Lecture 4: Recap and More on Model Free Methods and Approximation

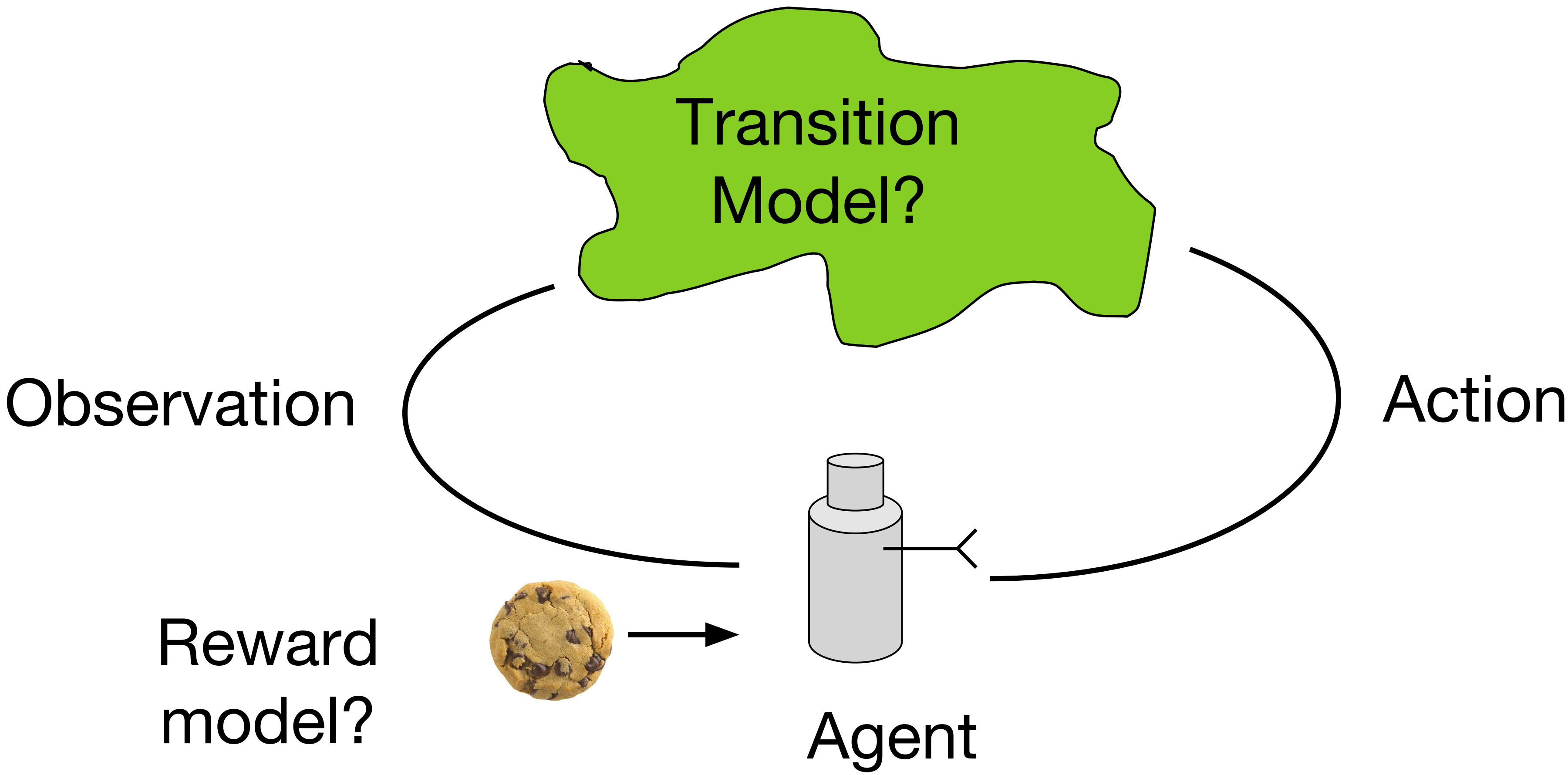
CS234: RL

Emma Brunskill

Spring 2017

Much of the content for this lecture is borrowed from Ruslan Salakhutdinov's class, Rich Sutton's class and David Silver's class on RL.

