Lecture 2: Exploration and Exploitation in Multi-Armed Bandits

Hado van Hasselt

Outline

1 Recap

2 Introduction

3 Multi-Armed Bandits

4 Contextual Bandits

5 Policy-Based methods

Previous lecture

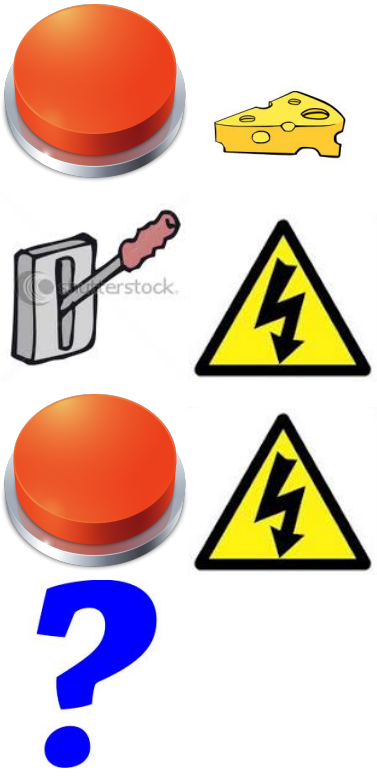
- Reinforcement learning is the science of learning to make decisions
- We can do this by learning one or more of:
  - policy
  - value function
  - model
- The general problem involves taking into account time and consequences
- Our decisions affect the reward, our internal knowledge, and the state of the environment

This Lecture

- Multiple actions, but (mostly) only one state
- Decisions do not affect the state of the environment
- Goal: optimize immediate reward in a repeated 'game against nature'
- History (no observations):

$$H_t = A_1, R_1, A_2, R_2, \ldots, A_t, R_t$$

Rat Example



Exploration vs. Exploitation

- Online decision-making involves a fundamental choice:
  - Exploitation: Maximize return given current knowledge
  - Exploration: Increase knowledge
- The best long-term strategy may involve short-term sacrifices
- Gather enough information to make the best overall decisions

Examples

- Restaurant Selection

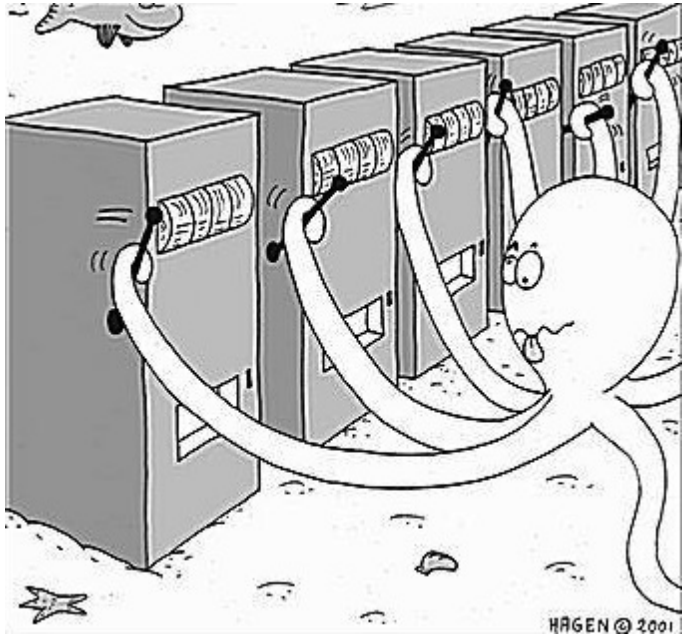
  Exploitation: Go to your favourite restaurant
  Exploration: Try a new restaurant

- Online Banner Advertisements

  Exploitation: Show the most successful advert
  Exploration: Show a different advert

The Multi-Armed Bandit

- A multi-armed bandit is a tuple $\langle \mathcal{A}, \mathcal{R} \rangle$
- $\mathcal{A}$ is a known set of actions (or "arms")
- $\mathcal{R}^a(r) = \mathbb{P}[R_t = r | A_t = a]$ is an unknown probability distribution over rewards
- At each step t the agent selects an action $A_t \in \mathcal{A}$
- The environment generates a reward $R_t \sim \mathcal{R}^{A_t}$
- The goal is to maximize cumulative reward $\sum_{i=1}^{t} R_i$
- Repeated 'game against nature'



Action values

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

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

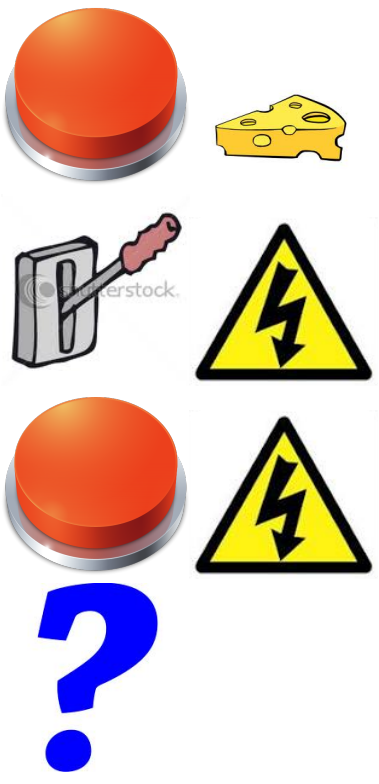
- We consider algorithms that estimate $Q_t(a) \approx q(a)$
- The *count* $N_t(a)$ is number of times we selected action a
- Monte-Carlo estimates:

$$Q_t(a) = \frac{1}{N_t(a)} \sum_{t=1}^{T} R_t \mathcal{I}(A_t = a)$$

- The *greedy* algorithm selects action with highest value

$$a_t^g = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\ Q_t(a)$$

Rat Example

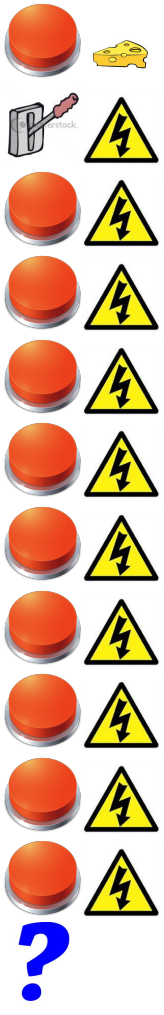


- Cheese: $R = +1$
- Shock: $R = -1$
- We can estimate action values:

$$Q_3(\text{button}) = 0$$
$$Q_3(\text{lever}) = -1$$

- When should we stop being greedy?

Rat Example



- Cheese: $R = +1$
- Shock: $R = -1$
- We can estimate action values:

$$Q_3(\text{button}) = -\mathbf{0.8}$$
$$Q_3(\text{lever}) = -1$$

- When should we stop being greedy?

Regret

- The *optimal value* v_* is

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

- Regret is the opportunity loss for one step

$$v_* - q(A_t)$$

  - I might regret fruit instead of pancakes for breakfast
  - I might regret porridge instead of pancakes even more

Regret

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

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

- Maximise cumulative reward $\equiv$ minimise total regret
- Note: cumulation here extends over termination of 'episode'
- View extends over 'lifetime of learning', rather than over 'current episode'

Counting Regret

- The *gap* Δ_a is the difference in value between action a and optimal action a_*, $\Delta_a = v_* - q(a)$
- Total regret depends on gaps and counts

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

- A good algorithm ensures small counts for large gaps
- Problem: gaps are not known...

