KI VO

Systems thinking like humans: Cognitive Science, Cognitive Neuroscience

Systems acting like humans: Turing Test

Systems thinking rationally: Laws of Thought

Systems acting rationally: Agents

Agents

: sensors
actuators,
agent function

Rationality: Performance measure
prior knowledge
percept sequence
agent actions

=> maximize performance measure

Task environment: Performance measure
Environment
Actuators
Sensors

Agent types: simple reflex agent
reflex agent with state
goal-based agent
utility-based agent

Uninformed Search

Problem formulation: initial state
successor function
goal test
path cost

↳ solution: sequence of actions leading from initial state to goal state

Strategy: order of node expansion
-> completeness
-> time complexity
-> space complexity
-> optimality

- Breadth first search — expansion-time
- Uniform cost search — after expansion
- Depth first search — expansion = after expansion
- Depth limited search — expansion -> limit = 0 still check
- Iterative deepening search

Informed and Local Search

Use problem- or domain-specific knowledge
↳ $f(n)$ estimates $f^*(n)$
↳ $h(n)$ heuristic function

Greedy search: $f(n) = h(n)$
↳ ignores already spent costs

A* search: $f(n) = g(n) + h(n)$

-> admissible: $h(n) \leq h^*(n)$
$h(n) \geq 0$
$h(g) = 0$

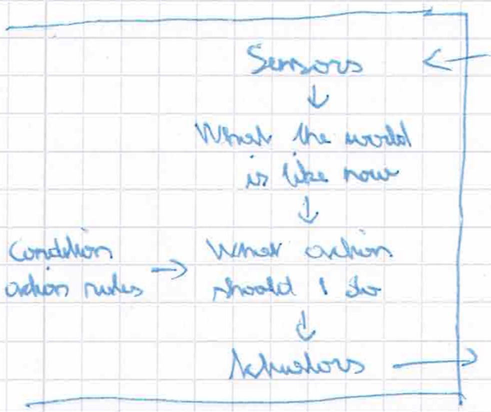
-> consistent: $h(n) \leq c(n, a, n') + h(n')$

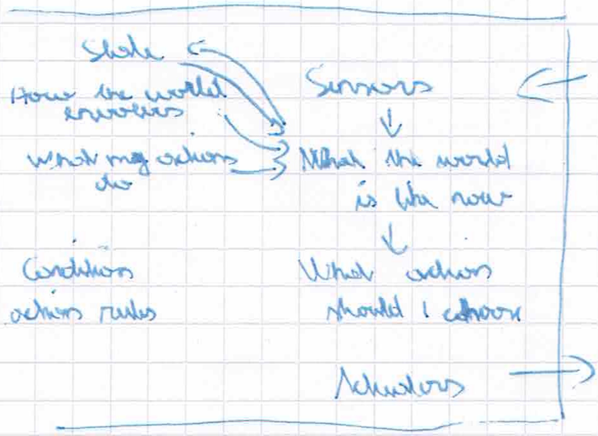
-> dominance: $h_2(n) \geq h_1(n)$, h_2 dominates h_1
↳ $\max(h_1, h_2)$

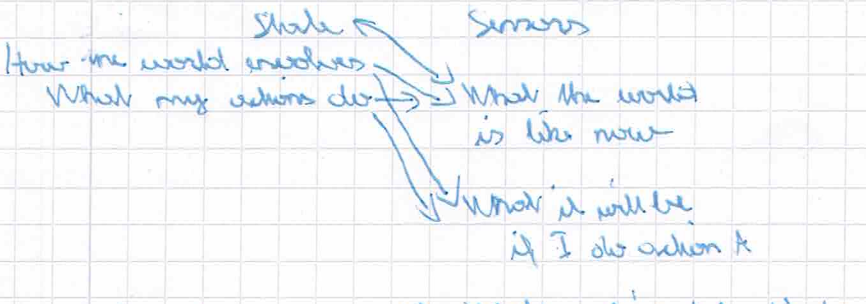
-> Relaxation for heuristic

(iterative improvement
-> try to improve current state!

Local Search: Hill climbing,
Simulated annealing
Local beam search
Genetic algorithms: Fitness, Selection, Crossover, Mutation

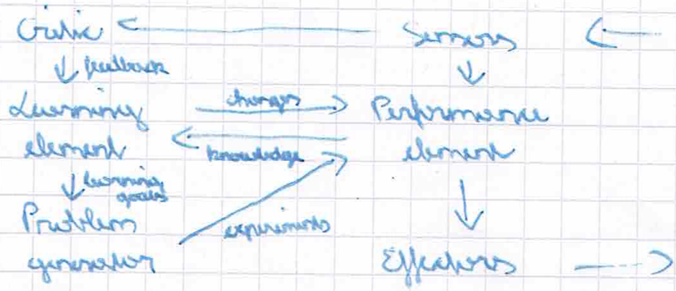






Learning

Learning Element : makes improvements
Critic : performance / result assessment
Problem generator : suggests actions



Modes: Unsupervised : no explicit feedback
Reinforcement : occasional feedback
Supervised : correct answers for each instance