Reinforcement Learning



Goal: Maximize expected sum of future rewards

Reinforcement Learning

Learn to make good sequences of decisions

1st: MDP Planning

Compute Optimal Policy when Models of Dynamics/Rewards are Known

Learn to make good sequences of decisions

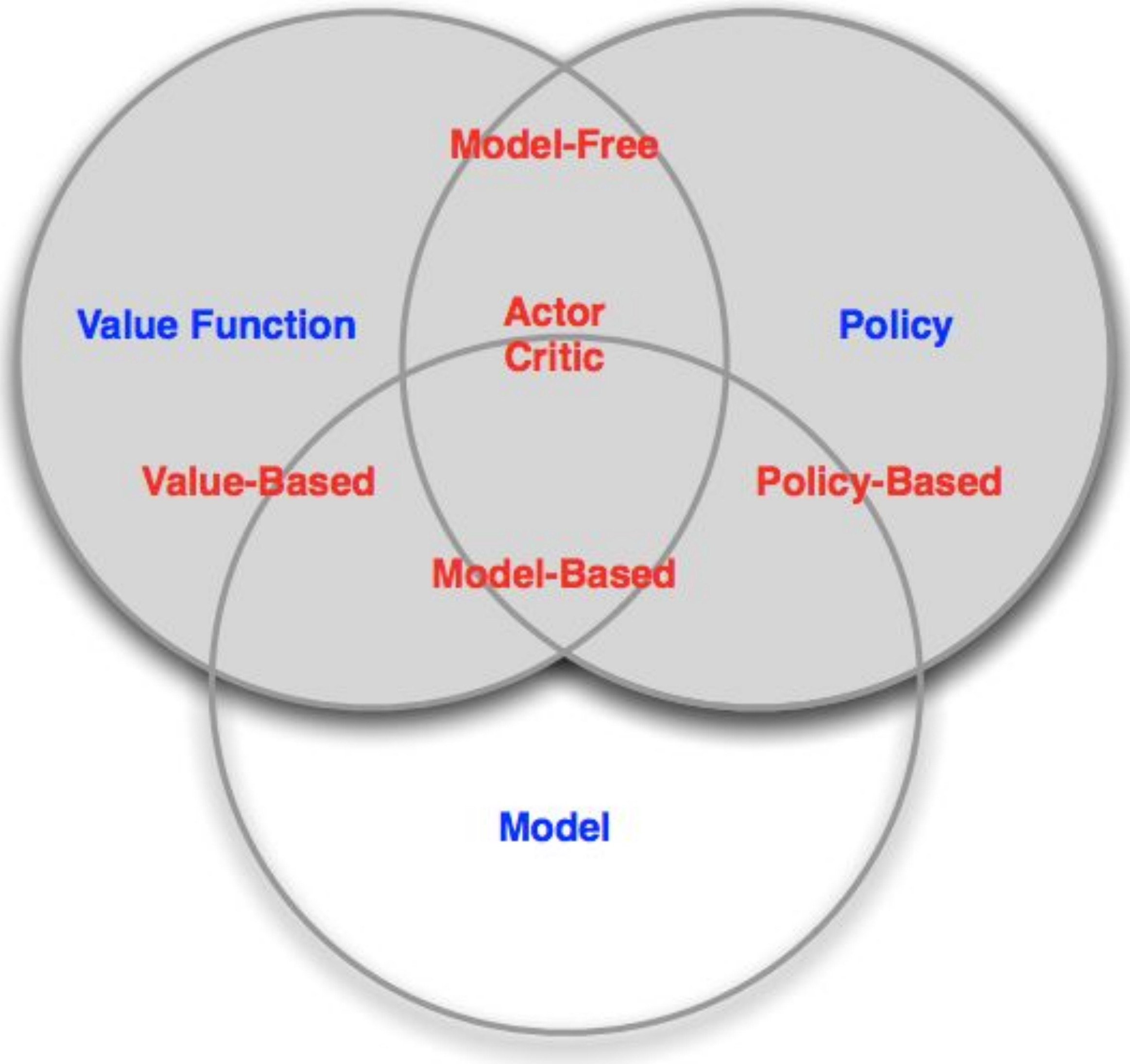
- Bellman equation
- Value iteration
- Policy iteration

2nd: Basic Reinforcement Learning with Lookup Tables

Learn to make good sequences of decisions

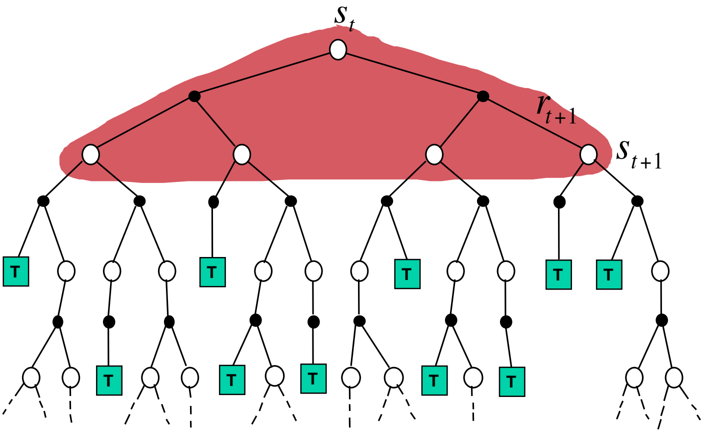
- Model based
  - Estimate dynamics & reward & do MDP planning
- Model free
  - Q-learning
  - Monte Carlo evaluation

