Eiki

What is AI?

making machines do
what requires intelligence if done by man

(by human interrogator)
if not distinguishable from another intelligent (human)
- not mathematical
- only human-like intelligent?

intelligence ≈ intelligent behaviour ⇐ chinese room argument
human isn't intelligent
But system is.

God: general / strong ↔ most: narrow / weak

by Learning: Adaptability
tasks in environment without guidance: Autonomous

Characteristics

Praxis

Law of thoughts
"correct" argument/reasoning
- sometimes no provable right

Rational Behavior
do 'right thing' = max achievement
+ mathematical
- how to define

system do like: Think Act / ~~Think~~ ~~Act~~

Rational goal?
⇕
Human
└ Cognitive Science: how do humans think
Neuroscience: neuronal level

History
neural networks
↓
expert systems
↓
machine learning

Foundations
philosophy: reasoning, language
mathematics: formal repres, computation
psychology: adaptation, perception
economics: rational decisions, gametheory
linguistics: grammar (knowledge repres.)
Neuroscience
Control Theory

AI-System

Environment

z.B. Schach.
Objekterkennung (ohne Liste Objekte).

discrete ⇔ continuous
(partially / un-) observable
possib. determine complete state of environment
static ⇔ dynamic
└ env. doesn't change while agent acting
single- / multiple agents - diff kinds?
accessible ⇔ unaccessible
└ agent can observe full environment
Deterministic ⇔ Non-" / stochastic
next state completly defined by current
Episodic ⇔ Non-" / sequential
quality of action just depends on episode

Agent

Perceive
Decide
Act

in abstraction of environment

sonst müsste actual out come

not omniscient

rational = max expected performance
└ function of sequence of actions → evaluation
task dependent
good design based on desired outcome not behaviour

Types

implementation: condition-action rules (state → action)

Reflex: acts only of current percept
Model-based: " + keeps track of world state (effects of action? desired world state?)
Goal-based: " + considers effect of action
knows desired state (goals) - only binary
state/sequence of states
Utility-based: utility function action/scenario → score based on goals
instead of goals
└ + continuous scale
+ conflicting goals

source of intelligence
Search Algorithm - find good action
Reinforcement Learning - Trial and Error ⇒ Experience
Genetic Algorithm - survival of fittest
population → selection → crossover → Mutation

Learning: Learning Element ⇄ (knowledge / changes) Performance Element
Critic (feedback)
Problem Generator - suggests new experiences
Performance Element: Sensors → Actuators
+ more robust
Basic Principle

für mehr Grafiken siehe Vorlesung 2

Planning (IState) Problem

considering future actions, outcomes their

Initial state and State Space (implicitly defined by initial, successor func. (state graph))
Descriptions of Actions - possible direct transitions between states
typically f: STATE → P(ACTIONS)
Goal test typically f: STATE → B
Costs f: ACTIONS → COST
typically additive for multiple actions

Search ⇓
(Optimal) Solution
solution with min path cost
Path from initial to goal
Sequence of states
connected by Sequence of Actions

Tree search Algorithms

init
while (Fringe.size > 0){
x = strategy(Fringe)
if (goal(x)) return solution
expand(x); // creates new nodes
}

Uninformed search - no info except problem definition

Uniform-cost-Search UCS
node = cost, accumulate (start = 0 cost) lowest FIFO
≈
BFS Breadth-First-Search
UCS but ∀ costs equal

Completeness: if no negative cost
Complexity: BFS $O(b^d)$, UCS $O(b^{1+\lfloor \frac{\text{opt costs of optimal solution}}{\text{steps}} \rfloor})$
Time
Space Complexity:
Optimality: ✓

Node - data structure

- state
- parent, action
- path costs
- current depth

State - Representation physical configuration

	Depth limited	Iterative Deepening	Bidirectional
			- actions reversible
completeness	✗	✓	✓
Time	$O(b^l)$	$\sum_{i=0}^{d} (d-i+1) \cdot b^i$	$(b^{d/2} + b^{d/2} \ll b^d)$
space	$O(b \times l)$	$O(b \cdot d)$	
Optimality	✗	✓	

Combination

DFS LIFO still often faster
not with loops or infinite depth without avoiding repeated or depth limit
$O(b^m)$ m = max depth
$O(b * m)$ +
✗

