Model-free, Model-based, and General Intelligence

Hector Geffner
ICREA & Universitat Pompeu Fabra
Barcelona, Spain

IJCAI-ECAI 2018

ICREA

upf. Universitat Pompeu Fabra Barcelona

Outline

- AI, Programming, and AI programming
- Problem of Generality
- Model-free Learners
- Model-based Solvers (Planners)
- Learners and Solvers: System 1 and System 2?
- Learners and Solvers: Need for Integration, Challenges

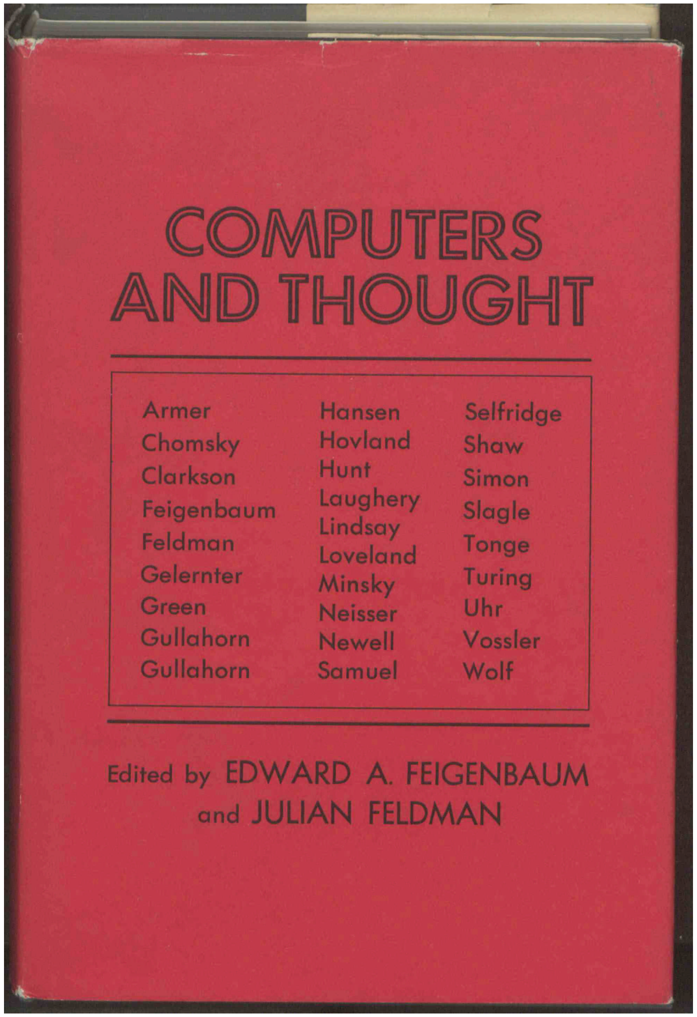
Outline

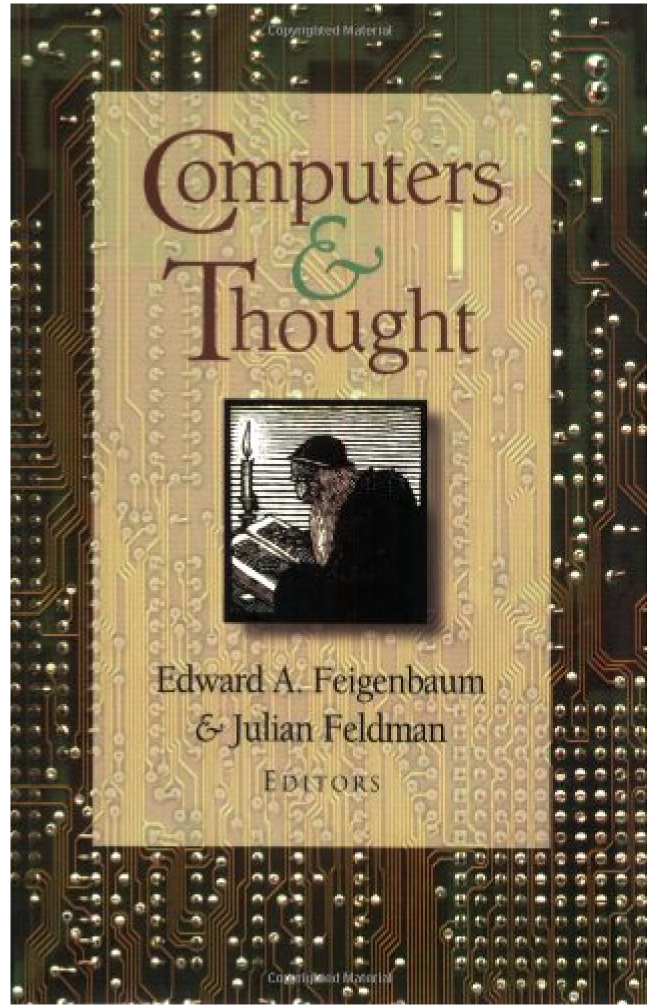
- AI, Programming, and AI programming
- Problem of Generality
- Model-free Learners
- Model-based Solvers (Planners)
- Learners and Solvers: System 1 and System 2?
- Learners and Solvers: Need for Integration, Challenges

Refs: *Model-free, Model-based, and General Intelligence.* H. Geffner, 2018.

Thanks: B. Bonet, G. Francès, N. Lipovetzky, M. Ramírez, H. Palacios, . . .

Computers and Thought (1963)





Early collection of AI papers describing programs for playing chess and checkers, proving theorems in logic and geometry, planning, etc.

Importance of Programs in Early AI Work

In preface of 1963 edition of the book:

We have tried to focus on papers that report results. In this collection, the papers . . . describe actual **working computer programs** *. . . Because of the limited space, we chose to avoid the more speculative . . . pieces.*

In preface of 1995 AAAI edition

A critical selection criterion was that the paper had to describe . . . **a running computer program** *. . . All else was talk, philosophy not science . . . (L)ittle has come out of the "talk".*

AI, Programming, and AI Programming

Many of the key AI contributions in 60s, 70s, and early 80s had to do with **programming** and the **representation of knowledge** in **programs**:

- Lisp (Functional Programming)
- Prolog (Logic Programming)
- Rule-based Programming
- Interactive Programming Environments and Lisp Machines
- Frame, Scripts, Semantic Networks
- Expert Systems Shells and Architectures
- . . .

Programming and Problem of Generality

- For writing an AI dissertation in the 60s, 70s and 80s, it was common to:
  - ▷ pick up a task and domain X
  - ▷ analyze/introspect/find out how task is solved
  - ▷ capture this reasoning in a program
- The dissertation was then
  - ▷ a **theory** about X (humor, story understanding, analogy, etc), and
  - ▷ a **program** implementing the theory, **tested** over a few examples.
- Great ideas came out from this work but . . . **a methodological problem:**
  - ▷ Programs written by hand were **not robust or general**

From Programs to Learners and Solvers

- Limitation led to **methodological shift**:
  - from writing **programs for ill-defined problems** . . .
  - to designing **algorithms for well-defined mathematical tasks**

From Programs to Learners and Solvers

- Limitation led to **methodological shift**:
  - from writing **programs for ill-defined problems** . . .
  - to designing **algorithms for well-defined mathematical tasks**

- New general programs **learners** and **solvers** have a **crisp functionality**: both can be seen as computing **functions** that map inputs into outputs

$$\textit{Input } x \Longrightarrow \boxed{\textsc{Function } f} \Longrightarrow \textit{Output } f(x)$$

- The algorithms are **general** in the sense that they are not tied to particular examples but to classes of **models** and **tasks** expressed in **mathematical form**