Exploration

- We need to explore to learn about the values of all actions
- What is a good way to explore?
- One common solution: ϵ-greedy
  - Select greedy action (exploit) w.p. $1 - \epsilon$
  - Select random action (explore) w.p. ϵ
- Used in Atari
- 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 $a = \underset{a \in \mathcal{A}}{\text{argmax}}\ Q_t(a)$
  - With probability ϵ select a random action
  - Constant ϵ ensures minimum expected regret

$$\mathbb{E}\left[v_* - q(A_t)\right] \geq \frac{\epsilon}{\mathcal{A}} \sum_{a \in \mathcal{A}} \Delta_a$$

- $\Rightarrow$ ϵ-greedy with constant ϵ has linear total regret

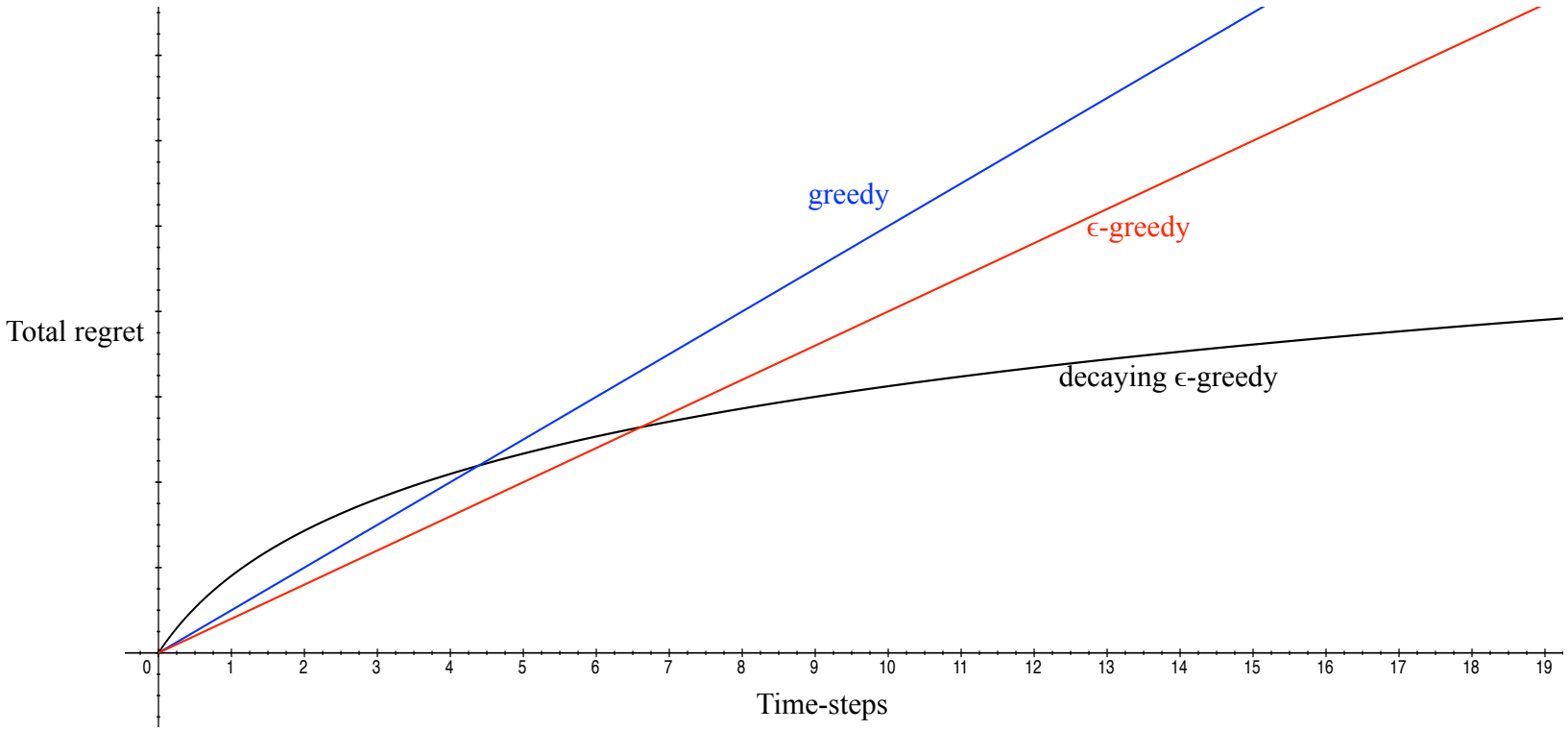
Decaying ϵ_t-Greedy Algorithm

- Pick a decay schedule for $\epsilon_1, \epsilon_2, ...$
- Consider the following schedule

$$c > 0$$
$$d = \min_{a|\Delta_a > 0} \Delta_i$$
$$\epsilon_t = \min \left\{ 1, \frac{c|\mathcal{A}|}{d^2 t} \right\}$$

- Decaying ϵ_t-greedy has *logarithmic* asymptotic total regret!
- Unfortunately, requires advance knowledge of gaps
- Goal: find an algorithm with sublinear regret for any multi-armed bandit (without knowledge of $\mathcal{R}$)

Linear or Sublinear Regret



Lower Bound

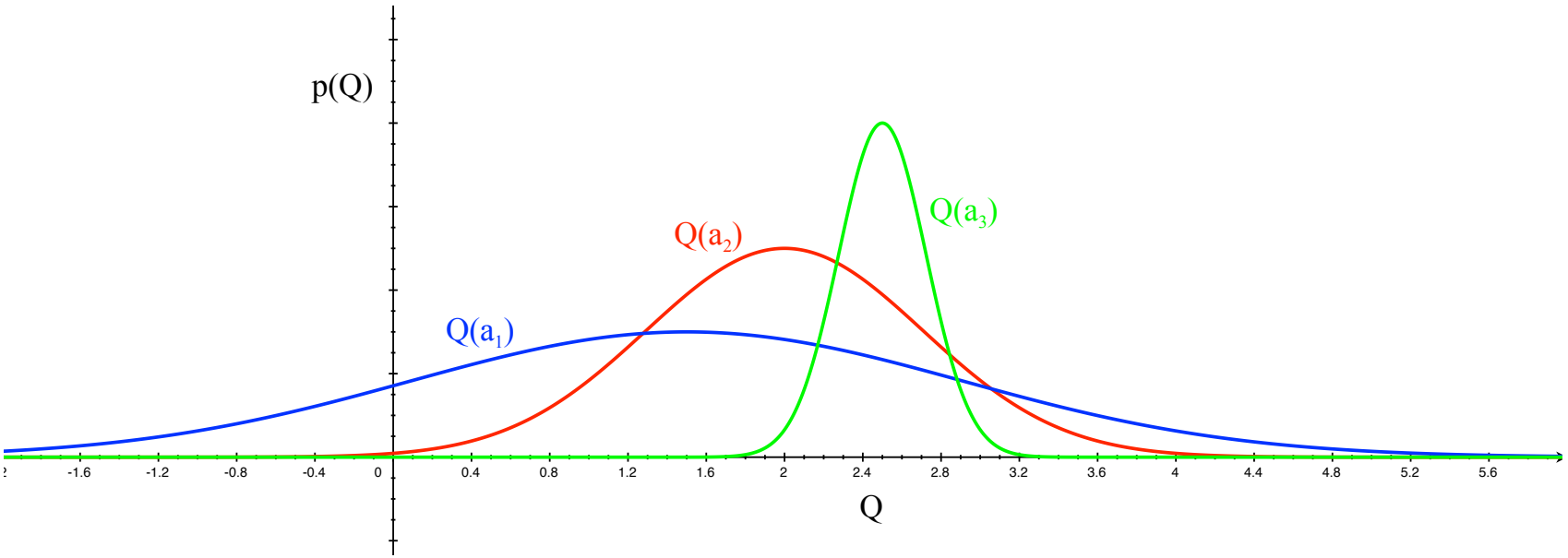
- The performance of any algorithm is determined by similarity between optimal arm and other arms
- Hard problems have arms with similar distributions but different means
- This is described formally by the gap Δ_a and the similarity in distributions $KL(\mathcal{R}^a||\mathcal{R}^a*)$

