Continual (Never-Ending) Reinforcement Learning

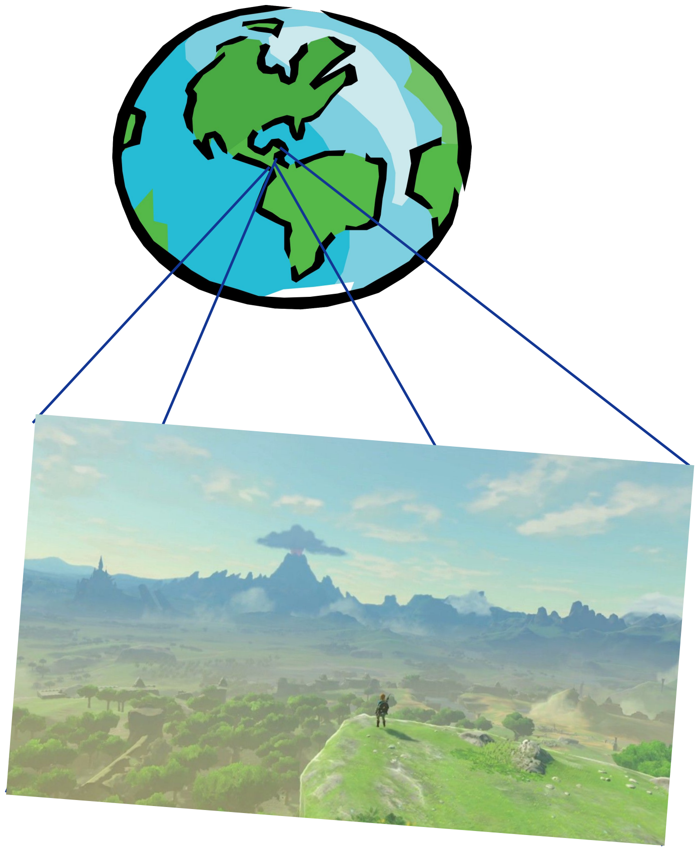
Doina Precup

A Mystery: Learning Efficiently from Interaction

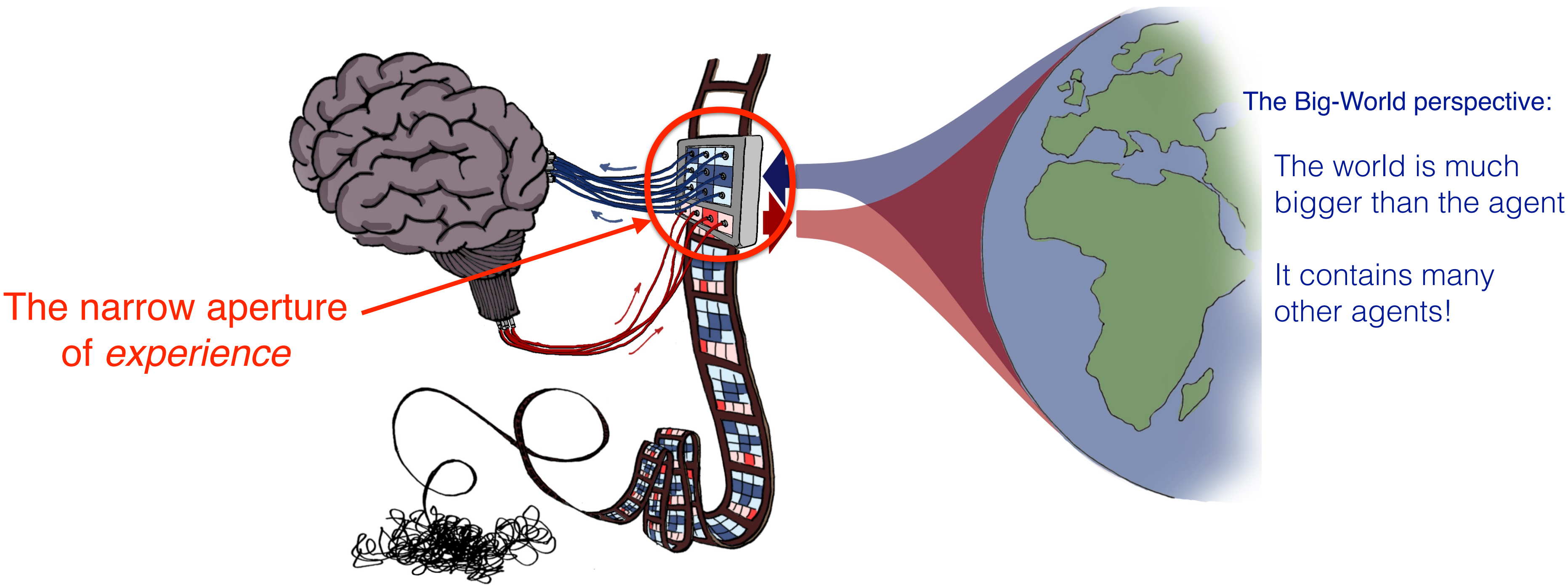
- People in their lifetime experience: 100 years x 365 days x 24 hours x 3600 seconds x 640 muscle activations/second = 20 trillion motor actions
- How can we learn from only this amount of data? In a very big world that is partially observable, complex, has other agents in it?
- And while consuming around 2000 calories/day?

