UPPSALA
UNIVERSITET

Reinforcement Learning

Lecture 10 – Exploration and exploitation

Dominik Baumann

2022

Department of Information Technology

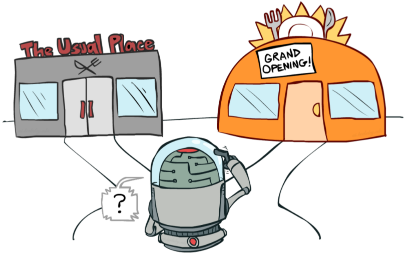
So far, we have covered the basics of RL:

- MDPs, Bellman equations, and dynamic programming
- Tabular RL methods such as SARSA and Q-learning
- Function approximation and policy gradient methods

The last three lectures:

- Exploration vs exploitation (this lecture)
- Recent advances in RL, and open problems (next lecture)
- Summary of the course

Exploration vs exploitation



- Reinforcement learning: we want to optimize a reward function by trial and error
- Example: we want to go out for dinner (high reward means good food)
    - **Exploitation:** go to my favorite restaurant
    - **Exploration:** try out a new restaurant
- Today's lecture:
    - What exploration strategies exist?
    - How can we quantify how *good* they are?
    - For which types of RL problems is exploration particularly challenging?

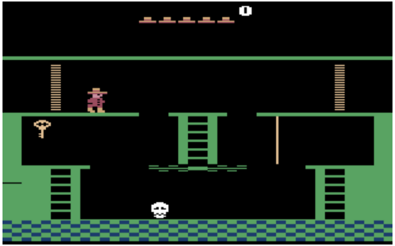
When is exploration challenging?

Two examples

Breakout

"Easy"

Montezuma's Revenge



Very hard!

- Try to destroy blocks with the ball
- Receive rewards when hitting blocks
- Over when all blocks are destroyed

- Try to escape the room
- Rewards for finding key and opening door
- Dead (but no negative reward) when hitting skull

Why is Breakout so easy?

UPPSALA UNIVERSITET

- **Reward structure:** we get a positive reward whenever we hit something

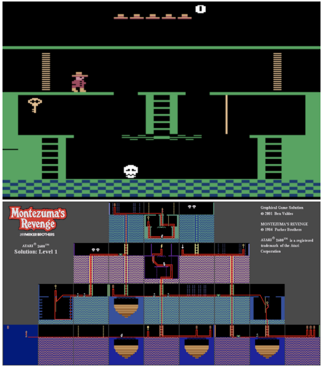
→ By trying out random action, we will sometimes hit blocks and learn that this is good

- Game finishes when all blocks are destroyed

→ Then we have also achieved a high reward

Why is Montezuma's revenge so hard?

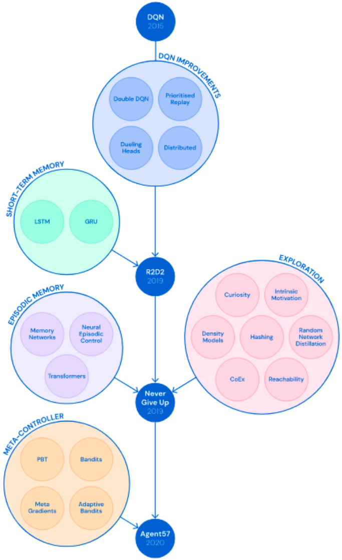
UPPSALA UNIVERSITET



- **Reward structure:** getting key = reward. Open door = reward. Killed by skull = nothing (good? bad?)
- The required action sequence for getting the first reward is relatively complex
- Finishing the game only weakly correlates with rewards
- A human can directly understand concept of key
- RL agent needs to learn them from trial and error

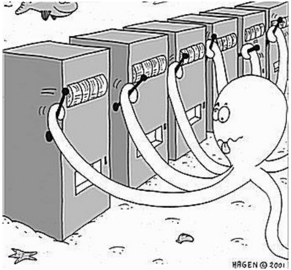
- In 2018, DeepMind and OpenAI published papers "solving" this game. Either by imitating a human or by restarting the game in different states.
- 2019: Agent57 by Deepmind manges to learn all 57 Atari games in Atari57 benchmark

UPPSALA UNIVERSITET

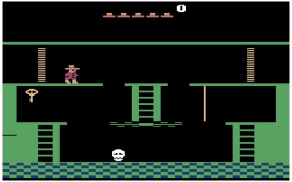


- Uses a lot of tricks to make it work for all games!
- Still used around 10^{11} frames (about 106 years of gaming)
- Massive computational resources
- Is it solved?

