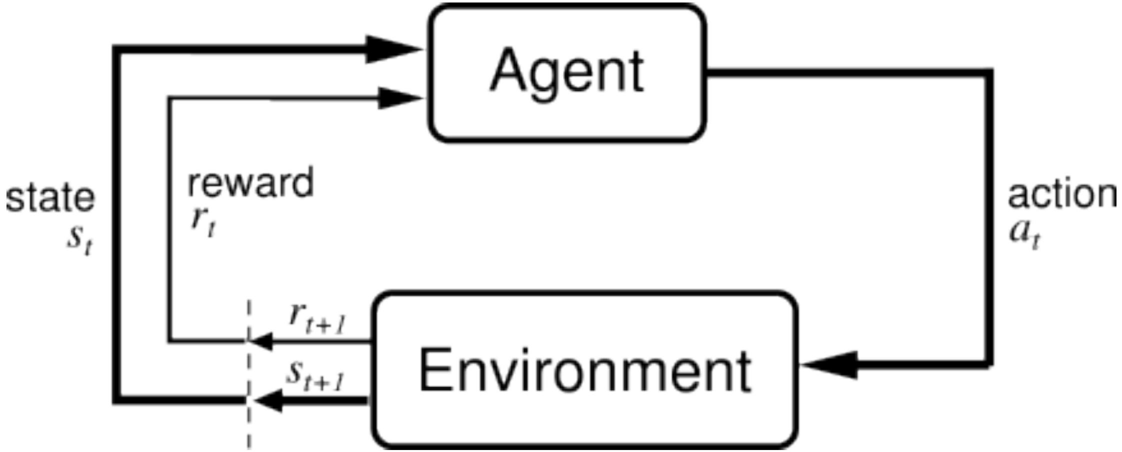
Solving Continuous MDPs with Discretization

Pieter Abbeel

UC Berkeley EECS

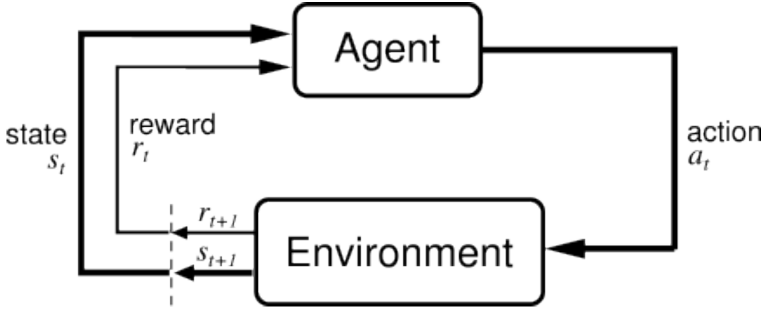
Markov Decision Process



Assumption: agent gets to observe the state

[Drawing from Sutton and Barto, Reinforcement Learning: An Introduction, 1998]

Markov Decision Process (S, A, T, R, γ, H)



Given

- S: set of states
- A: set of actions
- T: S x A x S x {0,1,…,H} → [0,1] $T_t(s,a,s') = P(s_{t+1} = s' \mid s_t = s, a_t = a)$
- R: S x A x S x {0, 1, …, H} → $\mathbb{R}$ $R_t(s,a,s')$ = reward for $(s_{t+1} = s', s_t = s, a_t = a)$
- γ in (0,1]: discount factor H: horizon over which the agent will act

Goal:

- Find π*: S x {0, 1, …, H} → A that maximizes expected sum of rewards, i.e.,

$$\pi^* = \arg\max_{\pi} \mathrm{E}[\sum_{t=0}^{H} \gamma^t R_t(S_t, A_t, S_{t+1}) | \pi]$$

Value Iteration

Algorithm:

Start with $V_0^*(s) = 0$ for all s.

For i = 1, … , H

For all states s in S:

$$V_{i+1}^*(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i^*(s')\right]$$

$$\pi_{i+1}^*(s) \leftarrow \arg\max_{a \in A} \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i^*(s')\right]$$

This is called a value update or Bellman update/back-up

$V_i^*(s)$ = expected sum of rewards accumulated starting from state s, acting optimally for i steps

$\pi_i^*(s)$ = optimal action when in state s and getting to act for i steps

Continuous State Spaces

- S = continuous set

- Value iteration becomes impractical as it requires to compute, for all states s in S:

$$V_{i+1}^*(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + V_i^*(s') \right]$$

Markov chain approximation to continuous state space dynamics model ("discretization")

