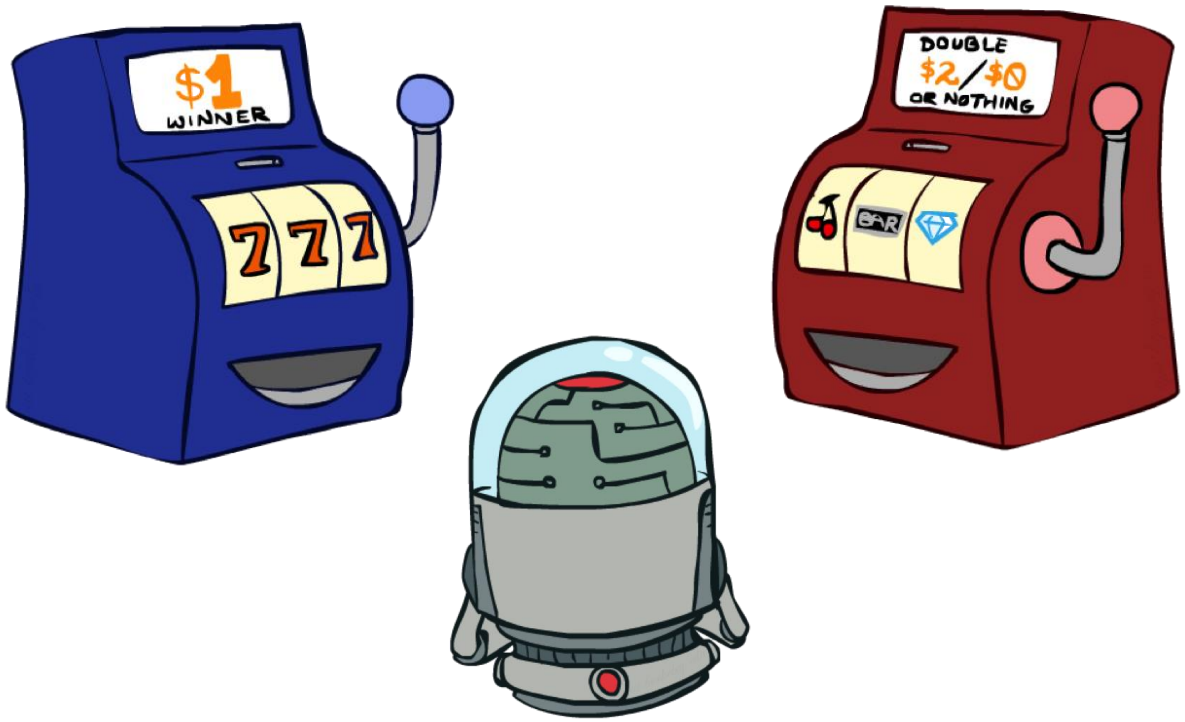
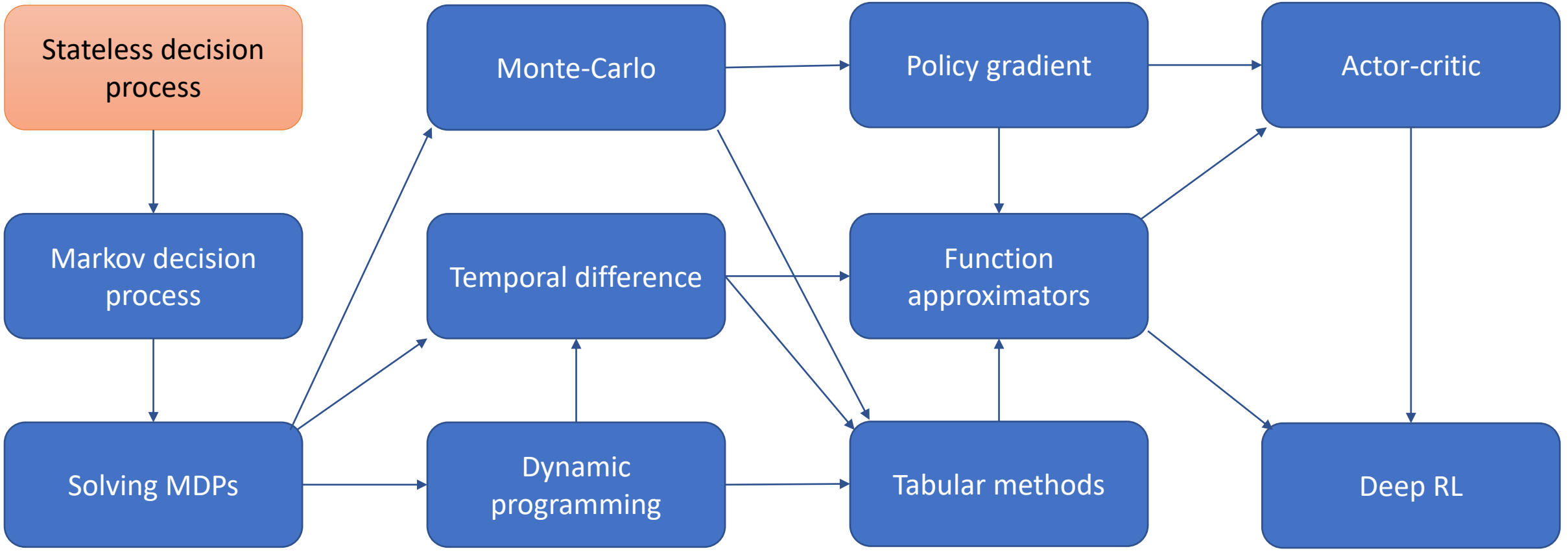
CSCE-642 Reinforcement Learning
Chapter 2: Multi-armed Bandit



Instructor: Guni Sharon

CSCE-689, Reinforcement Learning



Optimal action

$\mathbb{E}[a] = -\$0.5$

$\mathbb{E}[b] = -\$0.2$

$\mathbb{E}[c] = \$0.1$

$\mathbb{E}[d] = \$0.11$

Learn the optimal action

$\mathbb{E}[a] = ?$

$\mathbb{E}[b] = ?$

$\mathbb{E}[c] = ?$

$\mathbb{E}[d] = ?$

Learn the optimal action

-1, -1, 5

$\widehat{\mathbb{E}}[a] = 1$

-0.2, -0.2

$\widehat{\mathbb{E}}[b] = -0.2$

-0.5, -0.5, -0.5

$\widehat{\mathbb{E}}[c] = -0.5$

-2, -2

$\widehat{\mathbb{E}}[d] = -2$

Learn the optimal action

-1, -1, 5, -1, -1, -1, -1, -1, 2, 6, -1, -1, -1, -1, -1

$\mathbb{E}[a] = -\$0.5$

-0.2, -0.2

$\mathbb{E}[b] = -\$0.2$

-0.5, -0.5, -0.5

$\mathbb{E}[c] = \$0.1$

-2, -2

$\mathbb{E}[d] = \$0.11$

Notation

k	Number of actions (arms)
t	Discreate time step or play count
$q_*(a)$	True value (expected reward) of action a
$Q_t(a)$	Estimate of $q_*(a)$ at time step t
$N_t(a)$	Number of times action a was selected before step t
R_t	The reward observed at time step t
A_t	The action chosen at time step t

