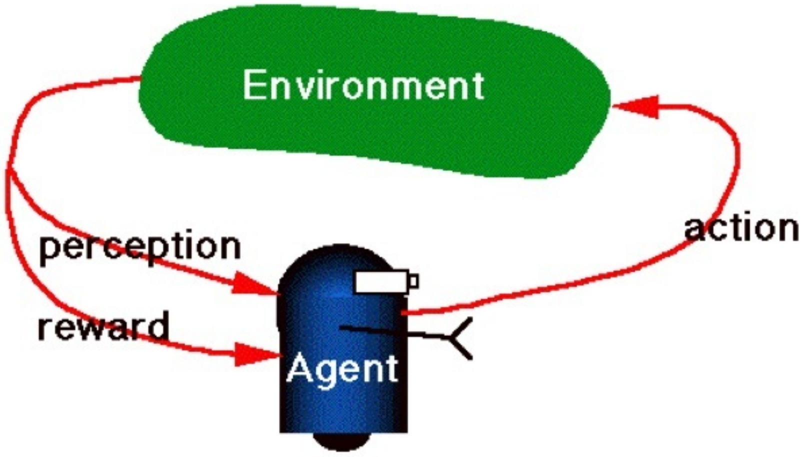
Thoughts on Reinforcement Learning for AI

Doina Precup

Reinforcement Learning

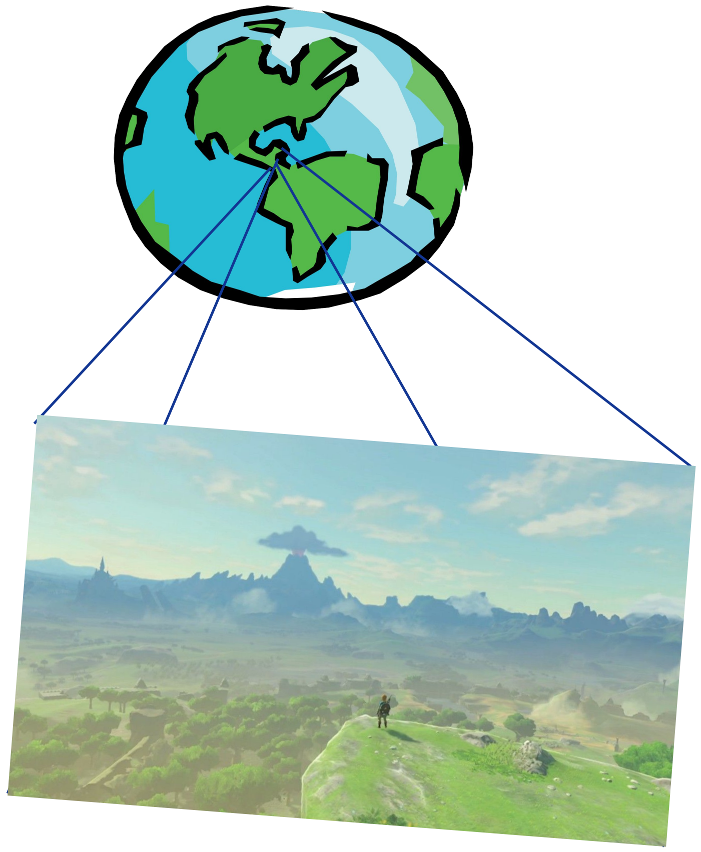


- Learning by interaction with a stochastic, unknown environment
- Maximizing long-term return allows formulating goals
- Agent needs to try actions and observe their outcomes, teacher/supervision is not necessary
- A very natural formulation of learning for intelligence (cf. Silver, Singh, Precup & Sutton, 2020)

Reinforcement Learning Can Solve Impressive Tasks

- Successes obtained with extensive simulation data and computation
- Task is clear and specific (win the game)
- More data and more compute lead to better agents (Rich Sutton's bitter lesson)

