Announcements

- Project 1.3 is up (Multi-Agent Pacman)
  - Due: 9/29
- Midterm: 10/10, in class
  - One page cheat sheet allowed
  - You must generate it yourself (no lecture slides, etc)
  - Basic calculators will be allowed, but not needed
- Pacman contest after midterm

Recap: Minimax

- In minimax search:
  - Max plans assuming that Min will act according to a minimax calculation
  - Min’s calculation depends on Min’s belief that Max is a minimax player, and so on
  - Max and Min’s thought processes interweave and we can sort the whole process out in one tree
- What if instead we think the opponent is random?
  - Or, smart but not minimax?

Expectimax Search

- What if we don’t know what the result of an action will be? E.g.,
  - In solitaire, shuffle is unknown
  - In minesweeper, don’t know where the mines are
  - In pacman, the ghosts act randomly
- Can do expectimax search
  - Chance nodes, like min’s actions except the outcome is not known
  - Calculate expected utility for nodes
  - Max nodes as in minimax search
  - Chance nodes take average (expectation) of value of children
- Later, we’ll learn how to formalize this as a Markov Decision Process

Expectimax Pseudocode

```python
def value(s):
    if s is a max node return maxValue(s)
    if s is an exp node return expValue(s)
    if s is a terminal node return evaluation(s)

def maxValue(s):
    values = [value(s') for s' in successors(s)]
    return max(values)

def expValue(s):
    values = [value(s') for s' in successors(s)]
    weights = [probability(s', s) for s' in successors(s)]
    return expectation(values, weights)
```
Maximum Expected Utility

- MEU: An agent should choose the action which maximizes its expected utility, given its knowledge.
- General principle for decision making.
- Often taken as the definition of rationality.
- We’ll see this idea over and over in this course.
- Let’s decompress this definition…

Reminder: Probabilities

- A random variable represents an event whose outcome is unknown.
- A probability distribution is an assignment of numbers to outcomes.
- Example: traffic on freeway?
  - Random variable: let T be whether there’s traffic.
  - Outcomes: (none, light, heavy).
  - Distribution: P(T=none) = 0.25, P(T=light) = 0.55, P(T=heavy) = 0.20.
- Some laws of probability (more later):
  - Probabilities are always non-negative.
  - Probabilities over all possible outcomes sum to one.
- As we get more evidence, probabilities may change:
  - P(T=heavy) = 1/5, P(T=heavy | Hour=8am) = 3/5.
  - We’ll talk about methods for reasoning and updating probabilities soon.

What Are Probabilities?

- Objectivist / frequentist answer:
  - Averages over repeated experiments.
  - E.g. empirically estimating P(rain) from historical observation.
  - Assertion about how future experiments will go (in the limit).
  - New evidence changes the reference class.
  - Makes one think of inherently random events, like rolling dice.
- Subjectivist / Bayesian answer:
  - Degrees of belief about unobserved variables.
  - E.g. an agent’s belief that it’s raining, given the temperature.
  - E.g. pacman’s belief that the ghost will turn left, given the state.
  - Often estimate probabilities from past experience (more later).
  - New evidence updates beliefs (more later).

Probabilities Everywhere

- Not just for games of chance!
  - I’m snuffling: am I sick?
  - Email contains “FREE!”: is it spam?
  - Tooth hurts: have cavity?
  - Safe to cross street?
  - 60 min enough to get to the airport?
  - Robot rotated wheel three times, how far did it advance?
- Why can a random variable have uncertainty?
  - Inherently random process (dice, etc).
  - Insufficient or weak evidence.
  - Unmodeled variables.
  - Ignorance of underlying processes.
  - The world’s just noisy!
- Compare to fuzzy logic, which has degrees of truth, or rather than just degrees of belief.

Reminder: Expectations

- Often a quantity of interest depends on a random variable.
- The expected value of a function is the average output, weighted by some distribution over inputs.
- Example: How late will I be?
  - Lateness is a function of traffic:
    \[ L(T=\text{none}) = -10, \quad L(T=\text{light}) = -5, \quad L(T=\text{heavy}) = 15. \]
  - What is my expected lateness?
    - Need to specify some belief over T to weight the outcomes.
    - Say P(T) = (none: 2/5, light: 2/5, heavy: 1/5).
    - The expected lateness:
      \[ \mathbb{E}_{P(T)}[L(T)] = \frac{2}{5} \times (-10) + \frac{2}{5} \times (-5) + \frac{1}{5} \times (15). \]

