COMP 3200
Artificial Intelligence

Lecture 17

Exploitation vs. Exploration

Bandit Algorithms

Exploitation vs. Exploration

- One of the main challenges in RL is the trade-off between exploitation and exploration
- To obtain a lot of reward, agents must prefer actions that it knows produce good results
- In order to learn which actions produce good rewards, it must try them out first
- The agent must exploit knowledge it has, but also explore in order to gain more knowledge
- Also one of the main challenges of *real life*

Learning

- Learning is the process of gaining new (or modifying existing) knowledge
- We have learned something if we have information that we didn't have before
- Even if we are given same results, we learn to trust them with more certainty (our estimate has less variance)

Exploitation

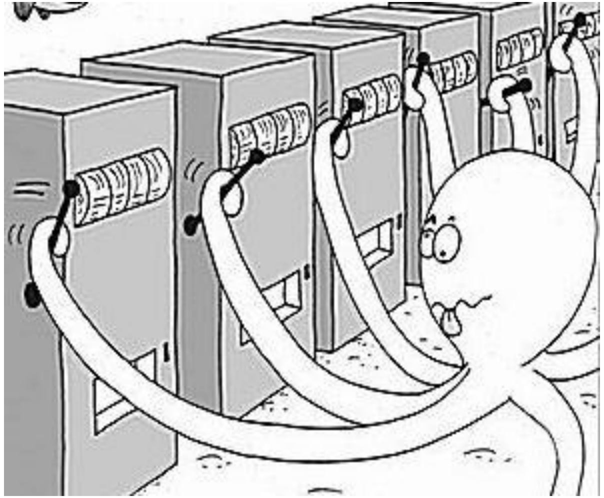
- Exploit our current knowledge
- Choose high-valued actions for which we already know the value
- Low risk, Low Reward (possibly)
  - We know what we will get
- Always choosing actions we are familiar with gives us little new information

Exploration

- Try actions we haven't before
- Helps us learn their values
- Possible that new actions have higher rewards than previously selected
- Once we have explored sufficiently, then we can exploit the best actions and know that they are the best

N-Armed Bandit Problem

- Repeatedly make a choice among n different actions
- After each action you receive a reward from a stationary probability distribution depending on the action
- Objective is to maximize your expected total reward over a number of action selections

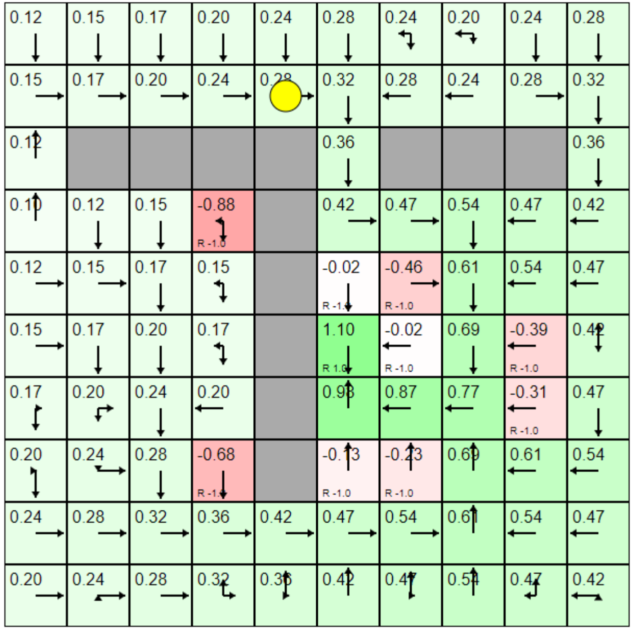


N-Armed Bandit Problem

- Also called Multi-Armed Bandit
- Name is an analogy to slot machines: "One-armed bandit"
- You have a limited amount of money, and you try to win as much as possible

- How do we select which levers to pull?

N-Armed Bandit is Everywhere



Exploitation vs. Exploration

- If we maintain an estimate of action values, at any time there is one greatest
  - The 'Greedy Action'
- Exploitation: Choosing the Greedy Action
  - Maximizes single action returns
- Exploration: Choosing a non-greedy action to improve your action estimates
  - Required for future reward maximization
- How to balance exploitation vs. exploration?

Two Important EvE Concepts

1. How to store and update value estimates as we learn over time from new info

2. How to choose which action to do next based on our current value estimate

Action-Value Methods

- How to store the current value estimate?
- $Q_t(a)$ = Estimate of $Q^*(a)$ after time step t
- $Q^*(a)$ = Actual Value of action a
  - "Actual Value" = Mean Reward
  - As time t goes to infinity

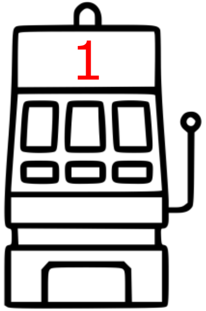
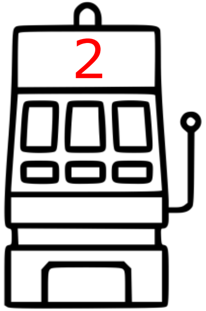
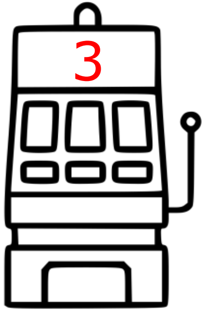
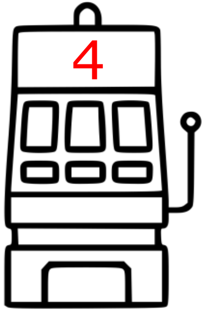
Q-Value Implementation

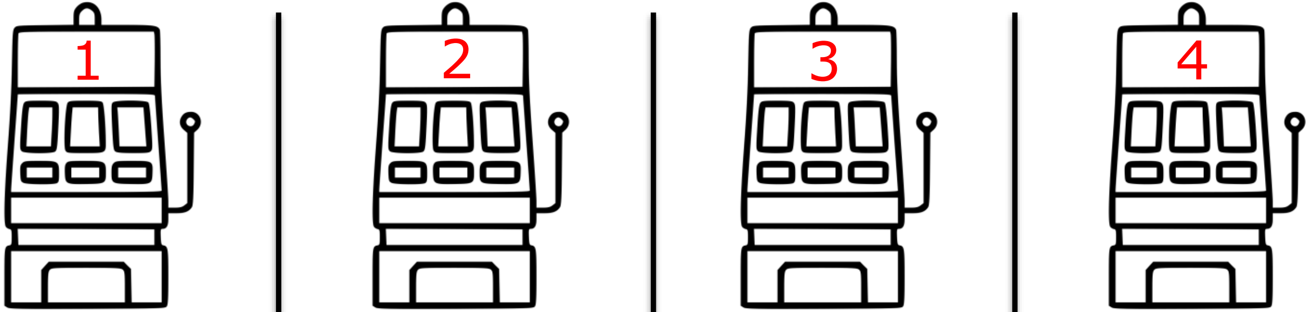
- Example: N-Armed Bandit Problem
- Q(a) = Estimate of bandit a reward
  - Choosing a bandit = one action
- Q can be an array of size n for n bandits

  $Q_0 = [0, 0, 0, 0]$ n=4
- After some number of time steps t

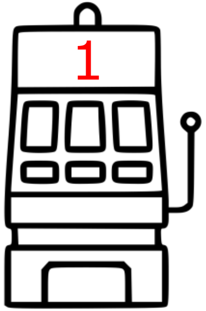
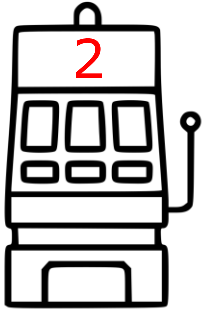
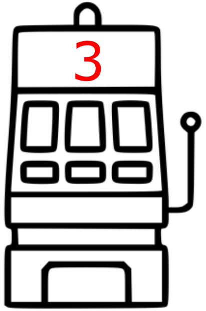
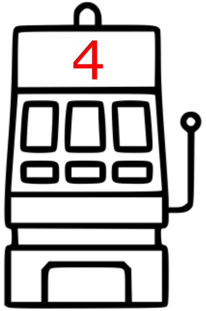
  $Q_t = [4, 7, 22, 10]$

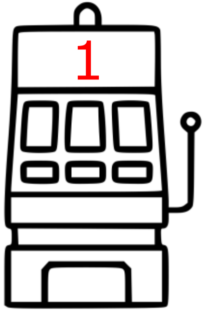
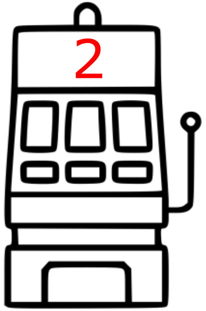
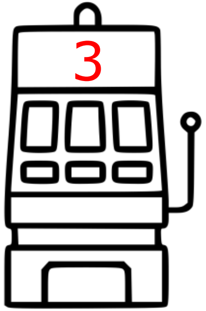
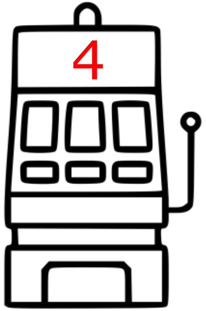
Bandit Example

Bandit Actions a	 1	 2	 3	 4
Distribution				
Rewards r_t				
Values Q_t	$Q_t(1)$	$Q_t(2)$	$Q_t(3)$	$Q_t(4)$

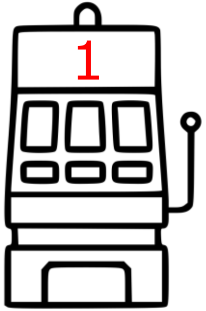
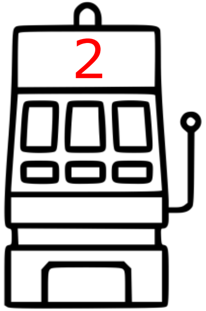
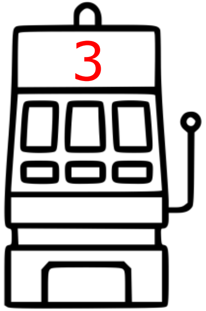
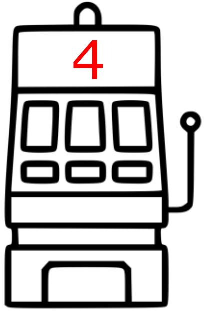


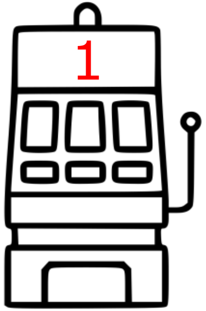
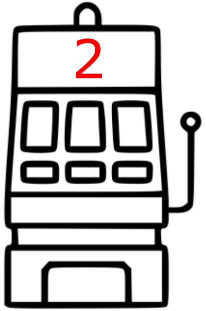
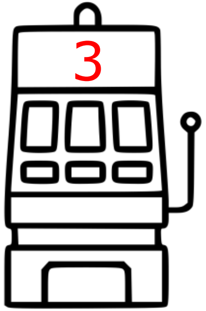
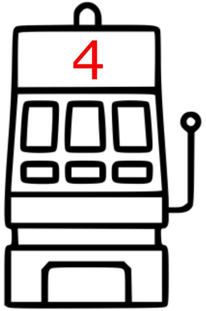
Bandit Actions a	1	2	3	4
Distribution	?	?	?	?
Rewards r_t				
Values Q_t				

Bandit Actions a				
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t				
Values Q_0				

Bandit Actions a				
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t				
Valu Q_0	?	?	?	?

