UPPSALA
UNIVERSITET

Reinforcement Learning

Lecture 11 - Recent advances and remaining challenges

Dominik Baumann

2022

Department of Information Technology

Repetition

- **The return:** $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$
- **State-value function:** $v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$
- **Action-value function:** $q_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$
- **Policy:** rules for how the agent should act in different states: $a \sim \pi(a \mid s)$

- One approach to RL is to find an estimate Q of the optimal q-function. Then use the policy $\pi(s) = \arg\max_a Q(s, a)$
- During training, however, we need exploration (e.g., ε-greedy, UCB)
- **Tabular:** if both $\mathcal{S}$ and $\mathcal{A}$ are finite we can represent $Q(s, a)$ as a table with one element for each s/a-pair
- **Monte Carlo:** let $Q(S, A)$ be the average G we have seen from S, A
- **SARSA:** (on-policy, bootstrapping)

$$Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma Q(S', A') - Q(S, A))$$

- **Q-learning:** (off-policy, bootstrapping)

$$Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma \max_a Q(S', a) - Q(S, A))$$

- If the state space is very large (or even continuous) it is not suitable to represent $Q(s, a)$ with a table:
  - The table will be too big (or even infinitely big)
  - No generalization: only s/a-pairs seen will be updated
- **Idea:** represent q with function approximation:

$$\hat{q}(s, a, \boldsymbol{w}) \approx q(s, a)$$

- **Stochastic (semi)-gradient:** the target U_t is some approximation of $q(s, a)$,

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha[U_t - \hat{q}(S_t, A_t, \boldsymbol{w}_t)]\nabla\hat{q}(S_t, A_t, \boldsymbol{w}_t)$$

- **Monte Carlo:** use $U_t = G_t$ computed from experience.
- **SARSA:** use $U_t = R_{t+1} + \gamma\hat{q}(S_{t+1}, A_{t+1}, \boldsymbol{w}_t)$ from experience and bootstrapping

- Define a feature vector $\boldsymbol{x}(s, a)$ and use

$$\hat{q}(s, a, \boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{x}(s, a)$$

- **Gradient:** $\nabla \hat{q}(s, a, \boldsymbol{w}) = \boldsymbol{x}(s, a)$
- Existing results show that gradient methods based on SARSA converge
- Can often encode prior knowledge into features
- To get good results in complex environments, we need a good choice of features

Convergence of prediction algorithms

Are there convergence guarantees?

On-policy methods

Method	Tabular	Linear	Nonlinear
MC	Yes	Yes	Yes
TD	Yes	Yes	No
Gradient-TD	Yes	Yes	Yes

Off-policy methods

Method	Tabular	Linear	Nonlinear
MC	Yes	Yes	Yes
TD	Yes	No	No
Gradient-TD	Yes	Yes	Yes

Convergence of control

Method	Tabular	Linear	Nonlinear
MC	Yes	(Yes)	No
SARSA	Yes	(Yes)	No
Q-learning	Yes	No	No
Gradient Q-learning	Yes	Yes	No

Policy gradient: REINFORCE converges to a local minimum even for nonlinear.

Some actor-critic algorithms have similar convergence properties.

The deadly triad

- Function approximation
- Bootstrapping
- Off-policy learning

When we combine these three concepts, we risk instability and divergence!

See textbook Chapter 11.3 for more discussion.

Do we need all of them?

- Function approximation needed for more complex environments
- We have seen that bootstrapping helps a lot for reducing variance
- Off-policy can help with sample efficiency (reuse old data)

Nonlinear: In the book they talk about function approximation in general. When we consider nonlinear cases (e.g. deep RL), the situation is even worse.

Deep Learning

- The goal is to find a function that maps inputs $\boldsymbol{x}$ to output y
- Using parametrized functions and find estimate $\widehat{\boldsymbol{\theta}}$,

$$\hat{y} = g(\boldsymbol{x}; \boldsymbol{\theta})$$

- **Linear regression:** use

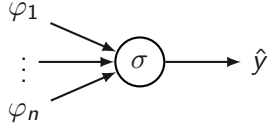
$$g(\boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{\varphi}(\boldsymbol{x})^\top \boldsymbol{\theta}$$