BFS path may not meet

BRUNNEN

Informed Tree Search Strategies

Heuristics h - function estimating remaining costs

- admissible $h(n) \leq h^*(n)$
- consistent $h(n) \leq c(n,a,n') + h(n')$

How to find heuristics (fewer restrictions)
optimal solution for relaxed problem

Dominance h_1, h_2 admissible
$h_2(n) \geq h_1(n) \Rightarrow h_2$ dominates h_1

Combining $h_2 \dots h_m$ admissible
$h(n) = \max(h_1(n), \dots h_m(n))$ admissible

greedy Best-first Search

$f(n) = h(n)$ only use heuristic
evaluation func
- loops
- not optimal
- $O(b^m)$

estimate path → goal via n

A^*

$f(n) = g(n) + h(n)$ (g: true costs so far)
$h^*(n)$ - actual path cost (optimal heuristic)

+ complete except infinite nodes with $f(n) \leq f(G)$
- exponentially unless $|h(n) - h^*(n)| \leq O(\log h^*(n))$
- has to keep all nodes in memory

Optimal for admissible heuristic
Sei $C^* = g(G)$ shortest path
$\forall n \; f(n) \leq C^* \not< f(G_2)$ (suboptimal solution)

reduce complexity
- detect repetitive states → graph search (general solution, works for all)
- ensure cheaper path taken first → heuristics

Space problem

Alternatives
- Iterative-deepening A^* (IDA*)
  cutoff is $f(n)$
- recursive best-first search (RBFS)
  keeps f value of best alternative
  updates ancestors with successors
- (Simple) Memory-bounded A^* ((S)MA*)
  drop worst leaf node if memory full
  update to parent, may be re-searched later

can handle $\leq 10^{700}$ states

Local search

Optimization Problems

min/max objective function of states
~~goal/not best~~ → for comparison
no path costs

Convergence: sequence approaching value more & more
Global Optimum: on entire input search space
Local Optimum: for a given region

Locale Search Algorithms

iterative modifying single state
+ space (little) constant)
+ reasonable solutions in very large spaces
- no guarantees (completeness, optimality)

Hill climbing / Greed local search

(in continuous state space)

expand current
move to best
until max

- Local Optima
  - Randomized Restart
  - Stochastic (prob. based on slope)
- Ridge Problem
  search space has uphill
  but all transitions go down

Multi-Dimensional State-Landscape

finding global optimum gets worse with dimensions

Gradient Descent + smooth spaces

derivat of multidim. function
cost function $J(x_1, \dots x_n)$

$$\nabla J(\underline{\theta}) = \left[\frac{\partial J(\underline{\theta})}{\partial \theta_0} \dots \frac{\partial J(\underline{\theta})}{\partial \theta_n}\right]$$

take step downhill, check if better
↳ width?

Hyperparameter Learning Rate
small ↔ large
more epochs ↔ might not converge

e.g. SAT

violated(Δ, I): # clauses violated
flip(l, v): change valuation of v

hyperparameter Temperature
accept bad move with $e^{\Delta E/T}$ (Temp.)
$\Delta E = v(\text{next}) - v(\text{current})$
usually decays exponential

Variants: Simulated Annealing: Local search + noise
Linear Programming
Beam Search: keep track of k states
choose k from list of all suc.

BRUNNEN

Adversarial Search - plan ahead for world/players in same space with conflicting goals

Perfekt ⇔ Imperfect Information

Zero-Sum Games: one loose ⇒ one win net change in wealth = 0

Initial state

Player(s) - who's turn
Action(s) - set of legal moves
Result(s, a) - Transition model returns s'
Terminal(s) - Test goal/terminal constraints
Utility(s, p) - terminal state s, player p perspective

Game Trees

levels ~ ~~active~~ player
root = active Player
terminal = leaf have outcome

Games

useful quantifiable metric
low-stake training
interest
predict opponent
time limit
deterministic, turn-taking, 2player, 0sum
⇔ reality

Minimax Algorithm

alternating turns
perfect opponent
max = you, min = opponent

completeness if game tree finite
time $O(b^m)$
space $O(b \cdot m)$ ← Problem
Optimal if optimal opponent

→ Heuristic evaluation function - Limit depth, evaluate non terminal states

Alpha-Beta Pruning

Max Alpha - best/max choice found so far on path to root init $-\infty$
only updated by Min Beta - min $+\infty$
└ only passed down

Max expand if ~~prune~~ $V <$ beta some min in path would'nt choose this anyway
Min $V >$ alpha

