

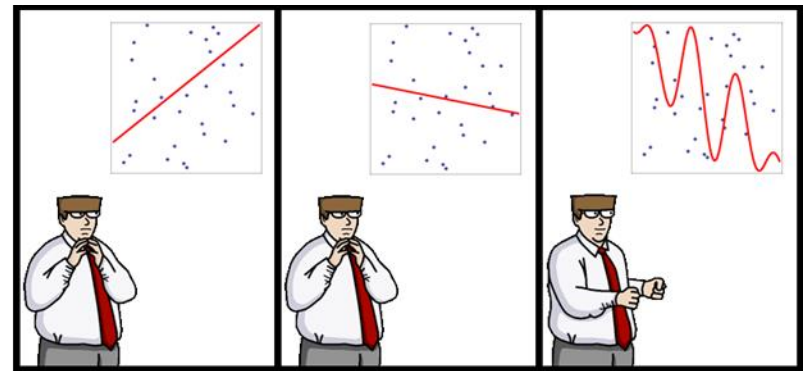
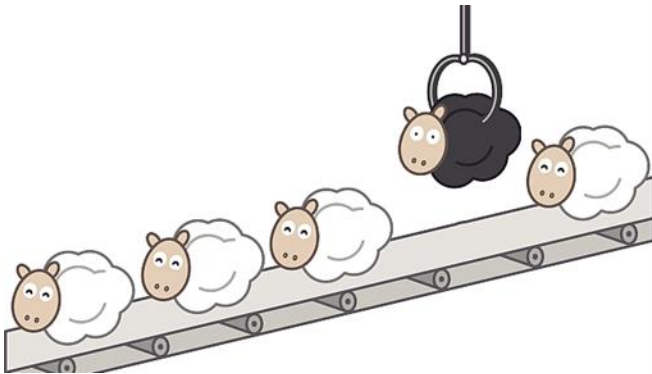
Machine Learning – COMS3**007**

Reinforcement Learning

Benjamin Rosman

Previously on ML...

- We've seen how to solve many cool problems around supervised and unsupervised learning



- Why do we make predictions on data?
 - Usually so a human can **make better decisions**
- **Decision making** itself is important in intelligence
 - How do we automate this?

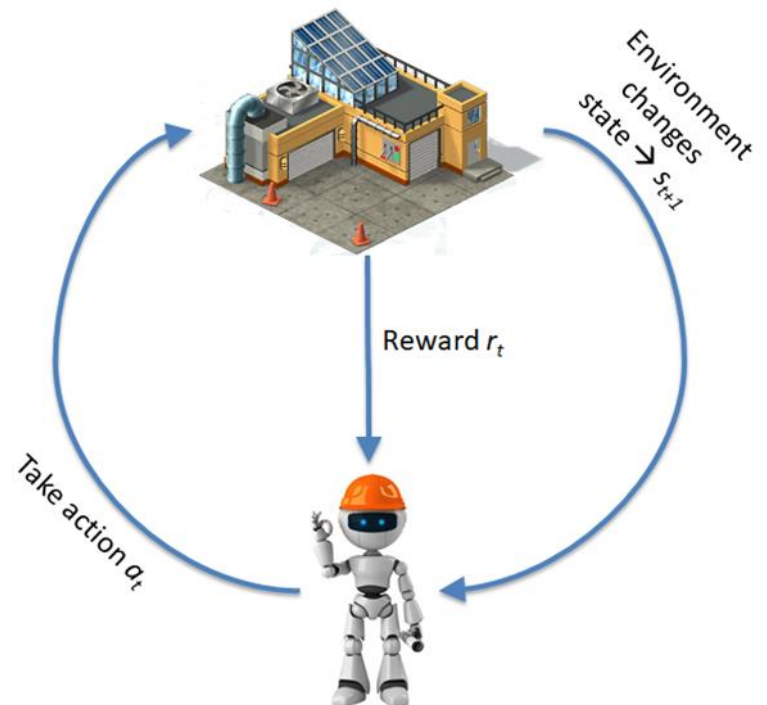
And now for something completely different...

- Reinforcement learning is the branch of machine learning relating to learning in **sequential decision making settings**
- Also think of as behaviour learning
- Supervised learning: single decision point
- **Multiple decision points**
 - How do I know if I'm doing the right thing?
 - How do my decisions now impact the future?
 - Actions affect the environment!



Interacting with the environment

- Decision maker (agent) exists within an environment
- Agent takes action a_t based on the environment state s_t
- Environment state updates
 $s_{t+1} \leftarrow s_t$
- Agent receives feedback as rewards r_t



Modeling the problem

“Future is independent of the past, given the present”

- Markov Decision Process (MDP)

$M = \langle S, A, T, R \rangle$

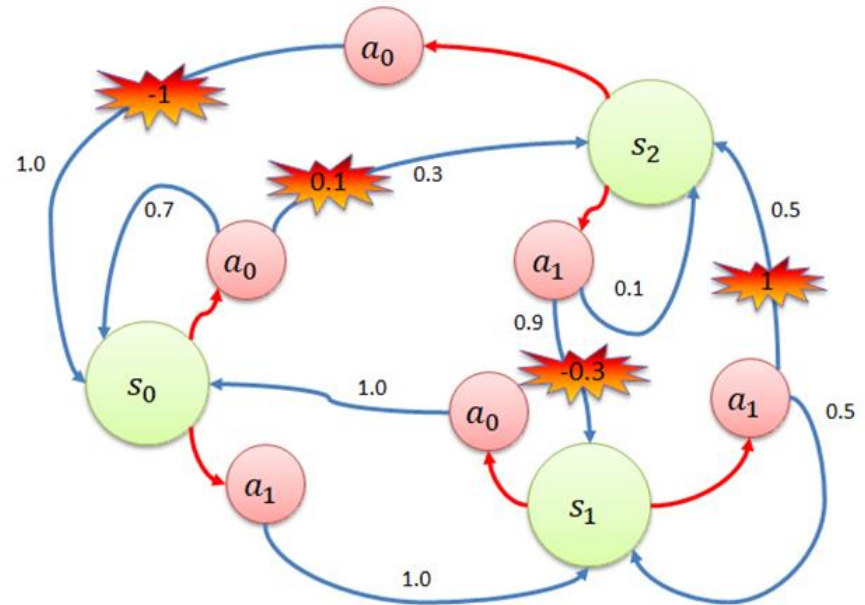
- **S**tates: encode world configurations
- **A**ctions: choices made by agent
- **T**ransition function: how the world evolves under actions

$$T(s, a, s') = P(s_{t+1} = s' | s_t = s, a_t = a)$$

- **R**ewards: feedback signal to agent

$$R(s, a) = E[r_t | s_t = s, a_t = a]$$

$E[.]$ = “expected” (think of as the average reward)