Learn the optimal action

$k=4$

$t=11$

$R = \{-1, -1, 5\}$

$N_{11}(a) = 3$

$q_*(a) = -\$0.5$

$Q_{11}(a) = 1$

$R = \{-0.2, -0.2\}$

$N_{11}(b) = 2$

$q_*(b) = -\$0.2$

$Q_{11}(b) = -0.2$

$R = \{-0.5, -0.5, -0.5\}$

$N_{11}(c) = 3$

$q_*(c) = \$0.1$

$Q_{11}(c) = -0.5$

$R = \{-2, -2\}$

$N_{11}(d) = 2$

$q_*(d) = \$0.11$

$Q_{11}(d) = -2$

Greedy action = *Exploit* current knowledge

-1, -1, 5

$Q_{11}(a) = 1$

-0.2, -0.2

$Q_{11}(b) = -0.2$

-0.5, -0.5, -0.5

$Q_{11}(c) = -0.5$

-2, -2,

$Q_{11}(d) = -2$

Or ***Explore*** for new knowledge

-1, -1, 5

$Q_{11}(a) = 1$

-0.2, -0.2

$Q_{11}(b) = -0.2$

-0.5, -0.5, -0.5

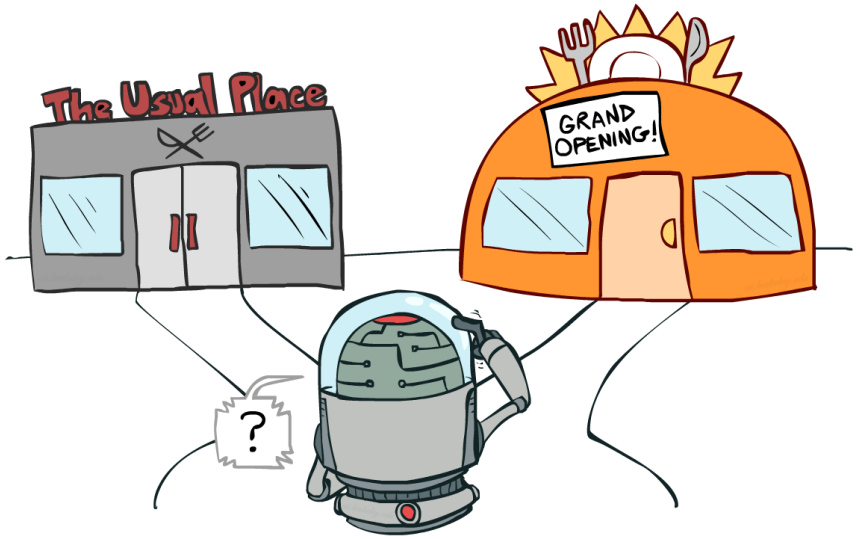
$Q_{11}(c) = -0.5$

-2, -2,

$Q_{11}(d) = -2$

Explore or exploit?

- Short term or long-term planning?
- How confident are you regarding your current knowledge?
- The answer is usually a function of the current action-value estimations, uncertainties, and planning horizon

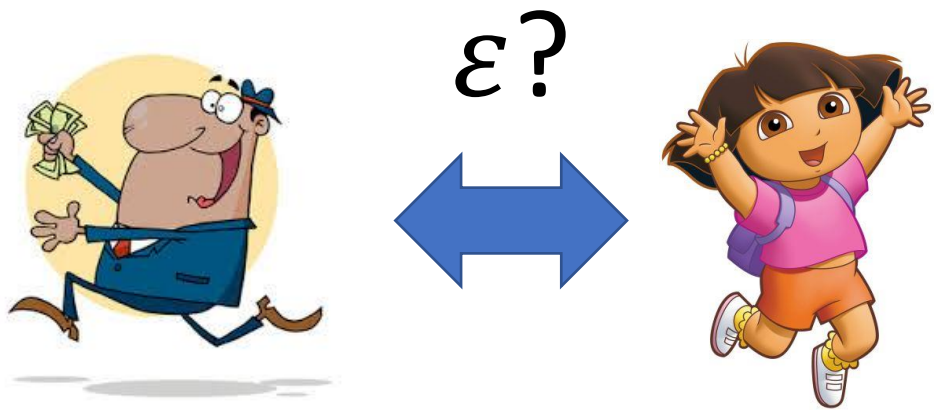


Action-value methods

- Assume a simple evaluation function: $Q_{t+1}(a) = \begin{cases} \frac{\sum_{t:A_t=a} r_t}{N_t(a)}, & N_t(a) > 0 \\ 0 \quad , & N_t(a) = 0 \end{cases}$
- What action should be chosen next?
  - Greedy action: $A_t = \text{argmax}_a Q_t(a)$
  - Exploit current knowledge to maximize immediate reward
  - Short-cited strategy
  - Never discovers more lucrative actions

ε-greedy

```
epsilon = 0.05
def get_action():
    if random.random() > epsilon:
        return argmax_a(Q(a))
    else:
        return random.choice(A)
```

