More on Games

Chapter 5.4-5.7

Some material adopted from notes by Charles R. Dyer, University of Wisconsin-Madison
Overview

• Stochastic games
• Other issues
• AlphaGo Zero
• General game playing
Stochastic Games

• In real life, unpredictable external events can put us into unforeseen situations
• Many games introduce unpredictability through a random element, such as the throwing of dice
• These offer simple scenarios for problem solving with adversaries and uncertainty
Example: **Backgammon**

- Popular two-player game with uncertainty
- Players roll dice to determine what moves can be made
- White has just rolled 5 & 6, giving four legal moves:
  - 5-10, 5-11
  - 5-11, 19-24
  - 5-10, 10-16
  - 5-11, 11-16
- Good for exploring decision making in adversarial problems involving skill and luck
Why can’t we use MiniMax?

• Before a player chooses a move, she rolls dice and only then knows exactly what moves are possible
• The immediate outcome of each move is also known
• But she does not know what moves she or her opponent will have available in the future
• Need to adapt MiniMax to handle this
MiniMax trees with Chance Nodes
Understanding the notation

Board state includes chance outcome determining available moves
Game trees with chance nodes

- **Chance nodes** (circles) represent random events.
- For random event with N outcomes, chance node has N children, each with a probability.
- 2 dice: 21 distinct outcomes.
- Use minimax to compute values for MAX and MIN nodes.
- Use expected values for chance nodes.

- Chance nodes over max node: $\text{expectimax}(C) = \sum_i (P(d_i) \times \text{maxval}(i))$
- Chance nodes over min node: $\text{expectimin}(C) = \sum_i (P(d_i) \times \text{minval}(i))$
Impact on lookahead

• Dice rolls **increase branching factor**
  – There are 21 possible rolls with two dice
• Backgammon: ~20 legal moves for given roll
  ~6K with 1-1 roll (get to roll again!)
• At depth 4: 20 * (21 * 20)**3 ≈ 1.2B boards
• As depth increases, probability of reaching a given node shrinks
  – lookahead’s value diminished and alpha-beta pruning is much less effective
• **TDGammon** used depth-2 search + good static evaluator to achieve world-champion level
Meaning of the evaluation function

- With probabilities & expected values we must be careful about meaning of values returned by static evaluator
- Relative-order preserving change of static evaluation values doesn’t change minimax decision, but could here
- Linear transformations are OK
Games of imperfect information

• E.g. card games where opponent's initial hand unknown
  – Can calculate probability for each possible deal
  – Like having one big dice roll at beginning of game
• Possible approach: minimax over each action in each deal; choose action with highest expected value over all deals
• Special case: if action optimal for all deals, it's optimal
• GIB bridge program, approximates this idea by
  1. Generating 100 deals consistent with bidding
  2. Picking action that wins most tricks on average
High-Performance Game Programs

• Many programs based on alpha-beta + iterative deepening + extended/singular search + transposition tables + huge databases + ...

• **Chinook** searched all checkers configurations with \( \leq 8 \) pieces to create endgame database of 444 billion board configurations

• Methods general, but implementations improved via many specifically tuned-up enhancements (e.g., the evaluation functions)
Other Issues

- Multi-player games, no alliances
  - E.g., many card games, like Hearts
- Multi-player games with alliances
  - E.g., Risk
  - More on this when we discuss game theory
  - Good model for a social animal like humans, where we must balance cooperation and competition
AI and video Games

• Many games include agents run by the game program as
  – Adversaries, in first person shooter games
  – Collaborators, in a virtual reality game
  – E.g.: AI bots in Fortnite Chapter 2

• Some games used as AI/ML challenges or learning environments
  – MineRL: train bots to play Minecraft
  – MarioAI: train bots for Super Mario Bros
  – Facebook Nethack Learning Environment for reinforcement learning
General Game Playing

• **General Game Playing** is an idea developed by Michael Genesereth of Stanford

• See his [site](#) for more information

• Idea: don’t develop specialized systems to play specific games (e.g., Checkers) well

• Goal: design AI programs to be able to play more than one game successfully

• Work from a description of a novel game
GGP

• Input: logical description of a game in a custom game description language
• Game bots must
  • Learn how to play legally from description
  • Play well using general problem solving strategies
  • Improve using general machine learning techniques
• Regular completions 2005-2016, $10K prize
• Java General Game Playing Base Package
Tic-Tac-Toe in GDL

(role xplayer)
(role oplayer)

;; Initial State
(init (cell 1 1 b))
(init (cell 1 2 b))

;; Dynamic Components
(<= (next (cell ?m ?n x))
  (does xplayer (mark ?m ?n))
  (true (cell ?m ?n b)))

(<= (next (cell ?m ?n o))
  (does oplayer (mark ?m ?n))
  (true (cell ?m ?n b)))

(<= (goal xplayer 100) (line x))
(<= (goal oplayer 0) (line x))

A example of General Intelligence

• Artificial General Intelligence describes research that aims to create machines capable of general intelligent action

• Harkens back to early visions of AI, like McCarthy’s Advise Taker
  – See Programs with Common Sense (1959)

• A response to frustration with narrow specialists, often seen as “hacks”
  – See On Chomsky and the Two Cultures of Statistical Learning
“Chess is the Drosophila of artificial intelligence. However, computer chess has developed much as genetics might have if the geneticists had concentrated their efforts starting in 1910 on breeding racing Drosophila. We would have some science, but mainly we would have very fast fruit flies.”

John McCarthy, Stanford

“Saying Deep Blue doesn’t really think about chess is like saying an airplane doesn't really fly because it doesn't flap its wings.”

Drew McDermott, Yale
AlphaGO

- Developed by Google’s DeepMind
- Beat top-ranked human grandmasters in 2016
- Used Monte Carlo tree search over game tree
  expands search tree via random sampling of search space
- Science Breakthrough of the year runner-up
- Match with grandmaster Lee Sedol in 2016 was subject of award-winning 2017 AlphaGo
Go - the game

• Played on 19x19 board; black vs. white stones
• Huge state space $O(b^d)$: chess:~$35^{80}$, go: ~$250^{150}$
• Rule: Stones on board must have an adjacent open point ("liberty") or be part of connected group with a liberty. Groups of stones losing their last liberty are removed from the board.

liberties capture
alphaGo implementation

- Trained deep neural networks (13 layers) to learn value function and policy function
- Performs Monte Carlo game search
  - explore state space like minimax
  - random "rollouts"
  - simulate probable plays by opponent according to policy function
AlphaGo implementation

- Hardware: 1920 CPUs, 280 GPUs
- Neural networks trained in two phases over 4-6 weeks
  - **Phase 1:** supervised learning from database of 30 million moves in games between two good human players
  - **Phase 2:** play against versions of self using reinforcement learning to improve performance
Why study games?

• Interesting, hard problems that require minimal “initial structure”
• Clear criteria for success
• A way to study problems involving {hostile, adversarial, competing} agents and the uncertainty of interacting with the natural world
• People have used them to assess their intelligence
• Fun, good, easy to understand, PR potential
• Games often define very large search spaces
  – chess $35^{100}$ nodes in search tree, $10^{40}$ legal states