

Expectimax search

Uncertain outcomes

- Single agent search tree assumed that you are the only agent taking actions and their result is predicted by the transition function $T(s, a) \rightarrow s'$
- Minmax assumed that there is also an opponent taking actions
 - But, in some sense, the zero sum opponent is also predictable
- What if we don't know the result of the action?
 - Inherent randomness in the environment: rolling a dice
 - Unpredictable opponents or bystanders with random behavior
 - Actions can fail or succeed partially (slipping wheels etc)

Probabilistic transition function

- A way to think about this is that the transition function is now probabilistic:
 $T(s, a, s') \in [0, 1]$
- We will talk more about this when we discuss Markov Decision Processes and reinforcement learning

Expectimax search

- We are still trying to compute the V value
- **max** nodes: return the max of successors
- **min** nodes: return the min of the successors
- **expectation** nodes: return the probability weighted average (*expectation*) of children

Reminder: expectation of a random variable

$$\mathbb{E}(f(x)) = \sum_i p(x_i) f(x_i)$$

or, in a continuous case:

$$\mathbb{E}(f(x)) = \int p(x) f(x) dx$$

Expectation of time to get to the airport:

- Drive time: 20 clear weather, 40 min in rain, 60 min in snow
- Likelihood of clear weather 80%, rain 15%, snow 5%
- Expectation: $20 * 0.8 + 40 * 0.15 + 60 * 0.05 = 25$

Backgammon (or other dice-based zero-sum games)

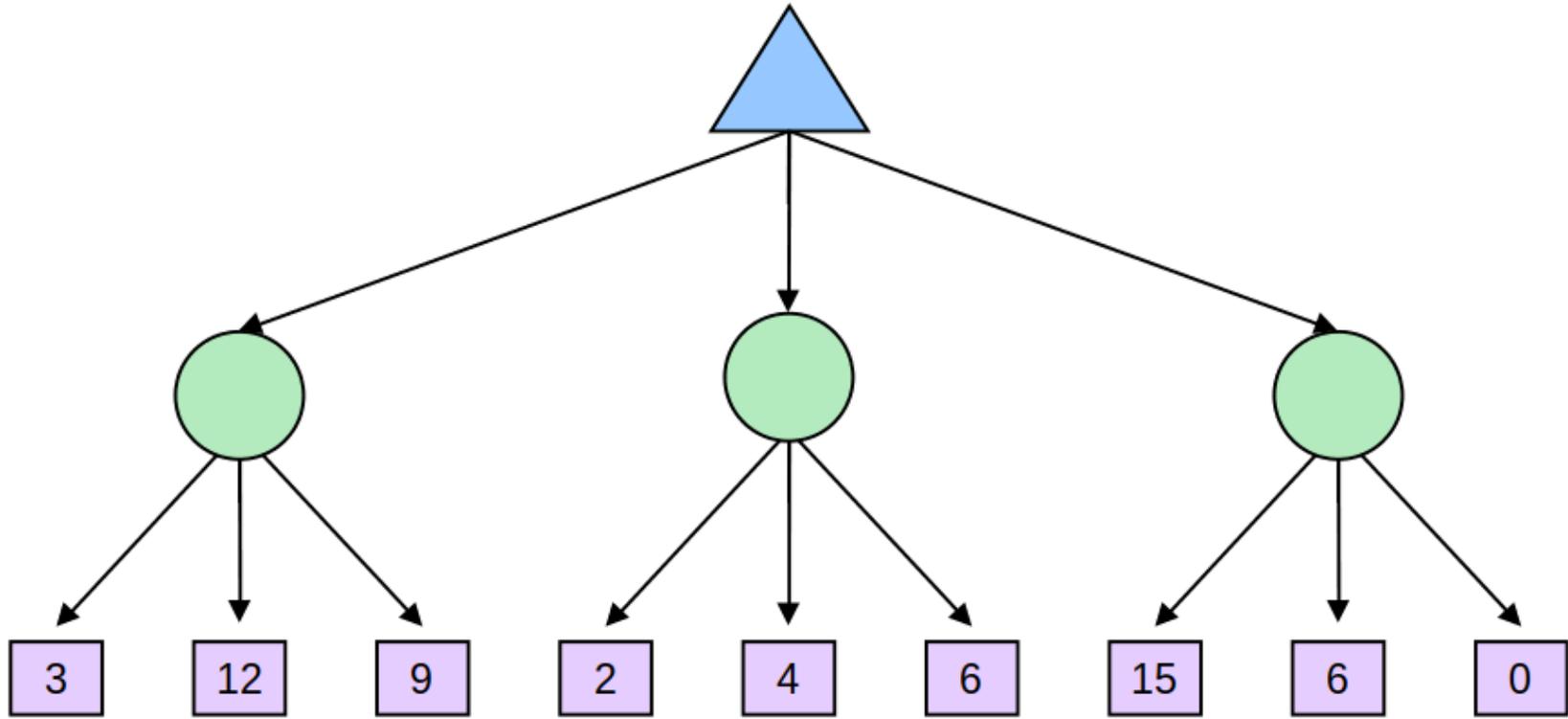
- Max node (move by ego, after knowing dice)
- Expect node (dice by opponent)
- Min node (move by opponent, after knowing dice)
- Expect node (dice by ego)
- Max node (move by ego, after knowing dice)... and so on