When is exploration hard?



- **Multi-armed bandit:** select a bandit, pull the arm, immediately get a reward
- Can formalize the exploration problem, analytically tractable



- **Learning in large MDPs**: large state space (pixels), temporally extended reward structure
- Analytically intractable

What is in between learning from pixels and bandits?

Multi-armed bandits | Contextual bandits | Small MDPs | Large MDPs

Analytically tractable ⟷ Analytically intractable

What exploration principles exist?

- We can analytically "solve" the multi-armed bandits exploration problem but computationally expensive
- Simpler strategies exist with which we can already do very well
- ε-**greedy:**
  - Choose random action with probability $1 - \varepsilon$
- **Optimistic initialization:**
  - Assume the best until proven otherwise
- **Optimism in the face of uncertainty:**
  - Prefer actions we are uncertain about
- **Probability matching:**
  - Select best action according to probabilistic belief
- We will discuss to what extent these strategies work for the 4 RL problem classes
- But before, we need a measure for when an exploration strategy is *good*

How to quantify exploration strategies?

Regret

UPPSALA UNIVERSITET

- Assume we are in the bandit setting (i.e., no state)
- **Action-value:**

$$q(a) = \mathbb{E}[R_t \mid A_t = a]$$

- **Optimal value:**

$$V^* = q(a^*) = \max_a q(a)$$

- **Regret:** the opportunity loss for one step

$$\ell_t = \mathbb{E}[V^* - q(a_t)]$$

- **Total regret:**

$$L_t = \mathbb{E}\left[\sum_{\tau=1}^{t} (V^* - q(a_\tau))\right] = \sum_{\tau=1}^{t} \ell_t$$

- Maximize cumulative reward $\iff$ Minimize total regret

Counting regret

- Assume we have chosen a strategy for exploration
- **Counts:** Let $N_t(a)$ be the number of times we selected action a after t steps
- **The gap:** $\Delta_a = V^* - q(a)$
- The regret can then be written as

$$\begin{aligned} L_t &= \mathbb{E}\left[\sum_{\tau=1}^{t} V^* - Q(a_\tau)\right] \\ &= \sum_a \mathbb{E}[N_t(a)](V^* - Q(a)) \\ &= \sum_a \mathbb{E}[N_t(a)]\Delta_a \end{aligned}$$

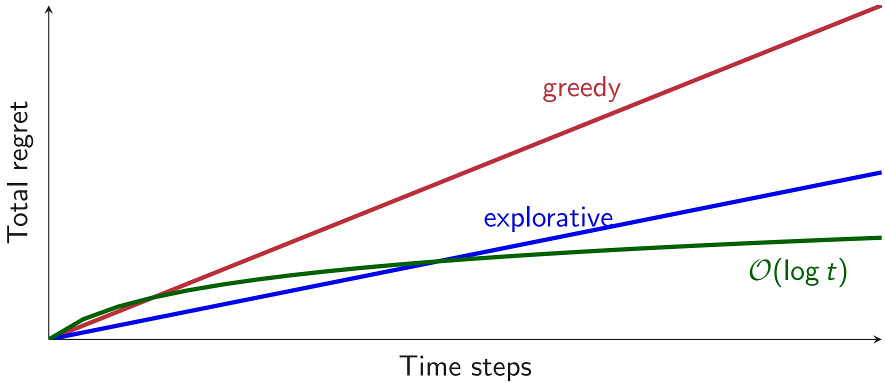
- We thus want small counts for large gaps!
- **Problem:** we do not know the gaps!

What happens to the regret if we never explore?