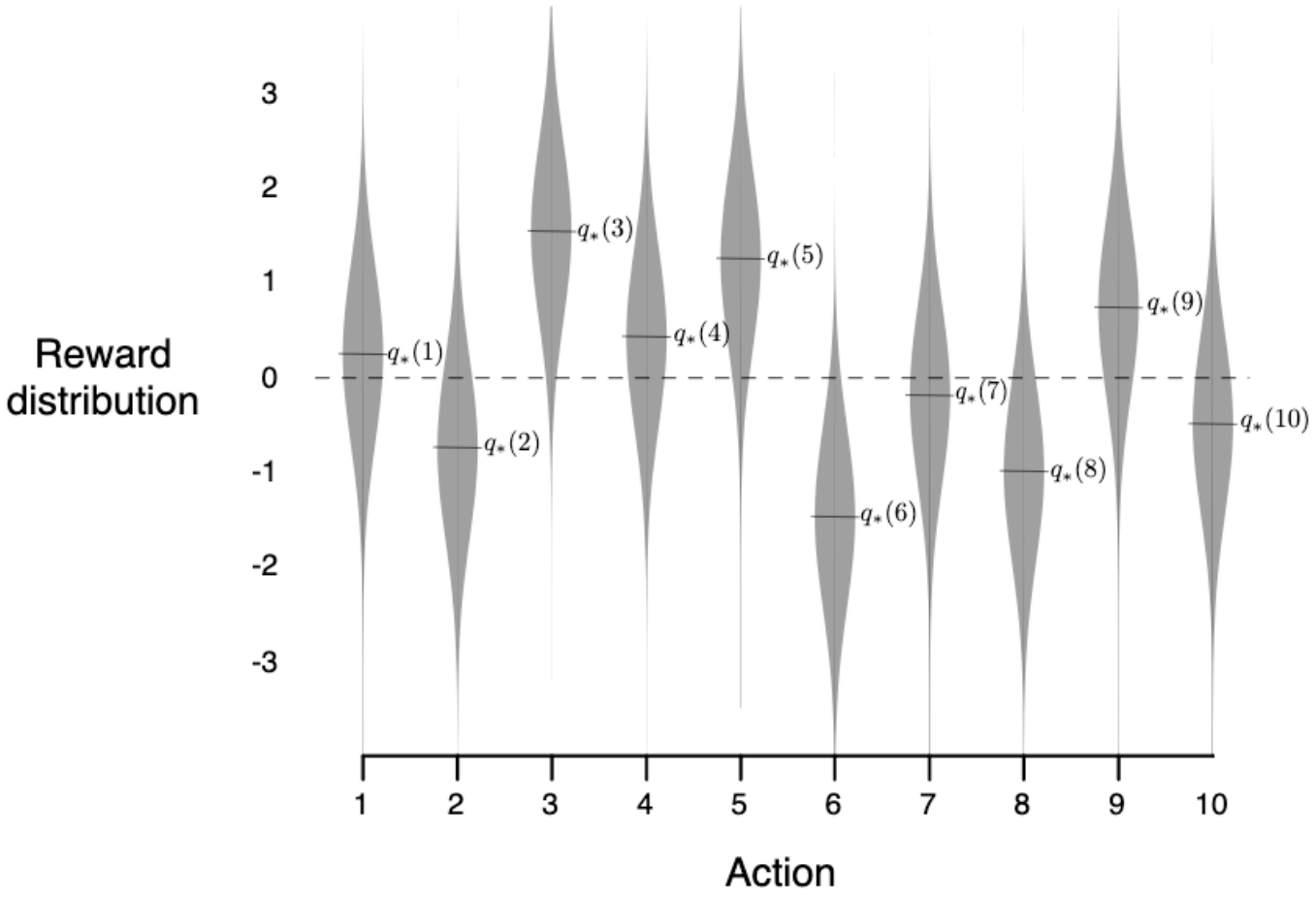


- If each action is sampled infinitely than $Q_t(a)$ will converge on $q_*(a)$

Example problem

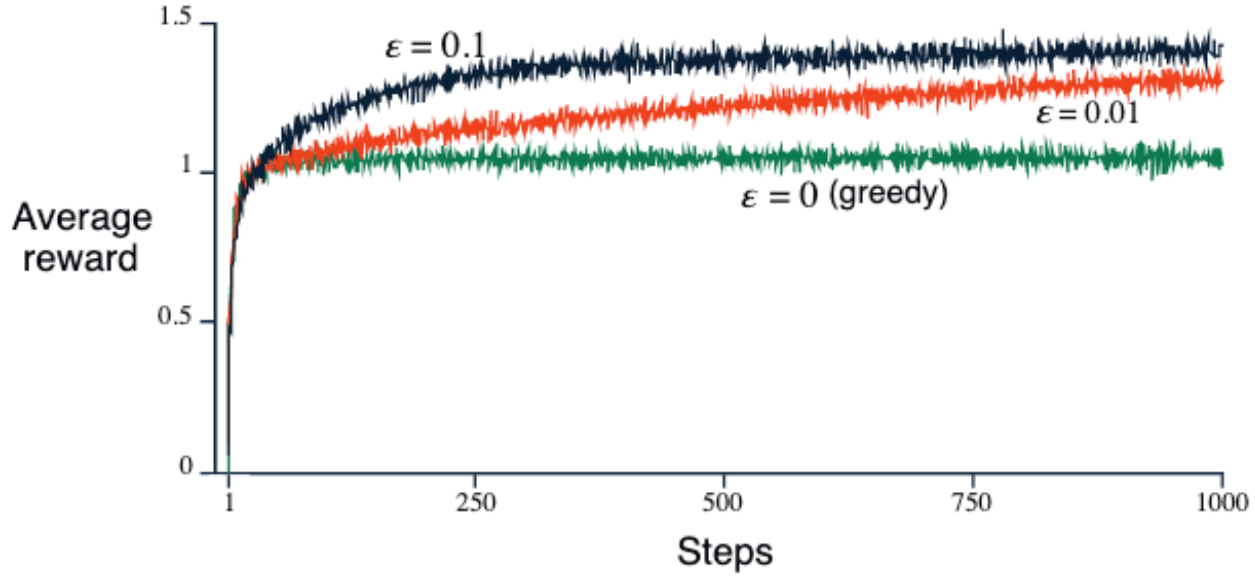
- $k = 10$
- $A = \{1, \dots, 10\}$
- $\Pr\{r|a\} \sim \mathcal{N}(q_*(a), 1)$

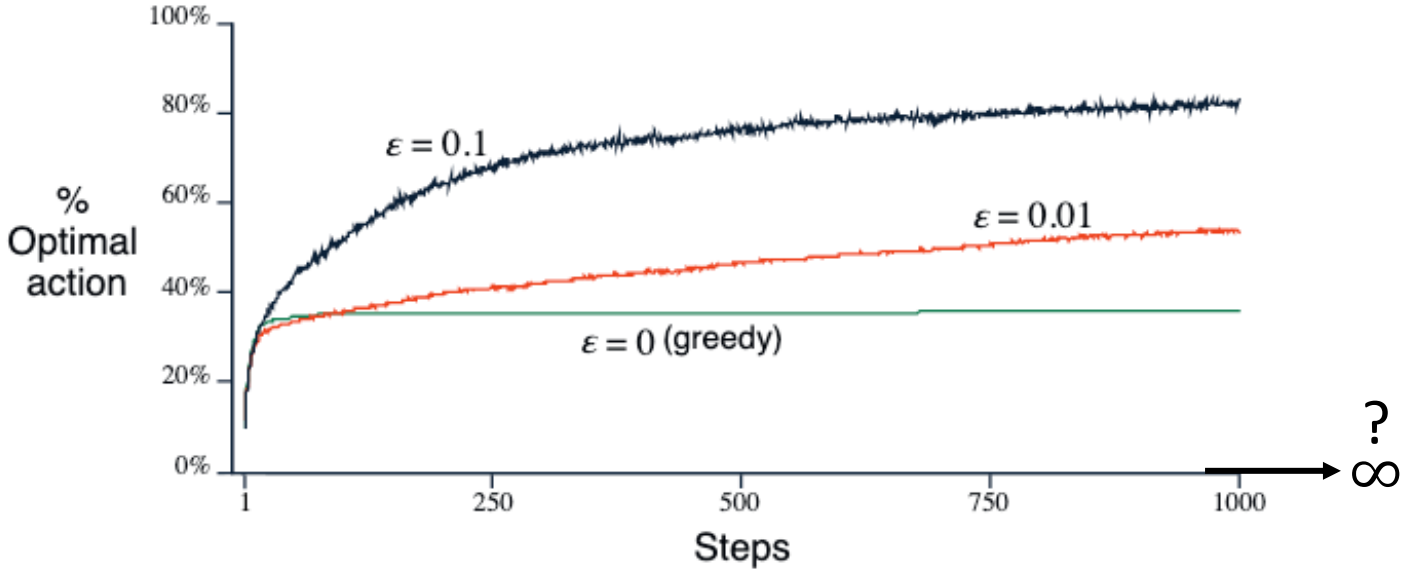
- The optimal policy is to always choose a_3
- But we don't know that initially



Results

- Average performance of ε-greedy action-value methods on the 10-armed testbed
  - $\varepsilon = 0$
  - $\varepsilon = 0.1$
  - $\varepsilon = 0.01$
- Averages over 2000 runs each with 1000 steps
- All methods used sample averages as their action-value estimates





Incremental sample averages

- sample averages: $Q_{t+1}(a) = \frac{\sum_{i=1}^{t} r_i}{N_t(a)}$
- Not memory and computational efficient, $O\left(N_t(a)\right)$
- Temporal difference:
- $Q_{t+1}(a) = \frac{1}{N_t(a)} \sum_{i=1}^{t} r_t = \frac{1}{N_t(a)} \left(r_t + \sum_{i=1}^{N_t(a)-1} r_i\right)$

$$= \frac{1}{N_t(a)} \left(r_t + (N_t(a) - 1) \frac{1}{N_t(a) - 1} \sum_{i=1}^{N_t(a)-1} r_i \right)$$

$$= \frac{1}{N_t(a)} \left(r_t + (N_t(a) - 1) Q_t(a)\right) = \frac{1}{N_t(a)} \left(r_t + N_t(a) Q_t(a) - Q_t(a)\right)$$

$$= Q_t(a) + \frac{1}{N_t(a)} \left[r_t - Q_t(a)\right]$$

- More generally:
  - $NewEstimate \leftarrow OldEstimate + w_t[Target - OldEstimate]$

Nonstationary outcomes

- Averaging over samples assumes stationary distribution of outcomes
- What if the reward probabilities change over time?

