Einführung in die künstliche Intelligenz

Zusammenfassung
15. November 2023

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Inhaltsverzeichnis

1 Lecture 1: What is AI?

What is, and what isn't AI?

- **John McCarthy**: The science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable.
- **Marvin Minsky**: The science of making machines do things that would require intelligence if done by men.

What is intelligence?

- **Turing Test**:
  - **Question**: When does a system behave intelligently?
  - **Assumption**: An entity is intelligent if it cannot be distinguished from another intelligent entity by observing its behavior.
  - **Test**:
    - 1.Human interrogator interacts with two players, A and B (one of them is a computer) blind.
    - 2.If the interrogator cannot determine which player, A or B, is a computer and which is a human, the computer is said to pass the test.
  - **But:** hard/not reproducible and constructie or amenable to mathematical analysis
  - **Followup question:** Does being a human-like mean you are intelligent?

- **The Chinese Room Argument**:
  - **Question**: Is intelligence the same as intelligence behavior?
  - **Assumption**: Even if a machine behaves in a intelligent manner, it does not have to be intelligent at all.
  - **Test**:
    - 1.A person who doesn't know Chinese is locked in a room. Outside the room is another person wo can slip notes written in Chinese inside the room but cannot interact with the person in the room else.
    - 2.The person inside the room has detailed instructions how to answer each question without translating it or understanding it.
  - **Followup question:** Is the person in the room intelligent? Is a self-driving car intelligent?

Characteristics of AI

- **General vs narrow AI**: A general/strong AI is defined to so that it can handle any intellectual task, while a narrow or weak one is specified to deal with a one concrete or a set of specified tasks. While strong AI ist generally a goal in research, currently we are using primarly narrow AI.
- **An AI should be:**
  - Adaptability: The ability to improve perfomance by learning from experience.
  - Autonomous: The ability to perform tasks in enviroments without constant guidance by a user/expert.
  - Law of Thoughts: Beginning of reasoning and the question of what 'correct' argument and thought processes are.
  - Rational Behavior: The question of 'doing the right thing'. In systems that act rationaly, the 'right thing' is that what is expected to maximize the achievement given the available information.

2 Lecture 2: AI Systems

What is an AI System?

- **AI system**: can be defined as the study of rational agents and their environments and has two parts:
  - Environment
  - Agent

What is an Environment?

- In the context of AI, an environment is simply the surrounding of an agent and is where the agent operates. An environment does not have to be real. It could also be artificial
- Self driving cars: Streets, traffic, weather, road signs, pedestrians,
- Chess: The chess boars, the chess pieces

Characteristics of Environments

- **Discrete vs. Continuous**: If the environment has a limited number of distinct, clearly defined states then it is called discrete; otherwise continuous
- **Observable vs. Partially Observable or Unobservable**: If it is possible to determine the complete state of the environment at each time point, then it is called observable; otherwise it is only partially observable or unobservable.
- **Static vs. Dynamic**: If the environment does not change while an agent is acting, then it is static; otherwise it is dynamic.
- **Single Agent vs. Multiple Agents**: The environment may contain other agents which may be of the same or different kind as that of the agent.

- **Deterministic vs. Non-deterministic**: If the next state of the environment is completely determined by the current state and the actions of the agent, then the environment is deterministic; otherwise it is non-deterministic.
- **Accessible and Observable**: Is this environment observable? Is this environment accessible?
- **Episodic and Non-Episodic**: In an episodic environment, each episode consists of the agent perceiving and then acting. The quality of its action depends just on the episode itself. In sequential environments, the agent requires memory of past actions.

What is an Agent?
An agent:

- perceives its environment (Sense)
- makes decisions autonomous (Think)
- acts upon the environment (Act)

Rules of AI agents:

1. An agent must be able to perceive the environment
2. The environment's observations must be used to make decisions
3. The decision should result in action
4. The action taken by the agent must be rational

The problem of rationality

- An action is rational, if it maximizes the performance and yields the best positive outcome for the agent.
- A rational agent is an agent that does the right thing
- Rational Agents are not perfect. Rationality maximizes the expected performance. This may not be the optimal outcome.

