Top-down representation learning for acting and planning

ELLIIT, Linköping, 31/10/2022

Hector Geffner
ICREA & Universitat Pompeu Fabra
Barcelona, Spain

Wallenberg Guest Professor
Linköping University, Sweden

with Blai Bonet and RLeap/RLP Team

Model-free 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

- Function 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** by stoch. gradient descent
  - ▷ In DL, error depends on inputs and target outputs in training set
  - ▷ In DRL, error depends on value of states and successor states

- A true revolution in AI still **unfolding**

- **Limitation:** transparency, amounts of data, OOD generalization, understanding

Model-based 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; e.g., POMDPs
- **Learners are solvers too:** $\text{argmin}_w \sum_{x \in D} L(x, f_w(x))$ (Diff. programming)
- **Challenge:** Scalability; computation of $f(x)$ is NP-hard
- **Limitation:** Models must be known and in the "right" language

Model-free Learners vs Model-based Solvers

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

- **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

Learners and Solvers: System 1 and System 2?

Dual process accounts of the human mind assume two processes (D. Kahneman: Thinking, Fast and Slow, 2011; K. Stanovich: The Robot's Rebellion, 2005)

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

Key Challenge in AI

- General **two-way integration** of System 1 and System 2 inference in AI systems
  - ▷ **Learn representations** that support reasoning and are reusable

- **Yoshua Bengio**'s challenges reflected in title of his IJCAI 2021 talk:
  - ▷ *System 2 Deep Learning: Higher-level cognition, agency, out-of-distribution generalization and causality*

- **Yann LeCun**'s three challenges, AAAI 2020:
  - ▷ AI must learn to represent the world
  - ▷ AI must think and plan in ways compatible with gradient-based learning
  - ▷ AI must learn hierarchical representation of action plans

Bottom-Up vs. Top-Down Representation Learning

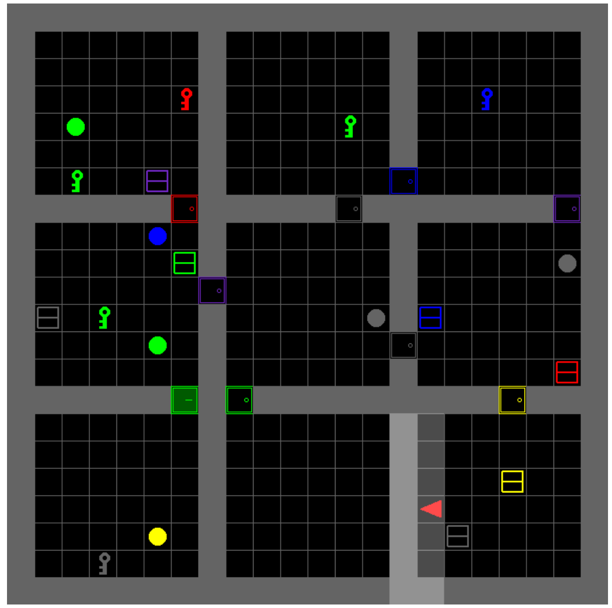
- **Bottom-up approach** (most common in deep learning)
  - ▷ Representations emerge from **architecture**, loss function, and "right" bias

- **Top-down approach**
  - ▷ Representations learned over **language** with "right" syntax and semantics
  - ▷ Meaningful learning bias, transparency, reasoning, **what** vs. **how**

In line with **"traditional AI":** just **learn** *from data the representations that have traditionally been crafted by hand*

Related but different than **neuro-symbolic AI** where representation languages used to encode **background knowledge**

Example: Minigrid/BabyAI [Chevalier-Boisvert *et al.*, 2019]



- **Task:** *Pick up grey box behind you, then go to grey key and open door*
- Red triangle is agent at bottom right. Light-grey is field of view
- Learn **controller** that accepts **goals** and **obs**, and outputs **action** to do
- Like a "classical planning problem" **but** state representation **not known**, and goals to be achieved **reactively** (not by planning) with policies that **generalize**

Bottom up vs. Top-Down Representation Learning

- Surprise is not that DL and DRL methods struggle in Minigrid, but that they manage to generate meaningful behavior at all, given **so little prior knowledge**

- Yet **methodology** largely **ad hoc:** from intuitions to **architectures** and **experiments** using baselines; performance improvements but **no crisp understanding**

- From perspective of language-based representation learning, **key questions** are:
  - ▷ What are the **domain-independent languages** for representing **dynamics**?
  - ▷ What are the **languages** for representing general **policies**, **subgoals**?
  - ▷ How **representations** over such languages can be **learned**?