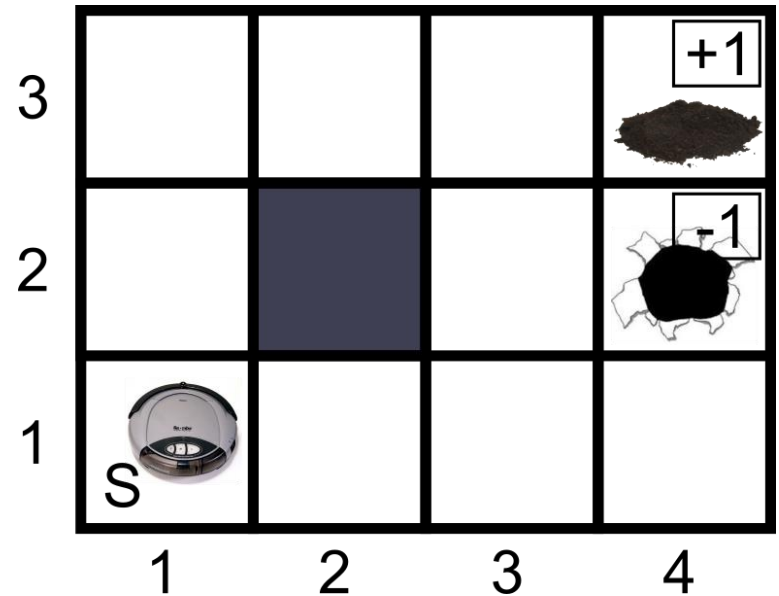


An example

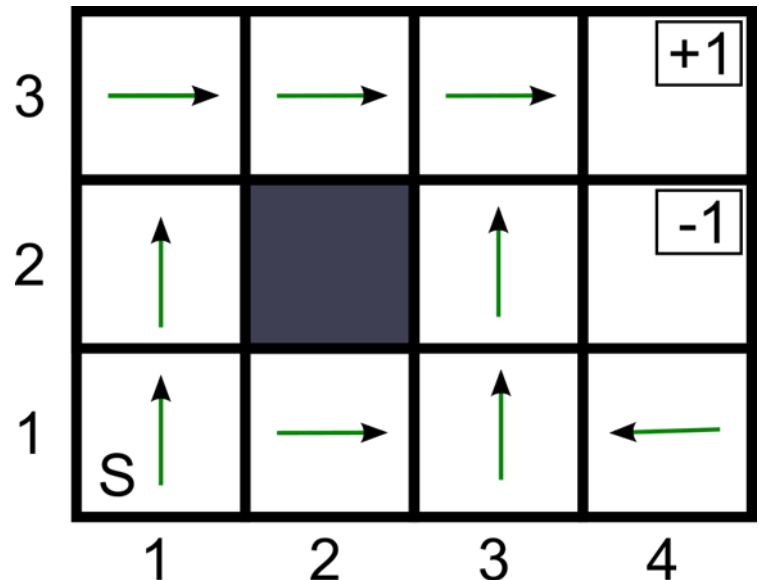
- Cleaning Robot
- States:
 - Position on grid e.g. S is (1,1), goal (4,3)

- Actions: 

- Reward:
 - +1 for finding dirt
 - -1 for falling into hole
 - -0.001 for every move



Optimal behaviour:



Policies

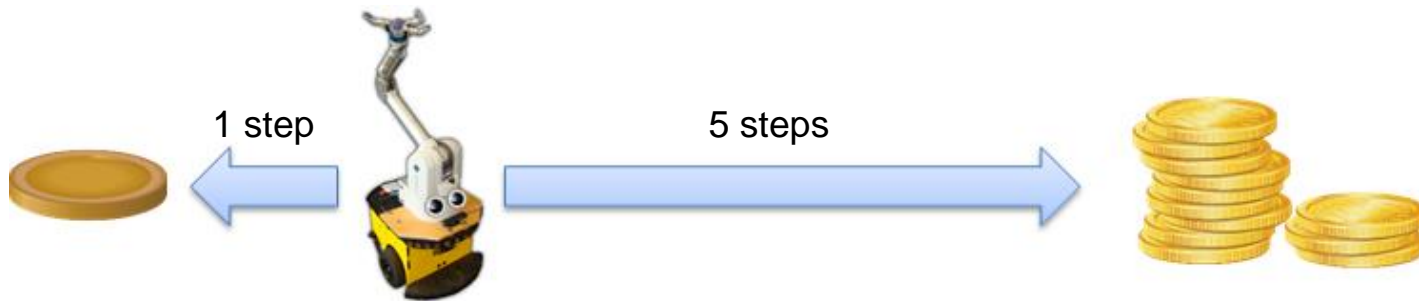
- A **policy** (or behaviour or strategy) π is any mapping from states to actions
 - Deterministic or stochastic

$$\pi(a|s) = P(a_t = a | s_t = s)$$

- Optimal policy π^*
 - Accumulates **maximal rewards** over a trajectory
 - **This is what we want to learn!**

Immediate vs delayed rewards

- Cannot just rely on the instantaneous reward function
 - Tradeoff: don't just act myopically (short term)



- Notion of **value** to codify the goodness of a state, considering a policy running into the future
 - Represented as a **value function**

Value functions

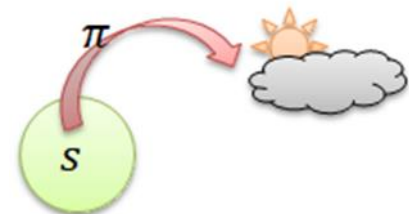
- Value function:
 - The **expected return (R)** starting at state s and then executing policy π

accumulated reward

Discounting: future values count less

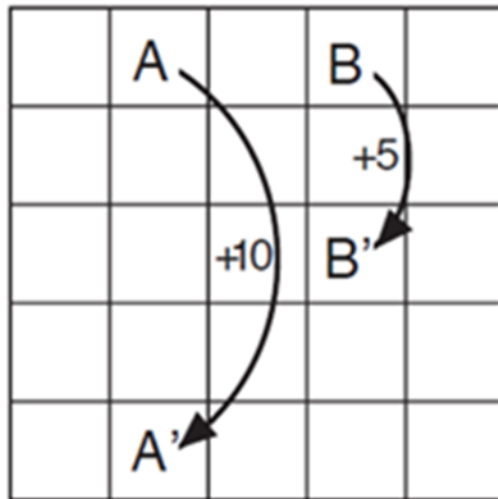
$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\} = E_{\pi}\left\{\sum_{t=0}^{\infty} \gamma^t r_{\pi(s_t)}(s_t, s_{t+1})\right\}$$

- “How good is s under π ?”



Example

- Reward -1 for every move



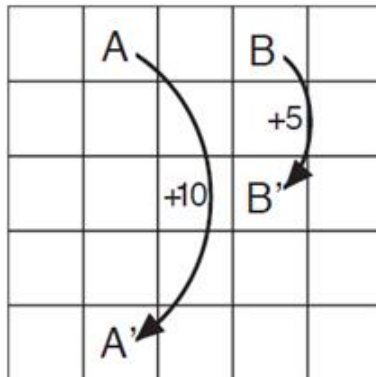
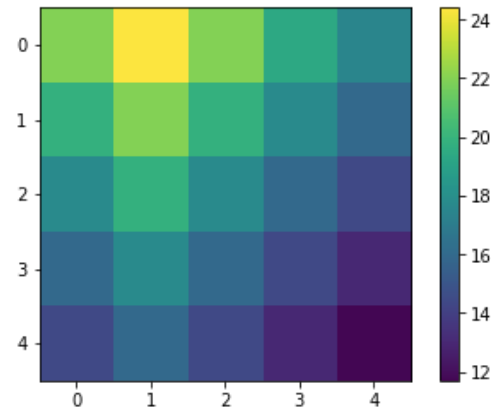
(a)



Actions

Example

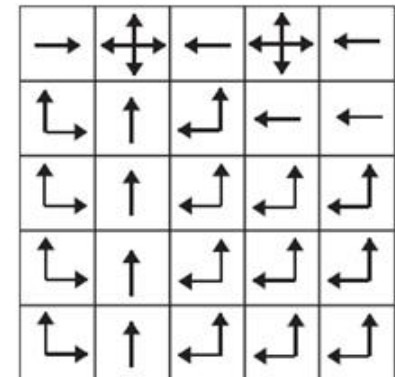
- Optimal policy



a) gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

b) v_*



c) π_*

Value functions: recursion

- $V(s) \Rightarrow$ expected return starting at s and following π
 - Suggests dependence on $V(s')$ from next state s'
- Bellman Equation:

$$\boxed{V^\pi(s)} = \boxed{R(s, \pi(s))} + \gamma \sum_{s'} \boxed{T(s, \pi(s), s')} \boxed{V^\pi(s')}$$

value of s immediate reward for all possible next states the probability of reaching that state with π value of s'

Value functions: optimality

- Similarly, for an optimal policy π^* with optimal value function V^* :
- Bellman Optimality Equation:

$$V^*(s) = \max_a \{ R(s, a) + \gamma \sum_{s'} T(s, a, s') V^*(s') \}$$

take the
best
possible
action

- Note: the optimal value function V^* gives us the optimal policy π^*
 - Choose the action that leads to the best next state