How to choose initial Q values?

Bandit Actions a				
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t				
Values Q_0	$Q_0(1)$	$Q_0(2)$	$Q_0(3)$	$Q_0(4)$

Bandit Actions a	 1	 2	 3	 4
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t				
Values Q_0	0	0	0	0

Value Estimation Methods

Two main variants of estimating value

1. Sample Average Estimation

2. Incremental Update Estimation

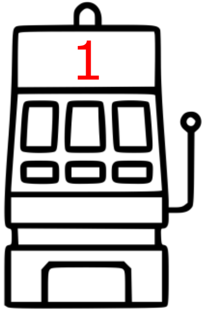
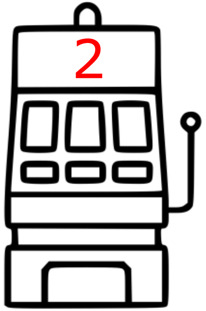
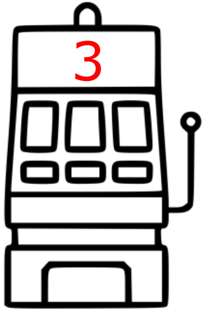
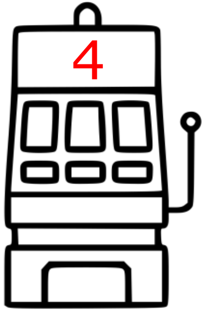
$Q_t(a)$ Sample Average Estimation

- Natural way of calculating $Q_t(a)$ is to average the rewards received so far after a number of plays
- If at play t, action a has been chosen k_a times, yielding rewards $r_1, r_2, \ldots, r_{k_a}$ then:

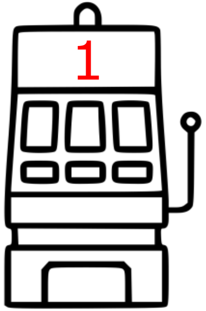
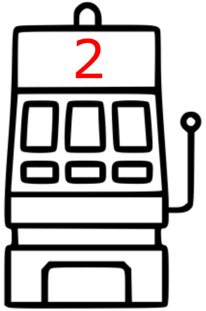
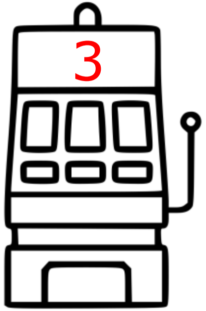
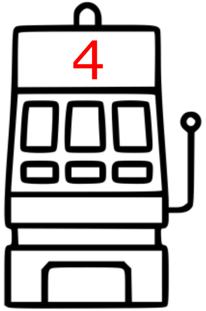
$$Q_t(a) = (r_1 + r_2 + \ldots + r_{k_a}) / k_a$$

- If $k_a = 0$, define Q_t as some default, $Q_t(a) = 0$
- As k_a gets large, $Q_t(a)$ converges to $Q^*(a)$
- Average of samples = “sample average” method

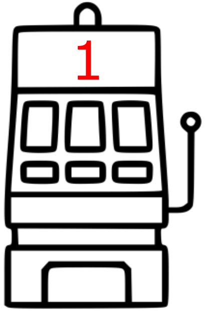
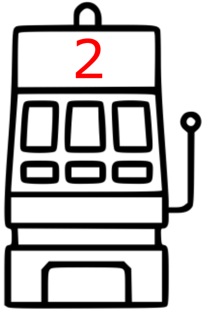
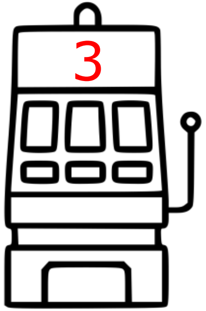
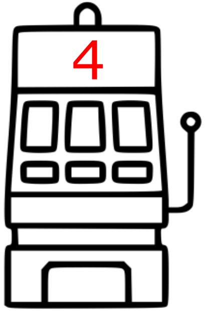
Sample Average Estimation

Bandit Actions a	1 	2 	3 	4 
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t				
Values Q_0	0	0	0	0

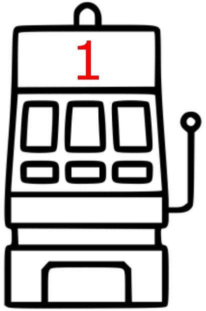
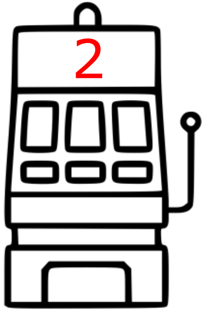
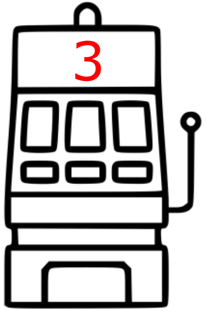
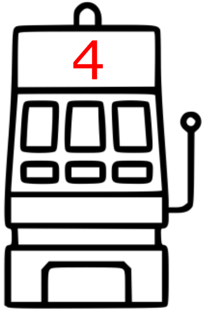
Sample Average Estimation

Bandit Actions a	 1	 2	 3	 4
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t	5			
Valu Q_1	5	0	0	0

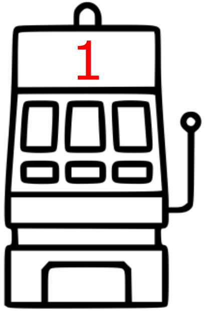
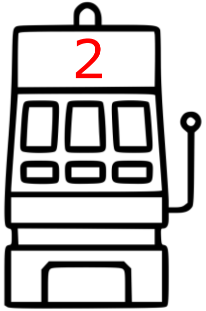
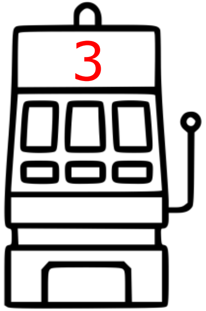
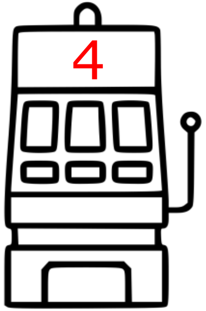
Sample Average Estimation

Bandit Actions a	 1	 2	 3	 4
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t	5	-20		
Values Q_2	5	-20	0	0

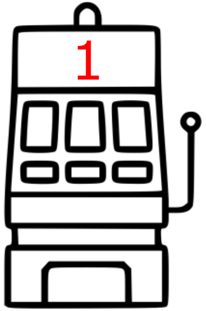
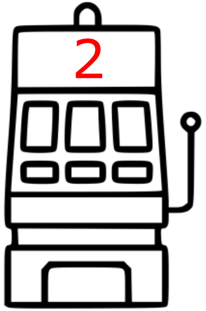
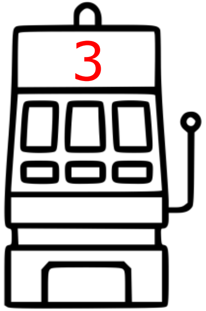
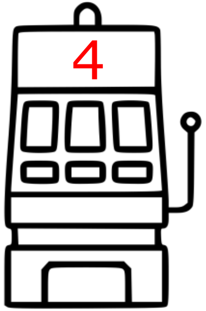
Sample Average Estimation

Bandit Actions a	 1	 2	 3	 4
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t	5	-20	4	
Values Q_3	5	-20	4	0

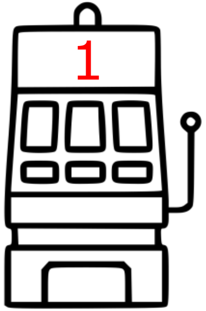
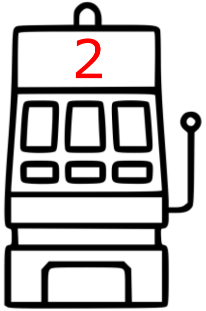
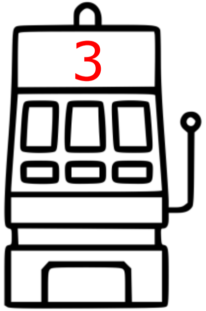
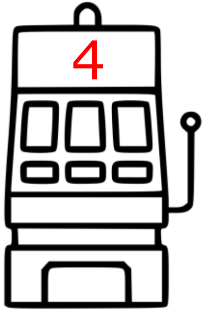
Sample Average Estimation

Bandit Actions a	 1	 2	 3	 4
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t	5	-20	4	8
Values Q_4	5	-20	4	8