Theorem (Lai and Robbins)

Asymptotic total regret is at least logarithmic in number of steps

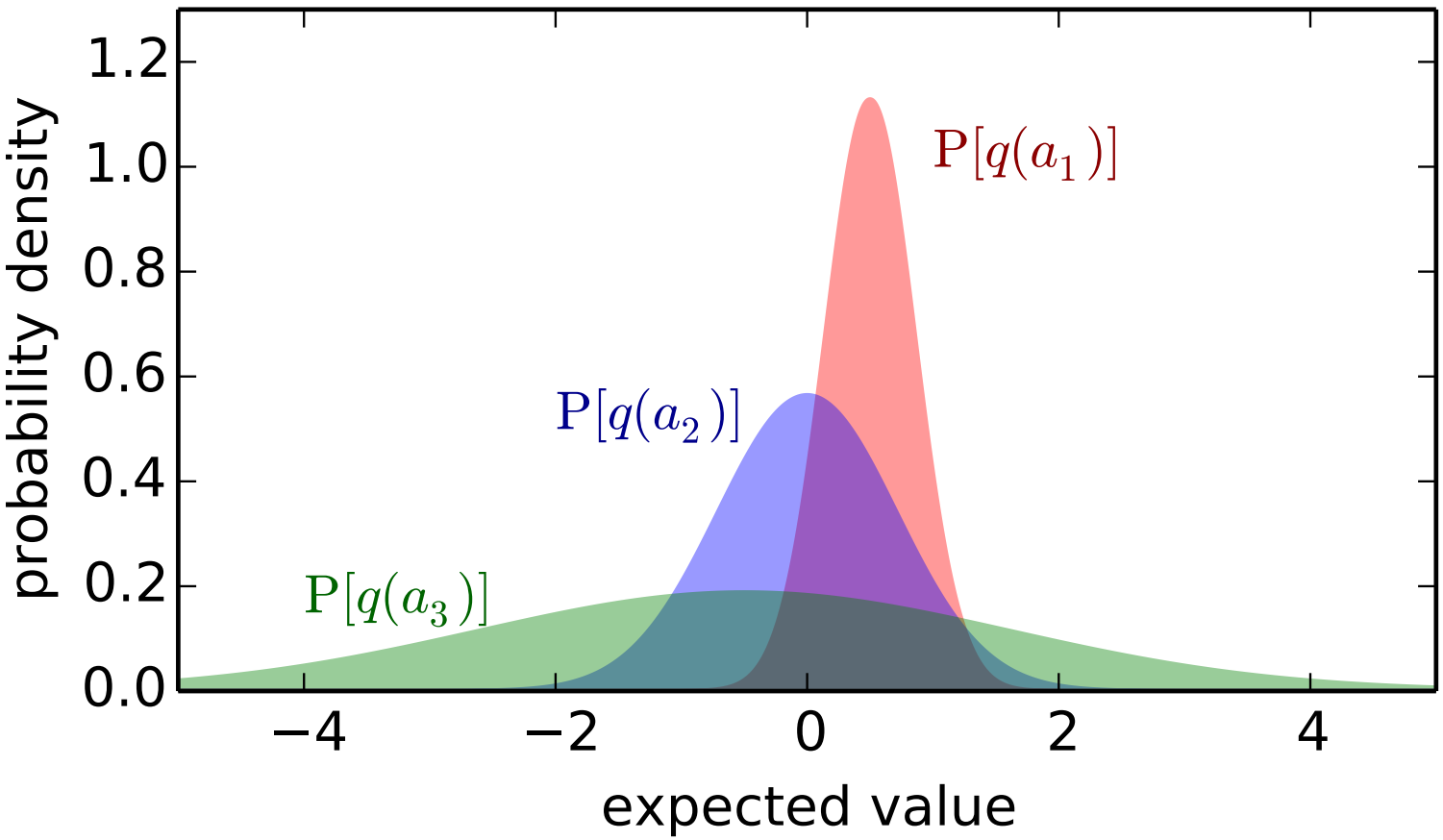
$$\lim_{t\to\infty} L_t \geq \log t \sum_{a|\Delta_a>0} \frac{\Delta_a}{KL(\mathcal{R}^a||\mathcal{R}^{a*})}$$

Optimism in the Face of Uncertainty

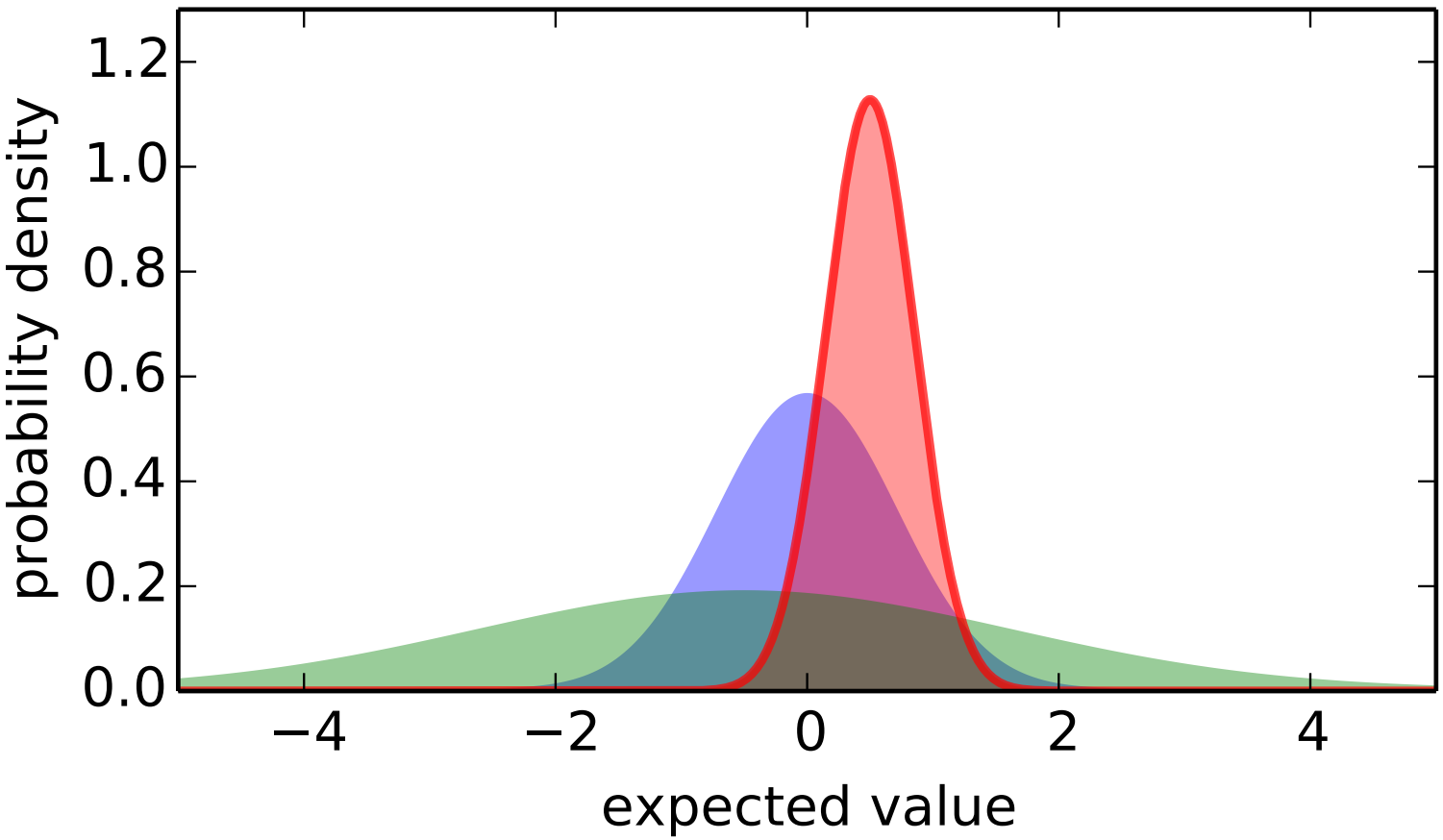


- Which action should we pick?
- More uncertainty: more important to explore that action
- It could turn out to be the best action