Give more weight to recent observations!

Constant

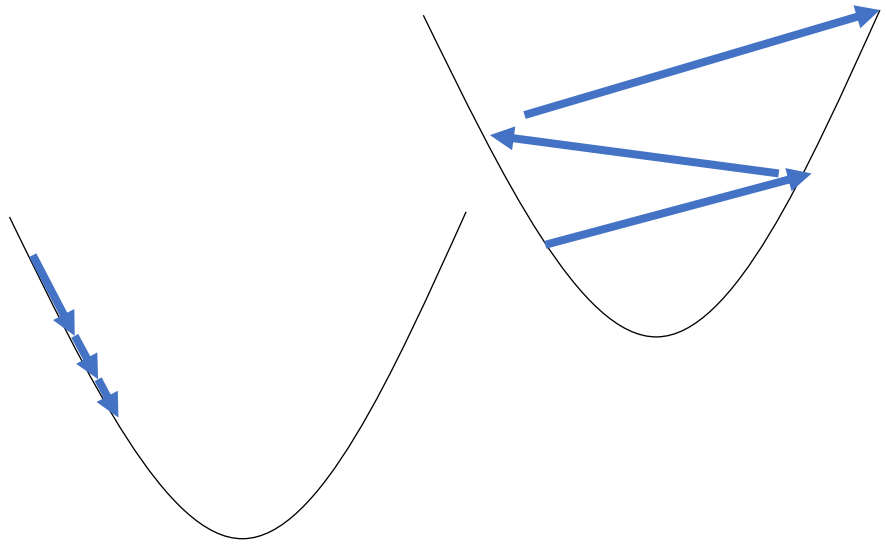
- $Q_{t+1}(a) = Q_t(a) + \alpha\ [r_t - Q_t(a)]$ $\quad 0 < \alpha < 1$

  $= \alpha r_t + (1-\alpha)\alpha r_{t-1} + (1-\alpha)^2 \alpha r_{t-2} + \cdots + (1-\alpha)^{t-1}\alpha r_1 + +(1-\alpha)^t Q_1(a)$

- A.K.A., Exponential decay

Setting the step size α

- The learning rate cannot be too large
  - Won't converge
- Or to small
  - Premature convergence
- Good practice: Start large and decrease
  - E.g., $\alpha_t(a) = \frac{1}{N_t(a)}$ (results in sample averages)
- Conditions that guarantee convergance to optimum (global if assuming convexity)
  - $\sum_t \alpha_t(a) = \infty$
  - $\sum_t \alpha_t^2(a) < \infty$



Quiz

- Different learning rates are utilized in different applications
- Which of the following rates guarantees convergence?

$\alpha_t =$	$\sum_T \alpha_t$	$\sum_T \alpha_t^2$
$1/t^2$		
$1/t$		
$1/t^{2/3}$		
$1/t^{1/2}$		
$1/100$		

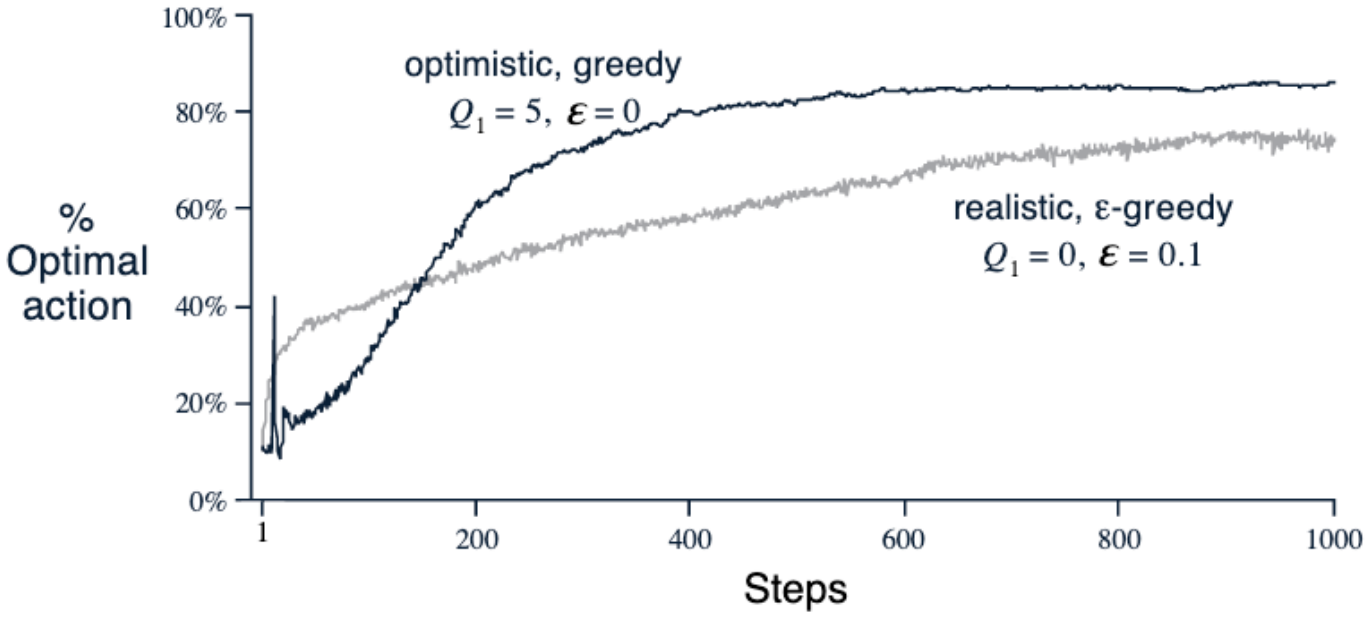
$\sum_i \frac{1}{i^c}$ converges if c > 1

