Lecture 2: Exploration and Exploitation

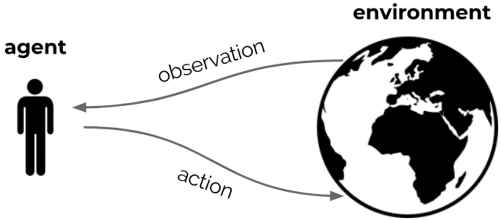
Hado van Hasselt

Thursday, January 16, 2020, UCL

Background

Sutton & Barto 2018, Chapter 2

Recap

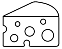


- Reinforcement learning is the science of learning to make decisions
- Agents can learn a **policy**, **value function** and/or a **model**
- The general problem involves taking into account **time** and **consequences**
- Decisions affect the **reward**, the **agent state**, and **environment state**

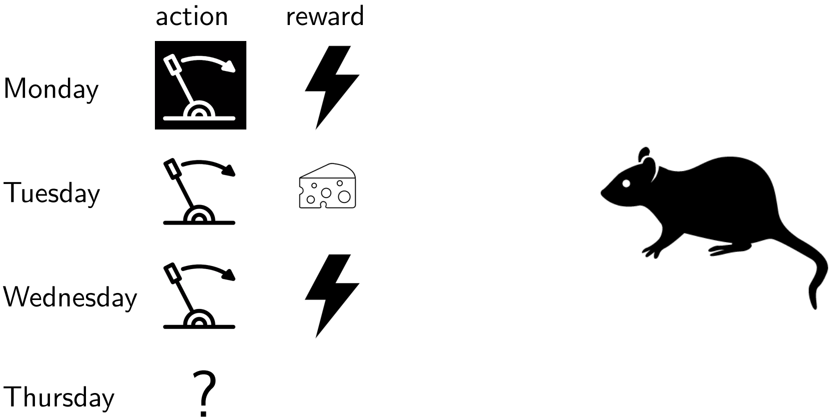
This Lecture

- Consider simple case: multiple actions, but only one state
- **No sequential structure** — past actions do not influence the future
- Formally: the distribution of R_t given A_t is identical and independent across time

Rat Example

	action	reward
Monday		
Tuesday		
Wednesday	?	

Rat Example

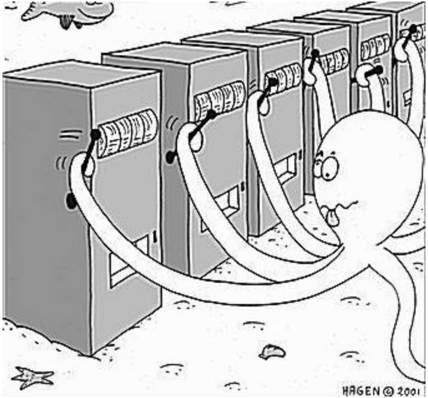


Exploration vs. Exploitation

- Online decision-making involves a fundamental choice:
  - **Exploitation**: Maximize performance based on current knowledge
  - **Exploration**: Increase knowledge
- The best long-term strategy may involve short-term sacrifices
- We want to gather **enough information** to make the **best overall decisions**
- This differs from just trading off rewards across time
  (which even an optimal policy may need to do)
- Instead, this is specific to learning actively — making decisions to help learning

The Multi-Armed Bandit

- We formalise the simplest setting
- $\mathcal{A}$ is a known set of actions (or “arms”)
- At each step t the agent selects an action $A_t \in \mathcal{A}$
- The environment generates a reward R_t
- The distribution $p(r \mid a)$ is fixed, but unknown
- The goal is to maximize cumulative reward $\sum_{i=1}^{t} R_i$
- Note: we sum over the whole lifetime of the agent
- Repeated ‘game against nature’



Action values

- The true **action value** for action a is the expected reward

$$q(a) = \mathbb{E}\left[R_t | A_t = a\right]$$

- A simple estimate is the average of the sampled rewards:

$$Q_t(a) = \frac{\sum_{n=1}^{t} R_n \, \mathcal{I}(A_n = a)}{\sum_{n=1}^{t} \mathcal{I}(A_n = a)}$$

  $\mathcal{I}(\cdot)$ is the **indicator** function: $\mathcal{I}(\text{True}) = 1$ and $\mathcal{I}(\text{False}) = 0$

- Let $N_t(a) = \sum_{n=1}^{t} \mathcal{I}(A_n = a)$ be the **count** for action a

Action values

- This can be updated incrementally:

$$Q_t(A_t) = Q_{t-1}(A_t) + \alpha_t \underbrace{(R_t - Q_{t-1}(A_t))}_{\text{error}},$$

$$\forall a \neq A_t : Q_t(a) = Q_{t-1}(a)$$

with

$$\alpha_t = \frac{1}{N_t(A_t)}, \qquad N_t(A_t) = N_{t-1}(A_t) + 1, \qquad \text{and} \qquad N_t(a) = 0, \forall a$$

- We will later consider other **step sizes** α
- For instance, constant α would lead to **tracking**, rather than averaging

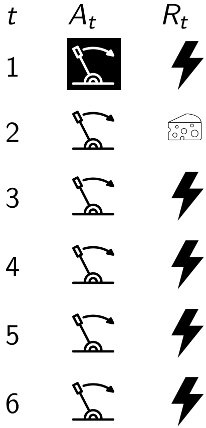
Rat Example

t	A_t	R_t
1		
2		
3		
4	?	