- Original MDP

(S, A, T, R, γ, H)

- Discretized MDP

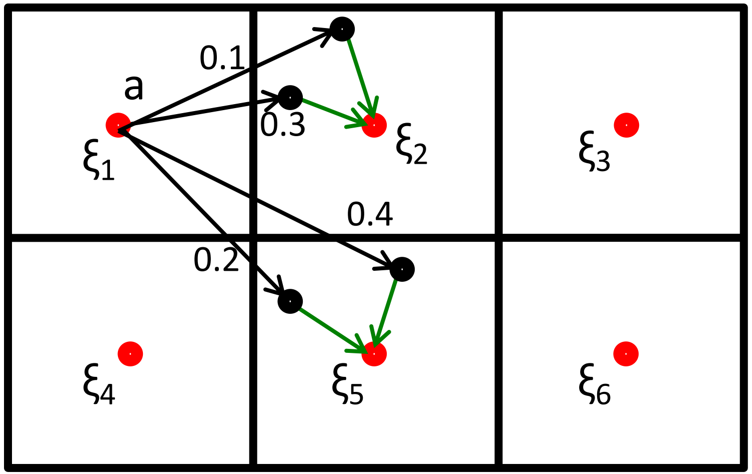
$(\bar{S}, \bar{A}, \bar{T}, \bar{R}, \gamma, H)$

- Grid the state-space: the vertices are the discrete states.
- Reduce the action space to a finite set.
  - Sometimes not needed:
    - When Bellman back-up can be computed exactly over the continuous action space
    - When we know only certain controls are part of the optimal policy (e.g., when we know the problem has a "bang-bang" optimal solution)
- Transition function: see next few slides.

Outline

- Discretization
- Lookahead policies
- Examples
- Guarantees
- Connection with function approximation

Discretization Approach 1: Snap onto nearest vertex



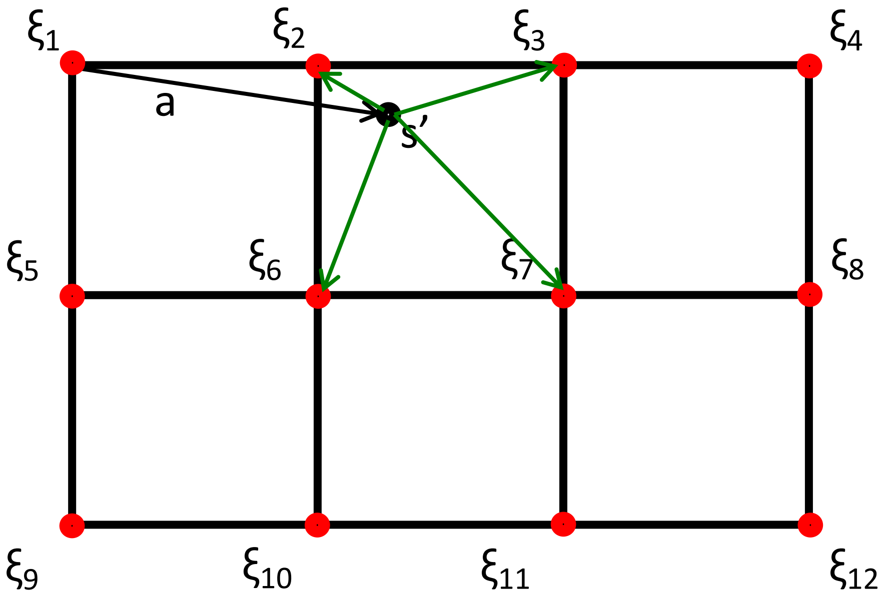
Discrete states: { ξ_1 , ..., ξ_6 }

$$P(\xi_2|\xi_1, a) = 0.1 + 0.3 = 0.4;$$
$$P(\xi_5|\xi_1, a) = 0.4 + 0.2 = 0.6$$

Similarly define transition probabilities for all ξ_i

- Discrete MDP just over the states {ξ_1, ...,ξ_6}, which we can solve with value iteration
- If a (state, action) pair can results in infinitely many (or very many) different next states: sample the next states from the next-state distribution