Solving Bellman

- Given the Bellman equation:

$$V^*(s) = \max_a \{ R(s, a) + \gamma \sum_{s'} T(s, a, s') V^*(s') \}$$

- Solve this as a large system of value function equations
 - But: non-linear (*max* operator)
 - So: solve iteratively
- What are we trying to do here?
 - Learn how good each state of the world is, when looking perfectly into the future

Dynamic programming

- **Value Iteration**: dynamic programming
- Iteratively update V (synchronous version)
 - At each iteration i :

- For all states $s \in S$:

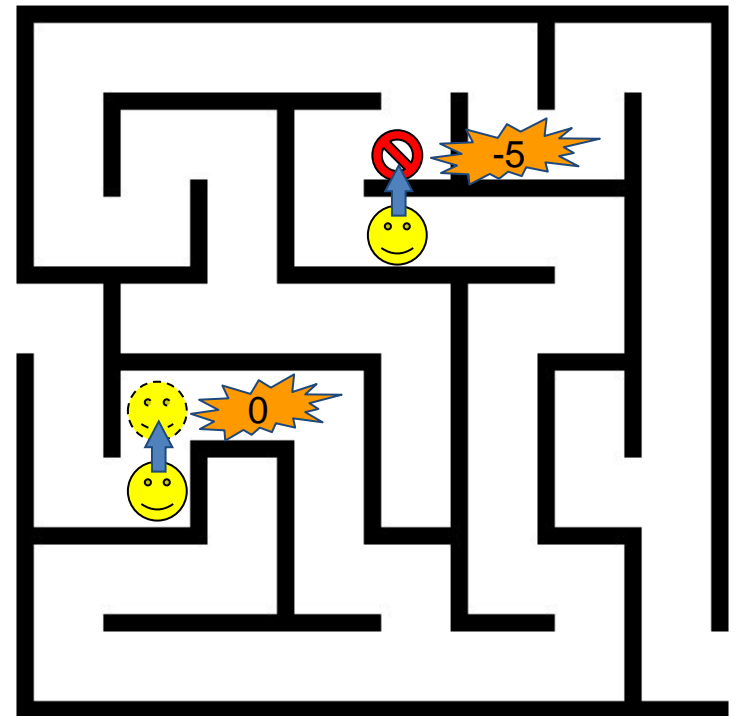
- Update $V(s)$:

$$V_{i+1}(s) := \max_a \left\{ \sum_{s'} T(s, a, s') (R(s, a, s') + \gamma V_i(s')) \right\}$$

- But: this requires the full MDP!!
 - In general, T and R are unknown

Learning from experience

- T and R unknown!
- Instead, generate samples of training data (s, a, r, s') from environment
- Learn from experience
- We need:
 - A way to choose actions
 - A model to store knowledge
 - Value function

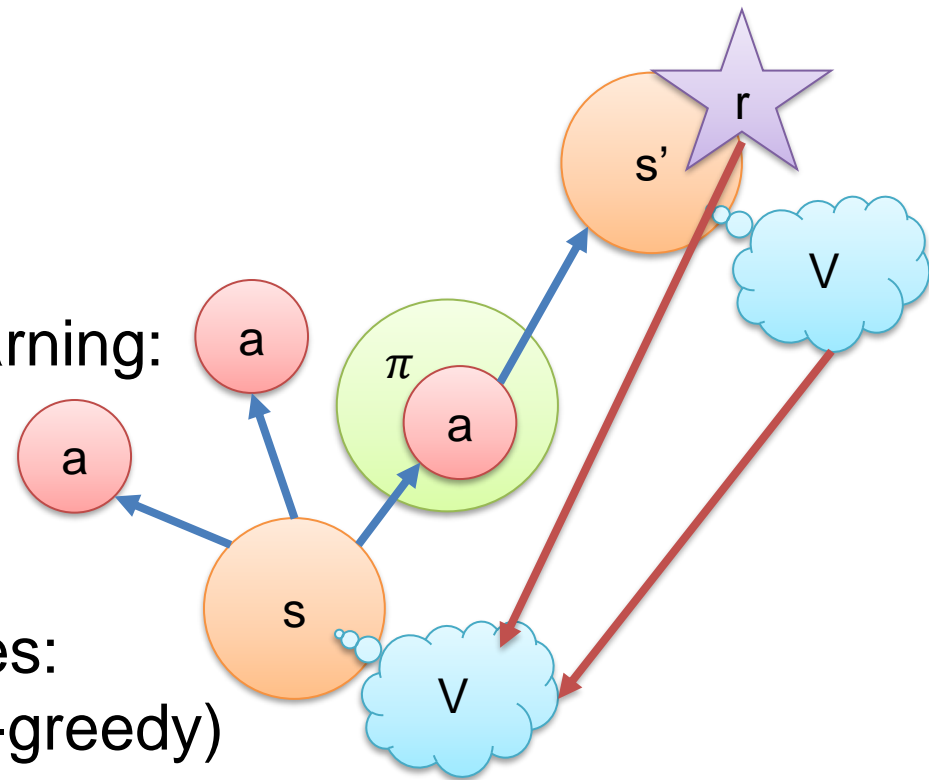


Action selection

- How do we collect data from the environment?
 - Run the best policy we have at the moment
 - But how does that learn anything new??
- **Exploration/exploitation tradeoff!**
 - Sometimes exploit what we have already learned
 - Other times try something new
- **ϵ -Greedy** ($0 < \epsilon \leq 1$):
 - With probability $1 - \epsilon$ **exploit**
 - Choose the best action for a state from π
 - With probability ϵ **explore**
 - Randomly choose action

TD learning

- Temporal Difference (TD) Learning:
 - Initialise V for all $s \in S$
 - For each episode:
 - Reset state s
 - Until episode terminates:
 - Choose action a (ϵ -greedy)
 - Execute a
 - Receive new state s' and reward r
 - Update V using (s, r, s') :



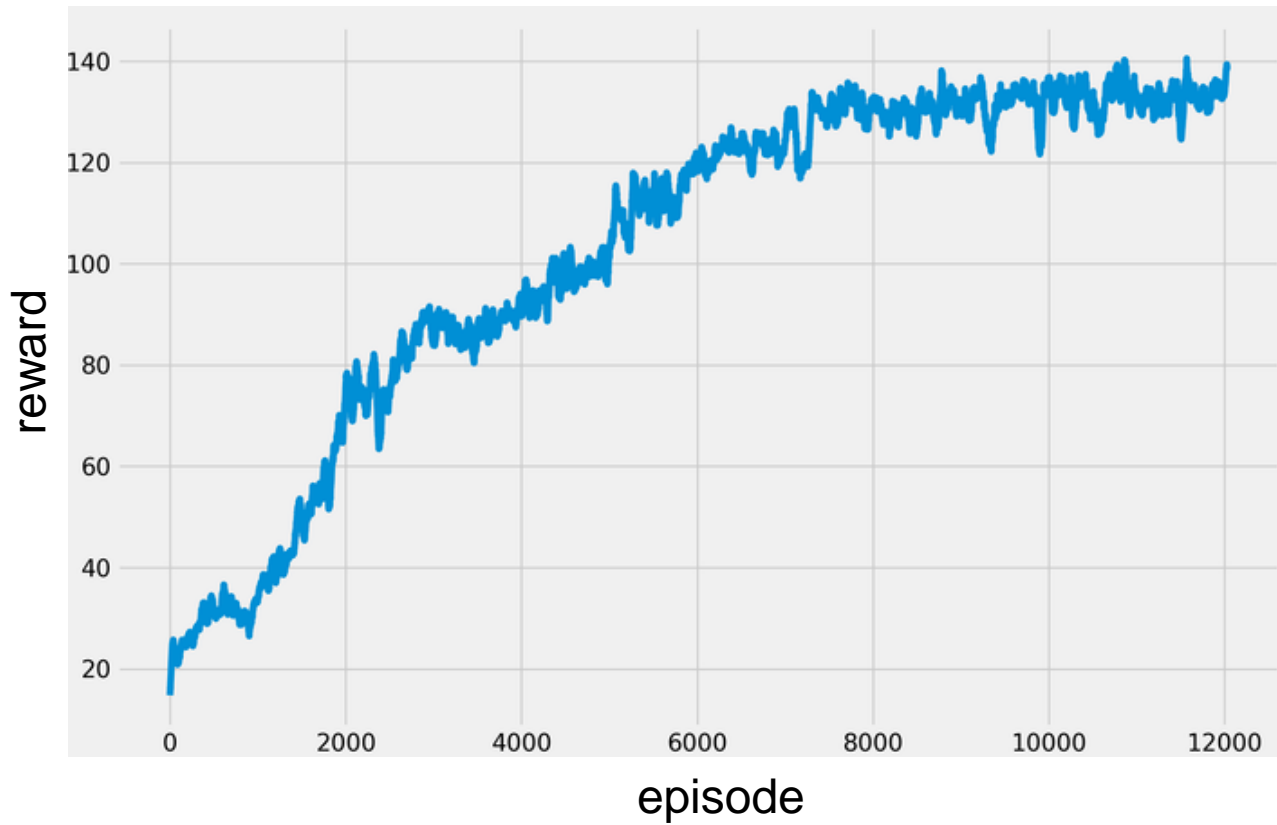
$$V_{i+1}(s) \leftarrow V_i(s) + \alpha \underbrace{(r + \gamma V_i(s') - V_i(s))}_{\text{TD error}}$$

