Multi-arm Bandits

Sutton and Barto, Chapter 2

The simplest reinforcement learning problem

Regret

- The *action-value* is the mean reward for action a,

$$Q(a) = \mathbb{E}\left[r | a\right]$$

- The *optimal value* V^* is

$$V^* = Q(a^*) = \max_{a \in \mathcal{A}} Q(a)$$

- The *regret* is the opportunity loss for one step

$$l_t = \mathbb{E}\left[V^* - Q(a_t)\right]$$

- The *total regret* is the total opportunity loss

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

- Maximise cumulative reward $\equiv$ minimise total regret

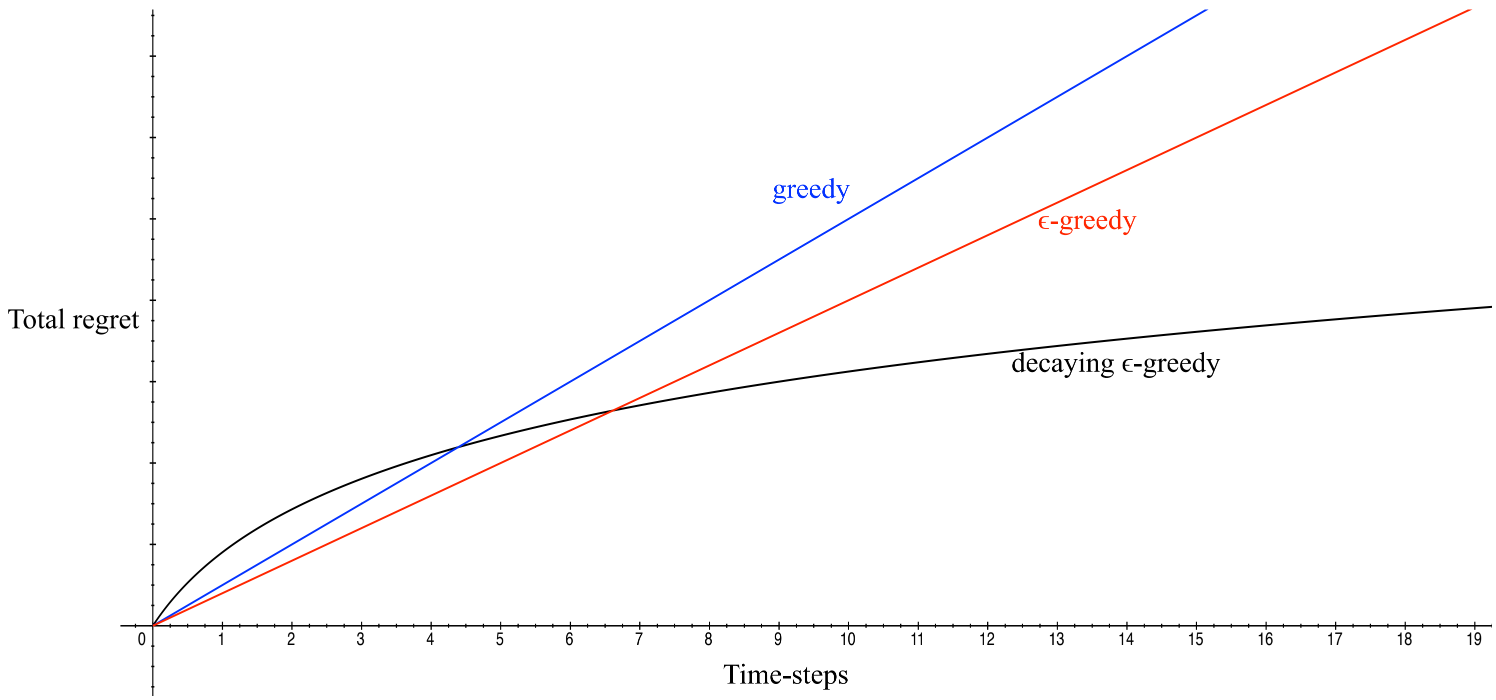
Counting Regret

- The *count* $N_t(a)$ is expected number of selections for action a
- The *gap* Δ_a is the difference in value between action a and optimal action a^*, $\Delta_a = V^* - Q(a)$
- Regret is a function of gaps and the counts

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

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

Linear or Sublinear Regret



- If an algorithm forever explores it will have linear total regret
- If an algorithm never explores it will have linear total regret
- Is it possible to achieve sublinear total regret?