Discretization Approach 2: Stochastic Transition onto Neighboring Vertices



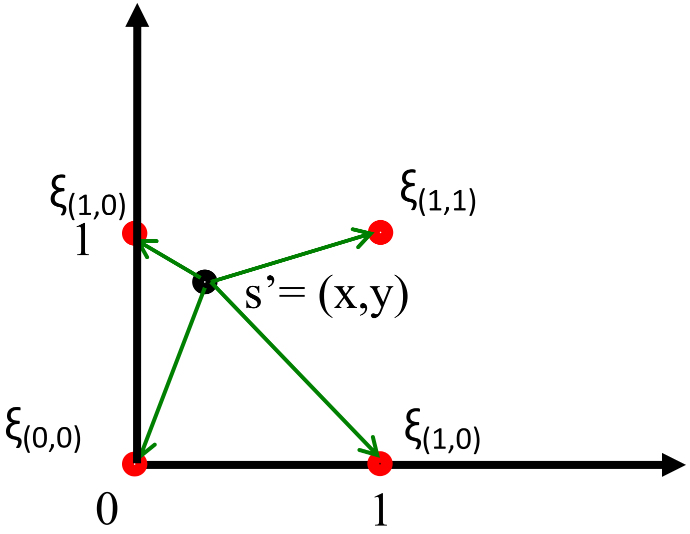
Discrete states: $\{\xi_1, \ldots, \xi_{12}\}$

$$P(\xi_2 \mid \xi_1, a) = p_A$$
$$P(\xi_3 \mid \xi_1, a) = p_B$$
$$P(\xi_6 \mid \xi_1, a) = p_C$$
$$P(\xi_7 \mid \xi_1, a) = p_D$$
$$\text{s.t. } s' = p_A \xi_2 + p_B \xi_3 + p_C \xi_6 + p_D \xi_7$$

- If stochastic dynamics: Repeat procedure to account for all possible transitions and weight accordingly
- Many choices for p_A, p_B, p_C, p_D

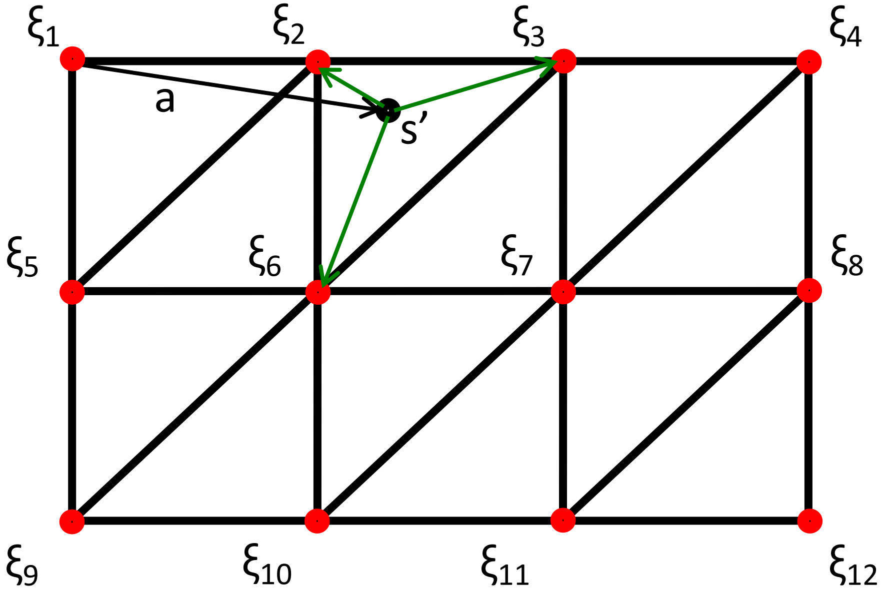
Discretization Approach 2: Stochastic Transition onto Neighboring Vertices

- One scheme to compute the weights: put in normalized coordinate system [0,1]x[0,1].



$$
\begin{aligned}
s' = (1-x)(1-y) &\quad \xi_{(0,0)} \\
+x(1-y) &\quad \xi_{(0,1)} \\
+(1-x)y &\quad \xi_{(1,0)} \\
+xy &\quad \xi_{(1,1)}
\end{aligned}
$$

Kuhn Triangulation**



Discrete states: $\{\xi_1, \ldots, \xi_{12}\}$

$$P(\xi_2|\xi_1, a) = p_A;$$
$$P(\xi_3|\xi_1, a) = p_B;$$
$$P(\xi_6|\xi_1, a) = p_C;$$
$$\text{s.t. } s' = p_A\xi_2 + p_B\xi_3 + p_C\xi_6$$

Kuhn Triangulation**

- Allows efficient computation of the vertices participating in a point's barycentric coordinate system and of the convex interpolation weights (aka its barycentric coordinates)

