Introduction to Reinforcement Learning

Artificial Intelligence for Games

Denis Zavadski

Overview

- Introduction
- Model Based Learning
- Model free Learning
  - Monte Carlo
  - Temporal Difference
- Function Approximation

Motivation

- Training without a supervisor, only with reward
- Feedback not always instantly received
- Acting in environments, where actions have an impact on subsequent data

  → Non i.i.d. data

Definitions

- Reward R_t , (G_t)
- Action $a_t \in A$
- State $s_t \in S$
- Model
- MDP
- Policy π
- Value Function v
- Action Value Function q

Definitions

- **Reward** R_t , (G_t)
- Action $a_t \in A$
- State $s_t \in S$
- Model
- MDP
- Policy π
- Value Function v
- Action Value Function q

- scalar feedback
- return G_t

$$G_t = R_t + \gamma R_{t+1} \cdots = \sum_{k=1}^{\infty} \gamma^k R_{t+k}$$

- discount $\gamma \in [0, 1]$

Definitions

- Reward R_t, (G_t)
- **Action** $a_t \in A$
- State $s_t \in S$
- Model
- MDP
- Policy π
- Value Function v
- Action Value Function q

Definitions

- Reward R_t, (G_t)
- Action $a_t \in A$
- **State** $s_t \in S$
- Model
- MDP
- Policy π
- Value Function v
- Action Value Function q

- environment state
- agent state
- Markov State

$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, \ldots, S_t]$$

Definitions

- Reward R_t, (G_t)
- Action $a_t \in A$
- State $s_t \in S$
- **Model**
- MDP
- Policy π
- Value Function v
- Action Value Function q

- predicts environment by
  - predicting next state

$$\mathcal{P}^a_{s's} = \mathbb{P}[s_{t+1} = s' \mid s_t = s, a_t = a]$$

  - predicting next Reward

$$\mathcal{R}^a_s = \mathbb{E}[R'_t \mid s_t = s, a_t = a]$$

- optional

Definitions

- Reward R_t , (G_t)
- Action $a_t \in A$
- State $s_t \in S$
- Model
- **MDP**
- Policy π
- Value Function V
- Action Value Function q

- Markov Decision Process is a formal environment representation
- tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$
- discount $\gamma \in [0, 1]$

Definitions

- Reward R_t, (G_t)
- Action $a_t \in A$
- State $s_t \in S$
- Model
- MDP
- **Policy π**
- Value Function V
- Action Value Function q

- defines agents behaviour
- maps from states to actions
  - Deterministically $\pi(s) = a$
  - Stochastically $\pi(a|s) = \mathbb{P}[a|s]$

Definitions

- Reward R_t , (G_t)
- Action $a_t \in A$
- State $s_t \in S$
- Model
- MDP
- Policy π
- **Value Function *v***
- Action Value Function q

- the state-value-function is the expected return starting from state s and following policy π

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid s]$$

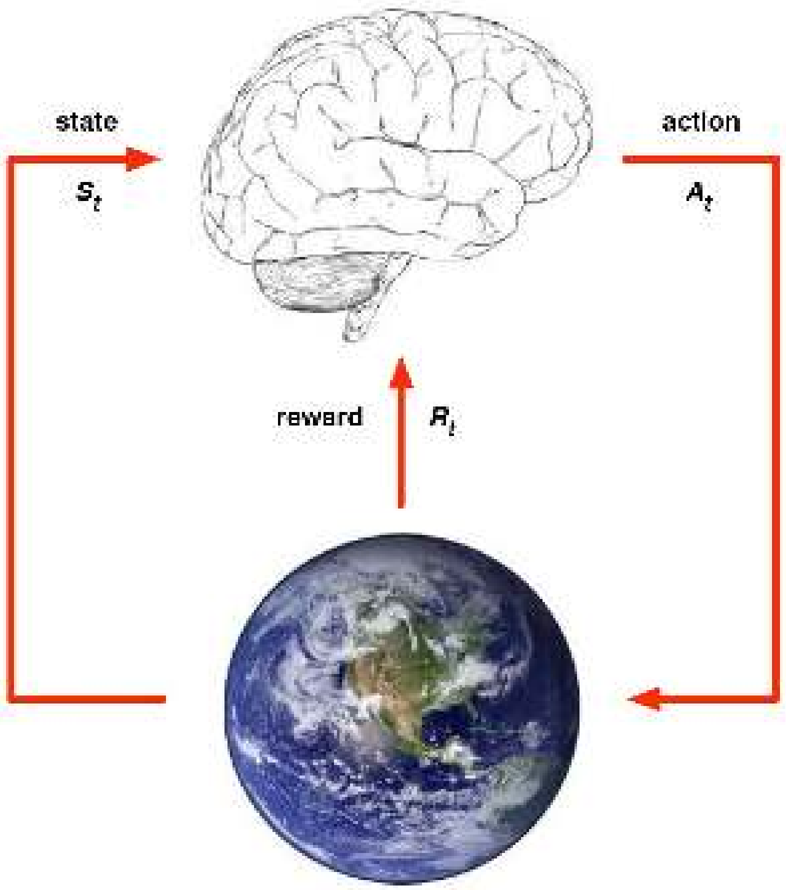
Definitions

- Reward R_t , (G_t)
- Action $a_t \in A$
- State $s_t \in S$
- Model
- MDP
- Policy π
- Value Function v
- **Action Value Function q**

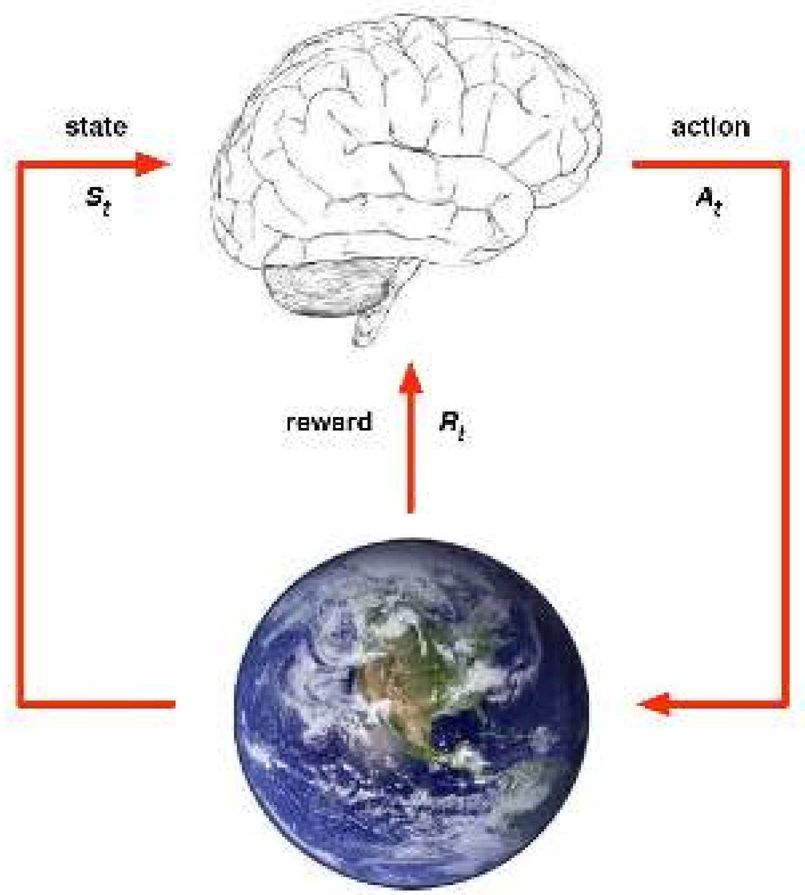
- the action-value-function is the expected return starting from state s taking action a and following policy π

