Never-Ending / Continual Reinforcement Learning

Doina Precup

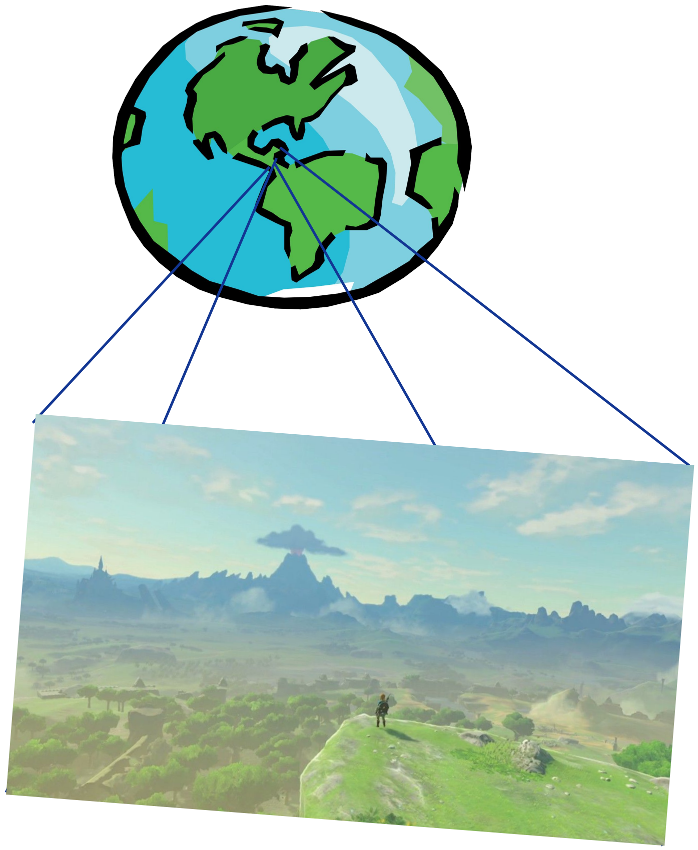
With thanks to Rich Sutton, Satinder Singh, David Silver, David Abel, Michal Bowling, Andre Barreto, Gheorghe Comanici, Daniel Toyama, Anita Gergely, Mia Glaese, Philippe Hamel, Shibl Mourad, Tyler Jackson, Adam White, Marlos Machado, Khimya Khetarpal, Zafarali Ahmed

RL Course

From Reinforcement Learning to General AI Agents

- Growing knowledge and abilities in an environment
- Learning efficiently from one stream of data
- Reasoning at multiple levels of abstraction
- Adapting quickly to new situations

Today's Perspective



High-Level View of Agent

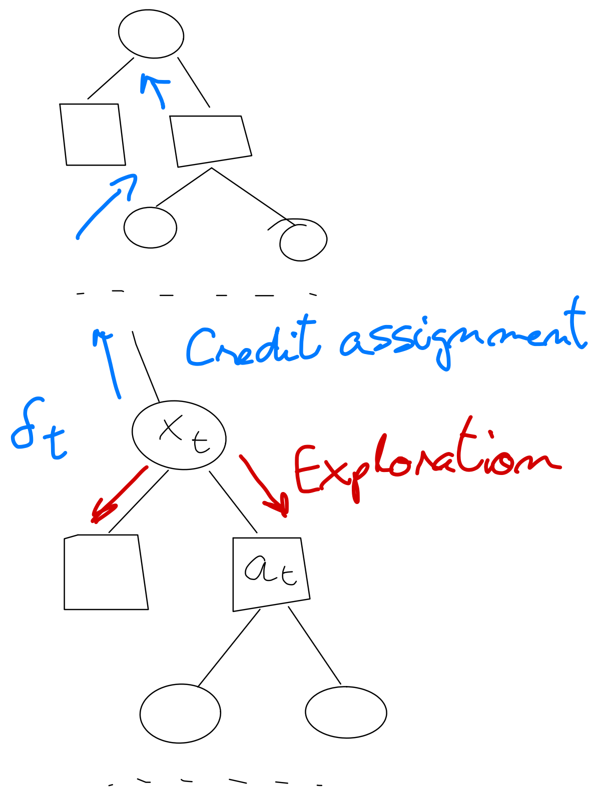
- Agent has *one stream of experience (observations, actions, rewards)* to support all learning processes
- *Agent is "smaller" than the entire environment*
  - Only has time to travel on a specific trajectory
  - Cannot compute arbitrarily fast or remember all the relevant experience in a replay buffer
- *Asynchronous learning*
  - The world moves at its own speed
  - Agent has a time scale at which it can perceive, act and learn
  - Agent can also choose the time scale at which it updates its representation