- Cheese: $R = +1$
- Shock: $R = -1$
- Then:

$$Q_3(\text{white}) = 0$$
$$Q_3(\text{black}) = -1$$

Rat Example



- Cheese: $R = +1$
- Shock: $R = -1$
- Then:

$$Q_6(\text{white}) = -\mathbf{0.6}$$
$$Q_6(\text{black}) = -1$$

- When to stop being greedy?

Regret

- How can we reason about the exploration trade off?
- Seems natural to account for **uncertainty** of value estimates
- Can we reason about this formally?
- Can we trade off exploration and exploitation **optimally**?

Regret

- The **optimal value** is

$$v_* = \max_{a \in \mathcal{A}} q(a) = \max_a \mathbb{E}\left[R_t \mid A_t = a\right]$$

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

$$v_* - q(A_t)$$

  - In hindsight, I might 'regret' taking the tube rather than cycling
  - I might have regretted taking a bus even more
- The agent cannot observe or even sample the regret
  (we don't know v_* or $q(A_t)$, can't directly sample v_*)
- But we can use it to analyze different learning algorithms

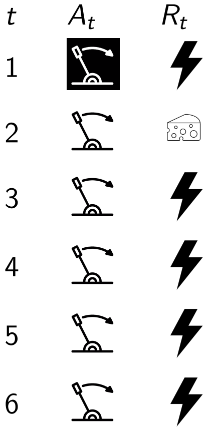
Regret

- Goal: Trade-off exploration and exploitation by minimizing **total regret**:

$$L_t = \sum_{i=1}^{t} (v_* - q(a_i))$$

- Note: maximise cumulative reward $\equiv$ minimise total regret
- Note: the sum extends beyond (single step) episodes
- View extends over 'lifetime of learning', rather than over 'current episode'

Regret of greedy



- Regret can grow unbounded
- More interesting is how **fast** it grows
- The **greedy** policy has **linear** regret
- This means that, in expectation, the regret grows as a function that is linear in t
  - Suppose $p(\text{cheese} \mid \text{white}) = 0.1$ and $p(\text{cheese} \mid \text{black}) = 0.9$
  - Then $v_* = q(\text{black}) = 0.8$ and $q(\text{white}) = -0.8$
  - The greedy rat incurs regret of $1.6t$ (If the first two actions and rewards are as shown on the left)

Counting Regret

- The **action regret** Δ_a for a given action is the difference between the optimal value and the **true** value of a:

$$\Delta_a = v_* - q(a)$$

- Total regret then depends on action regrets and action counts

$$L_t = \sum_{i=1}^{t} v_* - q(a_i) = \sum_{a \in \mathcal{A}} N_t(a)(v_* - q(a)) = \sum_{a \in \mathcal{A}} N_t(a)\Delta_a$$

- A good algorithm ensures small counts for large action regrets
- But, action regrets are unknown...

Exploration

- We need to **explore** to learn the values
- One common solution: ϵ-greedy
  - Select greedy action (**exploit**) w.p. $1 - \epsilon$
  - Select random action (**explore**) w.p. ϵ
- Is this enough?
- How to pick ϵ?

ϵ-Greedy Algorithm

- Greedy can lock onto a suboptimal action forever
  $\Rightarrow$ Greedy has linear expected total regret
- The **ϵ-greedy** algorithm continues to explore forever
  - With probability $1 - \epsilon$ select $a = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, Q_t(a)$
  - With probability ϵ select a random action
- Will continue to select all suboptimal actions with, at least, probability $\epsilon / |\mathcal{A}|$
  $\Rightarrow$ ϵ-greedy, with constant ϵ, has linear expected total regret

Lower Bound

- The performance of any algorithm is determined by similarity between optimal arm and other arms
- Hard problems have arms with similar distributions but different means
- This is described formally by the gap Δ_a and the similarity in distributions $KL(p(r|a)||p(r|a_*))$

Theorem (Lai and Robbins)

Asymptotic total regret is at least logarithmic in number of steps

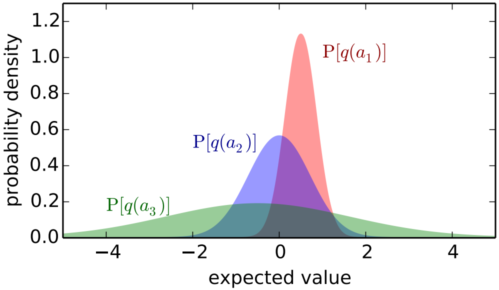
$$\lim_{t\to\infty} L_t \geq \log t \sum_{a|\Delta_a>0} \frac{\Delta_a}{KL(p(r|a)||p(r|a_*))}$$

Note: logarithmic is a whole lot better than linear!
(Note: $\mathrm{KL}_a \propto \Delta_a^2$)

Upper Bound?