Greedy Algorithm

- We consider algorithms that estimate $\hat{Q}_t(a) \approx Q(a)$
- Estimate the value of each action by Monte-Carlo evaluation

$$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{t=1}^{T} r_t \mathbf{1}(a_t = a)$$

- The *greedy* algorithm selects action with highest value

$$a_t^* = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\ \hat{Q}_t(a)$$

- Greedy can lock onto a suboptimal action forever
- $\Rightarrow$ Greedy has linear total regret

ϵ-Greedy Algorithm

- The ϵ-*greedy* algorithm continues to explore forever
  - With probability $1 - \epsilon$ select $a = \underset{a \in \mathcal{A}}{\text{argmax}}\ \hat{Q}(a)$
  - With probability ϵ select a random action
- Constant ϵ ensures minimum regret

$$l_t \geq \frac{\epsilon}{\mathcal{A}} \sum_{a \in \mathcal{A}} \Delta_a$$

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

Optimistic Initialisation

- Simple and practical idea: initialise $Q(a)$ to high value
- Update action value by incremental Monte-Carlo evaluation
- Starting with $N(a) > 0$

$$\hat{Q}_t(a_t) = \hat{Q}_{t-1} + \frac{1}{N_t(a_t)}(r_t - \hat{Q}_{t-1})$$

- Encourages systematic exploration early on
- But can still lock onto suboptimal action
- $\Rightarrow$ greedy + optimistic initialisation has linear total regret
- $\Rightarrow$ ϵ-greedy + optimistic initialisation has linear total regret

Decaying ϵ_t-Greedy Algorithm

- Pick a decay schedule for $\epsilon_1, \epsilon_2, \ldots$
- 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, schedule requires advance knowledge of gaps
- Goal: find an algorithm with sublinear regret for any multi-armed bandit (without knowledge of $\mathcal{R}$)

Lower Bound

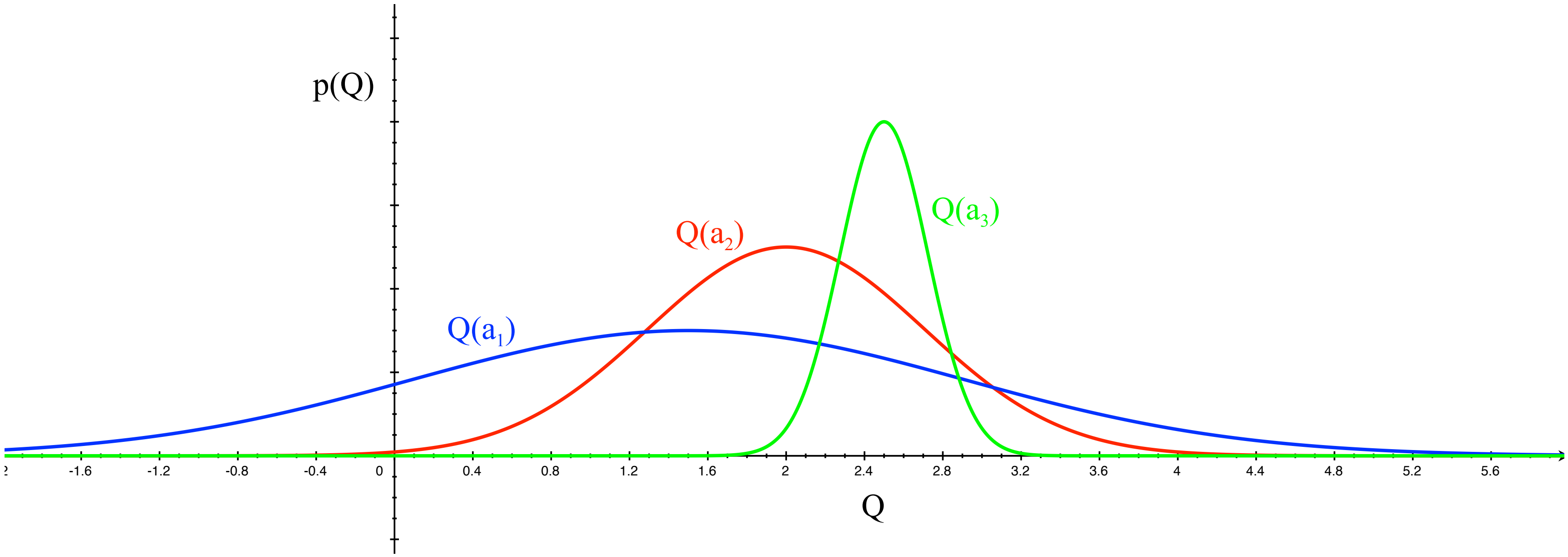
- The performance of any algorithm is determined by similarity between optimal arm and other arms
- Hard problems have similar-looking arms with different means
- This is described formally by the gap Δ_a and the similarity in distributions $KL(\mathcal{R}^a || \mathcal{R}^a *)$