Types of agents

- **Reflex agent**: Selects action on the basis of only the current percept but ignores the percept history. Problem:
  - Very limited decision making

  - No knowledge about anything which the agent cannot actively perceive
  - Can be very hard to handle in complex environments
- A rational agent is an agent that does the right thing
- Rational Agents are not perfect. Rationality maximizes the expected performance. This may not be the optimal outcome.
- **Model-based agent**: These agents choose their actions like reflex agents do, but they have a better comprehensive view of the environment, i.e. keep track of the world state. Problem:
  - How does my actions affect the world?
  - What world model do I desire?
- Input is not only interpreted, but mapped into an internal state description (a world model)
- **Goal-based agent**: These agents are build on the information that a model-based agent stores but in addition know what states are desirable. Problem:
  - Things become difficult when long sequences of actions are required to find the goal
- The agent knows what states are desirable and will try to choose actions accordingly
- Main difference to previous approaches is that it takes decision-making into account e.g. "What will happen if I do such-and-such?".
- **Utility-based agent**: Instead of providing goals, Utility-based agents use a utility function providing a way to rate each action/step based on the desired result.
  - A learning agent can learn from its experience i.e. is able to adapt automatically based on its experience
  - Is more robust towards unknown environments
  - A learning agent has four conceptual components:
    - Learning Element: It is responsible for making improvements by learning from the environment
    - Critic: Gives feedback, describing how well the agent is doing with respect to a fixed measurement

        - ⋆ Performable Element: It is responsible for selecting actions.
        - ⋆ Problem Generator: Responsible for suggesting actions that will lead to new experiences.

How to make agents intelligent?
Approaches:

- Search Algorithms
    - Understand/Define "finding a good actionäs a search problem and use search algorithms
    - Spoiler: Most common search algorithms are tree-based
- Reinforcement Learning
    - Developed within the field of psychology
    - Trial and error
    - Reactions/Actions are based on our observation and experience
- Genetic Algorithms
    - Survival of the fittest

3 Lecture 3: Uninformed and Informed Search

Problem Definitions:

- we have a goal $\Rightarrow$ goal-based agent
- State space = possible situation in our environment. The state space is a set of all possible situations.
- Transition = describes possible actions to take between one state and another. We only count direct transitions between two states (single actions).
- Cost = transitions aren't alike and differ $\Rightarrow$ we express this by adding a "cost"to each action. Often the goal in search algorithms is to minimize the cost to reach the goal.
- A single-state problem is defined in 4 items:
    1. **State space:** Description of all possible states and the initial environment as a state(state-space graph)
    2. **Descriptions of Actions:** Typically a function that maps a state to a set of possible actions in this state
    3. **Goal Test:** Typically a function to test if the current state fulfils the goal definition
    4. **Costs:** A cost function that maps actions to its cost
- **Planning problem:** we have an initial state and want to transform it into a desired goal considering future actions and outcomes of them. item[•] **Search:** The process of finding the (optimal) solution for such a problem in form of sequence of actions.

Tree Search Algorithms:

- **Idea:** Treat the state-space graph as a tree.
- can use simulated iterative exploration of the state space
- Needs a strategy to determine which node is expanded next
- **Fringe:** set of all nodes at the end of all visited paths is called frine.
- **Depth:** numbers of levels in the search tree
- **Expanding a node:** creates a new node(leaf/successor nodes of the chosen node) and updates the tree

Search Strategy:

- **Idea:** defined by picking the order of node expansion
- **Completeness:** Does it always find a solution if one exists?
- **Time Complexity:** Number of node expansions
- **Space complexity:** Maximum number of nodes in the memory
- **Optimality:** Does it always find the optimal (least costs) solution?