2nd: Basic RL with Lookup Tables



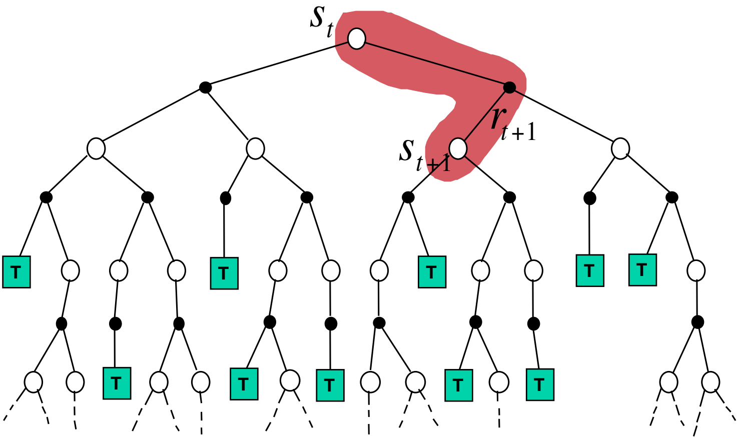
Dynamic Programming Backup

$$V(S_t) \leftarrow \mathbb{E}_\pi \left[R_{t+1} + \gamma V(S_{t+1})\right]$$



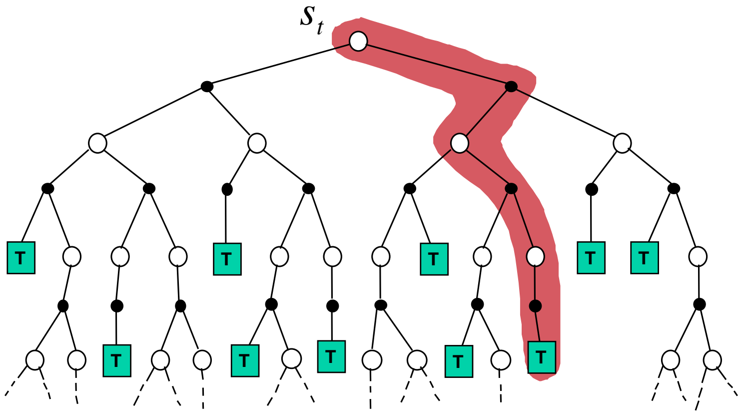
Temporal-Difference Backup

$$V(S_t) \leftarrow V(S_t) + \alpha \left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right)$$