Expectimax (and other variants) pseudocode

```
def value(state):
    if state is TERMINAL: return value
    if state is MAX: return maxvalue(state)
    if state is MIN: return minvalue(state)
    if state is EXPECT: return expvalue(state)

def maxvalue(s)
    v = -∞
    for s' in succ(s)
        v = min (v, maxvalue(s'))
    return v

def expvalue(s)
    v = 0
    for s' in succ(s)
        v = v + probability(s') * value(s')
    return v
```



Expectimax pruning

- Can we prune expectimax?
- Problem: expectation can go both up and down with new nodes!
 - You need all the subnodes to calculate the expectation
- Heuristics:
 - prune branches if their contribution to the expectation is small enough to be negligible (e.g. they are unlikely)
 - prune branches if you predict their values as being below a threshold

Depth limited expectimax

- Expectimax nodes can really blow up the computation time, because you need to evaluate everything below
- It is useless to make long plans when they depend on repeated dice throws to come out just so:
 - I will throw an 8 and move like this, then my opponent will throw a 4 and move like that, then I will throw an 11...
- Game programs for games with significant random component:
 - Think ahead only 1..4 plies
 - Use a **very good** evaluation function

Where do we get the probabilities from?

- In expectimax search, we need to know the probabilities of outcomes
 - Sometimes it is some uniform or near randomness (eg. dice)
 - Sometimes it is a small uncertainty on a positive or negative action.
- Where do we get the probabilities? - The **model**
 - Sometimes it is simple - eg. dice roll
 - Sometimes it is very complex
- We will revisit this later

Informed probabilities

- Expectimax can also handle situations where you try to model an imperfect opponent:
- Let us say that the opponent is doing the perfect minmax move 90% of the time, but moves randomly 10% of the time
 - This is an expectation node. But you don't know the probability of the moves ahead of time, you need to calculate it!
- You need to run a **simulation of your opponent**, with the opponent **simulating you**
 - This is very expensive for expectimax
 - It is much cheaper for minmax, because the two simulations are folded into the same tree.

Mixed layers

- Different layers (max/min/expectations) can be mixed randomly.
- Often, we consider the environment an additional "random" player.
- Each node computes the appropriate combination of its children.