Very good representation and very efficient learning algorithms

Today's Perspective



Aperture principle



High-Level View of Agent

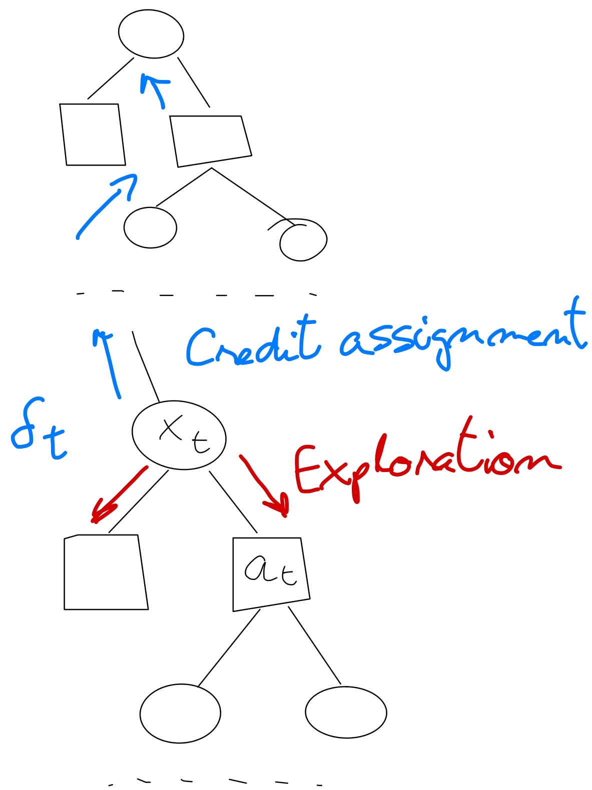
- Agent has *one stream of experience (observations, actions, rewards)* to support all learning processes
- *Agent is "smaller" than the entire environment*
  - Only has time to travel on a specific trajectory
  - Cannot compute arbitrarily fast or remember all the relevant experience in a replay buffer
- *Asynchronous, online 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 general intelligence, 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 view on algorithmic solutions and theoretical framing

Recall: Cartoon of 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
- Instead, consider 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 (allowed by Markovian structure)
- 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!

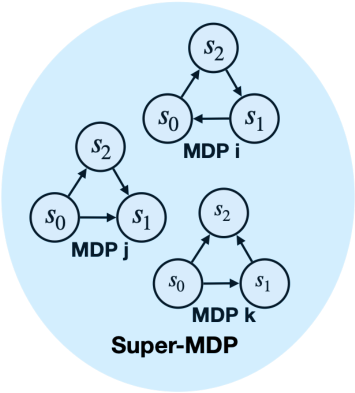
Desirable Algorithmic Properties

- *Stability and plasticity*: useful knowledge should be retained but the agent should remain able to learn
- *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
- More debatable: *Self-reliance:* the agent should be able to learn and understand the world from its own experience

Sequential Decision Making beyond MDPs

- At decision point t, the agent receives an observation $x_t \in \mathcal{X}$ and chooses an action $a_t \in \mathcal{A}$
- Let t' be the next decision point (as a special case, $t' = t + 1$)
- The agent also receives a reward for this period, with value $r_{t,t'}$, which depends on the agent's action
- There is a designated terminal observation, $\perp$, which ends the agent's trajectory
- Let $t_\perp$ designate the time at which this observation is received
- Assume $t_\perp$ is finite on all trajectories
- *The goal of the agent is to maximize the cumulative return received over its life time, expressed as a sum of rewards*: $\sum r_{t,t'}$ where the first $t = 0$ and the last $t' = t_\perp$
- *A learning algorithm will be evaluated in expectation over instantiations of environment-agent pairs*

Computational and Information Limitations are Important



- If the agent sees the identity of the MDP and the state, it's usual RL
- If the agent sees only the state, we need continual adaptation!
- Cf. Rich Sutton's aperture principle

