Announcements

- Written Assignment 1:
  - Due at the end of lecture
  - If you haven’t done it, but still want some points, come talk to me after class

- Project 1:
  - Most of you did very well
  - We promise not to steal your slip days
  - Come to office hours if you didn’t finish & want help

- Project 2:
  - Due a week from tomorrow (Wednesday)
  - Want a partner? Come to the front after lecture

Today

- Mini-contest 1 results
- Pruning game trees
- Chance in game trees

Mini-Contest Winners

- Problem: eat all the food in bigSearch
- Challenge: finding a provably optimal path is very difficult
- Winning solutions (baseline is 350):
  - 5th: Greedy hill-climbing, Jeremy Cowles: 314
  - 4th: Local choices, Jon Hirschberg and Nam Do: 292
  - 3rd: Local choices, Richard Guo and Shendy Kurnia: 290
  - 2nd: Local choices, Tim Swift: 286
  - 1st: A* with inadmissible heuristic, Nikita Mikhaylin: 284

GamesCrafters

http://gamescrafters.berkeley.edu/

Adversarial Games

- Deterministic, zero-sum games:
  - tic-tac-toe, chess, checkers
  - One player maximizes result
  - The other minimizes result

- Minimax search:
  - A state-space search tree
  - Players alternate turns
  - Each node has a minimax value: best achievable utility against a rational adversary
Computing Minimax Values

- Two recursive functions:
  - `max-value` maxes the values of successors
  - `min-value` mins the values of successors

```python
def value(state):
    if the state is a terminal state: return the state's utility
    if the next agent is MAX: return max-value(state)
    if the next agent is MIN: return min-value(state)

def max-value(state):
    Initialize max = -∞
    For each successor of state:
        Compute value(successor)
        Update max accordingly
    Return max
```

Pruning in Minimax Search

Alpha-Beta Pruning

- General configuration
  - `a` is the best value that MAX can get at any choice point along the current path
  - If `n` becomes worse than `a`, MAX will avoid it, so can stop considering `n`'s other children
  - Define `b` similarly for MIN

Alpha-Beta Pseudocode

```python
function MAX-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v = -∞
    for a ∈ Successors(state) do
        temp = MAX(a, MIN-VALUE(state))
        if temp > v then return v
    return v

function MIN-VALUE(state, a) returns a utility value
    inputs: state, current state is game
            a, the value of the best alternative for MAX along the path to state
    if TERMINAL-TEST(state) then return UTILITY(state)
    v = +∞
    for a ∈ Successors(state) do
        temp = MIN(a, MAX-VALUE(state, a))
        if temp < v then return v
    return v
```

Alpha-Beta Pruning Example

- This pruning has no effect on final result at the root
- Values of intermediate nodes might be wrong!
- Good move ordering improves effectiveness of pruning
- With "perfect ordering":
  - Time complexity drops to O(b^m/2)
  - Doubles solvable depth
  - Full search of, e.g. chess, is still hopeless!
- This is a simple example of metareasoning, and the only one you need to know in detail
Expectimax Search Trees

- What if we don’t know what the result of an action will be? E.g.,
  - In solitaire, next card is unknown
  - In monopoly, the dice are random
  - In pacman, the ghosts act randomly
- We can do expectimax search
  - Chance nodes are like min nodes, except the outcome is uncertain
  - Calculate expected utilities
  - Max nodes as in minimax search
  - Chance nodes take average (expectation) of value of children
- Later, we’ll learn how to formalize the underlying problem as a Markov Decision Process

Maximum Expected Utility

- Why should we average utilities? Why not minimax?
- Principle of maximum expected utility: 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 weights to outcomes
- Example: traffic on freeway?
  - Random variable: \( T \) = how much traffic is there
  - Outcomes: \( T \) in \{none, light, heavy\}
  - Distribution: \( P(T=\text{none}) = 0.25, P(T=\text{light}) = 0.5, P(T=\text{heavy}) = 0.25 \)
  - Common abbreviation: \( P(\text{light}) = 0.5 \)
- 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:
  - \( P(T=\text{heavy}) = 0.25, P(T=\text{heavy} | \text{Hour}=8 \text{am}) = 0.60 \)
  - We’ll talk about methods for reasoning and updating probabilities later

What are Probabilities?