- $s \leftarrow s'$

Learning rate

TD error

Learning curves



But the world is continuous!

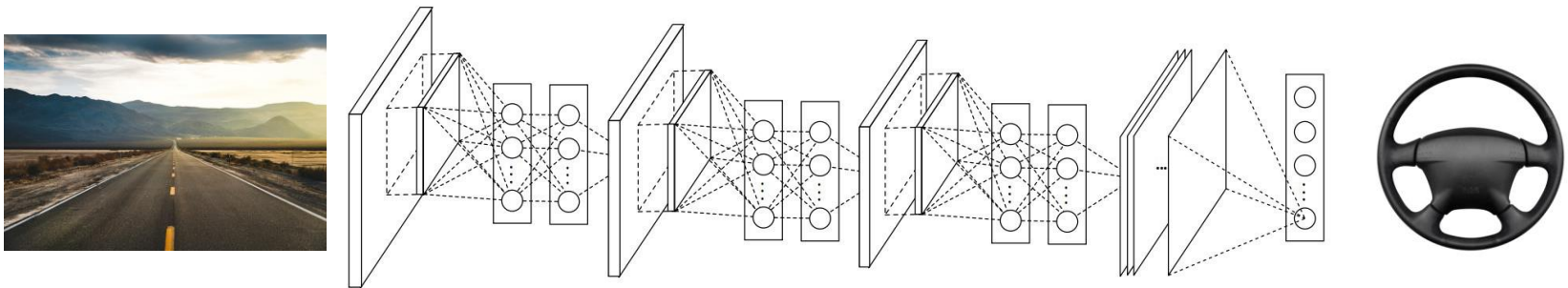


Function approximation

Instead of learning the best action for every state...

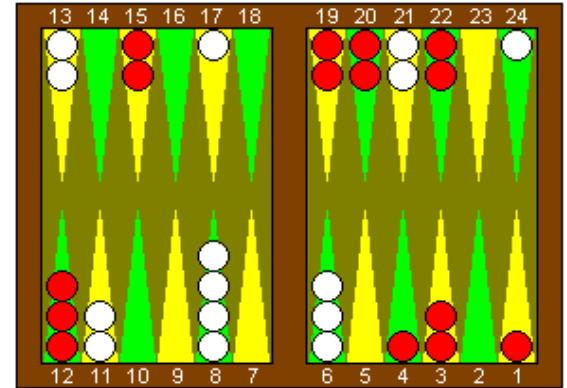
Use a **neural network** to learn a **representation** of the value function

- i.e. a mapping from states to value/actions



Import the whole deep learning toolbox into RL

Backgammon



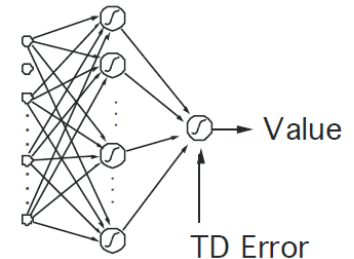
TD-Gammon: Tesauro (1992-1995)

- Learn to play Backgammon through self-play
- 1.5 million games
- Neural network function approximator

States = board configurations ($\approx 10^{20}$)

Actions = moves

$$\text{Rewards} = \begin{cases} 1 & \text{win} \\ -1 & \text{lose} \\ 0 & \text{else} \end{cases}$$



At/near best human play

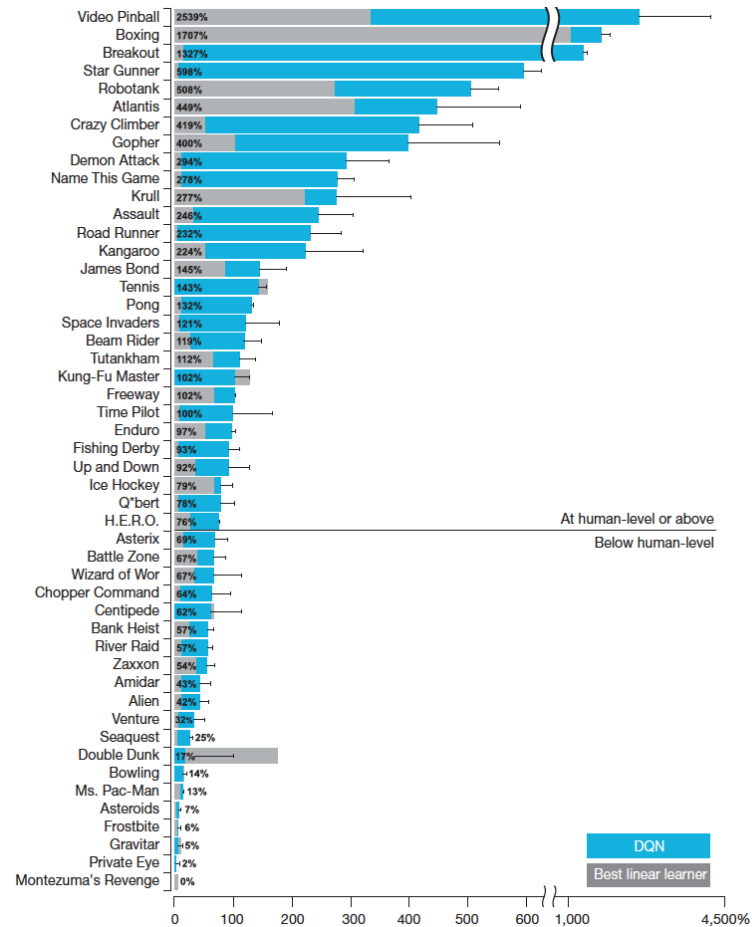
Changed the way the best human players played

Atari

Starting out - 10 minutes of training

**The algorithm tries to hit the ball back, but
it is yet too clumsy to manage.**

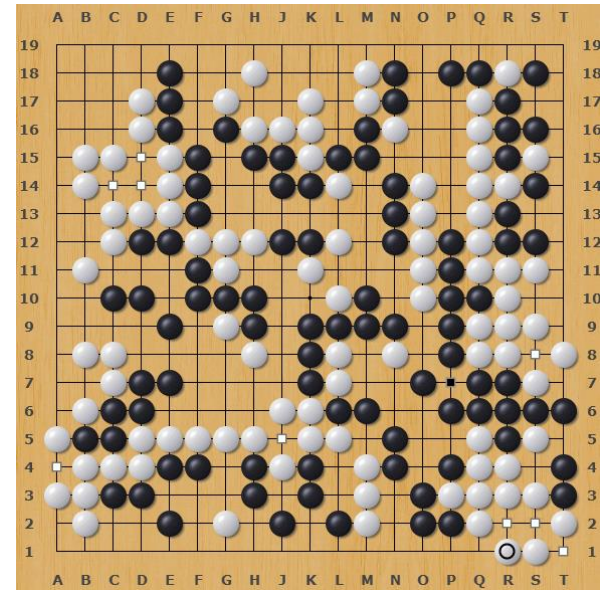
Atari results



Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G. and Petersen, S., 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540), pp.529-533.