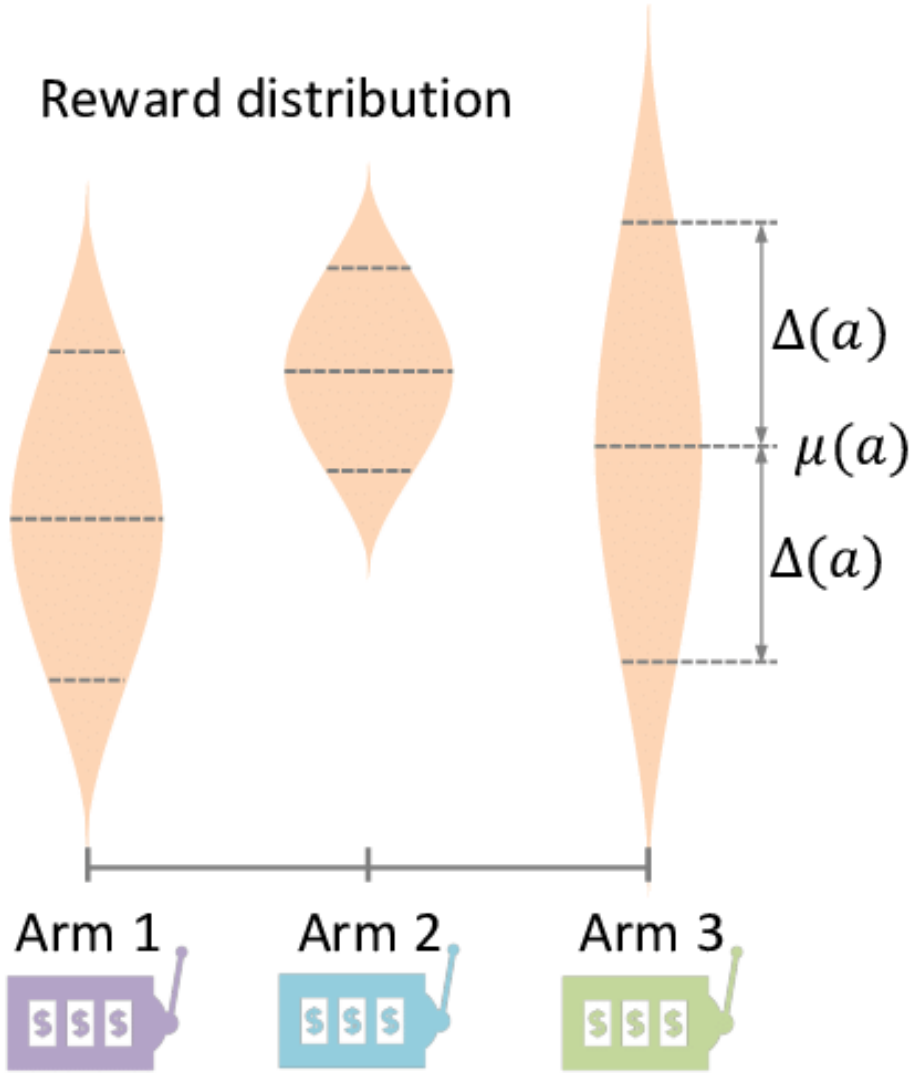
Initial Q values + greedy

- $Q_{t+1}(a) = \begin{cases} Q_t(a) + \alpha_t[r_t - Q_t(a)], & N_t(a) > 0 \\ ? \quad , & N_t(a) = 0 \end{cases}$
- What would happen if we set the default value to the minimal achievable outcome?
  - Initial values introduce bias
- What would happen if we set the default value to the maximal achievable outcome?
  - Initial values can encourage exploration!

10-bandits problem

- $k = 10$
- $A = \{1, \dots, 10\}$
- $\Pr\{r|a\} \sim \mathcal{N}(q_*(a), 1)$
- $\max[q_*(a)] = 3$
- $\alpha = 0.1$
- Setting initial value to 5 and then following greedy selection is more effective than ε-greedy

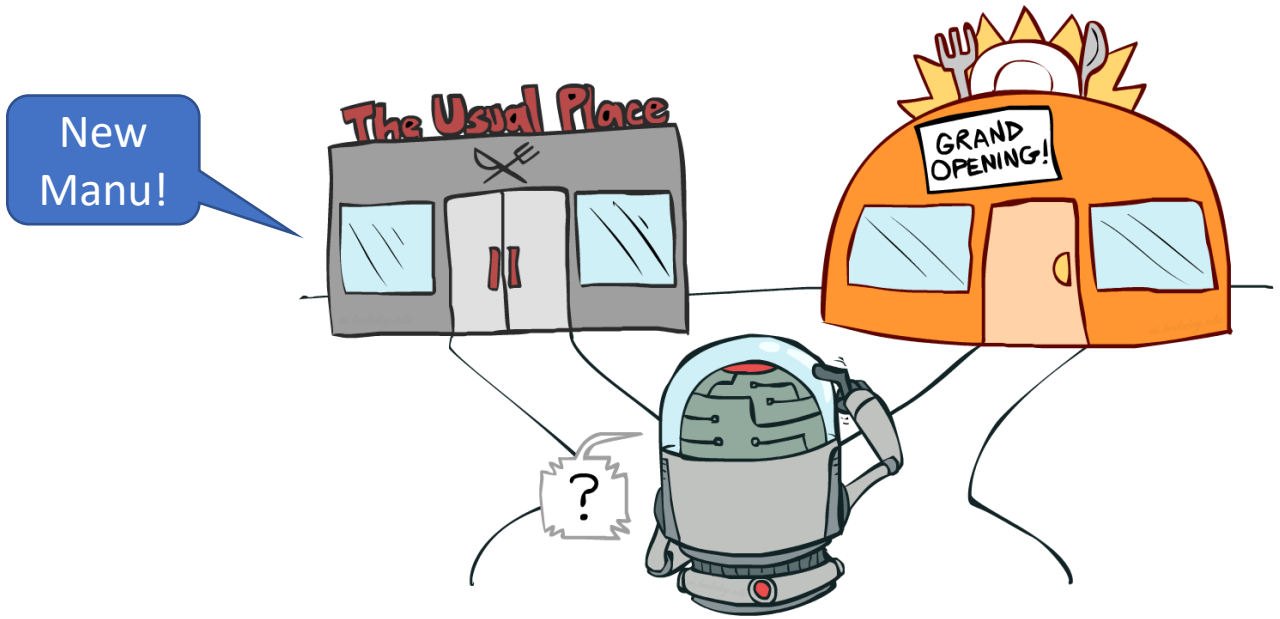



Reward distribution
$\Delta(a)$
$\mu(a)$
$\Delta(a)$
Arm 1
Arm 2
Arm 3


500m
N471
N209
Golfbaan De Rotterdamse
Lage Bergse Bos
Rotterdam The Hague Airport
Plaswijckpark
ZESTIENHOVEN
TERBREGGE
Hillegersberg
A13
A20
S109
S111
S114
Veilingterrein
Franciscus Gasthuis
OVERSCHIE
LISKWARTIER
BERGPOLDER
NOORD
KRALINGSE BOS
Crooswijk
Diergaarde Blijdorp
BLIJDORP

Optimistic initial Q values

- Powerful for stationary outcomes
- Unlikely to help with the general nonstationary case. Why?
  - Encourages exploration at early stages
  - Exploration incentive decays over time



