CSCI E-82a
Probabilistic Programming and AI
Lecture 11
Monte Carlo Reinforcement Learning

Steve Elston

HARVARD
Extension School

Introduction to Monte Carlo Reinforcement Learning

- What is Monte Carlo RL?
- Review of Monte Carlo sampling
- Monte Carlo state-value estimation
- Monte Carlo RL algorithms
- Monte Carol policy improvement

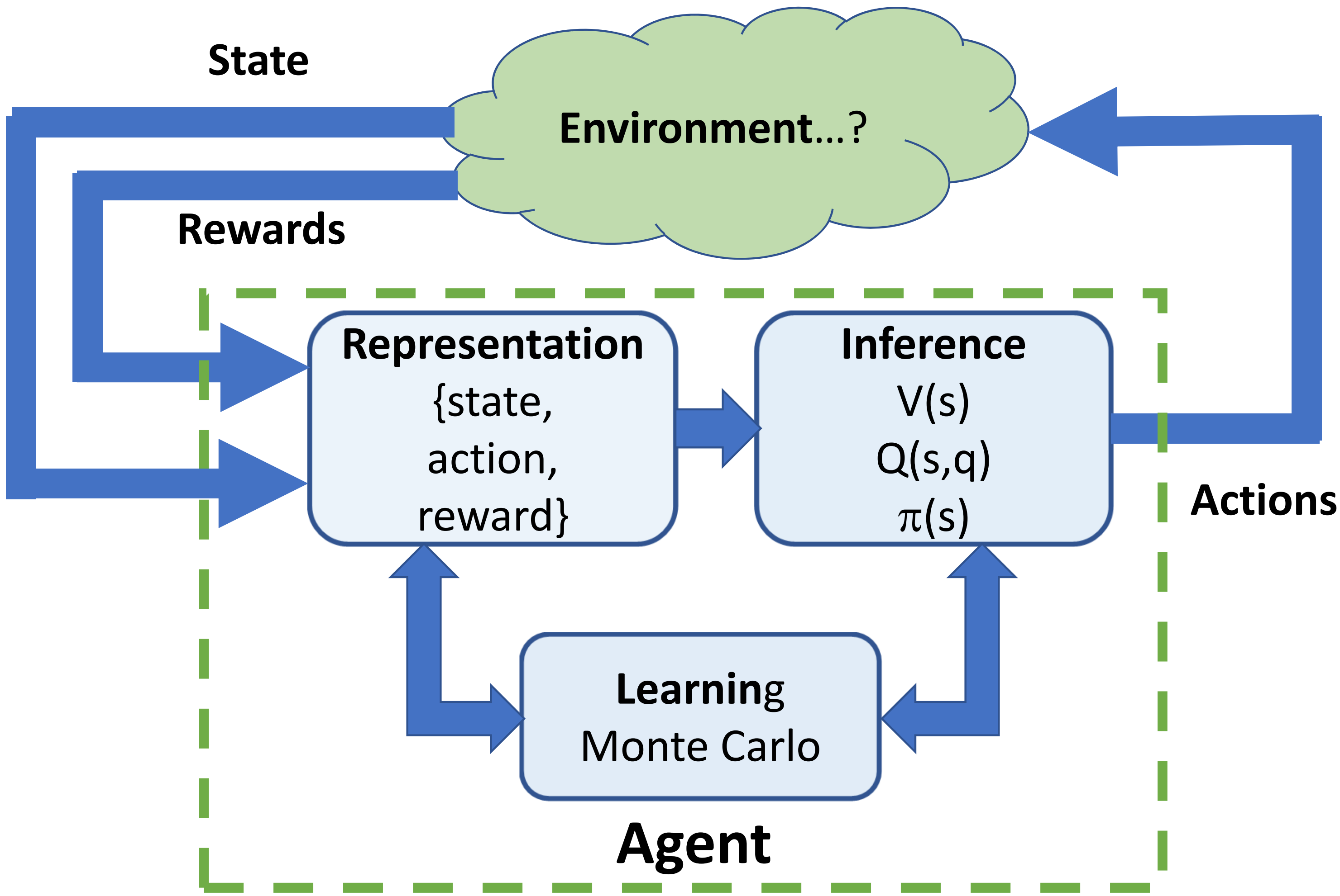
What is Monte Carlo Reinforcement Learning

- Reinforcement learning is **model free**
  - No specified model
  - Learn from experience; rewards
  - Learn state-value for evaluation
  - Learn action-value for control
- Monte Carlo agents takes samples of the values
  - Update average values with new samples
- Monte Carlo agents **must complete episodes**
  - Can only update values once episode terminates
- Monte Carlo RL is often used as a reference for performance of other algorithms