Learners

$$\textit{Input } x \Longrightarrow \boxed{\textsc{Function } f} \Longrightarrow \textit{Output } f(x)$$

- In **deep learning (DL)** and **deep reinforcement learning (DRL)**, training results in function f_θ

- f_θ given by structure of **neural network** and adjustable parameters θ

  ▷ In DL, **input** x may be an image and **output** $f_\theta(x)$ a classification label
  ▷ In DRL, **input** x may be state of game, and **output** $f_\theta(x)$, value of state

- Parameters θ learned by **minimizing error function**

  ▷ In DL, error depends on inputs and target outputs in training set
  ▷ In DRL, error depends on value of states and successor states

- Most common **optimization algorithm** is **stochastic gradient descent**

Learners: Success and Limitations

$$\textit{Input } x \Longrightarrow \boxed{\textsc{Function } f} \Longrightarrow \textit{Output } f(x)$$

- Excitement about AI due to **successes in DL and DRL**
  - ▷ Breakthroughs in image understanding, speech recognition, Go, . . .
  - ▷ Superhuman performance in Chess and Go from **self-play** alone
- The basic ideas underlying DL and DRL not new but from 80s and 90s
  - ▷ Recently, more CPU power, more data, deeper nets, attractive problems
- One key limitation: **Fixed input size** x
  - ▷ No problem for learning to play Chess or Go over **fixed size board**
  - ▷ But critical for tackling **arbitrary instances** of . . . **Blocks World**

Solvers

$$\textit{Input } x \Longrightarrow \boxed{\textsc{Function } f} \Longrightarrow \textit{Output } f(x)$$

- **Solvers** derive output $f(x)$ for **given input** x from **model**:
  - ▷ **SAT:** x is a formula in CNF, $f(x) = 1$ if x satisfiable, else $f(x) = 0$
  - ▷ **Classical planner:** x is a planning problem P, and $f(x)$ is plan that solves P
  - ▷ **Bayesian net:** x is a query over Bayes Net and $f(x)$ is the answer
  - ▷ **Constraint satisfaction, Markov decision processes, POMDPs, . . .**

- **Generality:** Solvers not tailored to particular examples
- **Expressivity:** Some models very expressive, "AI-Complete" (POMDPs)
- **Complexity:** Computation of $f(x)$ is (NP) hard; $|x|$ **not bounded**
- **Challenge:** Solvers shouldn't break just because x has many variables
- **Methodology:** Empirical, benchmarks, competitions, . . .

Solvers vs. Learners

$$\textit{Input } x \Longrightarrow \boxed{\textsc{Function } f} \Longrightarrow \textit{Output } f(x)$$

- **Learners** require **experience over related problems** x but then fast
  - ▷ They compute function f from training, then apply it
- **Solvers** deal with **completely new problems** x but need **to think**
  - ▷ They compute $f(x)$ for each input x from scratch

Thinking is hard but **computational limits** are important source of insight

Next: look at some powerful computational ideas in **planning**

Finding Plans in Huge Mazes: Relaxation, Heuristics

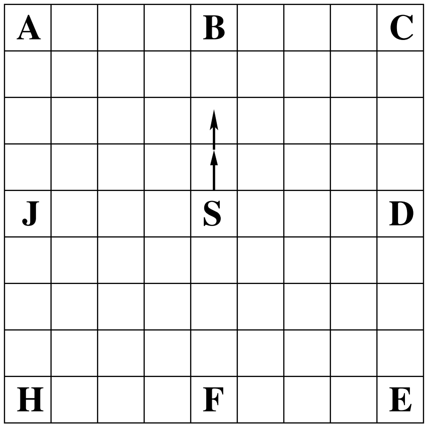
Old Idea: If you don't know how to solve P, **solve simpler problem** P', and use solution of P' for solving P (Polya, Minsky, Pearl)

