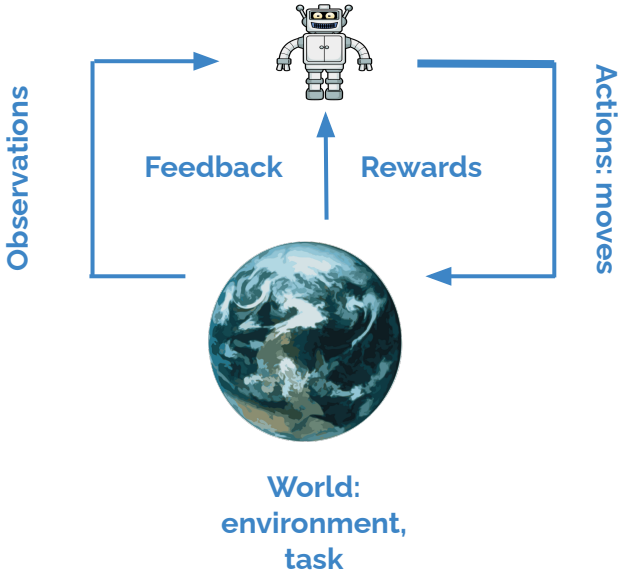
Understanding and Applying

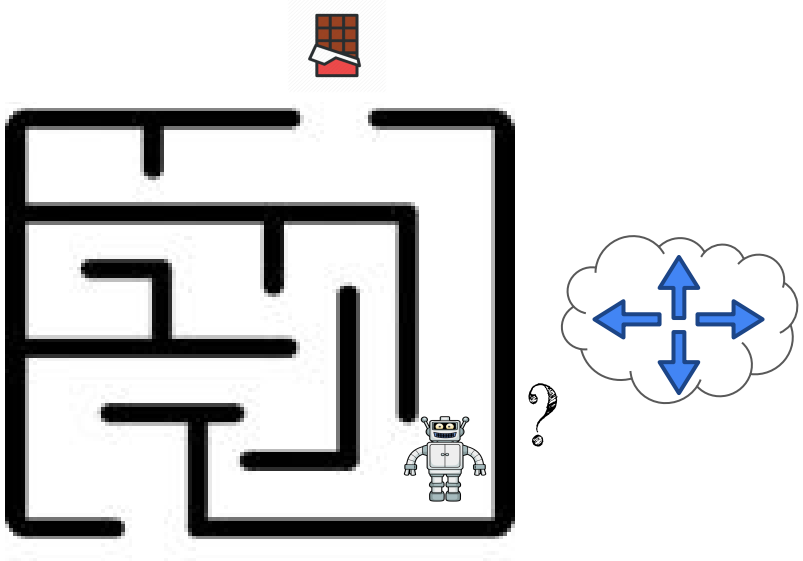
Reinforcement Learning

Dr. Claire Vernade

University of Tuebingen
Cluster of Excellence Machine Learning for Science

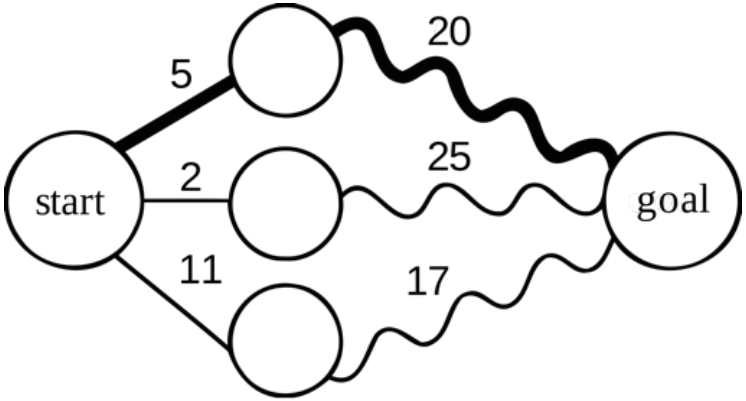
What is Reinforcement Learning?





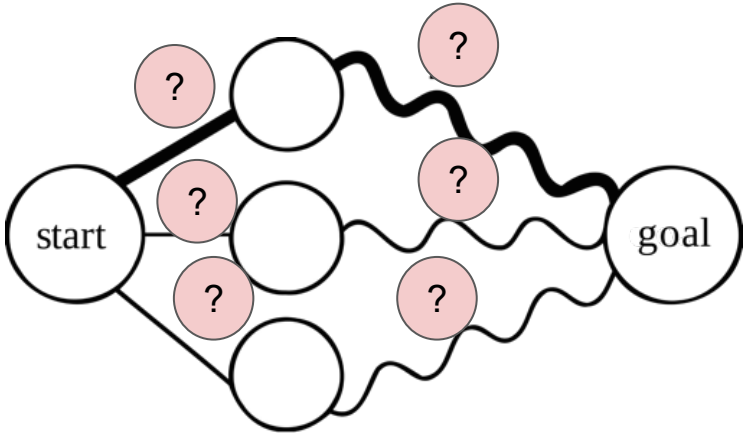
Planning + Exploration

Planning: Dynamic programming
ex: Shortest path problem



source: Wikipedia

RL: Need to **explore** in a noisy **environment**



A Short History of Reinforcement Learning

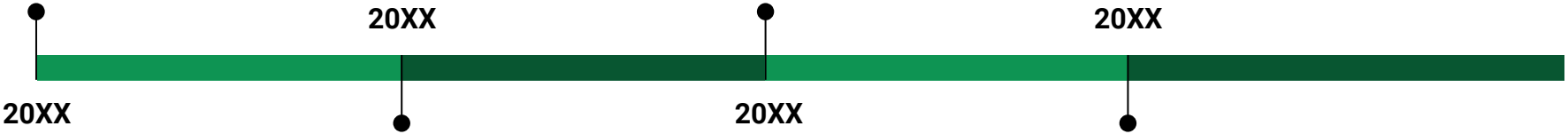
- Skinner's psychology studies on trial-and-error learning
- Computational models:
  - Optimal Control Theory (Bellman's dynamic programming)
  - Law-of-Effect (Thorndyke, 1911): combining Search (action selection) and Memory (remembering and associating actions)
- Temporal difference and Actor-Critic methods (Sutton, Barto 70s-80s)
- Q-Learning (Watkins 1989)
- Function approximation and Deep Q-Learning (2015-Now)

Vestibulum congue tempus

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor. Donec facilisis lacus eget mauris.