Today's Perspective



High-Level View of Agent

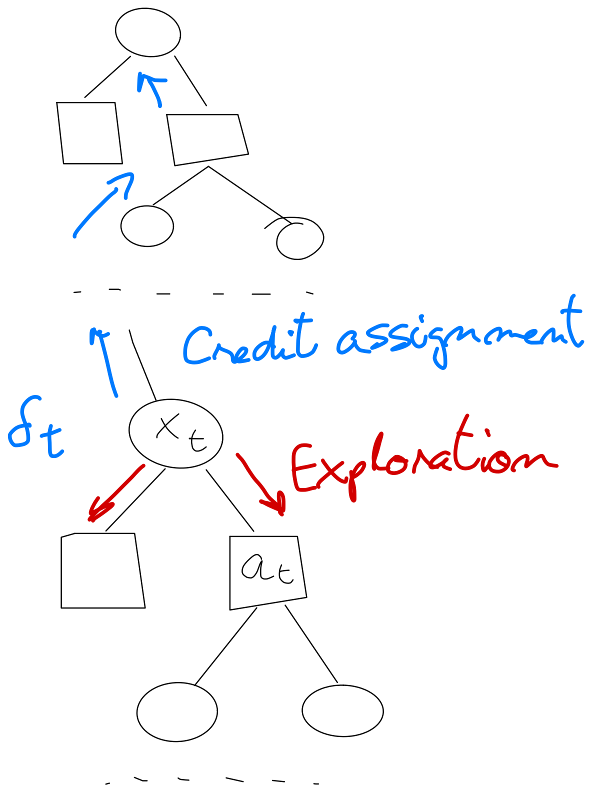
- Agent has *one stream of experience (observations, actions, rewards)* to support all learning processes
- Focus remains on maximizing return
- *Agent is "smaller" than the entire environment*
  - Only has time to travel on a specific trajectory
  - Cannot compute arbitrarily fast or remember all the 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

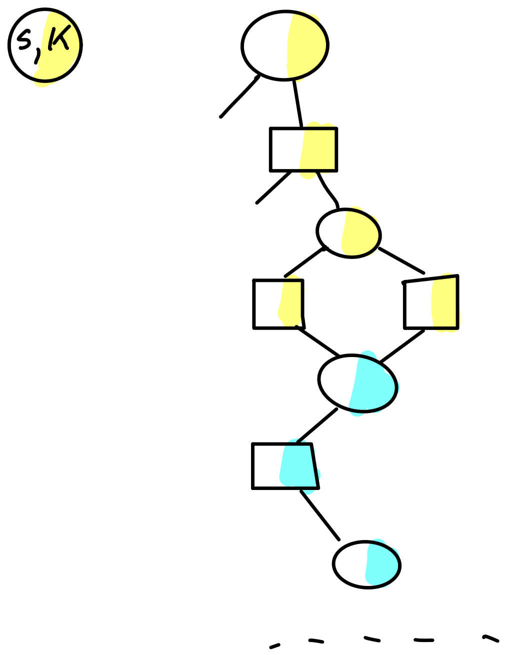
Exploration for "Small" Agents

- 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

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

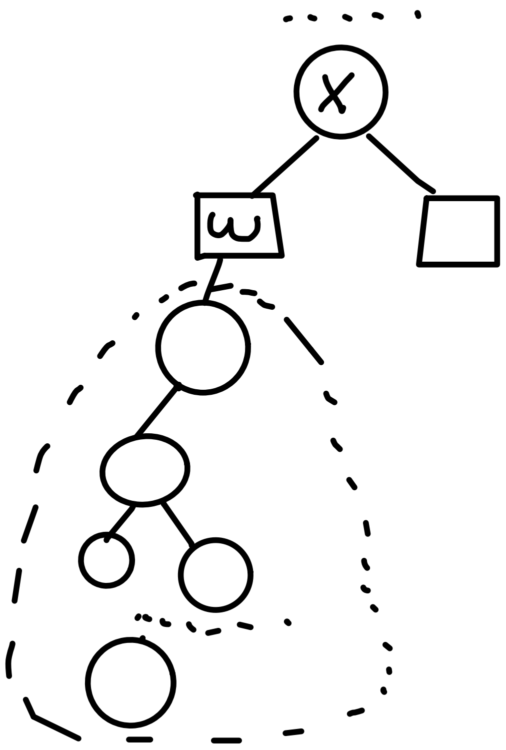


- Suppose the observation $x_t = \langle s_t, k_t \rangle$ where k_t is the index of a state at time t and s_t is the state inside the task
- The setup can be modelled as sequential decision making in this (much larger) problem
- Typically the task id is not available to the agent

Why is multi-task useful?