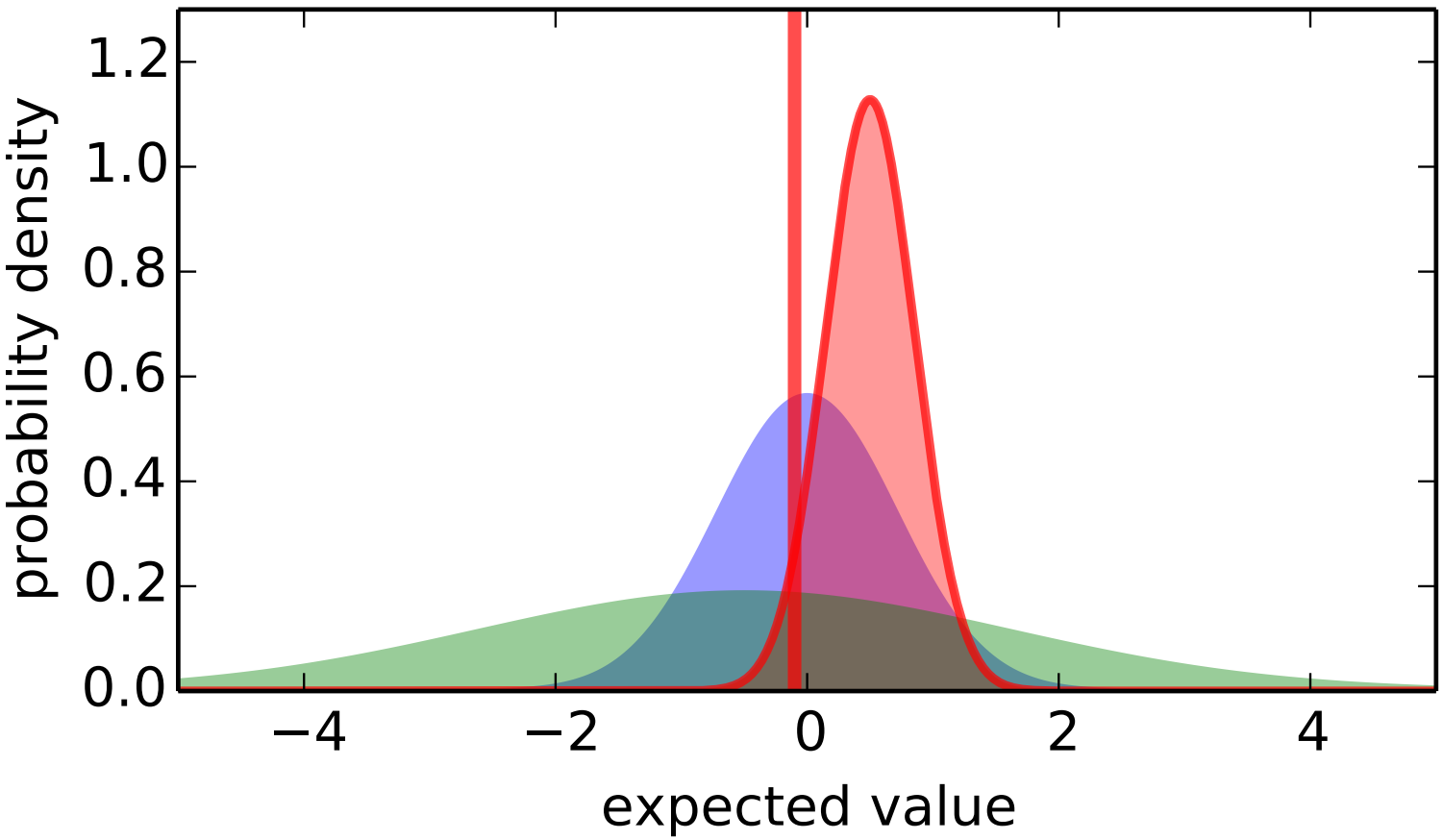
Optimism in the Face of Uncertainty



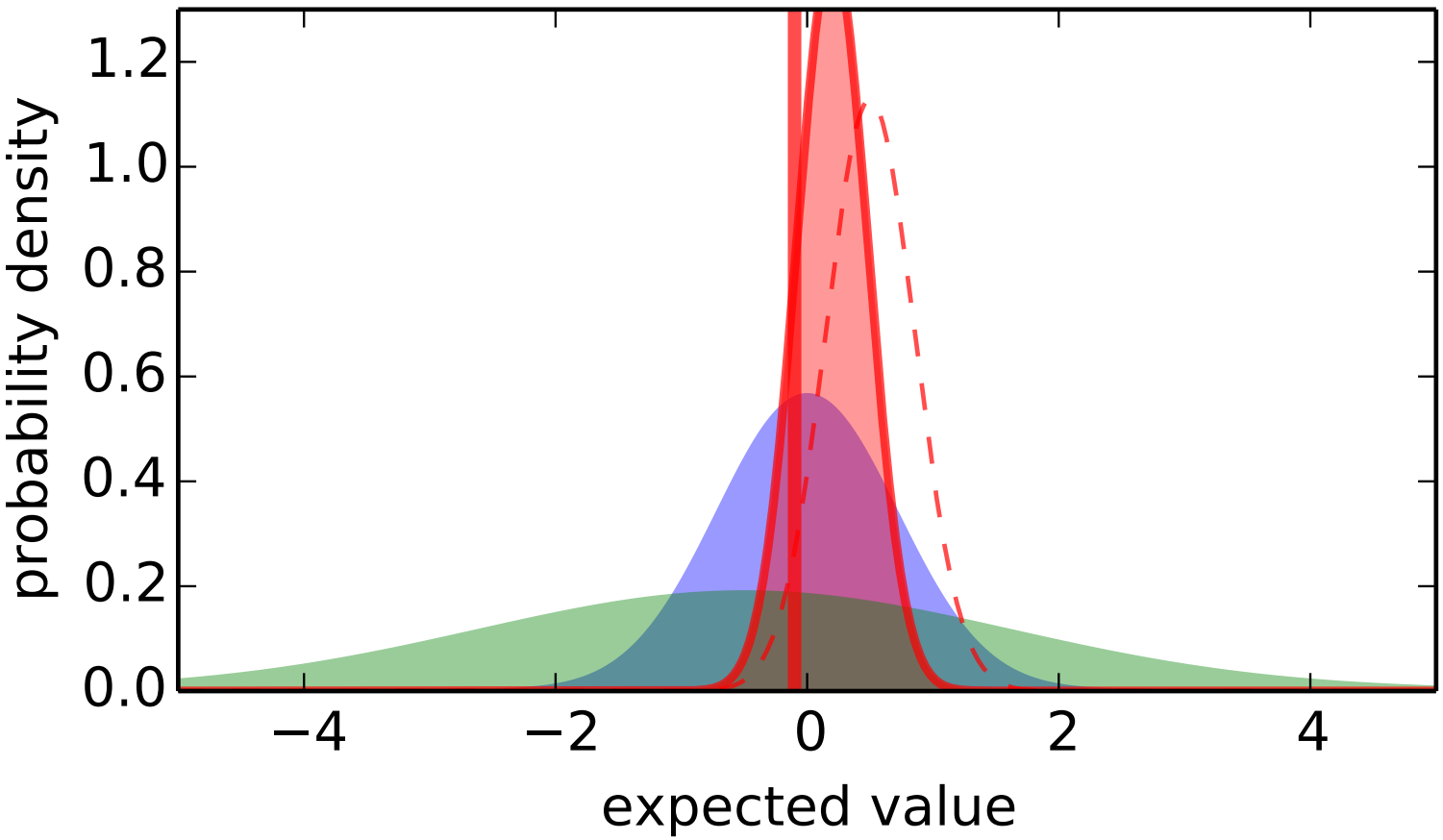
Optimism in the Face of Uncertainty



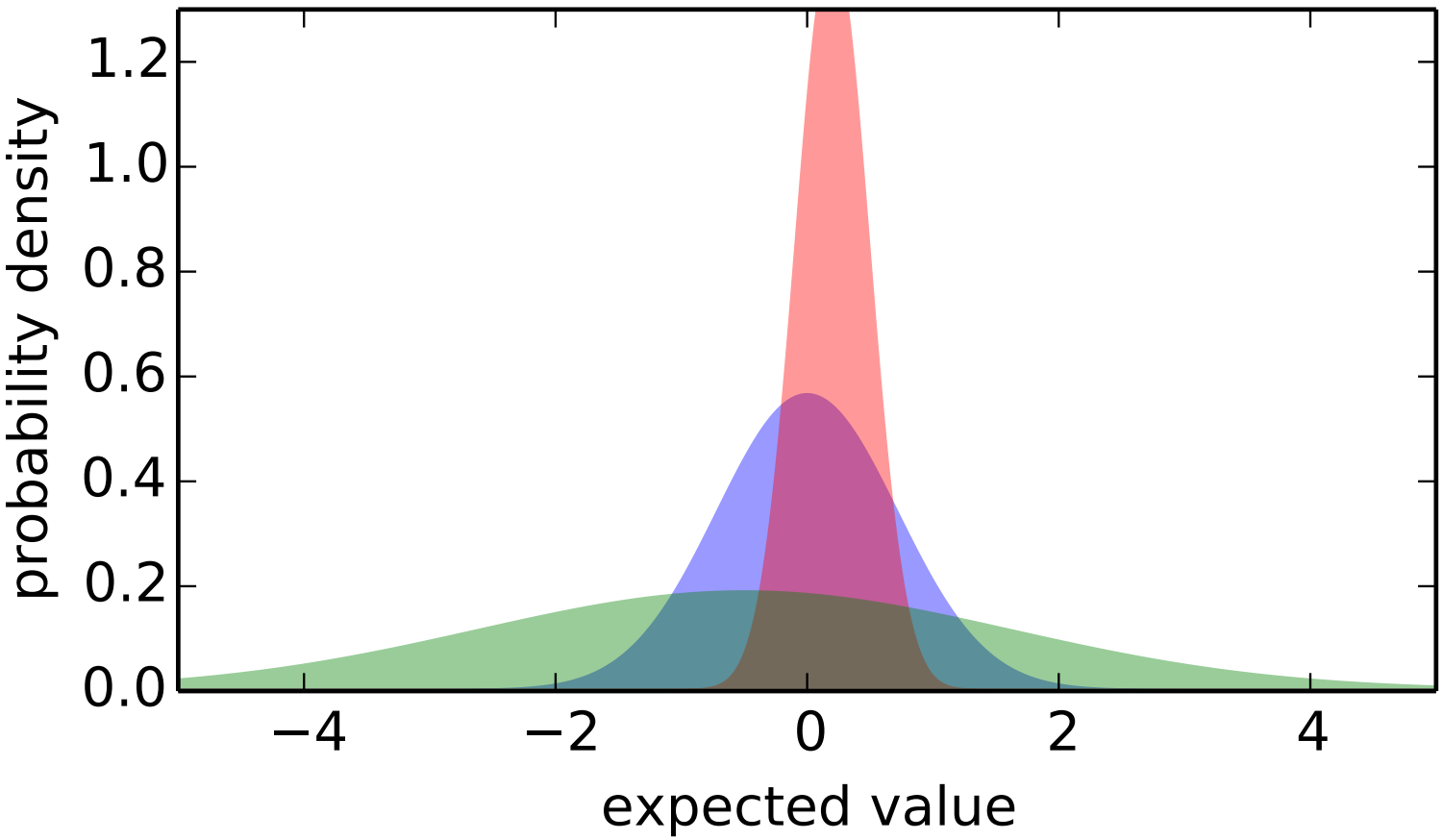
