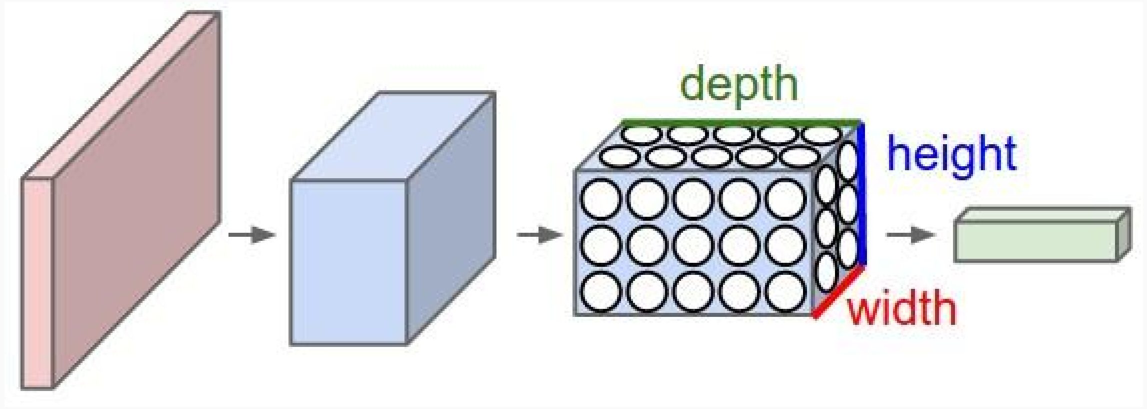
- Train on seeing human games
- Play neural networks against themselves to improve (reinforcement learning!)