Introduction to Monte Carlo Reinforcement Learning

Model Type	Model?	State	Labeled Data	Loss Function
Supervised Learning	Yes	No	Yes	Error Metric
Unsupervised Learning	Yes	No	No	Error Metric
Bandit Agent	No	No	No	Reward
Dynamic Programming	Yes	Yes	No	Reward
Reinforcement Learning	No	Yes	No	Reward

The Reinforcement Learning Agent



Introduction to Monte Carlo Reinforcement Learning

Model Type	**Backup Type**	**Bootstrap**	**On/Off-Line**
Bandit Agent	None	No	Online
Dynamic Programming	Full	Yes	Offline
Monte Carlo RL	Complete	No	Offline

Review of Monte Carlo Sampling

- Monte Carlo methods **randomly sample**
- Repetitive sampling creates a **Markov chain**
- Sample values are averaged
- Convergence of sample estimates by the **weak law of large numbers**

Review of Monte Carlo Sampling

- Sample estimates converge by the weak law of large numbers
- For **expected value** of underlying distributed, μ, use sample estimate of the mean

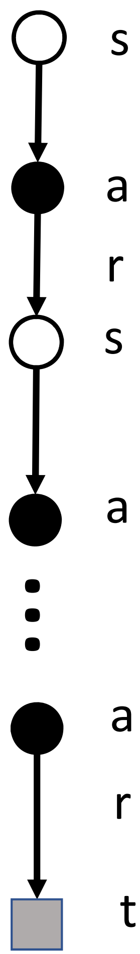
$$\bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i$$

Then by the weak law of large numbers

$$\bar{X} \rightarrow E(X) = \mu$$

as, $n \rightarrow \infty$

Monte Carlo State Value Estimation – Policy Evaluation



- The backup diagram aids understand the **MC RL state-value estimation** algorithm
- MC sampling algorithm:
    1. Start in state, s
    2. Take action, a, based on policy, π
    3. Record reward, r
    4. Transition to next state
    5. Repeat above 2-4, until terminal state, t
- MC value estimates are averaged over episodes
- MC algorithms **do not bootstrap**
    - **Complete backup**
    - **Strong convergence properties**
    - **High variance**
    - **Algorithm cannot work online**

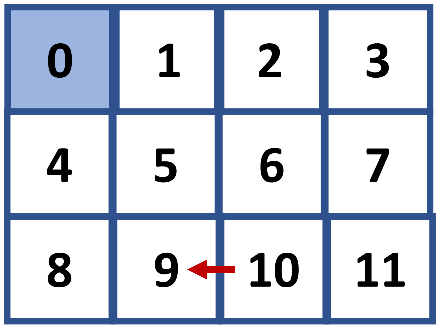
Monte Carlo State Value Estimation

- Upon termination of the Markov chain, compute return

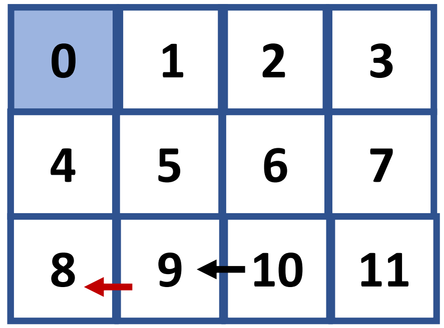
$$G_t = R_{t+1} + R_{t+2} + \ldots = R_T = \sum_{k=0}^{T} R_{t+k+1}$$

- Process is episodic so do not need to discount
- Average G_t over episodes for each state visited

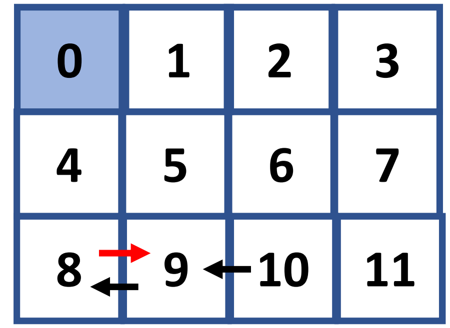
Monte Carlo State Value Estimation



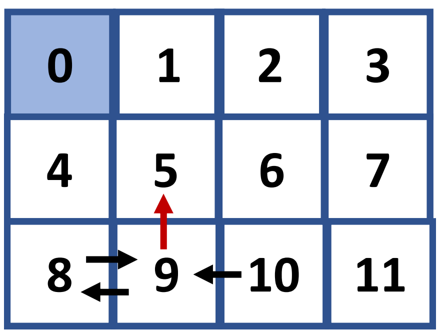
$\Sigma_t \; r_{10,t} = r_{10\text{-}9}$



