CS440/ECE448 Lecture 29: Monte Carlo Search & Expectiminimax

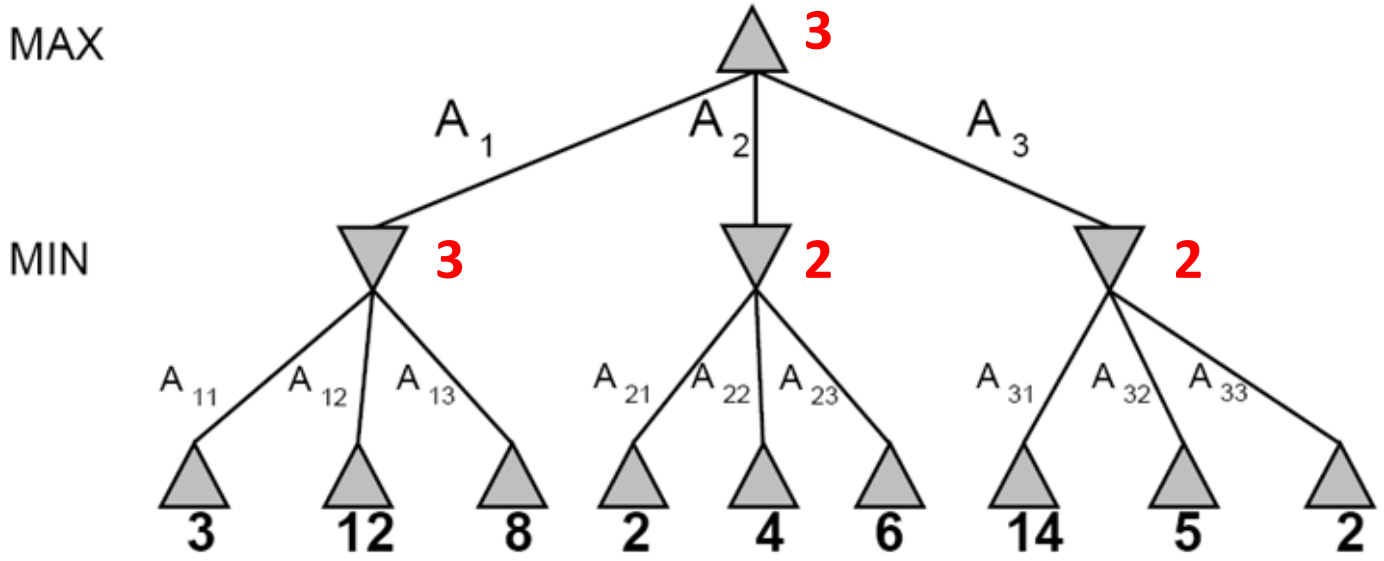
Mark Hasegawa-Johnson

A contemporary backgammon set. Public domain photo by Manuel Hegner, 2013, https://commons.wikimedia.org/w/index.php?curid=25006945

Outline

- Review: Minimax
- Monte Carlo search: Estimate value by playing randomly
- Expectiminimax: Optimal search if transitions are random
- Relationship of expectiminimax to MDP

Computing the minimax value of a node



- **Minimax**(*node*) =
  - Utility(*node*) if *node* is terminal
  - $\max_{action}$ **Minimax**(Succ(*node, action*)) if *player* = MAX
  - $\min_{action}$ **Minimax**(Succ(*node, action*)) if *player* = MIN

Evaluation functions

It's impossible to search all the way to the end of the game. At some fixed depth, we need to stop and estimate the value of s using some cheap but reasonably accurate estimate of $v(s)$. It should have the following properties:

- $v(s)$ should be a reasonable estimate of the outcome of the game, but
- It must be possible to compute $v(s)$ quickly, i.e., typically we desire polynomial complexity.

Outline

- Review: Minimax
- Monte Carlo search: Estimate value by playing randomly
- Expectiminimax: Optimal search if transitions are random
- Relationship of expectiminimax to MDP

Methods for estimating value of a board

- For many traditional games (e.g., chess, go, checkers), there are pretty good linear estimators: $v(s)$ is a weighted sum of some features
- A better estimate is possible by training a convolutional neural net (CNN): observe the board game as an image, and estimate $v(s)$
- An even better estimate can be obtained by combining the CNN with Monte Carlo search

Monte Carlo search

Monte Carlo search is named after the famous casino at Monte Carlo. It estimates the value of a board position by playing N random games:

- Choose a sequence of moves at random, until one player wins
- Repeat N times
- Estimate $v(s)$ as

$$v(s) = \frac{(\#\text{ times Max wins})}{N}$$

Monte Carlo Search in AlphaGo

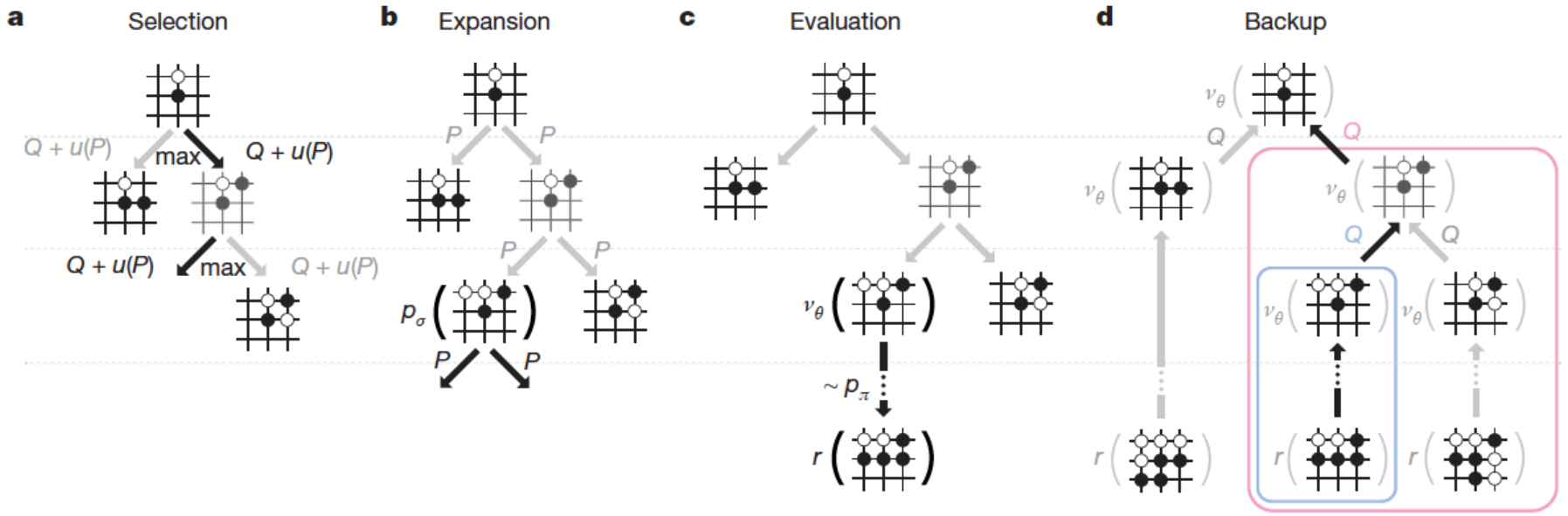


Figure 3 | Monte Carlo tree search in AlphaGo. a, Each simulation traverses the tree by selecting the edge with maximum action value Q, plus a bonus $u(P)$ that depends on a stored prior probability P for that edge. **b**, The leaf node may be expanded; the new node is processed once by the policy network p_σ and the output probabilities are stored as prior probabilities P for each action. **c**, At the end of a simulation, the leaf node is evaluated in two ways: using the value network v_θ; and by running a rollout to the end of the game with the fast rollout policy p_π, then computing the winner with function r. **d**, Action values Q are updated to track the mean value of all evaluations $r(\cdot)$ and $v_\theta(\cdot)$ in the subtree below that action.

D. Silver et al., Mastering the Game of Go with Deep Neural Networks and Tree Search, Nature 529, January 2016

Warnings about Monte Carlo search

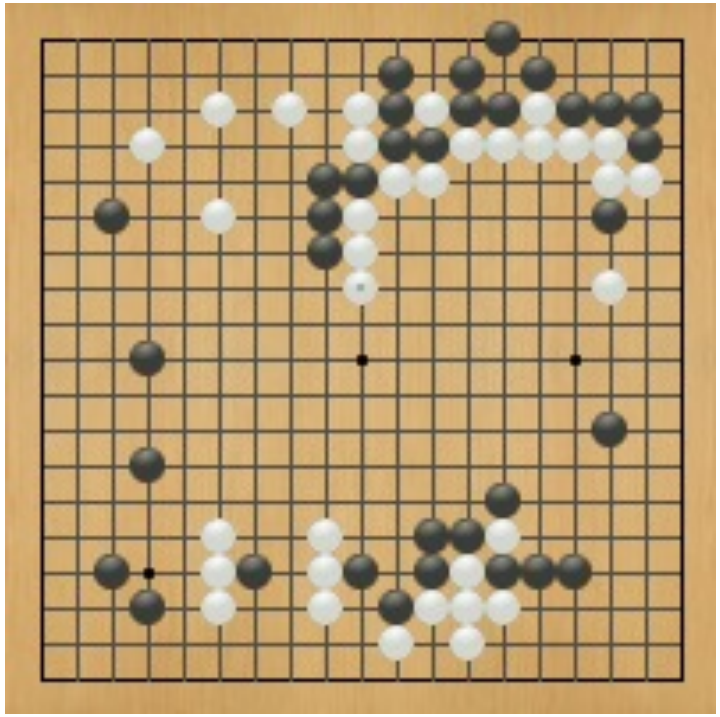
- Monte Carlo search is neither an upper bound nor a lower bound on the true value of a board
- There's no guarantee that it will give you a good estimate