Expectations

- Real valued functions of random variables:
  \[ f : X \rightarrow R. \]
- Expectation of a function a random variable:
  \[ \mathbb{E}_{X}[f(X)] = \sum_{x} f(x) P(x). \]
- Example: Expected value of a fair die roll:
  \[ 1 \times \frac{1}{6} + 2 \times \frac{1}{6} + 3 \times \frac{1}{6} + 4 \times \frac{1}{6} + 5 \times \frac{1}{6} + 6 \times \frac{1}{6} = \frac{21}{6} = 3.5. \]
Utilities

- Utilities are functions from outcomes (states of the world) to real numbers that describe an agent's preferences.
- Where do utilities come from?
  - In a game, may be simple (+1/-1)
  - Utilities summarize the agent's goals
- Theorem: any set of preferences between outcomes can be summarized as a utility function (provided the preferences meet certain conditions)
- In general, we hard-wire utilities and let actions emerge (why don't we let agents decide their own utilities?)
- More on utilities in a few lectures…

Expectimax Search

- In expectimax search, we have a probabilistic model of how the opponent (or environment) will act
  - Model could be a simple uniform distribution (roll a die)
  - Model could be sophisticated and require a great deal of computation
  - We have a node for every outcome out of our control: opponent or environment
  - The model can predict that smart actions are likely
  - For now, assume we are given \( P(A|S) \), a function which, for any state \( s \) maps states actions to probabilities: \( P(A=a|S=s) \).

Expectimax for Pacman

- Notice that we've gotten away from thinking that the ghosts are trying to minimize pacman's score
- Instead, they are now a part of the environment
- We assume they act according to our belief distribution
- Quiz: Can we see minimax as a special case of expectimax?
- Quiz: what would pacman's computation look like if we assumed that the ghosts were doing 1-ply minimax and taking the result 80% of the time, otherwise moving randomly?
- If you take this further, you end up calculating belief distributions over your opponents' belief distributions over your belief distributions, etc…
  - Can get unmanageable very quickly!

α-β Pruning Example

α-β Pruning

- General configuration
  - \( \alpha \) is the best value the Player can get at any choice point along the current path
  - If \( n \) is worse than \( \alpha \), \( \text{MAX} \) will avoid it, so prune \( n \)'s branch
  - Define \( \beta \) similarly for \( \text{MIN} \)
Expectimax Evaluation

- For minimax search, evaluation function insensitive to monotonic transformations
  - We just want better states to have higher evaluations (get the ordering right)

- For expectimax, we need the scales to be meaningful as well
  - We need to know whether 50/50 chances of A or B is better than C
  - 100 or -10 vs 0 is different then 10 or -100 vs 0

What Makes a Good Player?

- Two main ways to make a player good
  - Deep search (pruning, move ordering, etc)
  - Good evaluation functions

- How to make good evaluation functions?
  - Inventing features brings domain knowledge about what aspects of the game state are important (i.e. food good, ghost bad, pellets better when ghosts are near)
  - But, balancing the weights on these features is quite hard to do with intuition
  - Better to let the agent play and tune based on experience
  - Good application for machine learning (next time!)

Mixed Layer Types

- E.g. backgammon
- Expectiminimax
  - Environment is an extra player that moves after each agent
  - Chance nodes take expectations, otherwise like minimax

\[
\text{if state is a MAX node then} \\
\quad \text{return the highest \text{EXPECTIMINMAX-VALUE of Successors(state)}} \\
\text{if state is a MIN node then} \\
\quad \text{return the lowest \text{EXPECTIMINMAX-VALUE of Successors(state)}} \\
\text{if state is a chance node then} \\
\quad \text{return average of \text{EXPECTIMINMAX-VALUE of Successors(state)}}
\]

Stochastic Two-Player

- Dice rolls increase $b$: 21 possible rolls with 2 dice
- Backgammon = 20 legal moves
- Depth $4 = 20 \times (21 \times 20)^3 \approx 1.2 \times 10^9$
- As depth increases, probability of reaching a given node shrinks
  - So value of lookahead is diminished
  - So limiting depth is less damaging
  - But pruning is less possible…
- TDGammon uses depth-2 search + very good eval function + reinforcement learning: world-champion level play

Non-Zero-Sum Games

- Similar to minimax:
  - Utilities are now tuples
  - Each player maximizes their own entry at each node
  - Propagate (or back up) nodes from children
  - Can give rise to cooperation and competition dynamically…