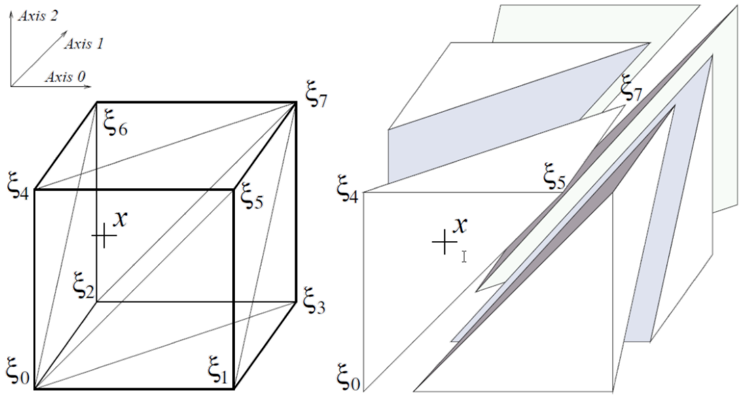


Figure 2. The Kuhn triangulation of a (3d) rectangle. The point x satisfying $1 \geq x_2 \geq x_0 \geq x_1 \geq 0$ is in the simplex $(\xi_0, \xi_4, \xi_5, \xi_7)$.

- See Munos and Moore, 2001 for further details.

Kuhn triangulation (from Munos and Moore)**

3.1. Computational issues

Although the number of simplexes inside a rectangle is factorial with the dimension d, the computation time for interpolating the value at any point inside a rectangle is only of order $(d \ln d)$, which corresponds to a sorting of the d relative coordinates $(x_0, ..., x_{d-1})$ of the point inside the rectangle.

Assume we want to compute the indexes $i_0, ..., i_d$ of the $(d+1)$ vertices of the simplex containing a point defined by its relative coordinates $(x_0, ..., x_{d-1})$ with respect to the rectangle in which it belongs to. Let $\{\xi_0, ..., \xi_{2^d}\}$ be the corners of this d-rectangle. The indexes of the corners use the binary decomposition in dimension d, as illustrated in Figure 2. Computing these indexes is achieved by sorting the coordinates from the highest to the smallest: there exist indices $j_0, ..., j_{d-1}$, permutation of $\{0, .., d-1\}$, such that $1 \geq x_{j_0} \geq x_{j_1} \geq ... \geq x_{j_{d-1}} \geq 0$. Then the indices $i_0, ..., i_d$ of the $(d+1)$ vertices of the simplex containing the point are: $i_0 = 0$, $i_1 = i_0 + 2^{j_0}$, ..., $i_k = i_{k-1} + 2^{j_{k-1}}$, ..., $i_d = i_{d-1} + 2^{j_{d-1}} = 2^d - 1$. For example, if the coordinates satisfy: $1 \geq x_2 \geq x_0 \geq x_1 \geq 0$ (illustrated by the point x in Figure 2) then the vertices are: ξ_0 (every simplex contains this vertex, as well as $\xi_{2^d-1} = \xi_7$), ξ_4 (we added 2^2), ξ_5 (we added 2^0) and ξ_7 (we added 2^1).

Let us define the *barycentric coordinates* $\lambda_0, ..., \lambda_d$ of the point x inside the simplex $\xi_{i_0}, ..., \xi_{i_d}$ as the positive coefficients (uniquely) defined by: $\sum_{k=0}^{d} \lambda_k = 1$ and $\sum_{k=0}^{d} \lambda_k \xi_{i_k} = x$. Usually, these barycentric coordinates are expensive to compute; however, in the case of Kuhn triangulation these coefficients are simply: $\lambda_0 = 1 - x_{j_0}$, $\lambda_1 = x_{j_0} - x_{j_1}$, ..., $\lambda_k = x_{j_{k-1}} - x_{j_k}$, ..., $\lambda_d = x_{j_{d-1}} - 0 = x_{j_{d-1}}$. In the previous example, the barycentric coordinates are: $\lambda_0 = 1 - x_2$, $\lambda_1 = x_2 - x_0$, $\lambda_2 = x_0 - x_1$, $\lambda_3 = x_1$.