Theorem (Lai and Robbins)

Asymptotic total regret is at least logarithmic in number of steps

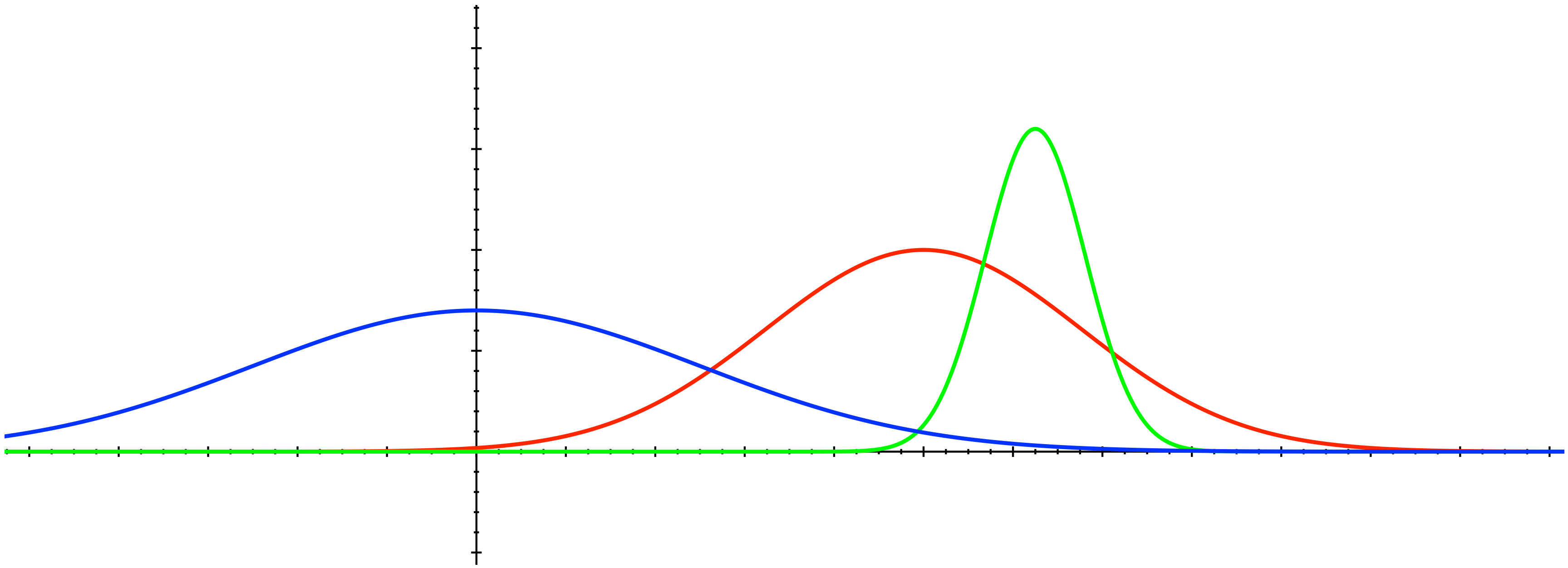
$$\lim_{t \to \infty} L_t \geq \log t \sum_{a | \Delta_a > 0} \frac{\Delta_a}{KL(\mathcal{R}^a || \mathcal{R}^{a^*})}$$