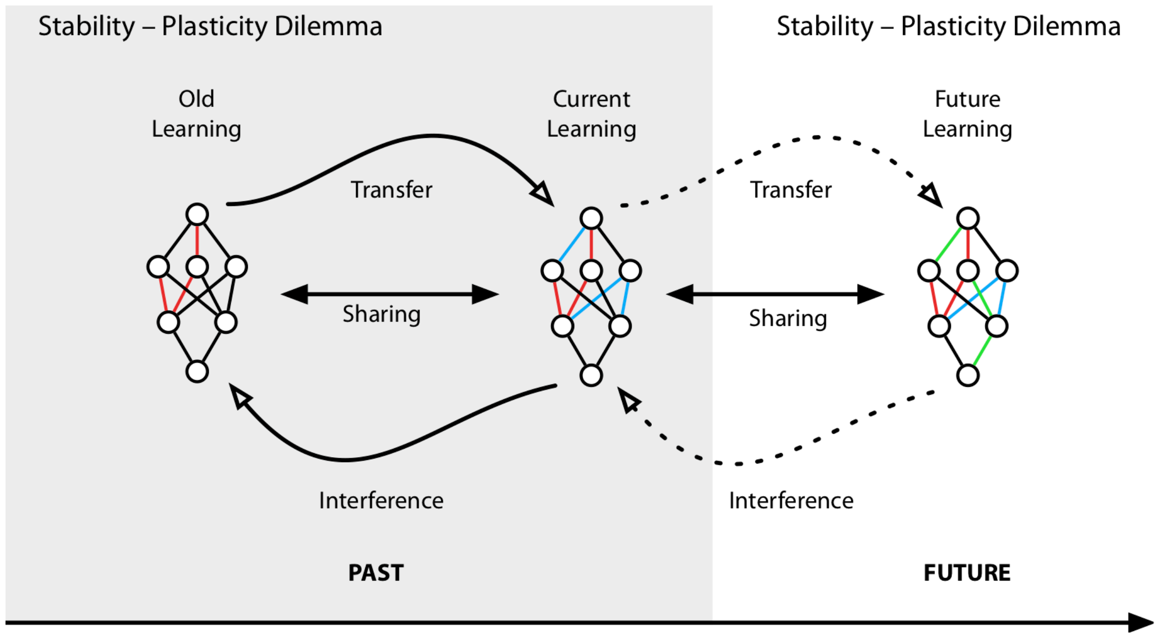
Some interesting special cases

- MDPs and POMDPs: assumptions on how $x_{t'}$ and $r_{t,t'}$ are generated by the environment as a function of x_t and a_t
- Online regression: the label is the action, the reward is the loss function
- Predictive state representations (Littman et al, 2002, Singh et al, 2004) and related models (eg Jaeger, 2002): low-rank linear structure on x, a trajectories

What is useful structure?

- The agent needs to be able to do induction: estimate potential future return from its past history
- We want to continue leveraging the compositionality of returns: $G_t = r_{t,t'} + G_{t'}$

Stability-plasticity dilemma

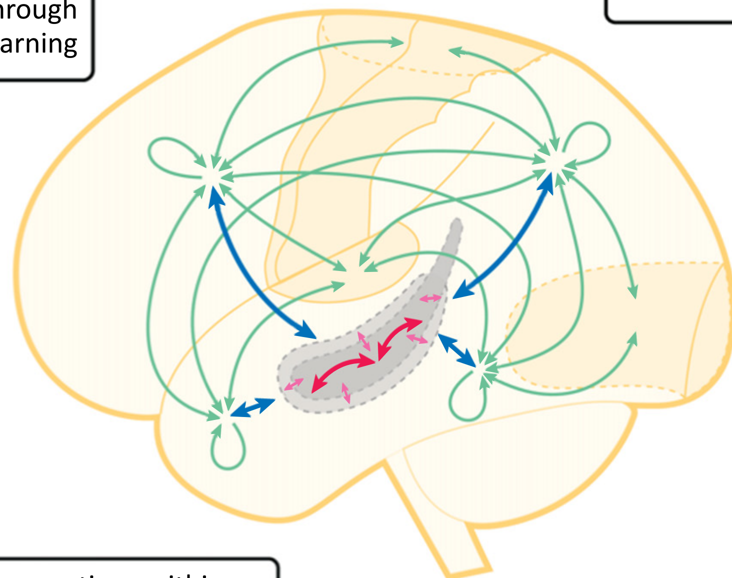


- DeepRL agents lose plasticity (cf Nikishin et al, 2022)
- Even deep supervised learning architectures lose plasticity (cf. Dohare et al, 2022)
- Solutions proposed above mainly focus on resetting weights - retains plasticity but loses stability

Complementary Learning Systems

Connections within and among neocortical areas (green) support gradual acquisition of structured knowledge through interleaved learning

Bidirectional connections (blue) link neocortical representations to the hippocampus/MTL for storage, retrieval, and replay