Outline

- Review: Minimax
- Monte Carlo search: Estimate value by playing randomly
- Expectiminimax: Optimal search if transitions are random
- Relationship of expectiminimax to MDP

Monte Carlo search: the game is deterministic, it's just too deep to search

- Monte Carlo search is used to estimate $v(s)$ when the game is deterministic, but too deep to search
- Examples: chess, go



Expectiminimax: The game itself is random

- Expectiminimax is a generalization of minimax for the case when the game itself is random.
- The key idea:
  - instead of Max trying to maximize their own score…
  - …they try to maximize the ***expected value*** of their score…
  - …while Min is trying to minimize the expected value.
- Expecti…mini…max

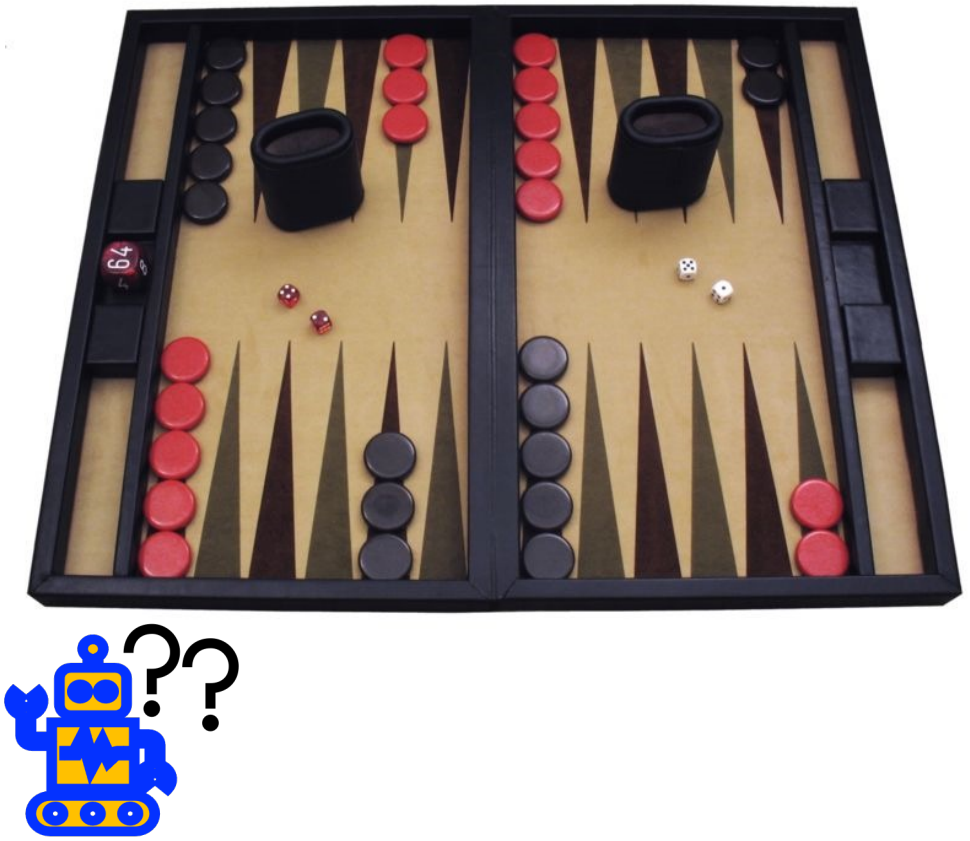
Expectiminimax

State evolves **stochastically** (when a player acts, the game changes RANDOMLY, with a probability distribution $P(S_{t+1} = s' | S_t = s, a)$ that depends on the action, a).

The player tries to maximize his or her own reward:

- **Maximize** (over all possible moves I can make) the
- **Expected value** (over all possible successor states) of the resulting utility:

$$\sum_{s'} P(S_{t+1} = s' | S_t = s, a) v(s')$$



Expectiminimax

State evolves **stochastically** (when a player acts, that action influences the state transition probability).

Current state is visible to both players.