- Let $Q(a)$ be an Monte Carlo estimate of $q(a)$
- Greedy action selection
$$a_t = \arg\max_a Q(a)$$
- Can lock onto a suboptimal action forever $\Longrightarrow$ Linear total regret

- **Example: Two doors**
  - $q(\text{left}) = 5$, $q(\text{right}) = 10$
  - Initial: $Q(\text{left}) = Q(\text{right}) = 0$
  - Start taking left and get positive reward
  - Continue to use left greedily
  - Gap $\Delta_{a_t} = 5$ for all t since we never try right
  - So total regret will grow linearly

Can we do better than linear regret?

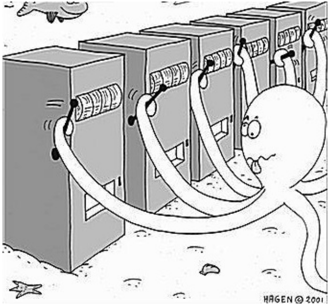


- **No exploration:** gives linear total regret
- **Always explore:** also gives linear total regret!
- Can we get sublinear total regret? Can be shown that $\mathcal{O}(\log t)$ is optimal.

Multi-armed bandits

UPPSALA UNIVERSITET

- A 1-step stateless RL problem
- We have m actions (m bandits to choose between)
- The reward is given by unknown probabilities $p(r \mid a)$
- At each time step t the agent choose action a_t and gets reward r_t
- **Goal:** Maximize the cumulative reward $\sum_{\tau=1}^{t} r_\tau$
- **Action-value:** $q(a) = \mathbb{E}[R_t \mid A_t = a]$
- **Optimal action:** $a^* = \arg\max_a q(a)$
- **With MC:** Estimate $Q(a)$ as the average reward seen from action a so far



ε-greedy algorithm

- Let $Q(a)$ be a Monte Carlo estimate of $q(a)$
- ε-greedy action selection:
  - With probability $1 - \varepsilon$ take $\arg\max_a Q(a)$
  - With probability ε take random action
- Now $Q(a) \to q(a)$. So in the limit the greedy action will be optimal (have gap 0).
- **But** we do not use the greedy policy, we use ε-greedy
- The probability of each action is it at least ε/m every time step
- Hence, the regret for each step satisfies

$$\ell_t \geq \frac{\varepsilon}{m} \sum_a \Delta_a$$

- and the total regret

$$L_t \geq t \frac{\varepsilon}{m} \sum_a \Delta_a$$

  grows linearly

Decaying ε_t-greedy

- The total regret grows linearly because we have $\varepsilon > 0$ at all times
- What if we set $\varepsilon = 0$ once we converged?
- Problem: We don't know when we converged...
- Instead, use ε_t-greedy algorithm, and let ε_t decay over time
- **Theoretical result:** it is possible to construct schedule ε_t that asymptotically gives *logarithmic* total regret
- The theoretical schedule requires that we know all gaps...
- But can still try to tune a good schedule in practice

Optimistic initialization

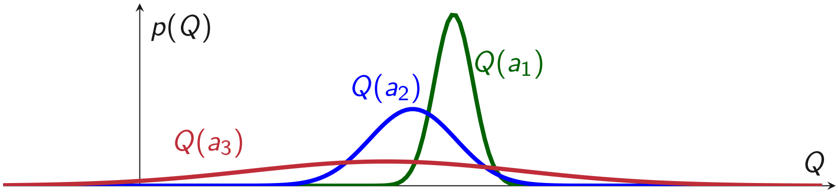
- Initialize $Q(a)$ for each action to a large value!

$\rightarrow$ Now actions that we have not tried before look good

- Encourages exploration
- We have implicitly used this in Notebooks. E.g. in MountainCar where we initialize $Q(s, a) = 0 > r_{\text{max}}$.
- **But** risk that exploration stops before we find optimal action, and thus lock on to suboptimal action…
- …and gives linear total regret.

Optimism in the face of uncertainty

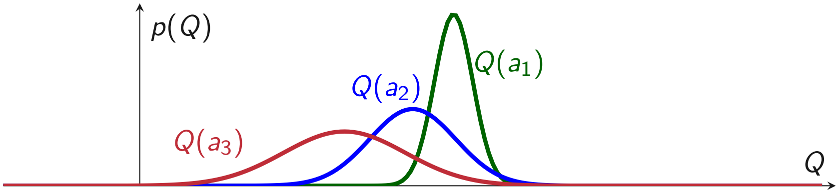
- What if we also measure uncertainty in $Q(a)$?