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid s, a]$$

Interaction with Environment



Interaction with Environment



Tasks

- Prediction
  - Evaluating given policy
- Control
  - Finding best policy

Characteristics

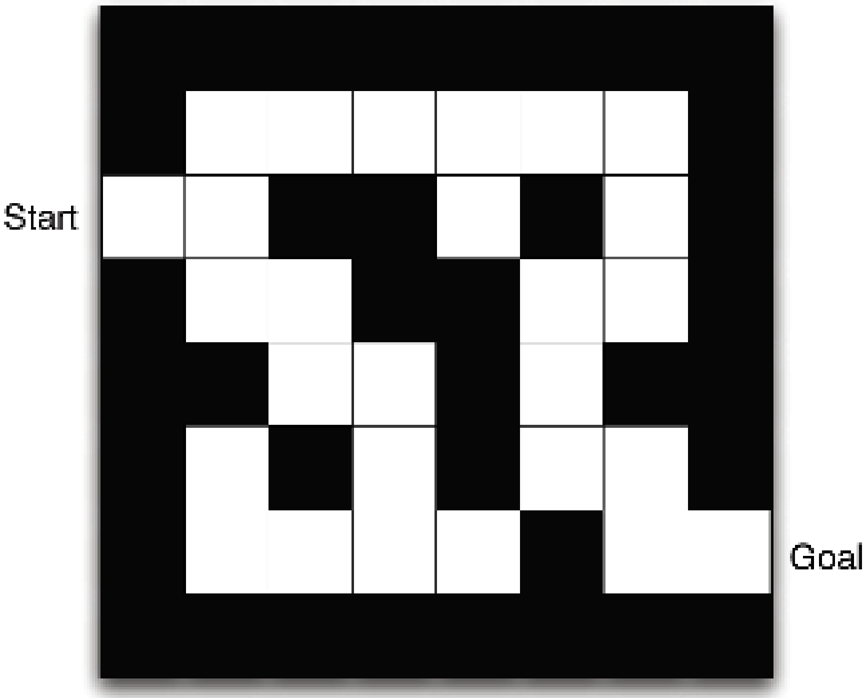
- Value based
- Policy based

- Model based
- Model free

Characteristics

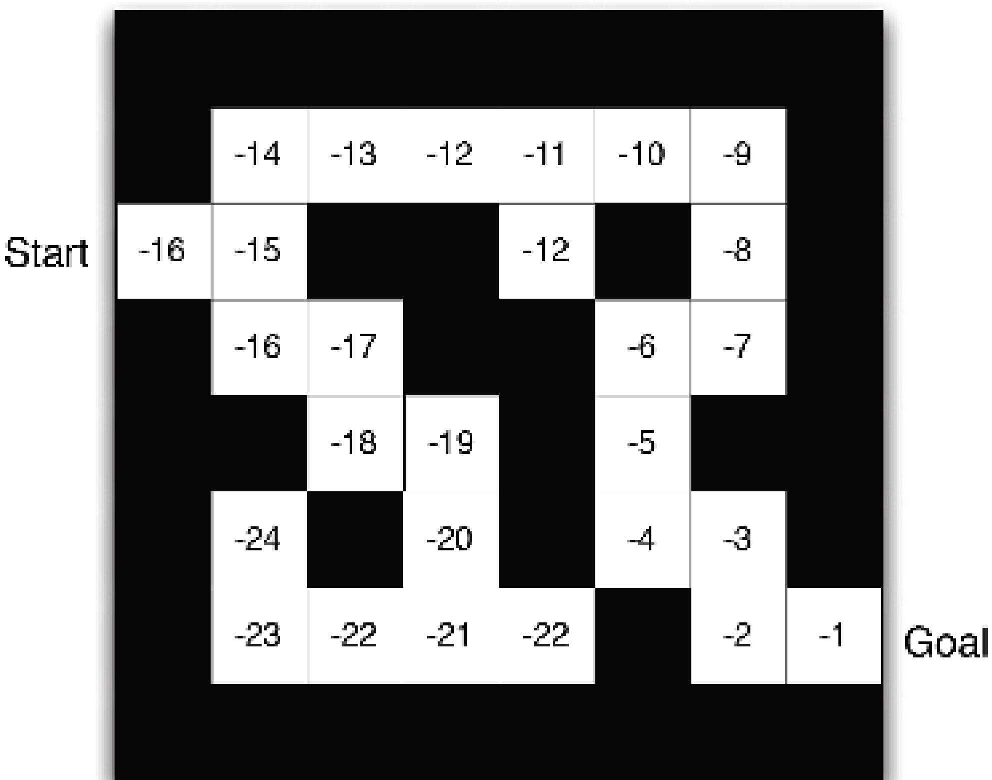
- Value based
- Policy based

- Model based
- Model free

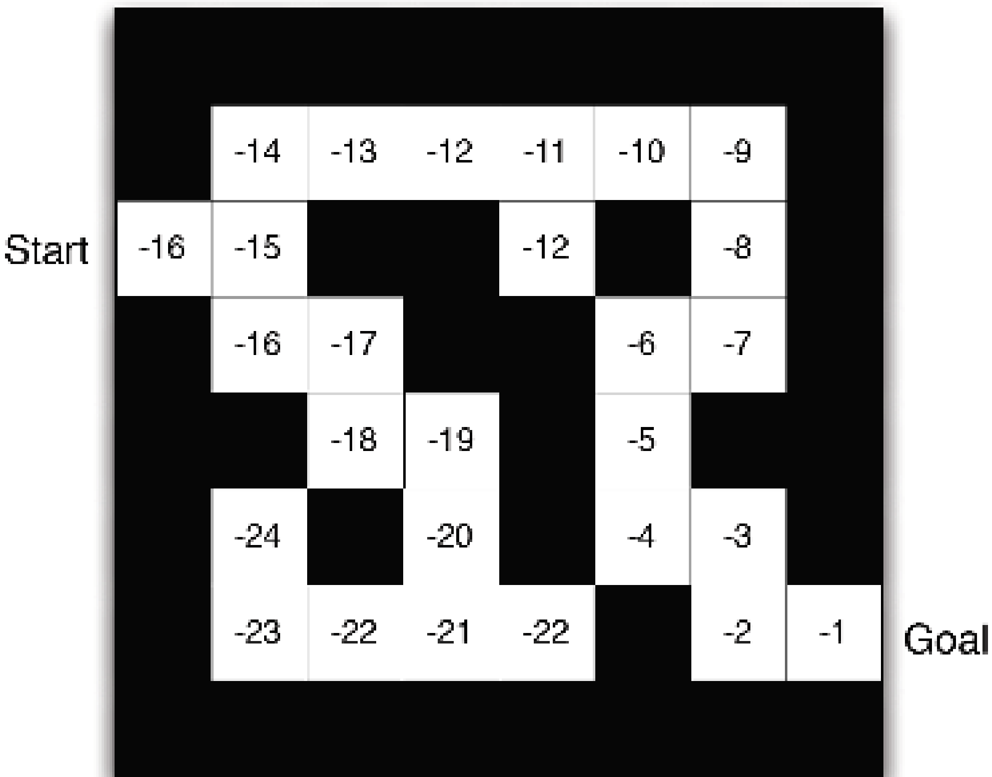


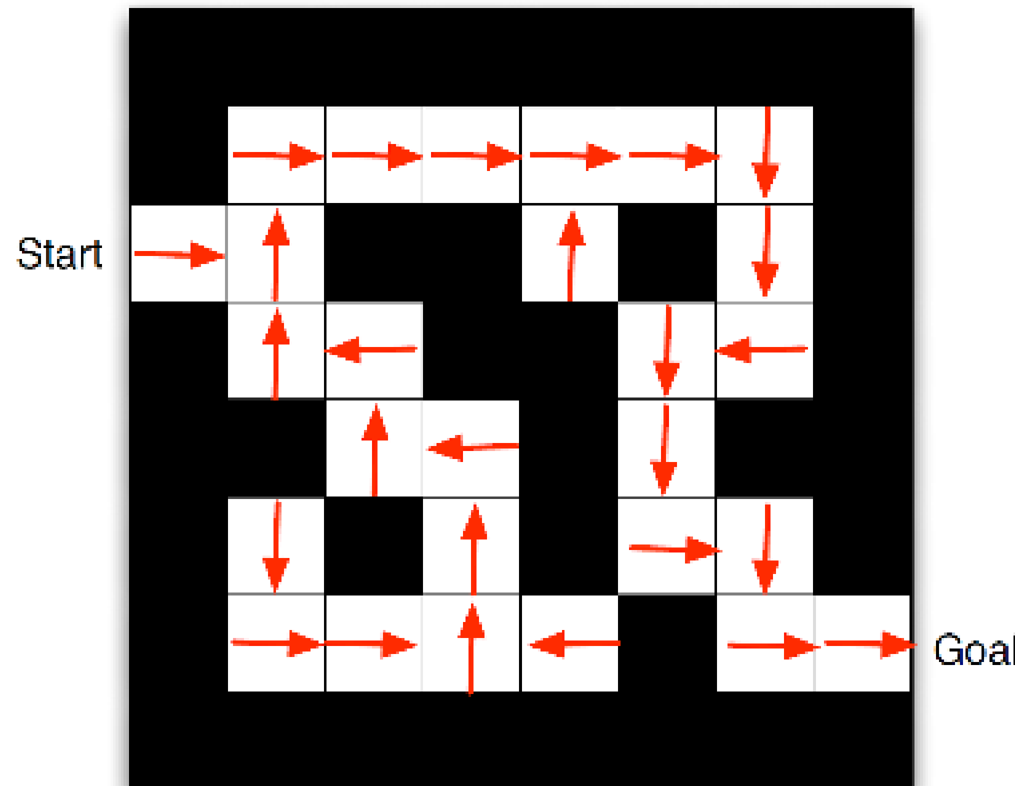
- Rewards: -1/step
- States: location

Value Based vs. Policy Based



Value Based vs. Policy Based





Bellman Equation

$$v(s)_\pi = \mathbb{E}_\pi[G_t \mid s]$$

Bellman Equation

$$
\begin{aligned}
v(s)_\pi =& \mathbb{E}_\pi[G_t \mid s] \\
=& \mathbb{E}_\pi[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots \mid s] \\
=& \mathbb{E}_\pi[R_t + \gamma (R_{t+1} + \gamma R_{t+2} + \dots) \mid s] \\
=& \mathbb{E}_\pi[R_t + \gamma v_\pi(s_{t+1}) \mid s]
\end{aligned}
$$

Bellman Equation

$$v(s)_\pi = \mathbb{E}_\pi[R_t + \gamma v_\pi(s_{t+1}) \mid s]$$

Bellman Equation

$$v(s)_\pi = \mathbb{E}_\pi[R_t + \gamma v_\pi(s_{t+1}) \mid s]$$

- matrix notation

$$v = \mathcal{R} + \gamma \mathcal{P} v$$

Bellman Equation

$$v(s)_\pi = \mathbb{E}_\pi[R_t + \gamma v_\pi(s_{t+1}) \mid s]$$

- matrix notation

$$v = \mathcal{R} + \gamma \mathcal{P} v$$

- solving the equation

$$\begin{aligned} v &= \mathcal{R} + \gamma \mathcal{P} v \\ (\mathbb{I} - \gamma \mathcal{P}) v &= \mathcal{R} \\ v &= (\mathbb{I} - \gamma \mathcal{P})^{-1} \mathcal{R} \end{aligned}$$

Bellman Equation

$$v(s)_\pi = \mathbb{E}_\pi[R_t + \gamma v_\pi(s_{t+1}) \mid s]$$

- matrix notation

$$v = \mathcal{R} + \gamma \mathcal{P} v$$

- solving the equation

$$\begin{aligned} v &= \mathcal{R} + \gamma \mathcal{P} v \\ (\mathbb{I} - \gamma \mathcal{P}) v &= \mathcal{R} \\ v &= (\mathbb{I} - \gamma \mathcal{P})^{-1} \mathcal{R} \end{aligned}$$

- analogous for *q*

Optimal Functions

- Optimal value function

$$v_*(s) = \max_\pi v_\pi(s)$$

- Optimal value action function

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

- Optimal policy

$$\pi_* \geq \pi \quad, \forall \pi$$

Optimal Functions

- Optimal value function

$$v_*(s) = \max_\pi v_\pi(s)$$

- Optimal value action function

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

- Optimal policy

$$\pi_* \geq \pi \quad ,\forall \pi$$