- In **monotonic relaxation** P', effects of actions on variables made **monotonic**
- Monotonicity makes relaxation P' **decomposable** and therefore **tractable**
- **Heuristic** $h(s)$ in P set to **cost of plan from** s **in relaxation** P'

Heuristic obtained and used to solve **any problem** P *from scratch*

No experience required *in problems related to* P

Goal Recognition: A Classification Problem

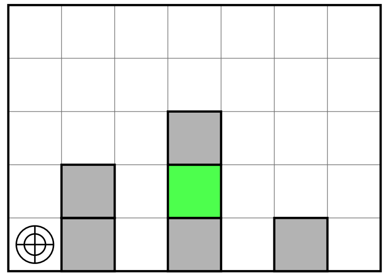


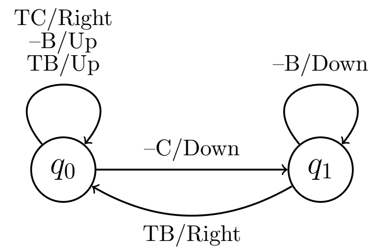
- **Task:** infer **agent goal** $G \in \mathcal{G}$ from **observations** O on behavior
- Bayes' rule: $P(G|O) = P(O|G)\,P(G)/P(O)$, priors $P(G)$ assumed given
- Likelihood $P(O|G)$ set as **monotonic function** f of difference between:
  - ▷ $c^-(G)$: cost of reaching G with plan that does not comply with observations
  - ▷ $c^+(G)$: cost of reaching G with plan that complies with observations

$P(G|O)$ *computed using* **Bayes' rule** *and* $2|\mathcal{G}|$ **calls to planner**

No experience required *in related problems*

Generalized Planning and One-Shot Learning





- **Task:** move 'eye' (mark) one cell at a time til **green block** found
- **Observables:** Whether marked cell contains a green block (G), non-green block (B), or neither (C); and whether on table (T) or not (–)
- **Controller** derived using **classical planner** over **transformed problem** where
  - ▷ one action $b = \langle q, o, a, q' \rangle$ for each possible controller edge
- **Generality:** Derived controller solves not just given instance but **any** instance; i.e., **any number of blocks** and **any configuration**

Generalized plan for problem x is not $f(x)$ but function f itself

Polynomial Algorithms for Exponential Spaces: Width

- IW(1) is a **breadth-first search** that **prunes** states s that don't make a feature true for first time in the search, from given **set of boolean features** F

- IW(k) is IW(1) but over set F^k made up of conjunctions of k features from F
  - ▷ Most domains have **small width** $w \leq 2$ when **goals are single atoms**
  - ▷ **Any** such instances solved **optimally** by IW(w) in **low poly time**

Polynomial Algorithms for Exponential Spaces: Width

- IW(1) is a **breadth-first search** that **prunes** states s that don't make a feature true for first time in the search, from given **set of boolean features** F
- IW(k) is IW(1) but over set F^k made up of conjunctions of k features from F
  - ▷ Most domains have **small width** $w \leq 2$ when **goals are single atoms**
  - ▷ **Any** such instances solved **optimally** by IW(w) in **low poly time**
- IW(k) can work with **simulators**. No PDDL or goal needed. **Variants:**
  - ▷ BFWS(R): SOTA planning algorithm which doesn't use **action structure**
  - ▷ Rollout IW(1): fast **on-line planner** that plays Atari from **screen pixels**

Learners and Solvers: Contrasts

- Rollout IW(1) **planner** and DQN **learner** perform comparably well in Atari
- They illustrate **key difference between learners and solvers:**
  - ▷ DQN requires lots of training data and time, and then plays very fast
  - ▷ Rollout IW(1) plays out of the box but thinking a bit before each move

Learners and Solvers: Contrasts

- Rollout IW(1) **planner** and DQN **learner** perform comparably well in Atari
- They illustrate **key difference between learners and solvers:**
  - ▷ DQN requires lots of training data and time, and then plays very fast
  - ▷ Rollout IW(1) plays out of the box but thinking a bit before each move

This is a general characteristic:

- **Learners** require **experience over related problems** x but then are fast
  - ▷ They compute function f from training, then apply it
- **Solvers** deal with **completely new problems** x but need **to think**
  - ▷ They compute $f(x)$ for each input x from scratch

Learners and Solvers: System 1 and System 2?