Vestibulum congue tempus

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor. Donec facilisis lacus eget mauris.



Vestibulum congue tempus

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor. Donec facilisis lacus eget mauris.

Vestibulum congue tempus

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor. Donec facilisis lacus eget mauris.

Reinforcement Learning in the Real World

Roadmap

1. A short introduction to Reinforcement Learning
    a. Markov Decision Processes
    b. Multi-Armed Bandits

2. Challenges and caveats in Reinforcement Learning

3. Lifelong Reinforcement Learning
    a. Meta-Learning with Bandits
    b. Planning in changing environments

Recent advances (since 2013)

- 2013: RL algorithms play Atari
- 2015: AlphaGO **plays Go at world-champion level**
- 2016: RL helps improve efficiency of cooling systems in data centers
- 2018-2020: Alpha Zero and MuZero (better and faster AlphaGo)
- 2020: Chip Design via RL
- ...

Markov Decision Processes (MDP)

A tuple: M=(S,A,P,R)

- S: a set of states
- A: a set of actions in each states
- P: transition probabilities for each state and action
- R: reward for each state and action

Learning problem: the model P and the reward function R are **unknown**!
The agent must choose actions and collect observations and estimate these key parameters

A **Policy** is a function $\pi : S \to A$ that chooses an action in each state (controller).

In state S_t, after taking action A_t, the learner observes $R(S_t, A_t)$ and $S_{t+1} \sim P(\cdot|S_t, A_t)$

Planning

In every state S, the **value** of policy π is the expectation of the **long-term** reward that can be obtained from this state when following π:

$$V^{\pi}(s) = \mathbb{E}_{a \sim \pi} \left[R(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} V^{\pi}(s') \right]$$

exp. immediate reward

exp. long-term future reward

discount factor: the future counts a bit less (geometrically) than the immediate reward. (Can be removed)

Example: RiverSwim

2 main issues:

- **Planning ahead** with an imperfect model
- **Exploration** to find the best policy

Planning: Dynamic Programming ideas

Introducing values to structure planning:

- Value function, Q-function
- Dynamic programming: Value iteration, policy iteration...
- Main caveat: assumes model is perfect, plans based on current state of knowledge, does not include exploration
- Planning with imperfect models: cite a few refs, mention [Khetarpal et al 2022]
-

Exploration: a hard problem

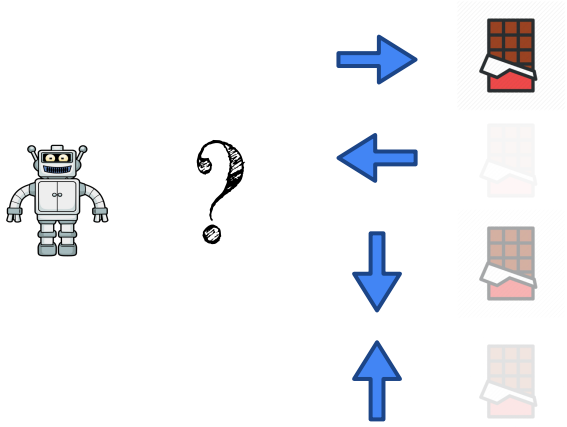
"Originally considered by Allied scientists in World War II, it proved so intractable that, according to Peter Whittle, the problem was proposed to be dropped over Germany so that German scientists could also waste their time on it."

--Wikipedia on Multi-Armed Bandit.

Peter Whittle, "Discussion on Dr. Gittins paper", Journal of the Royal Society of Statistics, 1979

Multi-Armed Bandits: Exploration without planning

At each round t, pick an action A_t



The performance of the agent is measured by its **regret**

$$R(T) = \sum_{t=1}^{T} \theta_{a^*} - \theta_{A_t}$$

Where $a^\star =$ 

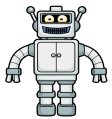
$$P(\text{reward}|\text{action } A_t = a) = \theta_{A_t}$$

