Reinforcement Learning

Lecture 10: Extended methods

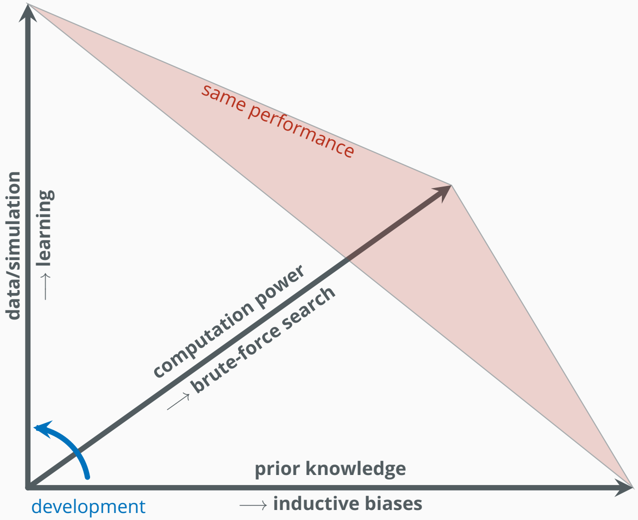
Robert Lieck

Durham University

Lecture overview

1. **The future of RL**
   - The trilemma in RL/ML/AI
   - The bitter lesson
   - Reward is (not) enough?
2. **Long-term dependencies**
   - DQN characteristics
   - Recurrent and distributed RL
   - R2D2 performance
   - Representation learning
3. **Intrinsic reward**
   - Exploration vs exploitation
4. **Prior knowledge**
   - AlphaStar and StarCraft
   - meta-knowledge and training heuristics

The trilemma in RL/ML/AI



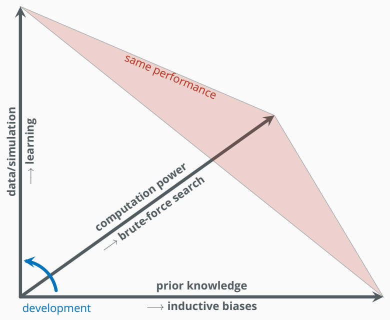
The trilemma in RL/ML/AI

Extreme strategies

- if you understand your problem
  → **hard-code** the solution
- if you have enough computation power
  → **brute-force** search the solution space
- if you have enough data (and computation power)
  → learn a general function approximator to **interpolate** between the data points

In reality...

...we need to combine these strategies. We cannot (efficiently) learn anything without making assumptions (prior knowledge) [1].



Long-term strategy/development

1. use prior knowledge to simplify problem if needed
2. replace *hard* and *specific* assumptions by *soft* and *general* heuristics
3. progressively move towards general-purpose solutions

The bitter lesson

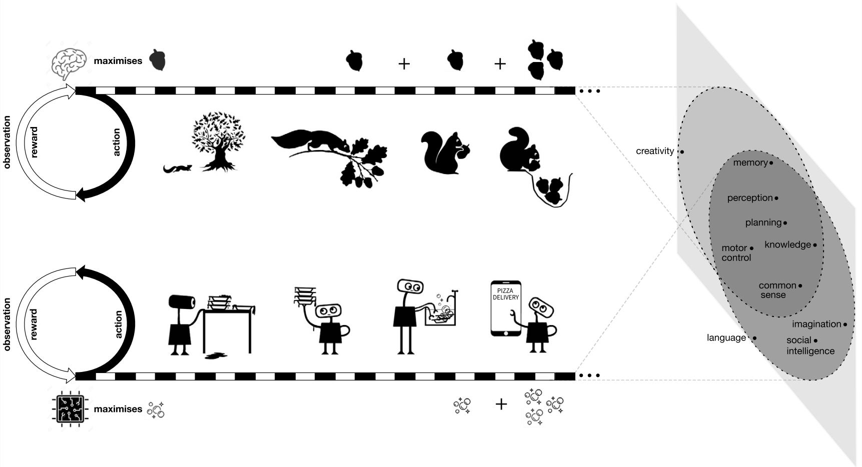
Rich Sutton, 2019

Rich Sutton's bitter lesson is that, despite it being tempting to incorporate domain knowledge, in the long run, general purpose agents win. **Link to article**

- AI researchers have often tried to build knowledge into their agents
- this always helps in the short term and is personally satisfying to the researcher
- but in the long run it plateaus and even inhibits further progress
- breakthrough progress eventually arrives by an opposing approach based on scaling computation by search and learning