- Objectivist / frequentist answer:
  - Averages over repeated experiments
  - E.g. empirically estimating \( P(\text{rain}) \) 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 learn probabilities from past experiences (more later)
  - New evidence updates beliefs (more later)

Uncertainty Everywhere

- Not just for games of chance!
  - I’m sick: will I sneeze this minute?
  - Email contains “FREE!”: is it spam?
  - Tooth hurts: have cavity?
  - 60 min enough to get to the airport?
  - Robot rotated wheel three times, how far did it advance?
  - Safe to cross street? (Look both ways!)
- Sources of uncertainty in random variables:
  - Inherently random process (dice, etc)
  - Insufficient or weak evidence
  - Ignorance of underlying processes
  - Unmodeled variables
  - The world’s just noisy — it doesn’t behave according to plan!
- Compare to fuzzy logic, which has degrees of truth, rather than just degrees of belief!

Reminder: Expectations

- We can define function \( f(X) \) of a random variable \( X \)
- The expected value of a function is its average value, weighted by the probability distribution over inputs
- Example: How long to get to the airport?
  - Length of driving time as a function of traffic: \( L(\text{none}) = 20, L(\text{light}) = 30, L(\text{heavy}) = 60 \)
  - What is my expected driving time?
    - Notation: \( E[L(T)] \)
    - Remember, \( P(T) = \{\text{none}:0.25, \text{light}:0.5, \text{heavy}:0.25\} \)
    - \( E[L(T)] = L(\text{none}) \cdot P(\text{none}) + L(\text{light}) \cdot P(\text{light}) + L(\text{heavy}) \cdot P(\text{heavy}) \)
    - \( E[L(T)] = (20 \cdot 0.25) + (30 \cdot 0.5) + (60 \cdot 0.25) = 35 \)
Expectimax Search

- In expectimax search, we have a probabilistic model of how the opponent (or environment) will behave in any state:
  - 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 might say that adversarial actions are likely!
- For now, assume for any state we have a distribution to assign probabilities to opponent actions / environment outcomes

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)
```

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
- Pacman has a belief (distribution) over how they will act
- Questions:
  - Is minimax a special case of expectimax?
  - What happens if we think ghosts move randomly, but they really do try to minimize Pacman's score?

Expectimax for Pacman

<table>
<thead>
<tr>
<th></th>
<th>Minimax Ghost</th>
<th>Random Ghost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pacman Won</td>
<td></td>
<td>Won 5/5</td>
</tr>
<tr>
<td>Avg. Score</td>
<td>493</td>
<td>483</td>
</tr>
<tr>
<td>Expectimax</td>
<td></td>
<td>Won 1/5</td>
</tr>
<tr>
<td>Pacman Won</td>
<td></td>
<td>Won 5/5</td>
</tr>
<tr>
<td>Avg. Score</td>
<td>-303</td>
<td>503</td>
</tr>
</tbody>
</table>

Pacman used depth 4 search with an eval function that avoids trouble
Ghost used depth 2 search with an eval function that seeks Pacman

Expectimax Pruning?

- For minimax search, evaluation function scale doesn’t matter
  - We just want better states to have higher evaluations (get the ordering right)
  - We call this property insensitivity to monotonic transformations
- For expectimax, we need the magnitudes to be meaningful as well
  - E.g. must know whether a 50% / 50% lottery between A and B is better than 100% chance of C
  - 100 or -10 vs 0 is different than 10 or -100 vs 0

Expectimax Evaluation
Mixed Layer Types

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

```python
if state is a MAX node then
    return the highest ExpectiMinimax-Value of Successors(state)
if state is a MIN node then
    return the lowest ExpectiMinimax-Value of Successors(state)
if state is a chance node then
    return average of ExpectiMinimax-Value of Successors(state)
```

Stochastic Two-Player

- Dice rolls increase b: 21 possible rolls with 2 dice
- Backgammon = 20 legal moves
- Depth 4 = 20 x (21 x 20)^4 = 1.2 x 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...