Three Concrete Challenges

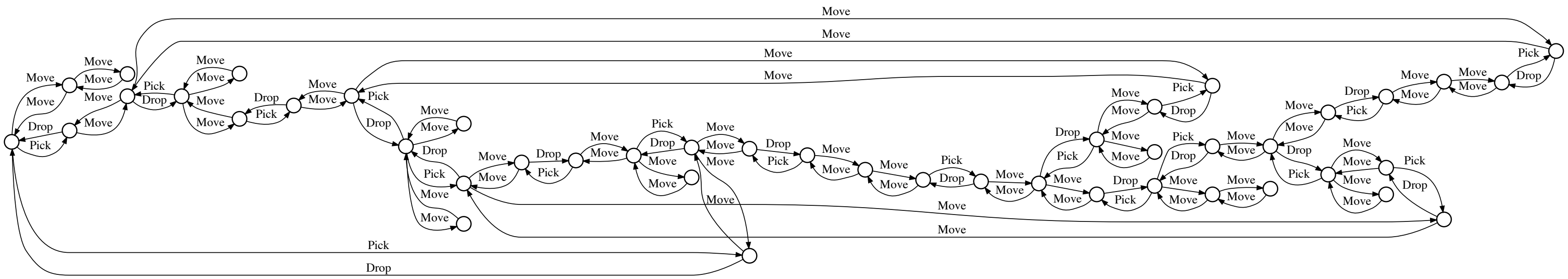
Three **tasks** in **language-based representation learning** for *acting* and *planning*:

- learn **general dynamics** (predicates and action schemas)
- learn **general policies** (general action strategies)
- learn **general subgoal structures** (intrinsic rewards, reward machines)

Language taken **off-the-shelf** in first case; **carefully designed** in the other two

Learning Task #1: Learning Dynamics from State Graphs

Input: State graph G of agent in $1{\times}3$ grid, moving/picking/dropping 2 pkgs



Output: Simplest STRIPS representation $P = \langle D, I \rangle$ that **generates** G

```
Move(?to,?from):
  Pre: neq(?to,?from), p5(?to,?from)
  Pre:  p2(?from), -p2(?to)
  Eff: -p2(?from),  p2(?to)

Pick(?p,?x):
  Pre:  p2(?x),  p1, -p3(?p),  p4(?p,?x)
  Eff:          -p1,  p3(?p), -p4(?p,?x)

Drop(?p,?x):
  Pre:  p2(?x), -p1,  p3(?p), -p4(?p,?x)
  Eff:           p1, -p3(?p),  p4(?p,?x)
```

Interpretation of learned predicates:

– p_1: gripper empty

– $p_2(x)$: agent at cell x,

– $p_3(p)$: agent holds pkg p,

– $p_4(p, x)$: pkg p in cell x

– $p_5(x, y)$: cell x adj to y

- Domain D correct for **any** grid, **any** $\#$ of packages. Structure of nodes uncovered.

Learning Task #1: Dynamics. Formulation

- $P = \langle D, I \rangle$ defines unique **state graph** $G(P)$
- Learning as **inverse task:** from graphs $G_1, \ldots, G_k$, learn problems $P = \langle D, I_i \rangle$:

> Given graphs $G_1, \ldots, G_k$, find **simplest** instances $P_i = \langle D, I_i \rangle$ such that graphs G_i and $G(P_i)$ are isomorphic, $i = 1, \ldots, k$.

- **Problem** cast and solved as combinatorial optimization task [Bonet and G., 2020]
- **Complexity** of P_i determined by $\#$ and arities of action schemas and predicates
- **Variations:** noisy graphs, gray-box states [Rodriguez *et al.*, 2021, Occhipinti *et al.*, 2022]

(**Open:** How to solve (a version of) this problem using **DL/gradient descent?**)

Learning Task # 2: General Policies

- **General policy** is general **strategy** for solving **multiple** domain instances
  - ▷ E.g., Move all packages to target cell; **any** grid size, **any** # of pkgs
- The two basic problems for **language-based representation learning**:
  - ▷ What are good **languages** for representing **general policies**?
  - ▷ How to **learn** such policies in language from sampled instances?
- **Learning general policies** also a key goal in **deep reinforcement learning**

Language and Semantics of General Policies [Bonet, G., 2018]

- **Example:** Move packages in $n \times m$ grid, one by one, to target location
- **Features** $\Phi = \{H, p, t, n\}$: hold, dist. to nearest pkg & target, $\#$ undelivered
- **General policy** π: **any** $\#$ of pkgs and distribution, **any** grid size

$\{\neg H, p > 0\} \mapsto \{p\downarrow, t?\}$	go to nearest package
$\{\neg H, p = 0\} \mapsto \{H, p?\}$	pick it up
$\{H, t > 0\} \mapsto \{t\downarrow, p?\}$	go to target cell
$\{H, t = 0\} \mapsto \{\neg H, n\downarrow, p?\}$	drop package