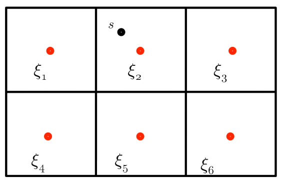
Discretization: Our Status

- Have seen two ways to turn a continuous state-space MDP into a discrete state-space MDP
- When we solve the discrete state-space MDP, we find:
  - Policy and value function for the discrete states
  - They are optimal for the discrete MDP, but typically not for the original MDP
- Remaining questions:
  - How to act when in a state that is not in the discrete states set?
  - How close to optimal are the obtained policy and value function?

How to Act (i): No Lookahead

- **For state s not in discretization set choose action based on policy in nearby states**
  - **Nearest Neighbor**

$$\pi(s) = \pi(\xi_i) \quad \text{for} \quad \xi_i = \arg \min_{\xi \in \{\xi_1, \ldots, \xi_N\}} \|s - \xi\|$$



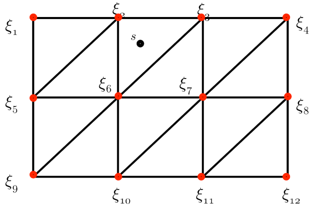
E.g., $\pi(s) = \pi(\xi_2)$

  - **Stochastic Interpolation:**

Find $p_1, \ldots, p_N$ s.t. $s = \sum_{i=1}^{N} p_i \xi_i$

Choose $\pi(\xi_i)$ with probability p_i

For continuous actions, can also interpolate: $\sum_{i=1}^{N} p_i \pi(\xi_i)$



E.g., for $s = p_2\xi_2 + p_3\xi_3 + p_6\xi_6$, choose $\pi(\xi_2), \pi(\xi_3), \pi(\xi_6)$ with respective probabilities p_2, p_3, p_6

How to Act (ii): 1-step Lookahead

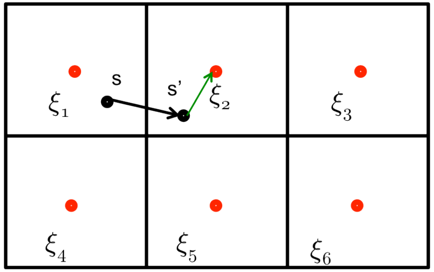
- Forward simulate for 1 step, calculate reward + value function at next state from discrete MDP

$$\max_{a_t} E\left[R(s_t, a_t) + \sum_i P(\xi_i; s_{t+1}) V(\xi_i)\right]$$

- if dynamics deterministic no expectation needed
- If dynamics stochastic, can approximate with samples

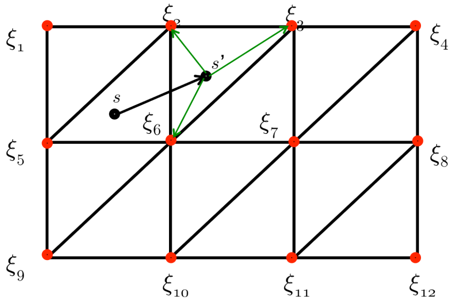
- **Nearest Neighbor**

$$P(\xi_i; s') = \begin{cases} 1 & \text{if } \xi_i = \arg\min_{\xi \in \{\xi_1, \ldots, \xi_N\}} \|s - \xi\| \\ 0 & \text{otherwise} \end{cases}$$



- **Stochastic Interpolation**

$$P(\xi_i; s') \text{ such that } \quad s' = \sum_{i=1}^{N} P(\xi_i; s')\xi_i$$



How to Act (iii): n-step Lookahead

$$\max_{a_t, a_{t+1}, \ldots, a_{t+k-1}} E\left[R(s_t, a_t) + R(s_{t+1}, a_{t+1}) + \ldots + R(s_{t+k-1}, a_{t+k-1}) + \sum_i P(\xi_i; s_{t+k}) V(\xi_i)\right]$$

- **What action space to maximize over, and how?**
  - Option 1: Enumerate sequences of discrete actions we ran value iteration with
  - Option 2: Randomly sampled action sequences ("random shooting")
  - Option 3: Run optimization over the actions
    - Local gradient descent [see later lectures]
    - Cross-entropy method

Intermezzo: Cross-Entropy Method (CEM)

- CEM = black-box method for (approximately) solving:

$$\max_x f(x)$$

with $x \in \mathbb{R}^n$ and $f : \mathbb{R}^n \rightarrow \mathbb{R}$

Note: f need not be differentiable

Intermezzo: Cross-Entropy Method (CEM)

$$\max_x f(x)$$

CEM:

sample $\mu^{(0)} \sim \mathcal{N}(0, \sigma^2)$

for iter i = 1, 2, …

for e = 1, 2. …

sample $x^{(e)} \sim \mathcal{N}(\mu^{(i)}, \sigma^2)$

compute $f(x^{(e)})$

endfor

$\mu^{(i+1)} = \text{mean}\{x^{(e)} : f(x^{(e)}) \text{ in top } 10\%\}$

Intermezzo: Cross-Entropy Method (CEM)