- Which action should we pick?
- a_1 looks best of we only look at mean value
- But we are very uncertain about a_3. It could very well be better than a_1!
- If we try a_3 again we will get more certain about the value

Optimism in the face of uncertainty

- What if we also measure uncertainty in $Q(a)$?



- Which action should we pick?
- a_1 looks best of we only look at mean value
- But we are very uncertain about a_3. It could very well be better than a_1!
- If we try a_3 again we will get more certain about the value

Upper confidence bounds

- **Idea:** estimate an upper confidence bound $U(a)$ for each action value (hence the algorithm is called UCB – upper confidence bound)
- I.e, such that $q(a) \leq Q(a) + U(a)$ with high probability
- Note: $U(a)$ should decrease when $N(a)$ increases
- **Action selection:**

$$a = \arg\max_{\tilde{a}} Q(\tilde{a}) + U(\tilde{a})$$

- A statistically motivated choice of $U(a_t)$ is

$$U_t(a) = c\sqrt{\frac{\ln t}{N_t(a)}}$$

- Asymptotically logarithmic regret!

- In the methods we have looked at so far we make no assumptions on the distribution of rewards

A Bayesian approach:

- **Likelihood:** assume that given an action a the reward is distributed as

$$p(r \mid a, \boldsymbol{\theta})$$

  where $\boldsymbol{\theta}$ are unknown parameters
- **Example:** p is Gaussian, and $\boldsymbol{\theta}$ contains the mean and variance of each action
- **Prior:** $p(\boldsymbol{\theta})$
- **Observations:** $\mathcal{D}_t = \{(a_1, r_1), \ldots, (a_t, r_t)\}$
- **Posterior:** using Bayes rule

$$p(\boldsymbol{\theta} \mid \mathcal{D}_t) \propto \prod_{\tau=1}^{t} p(r_\tau \mid a_\tau, \boldsymbol{\theta}) p(\boldsymbol{\theta})$$

UPPSALA UNIVERSITET

- Sample from the posterior: $\boldsymbol{\theta}_t^* \sim p(\boldsymbol{\theta} \mid \mathcal{D}_t)$
- Choose next action greedily w.r.t sampled $\boldsymbol{\theta}_t^*$:

$$a_{t+1} = \arg\max_a \mathbb{E}[r \mid a, \boldsymbol{\theta}_t^*]$$

How does this give exploration?

- Take the Gaussian p as an example
- If, given $\mathcal{D}_t$, we are still very uncertain of the mean value μ_a for action a, there is a high probability that we sample $\boldsymbol{\theta}^*$ with large μ_a
- Can be shown to achieve optimal regret in certain cases, and works well in practice (if the assumed likelihood and prior are good)

Summary

- ε-greedy
  - With decay schedule, sublinear regret possible
  - But for optimal decay schedule, need to know the gaps
- Optimistic initialization
  - Linear regret
  - Exploration may stop too early
- Optimism in the face of uncertainty, UCB
  - Asymptotically achieves logarithmic regret
  - Need to find an upper bound $U(a)$ such that $q(a) \leq Q(a) + U(a)$ holds (with high probability) for all a and t
- Probability matching, Thompson sampling
  - Can achieve optimal regret
  - Only works well if assumed likelihood and prior are good

Contextual bandits

- A 1-step RL problem with state
- A set of actions $\mathcal{A}$, and a set of states (context) $\mathcal{S}$
- In each step s_t is drawn from $\mathcal{S}$ and the agent takes action a_t
- A reward $r_t \sim p(r \mid s_t, a_t)$ is given
- **Goal:** maximize $\sum_{\tau=1}^{t} r_\tau$

Example application: recommendation systems and online advertisement

- State s contains information about user
- Action a can be what ad to show/article to recommend
- Reward r can be based on whether or not the user clicks on ad/article

- Ideas from multi-armed bandits relatively straightforward carry over to this setting