  - $\boldsymbol{\varphi}(\boldsymbol{x})$ should contain features that can describe the function we try to estimate
  - Analytical solutions, and a lot of theory
- **Nonlinear regression:** when $g(\boldsymbol{x}; \boldsymbol{\theta})$ depends on $\boldsymbol{\theta}$ in a nonlinear way
  - No analytical solution in general
  - Common to use the idea of stochastic gradient descent

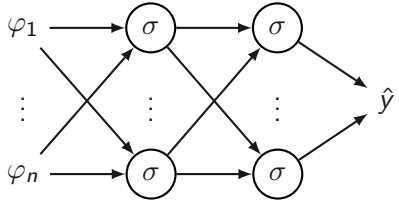
We can generalize linear regression by adding a nonlinear transformation:

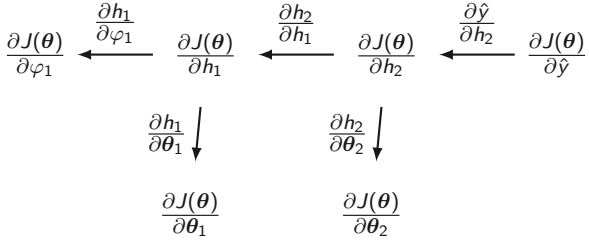
$$\hat{y} = \sigma(\boldsymbol{\varphi}(\boldsymbol{x})^\top \boldsymbol{\theta}),$$

where $\sigma(\cdot)$ is often called an *activation function*.



Can combine several of these and make the network deeper





- Assume we want to estimate a function $g(\boldsymbol{x}; \boldsymbol{\theta})$ using a deep neural network
- Choose a **loss function**, e.g., $J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$
- Try to find

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

- Can use the chain rule to compute gradient via **backpropagation**

- Sample gradient of expected loss $\bar{J}(\boldsymbol{\theta}) = \mathbb{E}[J(\boldsymbol{\theta})]$

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \sim \mathbb{E}\left[\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right] = \frac{\partial \bar{J}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

- Adjust $\boldsymbol{\theta}$ down the sampled gradient

$$\Delta\theta \propto \frac{\partial \bar{J}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

Deep Reinforcement Learning

Deep learning + reinforcement learning

UPPSALA UNIVERSITET

- Deep learning is a general-purpose framework for representation learning:
  - Given an **objective**
  - Learn **representation** that is required to achieve objective
  - Directly from **raw inputs**
  - Using minimal domain knowledge
- Reinforcement learning defines the objective

- Approximate the model, policy, and/or value functions using deep neural networks
- Several papers using neural networks in RL during the 1980s
- TD-Gammon (1992) used a multi-layered neural network for function approximation to play backgammon
- Really took of around 2013 with DQN playing Atari games and AlphaGO beating professional Go players (2015)

Why deep?

- **Universal function approximation theorem:** any continuous function $f(x)$ can be approximated by a neural network with arbitrary precision
- But might need to be large (many neurons and/or deep)

Drawbacks:

- Many parameters, so we often need much more training
- Nonlinear, so we lose many previous convergence guarantees.
- Introduce several new hyperparameters that are hard to tune

Value-based deep RL

Q-learning

- Try to learn $q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$
- Optimal Q-values should obey Bellman equation

$$q^*(s, a) = \mathbb{E}_{s'}[G_t + \gamma \max_{a'} q^*(s', a') \mid s, a]$$

- Treat expression in expectation as target
- Minimize MSE loss by stochastic gradient descent
- **Converges** to q^* for tabular approaches
- **Diverges** when using neural networks due to
  - Correlations between samples
  - non-stationary targets

Deep Q-networks (DQN): experience replay

- Over time, the agent collects samples $(s_t, a_t, r_{t+1}, s_{t+1})$
- Sample experiences from that collection and apply stochastic gradient descent

$\rightarrow$ Random sampling removes correlations

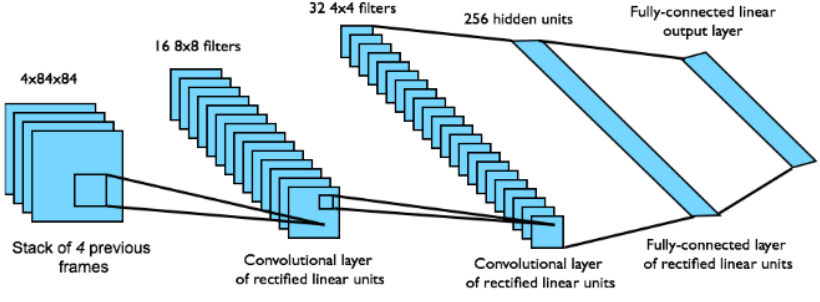
- To deal with non-stationarity, fix target parameters $\boldsymbol{\theta}$ when computing the loss