CEM:

sample $\mu^{(0)} \sim \mathcal{N}(0, \sigma^2)$

for iter i = 1, 2, …

for e = 1, 2, …

sample $x^{(e)} \sim \mathcal{N}(\mu^{(i)}, \sigma^2)$

compute $f(x^{(e)})$

endfor

$\mu^{(i+1)} = \text{mean}\{x^{(e)} : f(x^{(e)}) \text{ in top } 10\%\}$

- sigma and 10% are hyperparameters
- can in principle also fit sigma to top 10% (or full covariance matrix if low-D)
- How about discrete action spaces?
  - Within top 10%, look at frequency of each discrete action in each time step, and use that as probability
  - Then sample from this distribution

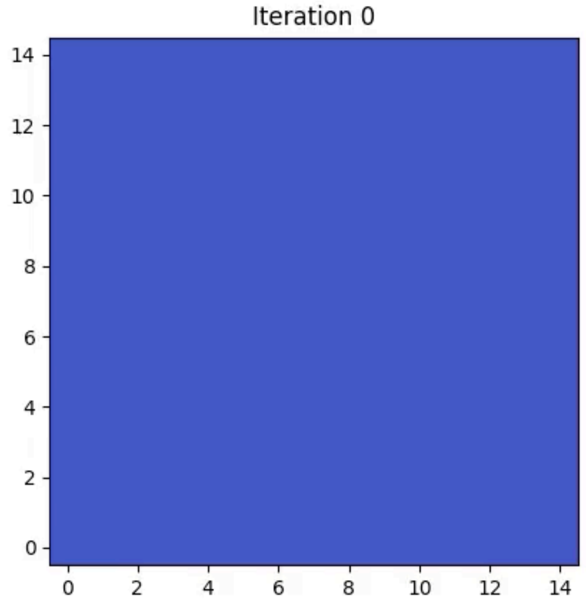
Note: there are many variations, including a max-ent variation, which does a weighted mean based on exp(f(x))

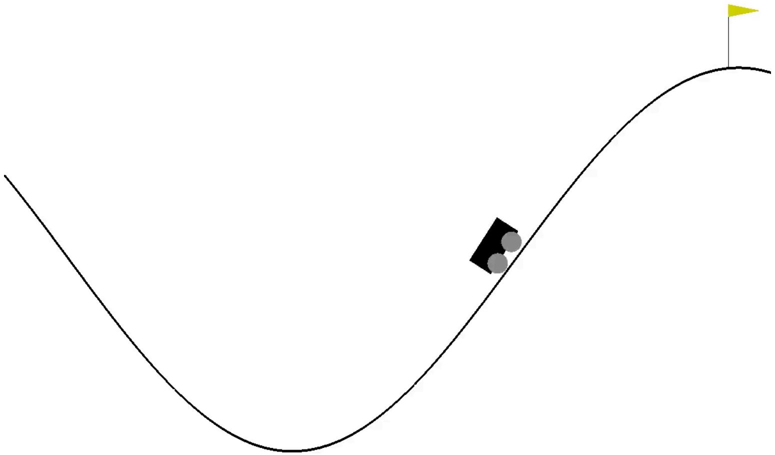
Outline

- Discretization
- Lookahead policies
- ***Examples***
- Guarantees
- Connection with function approximation

Mountain Car

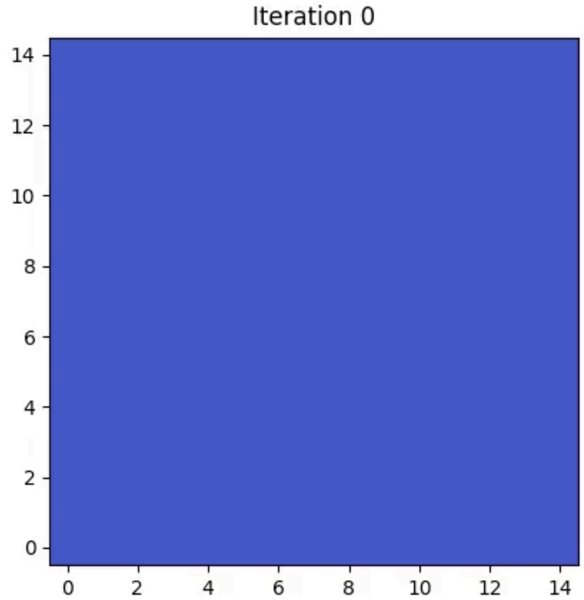
nearest neighbor
#discrete values per state dimension: 20
#discrete actions: 2 (as in original env)