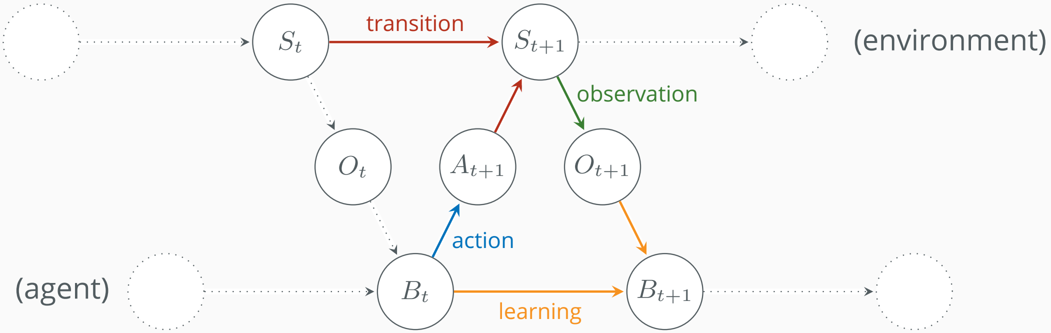
Reward is (not) enough?

Reward is enough [2] (Silver, Singh, Precup & Sutton). Others argue for intrinsic rewards in practice.



- For **defining** a problem?
- For **solving** a problem?

Long-term dependencies partially observable MDPs



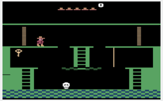
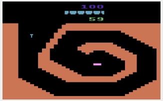
- S is not known $\rightarrow$ process is not Markov (w.r.t. the observations O)
- predictions depend on the entire history of actions and observations

Long-term dependencies DQN characteristics

Characteristics: DQN

DQNs optimise a function (neural network) to predict the Q-value (the expected reward) for a given state and aciton.

DQN doesn't work very well for long-term credit assignments:


Recap: function approximation

There are too many states/actions to fit into memory, so we estimate the value function:

$$\hat{v}(S, \mathbf{w}) \approx v_\pi(S),$$

or for control we'd do:

$$\hat{q}(S, A, \mathbf{w}) \approx q_\pi(S, A),$$

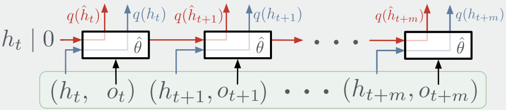
This usually requires several extra tricks:

- Double DQN
- Prioritised experience replay buffer
- Noisy linear layers

Definition: R2D2

Recurrent Replay Distributed DQN (R2D2) [3] uses RNNs, training on a sequence of $m = 80$ observations o_t and hidden states h_t:

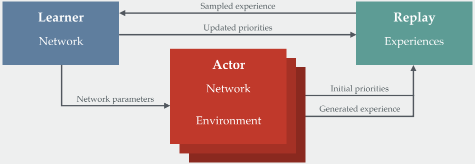
Computation of $\triangle Q$



Therefore it can backpropagate through the history, updating where earlier actions led to long-term future reward.

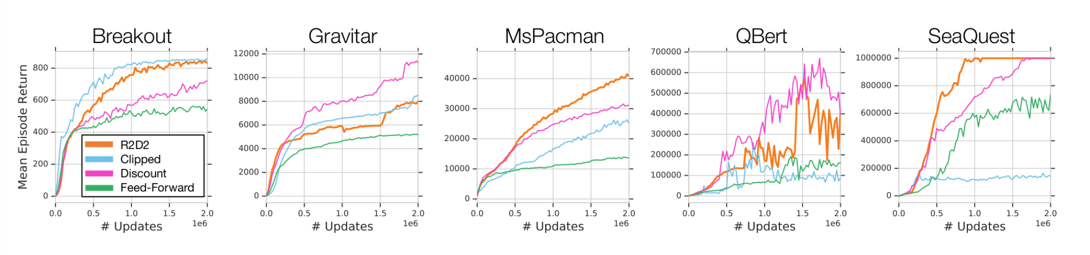
Definition: distributed RL

In distributed RL [4], a central learner (with some parameters θ) receives experience from multiple parallel workers $w_1, w_2, \ldots, w_n$ which run episodes independently:



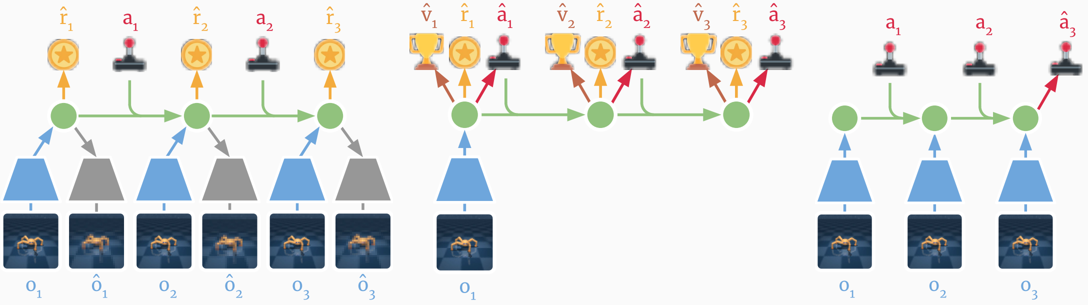
Recurrent and distributed RL R2D2 performance

