Lecture 22

Final Review

Lusi Li
Department of Computer Science
ODU

Review

- **Intelligent Agent**

- **Problem Solving Agent**
  - Classical search
  - Constraint satisfaction problem
  - Game tree search

- **Logical Agent**
  - Propositional Logic
  - First-Order Logic

- **Learning-based Agent**
  - Decision Tree
  - KNN algorithm
  - K-means algorithm

Problem-Solving Agents

- Problem-solving agent: goal-based agent
- Goal formulation
  - Set of one or more desirable world states
- Problem formulation
  - What actions and states to consider given a goal and an initial state
    - States
    - Initial State
    - Actions
    - Transition model
    - Goal test
    - Path cost
- Search for solution
  - Given the problem, search for a solution --- a sequence of actions to achieve the goal starting from the initial state
- Execution of the solution

Implementing a Problem-Solving Agent

- Search through the state space using a search tree
- Search tree is generated by the initial state and successor function
  - Expansion of nodes
    - generating a new set of child nodes and adding them to the frontier
  - Search strategy
    - determines the selection of the next node to be expanded
    - can be achieved by ordering the nodes in the frontier
- Search algorithms:
  - Tree-Search: frontier
  - Graph-Search: frontier and explored

Search Strategies

- **Uninformed search (Blind Search)**
  - **Can only generate successors and distinguish goals from non-goals**
  - **BFS, DFS, UCS**
- **Informed search (Heuristic Search)**
  - **Know whether one non-goal state is "more promising" than another**
  - **Greedy Search, A* Search**

- **Performance measure**

  Complete? Guaranteed to find a solution when there is one?
  Optimal? Find the cheapest solution?
  Time complexity? How long does it take to find a solution?
  Space complexity? How much memory is needed to perform the search?

Constraint Satisfaction Problem

- A CSP is defined as a triple <*V, D, C*>
  - finite set of variables $V = \{V_1, V_2, \ldots, V_n\}$
  - non-empty domains of possible values for each variable

    $D = \{D_{V1}, D_{V2}, \ldots, D_{Vn}\}$
  - finite set of constraints $C = \{C_1, C_2, \ldots, C_m\}$ that specify allowable combinations of values
    - each constraint consists of a pair <scope, relation>

- A state is an assignment of values to some or all variables.
- A solution to a CSP is a complete and consistent assignment.

Games

- **Two players: MAX and MIN**
- **MAX moves first and they take turns until the game is over.**
- **A game can be defined as a search problem:**
  - `initial state` s_0: **how the game is set up at the start**
  - `Player(s)`: **which player has the move in state s**
  - `Action(s)`: **set of legal moves in state s**
  - `Result(s, a)`: **the state resulting from action a in state s**
  - `Terminal-Test(s)`: **true if game is over (terminal state) otherwise false**
  - `Utility(s,p)`: **a numeric value of terminal state s for a player p, e.g., win (+1), lose (-1) and draw (0) in chess.**

- **The** s_0, `Action(s)` **and** `Result(s, a)` **define the game tree, where the nodes are game states and the edges are moves.**
- **Players use game tree to determine next move.**

Minimax Algorithm

- **Assumption: both players play optimally.**
- **Given the game tree, the optimal strategy can be determined by using the minimax value of each node MiniMax(s).**

```
MiniMax(s) =

if Terminal-Test(s) then Utility(s)
if Player(s) = MAX then
     max of MiniMax(Result(s, a)) for a in Actions(s)
if Player(s) = MIN then
     min of MiniMax(Result(s, a)) for a in Actions(s)
```

Minimax search problem: complete search is impractical for most games

Minimax with Alpha-Beta Pruning

Key points:

1. **Each node has to keep track of 3 values: α, β, v (minimax value).**
2. **Pruning condition: if α >= β for node n, stop expanding the children of node n and return its current v**
3. **MAX will update only α values and MIN player will update only β values. Both of them will update v values.**
4. **Return v values to parent nodes of the game tree**
5. **Pass α and β values to child nodes.**

Logical Agents

- Intelligent agents need **knowledge about the world** for making good decisions.
- The knowledge of an agent is stored in a knowledge base in the form of **sentences** in a **knowledge representation language**.
- A knowledge-based agent needs a **knowledge base** and an **inference mechanism**. It operates by storing sentences in its knowledge base, inferring new sentences with the inference mechanism, and using them to deduce which actions to take.
- A **representation language** is defined by its **syntax** and **semantics**, which specify structure of sentences and how they relate to world facts.
- The **interpretation** of a sentence is the fact to which it refers. If this fact is part of the actual world, then the sentence is true.