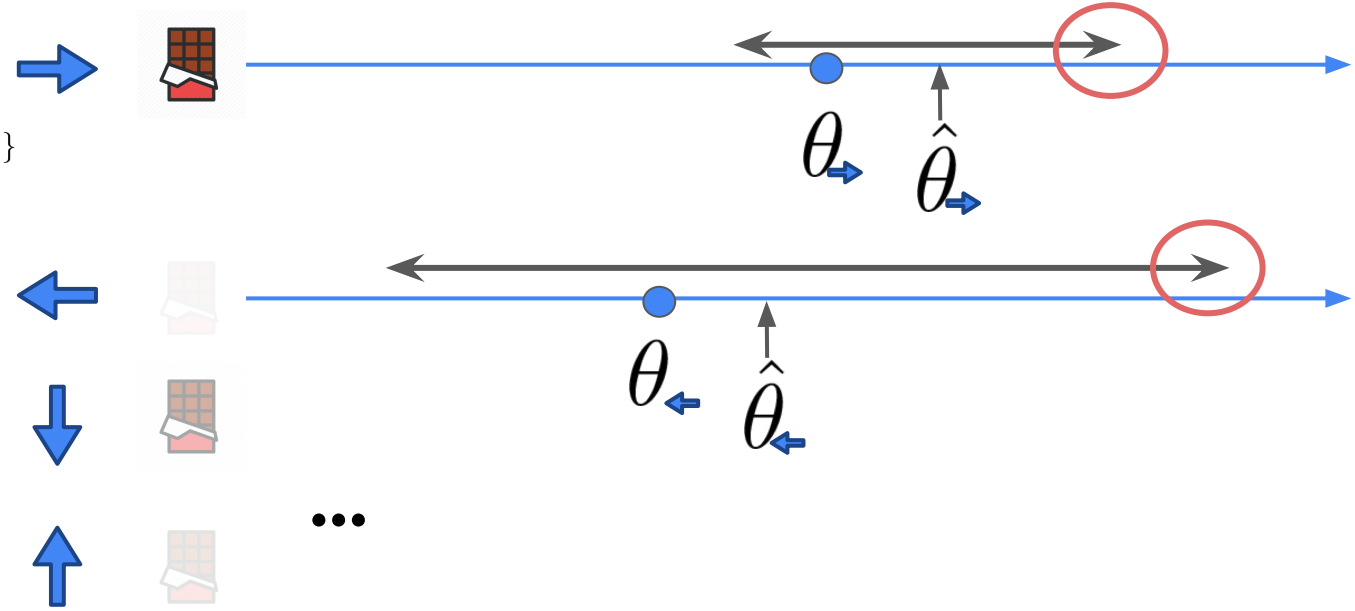
Methods and algorithms

Upper Confidence Bound:
Highest plausible value of the uncertain parameter

Bandit algorithms are based on **uncertainty estimation**, either using **high-probability confidence intervals.**

$\mathcal{H}_t = \{A_1, R_1, \ldots, A_{t-1}, R_{t-1}\}$





A crash-course in concentration inequalities

Let $Z, Z_1, Z_2, \ldots, Z_n$ be a sequence of independent and identically distributed random variables with mean $\mu \in \mathbb{R}$ and variance $\sigma^2 < \infty$

$$\text{empirical mean} = \hat{\mu}_n = \frac{1}{n}\sum_{t=1}^{n} Z_t$$

How close is $\hat{\mu}_n$ to μ?

Classical statistics says:

1. (law of large numbers) $\lim_{n\to\infty} \hat{\mu}_n = \mu$ almost surely
2. (central limit theorem) $\sqrt{n}(\hat{\mu}_n - \mu) \xrightarrow{d} \mathcal{N}(0, \sigma^2)$
3. (Chebyshev's inequality) $\mathbb{P}\left(|\hat{\mu}_n - \mu| \geq \varepsilon\right) \leq \frac{\sigma^2}{n\varepsilon^2}$

Wanted
Finite-time concentration inequality, stronger than Chebyshev's
=> Needs more assumptions

A crash-course in concentration inequalities

If a random variable Z is bounded or has bounded moments, then by (an extension of) Hoeffding's inequality, we can get fast concentration.
For example,

Random variable Z is σ-subgaussian if for all $\lambda \in \mathbb{R}$,

$$M_Z(\lambda) \doteq \mathbb{E}[\exp(\lambda Z)] \leq \exp\left(\lambda^2 \sigma^2 / 2\right)$$

Theorem If $Z_1, \ldots, Z_n$ are independent and σ-subgaussian, then

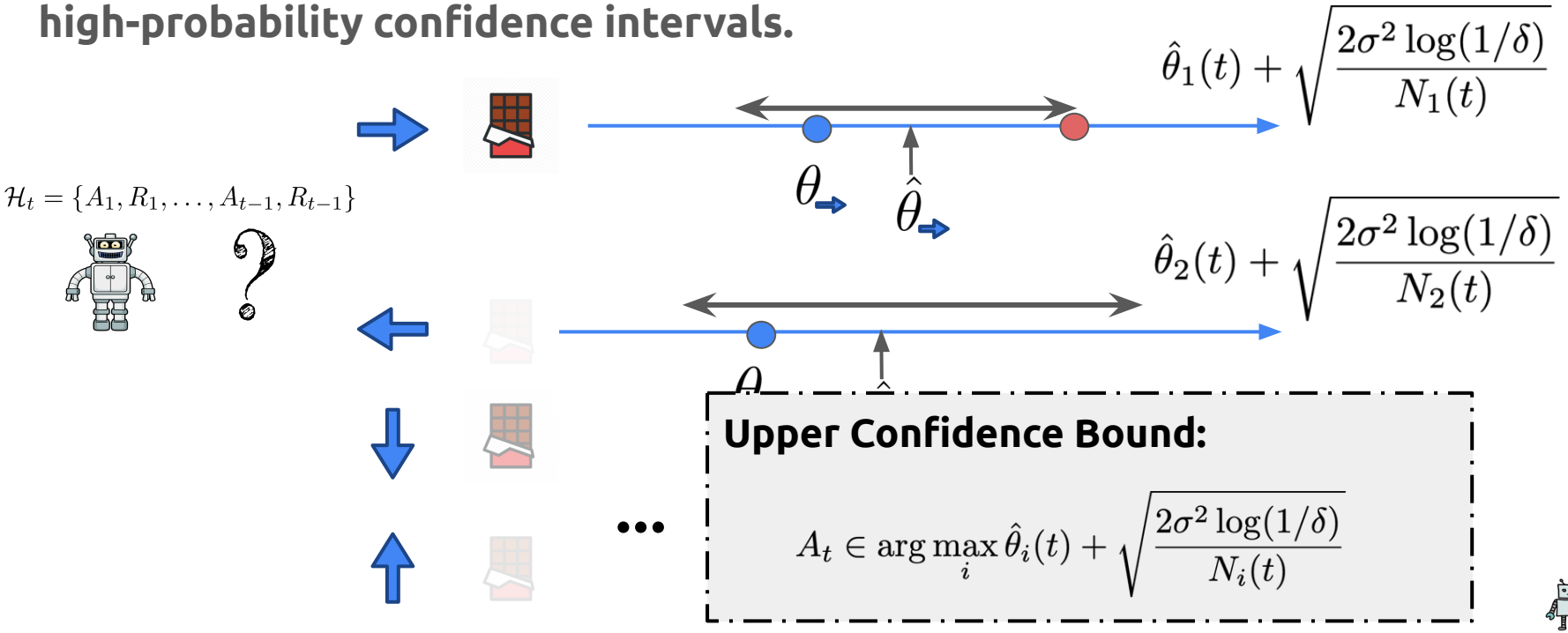
$$\mathbb{P}\left(\hat{\mu}_n \geq \sqrt{\frac{2\sigma^2 \log(1/\delta)}{n}}\right) \leq \delta$$