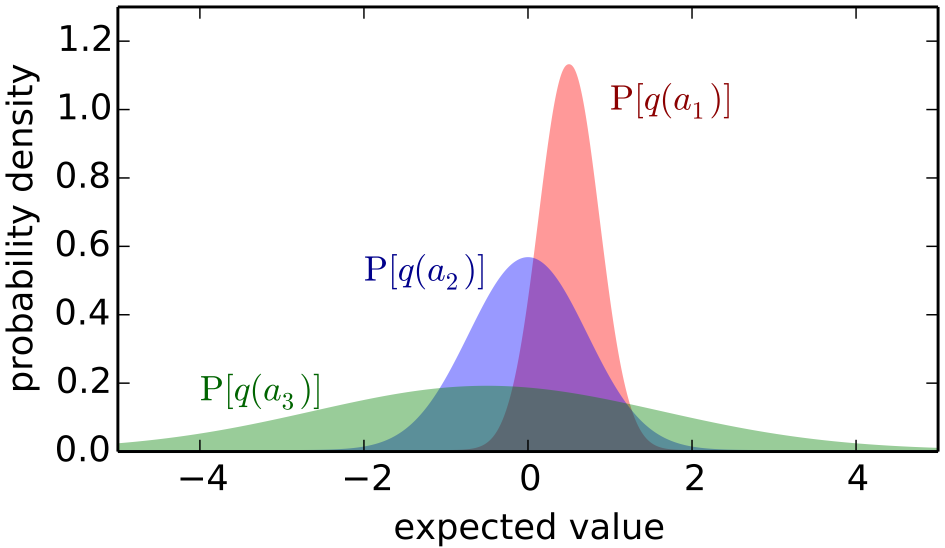
- We now have a lower bound on regret
- But what is attainable?
- Can we get close?

Optimism in the Face of Uncertainty

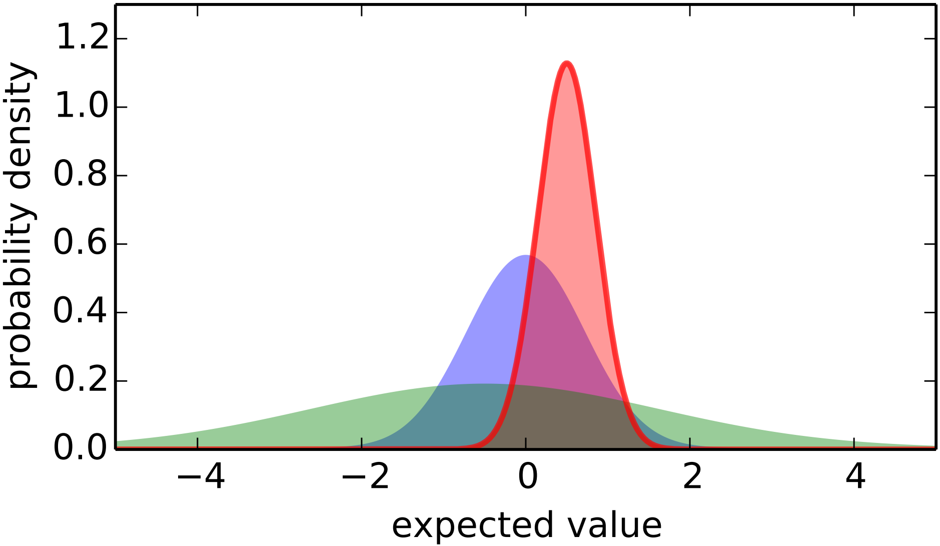


- Which action should we pick?
- More uncertainty about its value: more important to explore that action