Each player tries to maximize his or her own reward:

- **Maximize** (over all possible moves I can make) the
- **Minimum** (over all possible moves Min can make) of the
- **Expected value** (over all possible successor states) of the resulting utility:

$$v(s) = \max_{a} \sum_{s'} P(S_{t+1} = s' | S_t = s, a) v(s')$$

$$v(s') = \min_{a'} \sum_{s''} P(S_{t+2} = s'' | S_{t+1} = s', a') v(s'')$$

Expectiminimax: notation

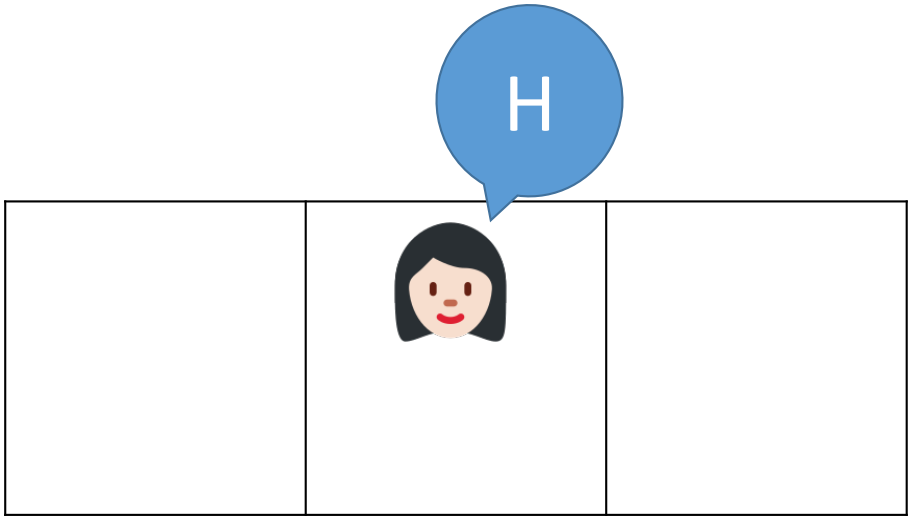
= MAX node. $v(s) = \max_{a \in A(s)} q(s,a)$

= MIN node. $v(s) = \min_{a \in A(s)} q(s,a)$

= Chance node. $q(s,a) = \sum_{s'} P(s'|s,a)v(s')$

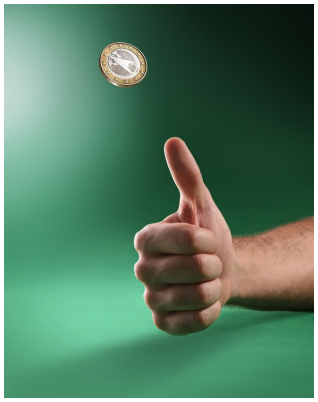
Expectiminimax example

- MIN: Min decides whether to count heads (action H) or tails (action T) as a forward movement.

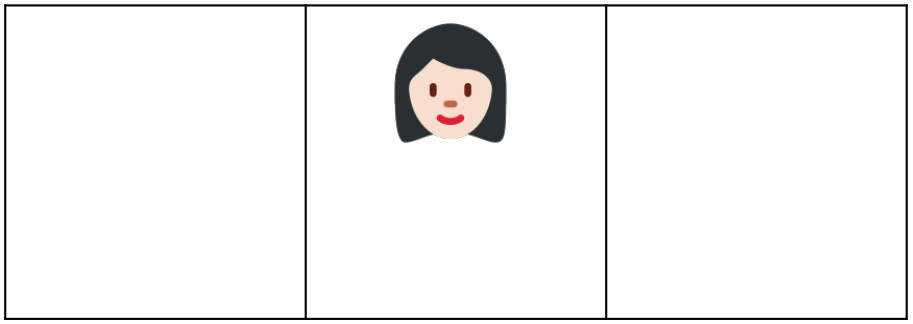


Expectiminimax example

- MIN: Min decides whether to count heads (action H) or tails (action T) as a forward movement.
- Chance: she flips a coin and moves her game piece in the direction indicated.