Chebyshev's: $\sqrt{\frac{\sigma^2}{\delta n}}$

Subgaussian: $\sqrt{\frac{2\sigma^2 \log(1/\delta)}{n}}$

Methods and algorithms: UCB

Bandit algorithms are based on **uncertainty estimation**, either using **high-probability confidence intervals.**



Upper Confidence Bound:

$$A_t \in \arg\max_i \hat{\theta}_i(t) + \sqrt{\frac{2\sigma^2 \log(1/\delta)}{N_i(t)}}$$

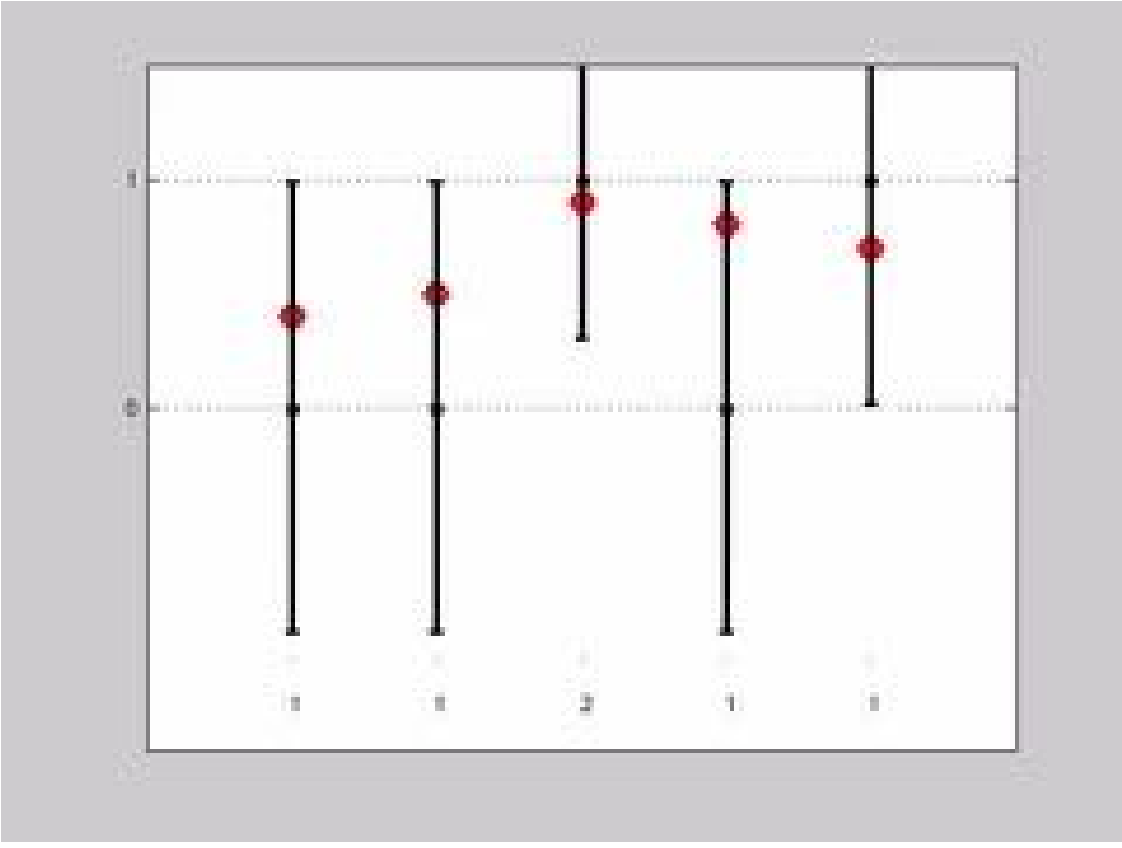
Theoretical guarantees: Regret bounds

Remember that the regret is defined by:

$$R(T) = \sum_{t=1}^{T} \theta_{a^*} - \theta_{A_t}$$

Theorem: On a K-armed bandit with 1-subGaussian rewards, the UCB algorithm applied with $\delta = 1/T^2$ has a T-step regret bounded by

$$R(T) \leq 3 \sum_{i=1}^{K} \Delta_i + \sum_{i:\Delta_i > 0} \frac{16 \log(n)}{\Delta_i}$$



More on Bandit Algorithms

- Recent and excellent **book** by Tor Lattimore and Csaba Szepesvari: *Bandits Algorithms. 2020. Cambridge University Press.* *www.banditalgs.com*

- **Contextual Bandit** models allow the learner to let the reward function depend on the environment

- Closely connected to
  - Online learning and online optimization
  - Portfolio optimization
  - Game Theory
  - Operations Research

RL algorithms: Planning + Exploration / Exploitation

- UCRL
- PSRL
- Policy Gradient
- TRPO

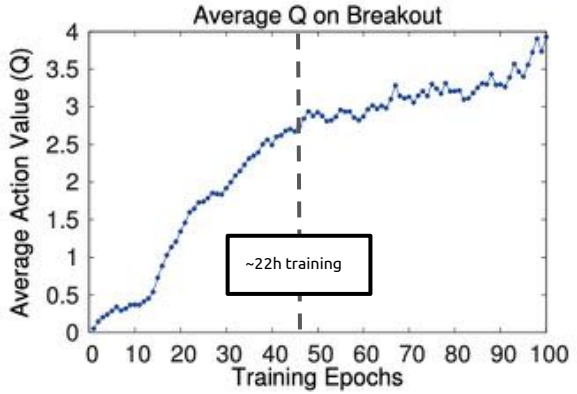
Reinforcement Learning is a **planning** problem under **uncertainty**. It faces a trade-off between **exploration** and exploitation at all levels of **decision-making**.