Rapid learning in connections within hippocampus (red) supports initial learning of arbitrary new information

Cf. Kumaran, Hassabis and McLelland, 2016

Simple RL Implementation

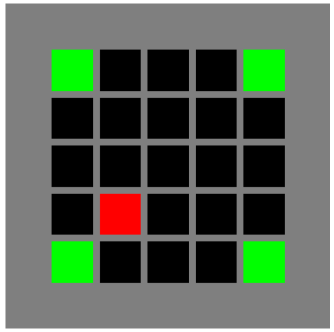
- Value function has two components:

$$V^{PT}(s) = V_{\theta}^{P}(s) + V_{\mathbf{w}}^{T}(s)$$

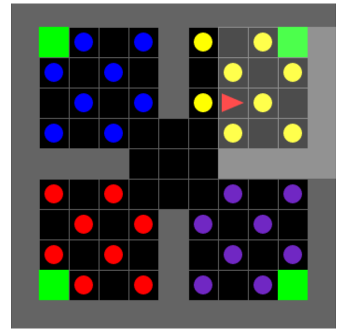
- *Permanent memory*: V^P should provide good estimates for any circumstances
- *Transient memory*: V^T should quickly compute corrections to V^P to adapt the the current distribution
- Both updated in parallel using TD-style updates
- This paper: both functions using the same features

Cf. Anand and Precup, NeurIPS'2023

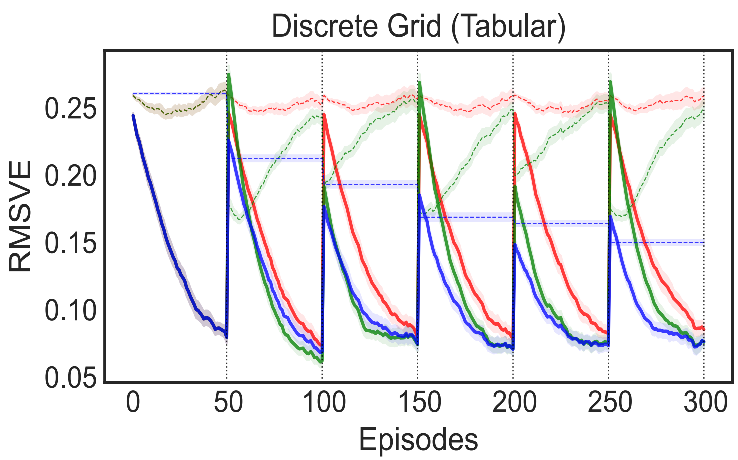
Prediction results

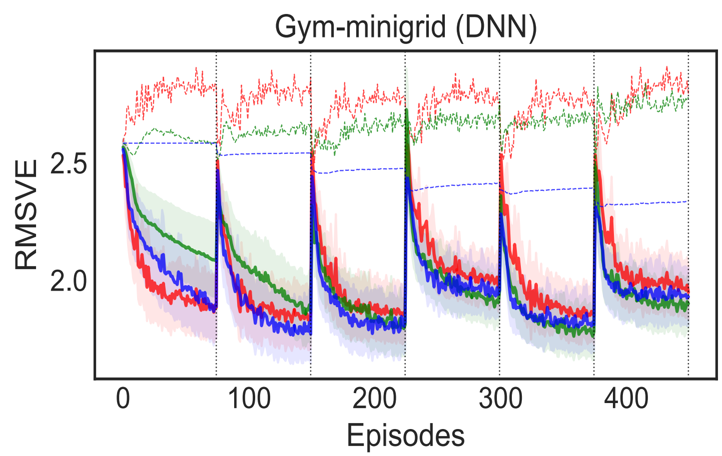


(a) Discrete grid.



(b) Gym-minigrid.

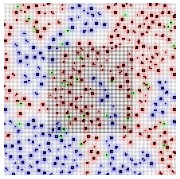




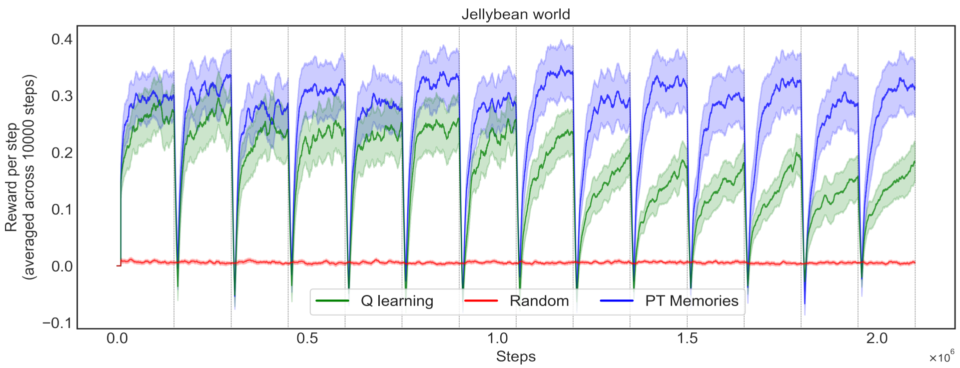
The agent retains knowledge while preserving plasticity!