Optimism in the Face of Uncertainty



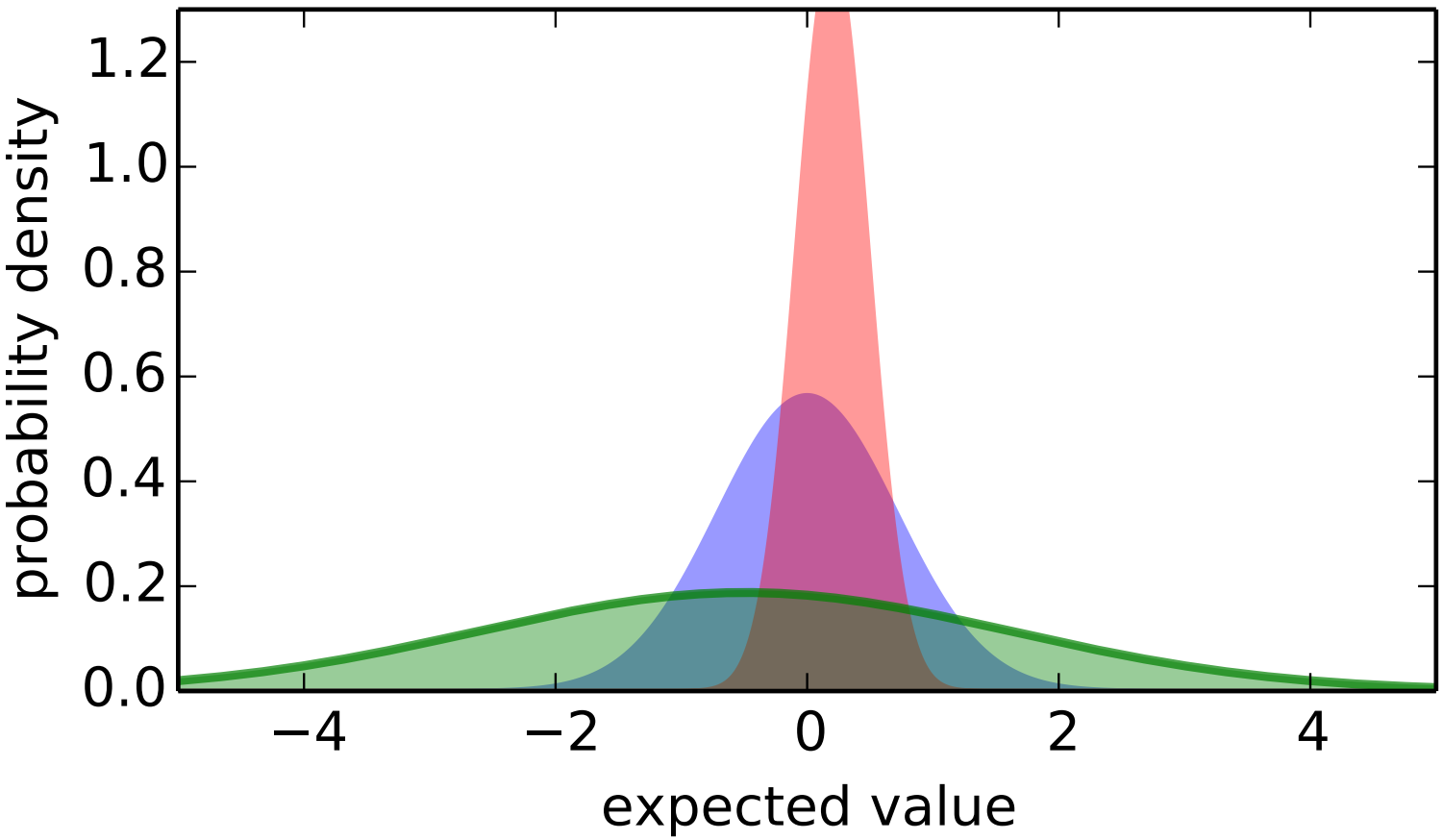
Optimism in the Face of Uncertainty



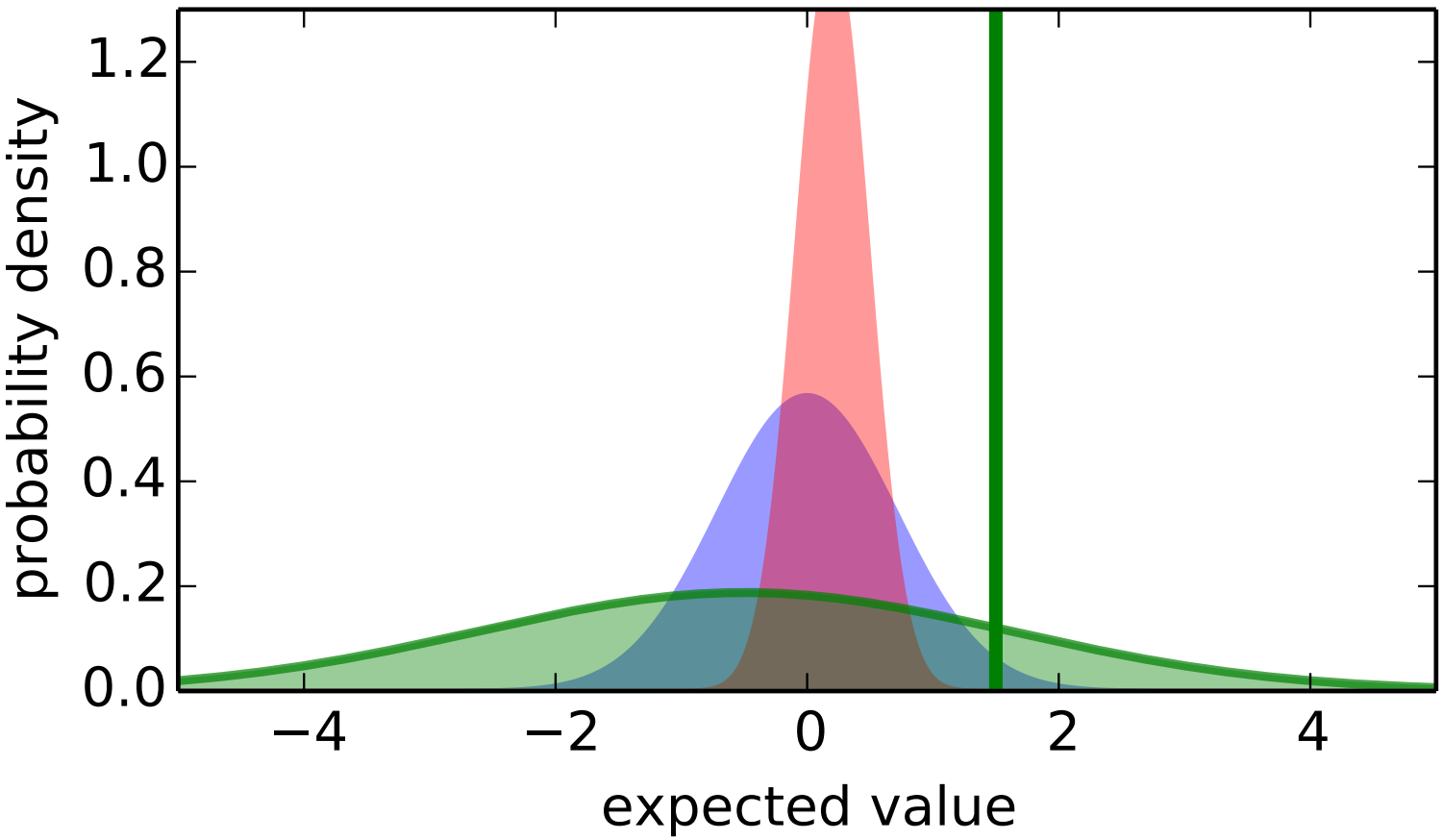
Optimism in the Face of Uncertainty



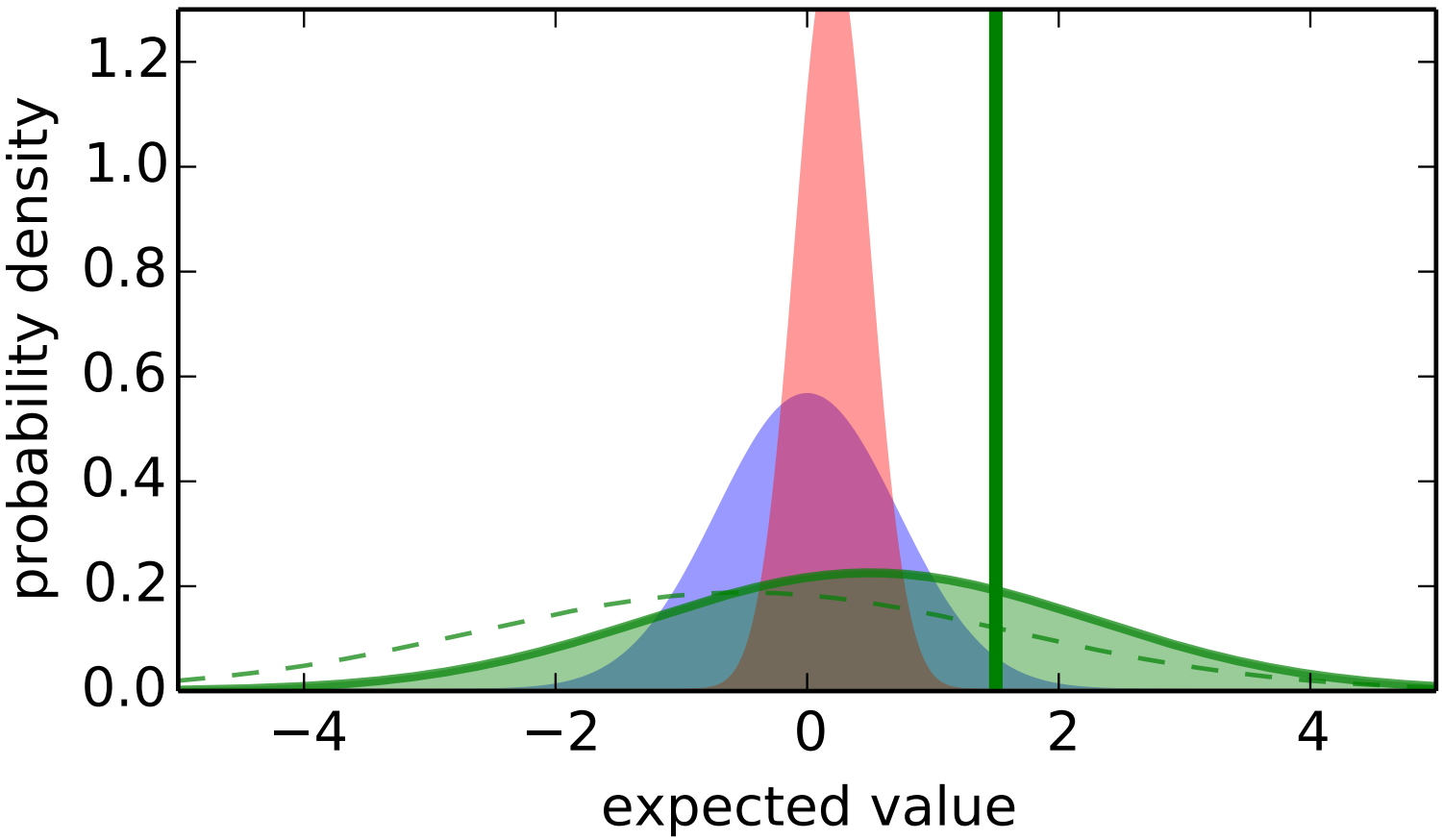
Optimism in the Face of Uncertainty



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
- 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)
- Select action maximizing Upper Confidence Bound (UCB)