But these are simple games!

- Go
 - 361 moves
 - $\sim 10^{174}$ states
 - Adversarial!



- What is the complexity of real-world decisions?

And more games...

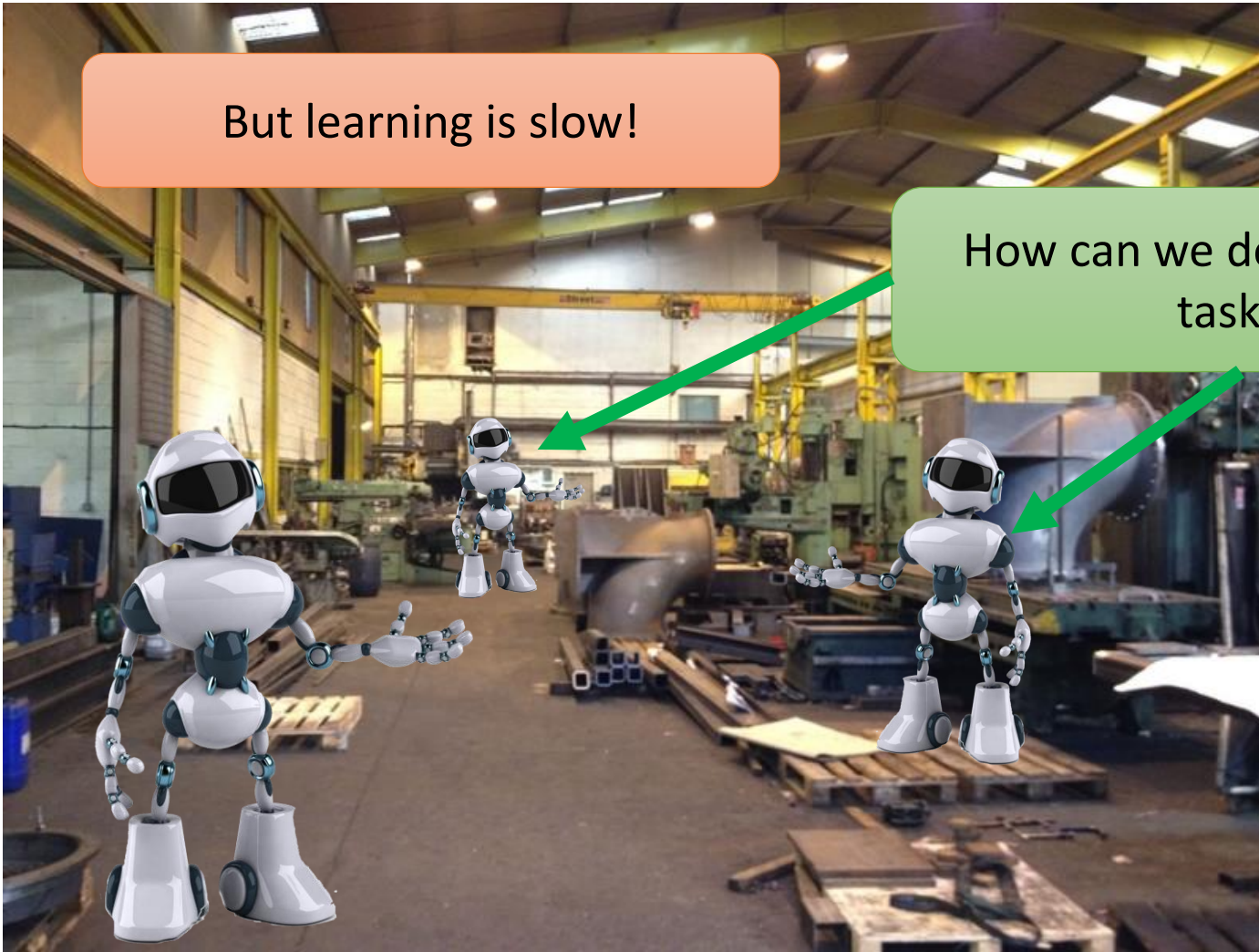


Increasing complexity



But learning is slow!

How can we deal with new tasks?



Learning skills



What about learning reusable **skills**?

- Local controllers / temporally-extended actions
- Use to solve new tasks faster



How?

Watch human demonstrations

Learn how to achieve commonly occurring goals



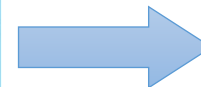
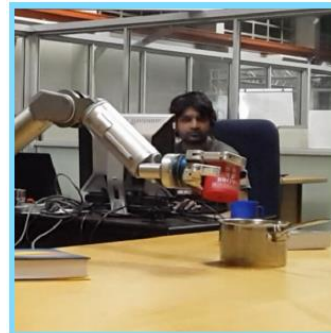
- P. Ranchod, B. Rosman, G. Konidaris. Nonparametric Bayesian Reward Segmentation for Skill Discovery Using Inverse Reinforcement Learning. International Conference on Intelligent Robots and Systems, 2015.
- M. Hiratsuka, N. Makondo, B. Rosman, O. Hasegawa. Trajectory Learning from Human Demonstrations via Manifold Mapping. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2016.



"grasp cup"



"carry to pot"

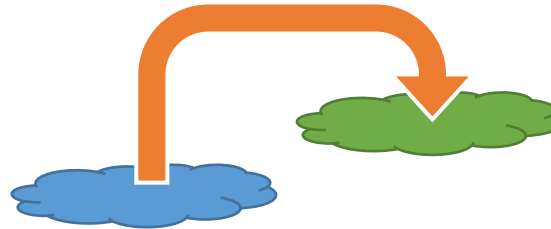


"pour into pot"

Effects of skills

Given **skills**, can we learn:

- When to use them?
 - Preconditions
- What they do?
 - Effects

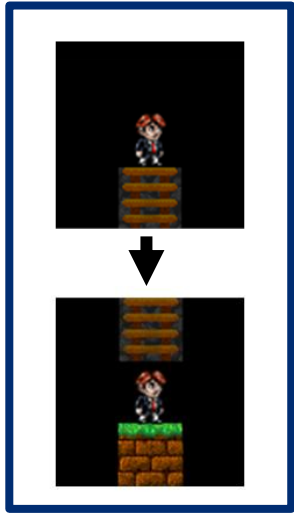


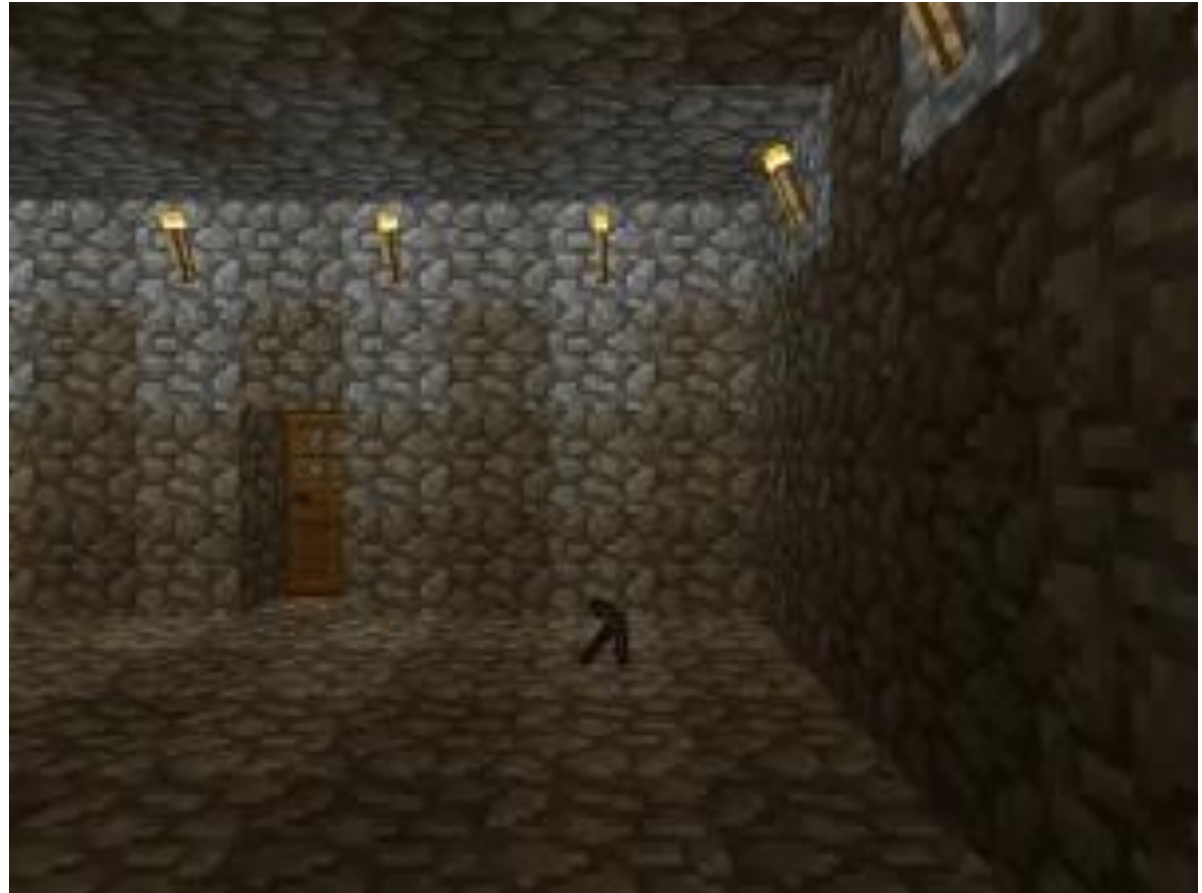
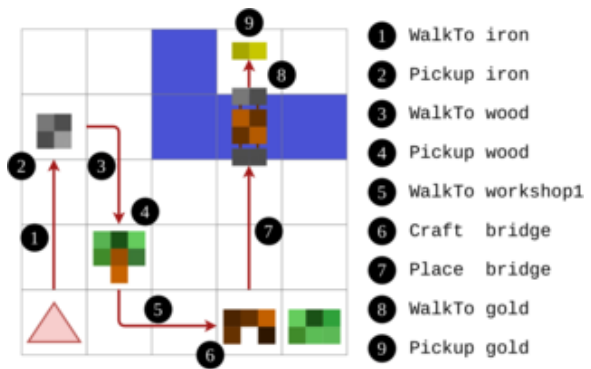
Then we can **plan over long horizons** with them



- S. James, B. Rosman, G. Konidaris. Learning to Plan with Portable Symbols. ICML/IJCAI/AAMAS 2018 Workshop on Planning and Learning, 2018.
- S. James, B. Rosman, G. Konidaris. Learning Object-Centric Representations for High-Level Planning in Minecraft. Object-Oriented Learning (OOL): Perception, Representation, and Reasoning Workshop at ICML, 2020.
- S. James, B. Rosman, G. Konidaris. Learning Portable Representations for High-Level Planning. International Conference on Machine Learning, 2020.







Learning models

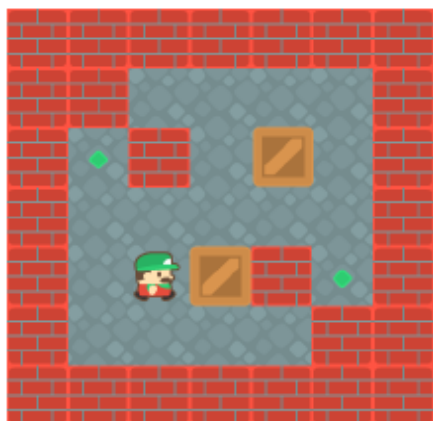
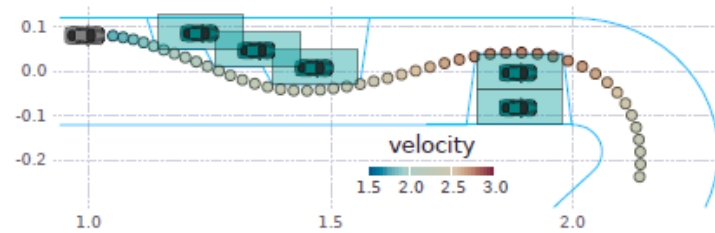
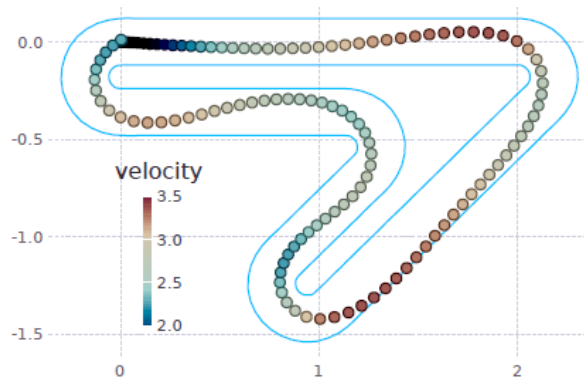
In complex settings, we may need to learn more about **generalisable** components of the environment

Learn **models**:

- Dynamics
- Object interactions



- B. van Niekerk, A. Damianou, B. Rosman. Online Constrained Model-based Reinforcement Learning. Uncertainty in Artificial Intelligence, 2017.
- O. Marom, B. Rosman. Zero-Shot Transfer with Deictic Object-Oriented Representation in Reinforcement Learning. Advances in Neural Information Processing Systems (NeurIPS), 2018.
- O. Marom, B. Rosman. Utilising Uncertainty for Efficient Learning of Likely-Admissible Heuristics. International Conference on Automated Planning and Scheduling (ICAPS), 2020.

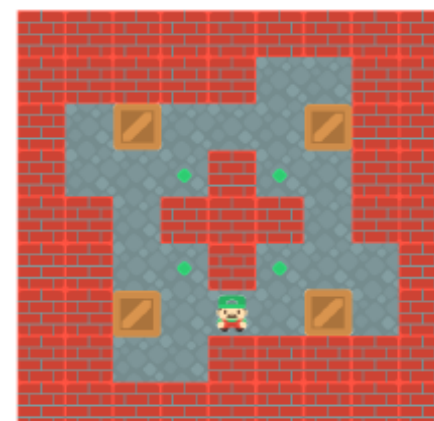


$\sim 8k$ states

$touch_{North}(Person, Wall)$

$Person.y \leftarrow Person.y + 1$

ϕ

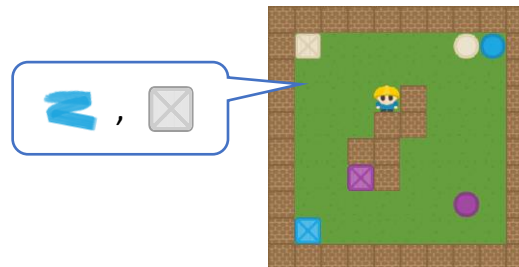


$\sim 1M$ states

Composing behaviours

Given that we can learn these skills, how do we maximise their utility?


- Can we learn to **combine** them to solve new problems?