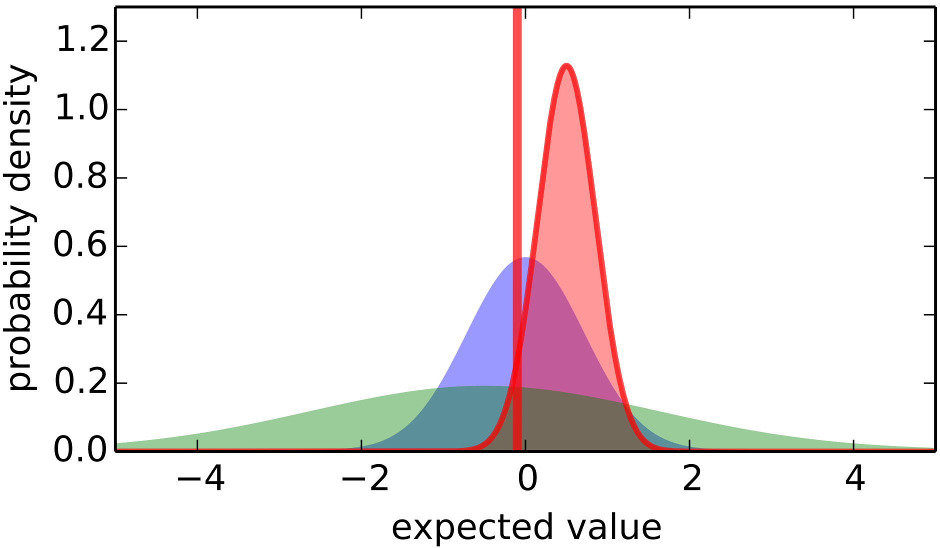
Optimism in the Face of Uncertainty



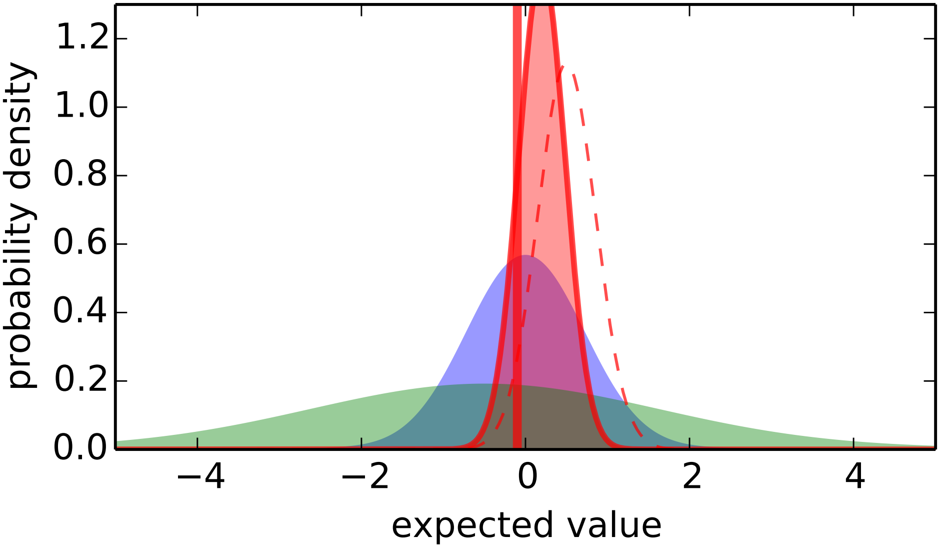
Optimism in the Face of Uncertainty



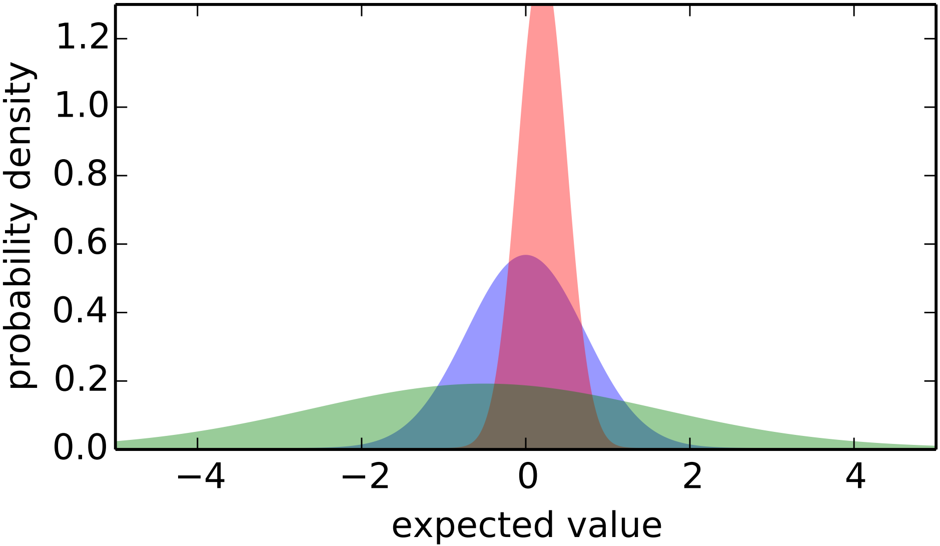
Optimism in the Face of Uncertainty



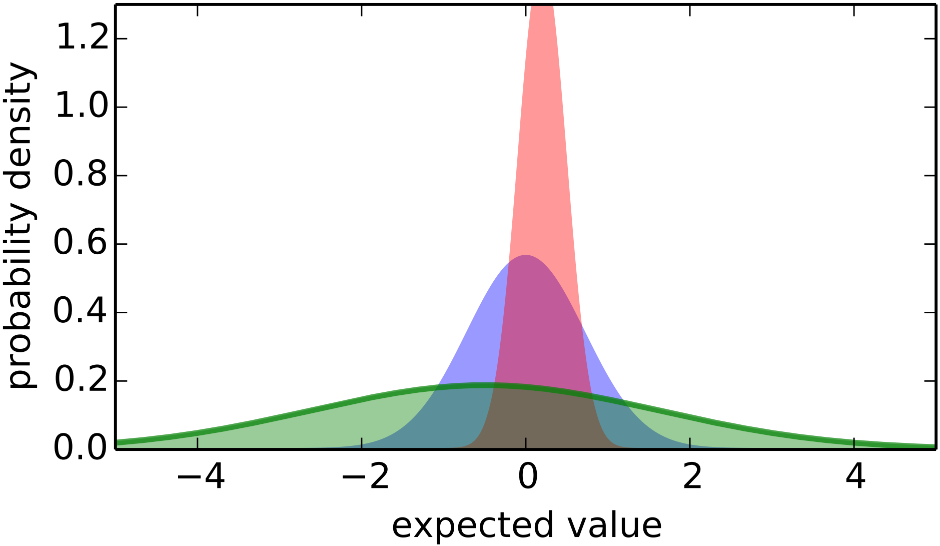
Optimism in the Face of Uncertainty



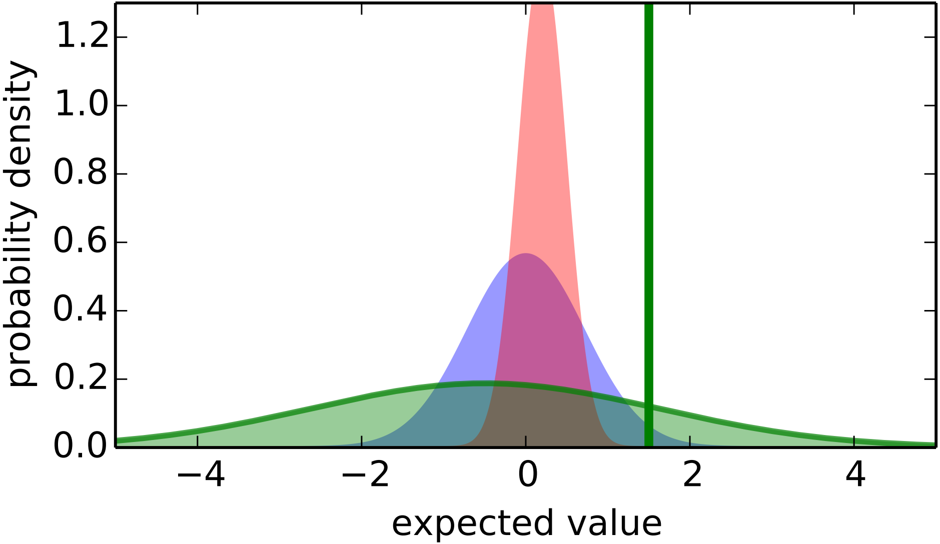
Optimism in the Face of Uncertainty



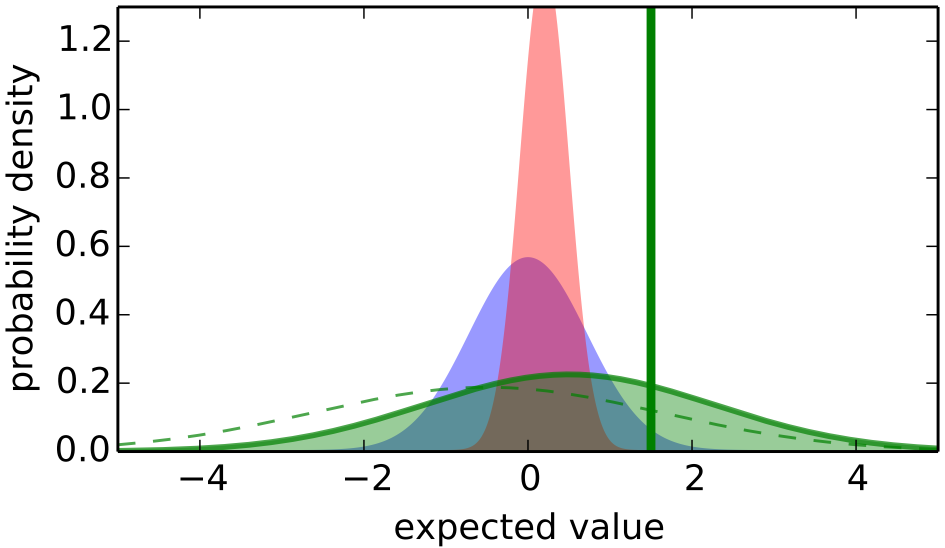
Optimism in the Face of Uncertainty



Optimism in the Face of Uncertainty



Optimism in the Face of Uncertainty



Upper Confidence Bounds

- Estimate an upper confidence $U_t(a)$ for each action value, such that $q(a) \leq Q_t(a) + U_t(a)$ with high probability
- Select action maximizing **upper confidence bound** (UCB)

$$a_t = \operatorname*{argmax}_{a \in \mathcal{A}} Q_t(a) + U_t(a)$$

- The uncertainty depends on the number of times $N(a)$ has been selected
  - Small $N_t(a) \Rightarrow$ large $U_t(a)$ (estimated value is uncertain)
  - Large $N_t(a) \Rightarrow$ small $U_t(a)$ (estimated value is accurate)
- For averages the uncertainty decreases as $\sqrt{N_t(a)}$, by the central limit theorem (If variance of rewards is bounded.)
- Can we **derive** an optimal algorithm?

Algorithm idea

- Recall, we want to minimize: $\sum_a N_t(a)\Delta_a$
- If regret Δ_a is big, we want count $N_t(a)$ to be small
- If count $N_t(a)$ is big, we want regret Δ_a to be small
- Not all $N_t(a)$ can be small (as their sum has to be t)
- We know $N_t(a)$; can we also know something about Δ_a?