Human-level control through deep reinforcement learning[1]

- DQN created widespread interest when it learned to play various Atari games at the level of human experts
- It learned $Q(a, s)$ directly from pixels
- Input state s is stack of raw pixels from last four frames
- Output is $Q(s, a)$ for 18 joystick/button positions
- Reward is change in score for that step

[1] Volodymyr Mnih et al., "Human-level control through deep reinforcement learning," Nature, 2015.

DQN in Atari games



Improvements over the last years

- **Double DQN:** use two identical neural network models
  - One is used to actually compute the Q-value, the other to select the next action
  - removes upward bias
- **Prioritized replay:** weight experience according to surprise
  - Sample experiences more often that had a huge learning impact
- **Dueling network:** split Q-network into $V(s, \boldsymbol{v})$ and $A(s, a, \boldsymbol{\theta})$
  - $V(s, \boldsymbol{v})$: value function
  - $A(s, a, \boldsymbol{\theta})$: advantage function
- If we combine all of them, can get three times higher mean Atari score than DQN

Policy-based deep RL

- Represent policy by a deep network with parameters ϕ

$$a = \pi(a \mid s, \phi)$$

- Define objective function as total discounted reward

$$J(\phi) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid \pi(a \mid s, \phi)]$$

- Optimize objective function end-to-end by stochastic gradient descent

- Gradient of a stochastic policy $\pi(a \mid s, \phi)$:

$$\frac{\partial J(\phi)}{\partial \phi} = \mathbb{E}\left[\frac{\partial \log \pi(a \mid s, \phi)}{\partial \phi} Q^{\pi}(s, a)\right]$$

- **Actor-critic:** estimate value function $Q(s, a, \theta) \approx Q^{\pi}(s, a)$
- Update policy parameters ϕ by stochastic gradient ascent

Asynchronous advantage actor-critic (A3C)[2]

- Estimate state-value function

$$V(s, \boldsymbol{v}) \approx \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \ldots \mid s]$$

- Q-value estimated by an n-step sample

$$Q_t = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V(s_{t+n}, \boldsymbol{v})$$

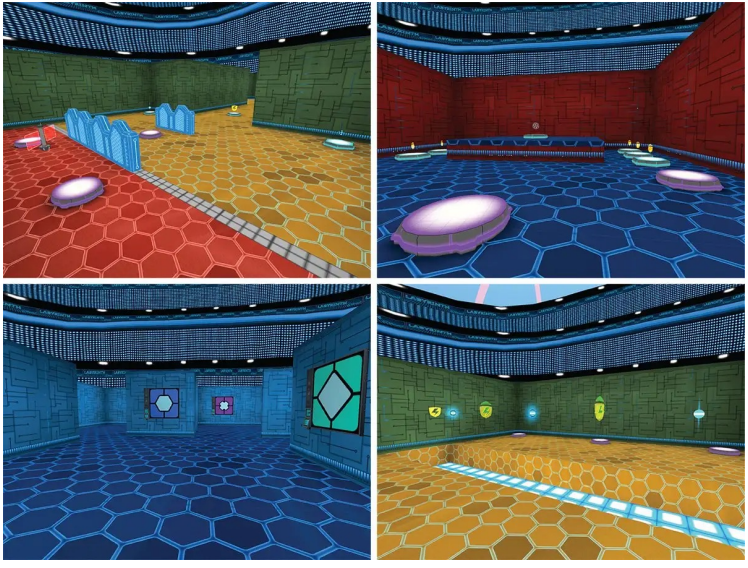
- Actor is updated toward target

$$\frac{\partial J(\phi)}{\partial \phi} = \frac{\partial \log \pi(a_t \mid s_t \phi)}{\partial \phi}(q_t - V(s_t, \phi))$$

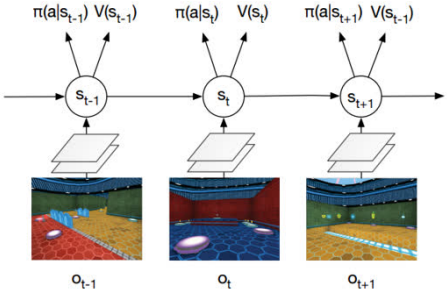
- Critic is updated to minimize MSE w.r.t target