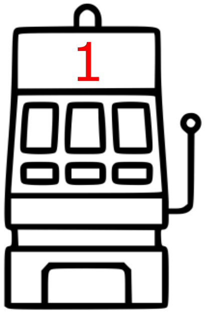
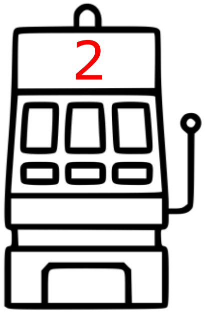
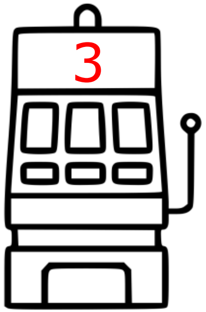
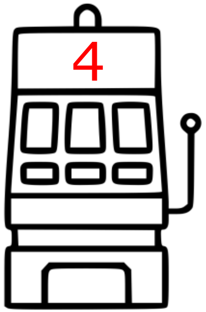
Sample Average Estimation

Bandit Actions a	 1	 2	 3	 4
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t	5 5	-20	4	8
Values Q_5	5	-20	4	8

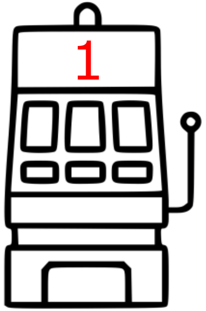
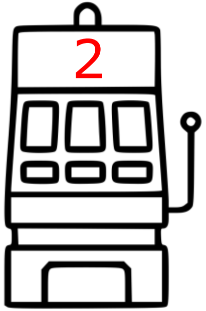
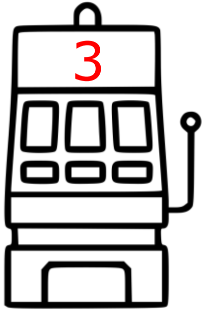
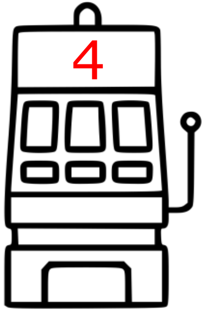
Sample Average Estimation

Bandit Actions a	 1	 2	 3	 4
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t	5 5	-20 20	4	8
Values Q_6	5	0	4	8

Sample Average Estimation

Bandit Actions a	 1	 2	 3	 4
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t	5 5	-20 20	4 0	8
Values Q_7	5	0	2	8

Sample Average Estimation

Bandit Actions a	 1	 2	 3	 4
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t	5 5	-20 20	4 0	8 -2
Values Q_8	5	0	2	3

Value Estimation Methods

Two main variants of estimating value

1. Sample Average Estimation
2. Incremental Update Estimation

Incremental Action-Value Est.

- Recall: $Q_t(a) = (r_1+r_2+\ldots+r_{k_a}) / k_a$
- Need to store all of the rewards
- Problem: Memory and computational requirements grow over time
- Let's derive an incremental formula so that memory is no longer an issue
  - Have: NewAvg = F(history of all samples)
  - Wanted: NewAvg = F(OldAverage, NewSample)

Incremental Average

$$Q_k = (r_1 + r_2 + \ldots + r_k) / k$$

$$= \frac{1}{k}\left(\sum_{i=1}^{k} r_i\right)$$

$$Q_{k+1} = \frac{1}{k+1}\left(\sum_{i=1}^{k+1} r_i\right)$$

$$= \frac{1}{k+1}\left(\sum_{i=1}^{k} r_i + r_{k+1}\right)$$

$$= \frac{1}{k+1}\left(kQ_k + r_{k+1}\right)$$

$$= \frac{1}{k+1}\left(r_{k+1} + kQ_k + Q_k - Q_k\right)$$

$$= \frac{1}{k+1}\left(r_{k+1} + (k+1)Q_k - Q_k\right)$$

$$= Q_k + \frac{1}{k+1}\left(r_{k+1} - Q_k\right)$$

Incremental Implementation

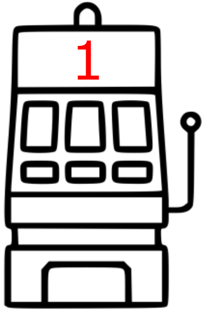
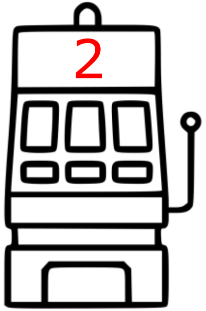
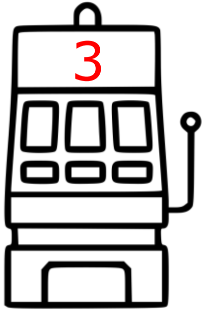
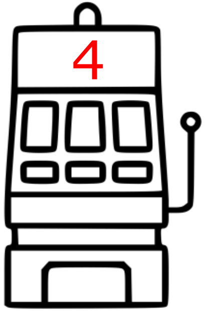
Q_k = average of first k rewards

$$Q_{k+1} = \frac{1}{k+1}\sum_{i=1}^{k+1} r_i$$

$$= Q_k + \frac{1}{k+1}(r_{k+1} - Q_k)$$

NewEst = OldEst + StepSize(NewSample - OldEst)

NewSample referred to as the ‘target’ value

Bandit Actions a				
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t				
Values Q_a				

What if distribution changes dramatically?

Changing Reward Distributions

- If the distributions remain the same, the problem is called stationary
- If the distributions are allowed to change over time, it is non-stationary
- Imagine having an average of 1 for a million time steps, and then changing to 100. Updating the average would be slow

Non-stationary Problems

- Averaging works fine for stationary rewards, but not if it changes over time
- Want to weight recent rewards more than old ones, in case the values change
- Use a constant step-size parameter $0 < \alpha \leq 1$

$$Q_{k+1} = Q_k + \alpha \left(r_{k+1} - Q_k\right)$$

Incremental Update Example

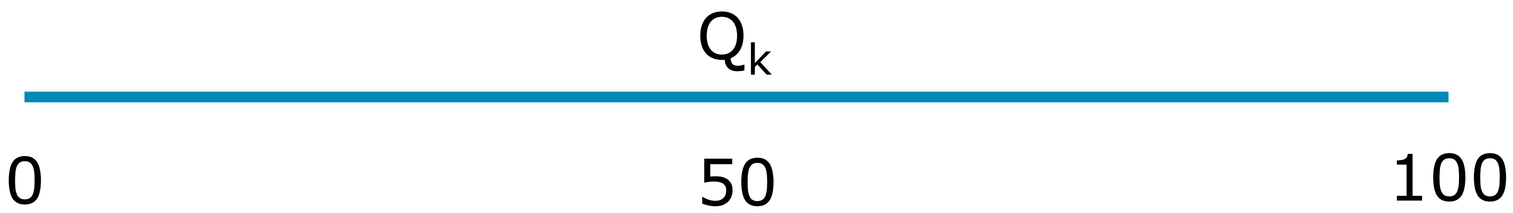
- $Q_{k+1} = Q_k + \alpha\,(r_{k+1} - Q_k)$
- New estimate pulled toward r_{k+1} by α

- $Q_k = 50$, $\alpha = 1$, $r_{k+1} = 100$
- $Q_{k+1} = 50 + 1*(100-50) = 50+50 = 100$

- $Q_k = 50$, $\alpha = 0.5$, $r_{k+1} = 100$
- $Q_{k+1} = 50 + 0.5*(100-50) = 75$

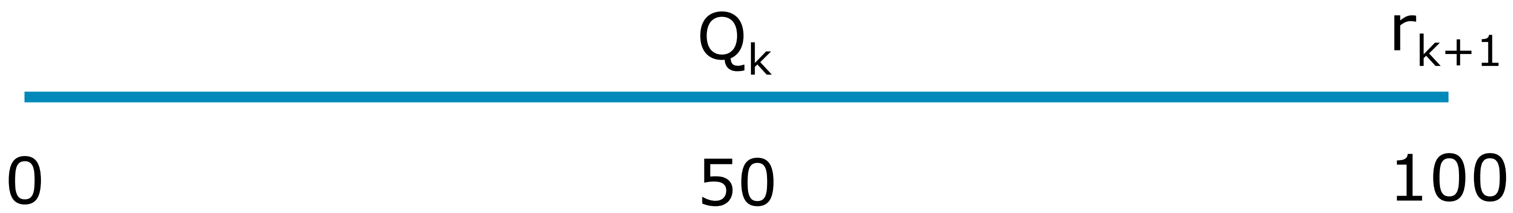
Incremental Update Example

- $Q_{k+1} = Q_k + \alpha\,(r_{k+1} - Q_k)$
- New estimate pulled toward r_{k+1} by α
- $Q_k \quad = 50$, $\alpha = 1$, $r_{k+1} = 100$
- $Q_{k+1} = 50 + 1*(100\text{-}50) = 50+50 = 100$



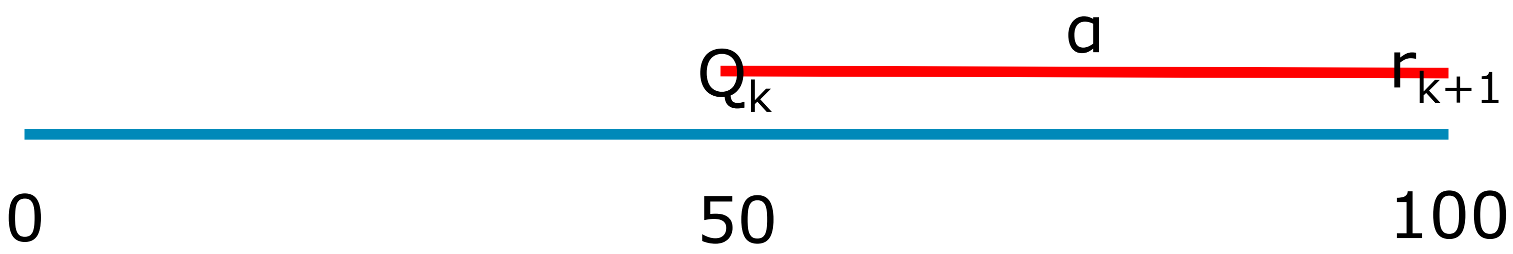
Incremental Update Example

- $Q_{k+1} = Q_k + \alpha\,(r_{k+1} - Q_k)$
- New estimate pulled toward r_{k+1} by α
- $Q_k = 50,\ \alpha = 1,\ r_{k+1} = 100$
- $Q_{k+1} = 50 + 1*(100-50) = 50+50 = 100$