Optimism in the Face of Uncertainty



- Which action should we pick?
- The more uncertain we are about an action-value
- The more important it is to explore that action
- It could turn out to be the best action

Optimism in the Face of Uncertainty (2)



- After picking blue action
- We are less uncertain about the value
- And more likely to pick another action
- Until we home in on best action

Upper Confidence Bounds

- Estimate an upper confidence $\hat{U}_t(a)$ for each action value
- Such that $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with high probability
- This depends on the number of times $N(a)$ has been selected
  - Small $N_t(a) \Rightarrow$ large $\hat{U}_t(a)$ (estimated value is uncertain)
  - Large $N_t(a) \Rightarrow$ small $\hat{U}_t(a)$ (estimated value is accurate)
- Select action maximising Upper Confidence Bound (UCB)

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \; \hat{Q}_t(a) + \hat{U}_t(a)$$

Hoeffding's Inequality

Theorem (Hoeffding's Inequality)

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

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

- We will apply Hoeffding's Inequality to rewards of the bandit
- conditioned on selecting action a

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

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 $t \rightarrow \infty$

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

UCB1

- This leads to the UCB1 algorithm

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

Theorem

The UCB algorithm achieves logarithmic asymptotic total regret

$$\lim_{t \to \infty} L_t \leq 8 \log t \sum_{a \mid \Delta_a > 0} \Delta_a$$

Gradient-Bandit Algorithms

- Let $H_t(a)$ be a learned *preference* for taking action a

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

Note that this allows us to work with unnormalized preferences and turn them into probabilities!

Same idea as using potentials in graphical models

Gradient-Bandit Algorithms

- Let $H_t(a)$ be a learned *preference* for taking action a

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha\big(R_t - \bar{R}_t\big)\big(1 - \pi_t(A_t)\big)$$

$$\bar{R}_t \doteq \frac{1}{t}\sum_{i=1}^{t} R_i$$

Gradient-Bandit Algorithms

- Let $H_t(a)$ be a learned *preference* for taking action a

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

$$H_{t+1}(a) \doteq H_t(a) + \alpha\big(R_t - \bar{R}_t\big)\big(\mathbf{1}_{a=A_t} - \pi_t(a)\big), \qquad \forall a,$$

$$\bar{R}_t \doteq \frac{1}{t}\sum_{i=1}^{t} R_i$$

Gradient-Bandit Algorithms

- Let $H_t(a)$ be a learned *preference* for taking action a

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

$$H_{t+1}(a) \doteq H_t(a) + \alpha\big(R_t - \bar{R}_t\big)\big(\mathbf{1}_{a=A_t} - \pi_t(a)\big), \qquad \forall a,$$