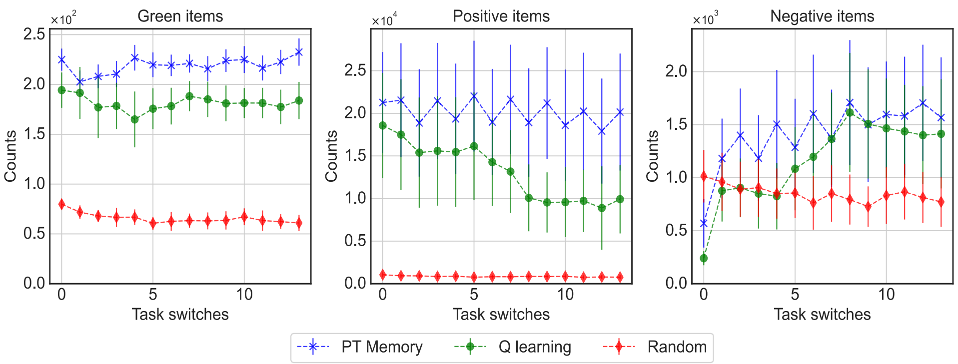
Control results

- Using JelllyBean World (Mitchell et al, 2020)



- Both spatial and reward non-stationarity (green objects give small rewards, blue and red objects flip between great and bad)

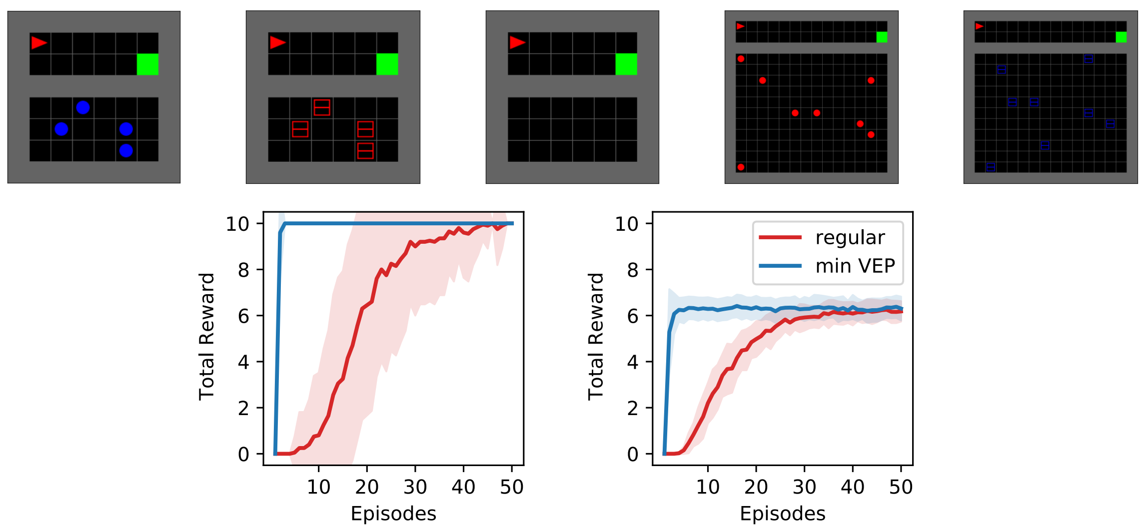




The agent retains knowledge which helps it perform well over time!

