Continuous Space MDPs

Last Time

Last Time

- What are the differences between online and offline solutions?
- Are there solution techniques that are *independent* of the state space size?

Sparse Sampling

Guiding Questions

Guiding Questions

- What tools do we have to solve MDPs with continuous S and A?

Current Tool-Belt

offline $\begin{cases} VI \\ PI \end{cases}$

online MCTS

Won't work with continuous

S, A

Continuous S and A

Continuous S and A

e.g. $S \subseteq \mathbb{R}^n$, $A \subseteq \mathbb{R}^m$

$S \in [-1, 1]^4$

Continuous S and A

e.g. $S \subseteq \mathbb{R}^n$, $A \subseteq \mathbb{R}^m$

The old rules still work!

$$U^*(s) = \max_a \left(R(s,a) + \gamma \underset{s' \sim T(s,a)}{E}\left[U^*(s')\right]\right)$$

$$\cancel{\sum_{s'} T(s'|s,a)\, U^*(s')}$$

$$U^\pi(s) = \ldots$$

$$B[U](s) = \underbrace{\max_a}_{\text{hard!}} \left(R(s,a) + \gamma \underset{s' \sim T(s,a)}{E}\left[U(s')\right]\right)$$

$$\underbrace{\int_S T(s'|s,a)\, U(s')\, ds'}_{\text{hard (but much easier than max)}}$$

Nonlinear Optimization

MC

Today: Four Tools

1. LQR
2. Value Function Approx
3. Sparse Sampling / Prog. Widening
4. MPC

$S = \mathbb{R}^n \quad A = \mathbb{R}^m$

$x_{k+1} = A x_k + B u_k$

$\cancel{\dot{x} = A_c x + B_c u}$

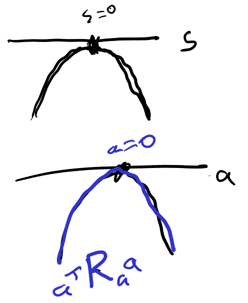
$A = e^{A_c t}$

1. Linear Dynamics, Quadratic Reward

$s' \sim \mathcal{N}(T_s s + T_a a, \Sigma)$

$s' = T_s s + T_a a + w \qquad w_t \sim \mathcal{N}(0, \Sigma)$

$R_s = \begin{bmatrix} -5 & \\ & -10 \end{bmatrix}$



$R(s,a) = s^T R_s s + a^T R_a a$

"optimal h-step policy" π_h^*

$U_h^*(s) = \max_\pi E\left[\sum_{t=0}^{h} R(s_t, a_t)\right]$

show that

$$\boxed{U_h^*(s) = s^T V_h s + q_h \qquad \pi_h^*(s) = -K_h s}$$

induction

base $U_1(s) = \max_a (s^T R_s s + a^T R_a a) = s^T R_s s \quad \therefore \quad V_1 = R_s, \; q_1 = 0$

inductive if $U_t(s)$ is quadratic, $U_{t+1}(s)$ is also quadratic

$$U_{t+1}(s) = \max_a \left(R(s,a) + E_{s' \sim T(s,a)}[U_t(s')]\right)$$

$$= \max_a \left(s^T R_s s + a^T R_a a + \int p(w)\, U_t(T_s s + T_a a + w)\, dw\right)$$

$$= s^T R_s s + \max\left(a^T R_a a + \int p(w) \left((T_s s + T_a a + w)^T V_t (T_s s + T_a a + w) + q_t\right) dw\right)$$

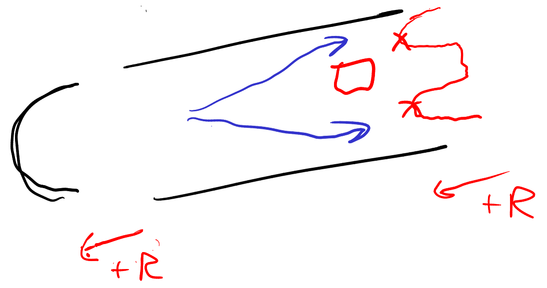
$$= s^T R_s s + s^T T_s^T V_t T_s s + \max_a \left(a^T R_a a + 2 s^T T_s^T V_t T_a a + a^T T_a^T V_t T_a a\right) + \int p(w)\, w^T V_t w\, dw$$