Theorem

- There exists an optimal policy
- The opt. Policy always archieves the optimal state-(action-)functions

Policy Iteration

- evaluate value functions iteratively

$$v^{k+1} = \mathcal{R}^{\pi} + \gamma \mathcal{P}^{\pi} v^{k}$$

- Improve the policy by acting greedily w.r.t. v

$$\pi^{new} = greedy(v_{\pi})$$

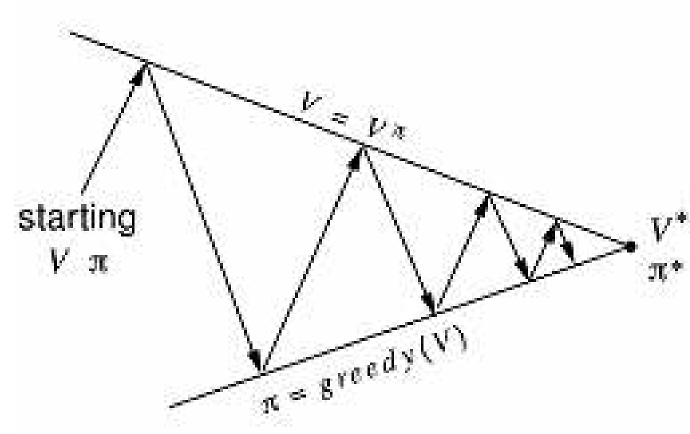
Policy Iteration

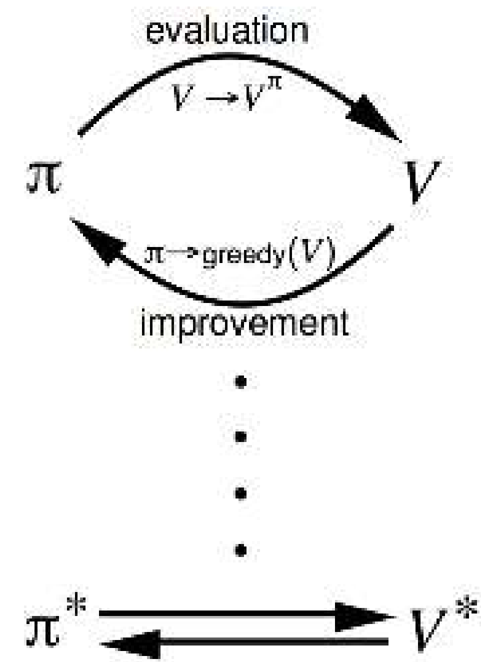
- evaluate value functions iteratively

$$v^{k+1} = \mathcal{R}^{\pi} + \gamma \mathcal{P}^{\pi} v^{k}$$

- Improve the policy by acting greedily w.r.t. v

$$\pi^{new} = greedy(v_{\pi})$$





Value Iteration

- like one-step policy iteration without policy

$$v^{k+1} = \max_a \mathcal{R}^a + \gamma \mathcal{P}^a v^k$$

- pseudo policy is to act greedily

Monte Carlo Policy Evaluation

- is model free
- Learning from experience

Monte Carlo Policy Evaluation

- is model free
- Learning from experience
- Every time-step *t* state *s* is visited in an episode, increment counter $N(s) \leftarrow N(s) + 1$

Monte Carlo Policy Evaluation

- is model free
- Learning from experience
- Every time-step *t* state *s* is visited in an episode, increment counter $N(s) \leftarrow N(s) + 1$
- Increment total return $S(s) \leftarrow S(s) + G_t$

Monte Carlo Policy Evaluation

- is model free
- Learning from experience
- Every time-step *t* state *s* is visited in an episode, increment counter $N(s) \leftarrow N(s) + 1$
- Increment total return $S(s) \leftarrow S(s) + G_t$
- Update $V(s) = S(s) / N(s)$

Monte Carlo Policy Evaluation

- is model free
- Learning from experience
- Every time-step *t* state *s* is visited in an episode, increment counter N(s) ← N(s) +1
- Increment total return
  S(s) ← S(s) + G_t
- Update V(s) = S(s) / N(s)

$$
\begin{aligned}
\mu_k &= \frac{1}{k}\sum_{j=1}^{k} x_j = \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right) \\
&= \frac{1}{k}(x_k + (k-1)\mu_{k-1}) \\
&= \mu_{k-1} + \frac{1}{k}(x_k - \mu_{k-1})
\end{aligned}
$$

Monte Carlo Policy Evaluation

- is model free
- Learning from experience
- Every time-step *t* state *s* is visited in an episode, increment counter N(s) ← N(s) +1
- Increment total return S(s) ← S(s) + G_t
- Update V(s) = S(s) / N(s)

$$\mu_k = \frac{1}{k}\sum_{j=1}^{k} x_j = \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right)$$
$$= \frac{1}{k}(x_k + (k-1)\mu_{k-1})$$
$$= \mu_{k-1} + \frac{1}{k}(x_k - \mu_{k-1})$$