[2]Volodymyr Mnih et al., "Asynchronous methods for deep reinforcement learning," International Conference on Machine Learning, 2016

A3C in Labyrinth



A3C in Labyrinth



- Observations o_t are raw pixels from current frame
- State s_t is a recurrent neural network
- Outputs both $V(s)$ and softmax over actions $\pi(a \mid s)$
- Task is to collect apples and escape
- Demo: https://youtu.be/nMR5mjCFZCw

Model-based deep RL

Model-based vs model-free RL

- In model-based RL we aim at creating a model of the environment and plan based on that model
- Pros:
  - Tends to have higher sample efficiency than model-free RL
  - Transferable to other tasks (more on that later today)
  - Model uncertainty can help to judge how confident we should be in our decision-making
- Cons:
  - Vapnik's principle: "When solving a problem of interest, do not solve a more general problem as an intermediate step"
  - Two sources of errors (model and policy)
  - Computationally more demanding
- Some of the most effective recent approaches combine both

- Usually, we try to learn a model $s_{t+1} = f(s_t, a_t)$
- For Δt small, changes become small and $s_{t+1} = s_t$ may be a good approximation
- Instead: learn $\hat{f}(s_t, a_t, \psi)$ such that $\hat{s}_{t+1} = s_t + \hat{f}(s_t, a_t, \psi)$
- Train dynamics model using stochastic gradient descent on the MSE

- Goal: find action sequence $A_t^H = (a_t, \ldots, a_{t+H})$ such that

$$A_t^H = \arg\max_{A_t^H} \sum_{t'=t}^{t+H} R(\hat{s}_{t'}, a_{t'}) \text{ s.t. } \hat{s}_t = s_t, \hat{s}_{t'+1} = \hat{s}_{t'} + \hat{f}(\hat{s}_{t'}, a_{t'}, \psi)$$

- Exact computation often infeasible but can get approximate solution
- Model predictive control: execute first action in A_t^H, then solve optimization problem again

Hybrid model-based model-free approach[3]

- Model-based approaches often more sample efficient but final performance can be worse than model-free approaches

$\rightarrow$ Initialize model-free learner with model-based learner

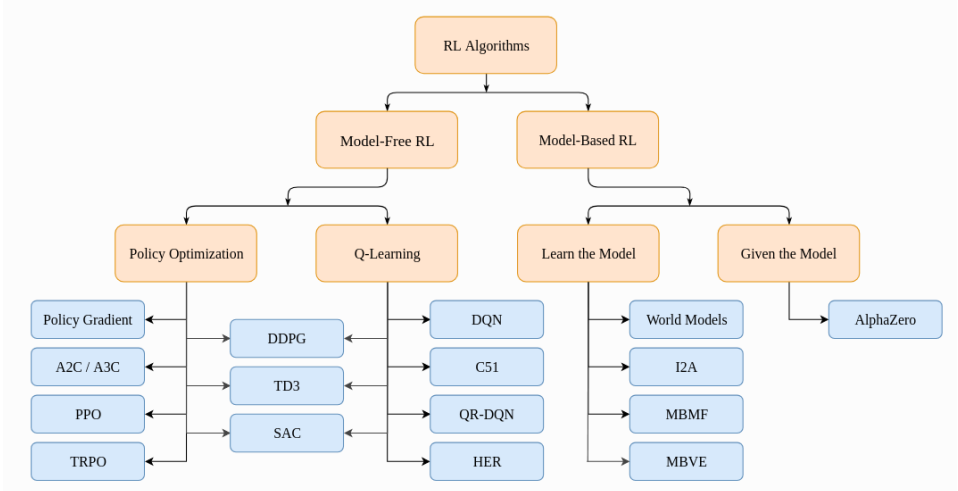
- Demo: https://sites.google.com/view/mbmf

[3] Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine tuning," IEEE International Conference on Robotics and Automation, 2018.

Summary of deep RL

Deep reinforcement learning methods

- Last 8-9 years: many new ideas for making deep RL more stable and efficient
- However, still no clear "winner"
- Taxonomy from OpenAI Spinning Up:



A non-exhaustive, but useful taxonomy of algorithms in modern RL. Citations below.