Entailment and Inference

- **KB |= α**
  - **If KB entails α, then all models (assigning 'true' or 'false' values to symbols) that evaluate the KB to True also evaluate α to True.**
- **KB |– α**
  - **Inference is a procedure for deriving a new sentence α from KB following some inference approach.**
- **Given:**
  - **KB: a set of sentences**
  - **α: a sentence**
- **The inference approaches:**
  - **Truth-table approach**
  - **Inference rules**
  - **Resolution algorithm**

Truth-Table Approach

- **KB |= α ?**
- **A two steps procedure:**
  - **Generate table for all possible models (n symbols $\rightarrow$ 2^n entries)**
  - **Check whether the sentence α is true whenever the sentences in KB are true**

- **To determine if one sentence is satisfiable and valid, we can also use truth-table to list out all the models for it.**
  - **If there exists at least one model that evaluates it to true, it is satisfiable.**
  - **If all the models evaluate it to true, it is valid.**

Resolution Algorithm

- To show that KB $\models \alpha$, we show that (KB $\wedge \neg\alpha$) is unsatisfiable

1. (KB $\wedge \neg\alpha$) is converted into CNF
2. Apply iteratively the resolution rule to the resulting clauses
   - Each pair that contains complementary literals is resolved to produce a new clause, which is added to the set if it's not present
3. Stop when:
   - Contradiction (empty clause {}) is reached:
     - E.g., P, $\neg$ P $\Rightarrow$ {}
     - Prove the entailment
   - No more new clauses can be derived
     - Reject the entailment

- { } is a disjunction of no disjuncts is equivalent to False, thus the contradiction

Basic Algorithm of Decision Tree

node = root of decision tree

Main loop:

1. A ← the "best" decision attribute for the next node.

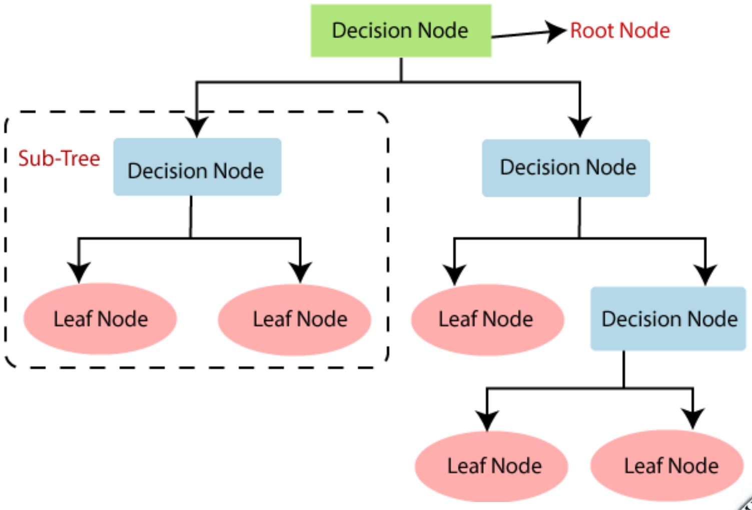
2. Assign A as decision attribute for node.

3. For each value of A, create a new child (sub-tree) of the node.

4. Sort training examples to leaf nodes.

5. If training examples are perfectly classified, stop.

Else, recurse over new leaf nodes.



Final Exam

Complete Student Opinion Survey for extra points

- **0am April 25th – 11:59pm April 26th under "Exams" on Canvas**
- **Time duration: 3 hours**
- **Open-book and open-note**
- **A final exam practice and its solutions under "Exams" on Canvas are available**

- **Total points: 100**
- **Five Problems**

  1) Search problem: BFS, DFS, UCS, Greedy, A*

  2) Games: minimax and alpha-beta pruning

  3) Inference in propositional logic

  4) Inference in propositional logic

  5) Decision tree