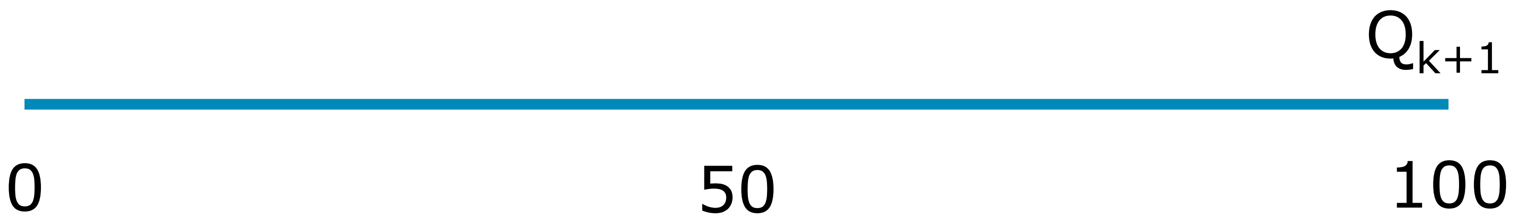
Incremental Update Example

- $Q_{k+1} = Q_k + \alpha\,(r_{k+1} - Q_k)$
- New estimate pulled toward r_{k+1} by α
- $Q_k = 50$, $\alpha = 1$, $r_{k+1} = 100$
- $Q_{k+1} = 50 + 1*(100-50) = 50+50 = 100$



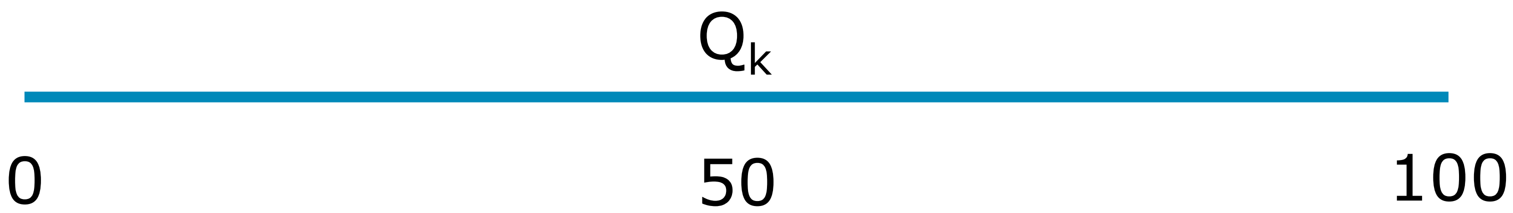
Incremental Update Example

- $Q_{k+1} = Q_k + \alpha\,(r_{k+1} - Q_k)$
- New estimate pulled toward r_{k+1} by α
- $Q_k \quad = 50,\ \alpha = 1,\ r_{k+1} = 100$
- $Q_{k+1} = 50 + 1*(100\text{-}50) = 50+50 = 100$



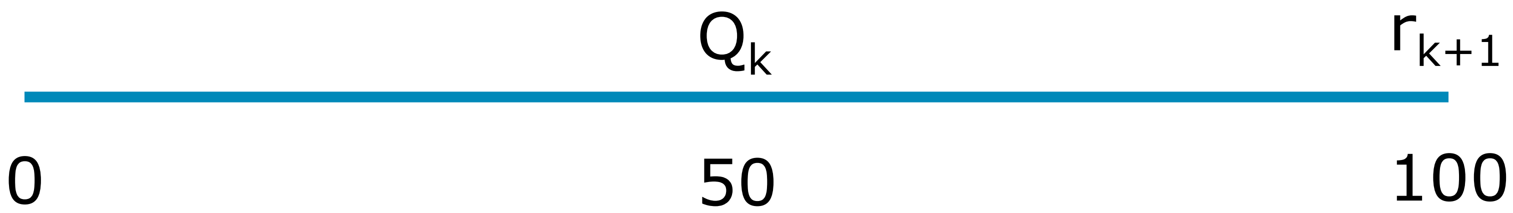
Incremental Update Example

- $Q_{k+1} = Q_k + \alpha\,(r_{k+1} - Q_k)$
- New estimate pulled toward r_{k+1} by α
- $Q_k = 50$, $\alpha = 0.5$, $r_{k+1} = 100$
- $Q_{k+1} = 50 + 0.5*(100-50) = 75$



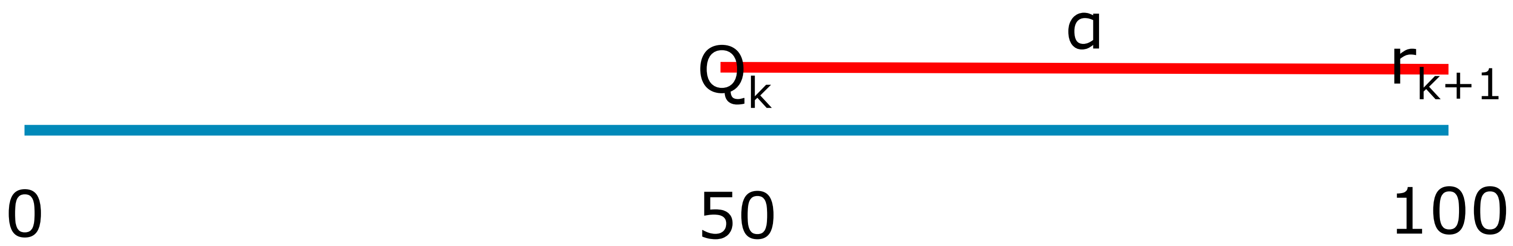
Incremental Update Example

- $Q_{k+1} = Q_k + \alpha\,(r_{k+1} - Q_k)$
- New estimate pulled toward r_{k+1} by α
- $Q_k = 50$, $\alpha = 0.5$, $r_{k+1} = 100$
- $Q_{k+1} = 50 + 0.5*(100-50) = 75$



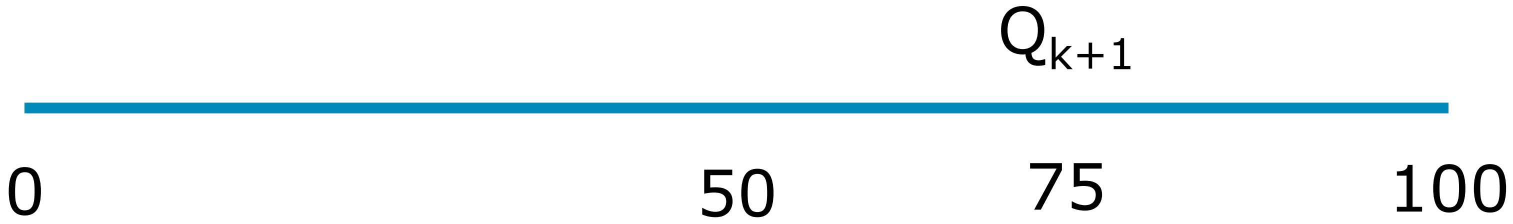
Incremental Update Example

- $Q_{k+1} = Q_k + \alpha\,(r_{k+1} - Q_k)$
- New estimate pulled toward r_{k+1} by α
- $Q_k = 50$, $\alpha = 0.5$, $r_{k+1} = 100$
- $Q_{k+1} = 50 + 0.5*(100-50) = 75$



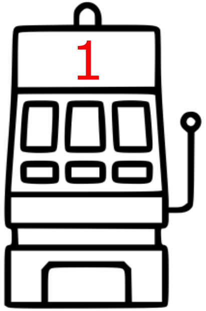
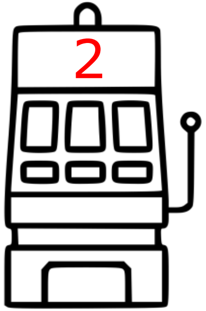
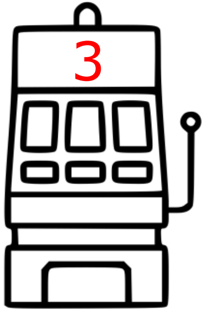
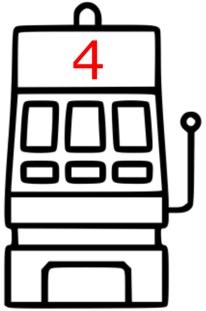
Incremental Update Example

- $Q_{k+1} = Q_k + \alpha\,(r_{k+1} - Q_k)$
- New estimate pulled toward r_{k+1} by α
- $Q_k = 50$, $\alpha = 0.5$, $r_{k+1} = 100$
- $Q_{k+1} = 50 + 0.5*(100-50) = 75$

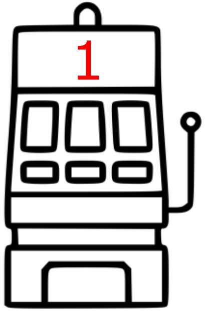
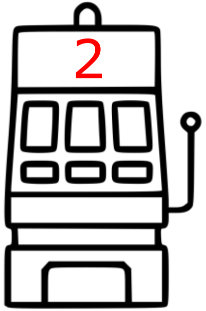
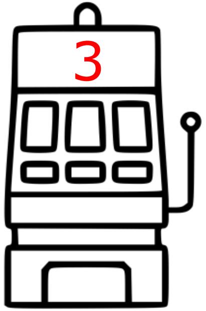
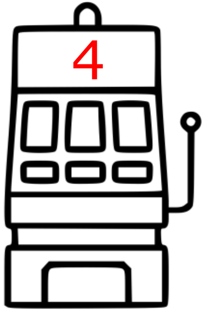


Incremental Update $\alpha=0.5$

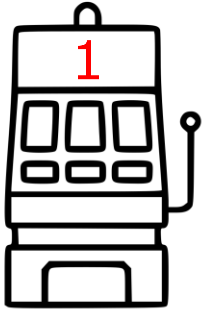
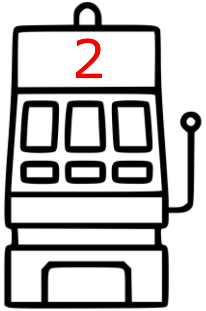
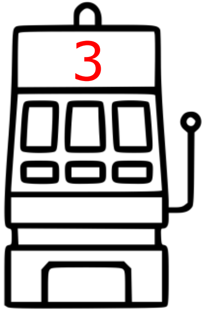
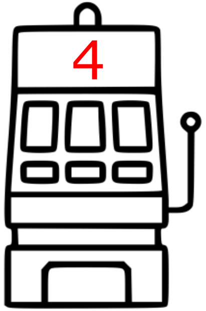
COMP 3200
David Churchill