-> semi-supervised

Inductive learning: learn a function from examples
-> find function h() that approximates f()

output type: classification, regression (real numbers)
consistent: h agrees with f on all examples

-> maximize simplicity under consistency (Ockham's razor)

Decision Tree learning: possible representation for hypothesis
-> sequence of tests in a tree
-> find small tree: choose good attributes

Information Gain: $B(q) = -(q \cdot \log_2(q) + (1-q)\log_2(1-q))$

$$Rem(A) = \sum \frac{p_i + n_i}{p+n} B(p_i / (p_i + n_i))$$

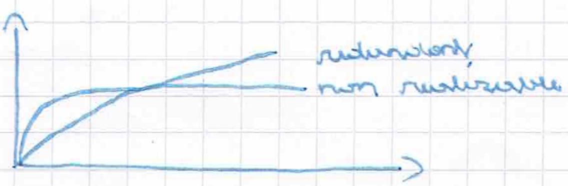
$$Gain(A) = B(p/(n+p)) - Rem(A)$$

Overfitting -> generalization : may consider irrelevant attributes
may break with new examples

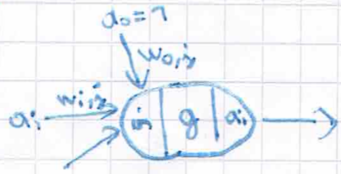
Measuring learning performance:

-> Learning curve : % correct on test set, as function of training set size

-> depends on: realizability - missing attributes, restrictive hypothesis space
redundant expressiveness - irrelevant attributes



Neural Networks



- input: weighted sum of inputs (+ fixed input)
- activation function: g (in)
- output: $a_i \leftarrow g(in)$

Activation functions: step functions
linear function
sigmoid function: $1/(1+e^{-x})$

-> If graph and activation functions are known,
f is determined by weights $w = (w_{i,j})$

$f(x) = h_w(x)$

Feed Forward Networks: information flow from input to output
nodes in layer i connected to $i-1, i+1$

-> no internal state: reflex agent

Recurrent Networks: directed cycles with delays

-> Hopfield networks, Boltzmann machines

Perceptrons: represents linear separator in input space
-> AND, OR, NOT
-> not XOR

Multi-Layer Perceptrons: 1 layer: linearly separable functions
2 layer: continuous functions
3 layer: all functions

Learning: $Err = y - h_w(x)$

Perceptron: $w_i \leftarrow w_i + \alpha \cdot Err \cdot g'(in) \cdot x_i$

Multi Layer:

Output Layer (a_k): $w_{j,k} \leftarrow w_{j,k} + \alpha \cdot \overbrace{Err \cdot g'(in)}^{\Delta k} \cdot a_j$

Hidden Layer (a_j): $w_{i,j} \leftarrow w_{i,j} + \alpha \cdot \Delta_j \cdot a_i$

(Backpropagation) $\Delta_j = g'(in) \cdot \sum_k w_{j,k} \cdot \underbrace{Err \cdot g'(in)}_{\Delta k}$

-> Pros: Easier to develop than statistical methods
complex pattern recognition tasks
Fault tolerance
Unstructured (difficult) input

Cons: Choice of parameters (layers, units) requires skill
Sufficient training material
Result cannot be understood easily
Behaviour prediction difficult

Constraint Satisfaction Problems

-> states and goals conform to a standard, structured representation

-> Search algorithms take advantage of the structure of the states. -> use general-purpose heuristic

Components: finite set of variables
non empty domain for each variable
finite set of constraints (subset of cartesian product)

-> state: assignment of values to some/all variables

-> assignment: consistent/legal: does not violate any constraints
complete: mentions every variable
solution: complete + consistent

-> Constraint graph for binary CSPs
↳ nodes: variables
edges: constraints

↳ Hypergraph

-> Varieties of CSPs:

- infinite domains: constraint language needed, algorithm for linear constraints, non-linear are undecidable
- continuous domains: linear programming methods

Backtracking search: depth first search with single variable assignment

-> Variable assignment is commutative
-> d^n
-> basic uninformed algorithm

- Minimum-remaining-values heuristic
- Degree heuristic: choose first variable, largest number of constraints on other unassigned variables
- Least-constraining-value: choose value for variable, select value that rules out fewest choices for neighbours

Forward checking: remove inconsistent values from domains of neighbours when assigning

Arc consistency: simplest form of constraint propagation

- $X \Rightarrow Y$ is consistent, iff for every value x of X there is some allowed value y of Y

Intelligent Backtracking

Planning ... coming up with a sequence of actions that will achieve some goal

-> frame problem: how to represent things that stay unchanged after performing an action

-> qualification problem: how to deal with required preconditions for an action

-> ramification problem: how to represent implicit effects after performing an action

-> classical planning environment: fully observable
deterministic
finite
static
discrete

Representation language STRIPS:

States: conjunction of positive literals
literals ... atomic formulas

any condition not mentioned is false

Goals: partially specified state; conjunction of pos. literals

Actions: action name + parameter-list

precondition: conjunction of function free positive literals

effect: conjunction of function free literals
-> negative literals are removed from state
-> positive are added

-> applicable if state satisfies precondition

Representation language ADL:

- negative literals in states
- unmentioned literals are unknown
- Quantified variables in goals
- Disjunction in goals
- Conditional effects
- Equality
- Variables can have Types

Planning algorithms

-> straightforward approach: state-space search

Progression planning: start with problem initial state
consider sequences of actions until finding
sequence that reaches goal state

-> in absence of functions, state space of planning problem
is finite
↳ any graph search algorithm is complete planning algorithm

Regression planning: start with goal state

-> advantage: consider only relevant actions

-> consistent: should not undo any desired literal

-> delete positive effects, add preconditions

Partial order planning

-> regression / progression planning: only strictly linear sequences of actions

-> plan components: actions
ordering constraints
causal links,
open preconditions

-> Empty plan: Start action: no precond., effect: add initial state
Finish action: precond.: goal, no effect

↳ actions: Start, Finish, planning problem actions
↳ ordering constraint: $A \prec B$
↳ causal link: $A \xrightarrow{p} B$: A achieves p for B
↳ open preconditions: not achieved by some action in plan

POP Algorithm: consistent plan: no cycles in ordering constraints
no conflicts with causal links

-> initial plan: start, finish action
Start < Finish ordering constraint
no causal link
all precond. from finish as open precond.

-> successor function: pick open Precondition p
generate successor plan for every consistent
way of choosing an action A that achieves p

-> add $A \xrightarrow{p} B$
add Start < A, A < Finish

solution: consistent
no open preconditions

-> resolve conflicts: $B < C$ or $C < A$
(if C conflicts)

Decision Theory

→ continuous measure of outcome (not good/bad)

... combines utility theory with probability theory,
... makes rational decisions based on beliefs and desires

... deals with choosing among actions based on desirability of their immediate outcome

... preferences are expressed by an utility function

→ nondeterministic environment: outcome states are represented in form of random variable

→ expected utility $EU = \sum_{s'} P(Result(a) = s' \mid a, e) \cdot U(s')$

... principle of maximum expected utility

→ States can be seen as Lottery

$L = [p_1, S_1; p_2, S_2; \dots; p_n, S_n]$

→ Preferences: $A \succ B$ (A, B ... States)

→ Axioms of Utility Theory:

Orderability: $A \succ B$, $B \succ A$ or $A \sim B$

Transitivity: $(A \succ B) \wedge (B \succ C) \Rightarrow (A \succ C)$

Continuity: $A \succ B \succ C \Rightarrow \exists p \; [p, A; 1-p, C] \sim B$

Substitutability: $A \sim B \Rightarrow [p, A; 1-p, C] \sim [p, B; 1-p, C]$

Monotonicity: $A \succ B \Rightarrow (p > q \Leftrightarrow [p, A; 1-p, B] \succ [q, A; 1-q, B])$

Decomposability:

$$[p, A; 1-p\,[q, B; 1-q, C]] \sim [p, A; (1-p)q, B; (1-p)(1-q), C]$$

... an agent violating these axioms will act irrational

→ Expected Utility of Lottery: $U([p_1, S_1; p_2, S_2 \dots]) = \sum_i p_i \, U(S_i)$

$U(A) > U(B) \Leftrightarrow A \succ B$

→ Environment: episodic, not sequential.
→ current decisions do not influence future decisions

→ linear curve: risk neutral

Utility Function

... agent behaviour does not change by linear (affine) transformations

... in deterministic environments numbers do not matter, just preference ranking on states

↳ ordinal utility function

Utility of money: EMV

$U(S_n)$

Descriptive Theory ... how humans really do act

→ Allais Paradox

... certainty effect

→ Ellsberg Paradox

... ambiguity aversion (clear known probability

Decision Networks ... general mechanism for making rational decisions

→ chance nodes (oval) ... random variables
→ decision nodes (rectangle) ... choice of actions
→ utility nodes (diamond) ... utility function

Value of Information ... choose what information to acquire

→ expected improvement in utility compared to making decision without the information

→ Decision Networks: contains information about agents current state, possible actions, states that will result from actions, utility of states

Philosophical Foundations

-> Weak AI hypothesis: machines could act as if they were intelligent
-> Strong AI hypothesis: machines that do so are actually thinking

Turing Test: three players: man, computer, interrogator
interrogator communicates with man and computer through written notes
interrogator tries to determine which one is the computer

Argument from Disability
... machine can never do X

Mathematical Objection
... Gödels incompleteness theorem
-> machines are limited

Argument from Informality
... human behaviour is too complex to be captured by a set of rules
-> qualification problem

Argument of consciousness
... machines have to be aware of their own mental states an actions

Mind-Body Problem
... mind and body must exist in separate realms -> dualist theory
... mind is not separated from body -> mental states are physical states
-> monist theory

Functionalism
... mental state is any intermediate causal condition between input and output
-> program could have same mental state as person
-> brain replacement experiment
↳ consciousness is a product of the electronic brain

Biological Naturalism
... mental states are high-level emergent features that are caused by low level physical process in the neurons
-> unspecified properties of the neurons matter
-> cannot be duplicated on the basis of functional structure
↳ require architecture with same causal powers as neurons