comp 250 - Lecture 36

Game playing programs

2-player, 0-sum games (one wins, one loses)

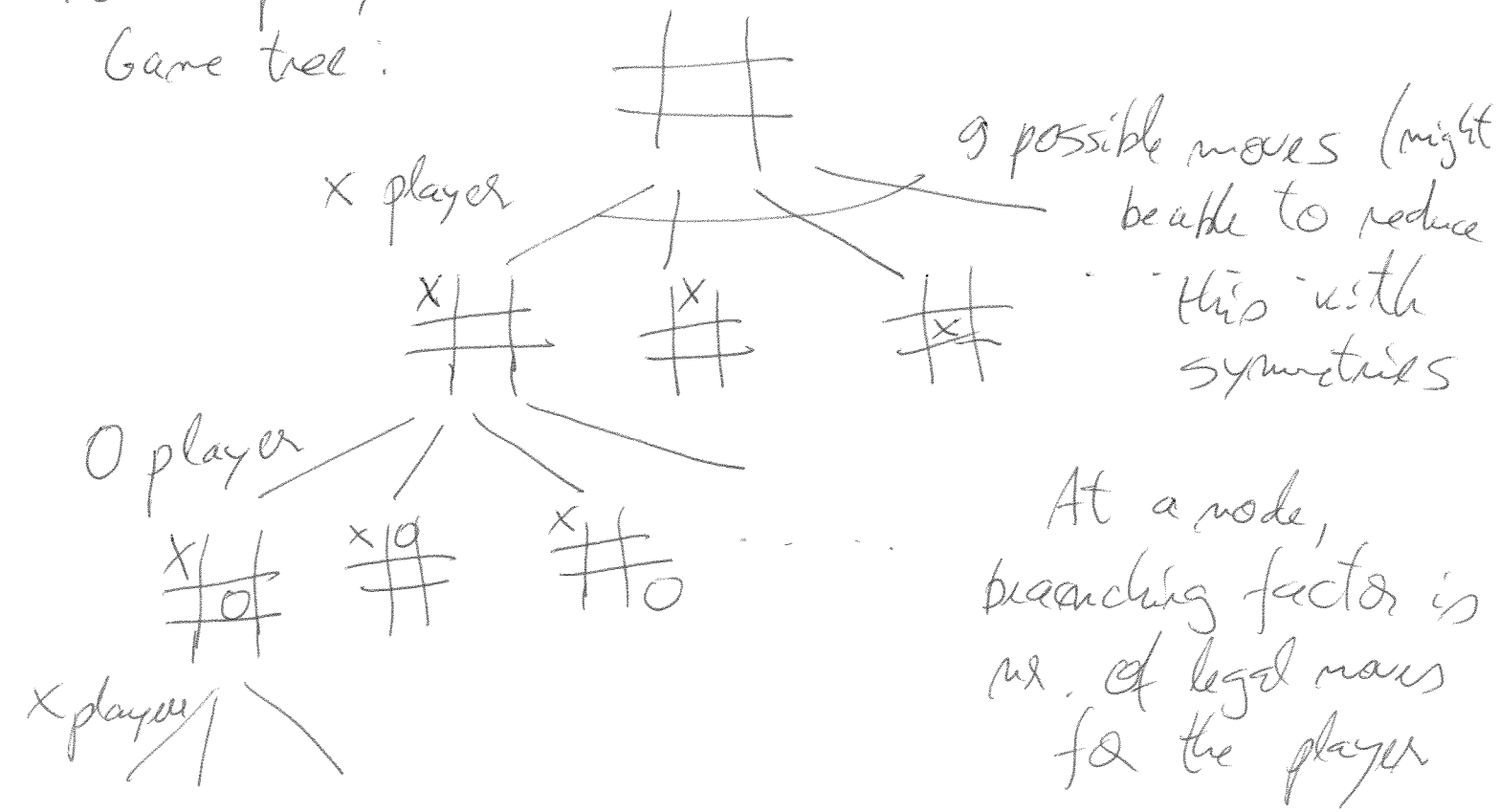
Players both see relevant info

- chess
- checkers
- go
- ...

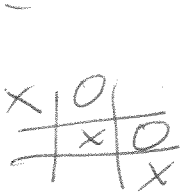
Not Poker, other card games

How to play well? → Data structures

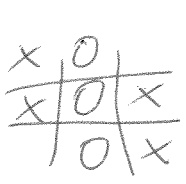
Game tree:



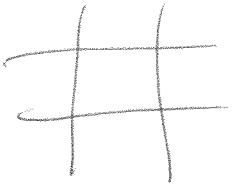
At a node, branching factor is nr. of legal moves for the player



Outcome: +1 (win for X)



-1 (loss for X)



ties

X: find a path to a +1 leaf → max player

O: -1 leaf → min player

Size of game tree is huge!