Bandit Actions a				
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t				
Values Q_0	0	0	0	0

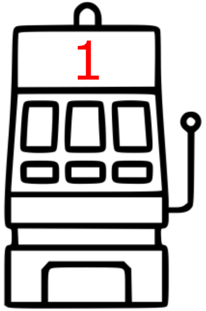
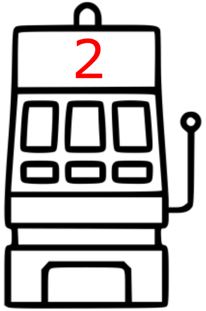
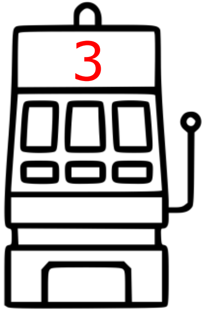
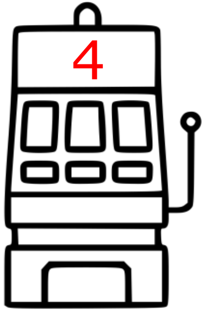
Incremental Update $\alpha=0.5$

Bandit Actions a	1 	2 	3 	4 
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t	5			
Values Q_1	2.5	0	0	0

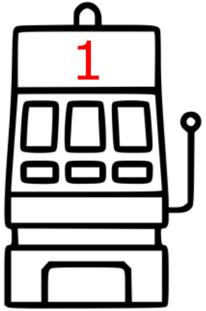
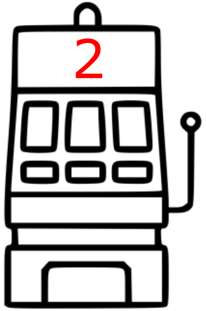
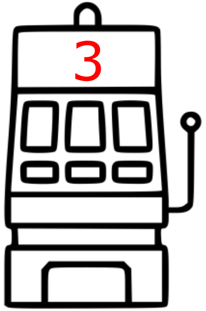
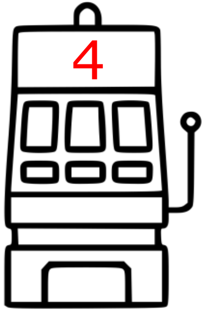
Incremental Update $\alpha=0.5$

Bandit Actions a	 1	 2	 3	 4
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t	5	-20		
Values Q_2	2.5	-10	0	0

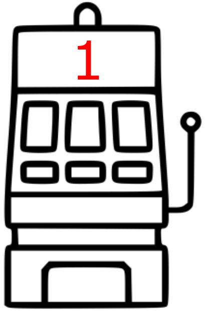
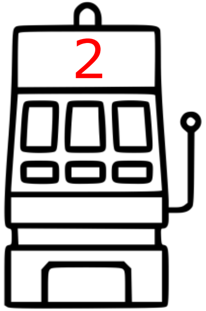
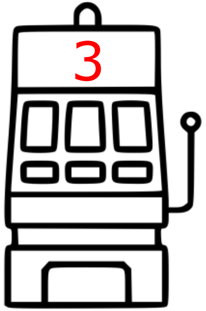
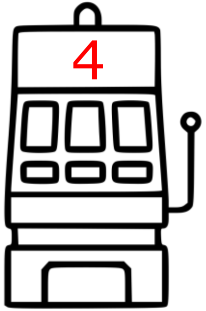
Incremental Update $\alpha=0.5$

Bandit Actions a	 1	 2	 3	 4
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t	5	-20	4	
Values Q_3	2.5	-10	2	0

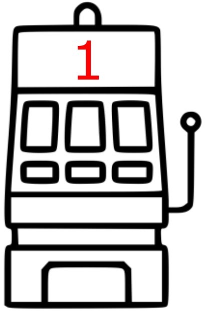
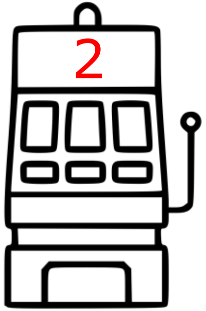
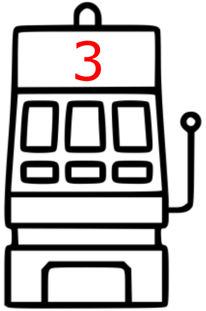
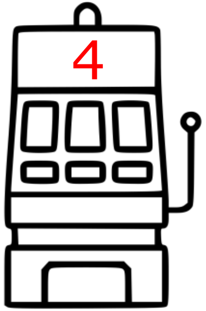
Incremental Update $\alpha=0.5$

Bandit Actions a	 1	 2	 3	 4
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t	5	-20	4	8
Values Q_4	2.5	-10	2	4

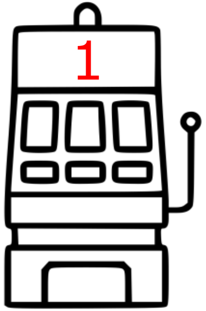
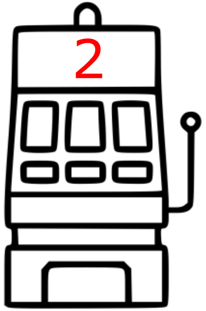
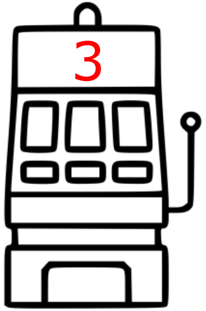
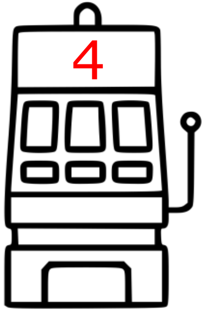
Incremental Update $\alpha = 0.5$

Bandit Actions a	1 	2 	3 	4 
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t	5 5	-20	4	8
Values Q_5	3.75	-10	2	4

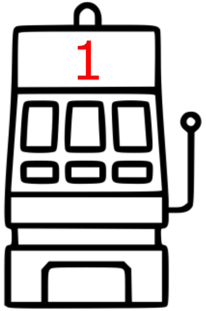
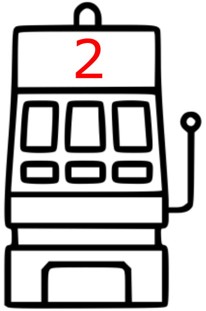
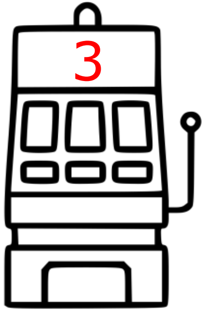
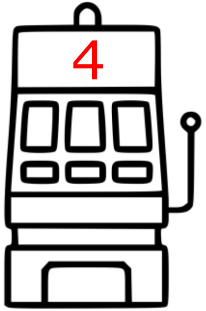
Incremental Update $\alpha=0.5$

Bandit Actions a	 1	 2	 3	 4
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t	5 5	-20 20	4	8
Values Q_6	3.75	5	2	4

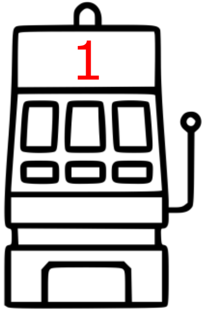
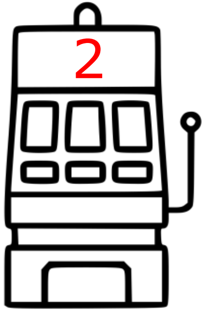
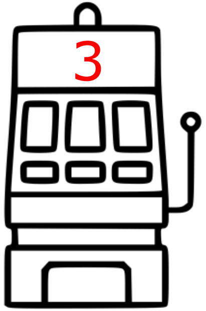
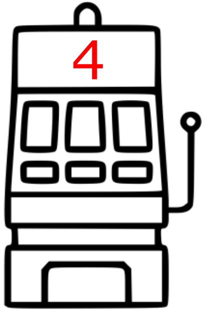
Incremental Update $\alpha=0.5$

Bandit Actions a	 1	 2	 3	 4
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t	5 5	-20 20	4 0	8
Values Q_7	3.75	5	1	4

Incremental Update α=0.5

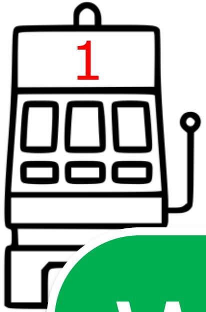
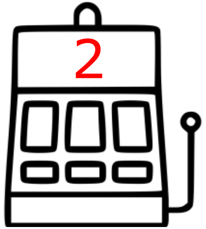
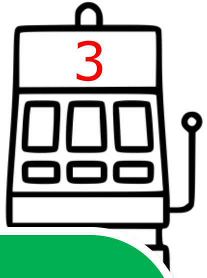
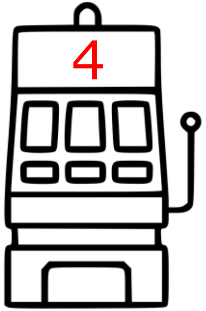
Bandit Actions a	 1	 2	 3	 4
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t	5 5	-20 20	4 0	8 -2
Values Q_8	3.75	5	1	1

Sample Average Estimation

Bandit Actions a	1 	2 	3 	4 
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t	5 5	-20 20	4 0	8 -2
Values Q_8	5	0	2	3

Action Selection Methods

Sample Average Estimation

Bandit Actions a	1 	2 	3 	4 
Distribution	Al		0)	Rand(-10,10)
Rewards r_t				8
Values Q_a	5	0	4	8

Which Action do we choose?