Should We Think This Way?

- Yes!
  - Naturalistic perspective: the conditions in which intelligence has developed in the natural world
  - Realistic perspective: the onus is on the agent to do well *given its current circumstances*
  - Natural for AGI, but also consistent with real applications like robotics, health care, energy management...
- No!
  - Are we handicapping ourselves too much?
  - Does this perspective go against the Bitter Lesson?
- Next: explore the implications of this approach on algorithmic solutions

Sequential decision making

- At time t, agent receives an observation from set $\mathcal{X}$ and can choose an action from set $\mathcal{A}$ (think finite for now)
- Goal of the agent is to maximize long-term return



Some observations

- We usually think of the infinite tree of all possible observations and actions
- Today: focusing on one specific path through the tree
- If there is no structure (ie every node is completely different), there is nothing interesting to learn!
- Markovian assumption: trajectories through the tree *cluster into equivalence classes*, which we call states
- This allows many ways of doing credit assignment: TD(0), TD(λ), Monte Carlo
- Because we cluster an infinite tree into a finite number of clusters, it makes sense to make *recurrence assumptions*: states will be revisited

An example of non-Markovian structure

- Linear predictive state representations (Littman et al, 2001, Singh et al, 2004)
- Make a systems dynamics matrix, with histories as rows and future sequences as columns
- Assume *systems dynamics matrix has finite rank*
- One can show that POMDPs, k-order Markov models are equivalent to linear PSRs

“Small Agent” Perspective

- Agent’s trajectory will cover a minuscule fraction of all possible trajectories
- Notions of recurrence like in MDPs no longer make sense (the agent is really transient)
- Yet the agent still needs to do as well as possible *along its current trajectory*
- So it needs to *construct a knowledge representation that allows it to generalize quickly*
- *Agent state:* the internal representation used by the agent to predict and act
- Agent state will have to be learned
- *The representation will inherently be lossy/imperfect*

An Evolution of Ideas

- Dynamic programming: agent needs to find an optimal policy at all states
- Reinforcement learning: agent focuses on states that are actually encountered during its experience

  This is what allows tackling large environments like Go!
- One step further: agent's learning should enable it to do well in the future on the trajectory that will be encountered!
- *Optimality is not an absolute notion*, but relative to the agent's circumstances, available data and capacity
- Eg child cooking at home vs chef