$O(b^{\frac{m}{2}})$ in ideal ordering, try interesting paths first

~~no ties~~ still expand if (no world changing improvement)

CSP - Constraint Satisfaction Problems ⊆ Identification ⇔ Planning
goal ⇔ path important

Assumptions: single agent, deterministic actions
fully observed ~~sp~~ state, discrete state space
observed

state = $(X_{1,...})$ Variables, values out of D
- no blackbox as in standard search
still structure + local search

{ consistent / legal ⇔ not
complete ⇔ partial

goal test = Set of Constraints (
z.B. mathematische Formeln
L allowable Combination of $X^{vars} \subseteq$ state
values of

{ unary
binary/are
Higher-order

Linear constraints
⇒ solveable in polynomial time

Beispiele
Assignment
Timetabling
Hardware config.
Spreadsheets
Scheduling
Floor planning
n-Queens
X = one for each row
D = position in row
4 Color Map
first computer assisted proof
Wildlife Corridor Design

Constraint Optimization Problems
Preferences
= soft constraints

Constraint Graphs + Abstraction
Variable → Node
Constraint → connects affected Nodes

How to solve

Search
start node: empty assignment
successors: 1 var more assigned
goal: all assigned + constraints

basic uninformed search for CSPs →

Backtracking - DFS with ~~one~~ 1 var assignment per level
iterative add assignments
only continue if no conflict with constraints
Worst case still exponential

improve by heuristics for / detect failures early / specific structure?
└ Domain-Specific

n variables
v values

Naive - try all values for all vars — $n! \cdot v^n$ leaves in search tree
↓
Commutative Variable Assignment - fixed assignment order of vars — v^n } search tree height = 1
↓

improve by heuristics for / detect failures early / specific structure?
order of tried values
order of vars assigned
(z.B. general heuristics)
- Minimum Remaining Values (first.)
- Degree - most remaining constraints first
- Least Constraining Value - rules out te fewest remaining possibilities / possibilities

} used in this Rider
in case of tie

Constraint Propagation
V vars maintain set of possible values
Node consistency - possible values → V unary constraints
conform to
Local consistency - V nodes consistent with its neighbours
by iterative entering edges

{ not all can be solved by this alone

Constraint Propagation and Backtracking Search
after each assignment → constraint propagation
only constraints with changed var in queue

1 Node Consistency
2- Arc Consistency
3- Path Consistency - V assignments a set $arc(X_i, X_j)$
∃ assignment set $arc(X_i, X_m) \wedge arc(X_j, X_m)$
↑ special fälle
k-Consistency - set of k values need to be consistent
checking up to n-ary constraints

Forward Checking - rule out possib. with direct violations
terminate if ∃ var: possibilities(var) = ∅
impl. iteratively visit all constraints
remove inconsistent values
schedule recheck of all neighbors

checking exponential in k
⇒ arc-cons. most common

example Convert 9-ary Sudoku constraints into pairwise

Local Search
only complete states
maybe unsatisfied
Operation: recussign var
Min-Conflicts Heuristic
random var
value violating min constraints
⇒ Hill climbing with h(n) = # violated constraints

BRUNNEN

+ performance
− can't solve all

Problem Decomposition
if groups of vars with no constraints between
+ linear complexity to decompose

Tree-structured CSPs
V two vars only connected by a single path in constrain graph
+ only $O(n \cdot d^2)$
choose root
remove inconsistent from bottom-up
top-down assign value consistent with parent

nearly tree-structured
collapsing nodes together
- decompose into tree-shaped

cutset conditioning
- remove nodes → Rest tree structured
try solutions for S until also finding corresponding solution for all

Search has no „understanding" =/=> Logic (by using)

Syntax - structure rules / language
Semantik - sense / relates to real world

Logic - allows (formal) knowledge, filter, draw conclusion for AI (used to) represented knowledge

Knowledge Base (KB) - set of sentences about world
Inference Engine - infer new knowledge

Intelligent Agent → goal directed agent
perception knowledge ↦ action

„Asks" it self (KB)
Declarative approach
tell what it needs to know

knowledge-based Agent
Incorporate percepts
Update internal representation
deduce hidden properties
appropiate actions

Observation —Reasoning→ Knowledge

Propositional Logic
Syntax in BNF
Symbol → ...
Sentence → True | False | Symbol | NOT(Sentence) | (Sentence OR Sentence) | (Sentence AND Sentence) | (Sentence => Sentence)

