Announcements

1. Reading and Homework for this week

2. Late policy on homework: 10% per day, unless you ask for extension beforehand

3. Presentation schedule – reminder for upcoming presentations
Game Playing: Adversarial Search
Game tree
From M. T. Jones, *Artificial Intelligence: A Systems Approach*

Current board:
X’s move

...
How many nodes?
Evaluation function $f(n)$ measures “goodness” of board configuration $n$. Assumed to be better estimate as search is deepened (i.e., at lower levels of game tree).

Evaluation function here: “Number of possible wins (rows, columns, diagonals) not blocked by opponent, minus number of possible wins for opponent not blocked by current player.”
**Minimax search**

From M. T. Jones, *Artificial Intelligence: A Systems Approach*

Minimax search: Expand the game tree by $m$ ply (levels in game tree) in a limited depth-first search. Then apply evaluation function at lowest level, and propagate results back up the tree.
**Minimax search**

From M. T. Jones, *Artificial Intelligence: A Systems Approach*

Current board:
X’s move

Calculate $f(n)$
Minimax search
From M. T. Jones, *Artificial Intelligence: A Systems Approach*

Current board:
X's move

Calculate $f(n)$

Propagate min value of children to parent

...
**Minimax search**

From M. T. Jones, *Artificial Intelligence: A Systems Approach*

Current board: X’s move

Propagate max value of children to parent

Propagate min value of children to parent

Calculate $f(n)$
**Minimax search**

From M. T. Jones, *Artificial Intelligence: A Systems Approach*

Current board: X’s move

- Propagate min value of children to parent
- Propagate max value of children to parent
- Propagate min value of children to parent
- Calculate $f(n)$
Minimax search
From M. T. Jones, *Artificial Intelligence: A Systems Approach*

Propagate max value of children to parent

Propagate min value of children to parent

Propagate max value of children to parent

Propagate min value of children to parent

Calculate $f(n)$
Minimax algorithm: Example

From M. T. Jones, *Artificial Intelligence: A Systems Approach*
Exercise

What is value at the root?
Alpha-Beta Pruning

Problem with minimax: **too expensive!**

Need to prune tree.

Solution: *alpha-beta pruning* algorithm

Basic idea: identify non-beneficial moves, and prune branches rooted in those moves
Alpha-Beta Pruning Example
From M. T. Jones, *Artificial Intelligence: A Systems Approach*

![Diagram showing alpha-beta pruning example](image)
**Alpha-Beta Pruning Example**

From M. T. Jones, *Artificial Intelligence: A Systems Approach*

**alpha** = value of the best possible move you can make, that you have computed so far

**beta** = value of the best possible move your opponent can make, that you have computed so far

If at any time, **alpha >= beta**, then your opponent's best move can force a worse position than your best move so far, and so there is no need to further evaluate this move.

[Diagram showing the process of pruning based on alpha-beta values]
Alpha-Beta Pruning Applet

Algorithm: Minimax with Alpha-Beta Pruning
http://en.wikipedia.org/wiki/Alpha-beta_pruning#Pseudocode

function alphabeta(node, depth, α, β, Player)
    if depth = 0 or node is a terminal node
        return the heuristic value of node
    if Player = MaxPlayer
        for each child of node
            α := max(α, alphabeta(child, depth-1, α, β, not(Player)))
            if β ≤ α
                break ; Prune
        return α
    else
        for each child of node
            β := min(β, alphabeta(child, depth-1, α, β, not(Player)))
            if β ≤ α break ; Prune
        return β
end
; Initial call
alphabeta(origin, depth, -infinity, +infinity, MaxPlayer)
Alpha-beta pruning exercise

(a) What is value at the root, using minimax alone?

(b) What nodes could have been pruned from the search using alpha-beta pruning? Show values of alpha and beta.
Alpha-beta pruning exercise

Remember:
alpha: best move for us seen so far
beta: best move for opponent seen so far

If alpha >= beta, prune

(a) What is value at the root, using minimax alone?

(b) What nodes could have been pruned from the search using alpha-beta pruning?
Note: Alpha Beta pruning effectiveness is highly depending on the move ordering!

How can we improve this ordering?
Minimax / Alpha-Beta

Assumptions:
Minimax / Alpha-Beta

Assumptions:

Opponent is rational

Both players have perfect information at each evaluation
Evaluation functions

Checkers and chess:

Weighted linear sum of features $f(s)$:

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$
Arthur Samuel’s checkers program, written in the 1950’s.

In 1962, running on an IBM 7094, the machine defeated R. W. Nealy, a future Connecticut state checkers champion.

One of the first machine learning programs, introducing a number of different learning techniques.
Samuel’s Checker Player

(http://www.ise.bgu.ac.il/faculty/felner/teaching/new8-11.ppt)

- **Rote Learning**
  - When a minimax value is computed for a position, that position is stored along with its value.
  - If the same position is encountered again, the value can simply be returned.
  - Due to memory constraints, all the generated board positions cannot be stored, and Samuel used a set of criteria for determining which positions to actually store.
Samuel’s Checker Player

(http://www.ise.bgu.ac.il/faculty/felner/teaching/new8-11.ppt)

- **Learning the evaluation function**
  - Comparing the static evaluation of a node with the backed-up minimax value from a lookahead search.
    - If the heuristic evaluation function were perfect, the static value of a node would be equal to the backed-up value based on a lookahead search applying the same evaluation on the frontier nodes.
    - If there’s a difference between the values, the evaluation the heuristic function should be modified.
Samuel’s Checker Player

- Selecting terms
  - Samuel’s program could select which terms to actually use, from a library of possible terms.
  - In addition to material, these terms attempted to measure following board features:
    - center control
    - advancement of the pieces
    - mobility
  - The program computes the correlation between the values of these different features and the overall evaluation score. If the correlation of a particular feature dropped below a certain level, the feature was replaced by another.
Deep Blue

• First Created in 1997
• **Algorithm:**
  – iterative-deepening alpha-beta search, transposition table, databases including openings of grandmaster games (700,000), and endgames (all with 5 pieces or more pieces remaining)
• **Hardware:**
  – 30 IBM RS/6000 processors
    • They do: high level decision making
  – 480 custom chess processors
    • all running in parallel
    • They do:
      – deep searches into the trees
      – move generation and ordering,
      – position evaluation (over 8000 evaluation points)
• **Average performance:**
  – 126 million nodes/sec., 30 billion position/move generated, search depth: 14
On May 11, 1997, the machine won a six-game match by two wins to one with three draws against world champion Garry Kasparov.[1] Kasparov accused IBM of cheating and demanded a rematch, but IBM refused and dismantled Deep Blue.[2] Kasparov had beaten a previous version of Deep Blue in 1996.

After the loss, Kasparov said that he sometimes saw deep intelligence and creativity in the machine's moves, suggesting that during the second game, human chess players had intervened on behalf of the machine, which would be a violation of the rules. IBM denied that it cheated, saying the only human intervention occurred between games.
On 14 December 2006, Topalov directly accused Kramnik of using computer assistance [from the Fritz chess computer] in their World Championship match.

On 14 February 2007, Topalov's manager released pictures, purporting to show cables in the ceiling of a toilet used by Kramnik during the World Championship match in Elista.