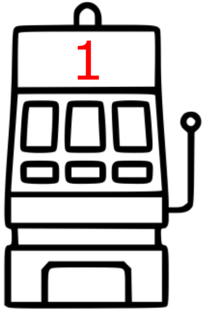
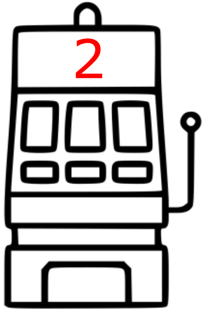
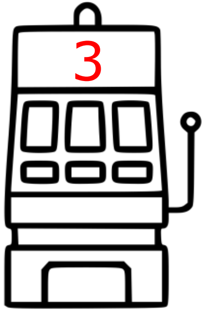
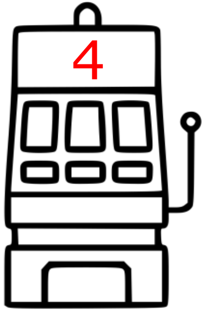
Greedy Action Selection

- How to select an action from value estimations?
- Simplest way: Greedy Selection
- Select on play t, a greedy action a* for which:

$$Q_t(a^*) = \max_a Q_t(a)$$

- This method always exploits current knowledge to maximize immediate reward
- No sampling or exploration to determine values of another action to see if they may be better

Sample Average Estimation

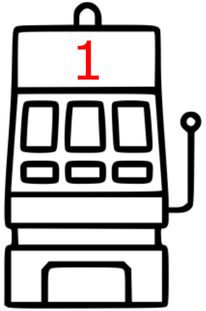
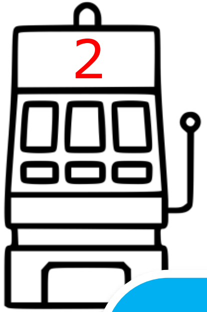
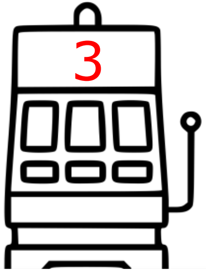
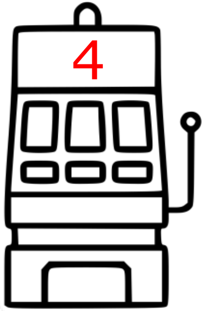
Bandit Actions a				
Distribution	Always 5	-20 or 20	Rand(0,10)	Rand(-10,10)
Rewards r_t	5			
Values Q_a	5	0	0	0

Greedy Will Always Choose Action 1

ε-Greedy Selection

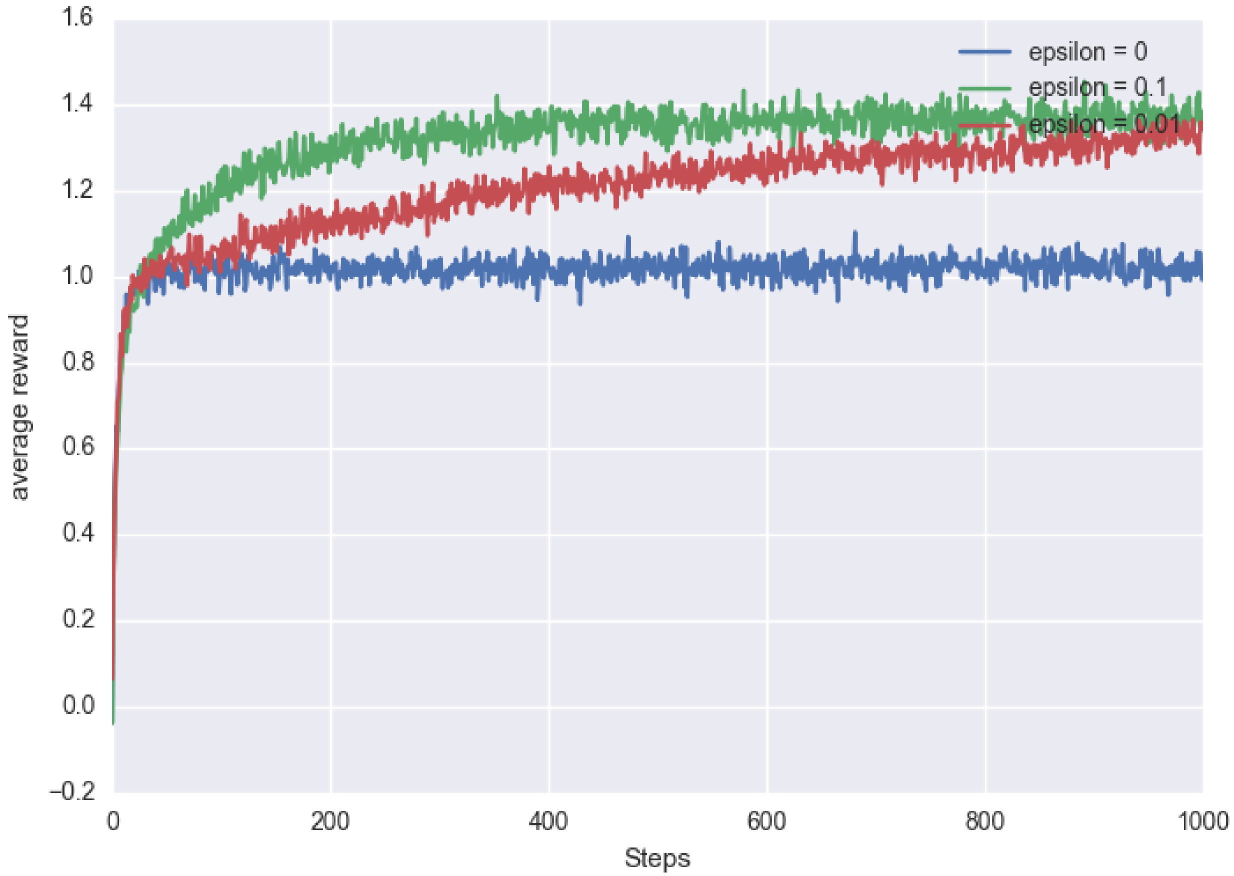
- To add exploration, choose a random action with small probability ε
- In the limit, as the number of plays increases, each action will be sampled infinite times
- This guarantees k_a -> infinity, and $Q_t(a)$ converges to $Q^*(a)$
- In theory this works, but in practice it may take a very, very long time to converge

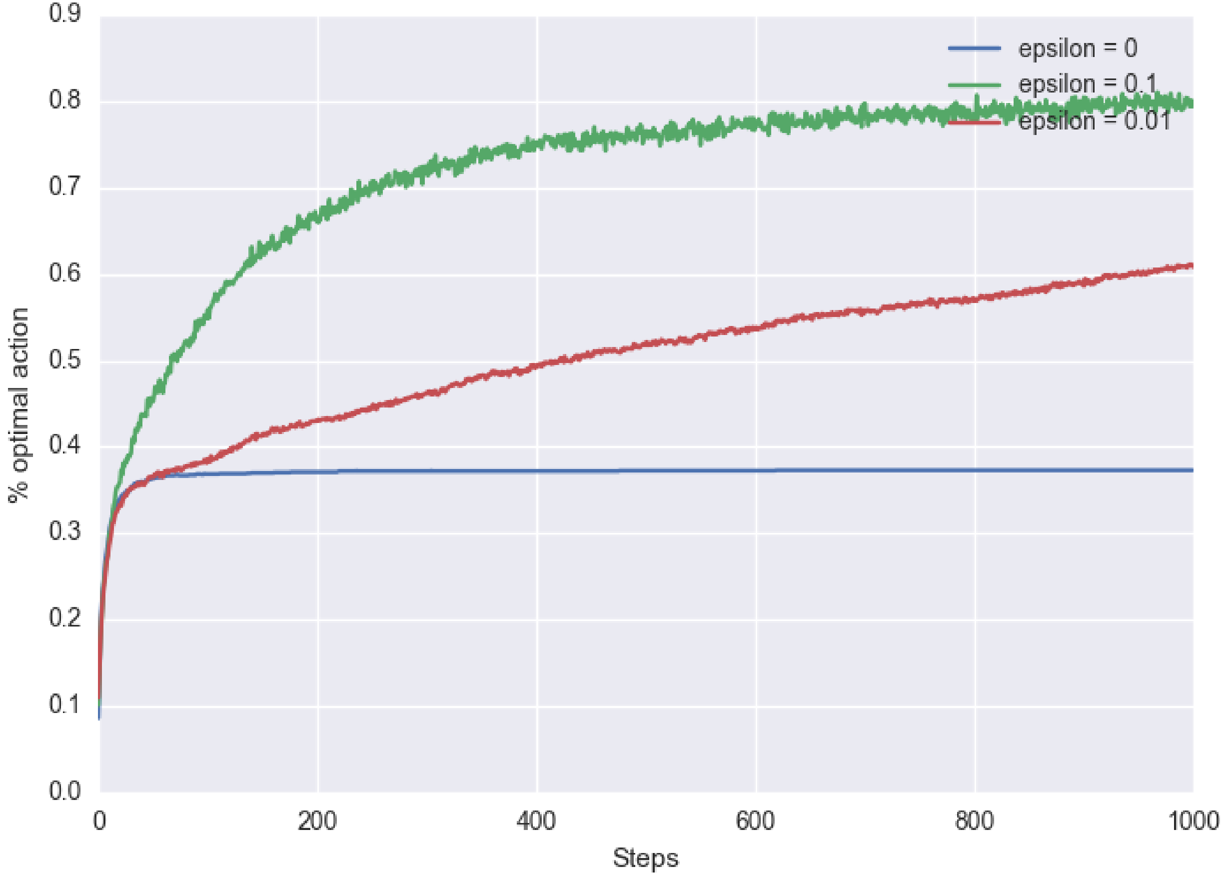
Sample Average Estimation

Bandit Actions a				
Distribution	Always 5	-20 c		nd(-10,10)
Rewards r_t	5			
Values Q_a	5	0	0	0

Random Action Chosen with Probability ε

Greedy Action Chosen With Probability 1-ε


epsilon = 0
epsilon = 0.1
epsilon = 0.01
average reward
Steps
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
200
400
600
800
1000


epsilon = 0
epsilon = 0.1
epsilon = 0.01
% optimal action
Steps
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
200
400
600
800
1000

Bandit Algorithm Implementation

1. Choose value update method
2. Choose action selection method
3. Repeat
    1. Choose an action to perform
    2. Get the reward value from bandit
    3. Update your value for the action

Bandit Algorithm Implementation

Incremental Avg + ε-Greedy

```
Function BanditAlgorithm(bandits)
Q[] = zeros(bandits.size)
N[] = zeros(bandits.size)
while (true)
    action = null
    if (rand() < ε)     action = randomAction()
    else                action = argmax_a(Q[a])
    R           = bandits[action].getValue()
    N[action]   = N[action] + 1
    Q[a]        = Q[a] + (1.0 / N[a])*(R – Q[a])
```