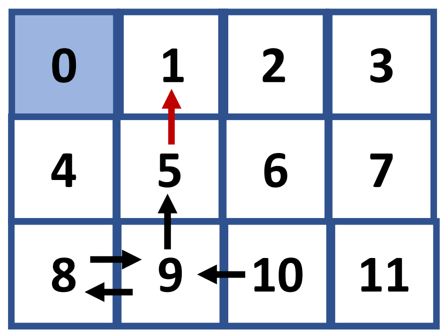
$\Sigma_t \; r_{10,t} = \Sigma_{t\text{-}1} \; r_{t\text{-}1} + r_{9\text{-}8}$



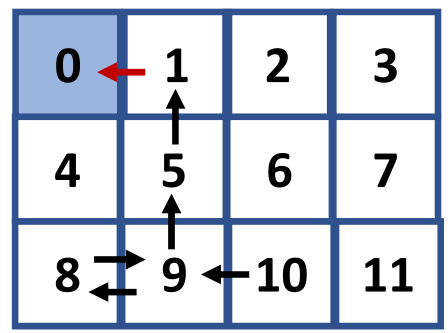
$\Sigma_t \; r_{10,t} = \Sigma_{t\text{-}1} \; r_{t\text{-}1} + r_{8\text{-}9}$



$\Sigma_t \; r_{10,t} = \Sigma_{t\text{-}1} \; r_{t\text{-}1} + r_{9\text{-}5}$



$\Sigma_t \; r_{10,t} = \Sigma_{t\text{-}1} \; r_{t\text{-}1} + r_{5\text{-}1}$



$\Sigma_t \; r_{10,t} = \Sigma_{t\text{-}1} \; r_{t\text{-}1} + r_{1\text{-}0}$

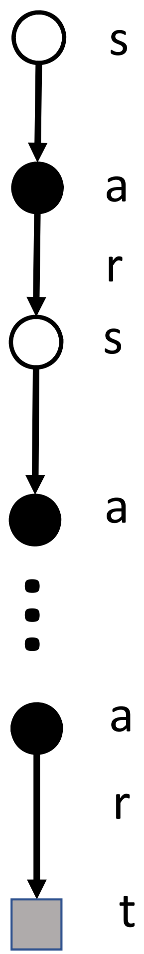
Monte Carlo RL Algorithms

- Return from Markov chain:

$$G_t = R_{t+1} + R_{t+2} + \ldots = R_T = \sum_{k=0}^{T} R_{t+k+1}$$

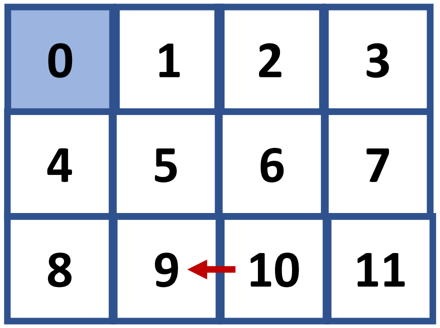
- **Bias-variance trade-off**
- MC gives an **unbiased estimate** of G_t
- What about the variance of G_t?
  - Recall **Weak Law of Large** numbers: $\bar{X} \to E(X) = \mu$, as $n \to \infty$
  - A **infinitely large sample required** to obtained **low variance** estimate
  - Estimate with **finite sample** will have **high variance!**

Monte Carlo RL Algorithms

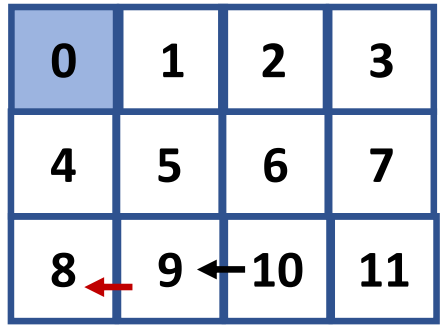


- Where to start the Markov chain?
  - From a **specific stating state**
  - **Random start** – e.g. Bernoulli sample
  - Random start samples entire environment
  - We primarily use random start
- Two possible sampling methods:
  - **First visit Monte Carlo** estimates returns from rewards of the first visit to a state in an episode
  - **Every visit Monte Carlo** accumulates the rewards for any visit to a state in an episode
- Use first-visit MC in this course