These graphs shows R2D2 performance for different ablations:



Watch R2D2 play Gravitar **Watch R2D2 play other Atari**

Dreamer [5] and DreamerV2 [6] use a recurrent neural network to ‘imagine’ and plan ahead, all in the latent (feature representation) space:



(a) Learn dynamics from experience (b) Learn behavior in imagination (c) Act in the environment

Initialize dataset $\mathcal{D}$ with S random seed episodes. Initialize neural network parameters θ, ϕ, ψ randomly.
while *not converged* **do**
  for *update step* $c = 1..C$ **do**
    `// Dynamics learning`
    Draw B data sequences $\{(a_t, o_t, r_t)\}_{t=k}^{k+L} \sim \mathcal{D}$.
    Compute model states $s_t \sim p_\theta(s_t \mid s_{t-1}, a_{t-1}, o_t)$.
    Update θ using representation learning.
    `// Behavior learning`
    Imagine trajectories $\{(s_\tau, a_\tau)\}_{\tau=t}^{t+H}$ from each s_t.
    Predict rewards $\mathrm{E}\big(q_\theta(r_\tau \mid s_\tau)\big)$ and values $v_\psi(s_\tau)$.
    Compute value estimates $\mathrm{V}_\lambda(s_\tau)$ via Equation 6.
    Update $\phi \leftarrow \phi + \alpha \nabla_\phi \sum_{\tau=t}^{t+H} \mathrm{V}_\lambda(s_\tau)$.
    Update $\psi \leftarrow \psi - \alpha \nabla_\psi \sum_{\tau=t}^{t+H} \frac{1}{2} \left\| v_\psi(s_\tau) - \mathrm{V}_\lambda(s_\tau) \right\|^2$.
  `// Environment interaction`
  $o_1 \leftarrow$ `env.reset()`
  for *time step* $t = 1..T$ **do**
    Compute $s_t \sim p_\theta(s_t \mid s_{t-1}, a_{t-1}, o_t)$ from history.
    Compute $a_t \sim q_\phi(a_t \mid s_t)$ with the action model.
    Add exploration noise to action.
    $r_t, o_{t+1} \leftarrow$ `env.step(`a_t`)`.
  Add experience to dataset $\mathcal{D} \leftarrow \mathcal{D} \cup \{(o_t, a_t, r_t)_{t=1}^{T}\}$.

Model components

Representation	$p_\theta(s_t \mid s_{t\text{-}1}, a_{t\text{-}1}, o_t)$
Transition	$q_\theta(s_t \mid s_{t\text{-}1}, a_{t\text{-}1})$
Reward	$q_\theta(r_t \mid s_t)$
Action	$q_\phi(a_t \mid s_t)$
Value	$v_\psi(s_t)$

Hyper parameters

Seed episodes	S
Collect interval	C
Batch size	B
Sequence length	L
Imagination horizon	H
Learning rate	α

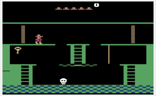
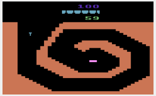
Exploration vs exploitation

We can take random actions for exploration, but there are other strategies

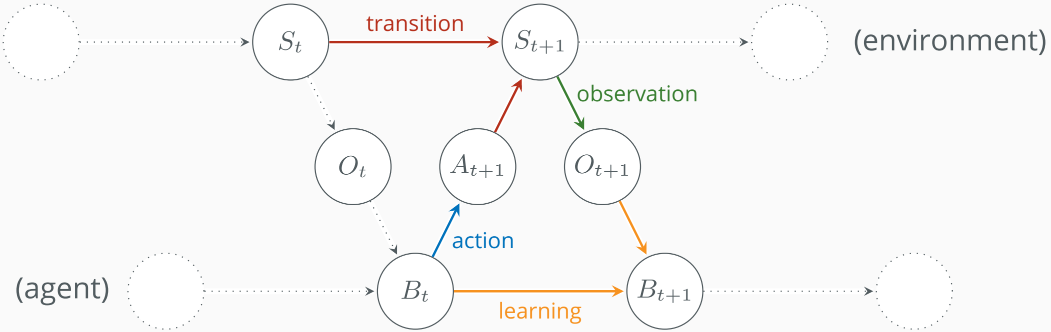
- random exploration (as before)
  - ϵ-greedy
  - softmax
- optimisim in the face of uncertainty
  - estimate uncertainty of the value
  - prefer exploring states/actions with higher uncertainty
- information/belief state space
  - the agent information is part of the state description
  - quantifies state information value

Exploration in Gravitar and AoE

Randomly choosing isn't always good:

Belief space partially observable MDPs

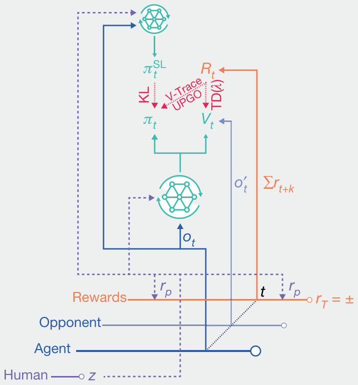


- S is not know $\rightarrow$ plan in belief space B instead
- uncertainty can be quantified and taken into account
- agent can actively plan for gathering useful information (*active learning*)
- hard to solve exactly $\rightarrow$ use approximations and Monte-Carlo methods

Prior knowledge AlphaStar and StarCraft

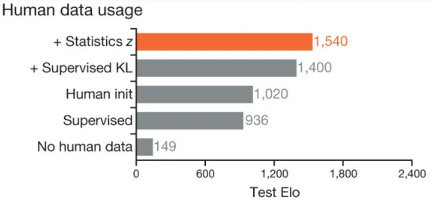
Architecture

AlphaStar [7] uses many components, supervised learning, and league-play.



Prior knowledge meta-knowledge and training heuristics

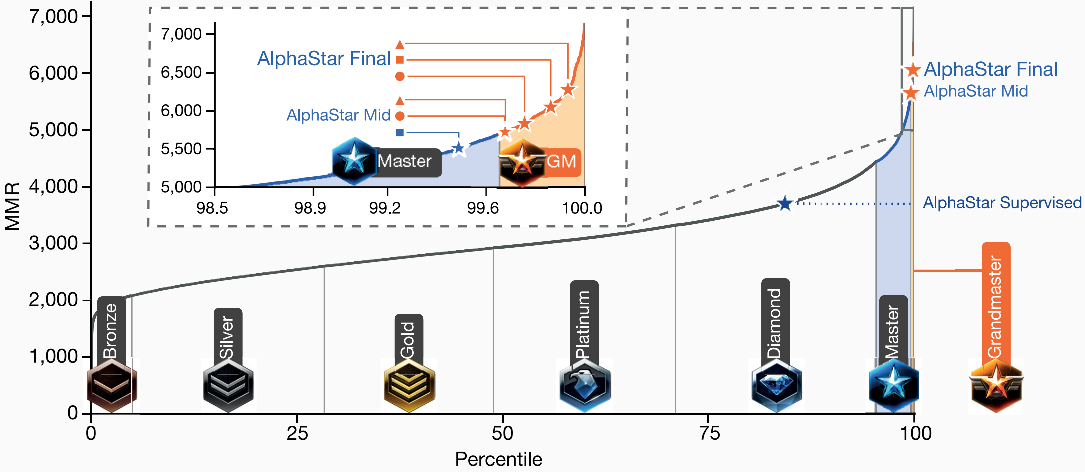
"We found our use of human data to be critical in achieving good performance with reinforcement learning" [7]



- start with training agent from human data (supervised learning)
- intrinsic rewards for following human statistics z
- self-play and league-play

- policy gradient similar to actor-critic
- temporal difference learning (TD(λ))
- clipped importance sampling (V-trace)
- *new* self-imitation algorithm (UPGO)
- use opponent's observations during training
- league training
  1. **main agents**: prioritized fictitious self-play (PFSP) against mixture of agents
  2. **main exploiters**: play only against main agents to find weaknesses
  3. **league exploiters**: also PFSP, but are not targeted by main exploiters

Prior knowledge AlphaStar results



Take away points

Summary

In summary:

- learn the foundations and concepts of the field, so you can speak the lingo...
- ...but you may want to approach overly complex papers more like an engineer
    - run the code and dismantle it back down to the concepts that make it work
- sample efficiency is an issue, which can be traded for with model-based imagination
- general purpose agents are the future

References I

[1] D. H. Wolpert and W. G. Macready. “No Free Lunch Theorems for Optimization”. In: IEEE Transactions on Evolutionary Computation (1997).

[2] David Silver et al. “Reward is enough”. In: Artificial Intelligence (2021), p. 103535.

[3] Steven Kapturowski et al. “Recurrent experience replay in distributed reinforcement learning”. In: International Conference on Learning Representations. 2018.

[4] Dan Horgan et al. “Distributed Prioritized Experience Replay”. In: International Conference on Learning Representations. 2018.

[5] Danijar Hafner et al. “Dream to Control: Learning Behaviors by Latent Imagination”. In: International Conference on Learning Representations. 2020.

[6] Danijar Hafner et al. “Mastering Atari with Discrete World Models”. In: International Conference on Learning Representations. 2021.

[7] Oriol Vinyals et al. “Grandmaster level in StarCraft II using multi-agent reinforcement learning”. In: Nature 575.7782 (2019), pp. 350–354.