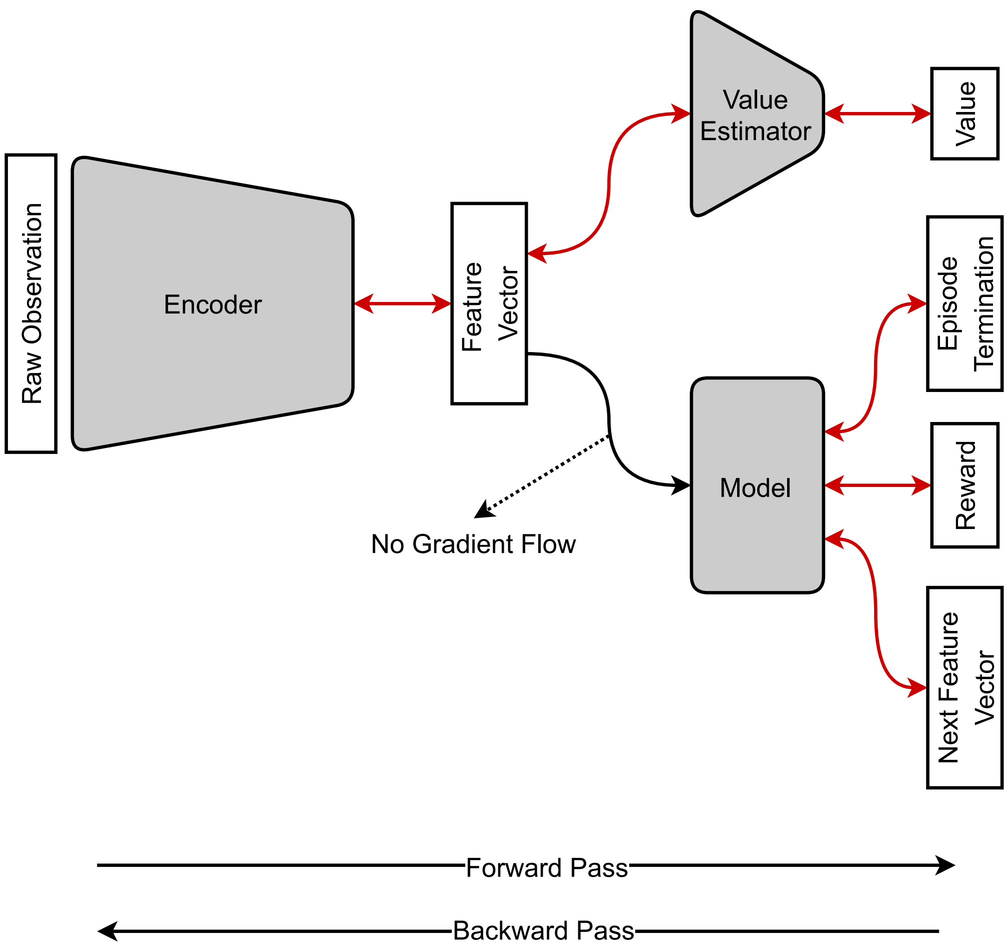
Partial Value-Equivalent Models

- Model only predicts a subset of features (not the entire observation) (cf. Talvitie & Singh, 2008)
- Goal is to obtain correct value estimates, not to maximize likelihood
- Example: minigrid

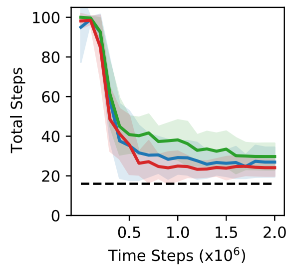


Partial models drastically improve solution speed! (cf Alver & Precup. 2023)

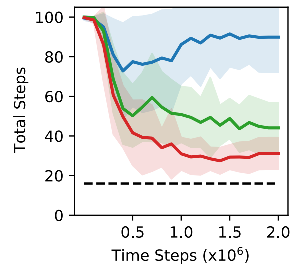
Learning Partial Value-Equivalent Models



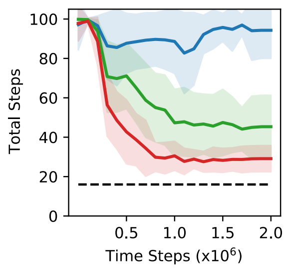
Learned partial models improve generalization