Hoeffding's Inequality

Theorem (Hoeffding's Inequality)

Let $X_1, ..., X_n$ be i.i.d. random variables in [0,1], and let $\overline{X}_t = \frac{1}{n}\sum_{i=1}^{n} X_i$ be the sample mean. Then

$$p\left(\mathbb{E}\left[X\right] \geq \overline{X}_n + u\right) \leq e^{-2nu^2}$$

- We can apply Hoeffding's Inequality to bandits with bounded rewards
- E.g., if $R_t \in [0, 1]$, then

$$p\left(q(a) \geq Q_t(a) + U_t(a)\right) \leq e^{-2N_t(a)U_t(a)^2}$$

Calculating Upper Confidence Bounds

- Pick a **maximal** probability p that the true value exceeds upper bound
- Now solve for $U_t(a)$

$$e^{-2N_t(a)U_t(a)^2} = p$$

$$\implies \qquad U_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}}$$

  We then know the **actual** probability that this happens is smaller than p
- Idea: reduce p as we observe more rewards, e.g., $p = 1/t$

$$U_t(a) = \sqrt{\frac{\log t}{2N_t(a)}}$$

- This ensures that we always keep exploring
- But we select optimal action much more often as $t \to \infty$

UCB

- This leads to the UCB algorithm

$$a_t = \operatorname*{argmax}_{a \in \mathcal{A}} Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}}$$

where c can be considered a hyper-parameter

Theorem (Auer et al., 2002)

The UCB algorithm (with $c = \sqrt{2}$) achieves logarithmic expected total regret

$$L_t \leq 8 \sum_{a|\Delta_a > 0} \frac{\log t}{\Delta_a} + O(\sum_a \Delta_a)\,, \qquad \forall t\,.$$

UCB

- UCB:

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \; Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}}$$

- Intuition:
  - Suppose that Δ_a is large.
  - Then, $N_t(a)$ will be small, because $U_t(a)$ rarely spans the whole gap.
  - So, either Δ_a is low, or $N_t(a)$ is low.
  - In fact, $\Delta_a N_t(a) \leq O(\log t)$, for all a

Values or Models?

- This is a value-based algorithm:

$$Q_t(A_t) = Q_{t-1}(A_t) + \alpha_t(R_t - Q_{t-1}(A_t)) .$$

- What about a model-based approach?

$$\hat{\mathcal{R}}_t(A_t) = \hat{\mathcal{R}}_{t-1}(A_t) + \alpha_t(R_t - \hat{\mathcal{R}}_{t-1}(A_t)) .$$

- Indistinguishable? (For bandits)
- We could model more, e.g., the **distribution of rewards**

Bayesian Bandits

- **Bayesian bandits** model parameterized distributions over rewards, $p(R_t \mid \theta, a)$
- Compute posterior distribution over θ

$$p_t(\theta \mid a) \quad \propto \quad p(R_t \mid \theta, a) p_{t-1}(\theta \mid a)$$

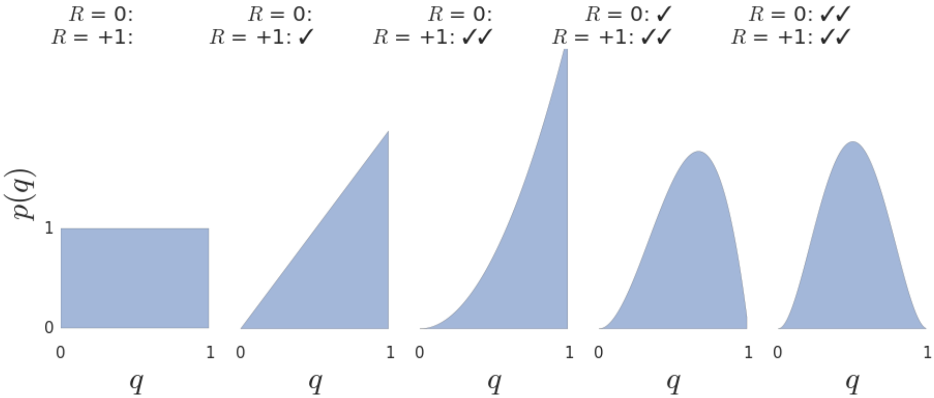
- Allows us to inject rich prior knowledge $p_0(\theta \mid a)$
- Use posterior to guide exploration
  - Upper confidence bounds
  - Probability matching