a^* is where $\nabla_a(\text{max term}) = 0$

$$0 = 2R_a a^* + 2T_a^T V_t T_s s + 2T_a^T V_t T_a a^*$$

$$-(2R_a + 2T_a^T V_t T_a) a^* = 2T_a^T V_t T_s s$$

$$a^* = \underbrace{-(R_a + T_a^T V_t T_a)^{-1} T_a^T V_t T_s}_{K_t} s$$



$$V_{t+1}(s) = s^T \underbrace{\left(R_s + T_s^T V_t T_s - (T_a V_t T_s)^T (R_a + T_a^T V_t T_a)^{-1} (T_a^T V_t T_s)\right)}_{V_{t+1}} s + \underbrace{\int p(w)\, w^T V_t w \, dw + q_t}_{q_{t+1}}$$

$$V_{t+1} = s^T V_{t+1} s + q_{t+1}$$

$$q_{t+1} = \sum_{i=1}^{t} \mathrm{Tr}[\Sigma V_i]$$

as $h \to \infty$

$$\to V_\infty = T_s^T \left(V_\infty - V_\infty T_a (T_a^T V_\infty T_a + R_a)^{-1} T_a^T V_\infty\right) T_s + R_s$$

$$K_\infty = (T_a^T V_\infty T_a + R_a)^{-1} T_a V_\infty T_s$$

does $V_\infty(s)$ depend on Σ

$V_\infty(s) = \infty$ through q_∞

$\pi^*_\infty(s) = -K_\infty s$

K_∞ has no dependence on Σ

Certainty-Equivalence Principle

V_∞ does not

Optimal Policy with noise = Optimal Policy w/o noise!!

In practice if dynamics close to linear + reward is convex LQR works

2. Value Function Approximation

2. Value Function Approximation

$V_\theta(s) = f_\theta(s)$ (e.g. neural network)

2. Value Function Approximation

$V_\theta(s) = f_\theta(s)$ (e.g. neural network)

$V_\theta(s) = \theta^\top \beta(s)$ (linear feature)

2. Value Function Approximation

$V_\theta(s) = f_\theta(s)$ (e.g. neural network)

$V_\theta(s) = \theta^\top \beta(s)$ (linear feature)

Fitted Value Iteration

while not converged

$\theta \leftarrow \theta'$

$\hat{V}' \leftarrow B_{\text{approx}}[V_\theta]$

$\theta' \leftarrow \text{fit}(\hat{V}')$

2. Value Function Approximation

$V_\theta(s) = f_\theta(s)$ (e.g. neural network)

$V_\theta(s) = \theta^\top \beta(s)$ (linear feature)

Fitted Value Iteration

while not converged

$\theta \leftarrow \theta'$

$\hat{V}' \leftarrow B_{\text{approx}}[V_\theta]$

$\theta' \leftarrow \text{fit}(\hat{V}')$

$$B_{\text{MC}(N)}[V_\theta](s) = \max_a \left(R(s,a) + \gamma \sum_{i=1}^{N} V_\theta(G(s,a,w_i)) \right)$$

2. Value Function Approximation

$V_\theta(s) = f_\theta(s)$ (e.g. neural network)

$V_\theta(s) = \theta^\top \beta(s)$ (linear feature)