Semantik as truth table for operators

model <=> interpretation in which ∀ sentences true
tautologie <=> sentence true in ∀ settings of propositional vars
≡ Logical equivalent <=> same truth in ∀ settings

Inference / Entailment

sentence entailed by KB - KB => Sentence

Simple Algorithm
Brute Force $O(2^{\#vars})$

⚠ Principle of Non-Contradiction
Inconsistend KB entails ∀ sentences

Reasoning Patterns - obtain new sentences directly from other

+ Modus Ponens $(a, a \Rightarrow b) \Rightarrow b$

Normal formen to reduce complexity

CNF - Conjunctive Normal Form
(... OR ... OR ...) AND (...) ... over Literals
∀ KBs can be written as CNF

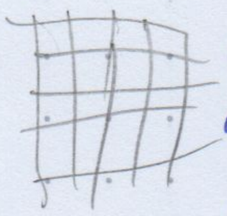
Unit Resolution shown through modus ponens
$(l_1 \text{ OR } l_2 \text{ OR } ..), \text{NOT}(l_i) \Rightarrow (l_1 \text{ OR } ...)$ without l_i

General Resolution
$(l_1 \text{ OR } .. l_k), (m_1 \text{ OR } .. m_n) \Rightarrow (l_1 \text{ OR } .. l_k \text{ OR } m_1 .. m_n)$
with $l_i = \text{NOT}(m_j)$ — without l_i and m_j and duplicates of them

special case
Horn Clauses
implications with only positive Literals.

Examples
english sentences if wet, because rain

wumpus game



find gold
avoid wumpus, pits
adjacent smelly / finds breeze
can shoot once in a field

commutativity
associativity
elimination NOT(NOT(a)) ≡ a
contrapositive
implication elimination
De Morgan
distributivity

Limitations of propositional Logic
No Notion of objects
relations among
z.B. ... hasn't any
... has ... many

Resolution Algorithm
1. Find 2 clauses with complementary literals
2. Apply Resolution
3. Add "
4. Test if Empty clause <=> not satisfiable

Algorithm
follow implication
x_j reachable <=> entailed
more efficient with counter for #impl
vars on LHS reached

First Order Logic - can't reason about Relations or Functions

∀ functions
z.B. no Inheritence of Properties

Elements
- Objects
- Relations
  - unary = properties
- Functions
  - Stelligkeit 0 = Konstante

) refer to Objects without name
⇒ allow infinite Objects
description of

- Equality
- Universal / Existential Quantifier

$\forall x : \varphi(x) \equiv \neg \exists x : \neg \varphi(x)$

Axioms
- Basic facts / initial KB

Theorems
- derived Statements

Applications
z.B. Verification, Synthesis of Programms

forward chaining
- data-driven / Bottom-up
  - make conclusions from known facts

backward chaining
- based on modus ponens / top-down
  - goal-driven
  - z.B. depth-first search for proofs

Substitution - replaces vars (var/replace)

$\forall x : a$ ⇓ any g → SUBST($\{x/g\}$, a)

$\exists x : a$ ⇓ neue const k → SUBST($\{x/k\}$, a)

) Skolem function abhängig von vorherigen $\forall_y$

UNIFY - bring formula together with substitution
keep as general as possible

Resolution - complete

CNF - canonical form — used

sentence entailed ? - semidecidable

1. Negation Normal Form
   - ¬ only before predicate symbols
2. var names not used twice
3. Skolemize drop ∀
   - first move all quantifiers out
4. clause set drop ∧, new implicit

Prolog //skiped in Vorlesung Klausur?

Gödels Incompleteness Theorem

1. FO Logic is not rich enough to model basic arithmetic
2. For any consitent system of axioms that can
   ∃ true sentences : ¬∃ proof from axioms
3. ⇒ can't prove everything

AI

Symbols ↔ Neuro-Symbolic ↔ Neurons

Levels
- Computational - Why does it work? goal? unifying principles?
- Algorithmic - representation?
- Implementational - hardware?