Bayesian Bandits: Example

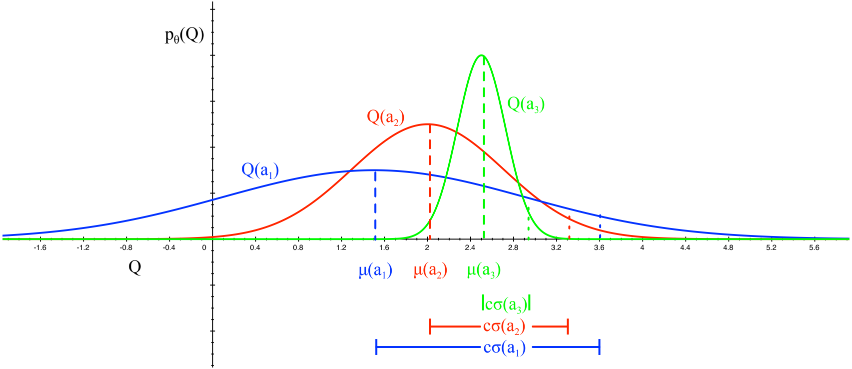
- Consider bandits with **Bernoulli** reward distribution: rewards are 0 or $+1$
- For each action, the prior could be a **uniform distribution** on $[0, 1]$
- This means we think each mean reward in $[0, 1]$ is equally likely
- The posterior is a Beta distribution $\text{Beta}(x_a, y_a)$ with initial parameters $x_a = 1$ and $y_a = 1$ for each action a
- Updating the posterior:
  - $x_{A_t} \leftarrow x_{A_t} + 1$ when $R_t = 0$
  - $y_{A_t} \leftarrow y_{A_t} + 1$ when $R_t = 1$

Bayesian Bandits: Example

Suppose: $R_1 = +1,\ R_2 = +1,\ R_3 = 0,\ R_4 = 0$



Bayesian Bandits with Upper Confidence Bounds



- Compute posterior distribution over action-values
- Estimate upper confidence from posterior
  - e.g., $U_t(a) = c\sigma_t(a)$ where $\sigma(a)$ is std dev of $p_t(q(a))$
- Pick action that maximizes $Q_t(a) + c\sigma(a)$

Probability Matching

- **Probability matching** selects action a according to probability that a is the optimal action

$$\pi_t(a) = p\left(q(a) = \max_{a'} q(a') \mid H_{t-1}\right)$$

- Probability matching is optimistic in the face of uncertainty: Uncertain actions have higher probability of being max
- Can be difficult to compute $\pi(a)$ analytically from posterior

Thompson Sampling

- Thompson sampling:
  - Sample $Q_t(a) \sim p_t(q(a)), \forall a$
  - Select action maximising sample, $A_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, Q_t(a)$
- **Thompson sampling** is sample-based probability matching

$$\begin{aligned} \pi_t(a) &= \mathbb{E}\left[\mathcal{I}(Q_t(a) = \max_{a'} Q_t(a'))\right] \\ &= p\left(q(a) = \max_{a'} q(a')\right) \end{aligned}$$

- For Bernoulli bandits, Thompson sampling achieves Lai and Robbins lower bound on regret, and therefore is **optimal**

Value of Information

- Exploration is valuable because information is valuable
- Can we quantify the value of information?
- You gain more information when you are uncertain
- Therefore it makes sense to explore novel situations more
- If we know value of information, we can trade-off exploration and exploitation

Information State Space

- We have viewed bandits as **one-step** decision-making problems
- Can also view as **sequential** decision-making problems
- At each step there is an **information state** $\tilde{S}_t$ summarising all information accumulated so far
- Each action A_t causes a transition to a new information state $\tilde{S}_{t+1}$ (by adding information), with probability $p(\tilde{S}_{t+1} \mid A_t, \tilde{S}_t)$
- We then have a Markov decision problem
- Here states = observations = internal information state
  (There is no external environment state)
- Even in bandits, actions affect the future after all, because they affect learning

Example: Bernoulli Bandits

- Consider a Bernoulli bandit, such that

$$
\begin{aligned}
p\left(R_t = 1 \mid A_t = a\right) &= \mu_a \\
p\left(R_t = 0 \mid A_t = a\right) &= 1 - \mu_a
\end{aligned}
$$

- e.g. Win or lose a game with probability μ_a
- Want to find which arm has the highest μ_a
- The information state is $I = (\boldsymbol{\alpha}, \boldsymbol{\beta})$
  - α_a counts the pulls of arm a where reward was 0
  - β_a counts the pulls of arm a where reward was 1

Solving Information State Space Bandits

- We formulated the bandit as an infinite MDP over information states
- This can be solved by reinforcement learning
- Model-free reinforcement learning
  - e.g. Q-learning (Duff, 1994)
- Bayesian model-based reinforcement learning
  - e.g. Gittins indices (Gittins, 1979)
- The latter approach is known as **Bayes-adaptive** RL
- Finds Bayes-optimal exploration/exploitation trade-off with respect to the prior distribution
- Can be unwieldy... unclear how to scale

Policy search

- What about learning policies $\pi(a)$ directly?
- Lets parameterize the policy — can we learn this without learning values?
- For instance, define **action preferences** $H_t(a)$ and a policy

$$\pi(a) = \frac{e^{H_t(a)}}{\sum_b e^{H_t(b)}} \qquad \textbf{(softmax)}$$

- The preferences do not have to have value semantics
- View them as learnable parameters
- Can we optimize the preferences

Policy gradients

- Idea: update policy parameters such that the expected value increases
- We can consider **gradient ascent** on the expected value
- So, in the bandit case, we want to update:

  $$\theta = \theta + \alpha \nabla_\theta \mathbb{E}[R_t | \pi_\theta] ,$$

  where θ are the policy parameters
- Can we compute this gradient?