Uninformed Search Algorithm

- Does not have any information except the problem definition
- Breadth-first search, Uniform-cost search, Depth-first search, Depth-limited search, Iterative deepening *Uniform-Cost Search and Breadth-First Search(BFS)*
  - **Uniform-Cost Search UCS:** Each node is associated with a fixed cost and they accumulate over the path within the search. Uniform-Cost Search uses the lower cumulative cost to find a path.
  - **Breadth-First Search:** A special case of the uniform-cost search, when all costs are equal. It starts at the tree root and explores the tree level by level. BFS stops as soon as it generates a goal, whereas UCS examines all the nodes at the goal's depth to see if one has a lower cost.
  - **Completeness:** Yes, if each step has a positive cost, otherwise infinite loops are possible
  - **Complexity:** $O(b^d)$ for BFS, $O(b^{1+\text{floor}(\text{OptCost}/\text{eps})})$ for UCS

- **Optimality:** Yes, since the nodes expand in increasing order of path costs.
- Both are often implemented using a priority queue ordered by costs.

Depth-First Search (DFS)

- It starts at the tree root and explores the tree as far as possible along one branch before going step-wise back and exploring alternative branches.
- **Completeness:** No, fails in infinite-depth search spaces and spaces with loops
- **Time Complexity:** Explores each branch until max. depth m, i.e. $O(b^m)$
- **Space Complexity:** Only a branch and their unexpanded siblings have to be stored
- **Optimality:** No, longer solutions may be found before shorter solutions

Depth-limited Search

- The depth within the search is limited to l. Nodes with depth d>l are not considered.
- **Completeness:** No
- **Time Complexity:**$O(b^l)$
- **Space Complexity:** $O(b \times l)$
- **Optimality:** No, see DFS.

Iterative Deepening Search

- Increase l after each failed search
- **Completeness:** Yes
- **Time Complexity:** first levels have to be searched d times
  $\Rightarrow d \cdot b + (d-1)b^2 + \cdots + 1 \cdot b^d = \sum\limits_{i=0}^{d}(d-i+1) \cdot b^i$
- **Space Complexity:** Linear complexity $O(b \cdot d)$
- **Optimality:** Yes, the shortest path is found.

Heuristics

- **Idea:** Uninformed search algorithms are inefficient. Try to be more clever with what nodes to expand, bring knowledge into the process.
- Informally denotes a "rule of thumb", i.e. a rule that may be useful in solving the problem. In tree-search, a heuristic denotes a function h that estimates the remaining distance to travel.

Informed Search Algorithm

- Have additional knowledge about the problem and an idea where to “look” for solutions

  Greedy Best-first Search

  - Using the function $f(n) = h(n)$ to estimate the cost from node n to goal, expand the tree with the smallest cost according to $f(n)$
  - **Completeness:** No, we can get stuck in loops.
  - **Time Complexity:**Worst Case $O(b^m)$, same as DFS but can be improved using good heuristics
  - **Space Complexity:** has to keep all nodes in memory, worst case $O(b^m)$
  - **Optimality:** No, solution depends on heuristic.

  A Search*

  - An informed tree search algorithm, build on best-first search. Tries to minimize not only the estimated cost $h(n)$ but also the true costs so far $g(n)$
  - **Idea:** Avoid expanding paths that are already expensive, evaluate complete path cost not only remaining cost, i.e. $f(n) = g(n) + h(n)$
  - **Completeness:** Yes
  - **Time Complexity:**It can be shown that the number of nodes grow exponentially unless the error of the heuristic $h(n)$ is bounded by the logarithm of the value of the actual path cost $h^*(n)$
  - **Space Complexity:** has to keep all nodes in memory
  - **Optimality:** Depends on the heuristic

- A heuristic is **admissible** if it never overestimates the cost to reach a goal.

- A heuristic is consistent if for every node n and every successor n' generated by any action a it holds that h(n) $\geq$ c(,n,a,n') + h(n')
- **Lemma 1:** Every consistent heuristics is admissible.
- **Lemma 2:** If h(n) is consistent, then the values of f(n) along any path are non-decreasing.
- **Relaxed Problems:** A problem with fewer restrictions on the actions is called a relaxed problem.
- If h1 and h2 are both admissible heuristics and h2(n) $\geq$ h1(n) for all n, then h2 dominates h1.

4 Lecture 4: Local Search and Adversarial Search

Optimization Problems

- An optimization problem is one where all states/nodes can be a solution (to different degrees) but the target is to find a state that optimizes (min, max, ...) the solution according to an objective function. The "bestßtates, there are no explicit goal states and also no path costs.
- **Objective/Evaluation Function:** An objective function tells us how good a state is, also in comparison to other states. Its value is either minimized or maximized depending on the optimization problem.