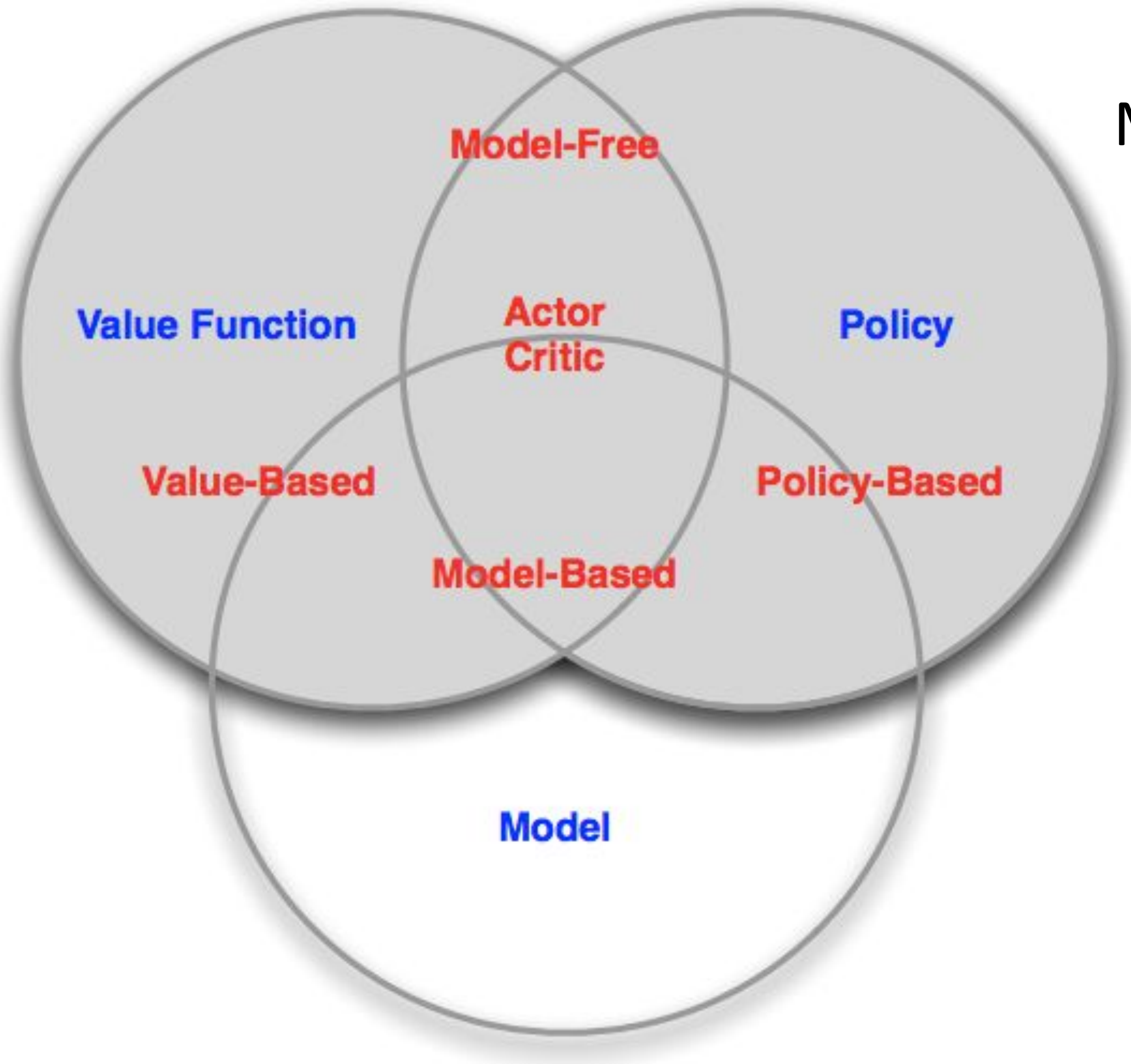


Monte-Carlo Backup

$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t - V(S_t) \right)$$



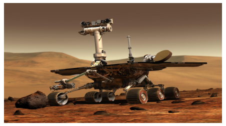
2nd: Basic RL with Lookup Tables



Q-learning
MC methods

Bootstrapping and Sampling

- Bootstrapping: update involves an estimate
  - MC does not bootstrap
  - DP bootstraps
  - TD bootstraps
- Sampling: update samples an expectation
  - MC samples
  - DP does not sample
  - TD samples

S1	S2	S3	S4	S5	S6	S7
Okay Field Site +1						Fantastic Field Site +10

- Policy: TryLeft (TL) in all states, use γ=1, H=4
- Start in state S3, take TryLeft, get r=0, go to S2
- Start in state S2, take TryLeft, get r=0, go to S2
- Start in state S2, take TryLeft, get r=0, go to S1
- Start in state S1, take TryLeft, get r=+1, go to S1
- Trajectory = (S3,TL,0,S2,TL,0,S2,TL,0,S1,TL,1,S1)
- First visit MC estimate of all states?
- Every visit MC estimate of S2?

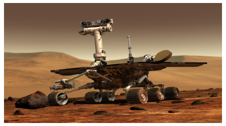
$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-1} R_T$$

  - Every-Visit MC: average returns for every time s is visited in an episode
  - First-visit MC: average returns only for first time s is visited in an episode

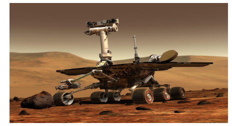
$$V_{samp}(s) = r + \gamma V^{\pi}(s')$$

- TD estimate of all states (init at 0)

$$V^{\pi}(s) = (1 - \alpha) V^{\pi}(s) + \alpha V_{samp}(s)$$

S1	S2	S3	S4	S5	S6	S7
Okay Field Site +1						Fantastic Field Site +10

- Policy: TryLeft (TL) in all states, use (discount) ϒ=1
- 1 episode yielded trajectory = (S3,TL,0,S2,TL,0,S2,TL,0,S1,TL,1,S1)
- First visit MC estimate of all states (assume 0 if unvisited) **[1 1 1 0 0 0 0]**
- TD estimate of all states (init at 0) **[α 0 0 0 0 0 0]**

S1	S2	S3	S4	S5	S6	S7
Okay Field Site +1						Fantastic Field Site +10

- Policy: TryLeft (TL) in all states, use (discount) γ=1
- 1 episode yielded trajectory = (S3,TL,0,S2,TL,0,S2,TL,0,S1,TL,1,S1)
- First visit MC estimate of all states (assume 0 if unvisited) **[1 1 1 0 0 0 0]**
- TD estimate of all states (init at 0) **[α 0 0 0 0 0 0]**
- **What if did updates not once, but many times?**
  - Repeatedly do TD updates on past tuples (pick any tuple and do update)
  - Repeatedly do MC on past episodes (pick any episode and do update)
- **What would be the resulting MC estimate in this case?**
- **What about the TD estimate? Try doing TD on the following tuples:**
  - **(**S3,TL,0,S2,TL,0,S2,TL,0,S1,TL,1,S1) (see solution above, start from there)
  - **then** (S2,TL,0,S2), (S1,TL,1,SL), (S1,TL,1,SL), (S1,TL,1,SL),
  - What are the resulting TD estimates?
- **Bonus: Does the order in which one samples prior (s,a,r,s') tuples impact the convergence rate of TD? If yes, what is a good order? Does it depend on alpha?**

In Limit of Repeated Updates

- Equivalence of model-based and TD-learning for policy evaluation with lookup table representations