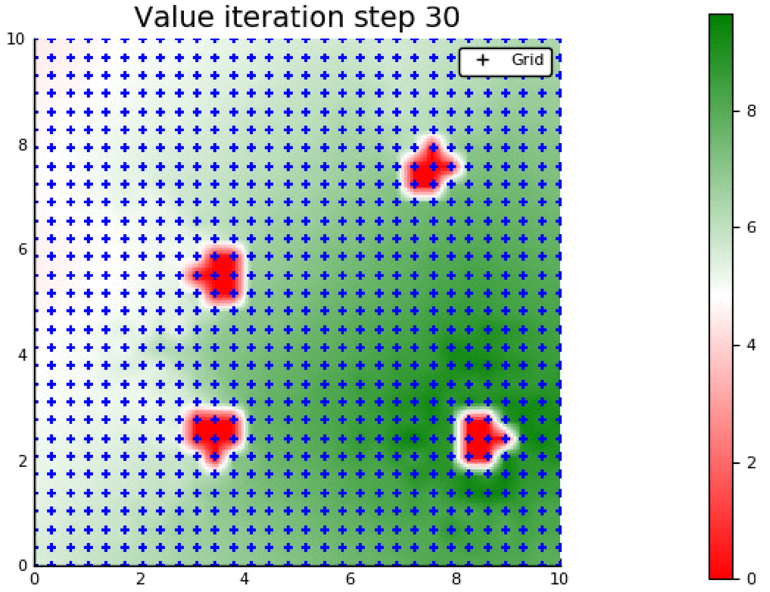
Fitted Value Iteration

while not converged

$\theta \leftarrow \theta'$

$\hat{V}' \leftarrow B_{\text{approx}}[V_\theta]$

$\theta' \leftarrow \text{fit}(\hat{V}')$



$$B_{\text{MC}(N)}[V_\theta](s) = \max_a \left(R(s,a) + \gamma \sum_{i=1}^{N} V_\theta(G(s,a,w_i)) \right)$$

2. Value Function Approximation

$V_\theta(s) = f_\theta(s)$ (e.g. neural network)

$V_\theta(s) = \theta^\top \beta(s)$ (linear feature)

