MCTS review

• Pick a move based on what performs best over many rollouts.

• Each rollout consists of four phases
  1. Selection
  2. Expansion
  3. Simulation
  4. Backpropagation

• Each rollout expands one node.
• Use selection when we have enough data.
• Use simulation to avoid too much data.
Videos

https://www.youtube.com/watch?v=g-dKXOlsf98

https://www.youtube.com/watch?v=tXlM99xPQC8
AlphaGo Neural Networks

Rollout policy

$P_{\pi}$

SL policy network

$P_{\sigma}$

RL policy network

$P_{\rho}$

Value network

$V_{\theta}$

Tree Policy

Classification

Human expert positions

Default Policy

Self-play Positions

Data

Neural Network

Policy Gradient
Step 1: learn to predict human moves

- Used a large database of online expert games.

- Learned two versions of the neural network:
  - A fast network $P_\pi$ for use in simulation.
  - An accurate network $P_\sigma$ for use in selection.
Step 2: improve $P_\sigma$ (accurate network)

- Run large numbers of self-play games.

- Update $P_\sigma$ using reinforcement learning
  
  - weights updated by stochastic gradient descent
Step 3: learn a better boardEval $V_\theta$

- use random samples from the self-play database

- prediction target: probability that black wins from a given board

**CS63 topic**

avoiding overfitting weeks 9-10
AlphaGo Tree Policy (selection)

- select nodes randomly according to weight:

\[ \text{value} + \frac{\text{prior}}{1 + \text{visits}} \]

- *prior* is determined by the improved policy network

\[ p_\rho \]
When expanding a node, its initial value combines:

- an evaluation from value network $V_\theta$
- a rollout using fast policy $P_\pi$

$$v(s) = \lambda \times \text{outcome} \quad + \quad (1 - \lambda) \times \text{eval}$$

A rollout according to $P_\pi$ selects random moves with the estimated probability a human would select them instead of uniformly randomly.
AlphaGo Results

- Played European champ Fan Hui (October 2015)
  - AlphaGo won 5-0.

- Played former world #1 Lee Sedol (March 2016)
  - AlphaGo won 4-1.

- Played online against top pros (Dec 2016 – Jan 2017)
  - AlphaGo won 60-0.

- Played current world #1 Ke Jie (May 2017)
  - AlphaGo won 3-0.

- AlphaGo Zero vs AlphaGo (summer 2017)
  - AlphaGo Zero won 89-11.
AlphaGo Zero

• Learns only from self-play.

• Initially plays randomly (MCTS with bad heuristics).

• Trains a neural network to predict:
  • What move it will play
  • Which player will win

• As the predictions get better, the MCTS gets better.

*We’ll come back to this later in the semester.*
MCTS vs Bounded Min/Max

**UCT / MCTS**
- Optimal with infinite rollouts.
- Anytime algorithm (can give an answer immediately, improves its answer with more time).
- A heuristic is not required, but can be used if available.
- Handles incomplete information gracefully.

**MinMax/Backward Induction**
- Optimal once the entire tree is explored or pruned.
- Can prove the outcome of the game.
- Can be made anytime-ish with iterative deepening.
- A heuristic is required unless the game tree is small.
- Hard to use on incomplete information games.
Discussion: why use MCTS for go?

• We’re using MCTS in lab because we don’t want to write new heuristics for every game.
  • AlphaGo is all about heuristics. They’re learned by neural networks, but they’re still heuristics.

• MCTS handles randomness and incomplete information better than Min/Max.
  • Go is a deterministic, perfect information game.

So why does MCTS make so much sense for go?