Certainty Equivalence

- MC converges to solution with minimum mean-squared error
  - Best fit to the observed returns

$$\sum_{k=1}^{K} \sum_{t=1}^{T_k} \left(G_t^k - V(s_t^k)\right)^2$$

  - In the AB example, $V(A) = 0$
- TD(0) converges to solution of max likelihood Markov model
  - Solution to the MDP $\langle \mathcal{S}, \mathcal{A}, \hat{\mathcal{P}}, \hat{\mathcal{R}}, \gamma \rangle$ that best fits the data

$$\hat{\mathcal{P}}_{s,s'}^{a} = \frac{1}{N(s,a)} \sum_{k=1}^{K} \sum_{t=1}^{T_k} \mathbf{1}(s_t^k, a_t^k, s_{t+1}^k = s, a, s')$$

$$\hat{\mathcal{R}}_{s}^{a} = \frac{1}{N(s,a)} \sum_{k=1}^{K} \sum_{t=1}^{T_k} \mathbf{1}(s_t^k, a_t^k = s, a) r_t^k$$

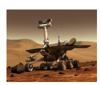
  - In the AB example, $V(A) = 0.75$

3rd:
Generalization in Model-free RL

Learn to make good sequences of decisions… in really large state spaces

- Model free
  - Q-learning
  - Monte Carlo evaluation

Scaling Up

S1	S2	S3	S4	S5	S6	S7
Okay Field Site +1						Fantastic Field Site +10

- Want to be able to tackle problems with enormous or infinite state spaces
- Tabular representation is insufficient

Linear Value Function Approximation (VFA)

- Represent value function by a linear combination of features

$$\hat{v}(S, \mathbf{w}) = \mathbf{x}(S)^\top \mathbf{w} = \sum_{j=1}^{n} \mathbf{x}_j(S)\mathbf{w}_j$$

- Objective function is quadratic in parameters w

$$J(\mathbf{w}) = \mathbb{E}_\pi\left[\left(v_\pi(S) - \mathbf{x}(S)^\top \mathbf{w}\right)^2\right]$$

- Update rule is particularly simple

$$\nabla_{\mathbf{w}} \hat{v}(S, \mathbf{w}) = \mathbf{x}(S)$$

$$\Delta \mathbf{w} = \alpha(v_\pi(S) - \hat{v}(S, \mathbf{w}))\mathbf{x}(S)$$

- Update = step-size × prediction error × feature value
- Later, we will look at the neural networks as function approximators.

Monte Carlo with VFA

- Return G_t is an unbiased, noisy sample of true value $v_\pi(S_t)$
- Can therefore apply supervised learning to "training data":

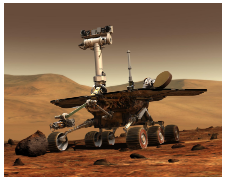
$$\langle S_1, G_1 \rangle, \langle S_2, G_2 \rangle, ..., \langle S_T, G_T \rangle$$

- For example, using linear Monte-Carlo policy evaluation

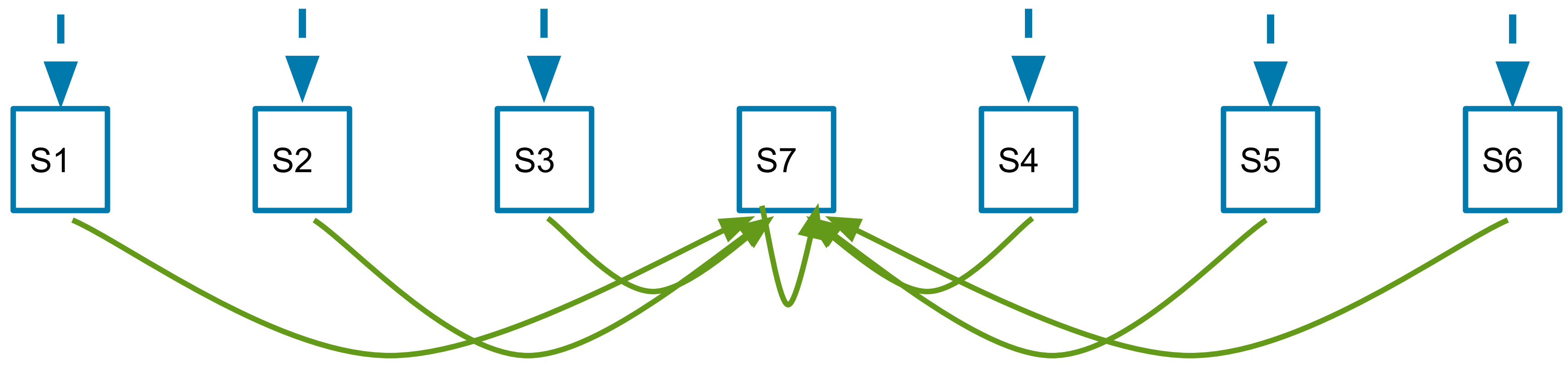
$$\begin{aligned} \Delta \mathbf{w} &= \alpha(G_t - \hat{v}(S_t, \mathbf{w}))\nabla_{\mathbf{w}} \hat{v}(S_t, \mathbf{w}) \\ &= \alpha(G_t - \hat{v}(S_t, \mathbf{w}))\mathbf{x}(S_t) \end{aligned}$$