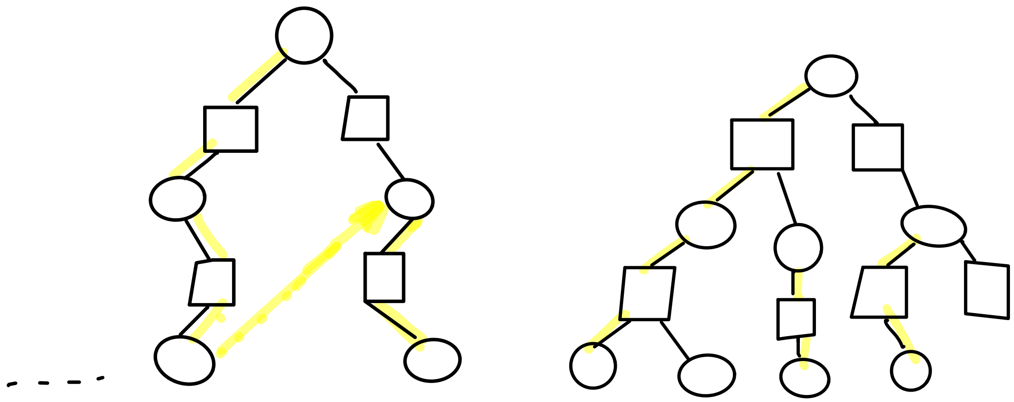
Desirable Algorithmic Properties

- *Scalability* (a la bitter lesson): the more data and compute are available, the better performance should be
- *Graceful degradation:* future performance should be really good if the agent is in similar situations to what it has seen, and is allowed to degrade as the situations are increasingly different
- *Self-reliance:* the agent should be able to learn and understand the world from its own experience

Exploration for "Small" Agents

- Every time step of experience matters: goal is cumulative, online return!
- The agent does NOT have enough time to visit all nodes!
- State coverage, visitation counts and similar measures are no longer useful
- *Exploration needs to improve the speed of learning on the agent's trajectory*
- Information-directed sampling is a promising algorithmic path, albeit difficult computationally at the moment

How to Make Exploration Easy

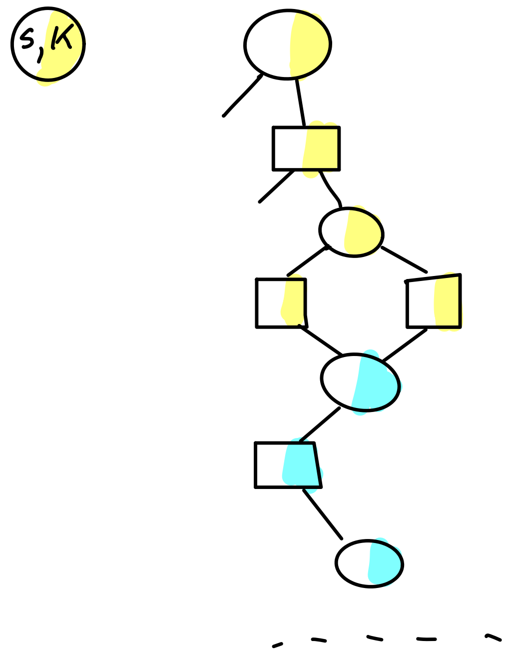


- Resets (left) allow the agent to teleport into a desired state
- Distributed agents (right) are almost as powerful (generate lifetimes from many different states)
- Neither seems conceptually well suited for never-ending learning (though may be useful to get off the ground)
- We should really re-think the fundamentals of exploration!

Credit Assignment and Generalization

- Agent needs to construct its state and decide on a time scale at which to make decisions
- Learning is driven by mismatch between predictions and observations
- The raw feedback signal is return, but the agent can *choose to learn about other signals*
- *Demultiplexing:* decompose a single signal (return) into a variety of signals