$$\bar{R}_t \doteq \frac{1}{t}\sum_{i=1}^{t} R_i$$

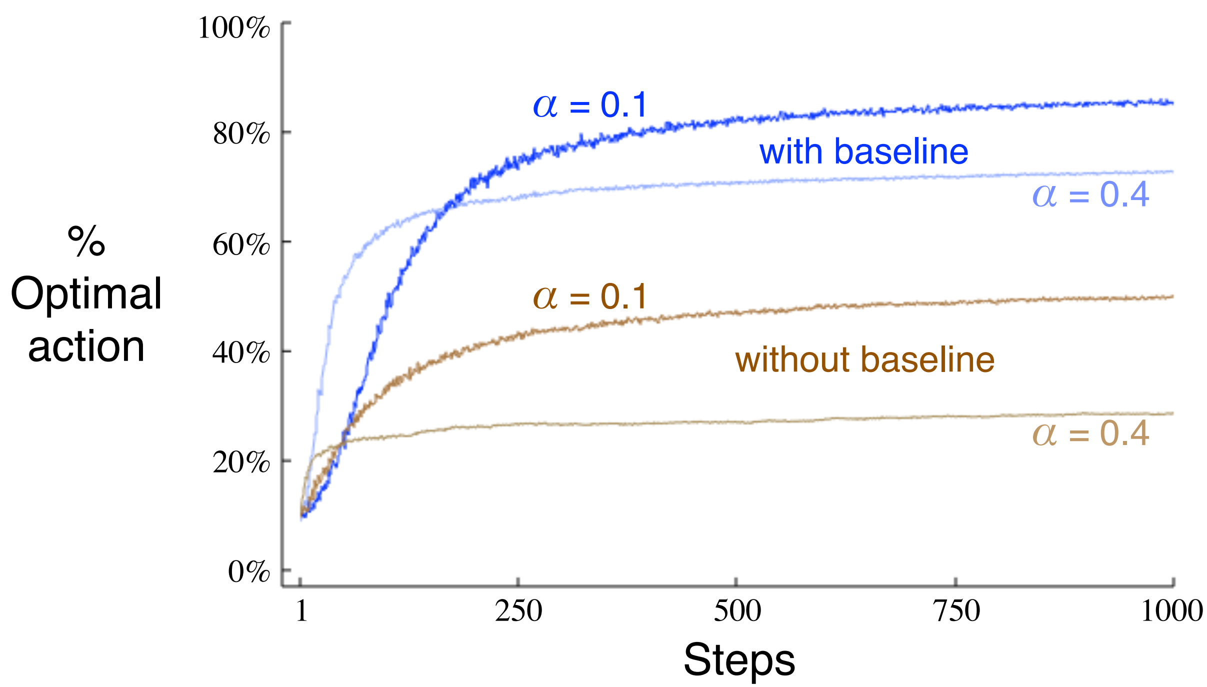
Gradient-Bandit Algorithms

- Let $H_t(a)$ be a learned *preference* for taking action a

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

$$H_{t+1}(a) \doteq H_t(a) + \alpha\big(R_t - \bar{R}_t\big)\big(\mathbf{1}_{a=A_t} - \pi_t(a)\big), \qquad \forall a,$$

$$\bar{R}_t \doteq \frac{1}{t}\sum_{i=1}^{t} R_i$$



Derivation of gradient-bandit algorithm

In exact *gradient ascent*:

$$H_{t+1}(a) \doteq H_t(a) + \alpha \frac{\partial \, \mathbb{E}\,[R_t]}{\partial H_t(a)}, \tag{1}$$

where:

$$\mathbb{E}[R_t] \doteq \sum_b \pi_t(b) q_*(b),$$

$$\begin{aligned}
\frac{\partial \, \mathbb{E}[R_t]}{\partial H_t(a)} &= \frac{\partial}{\partial H_t(a)} \left[\sum_b \pi_t(b) q_*(b) \right] \\
&= \sum_b q_*(b) \frac{\partial \, \pi_t(b)}{\partial H_t(a)} \\
&= \sum_b \left(q_*(b) - X_t \right) \frac{\partial \, \pi_t(b)}{\partial H_t(a)},
\end{aligned}$$

where X_t does not depend on b, because $\sum_b \frac{\partial \, \pi_t(b)}{\partial H_t(a)} = 0$.

$$
\begin{aligned}
\frac{\partial\, \mathbb{E}[R_t]}{\partial H_t(a)} &= \sum_b \big(q_*(b) - X_t\big) \frac{\partial\, \pi_t(b)}{\partial H_t(a)} \\
&= \sum_b \pi_t(b) \big(q_*(b) - X_t\big) \frac{\partial\, \pi_t(b)}{\partial H_t(a)} / \pi_t(b) \\
&= \mathbb{E}\left[\big(q_*(A_t) - X_t\big) \frac{\partial\, \pi_t(A_t)}{\partial H_t(a)} / \pi_t(A_t) \right] \\
&= \mathbb{E}\left[\big(R_t - \bar{R}_t\big) \frac{\partial\, \pi_t(A_t)}{\partial H_t(a)} / \pi_t(A_t) \right],
\end{aligned}
$$

where here we have chosen $X_t = \bar{R}_t$ and substituted R_t for $q_*(A_t)$, which is permitted because $\mathbb{E}[R_t | A_t] = q_*(A_t)$.