In practice

Learning by trial-and-error is costly:

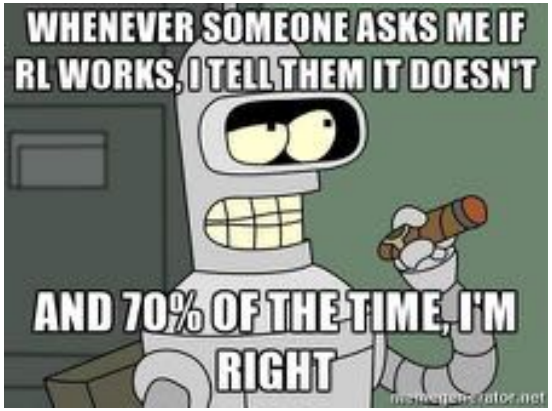
- Efficient exploration in RL is still an open problem (in general)
- Need to explore high-dimensional state spaces
- “Sample complexity” is infamously bad in RL

RL agents do not (yet) efficiently and systematically transfer knowledge across tasks.



“**One epoch** corresponds to 50000 minibatch weight updates or roughly **30 minutes of training time**”

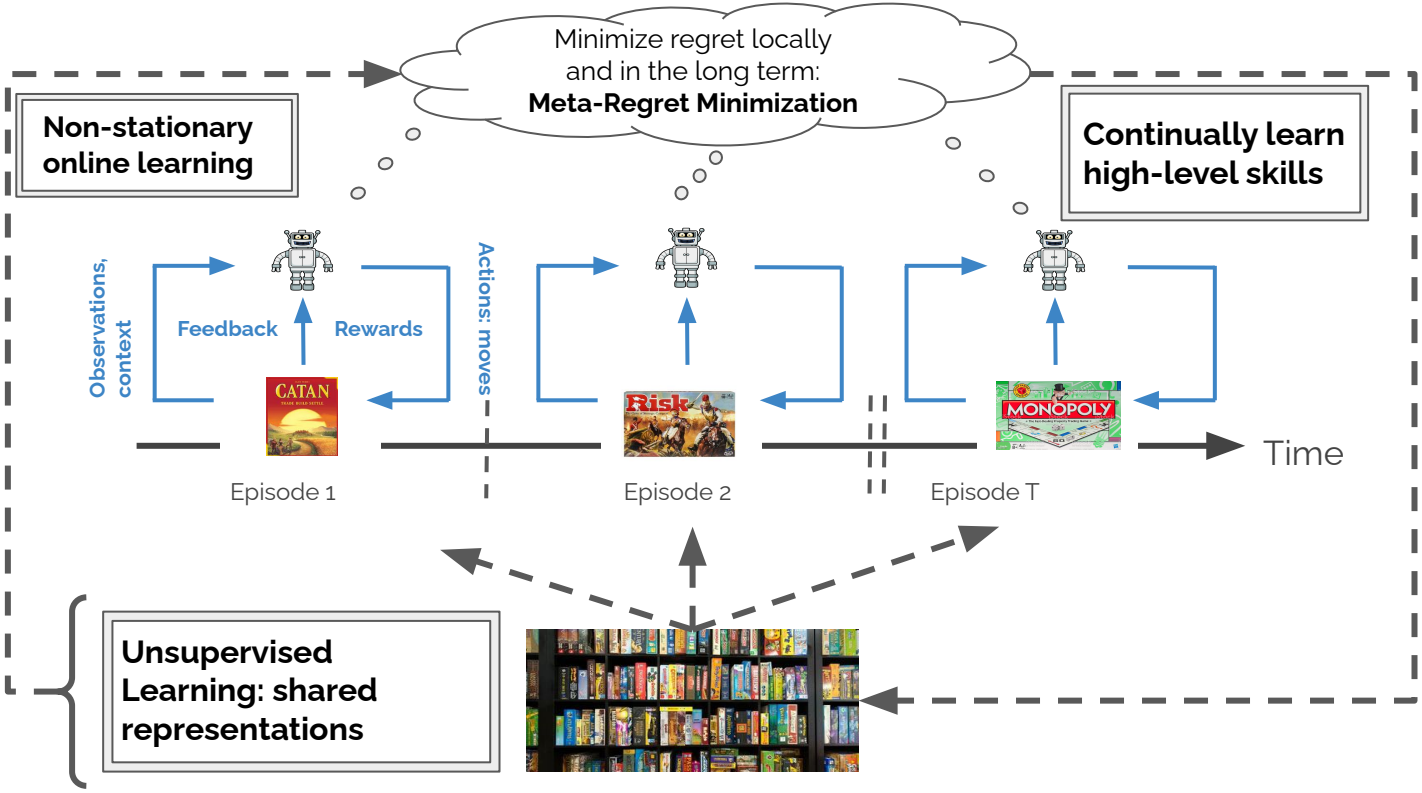
Brief diagnosis



Alex Irpan'blog:
https://www.alexirpan.com/2018/02/14/rl-hard.html

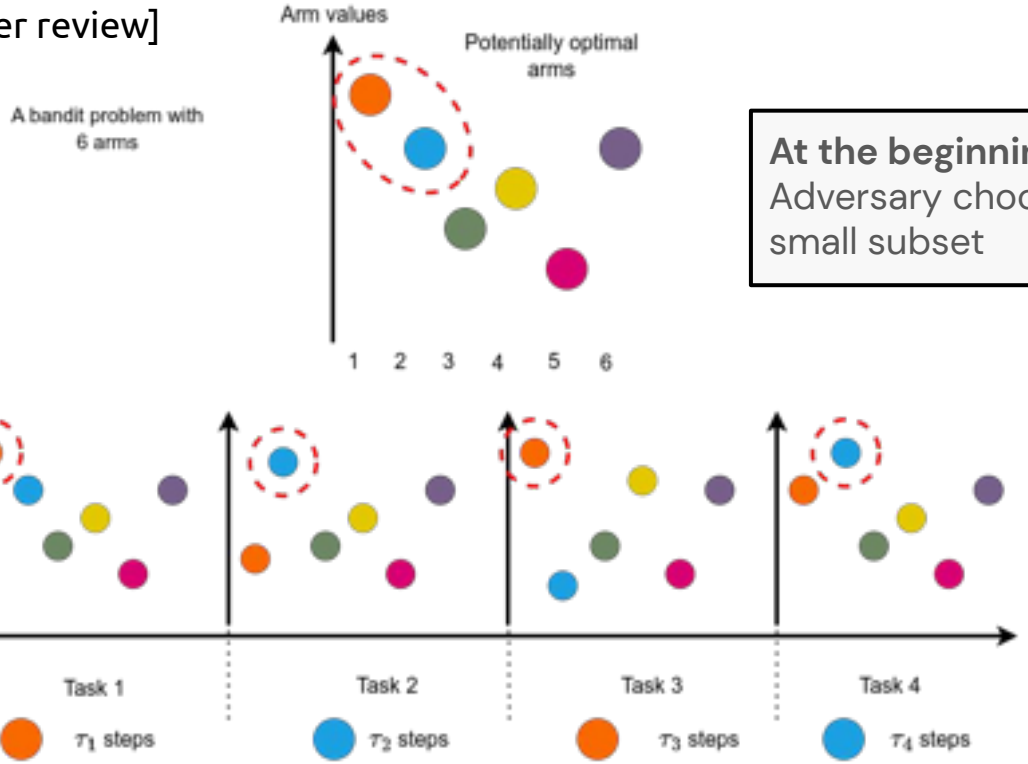
- RL agents learn **"from scratch"** -> Is it always necessary?
- RL agents do not use high-level models of the world
  - Can we learn better priors?
- Performance of RL agents are unstable.
  - Random seeds have a huge impact on performance
  - Can we reduce the variance of the value estimation process?
- Other unresolved questions:
  - Where does the reward function come from? Is it well designed?

Towards Lifelong Reinforcement Learning



Example 1: Meta-Learning for K-armed bandits with a subset of optimal arms

[Azizi et al. 2022, under review]



At the beginning of each task, Adversary chooses best arm in a small subset

Setting

- We consider N sequential tasks, each is K-armed stochastic bandit, with total length $T = \sum_{n=1}^{N} \tau_n$ (τ_n: task length)
- At the beginning of task $n \in [N]$, the adversary chooses the mean reward function $r_n \in \mathcal{R} = [0,1]^K$.
  - *Reward noise* is independent zero-mean and $[-1/2, 1/2]$-valued noise
- Adversary is restricted to choose the optimal arms in a smaller (unknown) subset of $M \leq K$ arms

Meta-Regret Definition