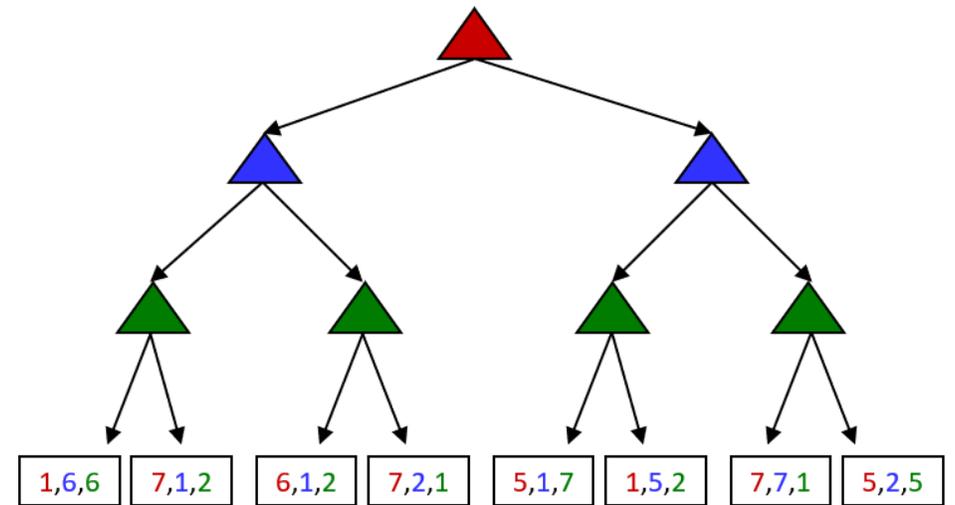
Example: Backgammon

- Dice rolls increase b : 21 possible rolls with 2 dice
 - Backgammon ≈ 20 legal moves
 - Depth 2 = $20 \times (21 \times 20)^3 = 1.2 \times 10^9$
- As depth increases, probability of reaching a given search node shrinks
 - So usefulness of search is diminished
 - So limiting depth is less damaging
 - But pruning is trickier...
- Historic AI: TDGammon uses depth-2 search + very good evaluation function + reinforcement learning: world-champion level play
- 1st AI world champion in any game!



Multi-agent games

- What if the game is not zero sum, or has multiple players?
- Node values are not tuples of utility
- Each player maximizes its own utility
- Emergent cooperation and competition



Utilities and rationality

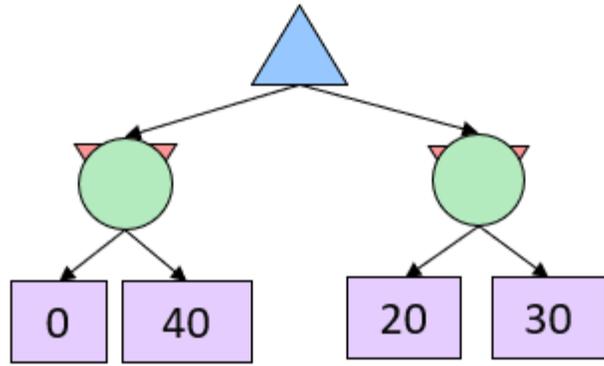
- We kept talking about **utility** but did not yet give a good definition what it is
 - Let us say that the values associated with the terminal outcomes are the utility
 - Agents pursue higher utility, so these values determine the behavior of the agent
- We say that a **rational agent** is one that chooses actions that **maximize its expected utility**, given its knowledge

Rationality

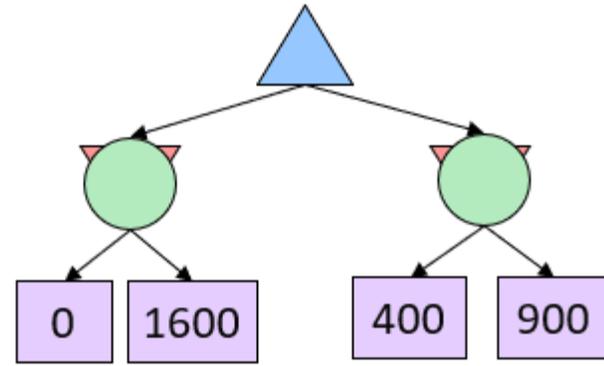
- So we defined rationality as utility maximization
 - You can also consider it as a definition of the kind of agent intelligence we are considering in this class
- The choice of utility function is up to the agent
 - power, art, scientific discovery
 - pleasure, procreation, public respect
 - number of followers, winning an election, winning a sport competition
 - a weighted sum of these or a preference ordered list of these
- There are a lot of choices, but not every possible choice allows a rational agent.

"Expected"?

- For minmax reasoning, utility is simple. The greater the utility outcome, the better.
 - So only **ordering** matters.
 - "insensitivity to monotonic transformations"
- But for expectimax, we take expectations, so suddenly the absolute values matter as well.
 - We should define utilities such that we can add and average them!



x^2



Utilities and preferences

- We define utilities as functions on states $U(s) \in \mathbb{R}$
- We will say that the utilities describe the **preferences of the agents**
 - It is a way to summarize the goals of the agent
- Two strategies to build an intelligent agent
 - **Behavior specification:** Describe its behavior for each state - i.e. write the $\pi(s) \rightarrow a$ function directly.
 - **Utility specification:** Provide a utility function $U(s)$. A rational agent will choose its own actions in the pursuit of the goal of maximizing expected utility.