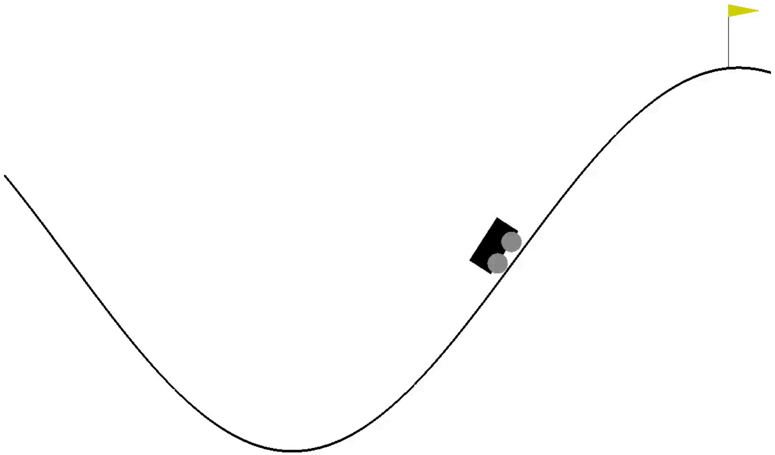




Mountain Car

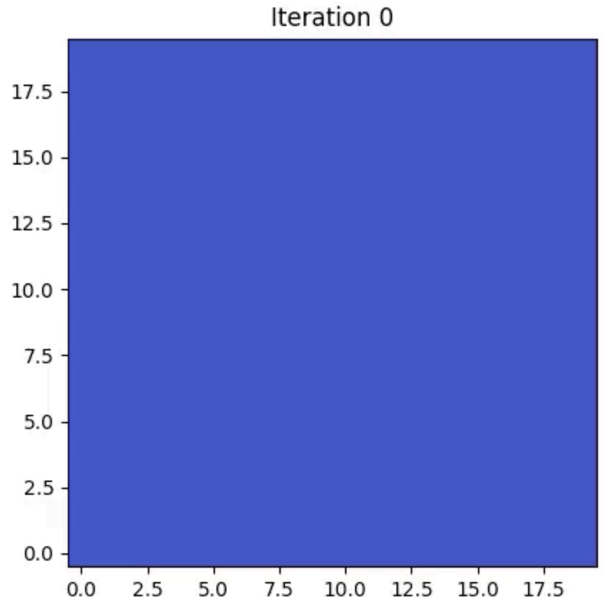
nearest neighbor
#discrete values per state dimension: 150
#discrete actions: 2 (as in original env)

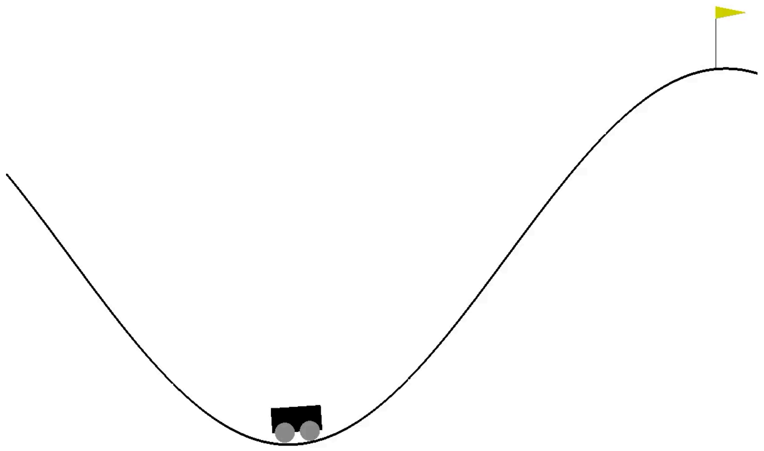




Mountain Car

linear
#discrete values per state dimension: 20
#discrete actions: 2 (as in original env)





Outline

- Discretization
- Lookahead policies
- Examples
- ***Guarantees***
- Connection with function approximation