- Update after each episode:

$$N(s_t) \longleftarrow N(s_t) + 1$$
$$v(s_t) \longleftarrow v(s_t) + \frac{1}{N(s_t)}(G_t - v(s_t))$$
$$\longleftarrow v(s_t) + \alpha(G_t - v(s_t))$$

for each state s_t with return G_t

Temporal Difference TD(0)

- Is model free
- Learns from incomplete episodes

Temporal Difference TD(0)

- Is model free
- Learns from incomplete episodes
- Update values $v(s_t)$ towards estimated return $R_t + \gamma v(s_{t+1})$

$$v(s_t) \longleftarrow v(s_t) + \alpha(R_t + \gamma v(s_{t+1}) - v(s_t))$$

Temporal Difference TD(0)

- Is model free
- Learns from incomplete episodes
- Update values *v(s$_t$)* towards estimated return $R_t + \gamma v(s_{t+1})$

$$v(s_t) \longleftarrow v(s_t) + \alpha(R_t + \gamma v(s_{t+1}) - v(s_t))$$

- With TD target $R_t + \gamma v(s_{t+1})$
- And TD error $\delta_t = R_t + \gamma v(s_{t+1}) - v(s_t)$

MC vs. TD

- TD can learn immediately before knowing the data → can learn on incomplete sequences

MC vs. TD

- TD can learn immediately before knowing the data → can learn on incomplete sequences
- MC must wait until sequence is terminated

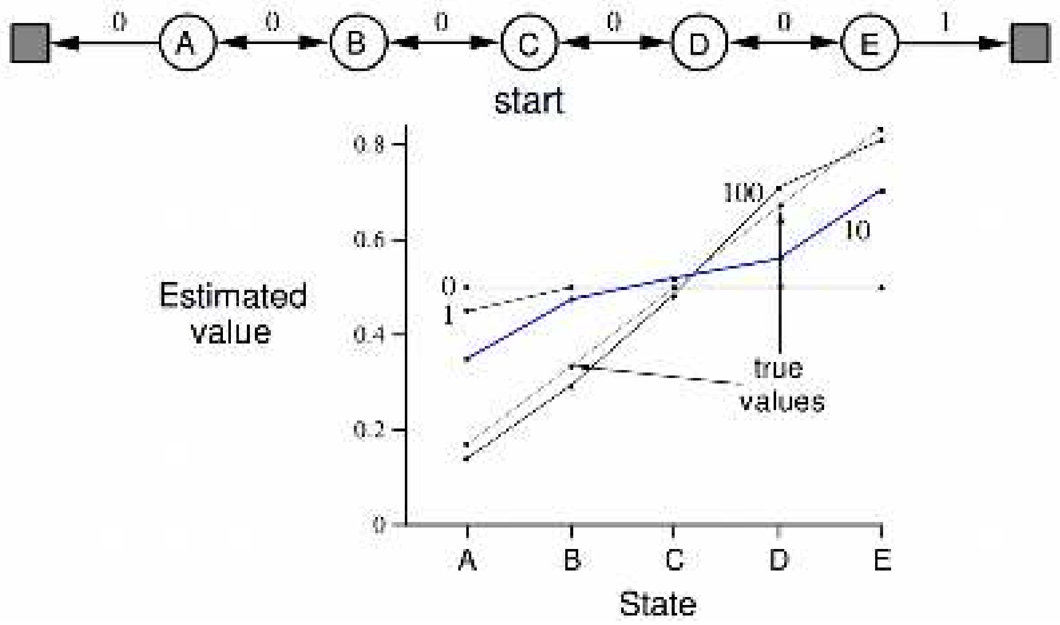
MC vs. TD

- TD can learn immediately before knowing the data → can learn on incomplete sequences
- MC must wait until sequence is terminated
- Both converges to v^*, but differently

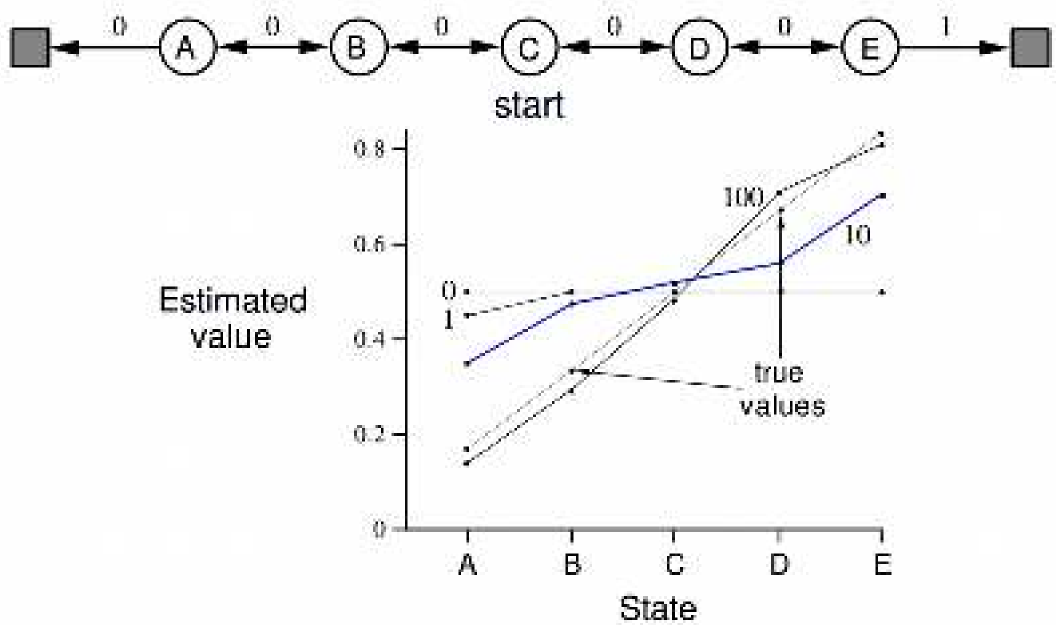
MC vs. TD

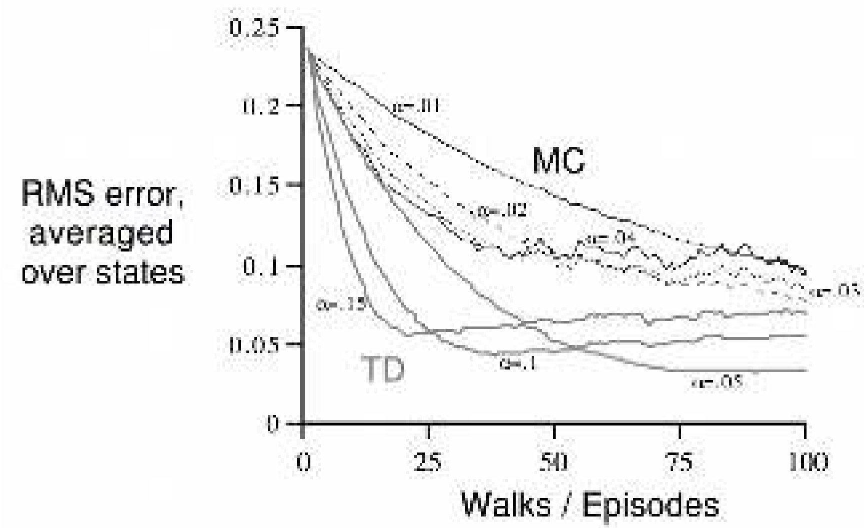
- TD can learn immediately before knowing the data → can learn on incomplete sequences
- MC must wait until sequence is terminated
- Both converges to v^*, but differently
- TD is usually more efficient than MC

Random Walk example



Random Walk example





AB Example

- Two states A,B; no discounting
  8 episodes of experience

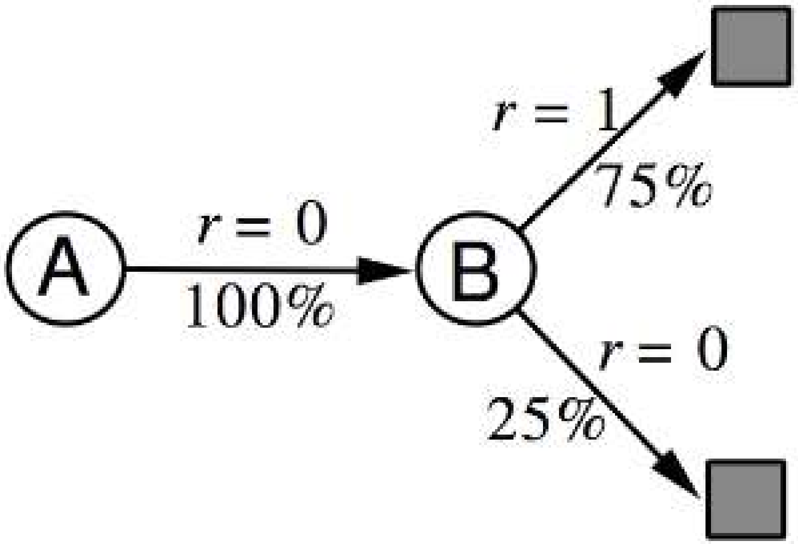
- A , 0, B, 0
  B, 1
  B, 1
  B, 1
  B, 1
  B, 1
  B, 1
  B, 0

- What is v(A), v(B) ?