Regret relative to an action sequence, $(a_n)_{n=1}^N$:

$$R(T, (a_n)_{n=1}^N) \doteq \mathbb{E}\left[\sum_{n=1}^{N}\sum_{t=1}^{\tau_n} r_n(a_n) - \sum_{n=1}^{N}\sum_{t=1}^{\tau_n} r_n(A_{n,t})\right],$$

task length

action

learner's action

reward

Regret relative to the best set of M arms:

$$R_T \doteq \max_{\substack{(a_n)_{n=1}^N \\ |\{a_n\}_{n=1}^N| \leq M}} R(T, (a_n)_{n=1}^N)$$

subset of at most M arms

Meta-Algorithm ideas

- The meta-learner picks a M-subset S_n
- Runs a **“reasonable” baseline bandit algorithm on the M chosen arms**.
  Its average regret is:

=f_n(S_n) : submodular function

$$\epsilon_n = \frac{1}{\tau_n}\left(\sum_{t=1}^{\tau_n} \max_{a\in S_n} r_n(a) - r_n(A_{n,t})\right) = O(\sqrt{|S_n|/\tau_n})$$

- The (meta-)regret over all tasks is bounded by

$$R_T \leq \sup_{f_1,\ldots,f_{N'}\in\mathcal{F}} \max_{S\in\mathcal{S}} \mathbb{E}\left[\sum_{n=1}^{N'}\left(f_n(S) - f_n(S_n) + B_{\tau',N_n,M}\right)\right]$$

  - If the optimal arm is in S, the regret is small
  - Otherwise the regret is large
  - New **meta exploration problem**

Main approach: bottom-up set discovery

Bottom-up: **Identify best arms** as they appear -> **(Greedy)-BASS**

- Requires an **Identifiability condition**: a best-arm identification algorithm should be able to find the best arm for each task with probability 1-\delta/N
- Meta-Explore: use BAI on all K arms to try to identify a new best arm
- Meta-Exploit: Construct an M-subset with the identified arms so far (and more)

Regret upper bound

Better than $O(N\sqrt{K\tau})$

Theorem

The regret of G-BASS under the Identifiability Condition with $\gamma_n = \sqrt{\frac{|S_n|K\tau}{NB_{\tau,K}}}$ *for all* n *is bounded as*