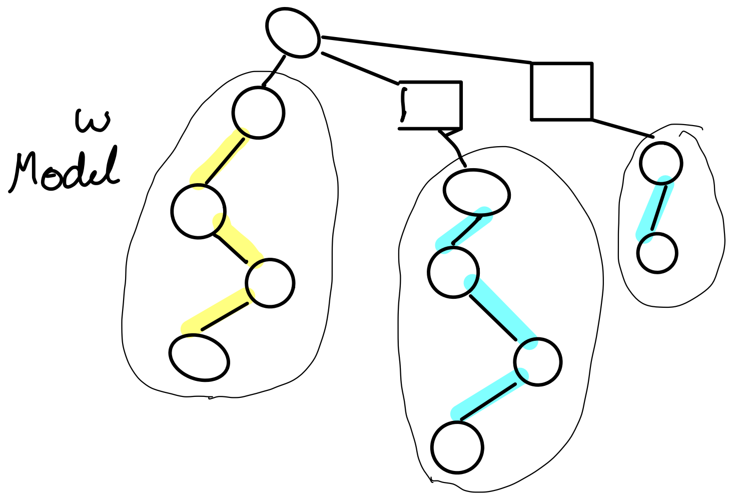
- Agent can propagate credit to many nodes in the tree! Not just temporal predecessors
- *Task structure exists only in the agent's head, in order to make credit assignment easier*
- Note that multi-task is the same as regular RL, just in a much larger and more structured problem

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

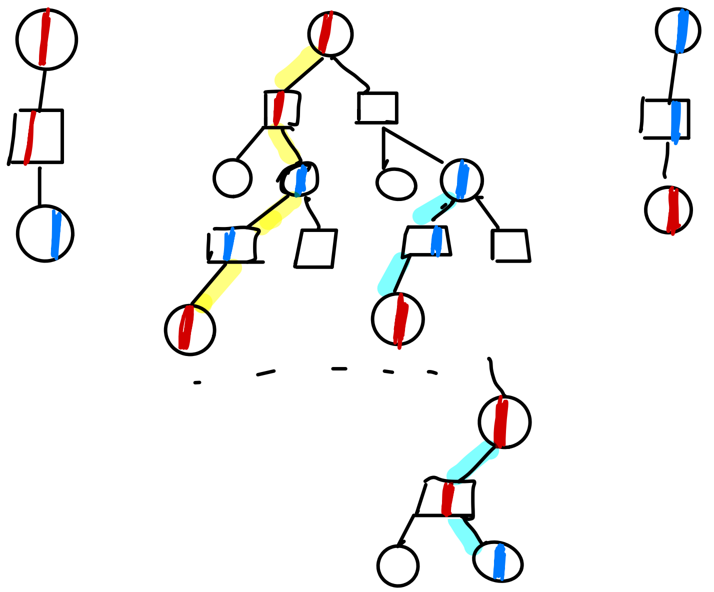
General Value Functions (GVFs)

- Given a cumulant function c, continuation function γ and policy π, the General Value Function $v_{\pi,\gamma,c}$ is defined as:

$$v_{\pi,c,\gamma}(s) = \mathbf{E}\left[\sum_{k=t}^{\infty} c(S_k, A_k, S_{k+1}) \prod_{i=t+1}^{k} \gamma(S_i) | S_t = s, A_{t:\infty} \sim \pi\right]$$