Exploration strategy

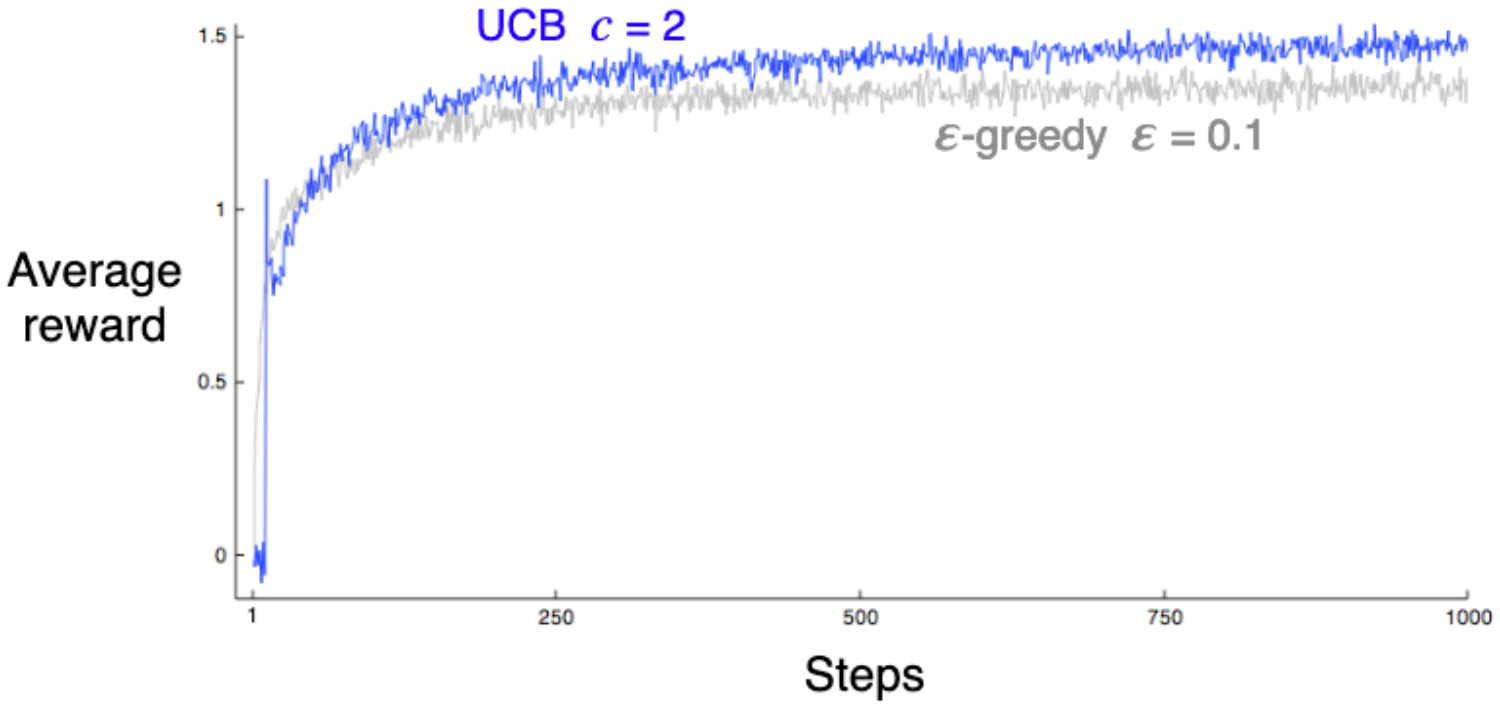
- Exploration is constantly needed as there is always uncertainty about the action-value (Q) estimates
- The greedy actions are those that look best at present, but some of the other actions may actually be better
- ε-greedy action selection forces the non-greedy actions to be tried, but indiscriminately, with no preference for those that are nearly greedy or particularly uncertain
- It would be better to select among the non-greedy actions according to their potential for actually being optimal, taking into account both how close their estimates are to being maximal and the uncertainties in those estimates

Upper-Confidence-Bounds action selection

- $A_{t+1} = \text{argmax}_a \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$
- $c > 0$ is a parameter for controlling the degree of exploration
- Provides a “bonus” in value estimation for underexplored actions
- Each time a is selected the uncertainty is presumably reduced: $N_t(a)$ increments, and, as it appears in the denominator, the uncertainty term decreases
- On the other hand, each time an action other than a is selected, t increases but $N_t(a)$ does not; because t appears in the numerator, the uncertainty bonus increases
- The use of the natural logarithm means that the increases get smaller over time, but are unbounded
- All actions will eventually be selected, but actions with lower value estimates, or that have already been selected frequently, will be selected with decreasing frequency over time.

10-bandits problem

- $k = 10$
- $A = \{1, \dots, 10\}$
- $\Pr\{r|a\} \sim \mathcal{N}(q_*(a), 1)$
- $\max[q_*(a)] = 3$
- $\alpha = 0.1$



Policy-based algorithms

- Forget about action-value (Q) estimates, we don't really care about them
- We care about what actions to chose

Let's assign a preference to each action and tweak its value

- Define $H_t(a)$ as a numerical score associated with action a
- Which action is selected?
  - $A_t = \underset{a}{\text{argmax}}[H_t(a)]$
    - Hardmax results in no exploration -- deterministic action selection

Softmax!

Softmax function (Boltzmann distribution)

- **Input**: vector of scores
- **Output**: vector of probabilities forming a valid distribution
- $\Pr\{a_t = a\} = \frac{e^{H_t(a)}}{\sum_{a' \in A} e^{H_t(a')}} = \Pr(a)$
- $H_t \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{bmatrix} 6 \\ 9 \\ 2 \end{bmatrix}$
- $\text{softmax} \begin{pmatrix} 6 \\ 9 \\ 2 \end{pmatrix} = \begin{bmatrix} 0.047 \\ 0.952 \\ 0.00087 \end{bmatrix}$
- That is, with $\Pr(0.95)$ choose a_2, $\Pr(0.05)$ choose a_1, and $< \Pr(0.01)$ choose a_3

Softmax function

- Exploration – checked!
- Softmax provides another important attribute – **a differentiable policy**
- Say that we learn that action a_1 results in good relative performance
- Hardmax: $A_t = \underset{a}{\text{argmax}}[H_t(a_1), H_t(a_2), H_t(a_3)]$
  - Change $H(a_1)$ such that $\Pr(a_1)$ is increased, $\frac{\partial \Pr(a_1)}{\partial H(a_1)} = NA$
- Softmax: $\Pr(a_1) = \frac{e^{H_t(a)}}{\sum_{a' \in A} e^{H_t(a')}}$
  - Change $H(a_1)$ such that $\Pr(a_1)$ is increased -> update towards $\frac{\partial \Pr(a_1)}{\partial H(a_1)}$

Gradient ascend

- $H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \overline{R_t})\frac{\partial \Pr(A_t)}{\partial H(A_t)}$?
- $\forall a \neq A_t, H_{t+1}(a) = H_t(a) + \alpha(R_t - \overline{R_t})\frac{\partial \Pr(A_t)}{\partial H(a)}$
- Update the preferences based on observed reward and a baseline reward ($\overline{R_t}$)
- If the observed reward is larger than the baseline:
  - Increase the preference of the chosen action, A_t
  - Decrease the preference of all other actions, $\forall a \neq A_t$
- Else do the opposite