Gradient bandits

- Log-likelihood trick (also known as REINFORCE trick, Williams 1992):

$$
\begin{aligned}
\nabla_\theta \, \mathbb{E}[R_t|\pi_\theta] &= \nabla_\theta \sum_a \pi_\theta(a) \, \overbrace{\mathbb{E}[R_t|A_t = a]}^{= q(a)} \\
&= \sum_a q(a) \, \nabla_\theta \pi_\theta(a) \\
&= \sum_a q(a) \, \frac{\pi_\theta(a)}{\pi_\theta(a)} \, \nabla_\theta \pi_\theta(a) \\
&= \sum_a \pi_\theta(a) \, q(a) \, \frac{\nabla_\theta \pi_\theta(a)}{\pi_\theta(a)} \\
&= \mathbb{E}\left[R_t \, \frac{\nabla_\theta \pi_\theta(A_t)}{\pi_\theta(A_t)}\right] \qquad = \mathbb{E}\left[R_t \nabla_\theta \log \pi_\theta(A_t)\right]
\end{aligned}
$$

Gradient bandits

- Log-likelihood trick (also known as REINFORCE trick, Williams 1992):

$$\nabla_\theta \mathbb{E}[R_t|\theta] = \mathbb{E}\left[R_t \nabla_\theta \log \pi_\theta(A_t)\right]$$

- We can sample this!
- So

$$\theta = \theta + \alpha R_t \nabla_\theta \log \pi_\theta(A_t)\,,$$

  this is **stochastic gradient ascent** on the (true) value of the policy
- Can use **sampled** rewards — does not need value estimates

Gradient bandits

- For soft max:

$$H_{t+1}(a) = H_t(a) + \alpha R_t \frac{\partial \log \pi_t(A_t)}{\partial H_t(a)}$$
$$= H_t(a) + \alpha R_t(\mathcal{I}(a = A_t) - \pi_t(a))$$

- $\Rightarrow$

$$H_{t+1}(A_t) = H_t(A_t) + \alpha R_t(1 - \pi_t(A_t))$$
$$H_{t+1}(a) = H_t(a) - \alpha R_t \pi_t(a) \qquad \text{if } a \neq A_t$$

- Preferences for actions with higher rewards increase more (or decrease less), making them more likely to be selected again

Policy gradients with baselines

- Note that $\sum_a \pi_\theta(a) = 1$. Therefore, for any b,

$$\sum_a b\nabla_\theta \pi_\theta(a) = b\nabla_\theta \sum_a \pi_\theta(a)$$
$$= 0$$

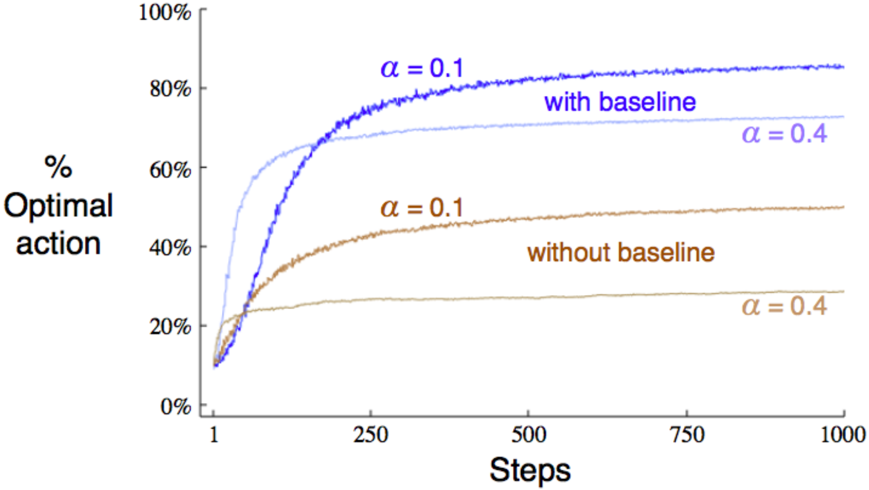
  as long as b does not depend on θ or a.
- This means we can subtract a **baseline**, and instead use

$$\theta = \theta + \alpha(R_t - b)\nabla_\theta \log \pi_\theta(A_t)$$

- Baselines <u>do not</u> change the expected update, but they <u>do</u> change variance

Policy gradients with baselines

A natural baseline is the average reward $\frac{1}{t}\sum_{i=1}^{t} R_i$

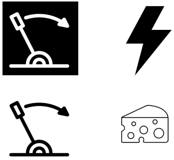


Gradient bandits

- These gradient methods can be extended
  - ...to include context
  - ...to full MDPs
  - ...to partial observability
- We will discuss them again in lecture on **policy gradients**

Rat Example

action reward



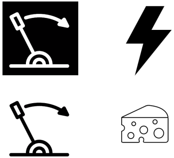
?

probability of selecting **black**

- Greedy: ?
- ϵ-greedy: ?
- UCB: ?
- Thompson sampling: ?

Rat Example

action reward



?

probability of selecting **black**

- Greedy: 0
- ϵ-greedy: $\epsilon/2$
- UCB: 0
- Thompson sampling: 0.25