- Monte-Carlo evaluation converges to a local optimum

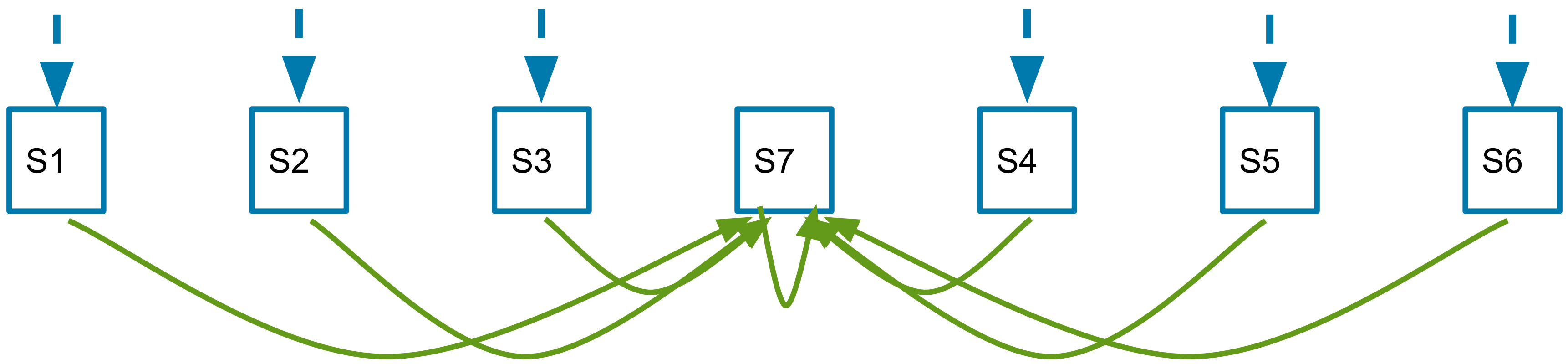
Mars Rover in a Boring Land



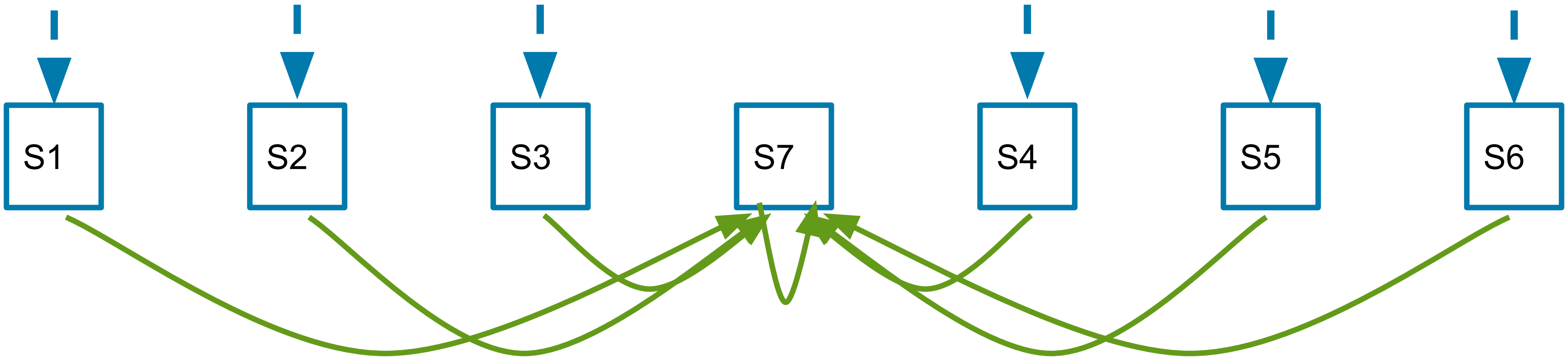
- 0 reward everywhere
- Slightly different dynamics
- Example from Baird



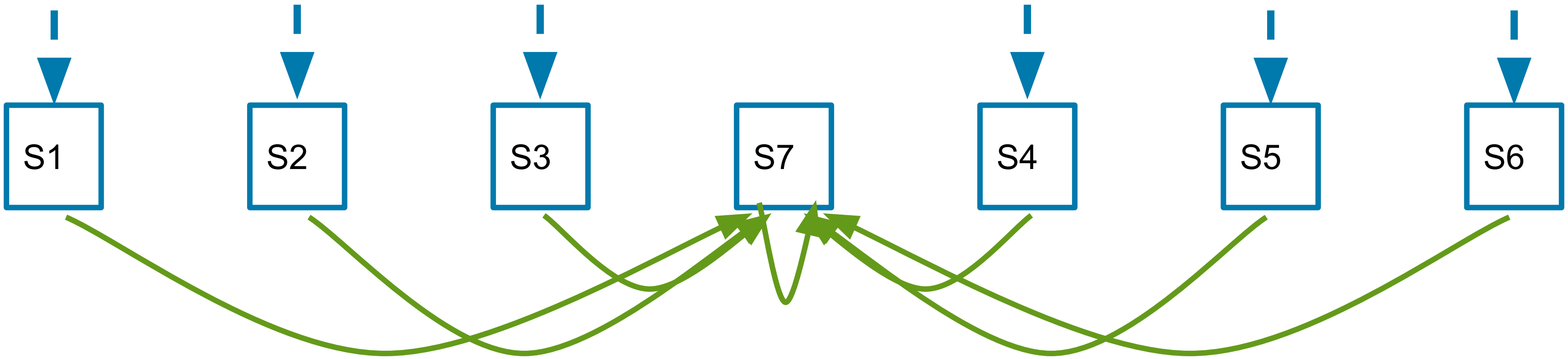
- a1 takes to states S1...S6 with probability 0.99/6
- a2 goes to state S7 with probability .99
- with prob 0.01 go to a terminal state s8 & episode ends
- Reward is 0 everywhere