Learning General Policies: Formulation

Given a known domain D, training instances $P_1, \ldots, P_k$, over D, and a **finite pool of domain features** $\mathcal{F}$, each with a cost, find min-cost policy π over $\mathcal{F}$ such that π solves all P_i, $i = 1, \ldots, k$

- Problem cast and solved as **combinatorial opt. task** [Francès *et al.*, 2021]
- Pool of **features** $\mathcal{F}$ generated from domain predicates using **2-variable** (description) logic grammar; feature cost given by syntax tree size
- **Deep learning** approaches also used [Toyer *et al.*, 2018; Garg *et al.*, 2020]; they don't need explicit pool $\mathcal{F}$ but not 100% correct in general
- Recent DL approach also avoids $\mathcal{F}$ and nearly 100% correct when **2-variable logic** features suffice [Ståhlberg *et al.*, 2022a and 2022b]; exploits relation between **GNNs** and 2-variable logic [Morris *et al.*, 2019; Barceló *et al.*, 2020; Grohe 2021]

Learning Task #3: Learning Subgoal Structure. Sketches

- **Width=2** Sketch:

$\{n > 0\} \mapsto \{n\downarrow\}$	deliver package

- **Width=1** Sketch:

$\{\neg H\} \mapsto \{H\}$	go and pick package
$\{H\} \mapsto \{\neg H, n\downarrow\}$	go and deliver package

- **Width=0** Sketch (full policy)

$\{\neg H, p > 0\} \mapsto \{p\downarrow, t?\}$	go to nearest package
$\{\neg H, p = 0\} \mapsto \{H, p?\}$	pick it up
$\{H, t > 0\} \mapsto \{t\downarrow, p?\}$	go to target cell
$\{H, t = 0\} \mapsto \{\neg H, n\downarrow, p?\}$	drop package

Features: Holding (H); Dist. to nearest Pkg (p), Target (t); # Undeliv Pkgs (n)

Sketch Width

- Sketch R **splits** problems P in class $\mathcal{Q}$ into **subproblems** $P[s, G_R(s)]$:

- **Width of problem** $w(P)$ bounds its **complexity** [Lipovetzky and G., 2012]

- **Width of sketch** R over $\mathcal{Q} = \max_{s, P \in \mathcal{Q}} \mathrm{w}(P[s, G_R(s)])$

Theorem: If sketch R is **terminating** and has **width** over $\mathcal{Q}$ bounded by k, then P in $\mathcal{Q}$ is **solvable** in $O(N^{|\Phi|+2k-1})$ time [Bonet and G., 2021]

▷ N: *Number of atoms in* P ; Φ: *Features in sketch* ; b: *branching factor*

Learning Sketches: Formulation [Drexler *et al.*, 2022]

Given a known domain D, training instances $P_1, \ldots, P_n$, a pool of features $\mathcal{F}$, and a non-negative integer k, find **min-cost sketch** R over $\mathcal{F}$ such that

- Subproblems induced by R on each P_i have all **width bounded** by k,
- Sketch R is **terminating** (structurally acyclic)

Possibly first approach for **learning subgoal structure** based on crisp principles

Many threads come together for this:

- Planning **width** [Lipovetzky and G., 2012]
- Language of **general policies** [Bonet and G., 2018]
- Semantics of **sketches** [Bonet and G., 2021]
- Termination notion from **QNPs** [Srivastava, Zilberstein, Immerman, G., 2011]

Summary: Top Down Representation Learning

- **Model-free** learners and **model-based** solvers like **Systems 1 and 2**
- **DL** and **DRL** deliver **System 1** boxes only
- **Main challenge** is two-way integration **learners** and **solvers**
- **Key problem** is **learning representation used by solvers**
- **Language-based representations learning** for acting and planning:
  - ▷ **What** to learn: **dynamics**, **policies**, **subgoal structure**
  - ▷ **How** to learn: comb. optimization, continuous optimization (**deep learning**)
  - ▷ **Inputs:** symbolic states, black box, images, parsed images, ...
- **Potential benefits** of learning-based representation learning:
  - ▷ semantics, meaningful bias, transparency, distinction **what** and **how**

Challenges: Language-based Representation Learning

- **Scalability** of combinatorial optimization approaches
- Use of **deep learning** (learning lifted dynamics, policies, sketches).
- **Continuous** actions, state space, time
- Learning **hierarchical policies**; learning and reusing "skills"
- **Stochastic** and **non-deterministic** domains
- **Grounded** vs. ungrounded representations
- **Inputs**: states as black-boxes, parsed images, images; goals in nat language ...
- . . .