Softmax derivative

$H_i \coloneqq H(a_i)$
$p_i \coloneqq \Pr(a_i)$

- $\frac{\partial p_i}{\partial H_j} = \frac{\partial \frac{e^{H_i}}{\sum_k e^{H_k}}}{\partial H_j}$
- $f = e^{H_i}, g = \sum_k e^{H_k}$

$$\left(\frac{f(x)}{g(x)}\right)' = \frac{f'(x)g(x) - f(x)g'(x)}{g^2(x)}$$

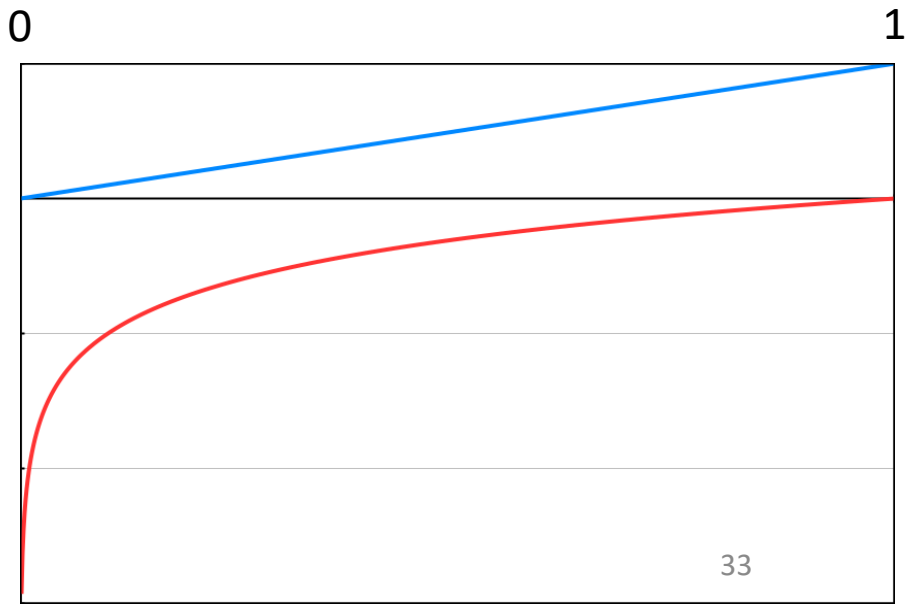
- Case $i = j$:
  - $\frac{\partial p_i}{\partial H_j} = \frac{e^{H_i} g - e^{H_i} e^{H_j}}{g^2} = \frac{e^{H_i}}{g} \cdot \frac{g - e^{H_j}}{g} = p_i(1 - p_j) = p_j(1 - p_j)$
- Case $i \neq j$:
  - $\frac{\partial p_i}{\partial H_j} = \frac{0 g - e^{H_i} e^{H_j}}{g^2} = -\frac{e^{H_i}}{g} \cdot \frac{e^{H_j}}{g} = -p_i p_j$

Softmax derivative

- $\text{softmax}\begin{pmatrix} H_1 \\ H_2 \\ H_3 \end{pmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$
- Partial derivative relates to a given input/output pair
  - Jacobian matrix with entries: H_j, p_i
- $\frac{\partial p_i}{\partial H_j} = \begin{cases} p_i(1 - p_j)\,, & i = j \\ -p_j p_i\,, & i \neq j \end{cases}$

Log-likelihood (gradient ascend)

- $H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \overline{R_t})\frac{\partial \Pr(A_t)}{\partial H(A_t)}$
- $\forall a \neq A_t, H_{t+1}(a) = H_t(a) + \alpha(R_t - \overline{R_t})\frac{\partial \Pr(A_t)}{\partial H(a)}$
- Replace $\Pr(A_t)$ with $\log(\Pr(A_t))$
- How does $\frac{\partial \log(\Pr(A_t))}{\partial H(A_t)}$ behave?
- How is it different from $\frac{\partial \Pr(A_t)}{\partial H(A_t)}$?
- Provides a helpful gradient!



Softmax derivative with log-likelihood

- $\frac{\partial p_i}{\partial H_j} = \begin{cases} p_i(1-p_i)\,, & i = j \\ -p_j p_i\,, & i \neq j \end{cases}$

- $\frac{\partial \log(p_i)}{\partial H_j} = \begin{cases} \left(\frac{\mathbf{1}}{\boldsymbol{p_i}}\right) p_i(1-p_i)\,, & i = j \\ -p_j p_i\,, & i \neq j \end{cases}$

$$f(x) = \ln x \rightarrow f'(x) = \frac{1}{x}$$
$$f\,(\,g(x)\,)' = f\,'\,(\,g(x)\,) \cdot g'(x)$$

- $\frac{\partial \log(p_i)}{\partial H_j} = \begin{cases} (1-p_j)\,, & i = j \\ -p_j\,, & i \neq j \end{cases}$ ← Not depended on p_i

Softmax derivative with log-likelihood

- $\frac{\partial p_i}{\partial a_j} = \begin{cases} p_i(1-p_j)\,, & i = j \\ -p_j p_i\,, & i \neq j \end{cases}$
- $\frac{\partial \log(p_i)}{\partial a_j} = \begin{cases} (1-p_j)\,, & i = j \\ -p_j\,, & i \neq j \end{cases}$ ← Not depended on p_i
- $H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \overline{R_t})\Pr(A_t)\,(1 - \Pr(A_t))$
- $\forall a \neq A_t, H_{t+1}(a) = H_t(a) - \alpha(R_t - \overline{R_t})\Pr(A_t)\Pr(a)$
- $H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \overline{R_t})(1 - \Pr(A_t))$
- $\forall a \neq A_t, H_{t+1}(a) = H_t(a) - \alpha(R_t - \overline{R_t})\Pr(a)$

Baseline, $\overline{R_t}$?

$$H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \overline{R_t})(1 - \Pr(A_t))$$
$$\forall a \neq A_t, H_{t+1}(a) = H_t(a) - \alpha(R_t - \overline{R_t})\Pr(a)$$

- If observed reward is bigger than baseline:
  - Increase the score of the chosen action, A_t
  - Decrease the score of all other actions, $\forall a \neq A_t$
- Else do the opposite



$H(a)$	3	2	1
$\Pr(a)$	0.665	0.245	0.09
$\boldsymbol{R}$	2	8	1

Baseline, $\overline{R_t}$?

$$H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \overline{R_t})(1 - \Pr(A_t))$$
$$\forall a \neq A_t, H_{t+1}(a) = H_t(a) - \alpha(R_t - \overline{R_t})\Pr(a)$$

*In this example we are updating each machine assuming it was played and the affiliated R_t was observed



$H(a)$	3	2	1
$\Pr(a)$	0.665	0.245	0.09
R_t	2	8	1
$(R_t - \overline{R_t})(1 - \Pr(A_t))$			
$\overline{R_t} = 10$	-2.68	-1.51	-8.19
$\overline{R_t} = 3$	-0.34	3.78	-1.82
$\overline{R_t} = -3$	1.68	8.31	3.64