Open challenges and new directions

Open challenges and new directions

UPPSALA UNIVERSITET

- Deep reinforcement learning (DRL) has seen tremendous successes and breakthroughs in recent years
- However, there are still many open problems
  - DRL often only works for specific tasks (e.g., playing Go)

  $\rightarrow$ Multi-task learning, transfer learning
  - Sample efficiency is a big problem, especially for learning on real systems

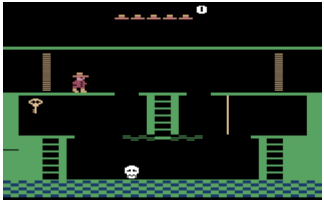
  $\rightarrow$ Meta-learning
  - DRL agents have no deeper understanding of themselves and the world around them

  $\rightarrow$ Causal reinforcement learning

Multi-task and transfer learning

- RL agents have mastered games like Go or StarCraft
- Usually, only able to play one particular game
- Humans can generalize and transfer knowledge about gaming in general obtained from learning, e.g., Go, to other games
- Can we also achieve this for RL?

Recap: Montezuma's revenge



- Why is this game more difficult for DRL agents than for humans?
- We can transfer knowledge from other games, real world experiences
- We know that keys can be used to open doors
- We know that we can use ladders to climb up and down
- We may not know exactly what the skull does but we can guess that it is probably better not to touch it

What can learning agents use to achieve the same?

UPPSALA UNIVERSITET

- Q-function: which actions and states are good
- Policy: which actions are useful
- Models (in case we use them): what are underlying physical laws
- Features/hidden states: provide us with good representation

Terminology

- **Transfer learning:** using experience from one task or one set of tasks for faster learning and better performance on a new task (task is an MDP)
- **Shot:** how often do we "give it a shot" in the new target domain
- **Forward transfer:** train on one task, transfer to a new task
- **Multi-task transfer:** train on many tasks, transfer to a new task

Forward transfer – overview

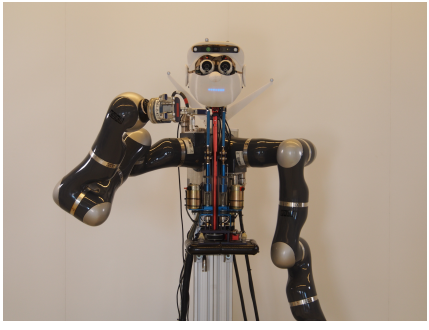
- Why is this challenging?
  - **Domain shift:** What works well in the source domain might not work in the target domain
  - **Different MDPs:** for instance, some actions might only be available in the source domain
- Approaches:
  - We can just train on one task and then hope for the best on the other task
  - We can increase robustness by manipulating the source domain
  - Include some randomness in the environment we use for training
  - Artificially make the source environment more challenging

Forward transfer – example

- In robotics, training on real hardware is time consuming and can damage the hardware

$\rightarrow$ Train agents in simulation and then directly evaluate on hardware

- **But:** simulation and real world are not the same
- Include noise in simulation environment to make it more challenging
- Example: learn resource-aware control policy for robot Apollo

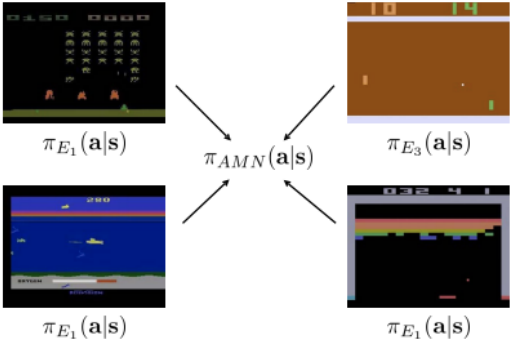


Forward transfer – example

Videos available at
https://sites.google.com/view/learn-event-triggered-control, taken from: Niklas Funk, Dominik Baumann, Vincent Berenz, and Sebastian Trimpe, "Learning event-triggered control from data through joint optimization," IFAC Journal of Systems and Control, 2021.