AB Example

- Two states A,B; no discounting
  8 episodes of experience

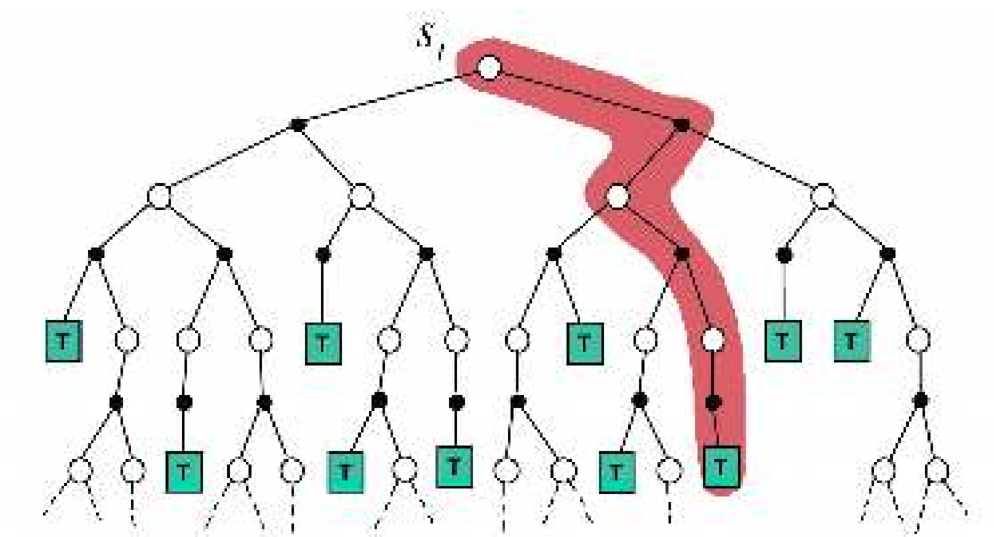
- A , 0, B, 0
  B, 1
  B, 1
  B, 1
  B, 1
  B, 1
  B, 1
  B, 0



- What is v(A), v(B) ?

Updates MC, TD(0)

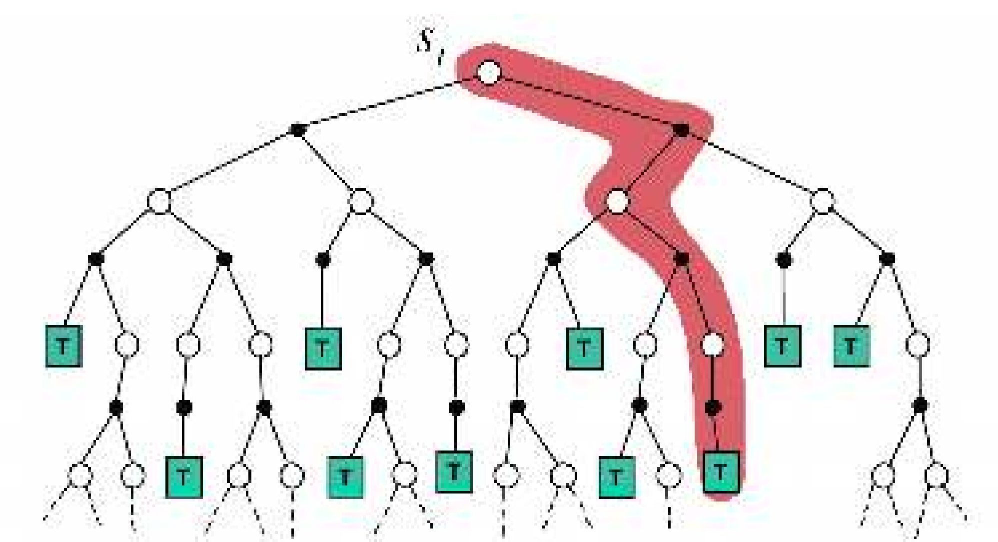
$$\mathrm{v}(\mathrm{s}_t) \longleftarrow v(s_t) + \alpha(G_t - v(s_t))$$

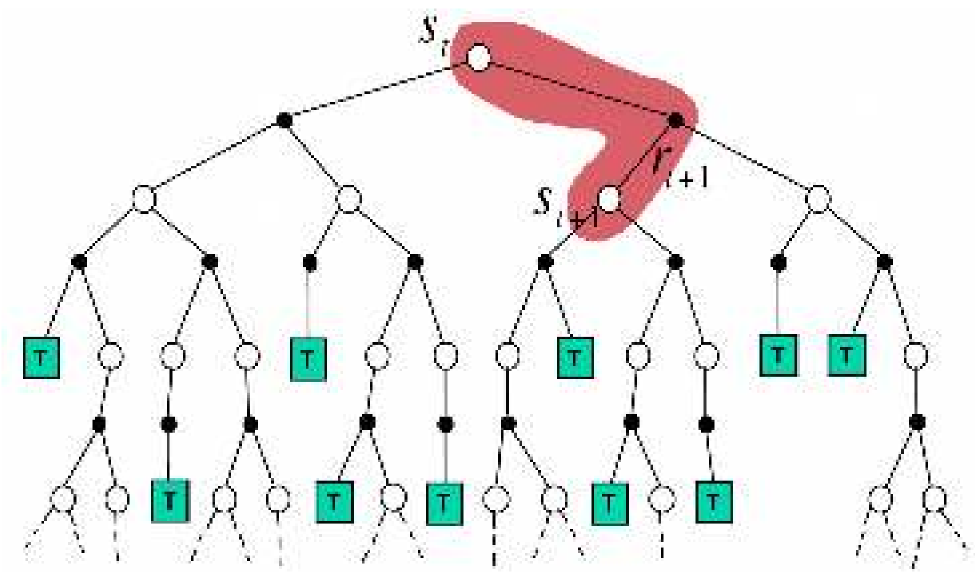


Updates MC, TD(0)