Convolutional Neural Networks



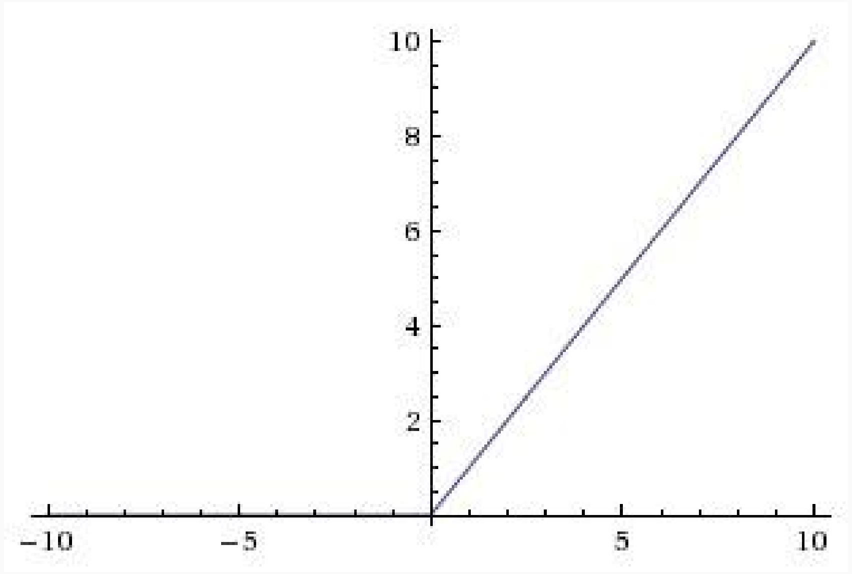
CNNs: Conv layer

- Compute dot product between weights and small region
- Learn a set of filters



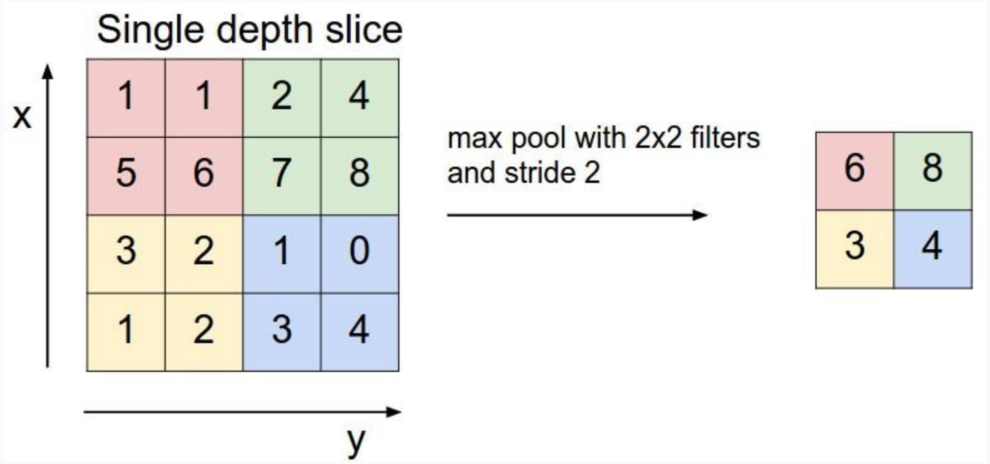
CNNs: ReLU

- max(0, x)
- Fast, without losing much accuracy



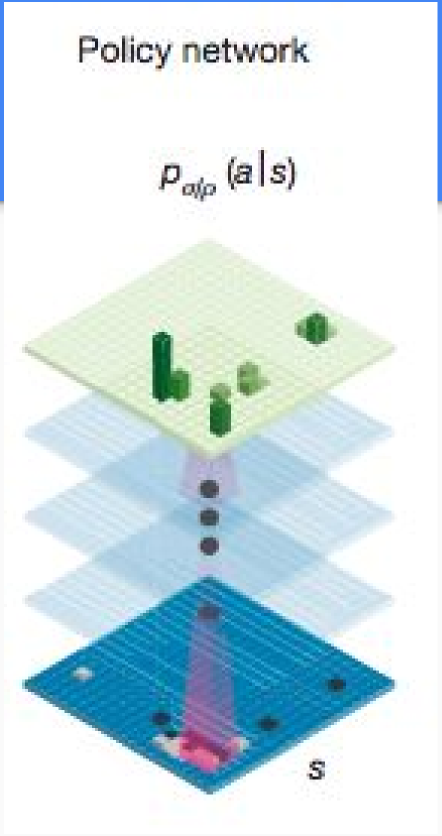
CNNs: Pooling/Subsampling