Problems

- Problems with behavior specification:
 - **Framing problem:** need to handle a very large number of cases
 - Buy milk, unless you already have it, it is 2am, it is a hurricane, it is a zombie attack, ...
 - The relationship between the behavior and good outcomes difficult to prove.
- Problems with utility specification:
 - Where does the utility coming from?
 - Can every rational behavior be expressed as utilities?
 - **Theorem:** any rational preferences over states can be summarized as a utility function.

Preferences

- Prizes: A, B
- Lotteries: situations with uncertain prizes: $L = [p, A; (1 - p)B]$
- The agent prefers A denoted by $A \succ B$
 - A good way to think about it is that the agent would pay at least \$0.01 to get A instead of B
- The agent is **indiferent** denoted by $A \sim B$

Rational preferences

- What kind of preferences can be considered rational?
- Let us imagine an agent with $A \succ B, B \succ C, C \succ A$.
- Such an agent can be induced to give away all his money!
 - This happens because the preferences are not transitive
 - "Dutch book" auctions in horse races.

A set of axioms that ensure rationality

- Orderability

$$(A \succ B) \vee (B \succ A) \vee (A \sim B)$$

- Transitivity

$$(A \succ B) \wedge (B \succ A) \Rightarrow (A \succ C)$$

- Continuity

$$A \succ B \succ C \Rightarrow \exists p [p, A; 1 - p, C] \sim B$$

- Substitutability

$$A \sim B \Rightarrow [p, A; 1 - p, C] [p, B; 1 - p, C]$$

- Monotonicity

$$A \succ B \Rightarrow (p \geq q \Leftrightarrow [p, A; 1 - p, B] \succeq [q, A; 1 - q, B])$$

Maximum expected utility principle

- Given preferences satisfying the axioms, there exists a utility function $U(s)$ such that

$$U(A) \geq U(B) \Leftrightarrow A \succeq B$$

$$U([p_1, S_1; \dots; p_n, S_n]) = \sum_i p_i U(S_i)$$

(Ramsey 1931, von Neumann & Morgenstern 1944)

Utilities and humans

- We can try to **elicit** human preferences by presenting humans with lotteries
 - Most studies concluded that *humans are not rational*
 - Psychologists had a field day with this!
- Some of it might be that we have limited computing power →
 - "Bounded rationality", "Satisficing" actions
 - Herbert Simon 1978 Nobel Prize in Economics
- But some of it would be also be that we do not restrict our thinking at the specific setting of the problem
 - i.e. only two choices with no other implications, no repeated games and no temporal setting

Examples of utility calculations in human affairs

- **Micromorts:** 10^{-6} chance of death
 - How much are you willing to pay for a 7th airbag in your car?
- **QALY:** quality adjusted life-years, useful for medical decisions
 - Who gets the heart transplant etc.

Money

- You can calculate an expected monetary value (EMV) of a transaction by calculating the expectation of probabilities
- But money does **not** behave as a utility function
- Most people are **risk averse**
 - A decrease in money by X triggers a greater utility change than the same increase
- When deep in debt, people are **risk prone**

Insurance

- How much are you willing to buy this lottery: [0.5, \$10000; 0.5, \$0]
 - i.e. the **certainty equivalent**
- The difference between the certainty equivalent and the EVM is the **insurance premium**
- Why does this work out for the insurance company?
 - They have a different utility curve (more rational)
 - They average over a different lotteries.

Utilities: building agents and robots

- Note that we can build a perfectly rational agent by behavior specification, without ever representing utilities.
- Historically, it had been difficult to build agents by utility specification
 - But this is changing, as we are moving towards more ML and less hardcoded behaviors
- How do you specify the utilities for a self-driving car?
 - Traffic rules?
 - Optimize time to goal, energy consumption?
 - Safety?

Utilities and AGI

- What should be the utilities of an artificial general intelligence?
- **Alignment problem:** the AGI should share preferences with humanity / smart humans / important humans / **me!**
- Couple of issues:
 - **Can** we specify the utility of humans?
 - Who gets to specify it? Likely differs from person to person.
 - Are we happy with the human utility function? Eg. pleasure seeking behavior?
 - Wouldn't we better of just placing limits on actions? Eg. Asimov's three laws of robotics.
 - Specification gaming
 - Many others...