$$\mathrm{v}(\mathrm{s}_t) \longleftarrow v(s_t) + \alpha(G_t - v(s_t))$$

$$v(s_t) \longleftarrow v(s_t) + \alpha(R_t + \gamma v(s_{t+1}) - v(s_t))$$





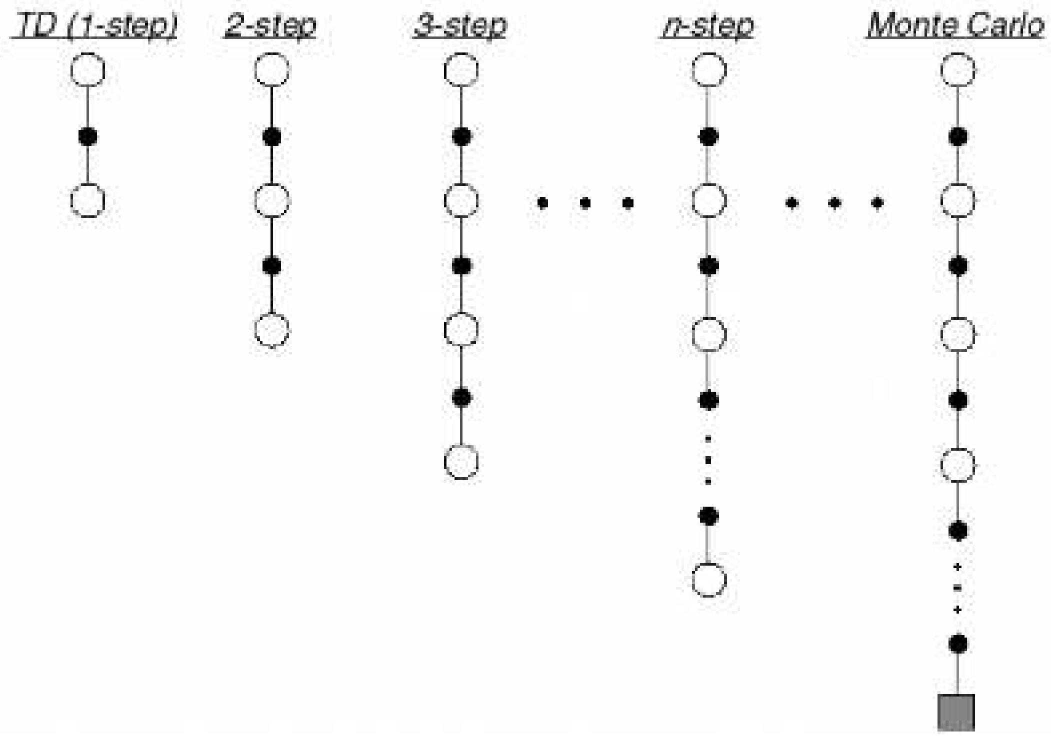
n-Step Temporal Difference

- No restriction to one-step look ahead

n Step Temporal Difference

- No restriction to one-step look ahead

$$G_t^{(n)} = \left(\sum_{i=0}^{n-1} \gamma^i R_{t+i} \right) + \gamma^n R_n$$

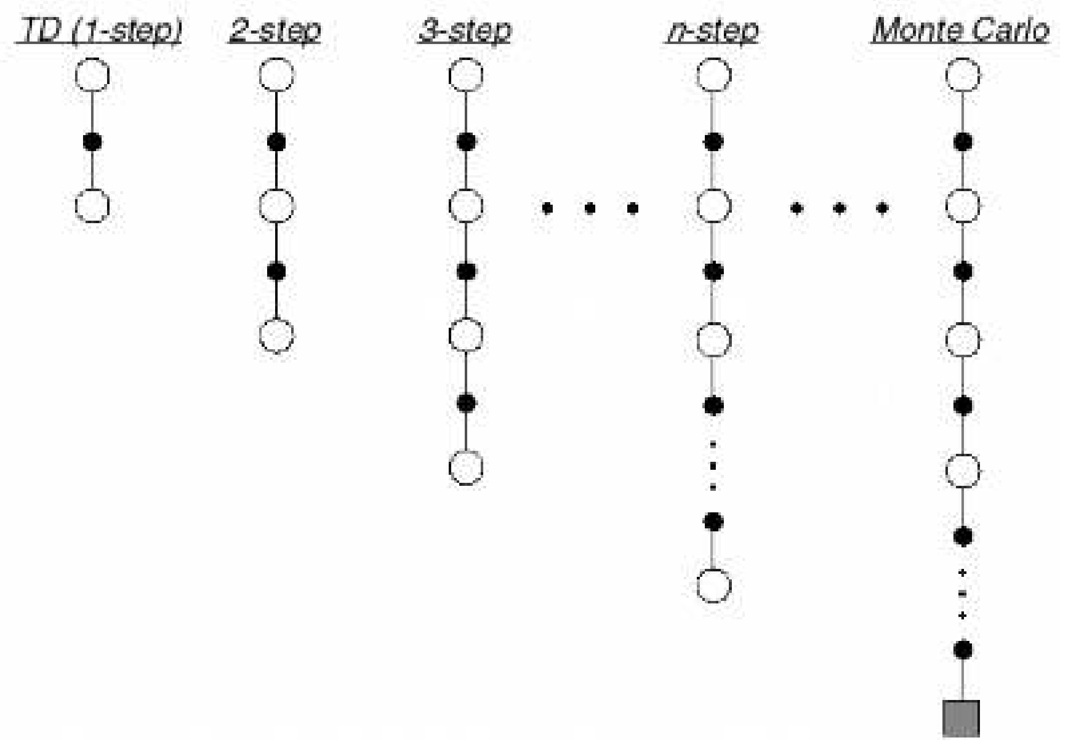


n Step Temporal Difference

- No restriction to one-step look ahead

$$G_t^{(n)} = \left(\sum_{i=0}^{n-1} \gamma^i R_{t+i} \right) + \gamma^n R_n$$

- Arbitrary averaging possible



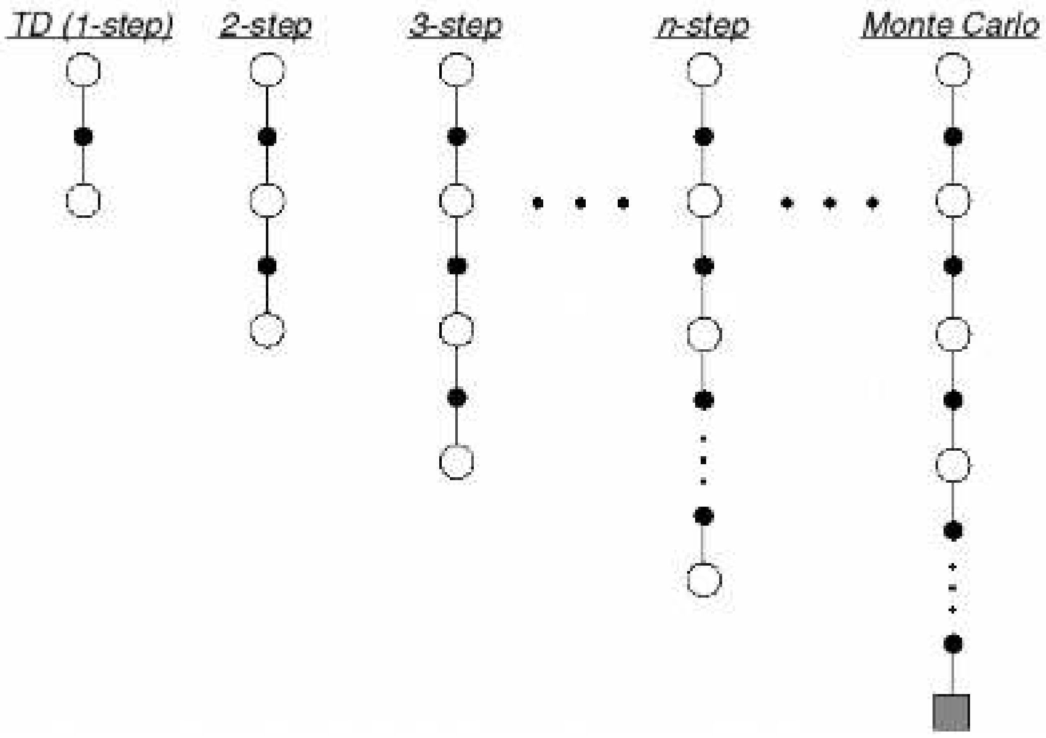
n Step Temporal Difference

- No restriction to one-step look ahead

$$G_t^{(n)} = \left(\sum_{i=0}^{n-1} \gamma^i R_{t+i}\right) + \gamma^n R_n$$

- Arbitrary averaging possible
- TD(λ) takes geometric average over all steps

$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$$



Model-free Policy Iteration

- Problem:

  policy iteration over *v(s)* requires a model

$$\pi^{'}(s) = \arg\max_a \mathcal{R}_s^a + \gamma \mathcal{P}_{ss'}^a v(s')$$

Model-free Policy Iteration

- Problem:

  policy iteration over *v(s)* requires a model

$$\pi^{'}(s) = \arg\max_a \mathcal{R}_s^a + \gamma \mathcal{P}_{ss'}^a v(s')$$

- Solution:

  use the action-value function *q(s, a)* instead

$$\pi^{'}(s) = \arg\max_a q(s, a)$$

Exploitation vs Exploration

Exploitation

exploits what is already known to maximize reward

Exploration

finds more information about the environment

Exploration vs Exploitation

Exploitation

exploits what is already known to maximize reward

Exploration

finds more information about the environment

- agent needs to find the optimal policy
- agent has to maximize his return

Exploration vs Exploitation

Exploitation

exploits what is already known to maximize reward

Exploration

finds more information about the environment

- agent needs to find the optimal policy

→ has to explore his possibilities

- agent has to maximize his return

→ should not give up on to much known rewards

ε-Greedy Exploration

- Chose a random action with probability *ε*
- with *m* actions available

$$\pi(a|s) = \begin{cases} \epsilon/m + 1 - \epsilon & \text{if } a^* = \arg\max_a q(s,a) \\ \epsilon/m & \text{otherwise} \end{cases}$$