  Key Terminologies

  - Convergence: describes the property of a sequence of values to approach a limit/value and more
  - Global optimum: A global optimum is the extremum (minimum or maximum) of the objective function for the entire input search space.
  - Local optimum: A local optimum is the extremum (minimum or maximum) of the objective function for a given region of the input space.

  Local search

  - Local search: Local search algorithm traversing only a single state rather than saving multiple paths in memory. It modifies its state iterative, trying to improve a criteria.
  - Advantages: Uses a very little/constant amount of memory, finds a reasonable solution in very large state spaces

- Disadvantages: No guarantees for completeness or optimality.

Local Search Algorithms

- *Basic idea:*
  - Start with a complete (but most likely suboptimal) tour.
  - Perform pairwise exchange
  - Variants of this approach get within 1% of an optimal solution very quickly with thousands of cities.
- *Hill Climbing / Greed Local Search:*
  - Expand the current state (generate all neighbours)
  - Move to the one with the highest evaluation
  - Do so until you reach a maximum (evaluation goes down again)
  - Problem: Local Optima
    - The algorithm will stop as it at the top of a hill
    - But the hill does not have to be optimal.
  - Solution: Local Optima
    1. Randomized Restart Hill Climbing Different initial positions result in different local optima.
    2. Stochastic Hill-Climbing Select the successor node randomly, better nodes have a higher probability of being selected.
  - Problem: Ridge Problem/Íssue
    - Every neighbour state appears to be downhill
    - The search space has an uphill, The neighbours not
- *Gradient descent:*
  - Gradient: A gradient is a derivative of a function that has more than one input variable. Known as the slope of a function in mathematical terms, the gradient simply measure the change in all weights with regard to the change in error.

  - Gradient descent: a very popular optimization strategy. It can be described as a Hill-Climbing in a continous state space.
  - Do so until you reach a maximum (evaluation goes down again)
  - Problem: Local Optima
  - Goal: Minimize cost-function J(Θ)

- *Gradient descent - Hill-climbing in continous spaces:*
  Gradient vector ∇J(Θ) $= \left[\frac{\delta J(\Theta)}{\delta\Theta_0} \frac{\delta J(\Theta)}{\delta\Theta_1} \dots\right]$ Assume we have some cost-function: J(x_1, x_2, x_3, $\dots$, x_n) and we want to minimize over continuous variables: x_1, x_2, x_3, $\dots$, x_n

  1. Compute the gradient: $\frac{\delta}{\delta x_i}J(x_1, x_2, x_3, \dots, x_n)\forall i$
  2. Take a small step downhill in the direction of the gradient: $x_i' = x_i - \lambda \frac{\delta}{\delta x_i} J(x_1, \dots, x_n)$
  3. Check if J$(x_1', x_2', x_3', \dots, x_n') <$ J$(x_1, x_2, x_3, \dots, x_n)$
  4. If true then accept move, if not reject.
  5. Repeat.

- *Gradient descent - The importance of the learning rate:*
  The learning rate is a hyperparameter, controlling how quickly the model is adapted to the problem.
  How big should we choose the steps the gradient descent takes?

  - Smaller learning rates: Smaller changes, require more training episodes
  - Larger learning rates: More rapid changes, needing fewer epochs, Can converge to a local optima or not at all.

- *Beam Search:*
  Idea: Keep track of k states rather than just one (k is called beam size)
  Algorithm

  - Start with k randomly generated states.
  - At each iteration, all the successor of all k states are generated,
  - Select the k best successors from the complete list and repeats.

- *Simulated Annealing:*
  Idea: Use conventional hill-climbing style techniques, but occasionally take a step in a direction other than that in which there is improvement.(downhill moves; away from solution)
  As time passes, the probability that a down-hill step is taken is gradually reduced and the size of any down hill step taken is decreased.
  - Allows some "bad moves"to escape local optima
  - Temperature: a hyperparameter, controlling how frequently we allow "bad moves"to escape local optima. Usually the temperature decays exponentional over the process. Effectiveness:
    - If lowered slowly enough $\Rightarrow$ converges to a global optimum
    - Unfortunately this can take a very very long time.
- *Simulated Annealing - Conclusion:*
  Superficially
  - Simulated Annealing is local search with some noise added. Noise starts high and is slowly decreased.