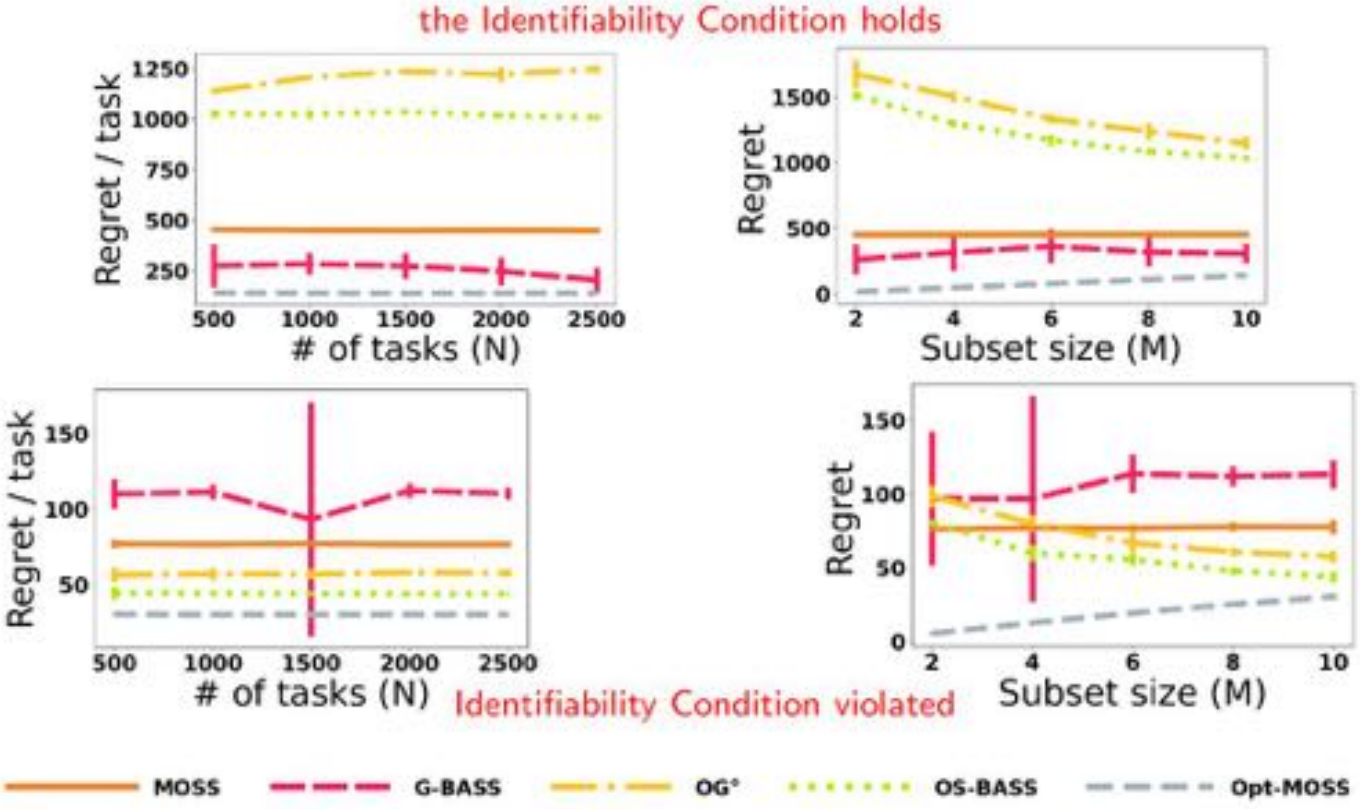
$$\tilde{O}(N\sqrt{M\tau} + N^{1/2}\sqrt{MK\tau})$$

Optimal rate when the subset is known

Cost to discover it

Take-home message: The meta-learner can learn a structure over the environment itself to reduce the complexity of the problem on each individual task.

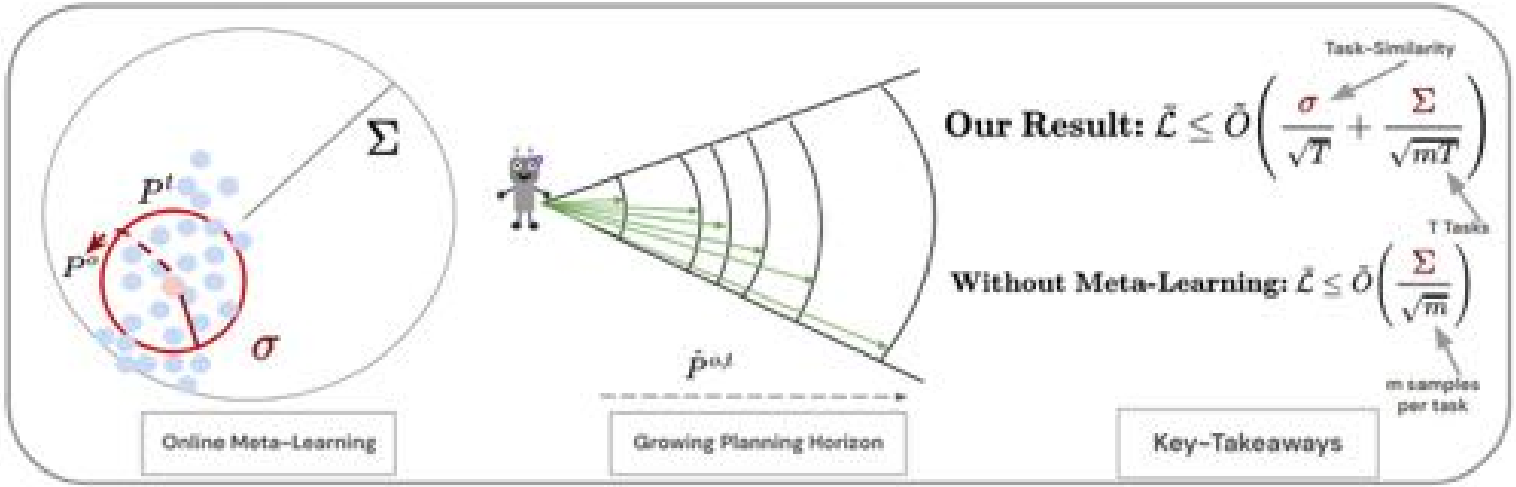
We actually tried



Default setting $N = 500, \tau = 4500, K = 30, M = 10.$

Example 2: Meta-Learning for Planning

[Khetarpal et al. 2022, under review]



Can we meta-learn the model across tasks and progressively learn to plan on longer horizon?