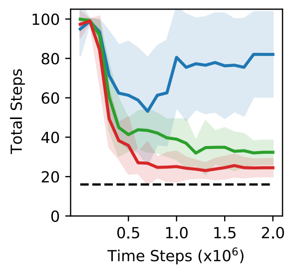
(c) 16x16 BlueBalls-R



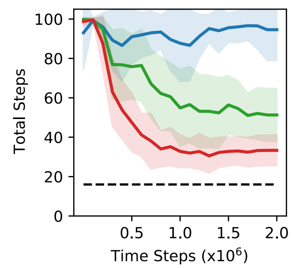
(d) 16x16 NoObstacles-R



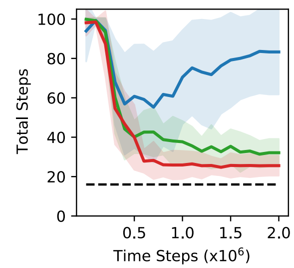
(i) 16x16 RedBalls-R



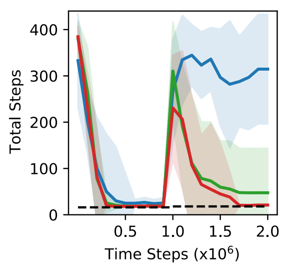
(j) 16x16 GreyBalls-R



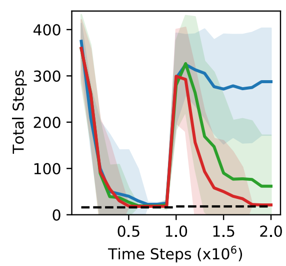
(k) 16x16 RedBoxes-R



(l) 16x16 GreyBoxes-R



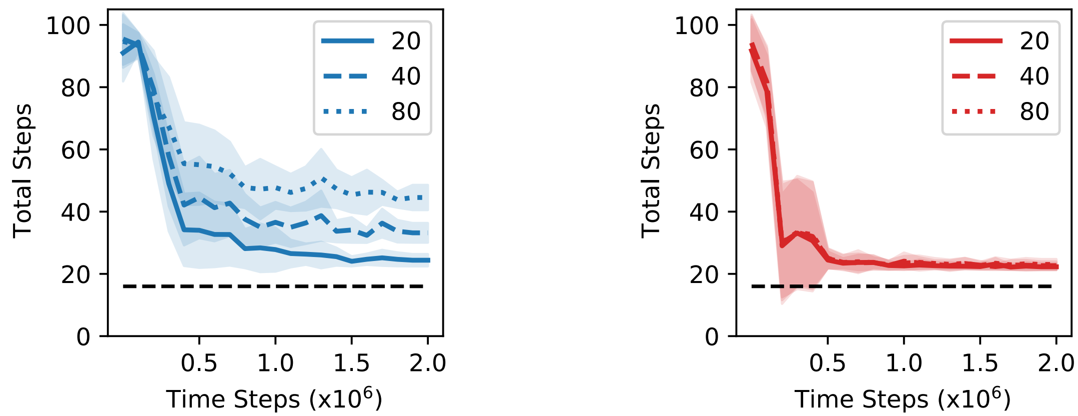
(e) 16x16 RedBalls-K



(f) 16x16 GreyBalls-K

Blue: Regular, Green: Value-Equivalent, Red: Value equivalent + models

Partial models allow deeper planning

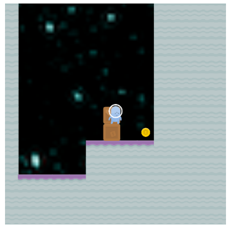


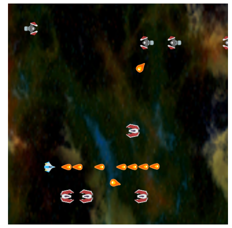
(g) The A_{REG} agent

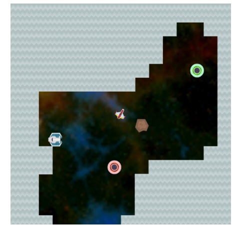
(h) The A_{VES+ME} agent

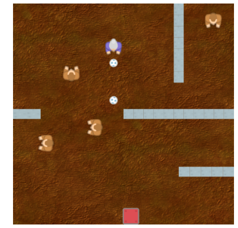
- Regular models (left) lead to worse performance when doing more planning steps, due to error propagation
- Partial models have better error propagation properties (see Alver & Precup. 2023, for details on the theory)

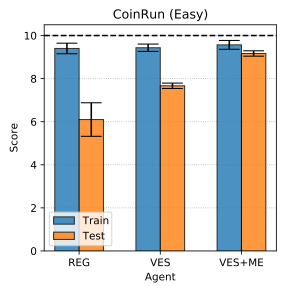
Scaling up: ProcGen


(a) CoinRun

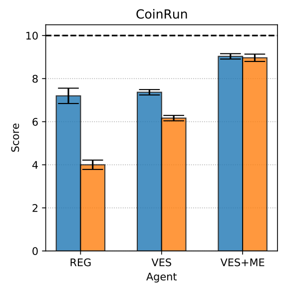

(b) StarPilot


(c) CaveFlyer

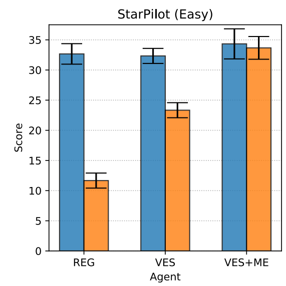

(d) DodgeBall



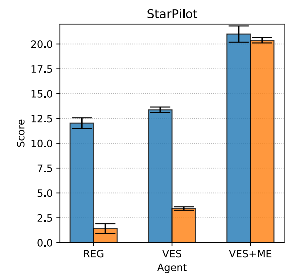
(a) CoinRun (Easy)



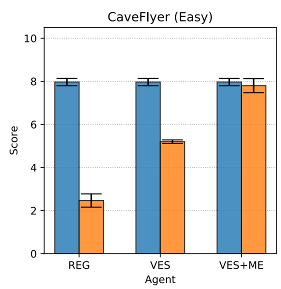
(b) CoinRun



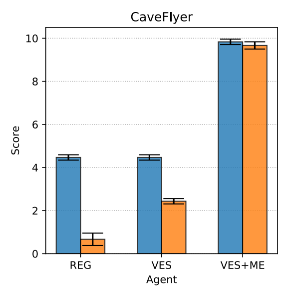
(c) StarPilot (Easy)