By ICMA Photos - Coin Toss, CC BY-SA 2.0, https://commons.wikimedia.org/w/index.php?curid=71147286

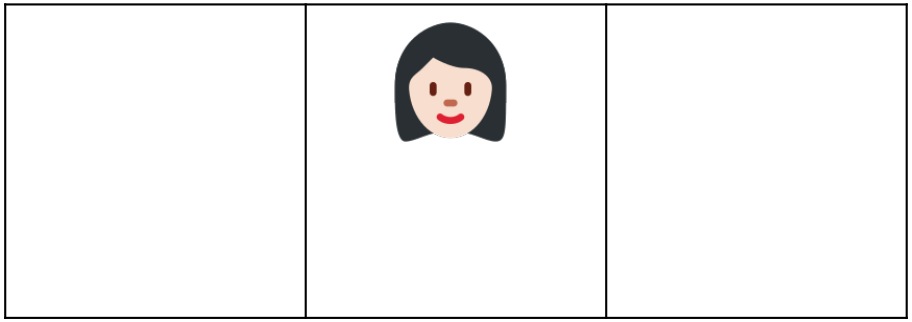


Emojis by Twitter, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=59974366

Expectiminimax example

- MIN: Min decides whether to count heads (action H) or tails (action T) as a forward movement.
- Chance: she flips a coin and moves her game piece in the direction indicated.

By NJR ZA - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=4228918

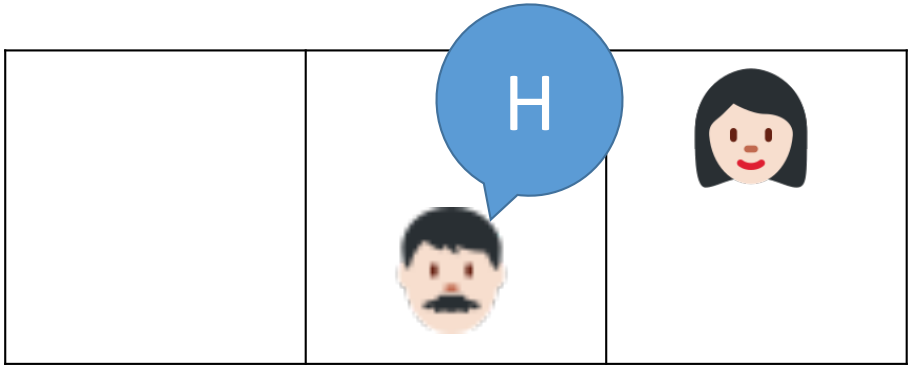


Emojis by Twitter, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=59974366

Expectiminimax example

- MIN: Min decides whether to count heads (action H) or tails (action T) as a forward movement.
- Chance: she flips a coin and moves her game piece in the direction indicated.
- MAX: Max decides whether to count heads (action H) or tails (action T) as a forward movement.

By NJR ZA - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=4228918



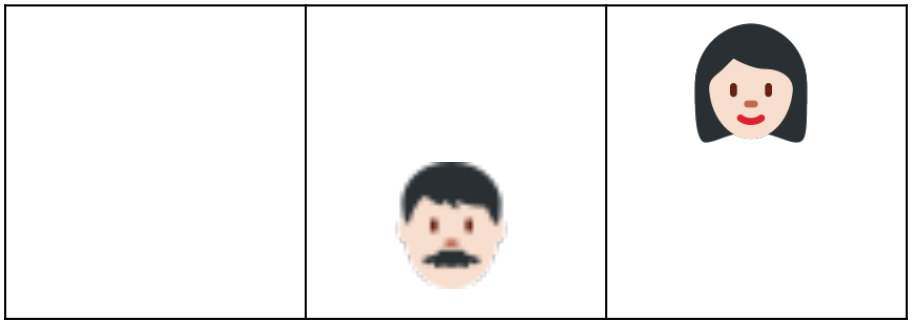
Emojis by Twitter, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=59974366

Expectiminimax example

- MIN: Min decides whether to count heads (action H) or tails (action T) as a forward movement.
- Chance: she flips a coin and moves her game piece in the direction indicated.
- MAX: Max decides whether to count heads (action H) or tails (action T) as a forward movement.
- Chance: he flips a coin and moves his game piece in the direction indicated.

By NJR ZA - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=4228918

By NJR ZA - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=4228918



Emojis by Twitter, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=59974366

Expectiminimax example

- MIN: Min decides whether to count heads (action H) or tails (action T) as a forward movement.
- Chance: she flips a coin and moves her game piece in the direction indicated.
- MAX: Max decides whether to count heads (action H) or tails (action T) as a forward movement.
- Chance: he flips a coin and moves his game piece in the direction indicated.

Reward: $2 to the winner, $0 for a draw.