Linear Upper Confidence Bounds

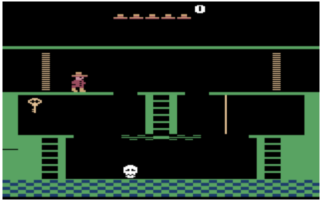
- If we use a linear function approximator

$$Q(s, a) = \varphi(s, a)^\top \boldsymbol{\theta}$$

  we can also compute the variance $\sigma_\theta^2(s, a)$ of the action-values due to uncertainty about $\boldsymbol{\theta}$
- Natural to use $U(s, a) = c\sigma_\theta(s, a)$ (c standard deviations above the mean)

MDPs

Exploration in MDPs



- A multi-step RL problem with state
- A set of actions $\mathcal{A}$ and a set of states $\mathcal{S}$
- In each step s_t we choose an action a_t and advance to the next state s_{t+1}
- We receive a reward r_t, which may be 0 until we reach the goal

$\rightarrow$ Temporally extended states and rewards make the exploration problem harder

- However, exploration strategies can be based on the same ideas, even though analytically often intractable

ε-greedy

- Easy to implement and often used for exploration in MDPs
- Plain ε-greedy has still linear regret
- More challenging to come up with good decay schedule
- **In practice** decaying ε_t is often used even in Deep RL. It is however common to let $\varepsilon_t \to c > 0$, to never stop exploring completely

Upper Confidence Bounds

- Use
$$a_t = \arg\max_a Q(s_t, a) + U(s_t, a)$$
where U is some confidence bound
- With tabular or linear approximation, uncertainties can be handled analytically

- For multi-armed bandits we used $U(a_t) = c\sqrt{\ln t / N(a)}$
- Lots of functions work, as long as they decay with $N(a)$

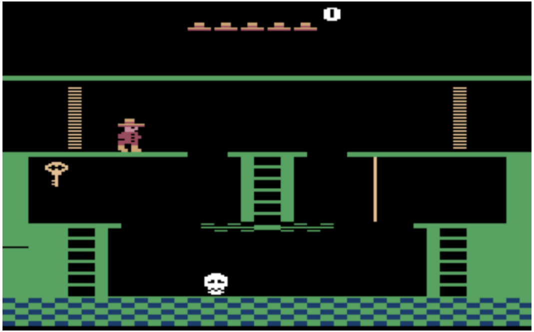
Count-based bonuses

- Give an exploration bonus to rewards:

$$r^+(s,a) = r(s,a) + \mathcal{B}(N(s))$$

- Simple, but requires tuning of bonus weight Use, e.g., $\mathcal{B}(N(s)) \propto \frac{1}{\sqrt{N(s)}}$
- This can be fine for small discrete MDPs (e.g., grid worlds)

But what is a count in a large MDP?



- State: 4 stacked frames
- Very unlikely to see same state several times! (Exactly the same pixels)
- *Bellemar et.al, 2016:* fit a density model $p_\theta(s)$ to data, and use this to find a pseudo-count. Then $p_\theta(s)$ may be large for states we have not seen before, but that are similar to states that we have seen.

Model-based:

- Estimate a parametrized model $p_\theta(s', r \mid s, a)$
- Use a prior and Bayes law to find a posterior distribution of $\boldsymbol{\theta}$
- Sample one θ^* from the posterior, and use planning to find a policy
- Use this model policy for one episode, update posterior, and sample new model

Directly on the Q-function: (Osband et al., 2016)

- Train N different Q-functions using, e.g., DQN
- For each episode, sample one Q-function and act greedily with respect to this
- Update your Q-functions with new experience

Why would this work?

- With ε-greedy policy we may just oscillate back and forth
- Being greedy with respect to random Q may have higher chance to go to interesting places

- We want our policy to optimize rewards, but also be as random as possible to explore
- **Idea:** encode this trade-off directly into the optimization criterion
- Let $\mathcal{H}(\pi)$ be the entropy (randomness) of the policy, and instead optimize

$$J(\pi) = \mathbb{E}\left[\sum_t r_t + \alpha\mathcal{H}(\pi)\right]$$

- Example: entropy search (Hennig and Schuler, 2012), explore actions that maximize information about the global optimum

Entropy-based methods in practice

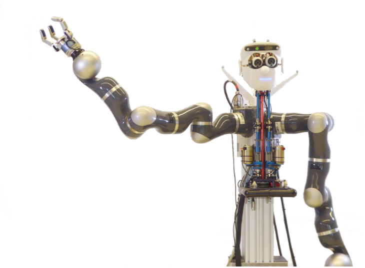
Video available at https://youtu.be/TrGc4qp3pDM, taken from Alonso Marco, Philipp Hennig, Jeannette Bohg, Stefaan Schaal, and Sebastian Trimpe, "Automatic LQR tuning based on Gaussian process global optimization," IEEE International Conference on Robotics and Automation, 2016.