Observed reward

Update direction for various baseline values

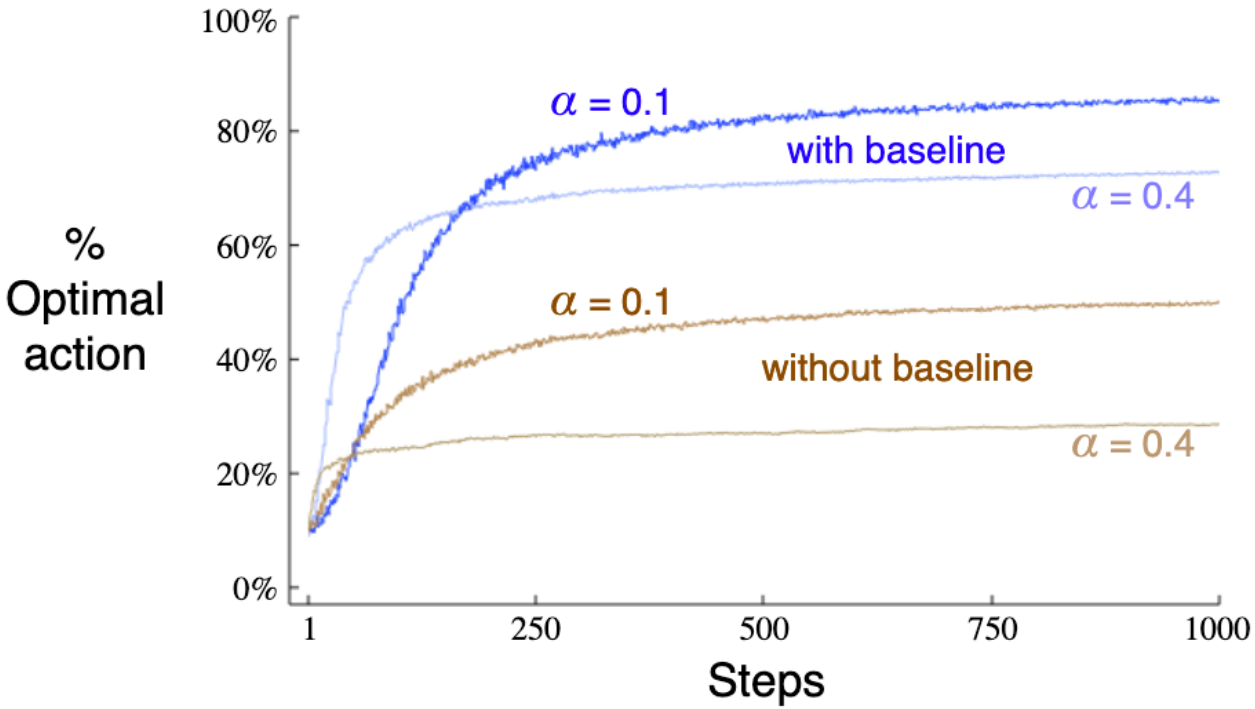
Which baseline value makes most sense?

Setting $\overline{R_t} = \frac{1}{t}\sum_t r$ is usually a good choice

*More on choosing a baseline later

Baseline impact

- Average performance of gradient ascend
- With baseline: $\overline{R_t} = \frac{1}{t}\sum_t r$
- No baseline: $\overline{R_t} = 0$
- 10-armed example problem

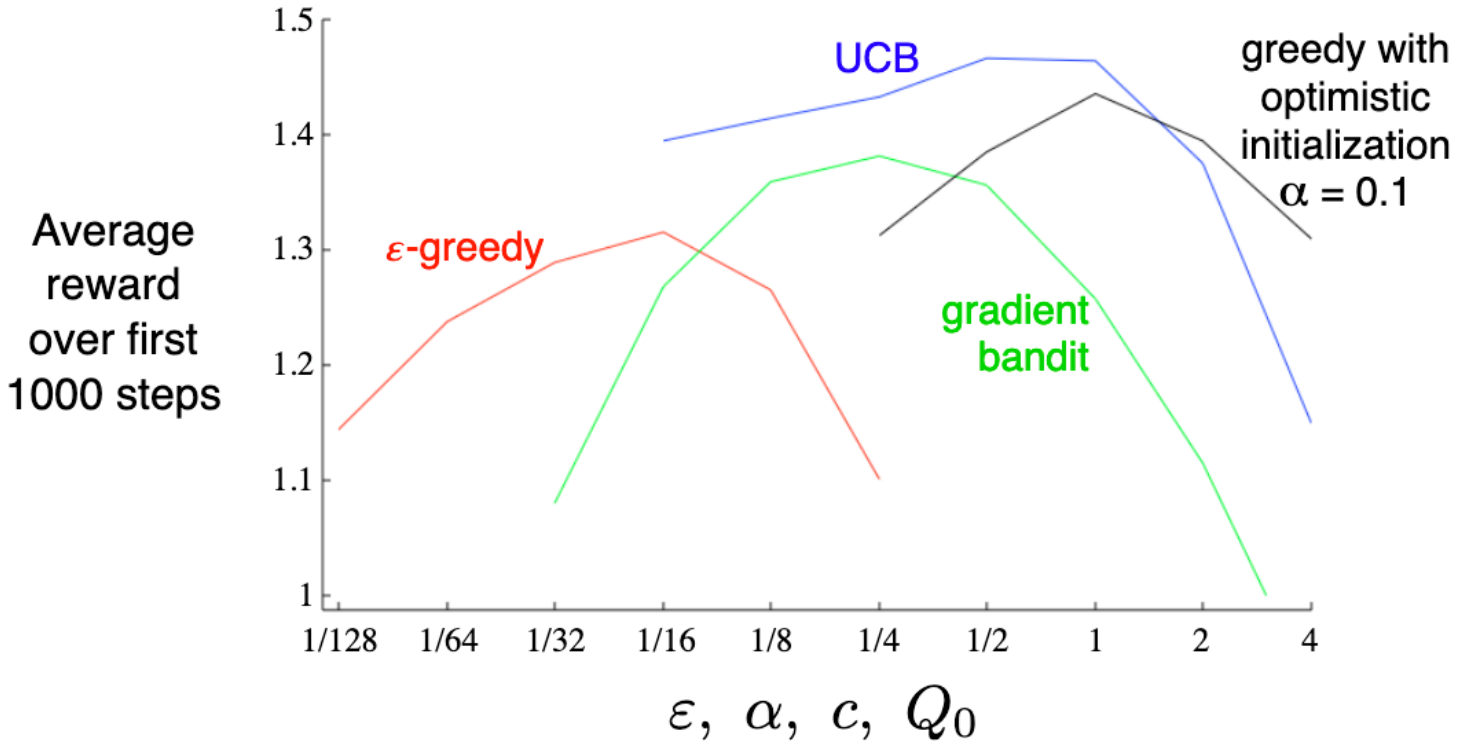


What did we learn?

- **Problem**: choose the action that results in highest expected reward
- **Assumptions**: (1) actions' expected reward is unknown, (2) we are confronted with the same problem over and over, (3) we are able to observe an action's outcome once chosen
- **Approach**: learn the actions' expected reward through exploration (value based) or learn a policy directly (policy based), exploit learnt knowledge to choose best action
- **Methods:** (1) greedy + initializing estimates optimistically, (2) epsilon-greedy, (3) Upper-Confidence-Bounds, (4) gradient ascend + soft-max
- Which works best?

Overall results

- A parameter study of the various bandit algorithms
- Each point is the average reward obtained over 1000 steps with a particular algorithm at a particular setting of its parameter



Next…

- **Class:**
  - Markov Decision Processes
- **Assignments**:
  - Quiz Multi Armed Bandits on Canvas by Sep-3, EOD.
  - Go over tutorials
    - Linear algebra
    - Basic probability
    - Python
    - Numpy
    - Gym (OpenAI)
- **Project**:
  - Go over the project description and timeline
  - Find a partner (a relevant discussion board is available on Campuswire)