By NJR ZA - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=4228918

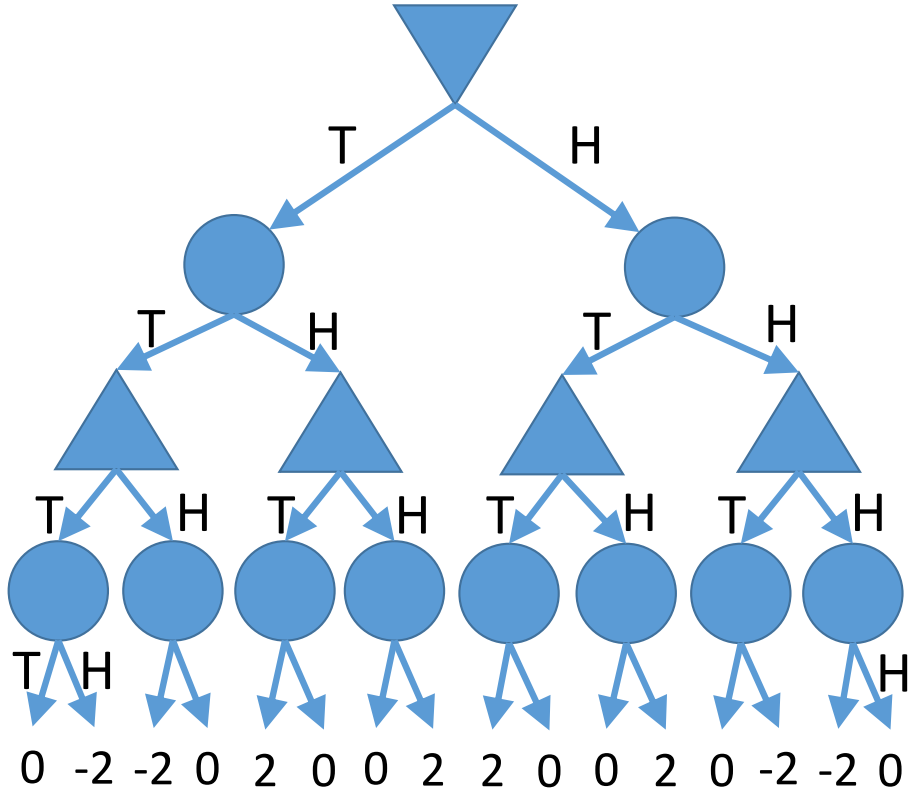
By NJR ZA - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=4228918

Emojis by Twitter, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=59974366.
$2 By Bureau of Engraving and Printing: U.S. Department of the Treasury - own scanned, Public Domain, https://commons.wikimedia.org/w/index.php?curid=56299470

Expectiminimax example

- MIN: Min decides whether to count heads (action H) or tails (action T) as a forward movement.
- Chance: she flips a coin and moves her game piece in the direction indicated.
- MAX: Max decides whether to count heads (action H) or tails (action T) as a forward movement.
- Chance: he flips a coin and moves his game piece in the direction indicated.

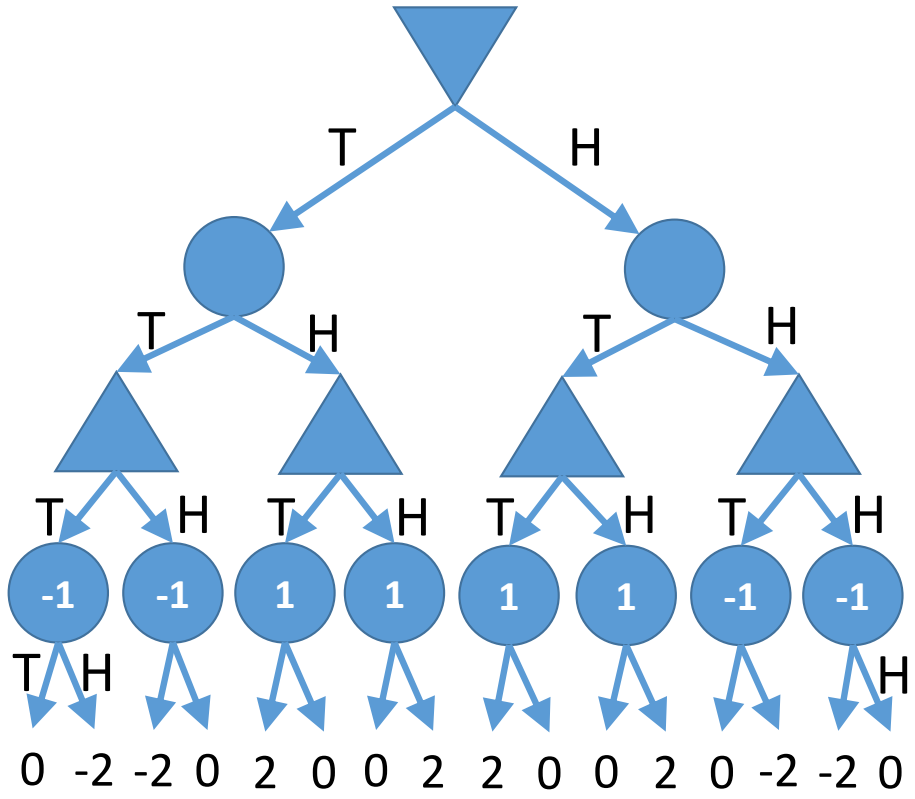
Reward: $2 to the winner, $0 for a draw.