- Downsampling
- I.e. replace 2x2 with 1 pixel, often with max



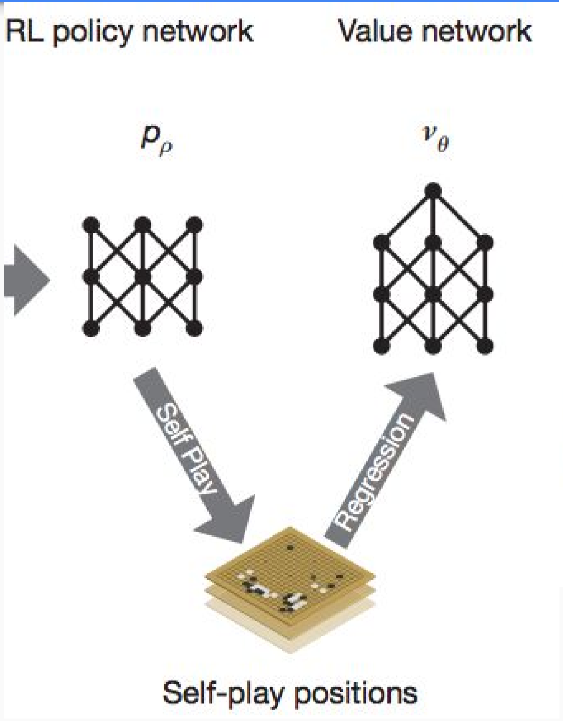
Policy Network

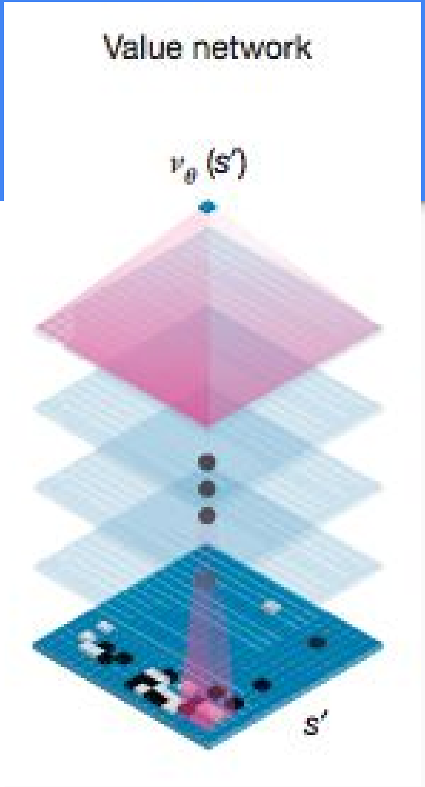
- Goal: find which states to explore in MCTS
- Alternate between convolutional layers and ReLU
- Output probability dist over all legal moves
- 13 layers!