First-Visit Monte Carlo Algorithm

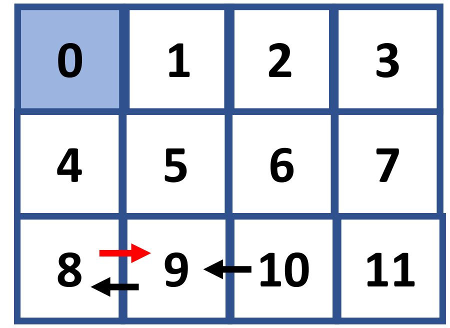


$\sum_t r_{10,t} = r_{10\text{-}9}$



$\sum_t r_{10,t} = \sum_{t-1} r_{t-1} + r_{9\text{-}8}$

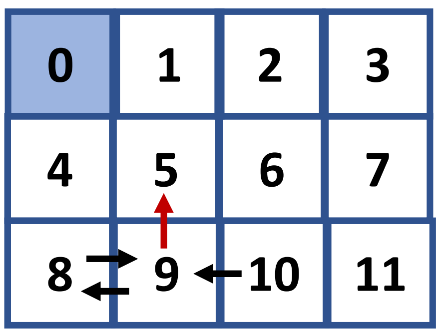
$\sum_t r_{9,t} = r_{9\text{-}8}$



$\sum_t r_{10,t} = \sum_{t-1} r_{t-1} + r_{8\text{-}9}$

…

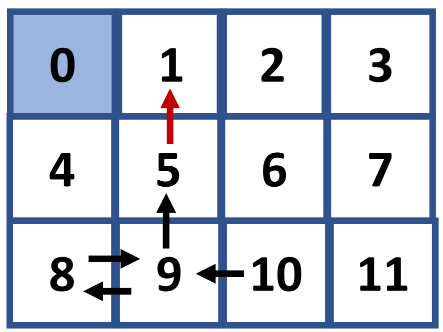
$\sum_t r_{8,t} = r_{8\text{-}9}$



$\sum_t r_{10,t} = \sum_{t-1} r_{t-1} + r_{9\text{-}5}$

…

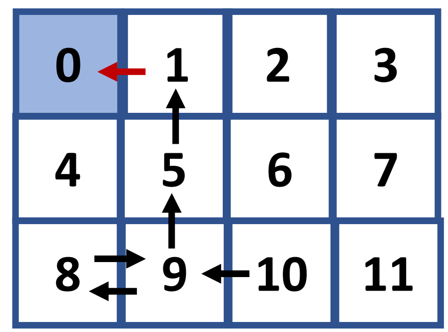
$\sum_t r_{8,t} = r_{9\text{-}5}$



$\sum_t r_{10,t} = \sum_{t-1} r_{t-1} + r_{5\text{-}1}$

…

$\sum_t r_{5,t} = r_{5\text{-}1}$



$\sum_t r_{10,t} = \sum_{t-1} r_{t-1} + r_{1\text{-}0}$

…

$\sum_t r_{1,t} = r_{1\text{-}0}$

Monte Carlo Policy Improvement - Control

- How to perform **policy improvement** with Monte Carlo algorithms?
- Policy improvement can be **on-policy** or **off-policy**
  - Basic **Monte Carlo control** is **on-policy** and updates the policy used for control
  - In **off policy Monte Carlo control**, agent follows a **behavior policy** and updates a **target policy**
- In this course we focus on on-policy MC control
- Off policy MC control more complicated
  - Learn from behavior policy
  - Requires **importance sampling**
  - Has significant bias
  - More on off-policy control in future lectures

Monte Carlo Policy Improvement - Control

- How to perform **policy improvement** with Monte Carlo algorithms?
- Recall the **action-value** the **policy improvement theorem**:

$$q_*(s,\ a) >= q_\pi(s,\ a)\ \forall\ \pi$$

Where

$$q_*(s, a) = max_\pi q(s, a)$$

- The optimal policy may **not be unique**

Monte Carlo Policy Improvement - Control

- Monte Carlo **policy improvement, or control, samples action-values**, q(s,a)
- Rewards are accumulated for each action, a, from each state, s, following policy, π(s,a)
- At end of episode return for each action, a, from each state, s, are computed
- Action values are averaged over visits to state-action
- After a specified number of episodes, the policy is updated
  - Greedy improvement
  - ε-greedy improvement
- Above steps may be repeated

Exploitation vs. Exploration

- The agent following a **greedy policy** maximizes short-term reward
- But, the greedy policy may not be optimal
  - Learning is stochastic
  - Action-value is **high variance** MC sample
  - There is always uncertainty in learned parameters
  - May be a better policy
- Improve policy by mixing **greedy exploitation** with **random exploration**

Monte Carlo Policy Improvement - Control

- Update policy with ε-greedy improvement to find improved policy π_{k+1} at *k*th step of algorithm

$$q_{\pi_{k+1}}(a \mid s) = \begin{cases} Greedy\ improvement\ with\ p = 1 - \epsilon \\ Random\ action\ with\ p = \epsilon \end{cases}$$

$$= \begin{cases} \max_a q_{\pi_k}(a \mid s)\ with\ p = 1 - \epsilon \\ a \sim\ Bernoulli\ with\ p = \epsilon \end{cases}$$

- Iterate until convergence – small change in policy evaluation
- Result is an **ε-greedy policy**
  - ε is small number; 0.05, 0.01, 0.001…..
  - Decrease ε as learning progresses: policy becomes greedier