- We often use model-based reinforcement learning in this setting
- Try to build a model from experience in one set of tasks and adapt the model to the new task
- What is challenging about this?
  - **Gradient interference:** improving in one domain might decrease performance in other domain
  - **Winner-take-all problem:** if we start to be good in one task algorithm might prioritize that one at expense of others
- Approaches:
  - **Distillation:** combine multiple policies into one
  - **Contextual policies:** policies that are told what to do
  - **Modular neural networks** for different tasks

Multi-task transfer – distillation[4]

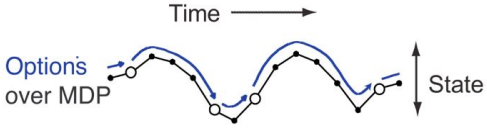


- **Ensemble models:** learn models for different tasks and average their predictions
- **Distillation:** train on the outputs of the ensemble's predictions
- Agent achieved expert-level performance in many Atari games simultaneously

[4] Emilio Parisotto, Jimmy Ba, and Ruslan Salakhutdinov, "Actor-mimic deep multitask and transfer reinforcement learning," arXiv, 2015.

Multi-task transfer – contextual policies

- Recap: contextual bandits from the last lecture
- Bandits whose reward probability depends on a context variable
- Similarly, introduce context for MDP
- Policy $\pi(a \mid s)$ becomes $\pi(a \mid s, c)$ with context parameter c
- Contexts can then represent different tasks

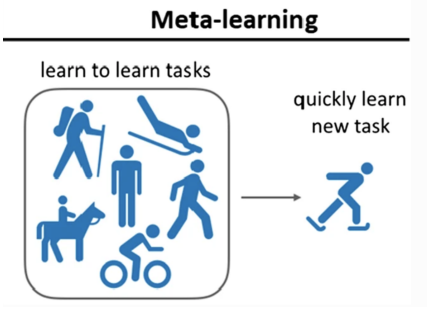


- Learning at different time scales, example: options framework:
  - At each time step, a policy over options determines in which option we are (e.g., which task)
  - For every option, we have an intra-option policy that decides on the action given the current state
  - Can efficiently combine discrete variables (different tasks/contexts) and continuous variables (actions)
- Early reference: Doina Precup, "Temporal abstraction in reinforcement learning," University of Massachusetts Amherst, 2000.

- Assume the dynamics $p(s_{t+1} \mid s_t, a_t)$ are the same in source and target domain but reward function is different
- Examples:
  - Autonomous car learns to drive in one city and then needs to drive in a new one
  - Robot learned various recipes and now needs to cook a new one
- What should we transfer?
  - **Model:** should be independent of reward, so simple to transfer
  - **Value function:** depends on dynamics and reward, so less straightforward
  - **Policy:** possible with contextual policies but otherwise tricky, policy contains *least* dynamics information

Meta learning

Core idea

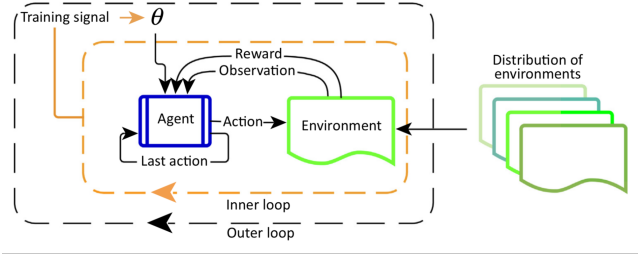


- Learning how to learn

→ End up with an RL agent that can learn more efficiently (fewer samples)

  - Smarter and more efficient exploration
  - Avoid actions we already learned are useless

Meta reinforcement learning problem setting



- Problem setting in regular RL: learn policy for a task $\pi^* = \arg\max_\pi \mathbb{E}_\pi[G_t]$
- Problem setting in meta RL: learn adaptation rule $\pi^* = \arg\max_\pi \sum_{i=1}^{n} \mathbb{E}_{\pi_i}[G_t]$ where i represents the MDP of task i
- Training and testing tasks are different, but from the same family of problems
- For instance, mazes with different layouts, running at different speeds, ...

How and what can we meta learn?