  True story is much more principled:
  - Simulated Annealing is a general sampling strategy to sample from a space according to a well-defined probability distribution.
  - The temperature has a strong effect on the quality of the Simulated Annealing process.

Adversial Search

- a type of search, where one examines the problem that arises when we try to plan ahead of the world and other players are planning against us or has conflicting goals to our while sharing the same search place. *Key Terminologies:*
  - Environment: The surroundings or conditions in which an agent operates.
  - Perfect Information vs. Imperfect Information: (No) Information is hidden.

- Deterministic vs. Non-deterministic: The next state of the environment is completely determined by the current state and the action executed by the agent versus not. In nondeterministic environments actions are characterized by their possible outcomes, but no probabilities are attached to them. If weighted by the probabilities than it is stochastic.
- Zero-Sum Games: If one party loses, the other party wins, and the net change in wealth is zero.

Formalization of the Problem:

1. Initial state: Specifies how the game is set up at the start
2. Player(s): Specifies which player turn it is
3. Action(s): Returns a set of legal moves in the state s.
4. Result(s, a): Transition model, specifies the resulting state s' doing move a in state s
5. Terminal(s): Tests if state s fulfills the goal/terminal constraints
6. Utility (s, p): The utility function returns a numeric value for a terminal state s from the perspective of player p

Game Trees

- To solve games, we build so called game trees
- Different to search trees, game trees are arranged in levels that correspond to players(The root node is always the active/current player)
- In game trees, leaf nodes are called terminal
- Each terminal node has a utility value corresponding to the outcome of the game.

Games vs Search Problems

1. ”Unpredictable” opponent: Specify a move for every possible reply, Different goals per agent (my interest vs your interest)
2. Time limits: Unlikely to have enough time to find a goal, needs approximation
3. Most games are deterministic, turn-taking, two-player, zero-sum
4. But most "real"problems are stochastic, parallel, multi-agent, utility based

Minmax Algorithm Idea: Build a game tree where the nodes represent the states of the game and edge represent the moves made by the players in the game.
The players are:

1. MIN: Decrease the chances of MAX to win the game.(Opponent)

2. Max: Increases his chances of winning the game.

Both play the game alternating, i.e., turn by tu rn and following the above strategy. Choose move to position with highest minmax value. Assuming opponent to play the best response to your action.

- Minmax strategy follows the DFS concept.
- Each level is either MIN or MAX.
- You play as MAX, oppenent is MIN.
- On a max level we maximize the value
- On a min level we minimize the value

The problem of massive game trees Problem: In many games, the game tree is simply way to big to traverse in full. Heuristic evaluation functions:

- Instead of traversing the full tree, we limit the depth.
- For each leaf node we calculate a value for the current situation based on heuristics. Estimation of the likely outcome of the game if we continue from here.

Pruning:

- Means to cut back the tree
- Ignoring unwanted portions of a search tree which make no difference to its final result
- E.g. Alpha-Beta Pruning.

Alpha-Beta Pruning:

- A modified version of the minmax algorithm. It uses a technique called pruning to reduce the amount of exploration without losing the correctness of Minmax. Alpha-Beta Pruning is based on two parameters:

- Alpha: The best (highest-value) choice we have found so far at any point along the path of Maximizer to the root. The initial value of alpha i $-\infty$.
- Beta: The best (lowest-value) choice we have found so far at any point along the Minimizer to the root. The initial value of beta is $+\infty$.
- Idea: Remove all the nodes which are not really affecting the final decision but making the algorithm slow, hence by pruning these nodes.
- Differences to minmax:
  - The max player will only update the value of alpha.
  - The min player will only update the value of beta.
  - While backtracking the tree, the node values will be passed to upper nodes instead of values of alpha and beta.
  - We will only pass the alpha and beta values to the child nodes.