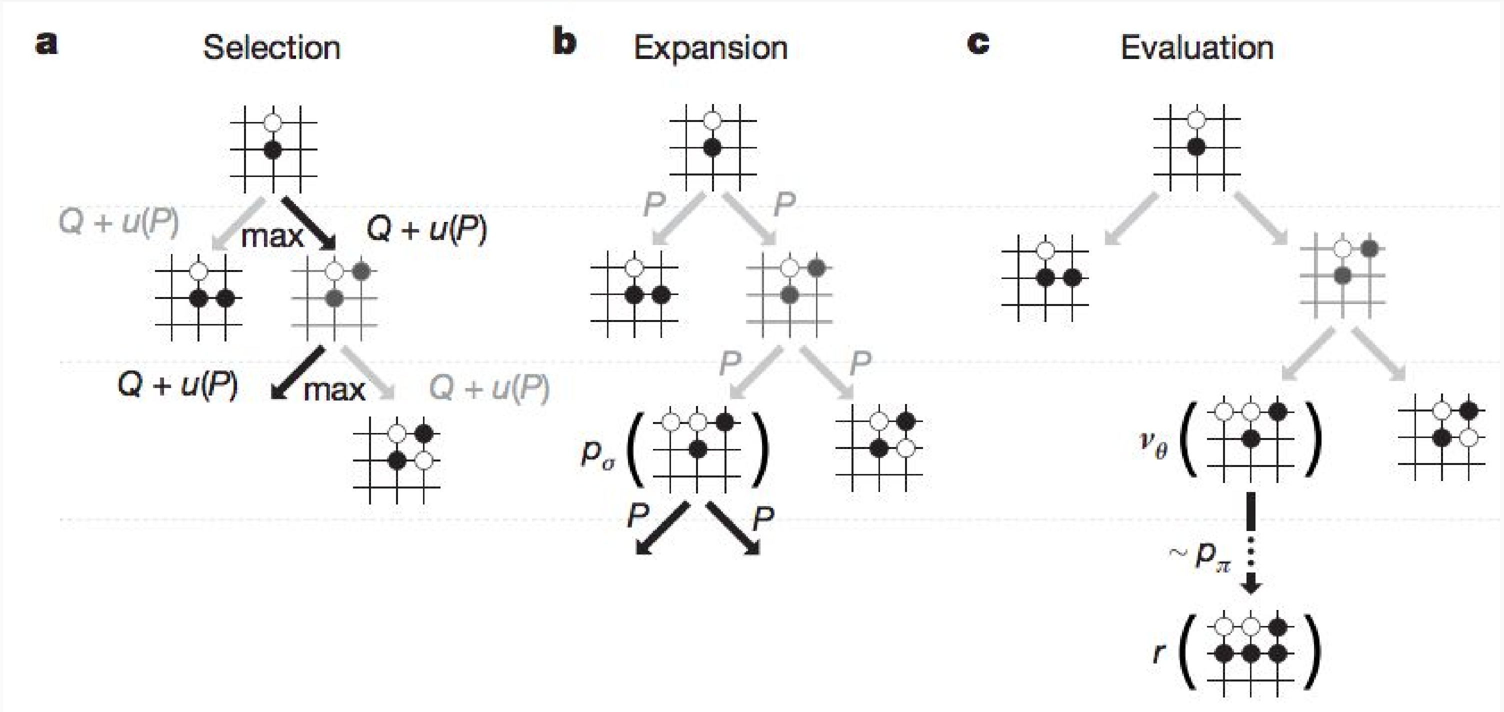
Value Network

- Calculate from policy network
- Outputs a single prediction

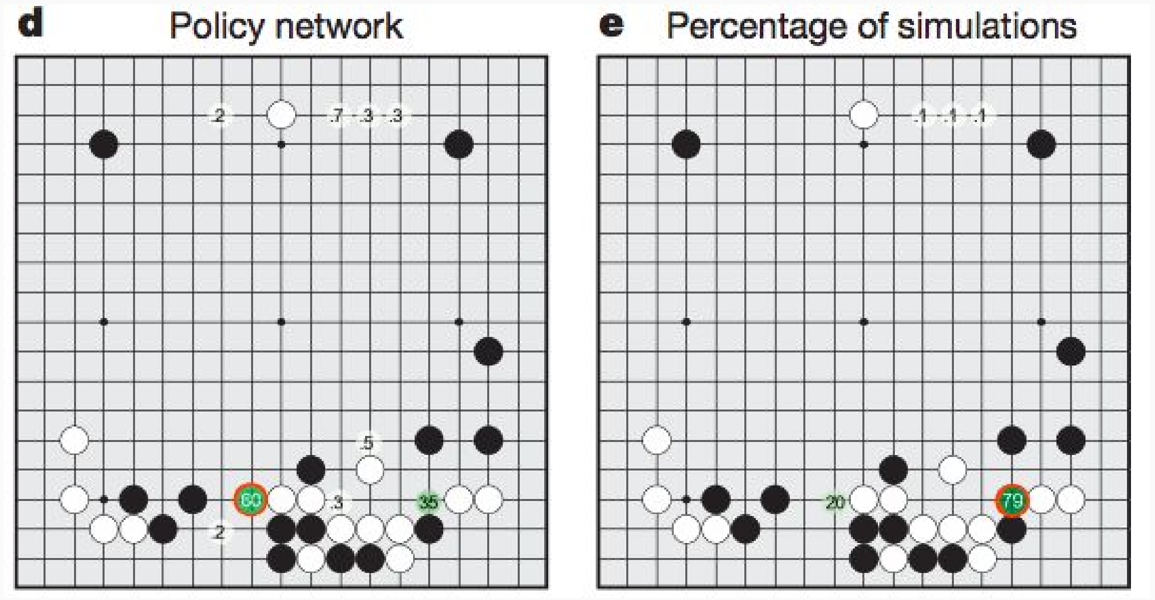




MCTS in AlphaGo

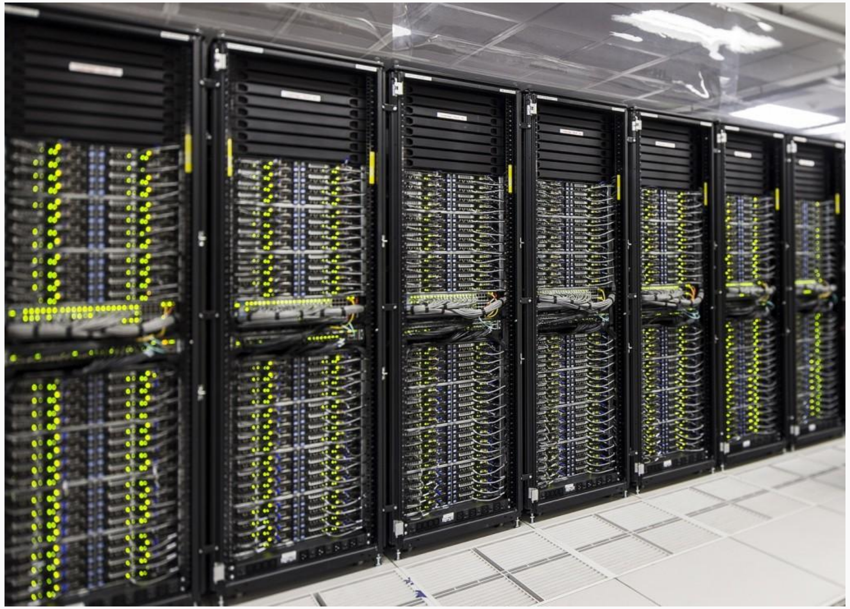


MCTS


d
Policy network
2
.7
.3
.3
.5
60
.3
35
2
e
Percentage of simulations
.1
.1
.1
20
79

Computational Power

- 1202 CPUs
- 176 GPUs
- Specialized hardware against Lee Sedol



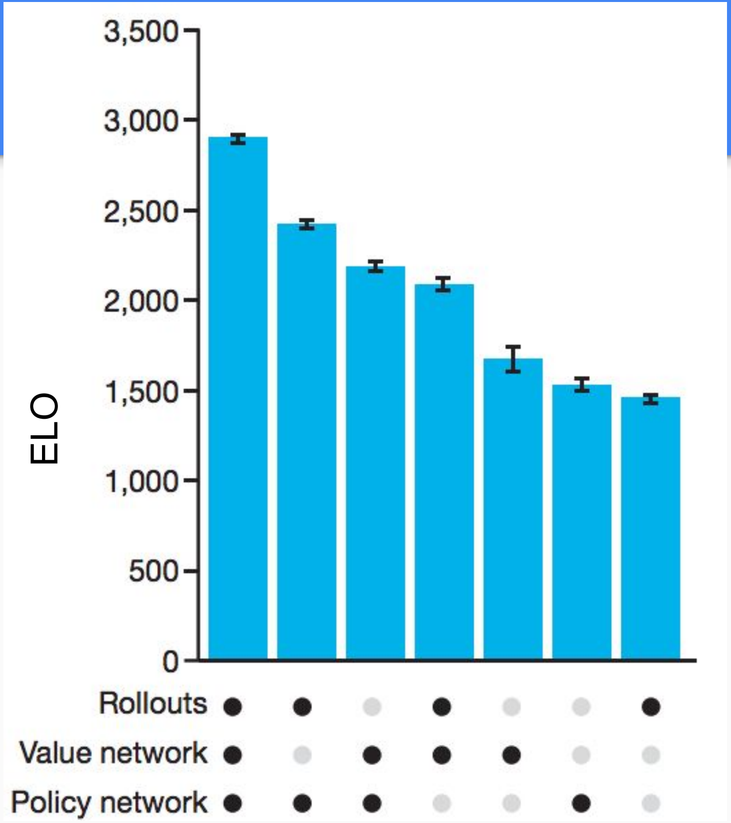
AlphaGo vs Lee Sedol, Game 4

- White 78 (Lee): unexpected move (even other professional players didn't see coming) - needle in the haystack
- AlphaGo failed to explore this in MCTS
- Policy network hadn't been trained for long enough

Summary

AlphaGo applied advanced versions of techniques in this class!

Name	ELO
Lee Sedol	3517
AlphaGo (2016)	~3594
Ke Jie (world champion)	3616


ELO
Rollouts
Value network
Policy network