Relation to Existing Frameworks: Multi-task RL

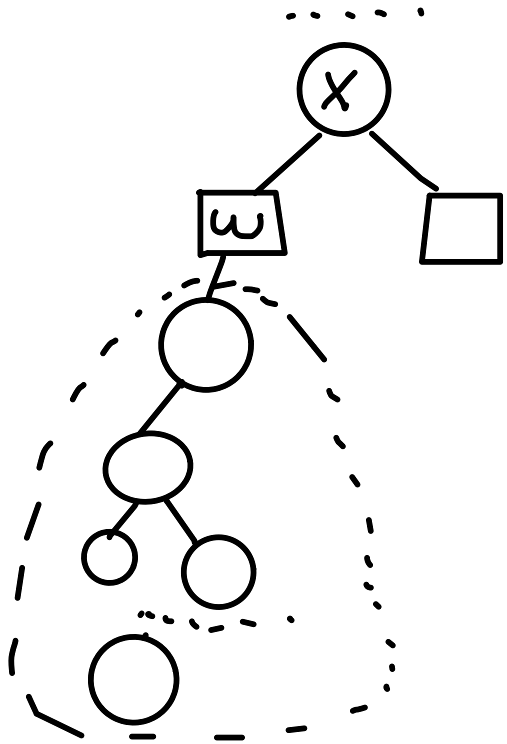


- Suppose the observation $x_t = \langle s_t, k_t \rangle$ where k_t is the index of a state at time t and s_t is the state inside the task
- The setup can be modelled as sequential decision making in this (much larger) problem
- Typically the task id is not available to the agent

Why is multi-task useful?

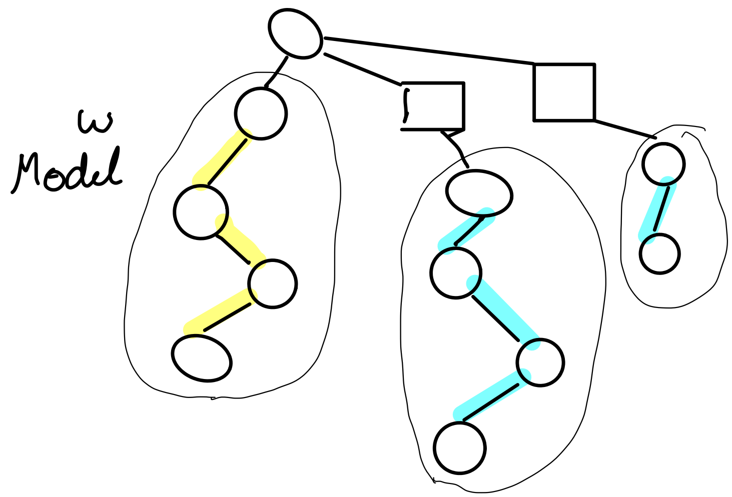
- Agent can propagate credit to many nodes in the tree! Not just temporal predecessors
- *Task structure exists only in the agent's head, in order to make credit assignment easier*
- Note that multi-task is the same as regular RL, just in a much larger and more structured problem

Hierarchical RL: Options



- A way of behaving (internal policy) and a termination condition
- *Impact on exploration!* DIAYN, action repeats, ...

Hierarchical RL: Temporally extended updates

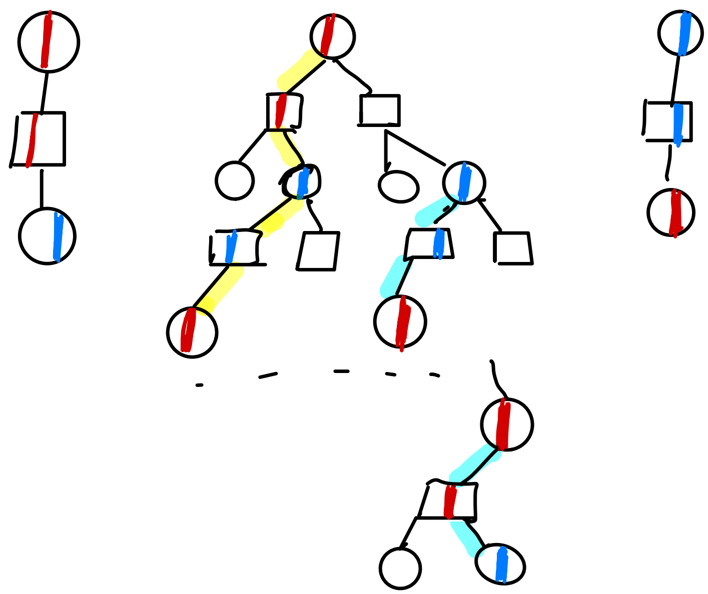


- Could be done through a model or through a value update
- *Impact on credit assignment!* More efficient credit propagation

