Game tree Search

What "Game" We're Considered?

1. Two players.
2. Zero-Sum: "How many A win = How many B loses" (i.e. no cooperation")

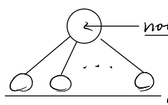
"Tic-Tac-Toe" as Search problem

1. Initial State: ① Initial board configuration ② indication of "who make the first move"
2. Operator(actions): Legal moves
3. Terminal test (Goal test): Determine when a terminal state is reached.
4. Utility function (payoff function): Score → outcome of the game (☆ this is not equivalent to evaluate function")

① Minimax with perfect decision

Procedure:

1. Expand the whole tree below n
2. Evaluate terminal nodes using utility function
3. Select a node which:



if no such nodes, then return (finished!)

4. if "MIN" node. value = min{all children node's value}.
   if "MAX" node value = max{all children node's value}.
5. repeat to 3

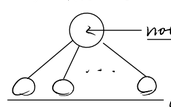
What is perfect decision?

"We draw the complete game tree" (no time limit imposed)
↓
We have all the possible states and we can get optimal solution"
↓
We can't lose!

Cons: Time/Space requirement are too large!

② Minimax with imperfect decision

1. Expand ~~the whole tree below n~~ ⇒ partial tree below n (i.e. we have depth limit)
2. Evaluate ~~terminal nodes using utility function~~ ⇒ evaluate the leaf node (i.e. where we stop) using evaluate function
3. Select a node which:

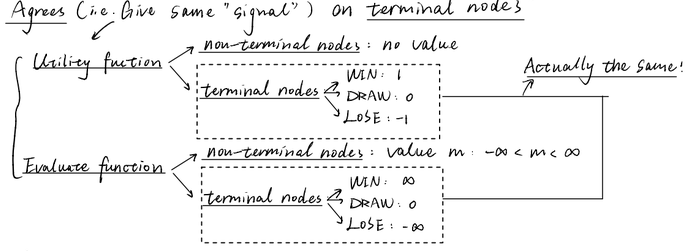


if no such nodes, then return (finished!)

4. if "MIN" node. value = min{all children node's value}.
   if "MAX" node value = max{all children node's value}.
5. repeat to 3

Evaluate function

1. "Evaluate function and Utility function Agrees (i.e. Give same "signal") on terminal nodes"

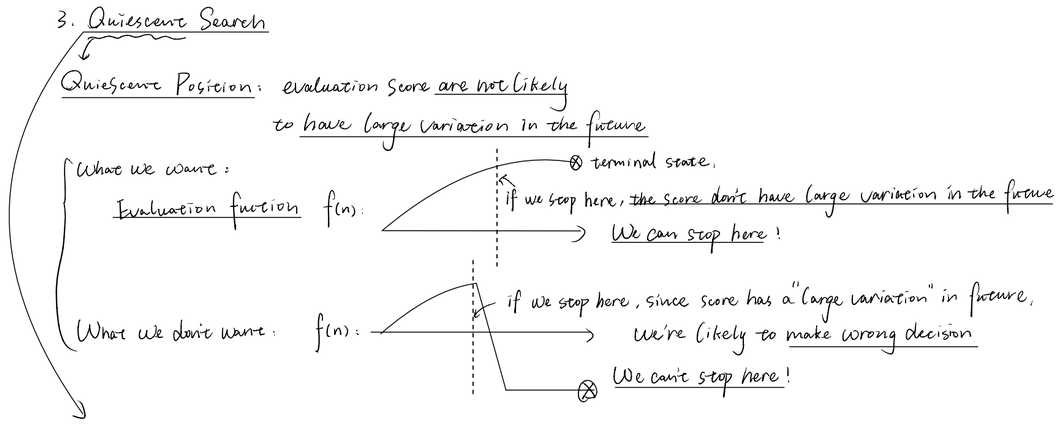


2. Reflect "Chance" of winning.
3. Evaluate function: $f(n) = w_1 f_1 + w_2 f_2 + \cdots + w_n f_n$
   Where f_i is "features", w_i is weighs → "Weighted sum of features"

