Minimax

Lecture 31
CS 312

Objectives
- Win tic-tac-toe using minimax

Using Minimax
- eval() function to evaluate a game state
- You (X) try to maximize eval()
- Opponent (O) tries to minimize eval()
- Like branch and bound, except players have to take turns.

Picking a move
- Assume your opponent will choose the move that leaves a minimum eval().

Picking a move
- 10 is your best move, but...

Picking a move
- If you take the -3 move, you only get stuck with 5.
Ideally...

- Perfect evaluation function.
- Able to explore all of the graph.
- For tic-tac-toe there are
  - 9 nodes at the first level, plus
  - 8*9 nodes at the second level, plus
  - 72*7 nodes at the third level, plus
  - 504*6 nodes at the fourth level, plus
  - etc.

$$\text{Eval() for tic-tac-toe}$$

<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center</td>
<td>10</td>
<td>-10</td>
</tr>
<tr>
<td>Corner</td>
<td>7</td>
<td>-7</td>
</tr>
<tr>
<td>One way to win on next move</td>
<td>20</td>
<td>-20</td>
</tr>
<tr>
<td>Two ways to win on 90 next move</td>
<td>90</td>
<td>-90</td>
</tr>
<tr>
<td>Win</td>
<td>90</td>
<td>-90</td>
</tr>
<tr>
<td>Tie</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

This board evaluates to -3

Suppose X goes first

Where should O go next?

Where should O go next?

As a Tree
Would you like to play a game?

- tic-tac-toe
- computer player will have 2 levels of look ahead
- class will be the human player
- flip to see who goes first.

Where should X go next?

Where should O go next?

Where should O go next?

Things to ponder

- If the computer plays itself, will it always tie?
- What if the opponent uses a different eval function?
- Is there a strategy that always wins for tic tac toe? — no matter what the opponent does?