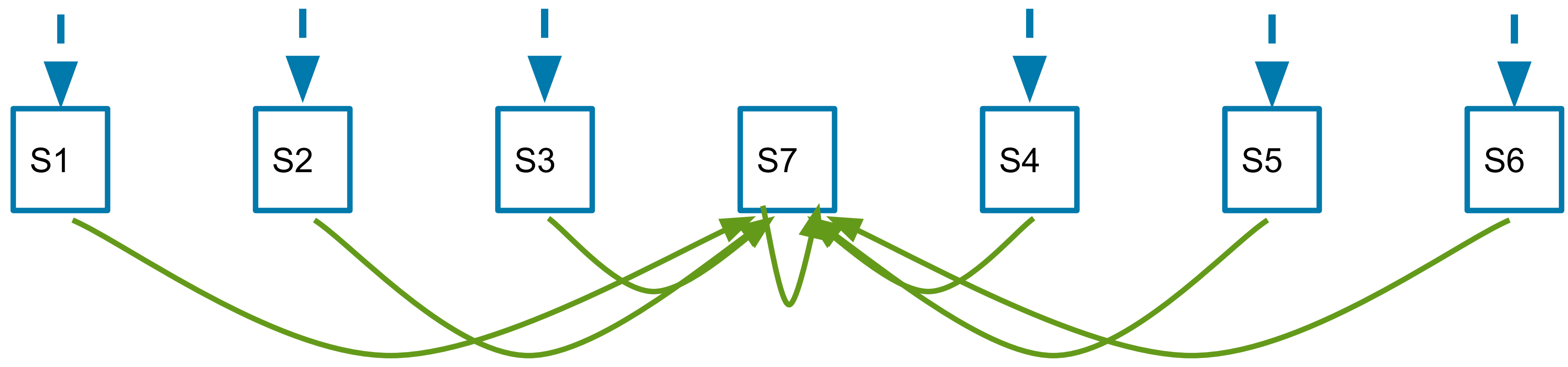
- a1 takes to states S1...S6 with probability 0.99/6
- a2 goes to state S7 with probability .99
- with prob 0.01 go to a terminal state s8 & episode ends
- Reward is 0 everywhere
- V~(s) = Σ_i w_i f_i (s)
- Gradient of V~ wrt w_i = f_i (s)



- a1 takes to states S1...S6 with probability 0.99/6
- a2 goes to state S7 with probability .99
- with prob 0.01 go to a terminal state s8 & episode ends
- Reward is 0 everywhere
- $V\sim(s) = \Sigma_i w_i f_i (s)$, Gradient of $V\sim$ wrt $w_i = f_i (s)$
- Consider feature representation of 8 features
  - State s1 = [2 0 0 0 0 0 0 1] ...
  - State s6 = [0 0 0 0 0 2 0 1]
  - State s7 = [0 0 0 0 0 0 1 2]



- a1 takes to states S1...S6 with probability 0.99/6
- a2 goes to state S7 with probability .99
- with prob 0.01 go to a terminal state s8 & episode ends
- Reward is 0 everywhere
- $V\sim(s) = \Sigma_i\, w_i\, f_i\,(s)$, Gradient of $V\sim$ wrt $w_i = f_i\,(s)$
- Consider feature representation of 8 features
  - State s1 = [2 0 0 0 0 0 0 1] ...
  - State s6 = [0 0 0 0 0 0 2 0 1]
  - State s7 = [0 0 0 0 0 0 1 2]
  - Gradient for s1 wrt $w_1 = f_1\,(s1)=2$.
  - Gradient for s1 wrt $w_2 = f_2\,(s1)=0$.
  - Gradient for s1 wrt $w_8 = f_8\,(s1)=1$.