UPPSALA UNIVERSITET

- Key components:
  - **Model with memory:** e.g., recurrent neural network
  - **Meta learning algorithm:** how to update the weights, often ordinary gradient descent updates
  - **A proper distributions of MDPs** to learn from
- What can we meta learn?
  - Hyperparameters (e.g., the discount factor)
  - The loss function
  - The exploration strategy

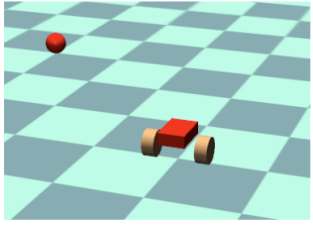
Meta-learning the exploration strategy[5]

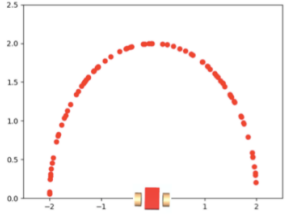
- Trading off exploration and exploitation in RL is difficult (see last lecture)
- Learn structured action noise from prior experience
- Policy $\pi(a \mid s, z_i)$, where $z_i \sim \mathcal{N}(\mu_i, \sigma_i)$

$\rightarrow$ Noise makes sure we explore new actions but we choose its distribution based on the current MDP

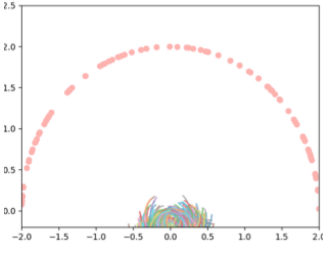
$\rightarrow$ How much should we explore in which task?

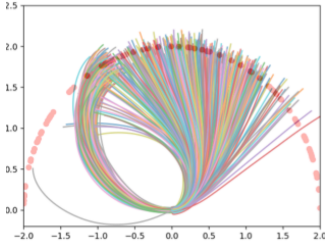
[5] Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine, "Meta-reinforcement learning of structured exploration strategies," Advances in Neural Information Processing Systems, 2018





- Wheeled robot should learn to reach different goal states
- Comparison: random exploration vs structured action noise





Causal reinforcement learning

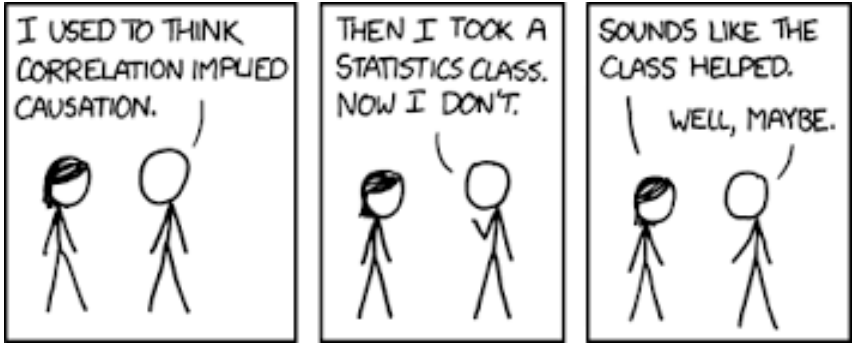
- From general statistics: correlation is not causation[6]
- However, if we had a causal model of the world, it would allow us to ask counterfactual questions and use them to reason about future actions
- What would have happened if I had chosen a certain action?

[6] John Aldrich, "Correlations genuine and spurious in Pearson and Yule," Statistical Science, 1995.

Story of King Midas

- King Midas was offered one free wish by the God Dionysos
- He chose that everything he touches would turn to gold
- That way he turned his whole family into gold
- In the end he died of starvation

$\rightarrow$ If we don't understand the consequences of our actions, we may maximize a short-term reward but deteriorate the long-term reward



- Causality in machine learning has been investigated for some time by a small community but is now gaining more and more interest
- Generally a hard problem, especially in MDPs
- Recently some research on causal reinforcement learning started
- Next slides: six tasks of causal reinforcement learning

Six tasks of causal reinforcement learning – Part I

UPPSALA UNIVERSITET

1. Generalized policy learning (combining online and offline learning)
   - Generally, for causal inference, we want to perform interventions
   - Often, cannot learn fully online (ethical constraints, expensive)

   $\rightarrow$ Try to get causal knowledge from offline data and leverage for necessary online learning
2. When and where to intervene? (Refine policy space)
   - When do we need interventions, when are there unintended side effects?
   - What should we change?