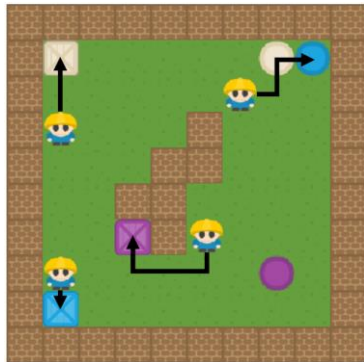


Towards safe AI

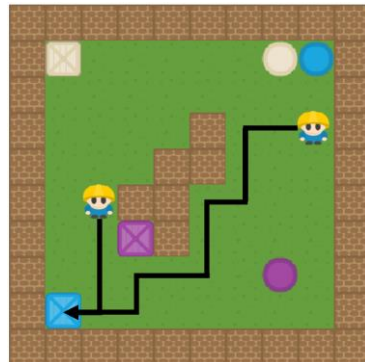



- A. Saxe, A. Earle, B. Rosman. Hierarchy Through Composition with Multitask LMDPs. International Conference on Machine Learning, 2017.
- A. Earle, A. Saxe, B. Rosman. Hierarchical Subtask Discovery with Non-Negative Matrix Factorization. International Conference on Learning Representations, 2018.
- B. van Niekerk, S. James, A. Earle, B. Rosman. Composing Value Functions in Reinforcement Learning. International Conference on Machine Learning, 2019.
- G. Nangue Tasse, S. James, B. Rosman. Logical Composition for Lifelong Reinforcement Learning. 4th Lifelong Learning Workshop at ICML, 2020.
- G. Nangue Tasse, S. James, B. Rosman. A Boolean Task Algebra for Reinforcement Learning. Advances in Neural Information Processing Systems (NeurIPS), 2020.
- V. Cohen, G. Nangue Tasse, N. Gopalan, S. James, M. Gombolay, B. Rosman. Learning to Follow Language Instructions with Compositional Policies. AAAI Fall Symposium on AI for Human-Robot Interaction, 2021.
- G. Nangue Tasse, S. James, B. Rosman. Generalisation in Lifelong Reinforcement Learning through Logical Composition. NeurIPS Deep Reinforcement Learning Workshop, 2021.

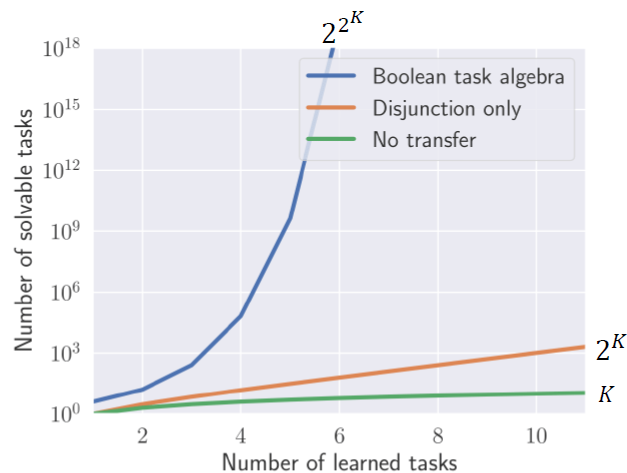
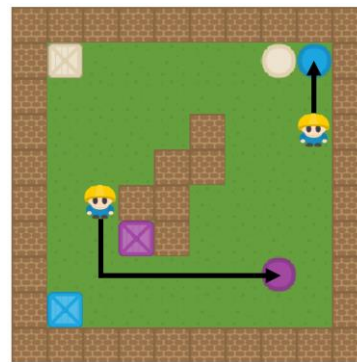
OR 



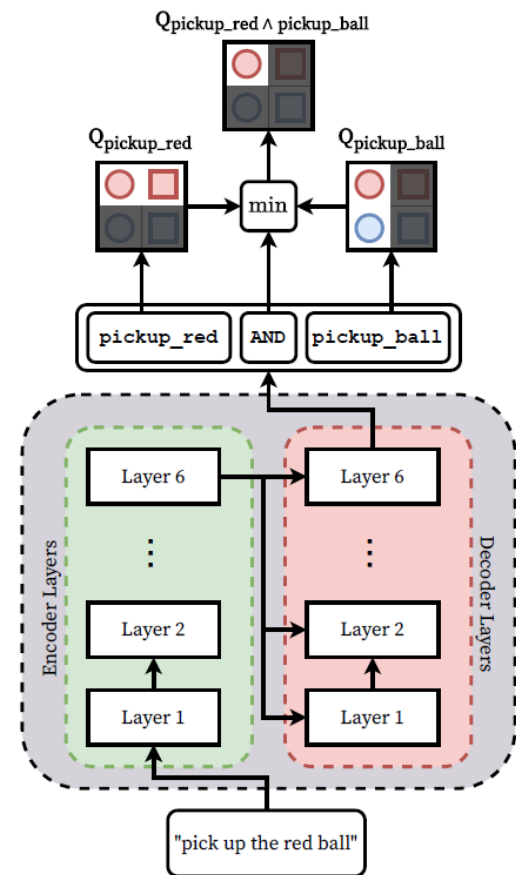
AND 

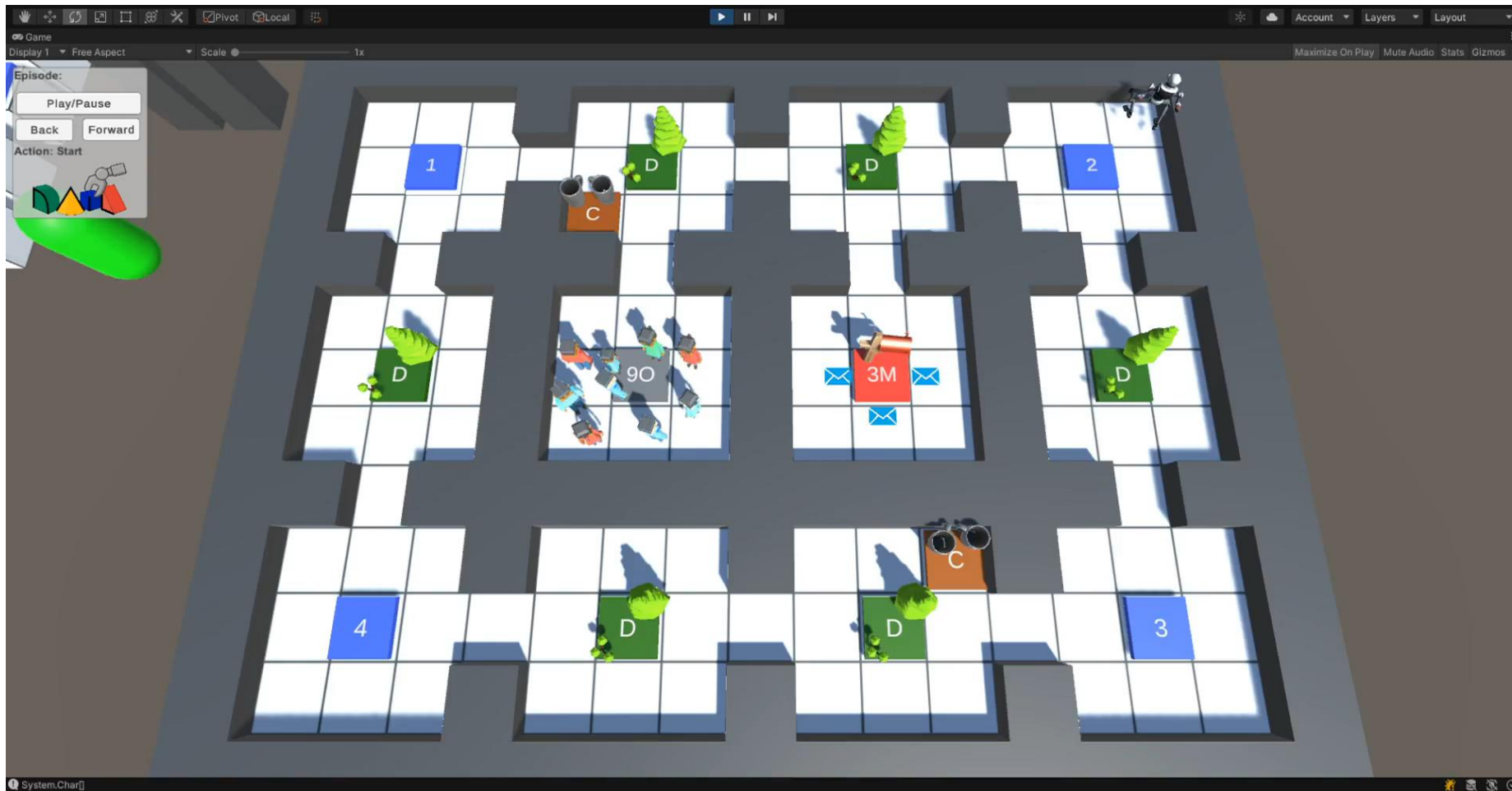


NOT 



- Deliver mail from the mail room to the appropriate recipients in the office.
- Then deliver coffee to the office, until there are no more orders.
- Then keep patrolling between 4 locations.
- Avoid collisions with delicate objects throughout the whole procedure.

