Expectimax Search

CSC384 – Introduction to Artificial Intelligence

Some slides borrowed/adapted UC Berkeley CS188 Intro to AI
Worst-Case vs. Average Case

Idea: If the opponent is not rational, may be worth it taking risk!
Worst-Case vs. Average Case

Expected payoff:
\[
a: 2 \\
b: 5.5
\]

Idea: If the opponent is not rational, may be worth it taking risk!
Worst-Case vs. Average Case

Even worse: with MinMax, “max” player has no incentive to choose b.
Expectimax Search

• If you don’t know what kind of strategy your opponent uses, then Minimax might playing it too safe.
  • Ensures that in the worst case (very smart opponent) you do better
  • But by playing it safe it might lead to much worse outcomes.
• One important generalization is to consider probabilistic opponents, where your opponent chooses its moves by chance.
  • Perhaps it is more likely to pick the best action, but occasionally picks the worst action.
• Also useful when your opponent is “nature”
• Or when there are chance moves in the game, like throwing of dice.

• Now MAX wants to pick a node that maximizes the expected value.

• **Expectimax search**: compute the average score
  • Max nodes as in minimax search
  • Chance nodes are like min nodes but which move will be picked is uncertain
  • At chance nodes we calculate **expected value**
Expectimax Search

Max should pick the child with greatest expected value
Worst-Case vs. Average Case

Idea: Uncertain outcomes controlled by chance, not an adversary!
For Minimax, utilities assigned to terminals have to get the relative ordering of the terminal values right. We can scale them as we want.

For Expectimax, we need both relative order and magnitude of the terminal values to have meaning!
Expectimax Search

• We can develop a simple algorithm for dealing with MAX and chance nodes.
• Each to extend to a combination of MAX/MIN/Chance nodes!
**Expectimax Search**

**Expectimax (pos):**  
Return best move for player(pos)  
and MAX’s value for pos.

```python
best_move = None
if terminal(pos):
    return best_move, utility(pos)
if player(pos) == MAX:    value = -infinity
if player(pos) == CHANCE: value = 0
for move in actions(pos):
    nxt_pos = result(pos, move)
    nxt_val, nxt_move = Expectimax(nxt_pos)
    if player == MAX and value < nxt_val:
        value, best_move = nxt_val, move
    if player == CHANCE:
        value = value + prob(move) * nxt_val
return best_move, value
#no best_move for CHANCE player
```
Practical Matters

• All “real” games are too large to enumerate tree
  • e.g., chess branching factor is roughly 35
  • Depth 10 tree: 2,700,000,000,000,000 nodes
  • Even alpha-beta pruning won’t help here!

• We must limit depth of search tree
  • Can’t expand all the way to terminal nodes
  • We must make heuristic estimates about the values of the non-terminal positions where we terminate the search.
  • These heuristics are often called evaluation function
  • evaluation functions are often learned
Heuristics in Games

• Example for tic tac toe: \( h(n) = [\# \text{ of 3 lengths that are left open for player A}] - [\# \text{ of 3 lengths that are left open for player B}] \).

• Alan Turing’s function for chess: \( h(n) = A(n)/B(n) \) where \( A(n) \) is the sum of the point value for player A’s pieces and \( B(n) \) is the sum for player B.

• Many evaluation functions can be specified as a weighted sum of features:
  \[
  h(n) = w_1*\text{feature}_1(n) + w_2*\text{feature}_2(n) + \ldots w_i*\text{feature}_i(n).
  \]
  • The weights can be learnt (Samuel’s checker player 1949)

• Deep Blue used about 6000 features in its evaluation function.
The Dangers of Optimism and Pessimism

Dangerous Optimism
Assuming chance when the world is adversarial

Dangerous Pessimism
Assuming the worst case when it’s not likely

Credit: UC Berkeley CS188 Intro to AI
### Assumptions vs. Reality

<table>
<thead>
<tr>
<th></th>
<th>(smart, devilish)</th>
<th>(unpredictable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimax Pacman</td>
<td>Won 5/5</td>
<td>Won 5/5</td>
</tr>
<tr>
<td>Avg. Score: 483</td>
<td></td>
<td>Avg. Score: 493</td>
</tr>
<tr>
<td>Expectimax Pacman</td>
<td>Won 1/5</td>
<td>Won 5/5</td>
</tr>
<tr>
<td>Avg. Score: -303</td>
<td></td>
<td>Avg. Score: 503</td>
</tr>
</tbody>
</table>

Results from playing 5 games

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

Credit. UC Berkeley CS188 Intro to AI
Mixed Layer Types

- E.g. Backgammon
- Expectiminimax
  - Environment is an extra “random agent” player that moves after each min/max agent
  - Each node computes the appropriate combination of its children

Credit. UC Berkeley CS188 Intro to AI