Summary

- Exploration vs exploitation is an important part of RL
- ε-greedy is a powerful idea
- Many approaches based on
    - Optimism in the face of uncertainty
    - Probability matching / Thompson sampling
- Relatively straightforward to carry over ideas from Multi-armed bandit to small MDPs
- Many recent ideas about how to tackle exploration in large MDPs

Outlook: safe exploration

Learning in games vs learning in the real world



- If we choose a random action in Breakout, the worst that can happen is that we miss the ball
- Can simply restart the game, no critical damage
- What if we choose a random action on a robot?
- We might break the hardware. . .

→ For learning on real world systems, need to consider whether actions are safe

Second example for unsafe exploration: robot Solo

Video available at https://youtu.be/RAiIo0l6_rE, taken from Alonso Marco, Dominik Baumann, Majid Khadiv, Ludovic Righetti, and Sebastian Trimpe, "Robot learning with crash constraints," IEEE Robotics & Automation Letters, 2021.

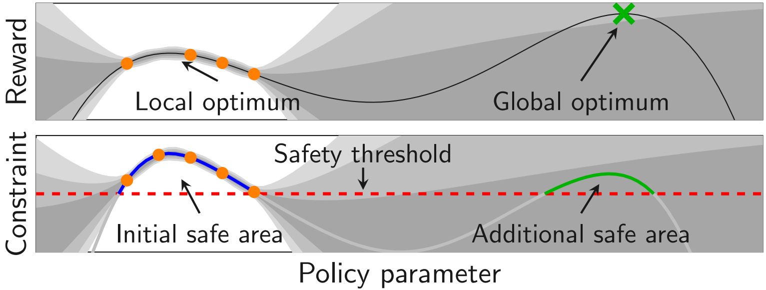
How to ensure safe exploration?

UPPSALA UNIVERSITET

- Safe learning is a very active research field[1]
- Can be model-based or model free
- Usually need some assumptions on dynamics, reward function
- *Safety* is often interpreted as not violating pre-defined constraints

[1] Overview of recent approaches: Lukas Brunke et al., "Safe learning in robotics: From learning-based control to safe reinforcement learning," Annual Review of Control, Robotics, and Autonomous Systems, 2021.

Safe Bayesian optimization – SafeOpt[2]



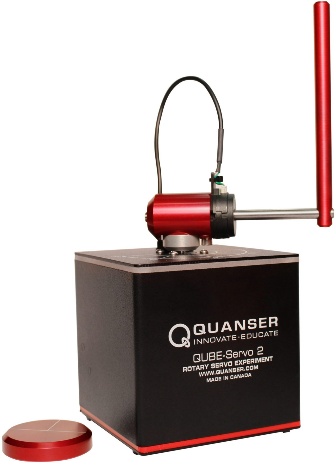
- We have a constraint function which must not become smaller than threshold
- Safe initial point given, reward and constraint obey some regularity conditions

→ Points close to safe initial point will also be safe with high probability

- However: cannot find safe regions disconnected in parameter space

[2] Felix Berkenkamp, Andreas Krause, and Angela P. Schoellig, "Bayesian Optimization with Safety Constraints: Safe and Automatic Parameter Tuning in Robotics," Machine Learning, 2021.

Can we do better?



- Assume we want to find a balancing controller for the rotary inverted pendulum
- After exploring the initial safe region, we found some stabilizing controllers
- If we now explore in regions where we cannot guarantee safety, we might get close to violating constraints

$\rightarrow$ Then, can use known safe controllers as *backup controllers* to restore safety

Safe global exploration

Video available at https://youtu.be/YgTEFE_Z0kc, taken from Dominik Baumann, Alonso Marco, Matteo Turchetta, and Sebastian Trimpe, "GoSafe: Globally optimal safe robot learning," IEEE International Conference on Robotics and Automation, 2021.