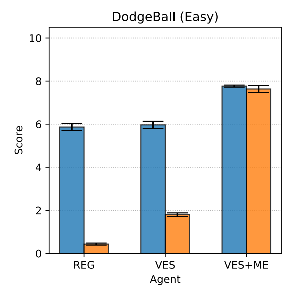
(d) StarPilot



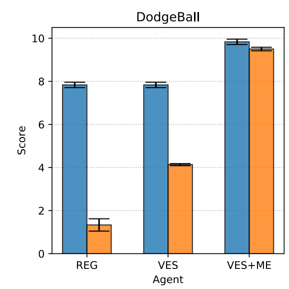
(e) CaveFlyer (Easy)



(f) CaveFlyer



(g) DodgeBall (Easy)



(h) DodgeBall

Partial models improve generalization!

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)

Some challenges

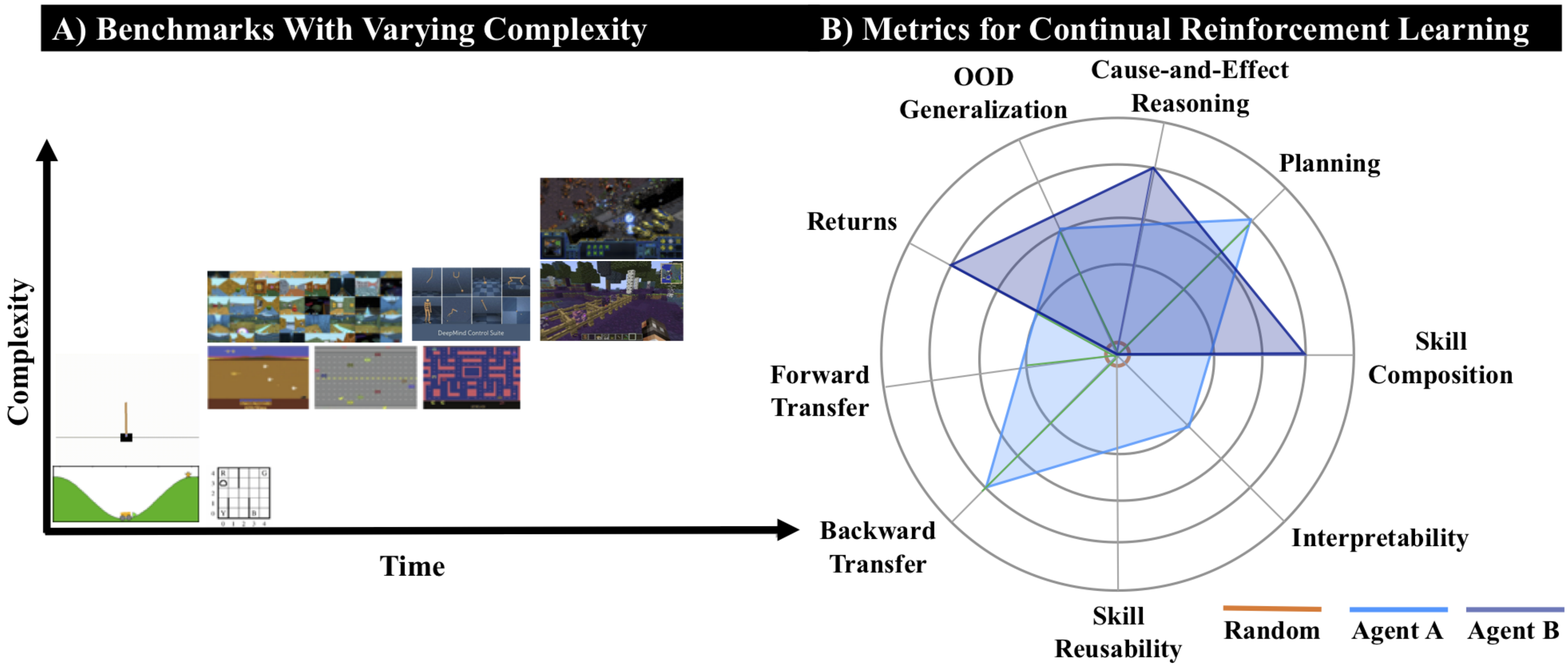
- From a theoretical point of view, we need to formalize the problem further

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

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

  Reconsider reward sparsity as a mark of interesting problems?

Evaluation for continual RL



Cf. Khetarpal, Riemer, Rish and Precup, 2022