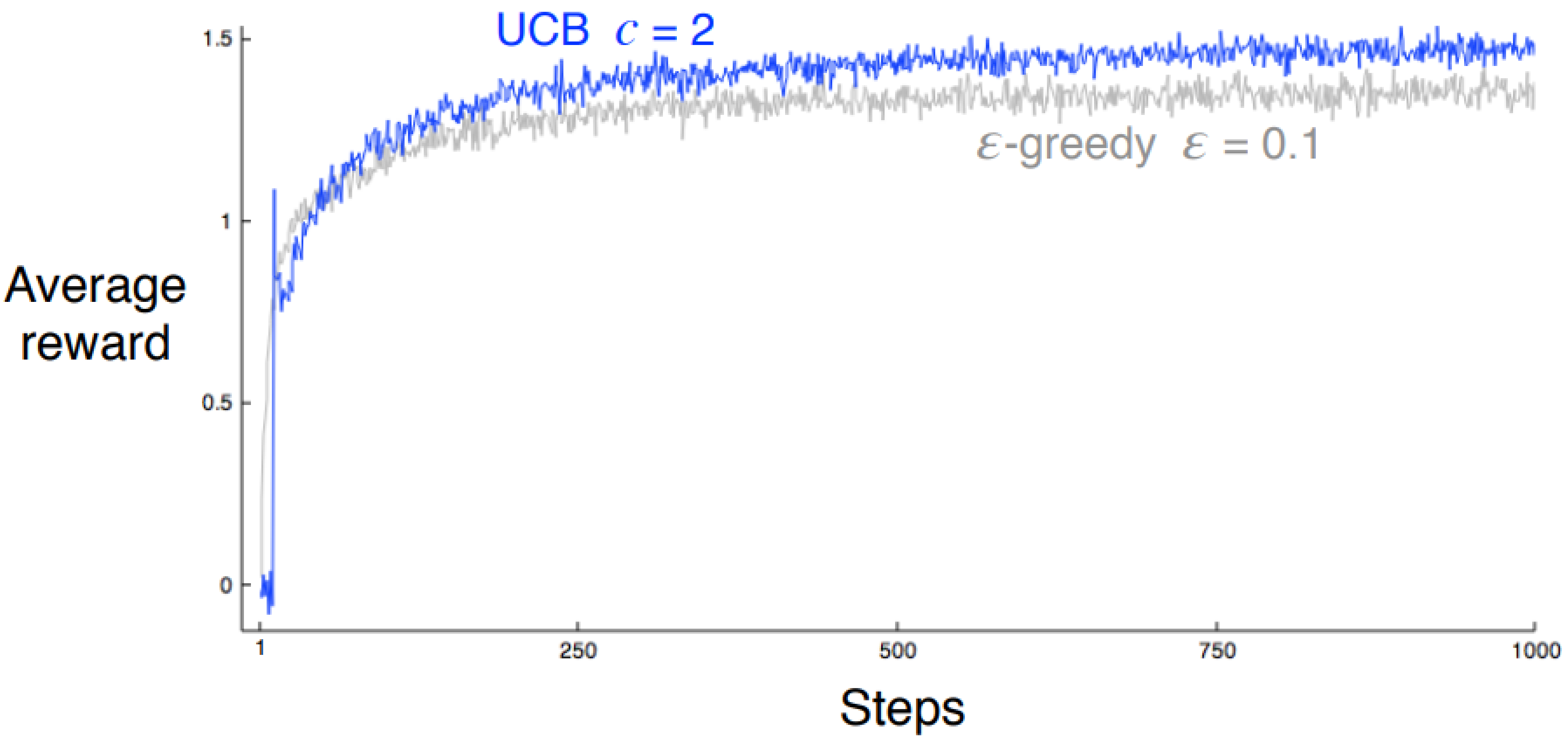
Upper Confidence Bound (UCB)

- Exploration needed because there is uncertainty about the accuracy of action value estimates
- Greedy looks good now, maybe not the best
- ε-Greedy explores, but randomly
- Would be better to select among non-greedy actions that may be close to optimal, with some measure for how certain we are about their values
- The UCB selection method does this

UCB

$$A_t \doteq \underset{a}{\arg\max} \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$$

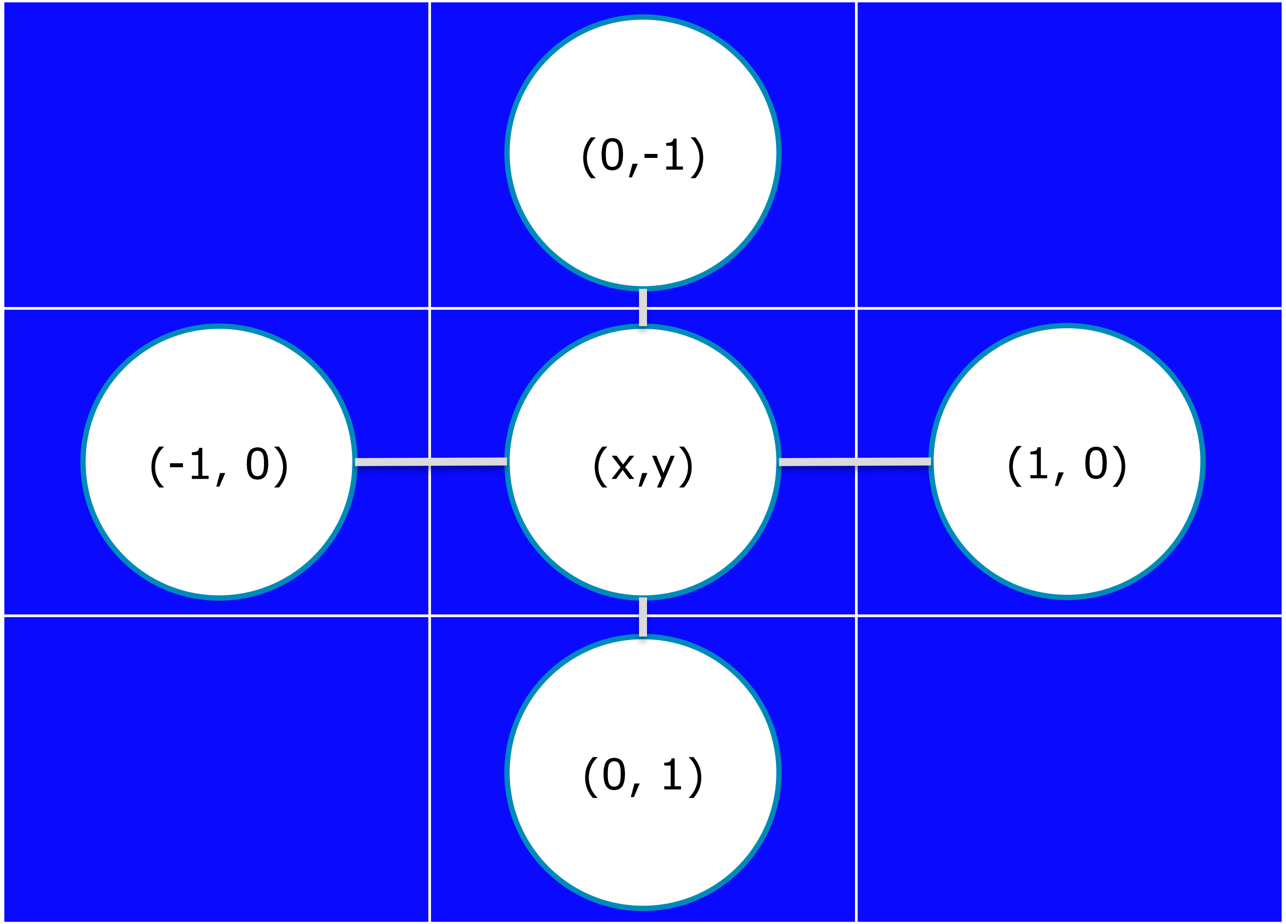
- ln(t) = natural log
- $N_t(a)$ = number of times a was selected
- c > 0 controls the degree of exploration
- Typically, all values tried once first
- sqrt term is the uncertainty of value estimate for action a
- Function calculates a sort of 'upper bound' on the possible true value of action a
- As $N_t(a)$ increases, the uncertainty goes down
- When t increases but not Nt(a), uncertainty goes up


UCB $c = 2$
ε-greedy $\varepsilon = 0.1$
Average reward
1.5
1
0.5
0
1
250
500
750
1000
Steps