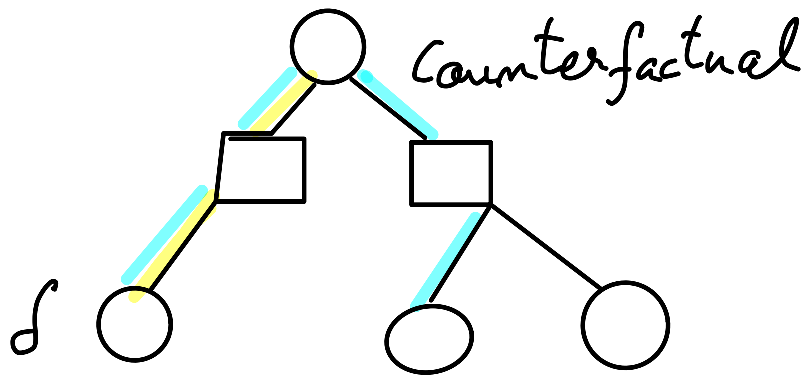
- Cf. Horde architecture (Sutton et al, 2011); Adam White's thesis; inspiration from Pandemonium architecture
- Special case: policy is optimal wrt c, γ, $v^*_{c,\gamma}$ - Universal Value Function approximation (UVFA) (Schaul et al, 2015)
- Agent can use a multitude of cumulants and time scales!

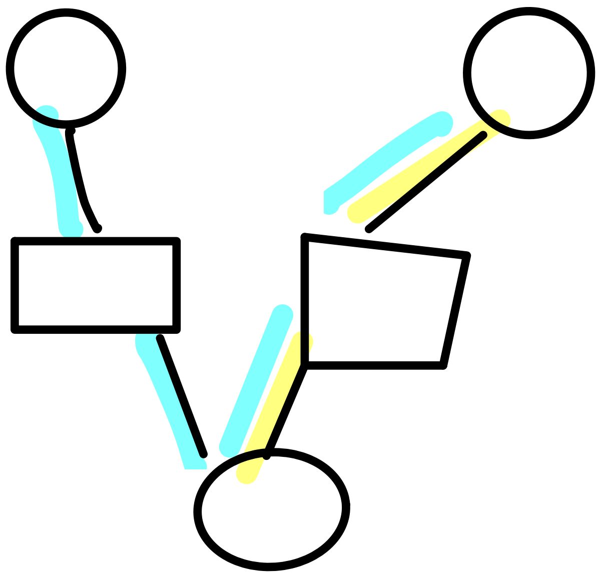
Partial Models / GVFs



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

Alternative credit assignment patterns



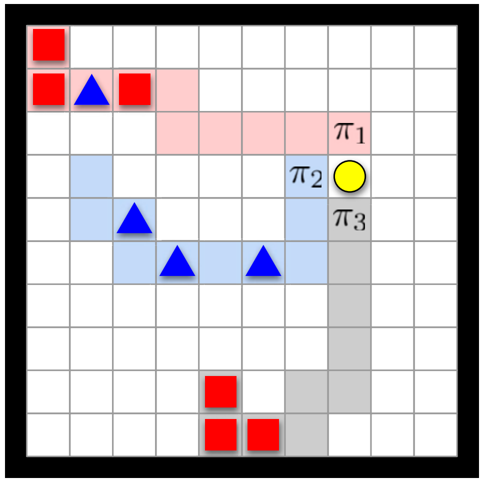


- Mixture of remembering history / backward models and using a forward model to update
- See recent work on backward models (Chelu, Van Hasselt & Precup, NeurIPS'2020) and expected traces (van Hasselt et al, 2020)

The Role of Planning

- Planning is often used with two (rather different) meanings:
  - Allowing an agent to change its mind on the current decision, by using a model (eg through lookahead search)
    Eg model-predictive control
  - Changing the value function estimates by using a model
    Eg Dyna
- Planning should always be useful if the agent's representation is imperfect / limited!
- Planning can allow us to recombine existing knowledge zero-shot

Illustration



- Planning over pre-trained GVFs achieves *75x improvement in sample size* compared to Q-learning

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?