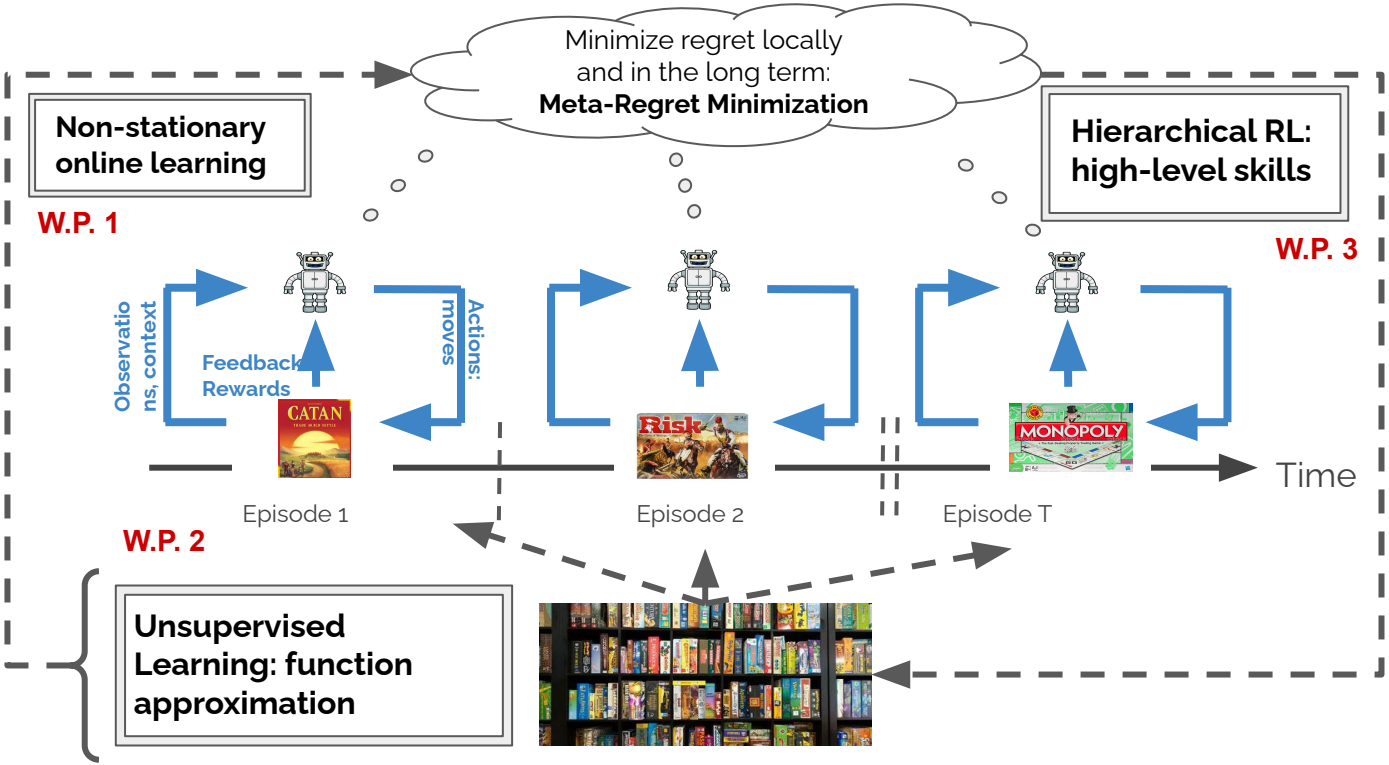
Foundations for Lifelong RL Group

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Non-stationarity is a fundamental element of Reinforcement Learning
- But **generalization** and **sample complexity** matter!
- Lifelong RL requires to assume models and structures but hopefully helps.

Open problems:

- Sample-efficient model-based algorithm
- Lower-Bounds for Meta-Bandits (and Meta-RL)
- High-level planning
- ...



EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Quick proof of the concentration theorem

Theorem If $Z_1, \ldots, Z_n$ are independent and σ-subgaussian, then

$$\mathbb{P}\left(\hat{\mu}_n \geq \sqrt{\frac{2\sigma^2 \log(1/\delta)}{n}}\right) \leq \delta$$

Proof We use **Chernoff's method**. Let $\varepsilon > 0$ and $\lambda = \varepsilon n / \sigma^2$.

$$\begin{aligned}
\mathbb{P}\left(\hat{\mu}_n \geq \varepsilon\right) &= \mathbb{P}\left(\exp\left(\lambda \hat{\mu}_n\right) \geq \exp\left(\lambda \varepsilon\right)\right) \\
&\leq \mathbb{E}\left[\exp\left(\lambda \hat{\mu}_n\right)\right] \exp(-\lambda \varepsilon) && \text{(Markov's)} \\
&\leq \exp\left(\sigma^2 \lambda^2 / (2n) - \lambda \varepsilon\right) \\
&= \exp\left(-n\varepsilon^2 / (2\sigma^2)\right)
\end{aligned}$$

Methods and algorithms

Bandit algorithms are based on **uncertainty estimation**, either using **high-probability confidence intervals** or **Bayesian posterior distributions.**