For now assume: $\frac{\partial\, \pi_t(b)}{\partial H_t(a)} = \pi_t(b)\big(\mathbf{1}_{a=b} - \pi_t(a)\big)$. Then:

$$
\begin{aligned}
&= \mathbb{E}\big[\big(R_t - \bar{R}_t\big) \pi_t(A_t) \big(\mathbf{1}_{a=A_t} - \pi_t(a)\big) / \pi_t(A_t) \big] \\
&= \mathbb{E}\big[\big(R_t - \bar{R}_t\big) \big(\mathbf{1}_{a=A_t} - \pi_t(a)\big) \big].
\end{aligned}
$$

$$
H_{t+1}(a) = H_t(a) + \alpha \big(R_t - \bar{R}_t\big) \big(\mathbf{1}_{a=A_t} - \pi_t(a)\big), \text{ (from (1), QED)}
$$

Thus it remains only to show that

$$\frac{\partial \pi_t(b)}{\partial H_t(a)} = \pi_t(b)\big(\mathbf{1}_{a=b} - \pi_t(a)\big).$$

Recall the standard quotient rule for derivatives:

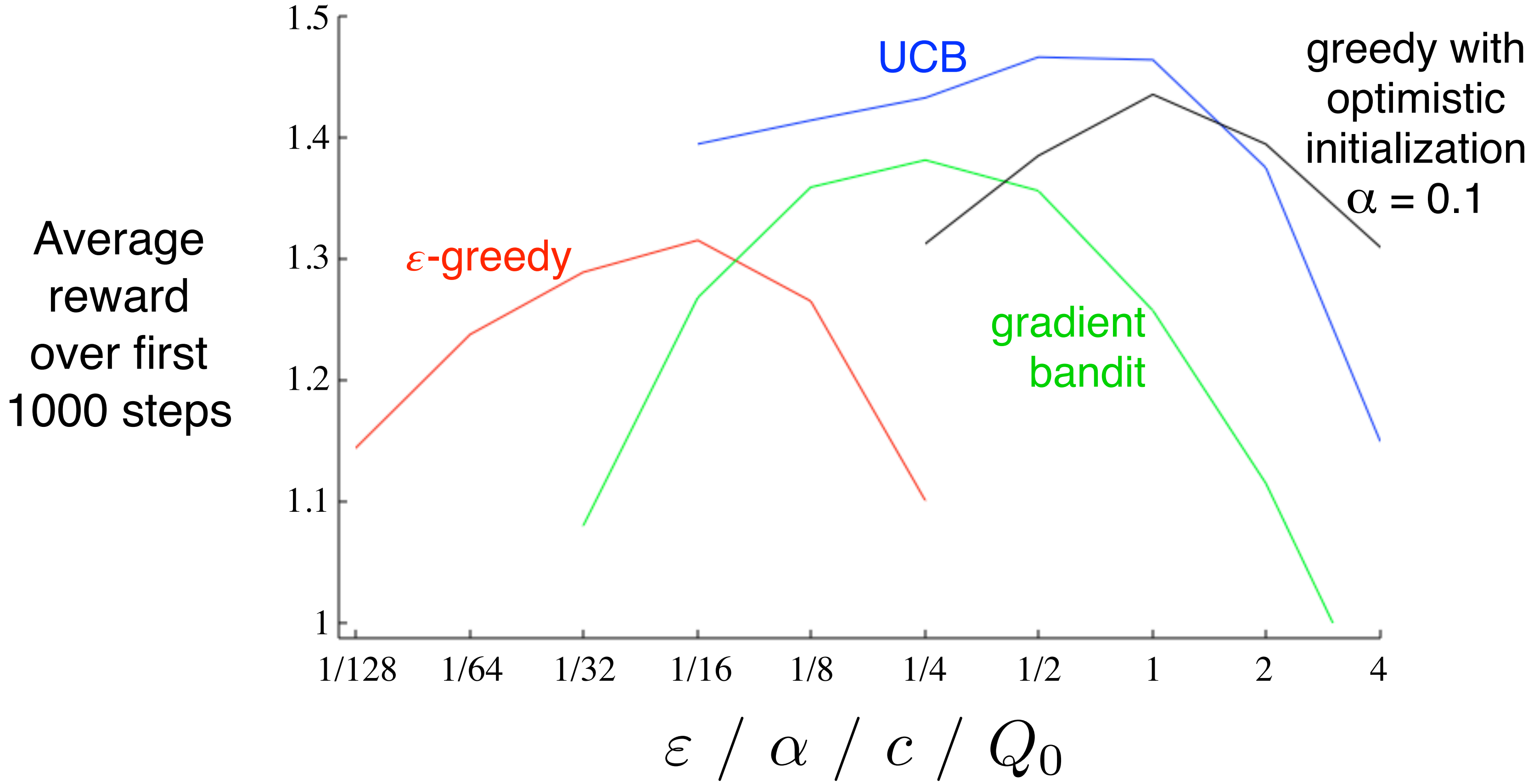
$$\frac{\partial}{\partial x}\left[\frac{f(x)}{g(x)}\right] = \frac{\frac{\partial f(x)}{\partial x} g(x) - f(x)\frac{\partial g(x)}{\partial x}}{g(x)^2}.$$