Uncertainty - how to handle?

e.g. Laziness to consider ∀ quanten
Theoretical Ignorance z.B. weather tomorrow
Practical z.B. traffic jam at A5
to hard to find out

we may **Probabilities** - degree of belief (not truth)

Ω - state/sample space $\omega \in \Omega$
$P: \Omega \to [0,1]$ - probability space $\sum_\omega P(\omega) = 1$

Kolmogorov's Axioms
1. $0 \leq P(a) \leq 1$
2. P(false) = 0; P(true) = 1

$A \subseteq \Omega$ - event $P(A) = \sum_{\omega \in A} P(\omega)$

3. $P(a \vee b) = P(a) + P(b) - P(a \wedge b)$

this
4. Restricts sets of reasonable beliefs
└ can be shown by an $ betting example

often sample points defined by = Cartesian Product of variable ranges

X: Events → some range - random variable

$P(X = x_i) = \sum_{\omega : X(\omega) = x_i} P(\omega)$

Aussagen/ Propositions
propositional: a = true, auch a
discrete: $X \in \{x_1, \dots x_n\}$ finite ↔ infinite
$X = x_1$ exhaustive, mutually exclusive
continuous: bounded ↔ unbounded
$x = 1$ oder $x > 1$

Joint Distributions
$P(x,y) = P(X = x \wedge Y = y)$

typical visualization: truth table
uncertain ↔ quantified

Marginalization / Summing (Variable) out
$P(Y) = \sum_{i=1}^{n} P(x_i, Y)$

Conditional probability
$P(x,y) = P(x|y) \cdot P(y) = P(y|x) P(x)$
↑ unter

$P(x|y) = \frac{P(x \wedge y)}{P(y)}$

$P(x_1, \dots, x_n) = \prod_{i=1}^{n} P(x_i | x_1, \dots x_{i-1})$

X and Y independent if
⇔ $P(X|Y) = P(X)$
⇔ $P(Y|X) = P(Y)$
⇔ $P(X,Y) = P(X) P(Y)$

⇒ **Bayes Rule**

x - hypothesis
y - evidence
z.B. für Krankheitstests

$$P(x|y) = \frac{P(y|x) P(x)}{P(y) = P(y|a) P(a) + P(y|\neg a) P(\neg a)}$$

n = # random vars (boolean)

$\Rightarrow$ complexity $O(2^n)$

$\Downarrow$ make use of independence

Naives Bayes model

independent given the cause

$P(\text{hypothesis}, \text{evidence}_1, \dots \text{evidence}_n) = P(\text{hypothesis}) \prod_i P(\text{evidence}_i \mid \text{hypothesis})$ $\Rightarrow$ linear parameters ☺

Hypothesis $\rightarrow$ evidence$_1$, $\rightarrow$ ..., $\rightarrow$ ...

Bayesian Networks - notation for conditional independence

directed acyclic graph

Nodes ~ random variable ~ conditional propability distribution $P(X_i \mid Pa(X_i))$ — $Pa(X_i)$ = Menge parents

Edges - X_i has direct influence on X_j

$\Leftrightarrow (X_i \perp \text{non descendents}) \mid Pa(X_i)$) Local Markov Assumption
($\perp$ unabhängig; non descendents = nicht Nachfahren) — keine indirekten Abhängigkeiten

given $\forall P(X_i \mid Pa(X_i)) \overset{\text{inference}}{\Rightarrow}$ compute $P(Y) = \sum_{\substack{X_1,\dots \text{ in } Pa(Y) \\ x_1 \in X_1 \\ x_2 \in X_2 \\ \dots}} P(Y \mid x_1, \dots) \prod_{j=1}^n P(x_j) = \sum_{x_i \in Y} \prod_{i=1}^n P(X_i \mid Pa(X_i))$

(rekursiv; alles reinziehen; Summe und alle deren Werte)