   $\rightarrow$ More informed and efficient exploration
3. Counterfactual decision-making (changing optimization function based on intentionality, free will, and autonomy)
   - Typically, agents act in a reflexive way without considering the causes for behaving in a particular way

   $\rightarrow$ Make agents aware of their intent through counterfactual reasoning

Example for bad policy learning

- Simple boat racing game: come out first and collect items on the way
- Agent found a sweet spot with many items that respawn
- Gets stuck there and never finishes the race

→ Not aware of its intent, no counterfactual reasoning

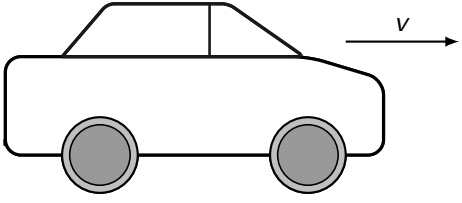
5. Generalizability & robustness of causal claims
   - Knowledge acquired is usually tied to specific environment

   $\rightarrow$ Allow agents to extrapolate
6. Learning causal model by combining observations and experiments
   - Construct fixed causal model from background data, offline information, etc.
   - Systematically combine observations and interventions to construct equivalence class of causal models

   $\rightarrow$ Increase efficiency
7. Causal imitation learning
   - Imitation learning often easier than pure RL
   - If learner and demonstrator have different causal models can have disastrous side effects

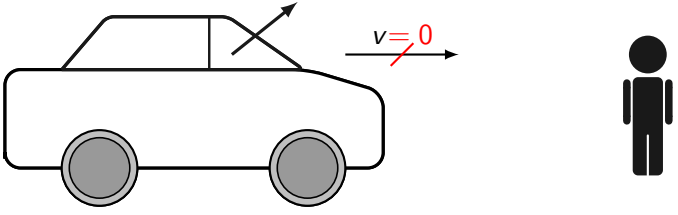
   $\rightarrow$ Understand conditions under which imitation learning is a good idea

A learning agent learns how to drive via imitation learning[7]



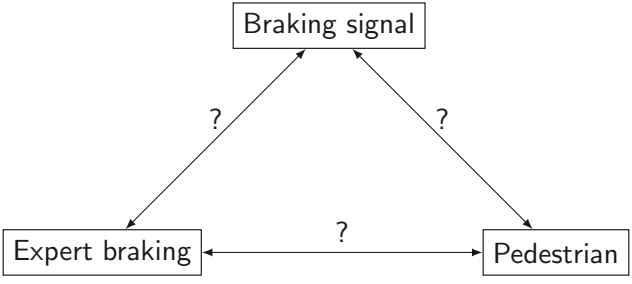
[7] Example from: Pim de Haan, and Dinesh Jayaraman, and Sergey Levine, "Causal confusion in imitation learning," Advances in Neural Information Processing Systems, 2019.

A learning agent learns how to drive via imitation learning[8]



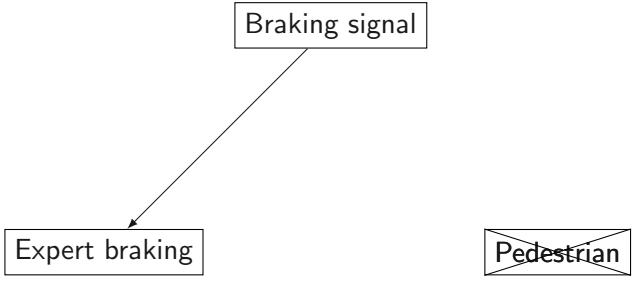
[8]Example from: Pim de Haan, and Dinesh Jayaraman, and Sergey Levine, "Causal confusion in imitation learning," Advances in Neural Information Processing Systems, 2019.

A learning agent learns how to drive via imitation learning[9]



[9]Example from: Pim de Haan, and Dinesh Jayaraman, and Sergey Levine, "Causal confusion in imitation learning," Advances in Neural Information Processing Systems, 2019.

A learning agent learns how to drive via imitation learning[10]



[10]Example from: Pim de Haan, and Dinesh Jayaraman, and Sergey Levine, "Causal confusion in imitation learning," Advances in Neural Information Processing Systems, 2019.

In summary

UPPSALA UNIVERSITET

- Deep RL has made amazing advancements during the last decade.
- But environments in literature are often "The hardest problem in the set of all simple problems"
- Algorithms are still not as stable and reliable as in supervised learning
- A lot of tricks for improving performance, still not completely clear which tricks are most important

Advertisement: interested in Master's thesis on RL?

UPPSALA UNIVERSITET

- We are offering Master's theses on RL topics
- For instance, causality, safe exploration, deep multi-agent reinforcement learning
- If you are interested, contact me: dominik.baumann@it.uu.se