Discretization Quality Guarantees

- Typical guarantees:
  - Assume: smoothness of cost function, transition model
  - For $h \rightarrow 0$, the discretized value function will approach the true value function

- To obtain guarantee about resulting policy, combine above with a general result about MDP's:
  - One-step lookahead policy based on value function V which is close to V* is a policy that attains value close to V*

Quality of Value Function Obtained from Discrete MDP: Proof Techniques

- Chow and Tsitsiklis, 1991:
  - Show that one discretized back-up is close to one "complete" back-up + then show sequence of back-ups is also close
- Kushner and Dupuis, 2001:
  - Show that sample paths in discrete stochastic MDP approach sample paths in continuous (deterministic) MDP [also proofs for stochastic continuous, bit more complex]
- Function approximation based proof (see later slides for what is meant with "function approximation")
  - Great descriptions: Gordon, 1995; Tsitsiklis and Van Roy, 1996

Example result (Chow and Tsitsiklis,1991)**

A.1: $|g(x, u) - g(x', u')| \le K\|(x, u) - (x', u')\|_\infty$, for all $x, x' \in S$ and $u, u' \in C$;

A.2: $|P(y \mid x, u) - P(y' \mid x', u')| \le K\|(y, x, u) - (y', x', u')\|_\infty$, for all $x, x', y, y' \in S$ and $u, u' \in C$;

A.3: for any $x, x' \in S$ and any $u' \in U(x')$, there exists some $u \in U(x)$ such that $\|u - u'\|_\infty \le K\|x - x'\|_\infty$;

A.4: $0 \le P(y \mid x, u) \le K$ and $\int_S P(y \mid x, u)\, dy = 1$, for all $x, y \in S$ and $u \in C$.

Theorem 3.1: There exist constants K_1 and K_2 (depending only on the constant K of assumptions A.1–A.4) such that for all $h \in (0, 1/2K]$ and all $J \in \mathscr{B}(S)$

$$\|TJ - \tilde{T}_h J\|_\infty \le (K_1 + \alpha K_2 \|J\|_S)h. \qquad (3.6)$$

Furthermore,

$$\|J^* - \tilde{J}_h^*\|_\infty \le \frac{1}{1-\alpha}(K_1 + \alpha K_2 \|J^*\|_S)h. \qquad (3.7)$$

Outline

- Discretization
- Lookahead policies
- Examples
- Guarantees
- ***Connection with function approximation***

Value Iteration with Function Approximation

Alternative interpretation of the discretization methods:

Start with $V_0^*(s) = 0$ for all s.

For i = 0, 1, … , H-1

for all states $s \in \bar{S}$,
($\bar{S}$ is the discrete state set)

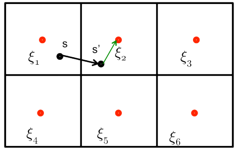
$$V_{i+1}^*(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \hat{V}_i^*(s')\right]$$

with:

$$\hat{V}_i^*(s') = \sum_j P(\xi_j; s') V_i^*(\xi_j)$$

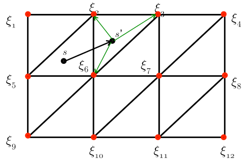
0'th Order Function Approximation

$$P(\xi_i; s') = \begin{cases} 1 & \text{if } \xi_i = \arg\min_{\xi \in \{\xi_1, \ldots, \xi_N\}} \|s - \xi\| \\ 0 & \text{otherwise} \end{cases}$$



1st Order Function Approximation

$$P(\xi_i; s') \text{ such that } \quad s' = \sum_{i=1}^N P(\xi_i; s') \xi_i$$



Discretization as Function Approximation

- Nearest neighbor discretization:
  - builds piecewise constant approximation of value function

- Stochastic transition onto nearest neighbors:
  - n-linear function approximation
  - Kuhn: piecewise (over "triangles") linear approximation of value function

Continuous time**

- One might want to discretize time in a variable way such that one discrete time transition roughly corresponds to a transition into neighboring grid points/regions

- Discounting: $\exp(-\beta \delta t)$

  δt depends on the state and action

See, e.g., Munos and Moore, 2001 for details.

Note: Numerical methods research refers to this connection between time and space as the CFL (Courant Friedrichs Levy) condition. Googling for this term will give you more background info.

!! 1 nearest neighbor tends to be especially sensitive to having the correct match [Indeed, with a mismatch between time and space 1 nearest neighbor might end up mapping many states to only transition to themselves no matter which action is taken.]