MC and TD control

- same as before, but with action-value function q

MC and TD control

- same as before, but with action-value function *q*
- Monte Carlo: $\mathrm{v}(\mathrm{s}_t) \longleftarrow v(s_t) + \alpha(G_t - v(s_t))$

MC and TD control

- same as before, but with action-value function *q*
- Monte Carlo: $\mathrm{v}(\mathrm{s}_t) \longleftarrow v(s_t) + \alpha(G_t - v(s_t))$

$\rightarrow$ $\mathrm{q}(\mathrm{s}_t, a_t) \longleftarrow q(s_t, a_t) + \alpha(G_t - q(s_t, a_t))$

MC and TD control

- same as before, but with action-value function q
- Monte Carlo: $\mathrm{v}(\mathrm{s}_t) \longleftarrow v(s_t) + \alpha(G_t - v(s_t))$

$\rightarrow \qquad \mathrm{q}(\mathrm{s}_t, a_t) \longleftarrow q(s_t, a_t) + \alpha(G_t - q(s_t, a_t))$

$$\epsilon \longleftarrow 1/k \qquad \pi \longleftarrow \epsilon - greedy(q)$$

MC and TD control

- same as before, but with action-value function *q*
- Monte Carlo: $\mathrm{v}(\mathrm{s}_t) \longleftarrow v(s_t) + \alpha(G_t - v(s_t))$

  $\rightarrow \qquad \mathrm{q}(\mathrm{s}_t, a_t) \longleftarrow q(s_t, a_t) + \alpha(G_t - q(s_t, a_t))$

- TD(0) → SARSA(0): $v(s_t) \longleftarrow v(s_t) + \alpha(R_t + \gamma v(s_{t+1}) - v(s_t))$

$$\epsilon \longleftarrow 1/k \qquad \pi \longleftarrow \epsilon - greedy(q)$$

MC and TD control

- same as before, but with action-value function *q*
- Monte Carlo: $\mathrm{v}(\mathrm{s}_t) \longleftarrow v(s_t) + \alpha(G_t - v(s_t))$

$\rightarrow$ $\mathrm{q}(\mathrm{s}_t, a_t) \longleftarrow q(s_t, a_t) + \alpha(G_t - q(s_t, a_t))$

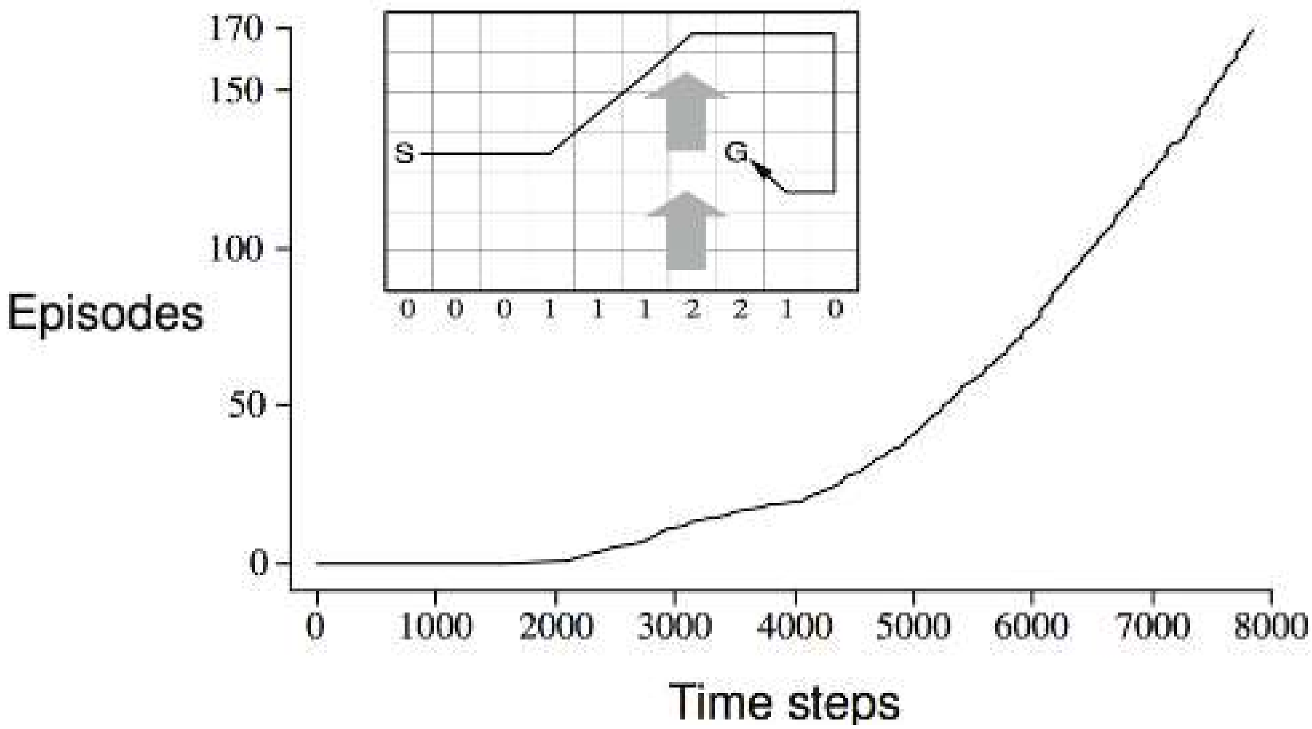
- TD(0) → SARSA(0): $v(s_t) \longleftarrow v(s_t) + \alpha(R_t + \gamma v(s_{t+1}) - v(s_t))$

$\rightarrow$ $q(s_t, a_t) \longleftarrow q(s_t, a_t) + \alpha(R_t + \gamma q(s_{t+1}, a_{t+1}) - q(s_t, a_t))$

$$\epsilon \longleftarrow 1/k \qquad \pi \longleftarrow \epsilon - greedy(q)$$

Typical Improvement (SARSA)

- seemingly no improvement at beginning



Problem with bigger Tasks

- Until now: small tasks with table lookup methods

Problem with bigger Tasks

- Until now: small tasks with table lookup methods
- What is about bigger tasks with more states?
  - Go: 10^{170} states

Problem with bigger Tasks

- Until now: small tasks with table lookup methods
- What is about bigger tasks with more states?
  - Go: 10^{170} states
  - Flying a helicopter: continuous state space

Problem with bigger Tasks

- Until now: small tasks with table lookup methods
- What is about bigger tasks with more states?
  - Go: 10^{170} states
  - Flying a helicopter: continuous state space

- **Solution:**

  → approximate the value functions instead of storing them

Value Function Approximation

- Represent state by feature vector

$$x(s) = (x_1(s), \ldots, x(_n(s))^T$$

Value Function Approximation

- Represent state by feature vector

$$x(s) = (x_1(s), \dots, x(_n(s))^T$$

- Represent value function by linear combination of features

$$\hat{v}(s, w) = x(s)^T w$$

Value Function Approximation

- Represent state by feature vector

$$x(s) = (x_1(s), \ldots, x(_n(s))^T$$

- Represent value function by linear combination of features

$$\hat{v}(s, w) = x(s)^T w$$

- Minimize mean squared error between true value and prediction by gradient descent

Value Function Approximation

- Represent state by feature vector

$$x(s) = (x_1(s), \ldots, x(_n(s))^T$$

- Represent value function by linear combination of features

$$\hat{v}(s, w) = x(s)^T w$$

- Minimize mean squared error between true value and prediction by gradient descent

- MSE

$$\mathcal{L}(w) = \mathbb{E}_\pi[(v_\pi(s) - \hat{v}_\pi(s, w))^2]$$

Value Function Approximation

- Represent state by feature vector

$$x(s) = (x_1(s), \ldots, x(_n(s))^T$$

- Represent value function by linear combination of features

$$\hat{v}(s, w) = x(s)^T w$$

- Minimize mean squared error between true value and prediction by gradient descent

- MSE

$$\mathcal{L}(w) = \mathbb{E}_\pi[(v_\pi(s) - \hat{v}_\pi(s, w))^2]$$

- Gradient of MSE w.r.t. *w*

$$\nabla_w \mathcal{L}(w) = -2\mathbb{E}_\pi[v_\pi(s) - \hat{v}_\pi(s, w)]\nabla_w \hat{v}(s, w)$$