Dual process accounts of the human mind assume two processes (Daniel Kahneman: Thinking, Fast and Slow):

System 1 (Intuitive Mind)	**System 2** (Analytical Mind)
fast	slow
associative	deliberative
unconscious	conscious
effortless	effortful
parallel	serial
specialized	general
. . .	. . .
Learners?	Solvers?

Learners and Solvers: Challenges (1)

- **Key challenge:** General **two-way integration** of System 1 and System 2 inference in AI systems; i.e. **learners** and **solvers**

- **AlphaZero** that learns Chess and Go by pure self-play is **effective integration of a learner and a solver**

  ▷ AlphaZero learns by imitating and improving (MCTS) planner used as teacher

- Yet AlphaZero can do Chess but not **much simpler** . . . Blocks World

Learners and Solvers: Challenges (1)

- **Key challenge:** General **two-way integration** of System 1 and System 2 inference in AI systems; i.e. **learners** and **solvers**
- **AlphaZero** that learns Chess and Go by pure self-play is **effective integration of a learner and a solver**
  - ▷ AlphaZero learns by imitating and improving (MCTS) planner used as teacher
- Yet AlphaZero can do Chess but not **much simpler** . . . Blocks World
  - ▷ "Doing" BW is near **100% coverage on arbitrary instances with general algorithm**; **not** 68% coverage on selected instances with 7 blocks!

Learners and Solvers: Challenges (2)

For **general and synergistic integration of learners and solvers**:

- **Learning the state variables** from streams of actions and observations
- **Learning useful general features** for planning
- **Model learning:** explanation and accountability require models
- **Learning finite-size abstract representations** for general plans

. . .

AI: Dreams and Nightmares. Systems 1 and 2 Again

AI: Dreams and Nightmares. Systems 1 and 2 Again

- AI **far from human-level intelligence** yet can be used for good or ill

AI: Dreams and Nightmares. Systems 1 and 2 Again

- AI **far from human-level intelligence** yet can be used for good or ill
- **Asilomar AI principles** good and timely but difficult to enforce

AI: Dreams and Nightmares. Systems 1 and 2 Again

- AI **far from human-level intelligence** yet can be used for good or ill
- **Asilomar AI principles** good and timely but difficult to enforce
- **AI aligned with human values** nice but why not tech, politics, economics?

AI: Dreams and Nightmares. Systems 1 and 2 Again

- AI **far from human-level intelligence** yet can be used for good or ill
- **Asilomar AI principles** good and timely but difficult to enforce
- **AI aligned with human values** nice but why not tech, politics, economics?
- **Markets and politics** focused on **bottom line** and aimed at our **System 1**

AI: Dreams and Nightmares. Systems 1 and 2 Again

- AI **far from human-level intelligence** yet can be used for good or ill
- **Asilomar AI principles** good and timely but difficult to enforce
- **AI aligned with human values** nice but why not tech, politics, economics?
- **Markets and politics** focused on **bottom line** and aimed at our **System 1**
- Life in modern world needs **System 2 informed by facts and common good**

AI: Dreams and Nightmares. Systems 1 and 2 Again

- AI **far from human-level intelligence** yet can be used for good or ill
- **Asilomar AI principles** good and timely but difficult to enforce
- **AI aligned with human values** nice but why not tech, politics, economics?
- **Markets and politics** focused on **bottom line** and aimed at our **System 1**
- Life in modern world needs **System 2 informed by facts and common good**
- If we want **good AI**, we need a **good and decent society**