Different Approaches (Variations):

1. Depth-limit search (legacy one)
2. Iterative deepening search (similar to the one in "uniformed search strategy")
3. Quiescent Search

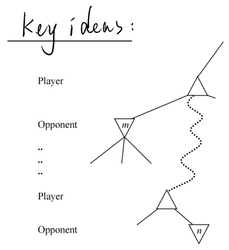
Quiescent Position: evaluation score are not likely to have large variation in the future



"Expansion of nonquiescent positions, until a quiescent position is reached"

Alpha-Beta Pruning Search

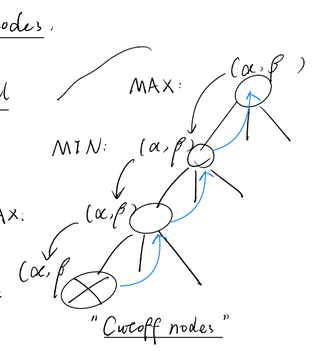
Key ideas:



if m is better than n, then n will never be reached

1. Expand the node Depth-first until "cutoff node", During Expansion, pass current α, β value of parent down to child nodes
2. Evaluate the cutoff nodes.
3. Update the value of all nodes that have been expanded (minimax)
4. Update (α, β)

MAX: $\alpha \leftarrow \max\{\alpha, \text{current nodes value}\}$.
MIN: $\beta \leftarrow \min\{\beta, \text{current nodes value}\}$



5. ☆ Prune all children of any node whose $\alpha \geq \beta$
6. Backtrack to a node that have not been pruned, then back to 1, if no such node, then return the root node (finished!)

Pseudo Codes

function MAX-VALUE(*state*, *game*, α, β) **returns** the minimax value of *state*
  inputs: *state*, current state in game
    game, game description
    α, the best score for MAX along the path to *state*
    β, the best score for MIN along the path to *state*

  if CUTOFF-TEST(*state*) **then return** EVAL(*state*)
  for each *s* **in** SUCCESSORS(*state*) **do**
    $\alpha \leftarrow$ MAX(α, MIN-VALUE(*s*, *game*, α, β))
    if $\alpha \geq \beta$ **then return** β
  end
  return α

function MIN-VALUE(*state*, *game*, α, β) **returns** the minimax value of *state*

  if CUTOFF-TEST(*state*) **then return** EVAL(*state*)
  for each *s* **in** SUCCESSORS(*state*) **do**
    $\beta \leftarrow$ MIN(β, MAX-VALUE(*s*, *game*, α, β))
    if $\beta \leq \alpha$ **then return** α
  end
  return β

Remarks: ① α-β pruning reduces the branching factor of each nodes from b to $\sqrt{b}$

② We always assume opponent plays optimally

Monte-Carlo Search:

① Expand Current node
② For each child, play a large number of random games to the end
   Compute the average payoffs (e.g. win? lose? draw? 1 / 0 / -1)
③ choose the child with the largest average value

What if: A child has small average value but a better outcome (e.g. a higher maximum)

"Average is misleading!"

"UCB": Upper Confidence Bound

$$UCB = \frac{w_i}{n_i} + c\sqrt{\frac{\log N}{n_i}}$$

Average payoff; Value of Exploration

if this action haven't been used frequently ⇒ n_i is small. ⇒ $\sqrt{\frac{\log N}{n_i}}$ is large

where:
- w_i: total times of winning under action i
- n_i: total number of plays for action i
- N: total number of plays for current node.
- c: a constant. "exploration parameter"

Monte-Carlo Search with Alpha-Beta Pruning

① Build the tree using α-β pruning (minimax)
② When: come to leaf nodes (touch depth limit)
   Continue on with MC search with UCB scoring rule
③ Propagate the value up