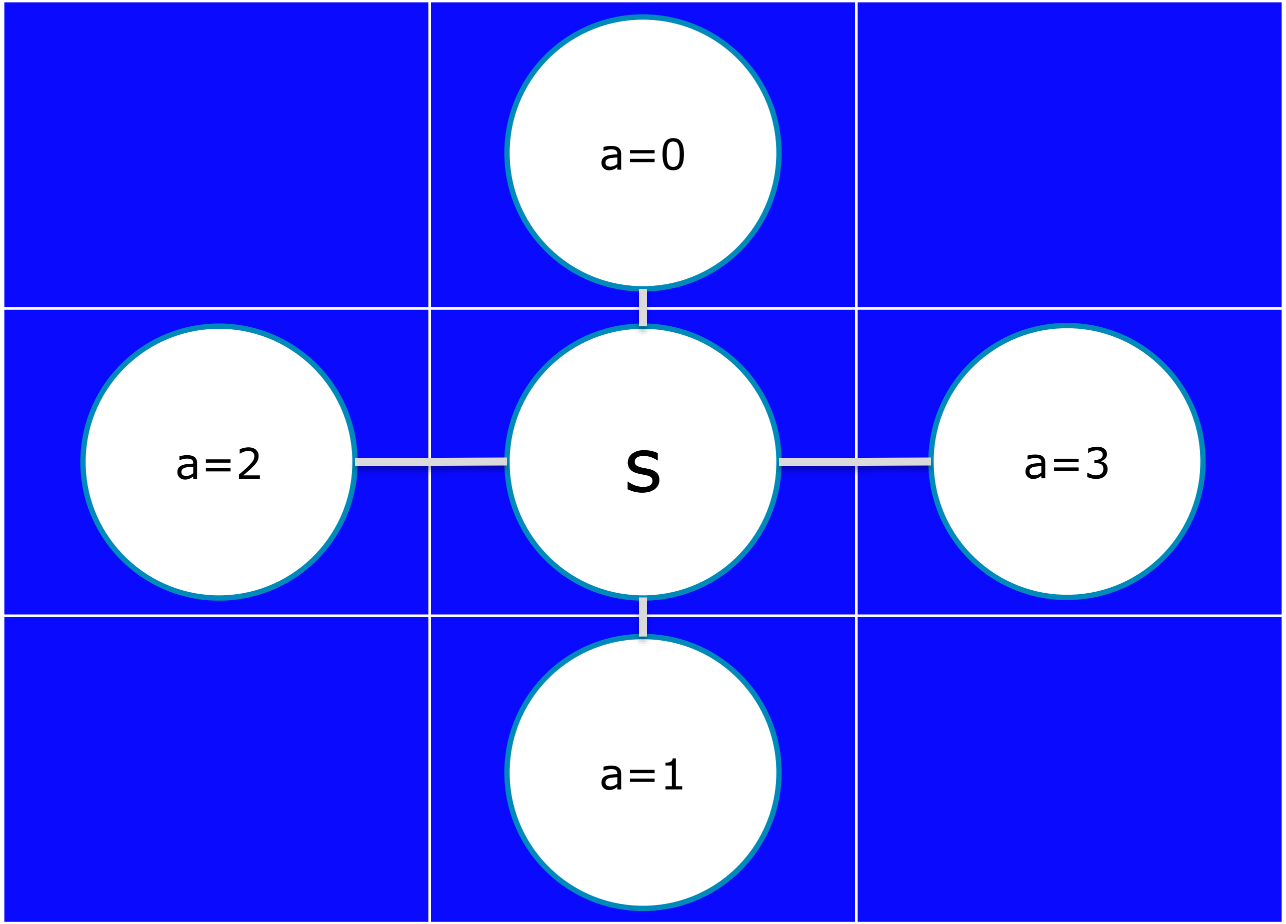
Q(a) vs Q(s,a)

- Q(a) = value of doing action **a**
- The value of a specific action will vary depending on the state it was issued
- For example: Moving up is good if the goal is up, but not if the goal is down
- Q(s,a) = value of action **a** at state **s**

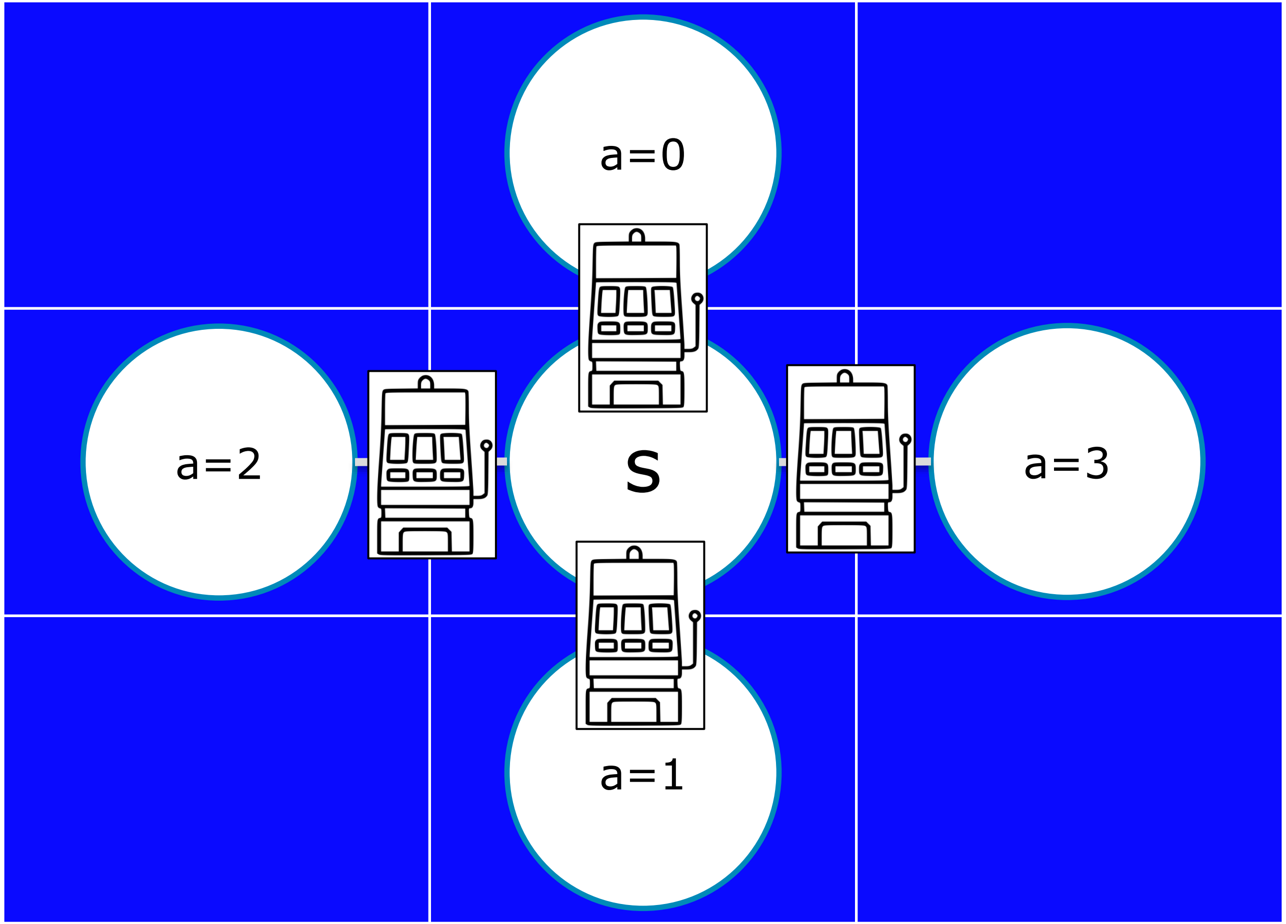
Action
[x,y]



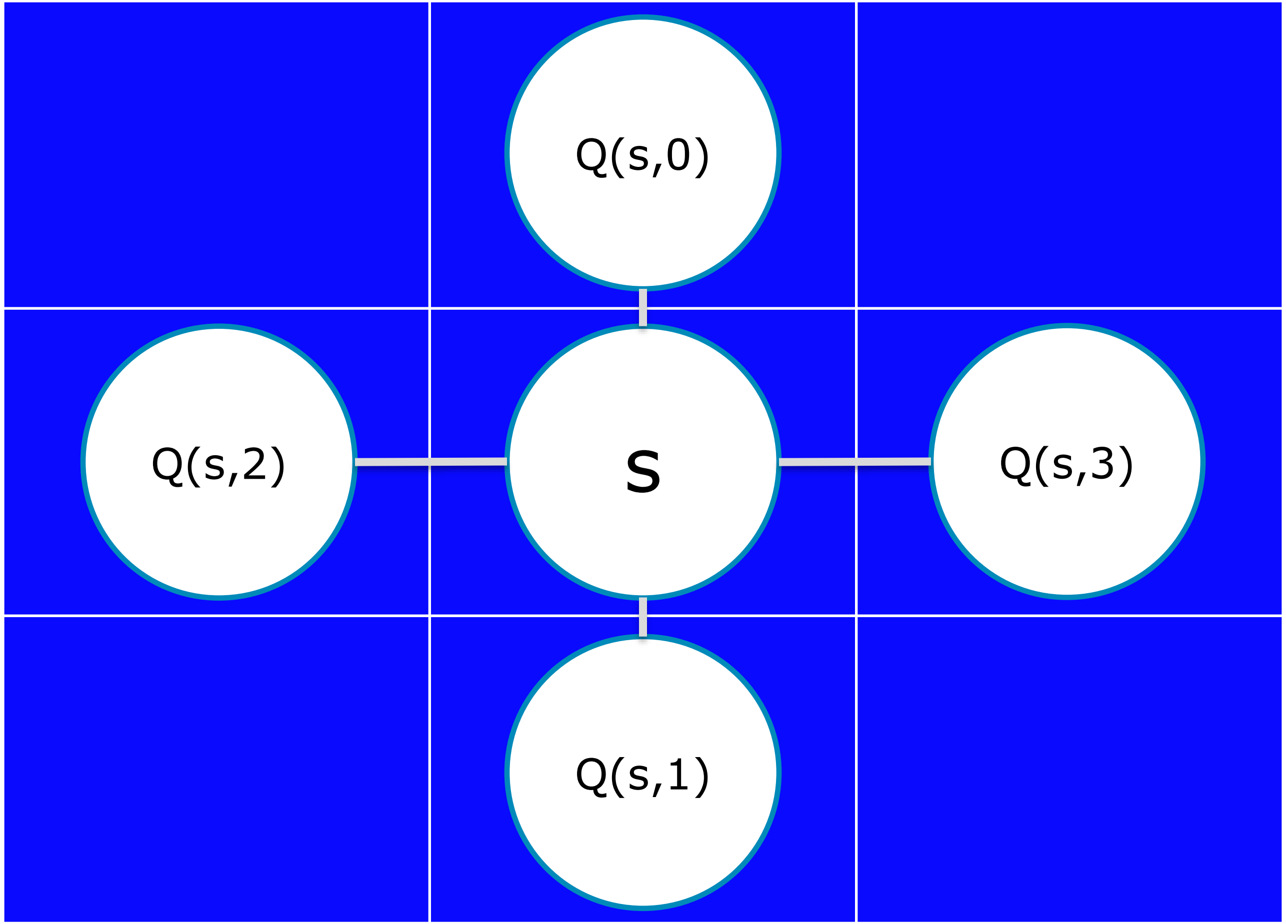
Action
N



Action N



Values Q(s,a)



Exam Questions

- Formula for Average, Incremental update
  - Be able to do an example like in slides
- Exploitation vs Exploration
- Action Selection Methods
  - Greedy, Epsilon Greedy, UCB
  - Effect of Epsilon as it goes up or down
- Bandit Algorithm