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

Hoeffding's Inequality

Theorem (Hoeffding's Inequality)

Let $\Sigma_1, ..., \Sigma_t$ be i.i.d. random variables in [0,1], and let $\overline{X}_t = \frac{1}{t}\sum_{i=1}^{t}\Sigma_i$ be the sample mean. Then

$$\mathbb{P}\left[\mathbb{E}\left[X\right] > \overline{X}_t + u\right] \leq e^{-2tu^2}$$

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

$$\mathbb{P}\left[q(a) > Q_t(a) + U_t(a)\right] \leq e^{-2N_t(a)U_t(a)^2}$$

Calculating Upper Confidence Bounds

- Pick a probability p that true value exceeds UCB
- Now solve for $U_t(a)$

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

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

- Reduce p as we observe more rewards, e.g. $p = t^{-4}$
- Ensures we select optimal action as $t \to \infty$

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

UCB1

- This leads to the UCB1 algorithm

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\ Q_t(a) + \sqrt{\frac{2 \log t}{N_t(a)}}$$

Theorem (Auer et al., 2002)

The UCB algorithm achieves logarithmic expected total regret

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

for any t

Values or Models?

- This is a value-based algorithm:

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

- (Same as before, but rewritten as update)
- What about a model-based approach?

$$\hat{\mathcal{R}}_t^{A_t} = \hat{\mathcal{R}}_{t-1}^{A_t} + \frac{1}{N_t(A_t)}(R_t - \hat{\mathcal{R}}_{t-1}^{A_t})\,.$$

- Indistinguishable?
- Not if we model *distribution of rewards*

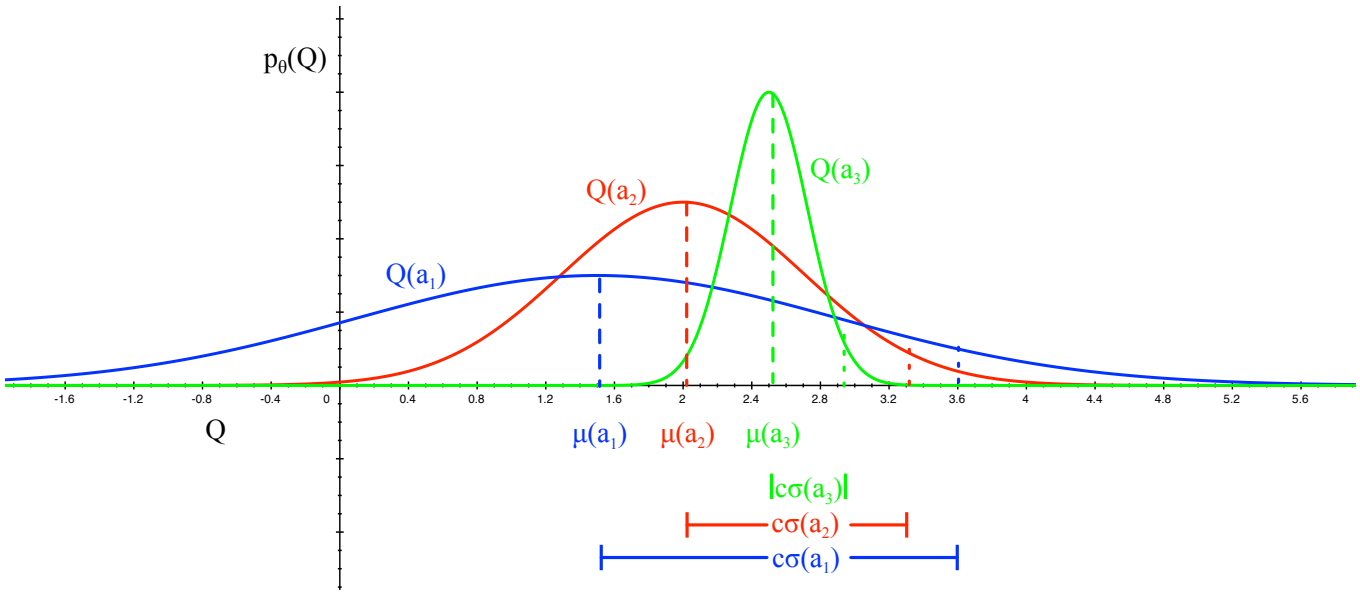
Bayesian Bandits

- *Bayesian bandits* model parameterized distributions over rewards, $p\left[\mathcal{R}^a|\theta\right]$
  - e.g., Gaussians: $\theta = [\mu(a_1), \sigma^2(a_1), ..., \mu(a_{|\mathcal{A}|}), \sigma^2(a_{|\mathcal{A}|})]$
- Compute posterior distribution over θ

$$p\left[\theta|H_t\right] \quad \propto \quad p\left[H_t|\theta\right] p\left[\theta\right]$$

- Allows us to inject rich prior knowledge $p\left[\theta\right]$
- Use posterior to guide exploration
  - Upper confidence bounds
  - Probability matching
- Better performance if prior is accurate

Bayesian Bandits with Upper Confidence Bounds



- Compute posterior distribution over action-values

$$p\left[q(a)|H_{t-1}\right] = \int_\theta p\left[q(a)|\theta\right] p\left[\theta|H_{t-1}\right] \, \mathrm{d}\theta$$

- Estimate upper confidence from posterior, e.g., $U_t(a) = c\sigma_t(a)$
  - where $\sigma(a)$ is std dev of $p(q(a) \mid \theta)$
- 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

$$\pi_t(a) = \mathbb{P}\left[q(a) = \max_{a'} q(a') \mid H_{t-1}\right]$$

- 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 $Q_t(a) \sim p\left[q(a)|H_{t-1}\right], \forall a$
  - Select action maximising sample, $a_t = \underset{a \in \mathcal{A}}{\text{argmax}}\ Q_t(a)$
- Thompson sampling is sample-based probability matching

$$\pi_t(a) = \mathbb{E}\left[\mathcal{I}(Q_t(a) = \max_{a'} Q_t(a')) \mid H_{t-1}\right]$$
$$= \mathbb{P}\left[q(a) = \max_{a'} q(a') \mid H_{t-1}\right]$$

- For Bernoulli bandits, Thompson sampling achieves Lai and Robbins lower bound on regret!

Value of Information

- Exploration is valuable because information is valuable
- Can we quantify the value of information?
- Information gain is higher in uncertain situations
- Therefore it makes sense to explore uncertain situations more
- If we know value of information, we can trade-off exploration and exploitation *optimally*

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* $\tilde{s}$ summarising all information accumulated so far
- Each action a causes a transition to a new information state $\tilde{s}'$ (by adding information), with probability $\mathcal{P}^a_{\tilde{s},\tilde{s}'}$
- We then have a Markov decision problem
- Here states = observations = internal information state

Example: Bernoulli Bandits

- Consider a Bernoulli bandit, such that

$$\mathbb{P}[R_t = 1 \mid A_t = a] = \mu_a$$
$$\mathbb{P}[R_t = 0 \mid A_t = a] = 1 - \mu_a$$

- 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 $\tilde{s} = \langle \alpha, \beta \rangle$
  - 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 have formulated the bandit as an infinite MDP over information states
- 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)
- Latter approach is known as *Bayes-adaptive* RL
- Finds Bayes-optimal exploration/exploitation trade-off with respect to prior distribution

Contextual Bandits

- Lets bring back external observations
- In bandits, this is often called context
- A contextual bandit is a tuple $\langle \mathcal{A}, \mathcal{C}, \mathcal{R} \rangle$
- $\mathcal{A}$ is a known set of actions (or "arms")
- $\mathcal{C} = \mathbb{P}[s]$ is an unknown distribution over states
- At each step t
  - Environment generates state $S_t \sim \mathcal{C}$
  - Agent selects action $A_t \in \mathcal{A}$
  - Environment generates reward $R_t \sim \mathcal{R}_{s_t}^{a_t}$
- Goal is to maximise cumulative reward $\sum_{i=1}^{t} R_i$
- Actions do not affect state!