Expectiminimax example

Chance node:

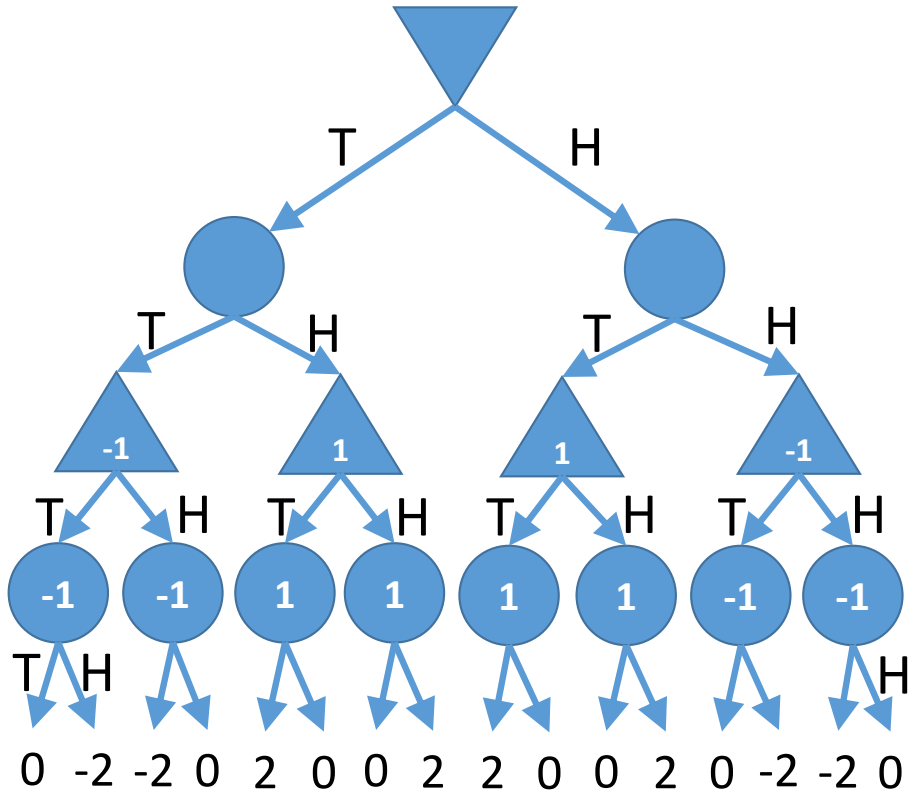
$$q(s,a) = \sum_{s'} P(s'|s,a)v(s')$$



Expectiminimax example

Max node:

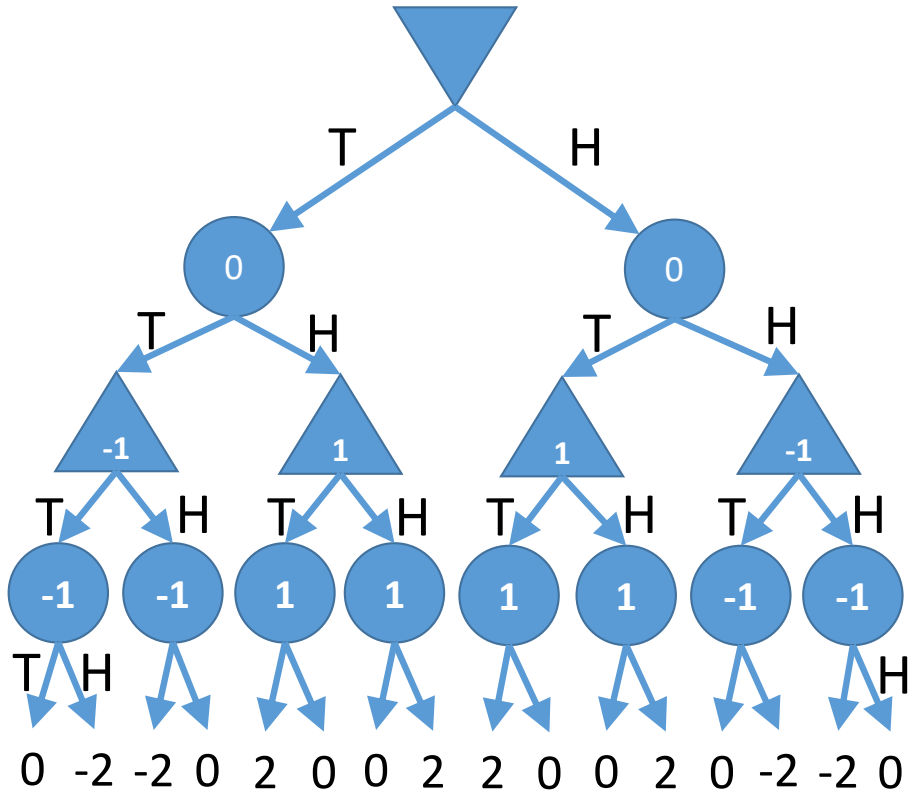
$$v(s) = \max_{a \in A(s)} q(s, a)$$



Expectiminimax example

Chance node:

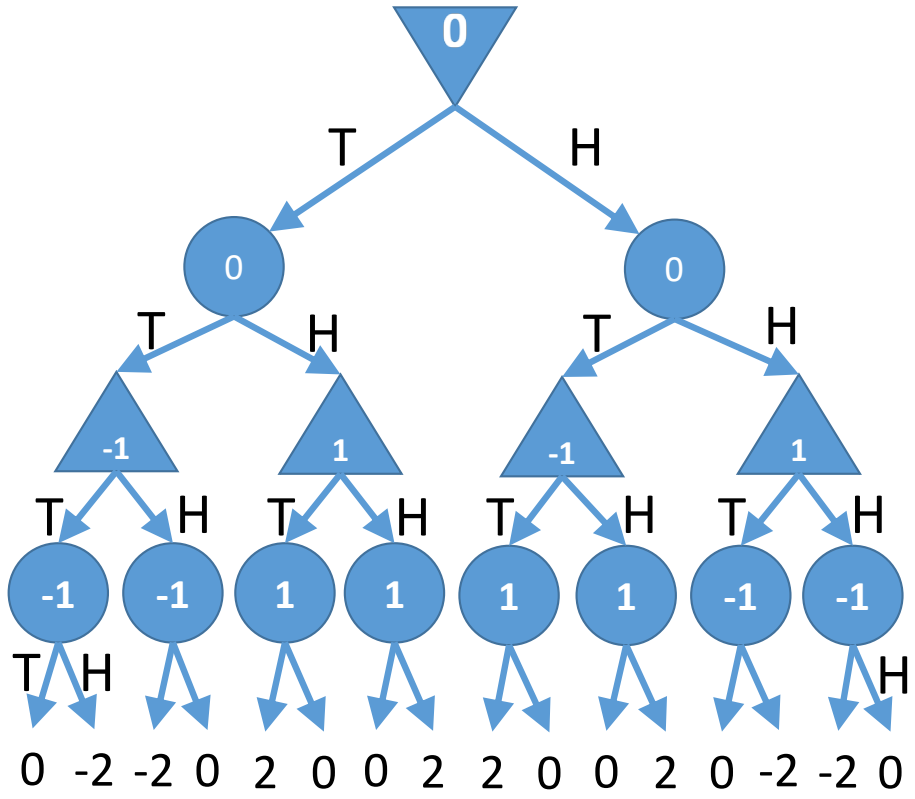
$$q(s,a) = \sum_{s'} P(s'|s,a)v(s')$$



Expectiminimax example

Min node:

$$v(s) = \min_{a \in A(s)} q(s, a)$$



Try the quiz!

Try the quiz!

Outline

- Review: Minimax
- Monte Carlo search: Estimate value by playing randomly
- Expectiminimax: Optimal search if transitions are random
- Relationship of expectiminimax to MDP

Relationship of expectiminimax to MDP

Remember that the solution to a Markov decision process is the policy that maximizes the Bellman equation, which we could call the expectimax equation:

$$u(s) = \max_a \sum_{s\prime} P(S_{t+1} = s' | S_t = s, a) u(s')$$

Expectiminimax maximizes a generalized Bellman equation:

$$v(s) = \begin{cases} \max_a \sum_{s\prime} P(S_{t+1} = s' | S_t = s, a) v(s') & s \text{ is a max state} \\ \min_a \sum_{s\prime} P(S_{t+1} = s' | S_t = s, a) v(s') & s \text{ is a min state} \end{cases}$$

These are the same operation. They are both NP-complete and have similar approximate solution methods, e.g., value iteration, policy iteration, etc.

Summary

- Monte Carlo search:

$$v(s) = \frac{(\#\text{ times Max wins})}{N}$$

- Expectiminimax:

$$v(s) = \max_a \sum_{s'} P(S_{t+1} = s' | S_t = s, a) v(s')$$

$$v(s') = \min_{a'} \sum_{s''} P(S_{t+2} = s'' | S_{t+1} = s', a') v(s'')$$