Algorithm Variable Elimination

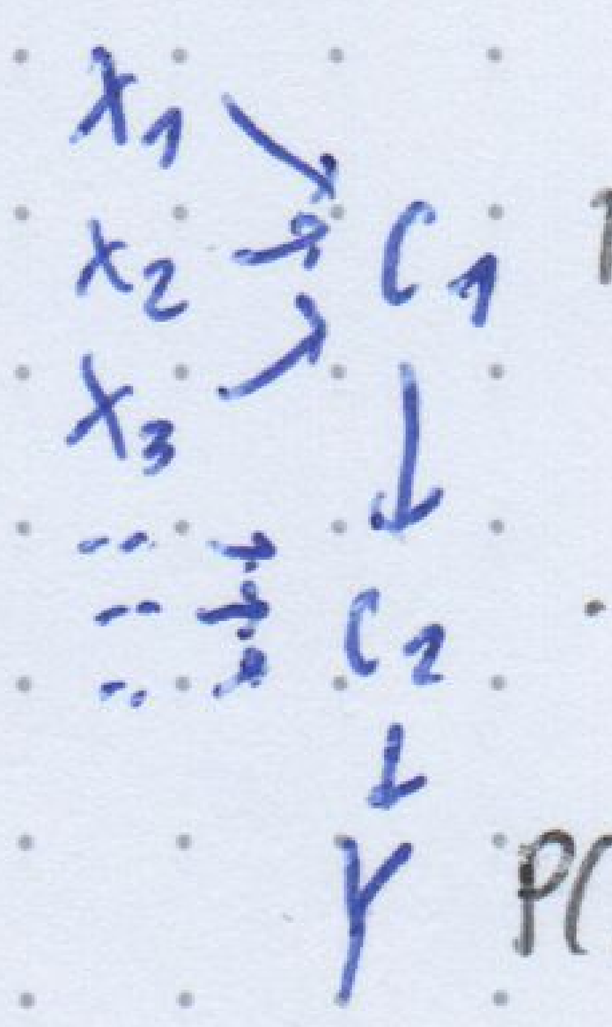
$\left(a \cdot \sum_{b \in \dots} b = \sum ab\right)$

Start with $\forall P(X_i \mid Pa(X_i))$ über Y

Beginnend bei Blättern x alle Vorkommen ersetzen durch $f_x(\text{abhängigkeiten}) = \sum_x (\prod \text{alle Vorkommen})$

Inference is NP-hard (even #P) reduktion durch 3-SAT

$(\bar{X}_1 \vee X_2 \vee X_3) \wedge (\dots$



C_1 True with $p = 1$, wenn C_1 satisfiable

$P(Y = t) > 0 \Leftrightarrow$ 3-SAT formula satisfiable

Approximate Inference - Learning from seen samples

Stochastic Sampling converges to true propability (still NP)

Drew N samples from Sampling distribution S

in Network

sample in topological

use specified conditional probabilities and parent samples as S

Probability Estimation

$\hat{P}_{PS}(x_1 \dots x_n) = N_{PS}(x_1 \dots x_n) / N$

Gibbs Sampling - Markov Chain Monte Carlo (MCMC) Algorithmus

Asking $P(X \mid e)$

Initialize network state x randomly only e always true

N mal

in beliebiger Rheihenfolge $\forall x_i \notin e$ von Markov Blanket sample // x_1

Markov Blanket von Knoten
= parents + children
+ children's parents

given Markov Blanket node conditionally independent of $\forall$ other

$\Rightarrow$ random walk

converges
$\Leftrightarrow$ irreducible - $\forall$ states from $\forall$ reachable
$\wedge$ aperiodic - returns to state can happen at irregular times
$\wedge$ ergodic - return to $\forall$ states mit $p = 1$

BRUNNEN

Machine Ethics

recent AI scale
Deep Neural Networks
but in principle 1950

- Transforms
- Self-Supervised Learning - look if neighbor words find you plausible
- Scale - Internet ⇒ data

Multifusion - multi module inputs

Hybrid AI - deep learning + reasoning

Visual Logical Learning?

Sentient ↔ Stochastic Parrots

~~Modell~~

Letter to Pause 2023
incl. Musk

Potential Problems
Injecting input backdoors in Training

bisher keine kausalen Modelle
z.B für physics

Cultural Bias z.B. auch different letters → diff result
mimics stereotypes
sense of right wrong

filter ↔ in Diffusion Gegensteuern
wieder gegen unerwünschte Trainingsdata
↳ Safe Diffusion
Fair Diffusion - achtet auf Vielfalt

Model of human decision making ↔ Super human?