Value Function Approximation

- Represent state by feature vector

$$x(s) = (x_1(s), \ldots, x(_n(s))^T$$

- Represent value function by linear combination of features

$$\hat{v}(s, w) = x(s)^T w$$

- Minimize mean squared error between true value and prediction by gradient descent

- MSE

$$\mathcal{L}(w) = \mathbb{E}_\pi[(v_\pi(s) - \hat{v}_\pi(s, w))^2]$$

- Gradient of MSE w.r.t. *w*

$$\nabla_w \mathcal{L}(w) = -2\mathbb{E}_\pi[v_\pi(s) - \hat{v}_\pi(s, w)]\nabla_w \hat{v}(s, w)$$

- Gradient update *Δw*

$$\Delta w = -\frac{1}{2}\alpha \nabla_w \mathcal{L}(w)$$

Value Function Approximation

- Represent state by feature vector

$$x(s) = (x_1(s), \ldots, x(_n(s))^T$$

- Represent value function by linear combination of features

$$\hat{v}(s, w) = x(s)^T w$$

- Minimize mean squared error between true value and prediction by gradient descent

$$\Delta w = \alpha \mathbb{E}_\pi [v_\pi(s) - \hat{v}_\pi(s, w)] x(s)$$

- MSE

$$\mathcal{L}(w) = \mathbb{E}_\pi [(v_\pi(s) - \hat{v}_\pi(s, w))^2]$$

- Gradient of MSE w.r.t. *w*

$$\nabla_w \mathcal{L}(w) = -2\mathbb{E}_\pi [v_\pi(s) - \hat{v}_\pi(s, w)] \nabla_w \hat{v}(s, w)$$

- Gradient update *Δw*

$$\Delta w = -\frac{1}{2} \alpha \nabla_w \mathcal{L}(w)$$

Value Function Approximation

- Represent state by feature vector

$$x(s) = (x_1(s), \ldots, x(_n(s))^T$$

- Represent value function by linear combination of features

$$\hat{v}(s, w) = x(s)^T w$$

- Minimize mean squared error between true value and prediction by gradient descent

$$\Delta w = \alpha \mathbb{E}_\pi [v_\pi(s) - \hat{v}_\pi(s, w)] x(s)$$

- Update weights with stochastic gradient descent

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

- MSE

$$\mathcal{L}(w) = \mathbb{E}_\pi [(v_\pi(s) - \hat{v}_\pi(s, w))^2]$$

- Gradient of MSE w.r.t. *w*

$$\nabla_w \mathcal{L}(w) = -2\mathbb{E}_\pi [v_\pi(s) - \hat{v}_\pi(s, w)] \nabla_w \hat{v}(s, w)$$

- Gradient update *Δw*

$$\Delta w = -\frac{1}{2} \alpha \nabla_w \mathcal{L}(w)$$

Updates for MC, TD(0), TD(λ)

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

Substitute the target with the corresponding returns

Updates for MC, TD(0), TD(λ)

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

Substitute the target with the corresponding returns

- MC $\quad v(s) \longrightarrow G_t$

$$\Delta w = \alpha(G_t - \hat{v}_\pi(s, w))x(s)$$

Updates for MC, TD(0), TD(λ)

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

Substitute the target with the corresponding returns

- MC $\quad v(s) \longrightarrow G_t$

$$\Delta w = \alpha(G_t - \hat{v}_\pi(s, w))x(s)$$

- TD(0) $\quad v(s_t) \longrightarrow R_t + \gamma \hat{v}(s_{t+1}, w)$

$$\Delta w = \alpha(R_t + \gamma \hat{v}(s_{t+1}, w) - \hat{v}_\pi(s_t, w))x(s_t)$$

Updates for MC, TD(0), TD(λ)

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

Substitute the target with the corresponding returns

- MC $\quad v(s) \longrightarrow G_t$

$$\Delta w = \alpha(G_t - \hat{v}_\pi(s, w))x(s)$$

- TD(0) $\quad v(s_t) \longrightarrow R_t + \gamma\hat{v}(s_{t+1}, w)$

$$\Delta w = \alpha(R_t + \gamma\hat{v}(s_{t+1}, w) - \hat{v}_\pi(s_t, w))x(s_t)$$

- TD(λ) $\quad v(s) \longrightarrow G_t^\lambda$

$$\Delta w = \alpha(G_t^\lambda - \hat{v}_\pi(s, w))x(s)$$

Action-Value Function Approx.

- Represent state by feature vector

$$x(s,a) = (x_1(s,a), \ldots, x(_n(s,a))^T$$

Action-Value Function Approx.

- Represent state by feature vector

$$x(s,a) = (x_1(s,a), \ldots, x(_n(s,a))^T$$

- Represent value function by linear combination of features

$$\hat{q}(s,a,w) = x(s,a)^T w$$

Action-Value Function Approx.

- Represent state by feature vector

$$x(s,a) = (x_1(s,a), \ldots, x(_n(s,a))^T$$

- Represent value function by linear combination of features

$$\hat{q}(s,a,w) = x(s,a)^T w$$

- Minimize mean squared error between true value and prediction by gradient descent

$$\Delta w = \alpha \mathbb{E}_\pi [q_\pi(s,a) - \hat{q}_\pi(s,a,w)] x(s,a)$$

Action-Value Function Approx.

- Represent state by feature vector

$$x(s,a) = (x_1(s,a), \ldots, x(_n(s,a))^T$$

- Represent value function by linear combination of features

$$\hat{q}(s,a,w) = x(s,a)^T w$$

- Minimize mean squared error between true value and prediction by gradient descent

$$\Delta w = \alpha \mathbb{E}_\pi[q_\pi(s,a) - \hat{q}_\pi(s,a,w)]x(s,a)$$

- Update weights with stochastic gradient descent

$$\Delta w = \alpha(q_\pi(s,a) - \hat{q}_\pi(s,a,w))x(s,a)$$

Updates for MC, TD(0), TD(λ)

$$\Delta w = \alpha(q_\pi(s,a) - \hat{q}_\pi(s,a,w))x(s,a)$$

Substitute the target with the corresponding returns

Updates for MC, TD(0), TD(λ)

$$\Delta w = \alpha(q_\pi(s, a) - \hat{q}_\pi(s, a, w))x(s, a)$$

Substitute the target with the corresponding returns

- MC $\qquad q(s, a) \longrightarrow G_t$

$$\Delta w = \alpha(G_t - \hat{q}_\pi(s, a, w))x(s, a)$$

Updates for MC, TD(0), TD(λ)

$$\Delta w = \alpha(q_\pi(s,a) - \hat{q}_\pi(s,a,w))x(s,a)$$

Substitute the target with the corresponding returns

- MC $\quad q(s,a) \longrightarrow G_t$

$$\Delta w = \alpha(G_t - \hat{q}_\pi(s,a,w))x(s,a)$$

- TD(0) $\quad q(s_t,a_t) \longrightarrow R_t + \gamma\hat{q}(s_{t+1},a_{t+1},w)$

$$\Delta w = \alpha(R_t + \gamma\hat{q}(s_{t+1},a_{t+1},w) - \hat{q}_\pi(s_t,a_t,w))x(s_t,a_t)$$

Updates for MC, TD(0), TD(λ)

$$\Delta w = \alpha(q_\pi(s,a) - \hat{q}_\pi(s,a,w))x(s,a)$$

Substitute the target with the corresponding returns

- MC $\quad q(s,a) \longrightarrow G_t$

$$\Delta w = \alpha(G_t - \hat{q}_\pi(s,a,w))x(s,a)$$

- TD(0) $\quad q(s_t,a_t) \longrightarrow R_t + \gamma \hat{q}(s_{t+1},a_{t+1},w)$

$$\Delta w = \alpha(R_t + \gamma \hat{q}(s_{t+1},a_{t+1},w) - \hat{q}_\pi(s_t,a_t,w))x(s_t,a_t)$$

- TD(λ) $\quad q(s,a) \longrightarrow G_t^\lambda$

$$\Delta w = \alpha(G_t^\lambda - \hat{q}_\pi(s,a,w))x(s,a)$$

Thank you for your attention

Sources

- David Silver, UCL Course on RL

  http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html

  Lectures 1 to 6