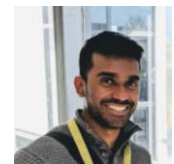
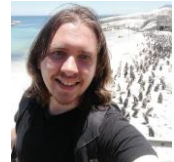
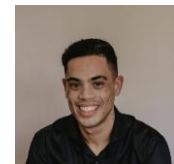
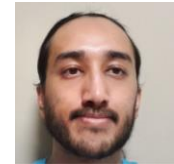






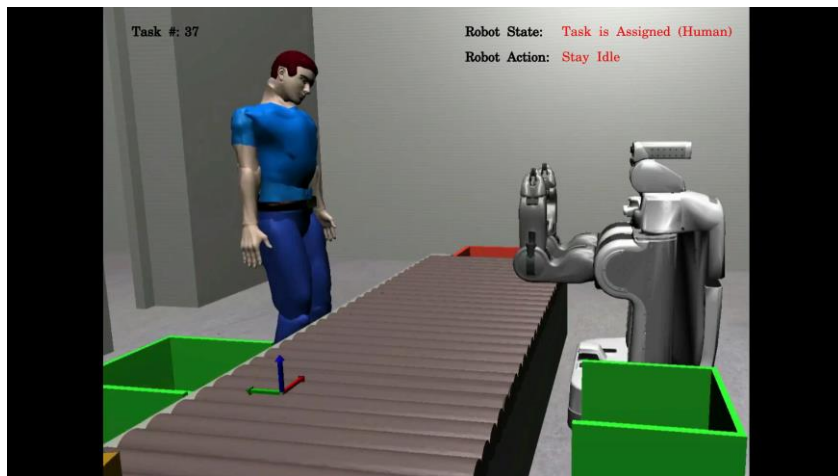
Human-Robot Interaction

- **React** faster to changes in human behaviour
- **Adapt** to different humans
- Learn to **take advice** from humans
- **Teach** humans



- B. Rosman, M. Hawasly, S. Ramamoorthy. Bayesian Policy Reuse. Machine Learning Journal, 104(1), pp. 99-127, 2016.
- O.C. Görür, B. Rosman, G. Hoffman, S. Albayrak. Toward Integrating Theory of Mind into Adaptive Decision-Making of Social Robots to Understand Human Intention. Workshop on the Role of Intentions in Human-Robot Interaction at the International Conference on Human-Robot Interaction, 2017.
- O.C. Görür, B. Rosman, F. Sivrikaya, S. Albayrak. Social Cobots: Anticipatory Decision-Making for Collaborative Robots Incorporating Unexpected Human Behaviors. ACM/IEEE International Conference on Human-Robot Interaction, 2018.
- O.C. Görür, B. Rosman, S. Albayrak.. Anticipatory Bayesian Policy Selection for Online Adaptation of Collaborative Robots to Unknown Human Types. International Conference on Autonomous Agents and Multiagent Systems, 2019.
- T. Love, R. Ajoodha, B. Rosman. Should I Trust You? Incorporating Unreliable Expert Advice in Human-Agent Interaction. Workshop on Human-aligned Reinforcement Learning for Autonomous Agents and Robots at ICDL, 2021.
- S. Singh, B. Rosman. The Challenge of Redundancy on Multi-Agent Value Factorisation. NeurIPS Workshop on Cooperative AI, 2021.

Human-Robot Interaction



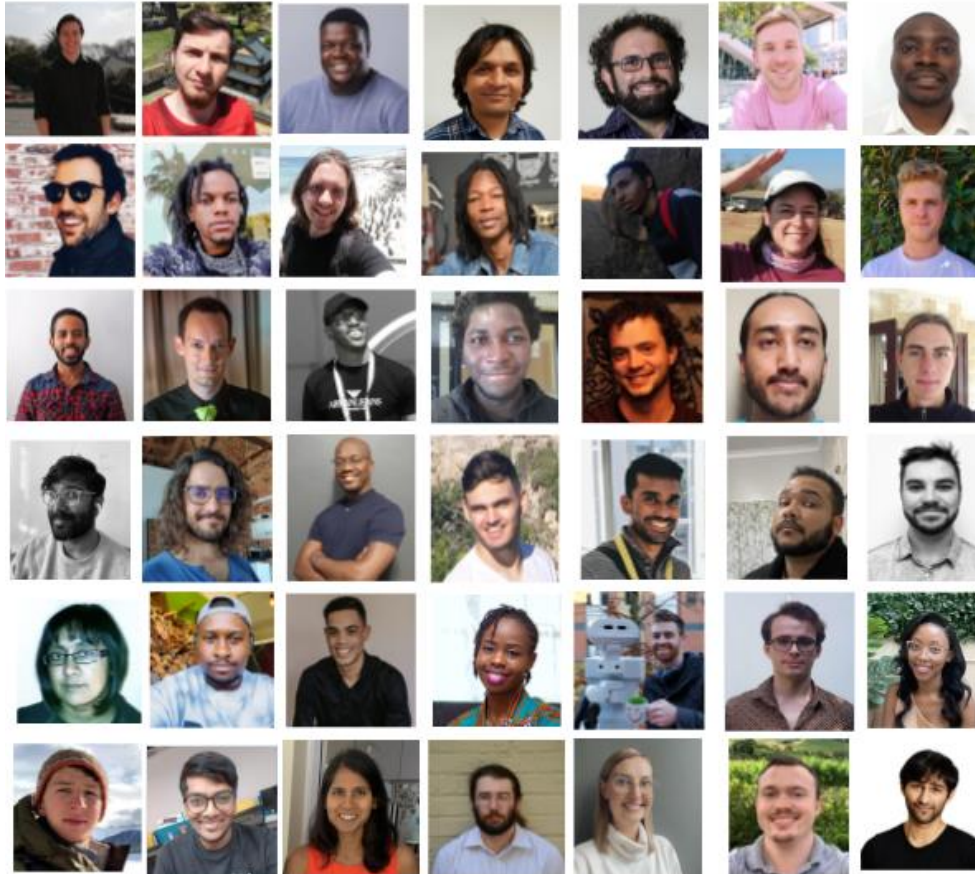
Applications

- **Medical** (diagnostic behaviour)
- **Education** (personalised content)
- **Robotics** (learning generalisable behaviour)
- **Logistics** (autonomous fleet coordination)
- **Agriculture** (maximising crop yield)
- **Bio-sciences** (controlling invasive species)
- ...



- G. Singh, C. Reynolds, M. Byrne, B. Rosman. A Remote Sensing Method to Monitor Water, Aquatic Vegetation, and Invasive Water Hyacinth at National Extents. Remote Sensing 2020, 12(24), 4021.
- H. Combrink, V. Marivate, B. Rosman. A Framework for Undergraduate Data Collection Strategies for Student Support Recommendation Systems in Higher Education. Southern African Conference for Artificial Intelligence Research, 2021.

Thank you!



Thanks to our funders:



www.raillab.org