Computational Ethics framework

⇒ Explanations & Interacting

- (leave one out) influence functions
- constraining their rationals - by also checking a reason given by AI

↳ Revision Transformers (mental) - change values of model

Learning

modify agents decision mechanism ⇒ essential in unknown environments, as system construction method

Memorization / declerative - limited: time, memory
accumulate facts

Generalization / imperative - limited: accuracy prediction
deduce new facts / predict
assumes past ~ future

Inductive Learning
learns function f by examples
target
hypothese $h \approx f$ (Ziel)
consistend ⇔ agrees with example
Ockham's Razor: best = simpels fitting data
Overfitting Avoidance
max combination of both

Machine Learning ⊆ AI but often used synonym

representation, algorithm, preprocessing ← human choice

Aprôh
generate Programm fom inputs, outputs

≠ Human learning (← but borrows ideas)
+ multi modal
+ data, knowledge efficient
- time inefficient

Types of Learning

Supervised - labeld data set $(x_1,y_1),(x_2,y_2),\dots(x_n,y_n)$
learn $h(x)$ = predict y

Regression ↔ Classification Task
↳ y continous ↳ discrete class label
⇒ predict label probability continuous

Unsupervised - unlabeld data
Search for patterns
z.B. task: identify two groups without labeling / correcting it?

Reinforcement - interacting with environment
± reward

Representation
Feature vectors
↳ linear Algebra
independent variables

Feature engineering
select, manipulate, transform raw data → features

common
Multiple linear Regression
Trees
non-linear
polynomial

for Classification:
Random Forest
Decision trees
Logistic Regression
Naïve Bayes
Support Vector Machines

Garbage In, Garbage out
data reflect reality?
distribution?; biased?

Evaluation

Goal (z.B. Spam filter lieber negative)
Accuracy
Precision
Recall
Mean Squared Error

⚠ **Overfitting** - memorized data set not generalized
⇒ regularization
more data with noise
Feature Engineering
cross validation

Split dataset
Training / Testing
stop training when Test gets worse

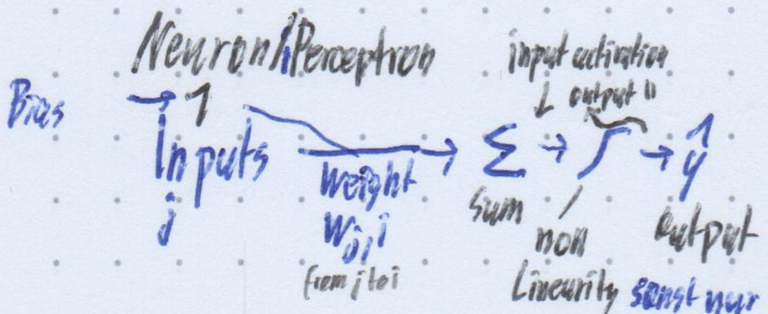
Regularization - set of techniques to prevent Overfitting

BRUNNEN

Artifical Neural Networks ⊆ Maschine Learning

Deep Learning ⊆ Artifical Neuronal Network
↳ works directly on data

Neuron/Perceptron



$\hat{y} = g(w_0 + \sum_{i=1}^{n} x_i w_i)$

$W = \theta =$ All weights

Network = Multilayer Perceptrons (MLP)
with unidirectional information flow / feedforward
can have n outputs

training

~~Backpropagation~~

Loss

$L(f(x^{(i)}; W), y^{(i)})$ goal: W^* minimizing loss

empirical - of dataset

$J(W) = \frac{1}{n} \sum_{i=1}^{n} L(x^{(i)}; W), y^{(i)})$

Backpropagation ~ Gradient Descend

1. init random
2. $W \leftarrow W - \alpha \frac{\partial J(W)}{\partial W}$
   - α: learning rate
   - $\frac{\partial J(W)}{\partial W}$: Gradient
3. repeat 2 till convergence

Beispiel $x_1 \xrightarrow{w_1} z_1 \xrightarrow{w_2} \hat{y} \rightarrow J(W)$

$\frac{\partial J(W)}{\partial w_2} = \frac{\partial J(W)}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_2}$

Ableitung f: $g'(\hat{y}) \cdot z_1$

$\frac{\partial J(W)}{\partial w_1} = \frac{\partial J(W)}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_1}$

1

BRUNNEN

Reinforcement Learning (RL)

not known to agent

Agent ⇄ Action A_t / Reward R_t / State S_t ⇄ Environment

typically → MDP - Markov Decision Process

- given present: future and past independent

(S, A, T_a, R_a)

S - set of states

A - " actions

T_a - transition function

$T(s, a, s') = P(s' \mid s, a)$

R_a - reward function

$R(s, a, s')$

start state / distribution

discount factor γ

optional terminal state

- defines reflex agent

⇒ optimal policy $\pi^*: S \to A$

deterministic $a_t = \mu(s_t)$

stochastic $a_t \sim \pi(\cdot \mid s_t)$

maximizing $G_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}$

- G_t: already taken time
- utility / return / Nützlichkeit
- Discount factor ≤ 1: helps ensure convergence