- a1 takes to states S1...S6 with probability 0.99/6
- a2 goes to state S7 with probability .99
- with prob 0.01 go to a terminal state s8 & episode ends
- Reward is 0 everywhere
- $V\sim(s) = \Sigma_i w_i f_i(s)$, Gradient of $V\sim$ wrt $w_i = f_i(s)$
- Policy is always take a2.
- One episode: act for 100 timesteps
- Observe [s1,a2,0,s7,a2,0,s7,... 0]
- Consider feature representation of 8 features
  - State s1 = [2 0 0 0 0 0 0 0 1] …
  - State s6 = [0 0 0 0 0 0 2 0 1]
  - State s7 = [0 0 0 0 0 0 0 1 2].
- Assume initial **w** = [1 1 1 1 1 1]
- Can true value of V^{π} be represented with these linear features (e.g. with $V\sim$)?
- What is initial $V\sim(s1)$?
- What is new weight vector using MC with VFA?
- Imagine we get the same trajectory many times. What will we converge to?

MC with VFA
Generate episode of length T
for t=0,1,...T-1

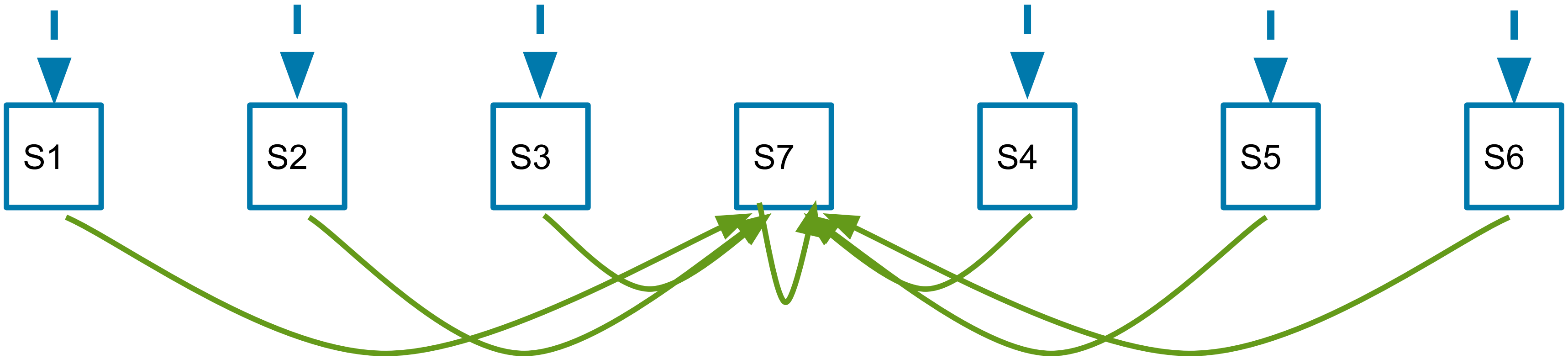
$$\mathbf{w} \leftarrow \mathbf{w} + \alpha[G_t - V\sim(s_t,\mathbf{w})]\mathbf{f}(s_t)$$

TD Learning with VFA

Semi-gradient TD(0) for estimating $\hat{v} \approx v_\pi$

Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S}^+ \times \mathbb{R}^n \to \mathbb{R}$ such that $\hat{v}(\text{terminal},\cdot) = 0$

Initialize value-function weights $\boldsymbol{\theta}$ arbitrarily (e.g., $\boldsymbol{\theta} = \mathbf{0}$)
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        Choose $A \sim \pi(\cdot|S)$
        Take action A, observe R, S'
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\big[R + \gamma\hat{v}(S',\boldsymbol{\theta}) - \hat{v}(S,\boldsymbol{\theta})\big]\nabla\hat{v}(S,\boldsymbol{\theta})$
        $S \leftarrow S'$
    until S' is terminal



- a1 takes to states S1...S6 with probability 0.99/6
- a2 goes to state S7 with probability .99
- with prob 0.01 go to a terminal state s8 & episode ends
- Reward is 0 everywhere
- $V\sim(s) = \Sigma_i w_i f_i(s)$, Gradient of V wrt $w_i = f_i(s)$
- Policy is always take a2.
- One episode: act for 100 timesteps
- Observe [s1,a2,0,s7,a2,0,s7,... 0]
- Consider feature representation of 8 features
  - State s1 = [2 0 0 0 0 0 0 0 1] …
  - State s6 = [0 0 0 0 0 0 2 0 1]
  - State s7 = [0 0 0 0 0 0 0 1 2].
- Assume initial **w** = [1 1 1 1 1 1], Υ=1
- What is new weight vector using TD learning with VFA at end of episode?
- Imagine we get the same trajectory many times. What will we converge to?

TD(0) with VFA
For each episode
Initialize S
Repeat for each step of the episode
Choose a, observe r, s'
w ← **w** + α[r + Υ V~(s') - V~(s,**w**)]**f**(s)

Impact of Selected Features

- Crucial
- Features affect
  - **How well can approximate the optimal V / Q**
    - **Approximation error**
  - Memory
  - Computational complexity

If We Can Represent Optimal V/ Q Can We Always Converge to It?

Feature Selection

1. Use domain knowledge
2. Use a very flexible set of features & regularize
    - Supervised learning problem!
    - Success of deep learning inspires application to RL
    - With additional challenge that have to gather data