Some observations

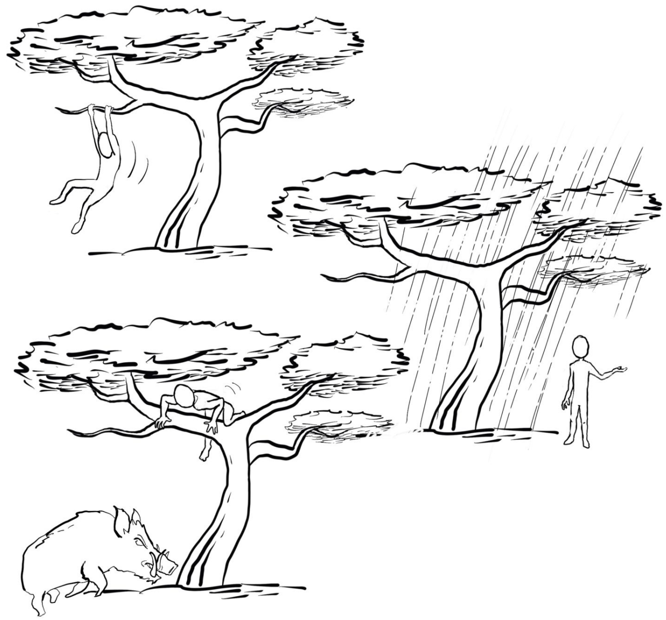
- The options paper describes options as a way of behaving, which has an associated model
- In that paper, models are built for the options that are executing
- In reality, *options that execute could (should?) be disconnected from extended models used for credit assignment!*
- Exploration options need to make the agent move consistently away from where it is
- Credit assignment should likely be done considering "smarter" options

Relation to Existing Frameworks: Partial Models / General Value Functions



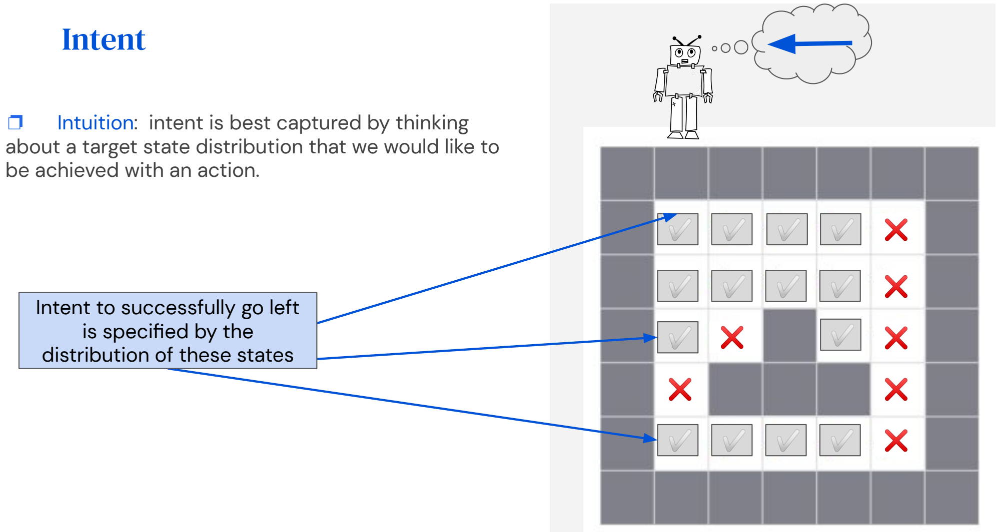
- Apply only in specific circumstances
- Predict only specific features / cumulants

Affordances



- Animals and humans understand intuitively what is "possible" in their current situation, termed *affordances* (cf Gibson, 1966)
- This is useful both for exploration (limit possible actions) and for planning (can anticipate action effects)

Formalizing Affordances in Reinforcement Learning



- Suppose the agent has some *intent* (eg change in a feature value)
- An *affordance relation* specifies for which agent states and options/actions the intent can be partially achieved
- Affordances can be used to restrict choices during exploration but also lead to partial models that are *approximately causal*
- Both learning and planning can be faster using affordances (cf Khetarpal et al, ICML'2020)