→ branching factor (nr. of legal moves)

→ length of the game (how many moves to the end)

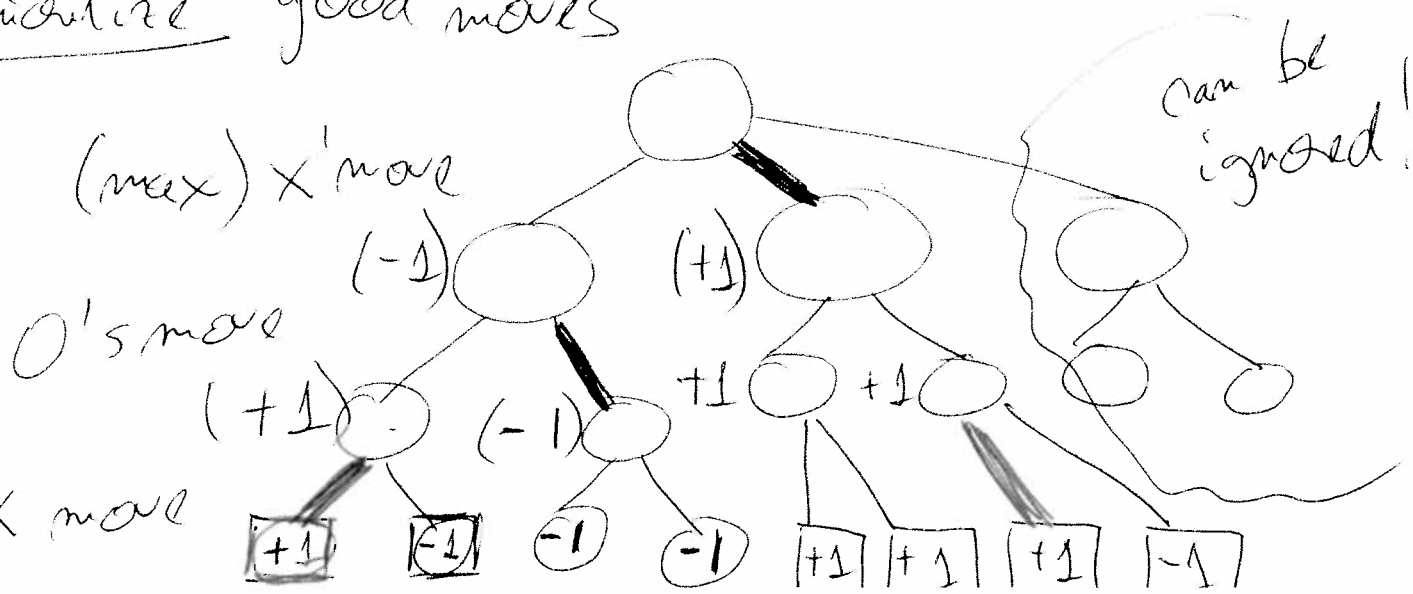
Chess: branching factor ≃ 20
depth > 40

Go: branching factor ≃ 40
depth > 100
Legal boards ≃ 10^{300}

a) Evaluating (approximately) a board position

b) Search strategy → focus on lines of play that are promising

Prioritize good moves



Max nodes (X's moves)
Min nodes (O's moves)

Recursion: base case is the leaves (aka positions where outcome is known)

Recursion → compute value of a node as max or min of children

⇒ Minimax search

Cut off the tree at a given depth → evaluation fn will "guess" (est) value of a position

Pruning (α-β pruning) → not going to search lines of play that cannot be better than the alt. best.

Algorithm MiniMaxValue (Board b, boolean minOrMax, int depth)

Output: value of board pos.

if depth = 0 return evalFn (b)

if (minOrMax = true) // min node

$\min_{b' \in \text{successors of } b}$ MiniMaxValue (b', !minOrMax, depth - 1)

else max

- Not generate successors worse than what we can get so far
- Not investigate successors for which eval(b') is bad

eval( ) → ask an expert!

e.g. pawn = 1
rook = 3
queen = 7
} value of board (simplistically) = sum of vals of pieces

"mobility"
"territory dominance"
patterns
→ features → simple quantity that measures approximately something important about the game (domain knowledge)

Combining features: linear combination:

$$eval(b) = \sum w_i f_i(b)$$

? ↑ features

e.g. w_i: for pieces you own is pos
opponent neg

How to come up with w_is (??)

→ Team of experts!

Eg. 1995 → IBM Deep Blue beat Gary Kasparov (chess world champion) → 1st time

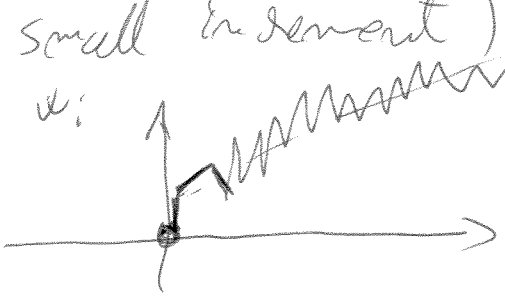
→ Learn weights from data!
(machine learning)

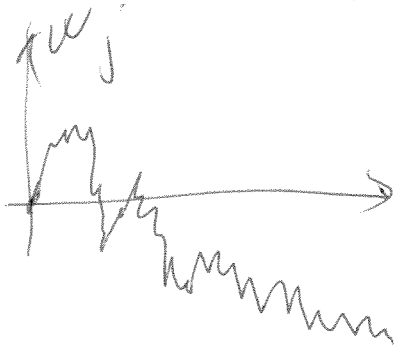
Trial-and-error: play the game, using est guess about the evaluation fn weights
⇒ observe the outcome (+1 or −1)

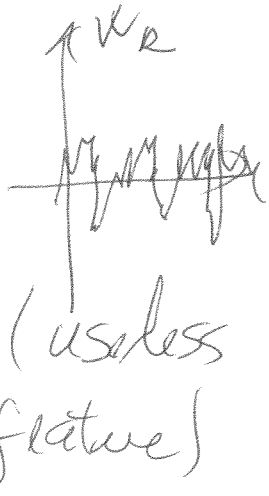
Increase the value of all our choices & boards (by a little bit)

(Decr value of boards where opponent made the move, from the opponent's point of view)

(small increment)







(useless feature)