CAP - Credit Assignment Problem

temporal - given sequence

What action produced reward

Value Function

$V^*(s)$ = expected utility starting in s, thereafter acting optimally

Action-Value Function

$Q^*(s, a)$ = expected utility starting s, taking a, thereafter acting optimally

π^* exists and V share these functions

Bellman Equation of Optimality

$$V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s'} T(s' \mid s, \pi(s)) V^\pi(s')$$

finding optimal policy knowing MDP

not RL

Value Iteration

$V_0^*(s) = 0$

$$V_{k+1}(s) = \max_a \sum_{s'} T(s, a, s') + \left(R(s, a, s') + \gamma V_k(s')\right)$$

↓ jetzt von a, s' abhängig

Sonst Bellmann Equation

- value / Bellman update $O(S^2 A)$

⇒ converges to optimal values

π^* may converge long before

model based ↔ free

On- ↔ Off- policy — Exploitation ↔ Exploration

- learns about executing policy
- learns different policy than executed

Passive ↔ Active Learning

- execute given policy

Model-based Learning passiv

learn model empirical
(count outcomes, normalize)
solve learned as if correct
z.B. iterative policy evaluation
$V^{\pi}_{i+1}(s) \leftarrow \sum_{s'} T(s,\pi(s),s')\left(R(s,\pi(s),s') + \gamma V^{\pi}_{i}(s')\right)$

optimization
⇒ Sample-Based Policy Evaluation

$sample_1 = R(s,\pi(s),s'_1) + \gamma V^{\pi}_{k}(s'_1)$
$sample_2 = R(s,\pi(s),s'_2) + \gamma V^{\pi}_{k}(s'_2)$
...
$sample_n$

$V_{k+1}(s) \leftarrow \frac{1}{n} \sum_i sample_i$

TD-Temporal-Difference Learning passiv, model free for Policy Evaluation

@Vexperience
update V
$V^{\pi}(s) \leftarrow (1-\alpha) V^{\pi}(s) + \alpha \cdot sample$
can't create new policy from that ?

learn & fit
learn Q directly

Active Learning full RL, exploration ↔ exploitation

force exploration

Q-Learning $Q_{k+1}(s,a) = \sum_{s'} T(s,a,s') \cdot \left(R(s,a,s') + \gamma \max_{a'} Q_k(s',a')\right)$

$sample = R(s,a,s') + \gamma \max_{a'} Q(s',a')$
$Q(s,a) \leftarrow (1-\alpha) Q(s,a) + \alpha \, sample$

+ converges if
explored enough
learning rate small enough
- needs to store all Q-values

ε-greedy
with
$p = \varepsilon$ ⇒ act random
sonst ⇒ according to π
- keeps thrashing when learning done
⇒ lower ε
exploration functions

Policy Search

learn policy directly

+ no model
as Q-values

parameterize policy
gradient optimization

DQN - Deep Q-Networks

Q-learning with deep neural network
function approximator
needs discrete, finite A
Use ε-greedy

Beispiel AlphaGo

Monte Carlo Tree Search
Value & Policy network
self play; RL: Learning

Planning — White Box

set of actions
initial state
goal
⇒ sequence of actions

as set of sentences (first order/relational logic)

explicit representation ↔ black box in problem solving
↳ may allow planning subgoals independently
(search plan space not state space)

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

states, goals ~ Λ literals
actions ~ rules
plan ~ proof of goal

situation calculus - state in situation variables
⇒ allows KB to be constant

Planning ~ Inferenz

action sequences ~ nested function expressions

Rule
head :- Cond, ...

Frame Problem

what parts are relevant?

how to specify what doesn't change by a rule

representational: frame axioms for all ⇒ too many
inferential: to compute complex

Qualification Problem

Limit of preconditions for actions?
many small obvious risks

Ramification Problem

∀ effects from action?

Complexity exponential

Optimality: can only find plan
proof optimality?

↓ not all FO
einfacher

STRIPS - endlich

CWA - Closed World assumption
not known true → assumed false

state = Λ literals

```
Action( Prädikat
        func(...) )
Precond: ...
Add: facts true after
Delete: " no longer true
)
```

Progression ↔ Regression planners
forward backward