Using this, we can write...

$$\text{Quotient Rule:} \quad \frac{\partial}{\partial x}\left[\frac{f(x)}{g(x)}\right] = \frac{\frac{\partial f(x)}{\partial x} g(x) - f(x)\frac{\partial g(x)}{\partial x}}{g(x)^2}$$

$$\begin{aligned}
\frac{\partial\, \pi_t(b)}{\partial H_t(a)} &= \frac{\partial}{\partial H_t(a)}\, \pi_t(b) \\
&= \frac{\partial}{\partial H_t(a)} \left[\frac{e^{H_t(b)}}{\sum_{c=1}^{k} e^{H_t(c)}}\right] \\
&= \frac{\frac{\partial e^{H_t(b)}}{\partial H_t(a)} \sum_{c=1}^{k} e^{H_t(c)} - e^{H_t(b)} \frac{\partial \sum_{c=1}^{k} e^{H_t(c)}}{\partial H_t(a)}}{\left(\sum_{c=1}^{k} e^{H_t(c)}\right)^2} && \text{(Q.R.)} \\
&= \frac{\mathbf{1}_{a=b} e^{H_t(a)} \sum_{c=1}^{k} e^{H_t(c)} - e^{H_t(b)} e^{H_t(a)}}{\left(\sum_{c=1}^{k} e^{H_t(c)}\right)^2} && \left(\frac{\partial e^x}{\partial x} = e^x\right) \\
&= \frac{\mathbf{1}_{a=b} e^{H_t(b)}}{\sum_{c=1}^{k} e^{H_t(c)}} - \frac{e^{H_t(b)} e^{H_t(a)}}{\left(\sum_{c=1}^{k} e^{H_t(c)}\right)^2} \\
&= \mathbf{1}_{a=b} \pi_t(b) - \pi_t(b)\pi_t(a) \\
&= \pi_t(b)\big(\mathbf{1}_{a=b} - \pi_t(a)\big). && \text{(Q.E.D.)}
\end{aligned}$$

Summary Comparison of Bandit Algorithms



Discussion

- These are all simple methods
  - but they are complicated enough—we will build on them
  - we should understand them completely
  - there are still open questions
- Our first algorithms that learn from evaluative feedback
  - and thus must balance exploration and exploitation
- Our first algorithms that appear to have a goal
  —that learn to maximize reward by trial and error

Our first dimensions!

- Problems vs Solution Methods
- Evaluative vs Instructive
- Associative vs Non-associative

Problem space

	Single State	Associative
Instructive feedback		
Evaluative feedback		

Problem space

	Single State	Associative
Instructive feedback		
Evaluative feedback	Bandits (Function optimization)	

Problem space

	Single State	Associative
Instructive feedback		Supervised learning
Evaluative feedback	Bandits (Function optimization)	

Problem space

	Single State	Associative
Instructive feedback	Averaging	Supervised learning
Evaluative feedback	Bandits (Function optimization)	

Problem space

	Single State	Associative
Instructive feedback	Averaging	Supervised learning
Evaluative feedback	Bandits (Function optimization)	Associative Search (Contextual bandits)