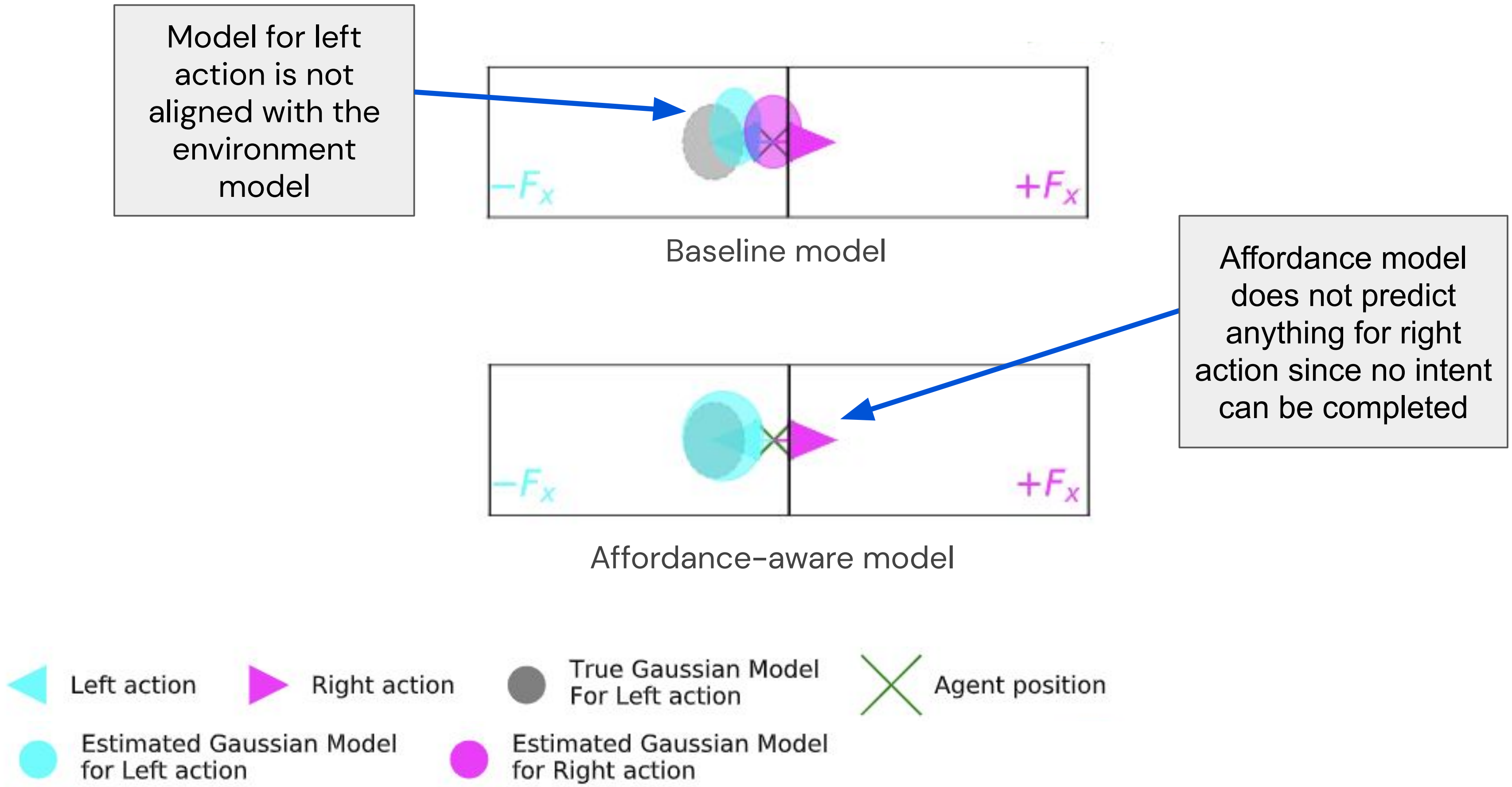
Affordance-aware partial model learning for actions