Linear Regression

- Action-value is expected reward for state s and action a

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

- Suppose we have feature vectors $\phi_t \equiv \phi(S_t)$, where $\phi : \mathcal{S} \to \mathbb{R}^n$
- We can estimate value function with a linear approximation

$$\hat{q}(s, a; \{\theta(a)\}_{a \in \mathcal{A}}) \;=\; \phi(s)^\top \theta(a) \;\approx\; q(s, a)$$

- Estimate parameters by least squares regression

Linear Regression

Estimate parameters by least squares regression...

$$\theta_*(a) = \operatorname*{argmin}_\theta \mathbb{E}\left[\left(q(S_t, a) - \phi_t^\top \theta\right)^2\right]$$

$$\Rightarrow \theta_*(a) = \mathbb{E}\left[\phi_t \phi_t^\top | A_t = a\right]^{-1} \mathbb{E}\left[\phi_t R_t | A_t = a\right]$$

$$\Sigma_t(a) = \sum_{i=1}^{t} \mathcal{I}(A_i = a)\phi_i \phi_i^\top \qquad \text{(feature statistics)}$$

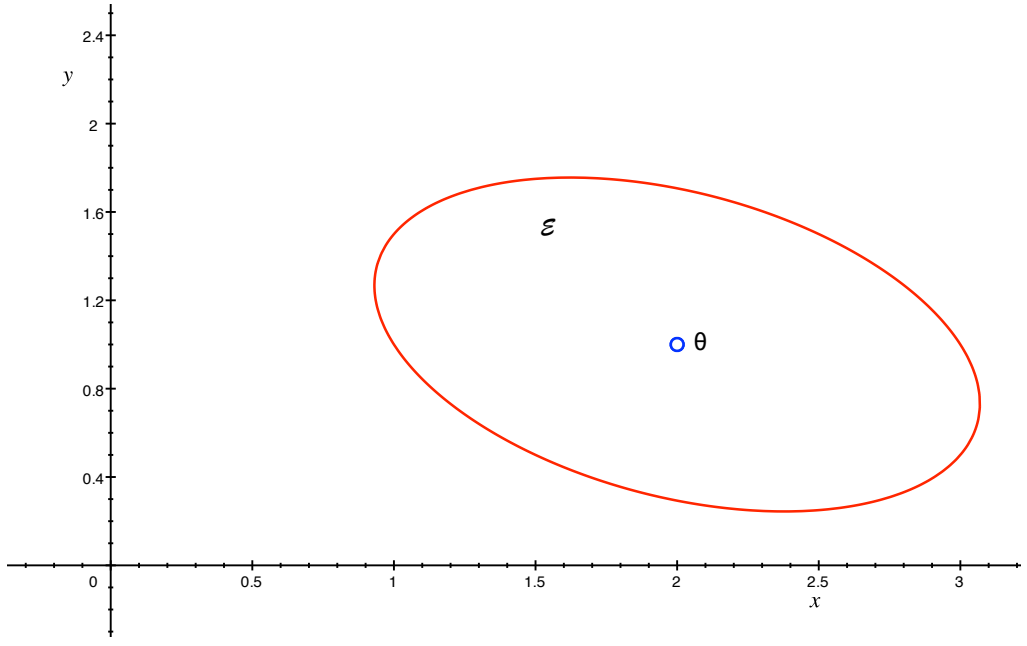
$$b_t(a) = \sum_{i=1}^{t} \mathcal{I}(A_i = a)\phi_i R_i \qquad \text{(reward statistics)}$$

$$\theta_t(a) = \Sigma_t(a)^{-1} b_t(a)$$

Linear Upper Confidence Bounds

- Least squares regression estimates the mean
- Can also estimate the value uncertainty due to parameter estimation error $\sigma^2(s, a; \theta)$
- Can use as uncertainty bonus: $U_\theta(s, a) = c\sigma(s, a; \theta)$
- i.e. define UCB to be c standard deviations above the mean

Geometric Interpretation



- Define confidence ellipsoid $\mathcal{E}_t$ that includes true parameters θ_* with high probability
- Use this to estimate the uncertainty of action values
- Pick parameters within ellipsoid that maximize action value

$$\underset{\theta \in \mathcal{E}}{\text{argmax}}\ \hat{q}(s, a; \theta)$$

Calculating Linear Upper Confidence Bounds

- For least squares regression, parameter covariance is $\Sigma_t(a)^{-1}$
- Action-value is linear in features: $\hat{q}(s, a; \theta) = \phi(s)^\top \theta_a$
- So action-value variance is quadratic,

$$\sigma_\theta^2(s, a) = \phi(s)^\top \Sigma_t(a)^{-1} \phi(s)$$

- Upper confidence bound is $\hat{q}(s, a; \theta) + c\sigma(s, a; \theta)$
- Select action maximising upper confidence bound

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\ \hat{q}(S_t, a; \theta) + c\sigma(s, a; \theta)$$

Gradient bandits

- What about learning policies $\pi(a) = \mathbb{P}[A_t = a]$ directly?
- For instance, define action preferences $Y_t(a)$ and the use

$$\pi(a) = \frac{e^{Y_t(a)}}{\sum_b e^{Y_t(b)}} \qquad \text{(soft max)}$$

- The preferences do not have to have semantics of cumulative rewards
- Instead, view them as tunable parameters
- We can then optimize preferences

Gradient bandits

- Gradient ascent on value:

$$
\begin{aligned}
Y_{t+1}(a) &= Y_t(a) + \alpha \frac{\partial \mathbb{E}\left[R_t | \pi_t\right]}{\partial Y_t(a)} \\
&= Y_t(a) + \alpha \frac{\partial}{\partial Y_t(a)} \sum_a \pi_t(a) q(a) \\
&= Y_t(a) + \alpha \sum_a q(a) \frac{\partial \pi_t(a)}{\partial Y_t(a)} \\
&= Y_t(a) + \alpha \sum_a \pi(a) q(a) \frac{\partial \log \pi_t(a)}{\partial Y_t(a)} \\
&= Y_t(a) + \alpha \mathbb{E}\left[R_t \frac{\partial \log \pi_t(a)}{\partial Y_t(a)}\right]
\end{aligned}
$$

Gradient bandits

- For soft max:

$$\begin{aligned} Y_{t+1}(a) &= Y_t(a) + \alpha \mathbb{E}\left[R_t \frac{\partial \log \pi_t(a)}{\partial Y_t(a)}\right] \\ &= Y_t(a) + \alpha \mathbb{E}\left[R_t(\mathcal{I}(a = A_t) - \pi_t(a))\right] \end{aligned}$$

- $\Rightarrow$

$$\begin{aligned} Y_{t+1}(a) &= Y_t(a) + \alpha R_t(1 - \pi_t(a)) && \text{if } a = A_t \\ Y_{t+1}(a) &= Y_t(a) - \alpha R_t \pi_t(a) && \text{if } a \neq A_t \end{aligned}$$

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

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