AI Paradigm: Symbolic AI

A* Search

COMP 741/841 Week 3

Spring 2024

Agenda

- Symbolic AI and A* search
- Lab 2
- Assigned reading
  - Data poisoning: A double-edge sword
- Due next week

Symbolic AI

- AI paradigm, born in mid-1950s, went through “summers” and “winters”
- Inspired by how humans reason (system 2 in the dual process theory)
- "Solves" problems:
  - Based on the **symbolic representations** of the problem:
    - entities and relationships that have meanings for humans
  - Using **symbolic reasoning** that's based on knowledge representation
    - inferences, rules, algorithmic steps

Examples of symbolic AI

- Search
  - Heuristic (combinatorial) search
- Logic-based problem solvers, mathematical reasoning, automatic theorem proving
- Game playing: mini-max algorithm, alpha-beta pruning
- Knowledge-based systems
  - Expert systems, e.g., IBM's Deep Blue (encoded chess expert knowledge, used alpha-beta search, and VLSI chips to parallelize the algorithm)
- Probabilistic reasoning
  - e.g., Hidden Markov Models (speech recognition)
- Planning, Natural Language Processing (NLP), Satisfiability, Constraint Satisfaction, …

Symbolic AI benefits

- Explicit representation of the domain knowledge (related to the problem)
- Well-deinfed logical, deductive, inferential reasoning steps of the problem solving process
- Transparency and explainability
  - Humans can understand and explain the automated decision-making process

Symbolic AI limitations

- Knowledge acquisition bottleneck
  - Domain knowledge experts are needed to create adequate knowledge representations
- Difficulties representing the domain knowledge, which can be:
  - ambiguous, incomplete, uncertain, complex
- Brittleness if the knowledge representation needs changes
- Limited scalability

Neural AI

- AI paradigm, born in mid-1940s, went through ups and downs
- Inspired by how the brain functions (system 1 of dual process theory)
- Requires large amounts of data
  - MNIST (a simple task for today's standards) requires 60,000 training images
- "Learns" from data
  - Finds patterns and makes predictions based on complex and unstructred data
- Non-explainable, opaque
  - Input to output can be traced
  - But the trace does not help with explaining the outputs

A* search algorithm

Source: Wikipeida A* search algorithm

Solves the following **problem**: Find the most cost-effective path from a specified **source** to a specified **goal**

- Problem **input representation**: weighted graph (nodes and edges)
- Problem **output representation**: A path in the graph, from source to goal, having the smallest cost (sum of the weights)

Uses a **heuristic function h(n)**: calculates an estimate of the cheapest path from a node **n** to the **goal**.

A* optimality condition

- If **h(n)** is **admissible**, that is, it **never overestimates** the **actual cost** to get to the **goal**
- Then A* guarantees optimality, that is, finds the **least-cost path** from **source** to **goal**

A* applications

- Find the shortest route on a map, **h(n)** can be the straight-line distance to the goal
- Find the shortest route on a **grid map, h(n)** can be the **Manhattan distance**

A* algorithm idea

At each iteration, determine which of the paths to extend with the **next node n** by calculating

- **g(n)**: The cost of the path from the **source** to **next node n** and
- **Heuristic function h(n)**, which
  - Estimates the cost of the **cheapest path** from **n** to the **goal**
- Such that **f(n) = g(n) + h(n)** is minimized
- Pseudocode: https://en.wikipedia.org/wiki/A*_search_algorithm#Pseudocode
- Examples with animation: https://en.wikipedia.org/wiki/A*_search_algorithm#Example

A* algorithm data structure

- Uses a **priority queue** to select the next node to expend
- The queue is known as `fringe` or `frontier`
- At each step, the node `x` with the lowest `f(x)` is removed from the queue
  - `f` and `g` values of the neighhbors of `x` are calculated
  - the neighbors are added to the queue
- Algorithm continues until:
  - the removed node is the **goal** node, OR

A* search algorithm implementation

Andreas Soularidis: An Introduction to A* Algorithm in Python
https://medium.com/p/79475244b06f

- Also published on PlainEnglish at https://plainenglish.io/blog/a-algorithm-in-python

GitHub URL from Andreas Soularidis public repo **medium_articles**
https://github.com/AndreasSoularidis/medium_articles.git

- See **AStarAlgorithm** folder

Lab 2

See description in Canvas.

Assigned Reading (RN2)

Dhar, Payal. 2023. “Protecting AI Models from “Data Poisoning".” 2023. IEEE Spectrum. https://spectrum.ieee.org/ai-cybersecurity-data-poisoning

Heikkila, Melissa. 2023. This New Data Poisoning Tool Lets Artists Fight Back against Generative AI. Magazine. MIT Technology Review. https://www.technologyreview.com/2023/10/23/1082189/data-poisoning-artists-fight-generative-ai/.

Due Next Week

Monday, Feb 12, midnight

- Complete and submit Lab 2
- Read, annotate, and be prepared to discuss assigned reading