- Consider estimating a model for the transition dynamics, $P_\phi(s'|s,a)$ or $P_\phi(s'|s,\omega)$
- Usually we would estimate the model parameters ϕ through a maximum likelihood approach
- With affordances, we can estimate a *partial model* only for state-action or state-option pairs that are in the affordance:

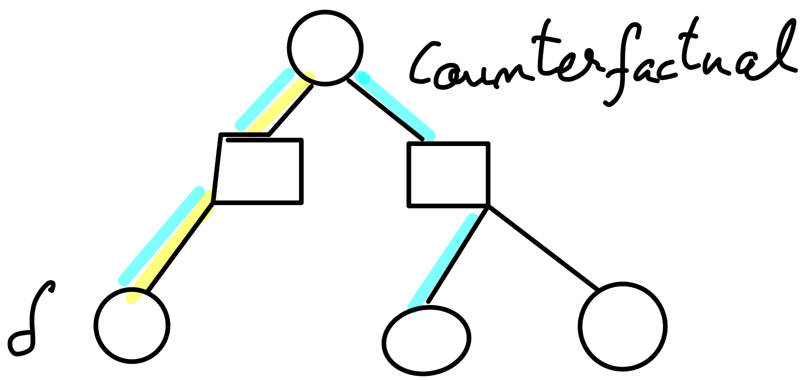
$$\mathcal{O}_{\text{aff}}(\phi) = \sum_{(s,a,s')\in\mathcal{D}} \boxed{1\left[\max_{\forall I\in\mathcal{I}} A_\theta(s,a,I) > k\right]} \log P_\phi(s'|s,a)$$

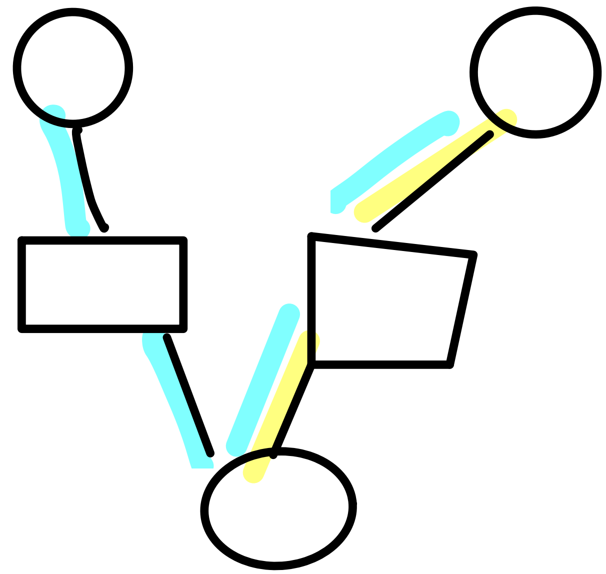
Mask based on if at least one intent is completable.

Empirical Illustration: Partial Model Learning



Alternative credit assignment patterns





- Mixture of remembering history / backward models and using a forward model to update
- See recent work on backward models (Chelu, Van Hasselt & Precup, NeurIPS'2020) and expected traces (van Hasselt et al, 2020)

Conclusion

- An agent that is much smaller than its environment will be pressured to find structure on its current trajectory: continually, online, not striving for optimality but for gradual improvement.
- The structure it builds drives two important computations: exploration decisions and credit assignment
- While agent implementations often link these two computations, they can and perhaps should be more decoupled
- Many of the ingredients needed already exist (information-directed sampling, GVFs, options, affordances, partial models)

Looking Ahead

- From a theoretical point of view, we need to formalize the problem further

  Moving away from usual stationarity/recurrence assumptions to fully transient agents

- From an empirical point of view, we should think of the appropriate environments

  Reconsider reward sparsity as a mark of interesting problems?