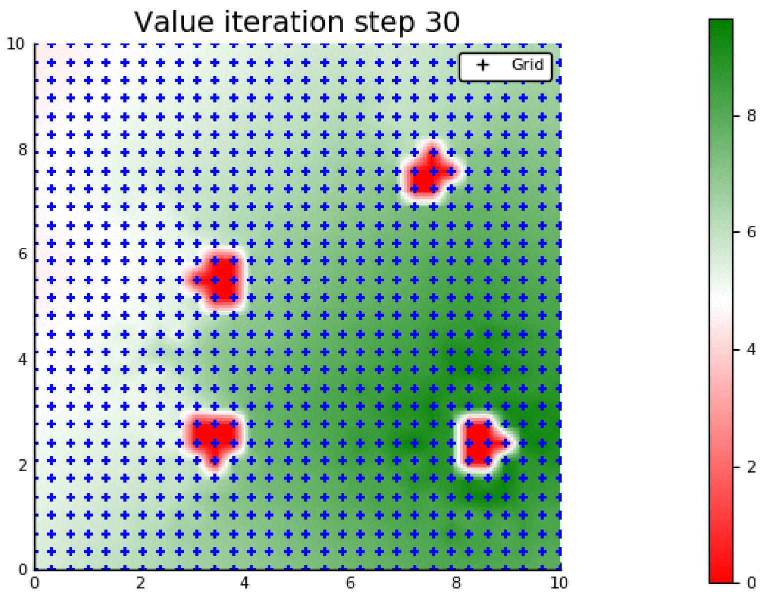
Fitted Value Iteration

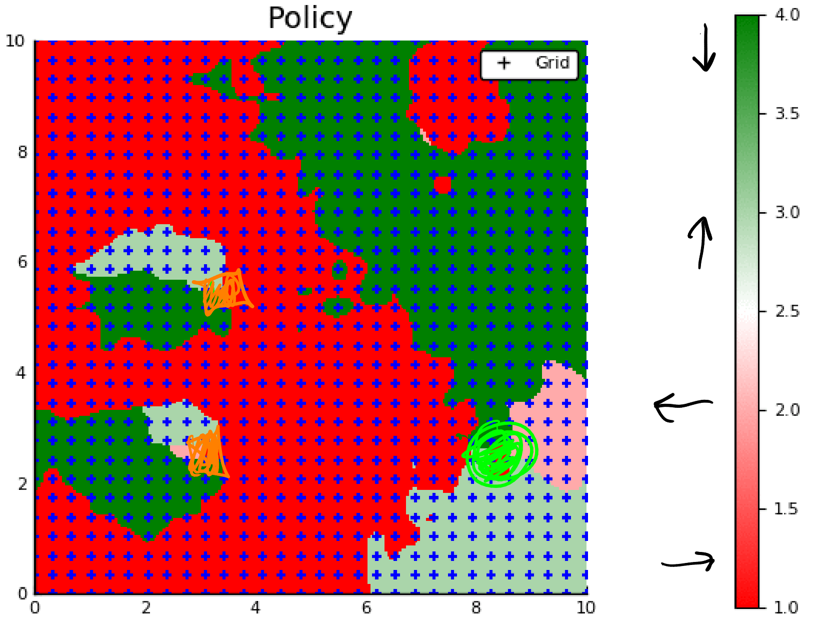
while not converged

$\theta \leftarrow \theta'$

$\hat{V}' \leftarrow B_{\text{approx}}[V_\theta]$

$\theta' \leftarrow \text{fit}(\hat{V}')$





$$B_{\text{MC}(N)}[V_\theta](s) = \max_a \left(R(s, a) + \gamma \sum_{i=1}^{N} V_\theta(G(s, a, w_i)) \right)$$

Function Approximation

Weighting of 2^d points

Weighting of only $d + 1$ points!

Function Approximation

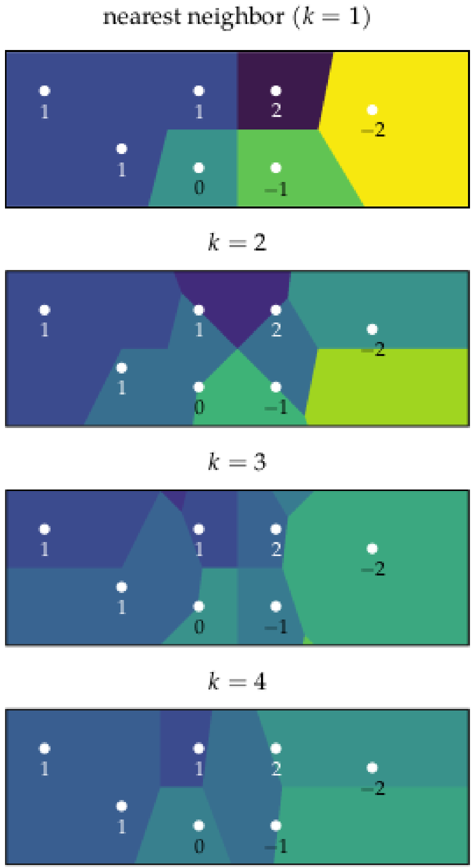
- Global: (e.g. Fourier, neural network)
- Local: (e.g. simplex interpolation)

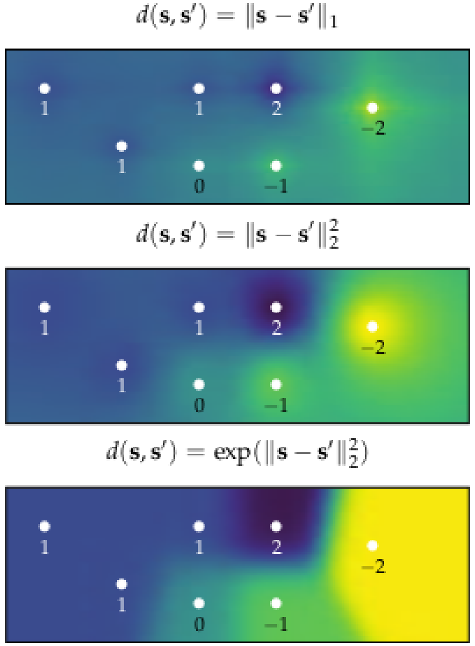
Weighting of 2^d points

Weighting of only $d+1$ points!

Function Approximation

- Global: (e.g. Fourier, neural network)
- Local: (e.g. simplex interpolation)





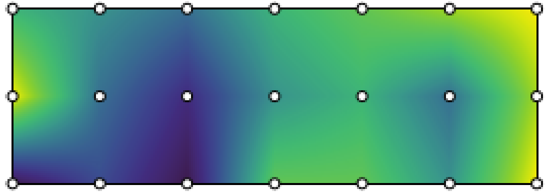


Figure 8.9. Two-dimensional linear interpolation over a 3×7 grid.

Weighting of 2^d points

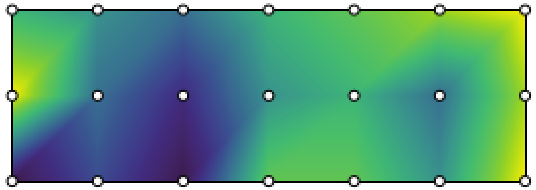
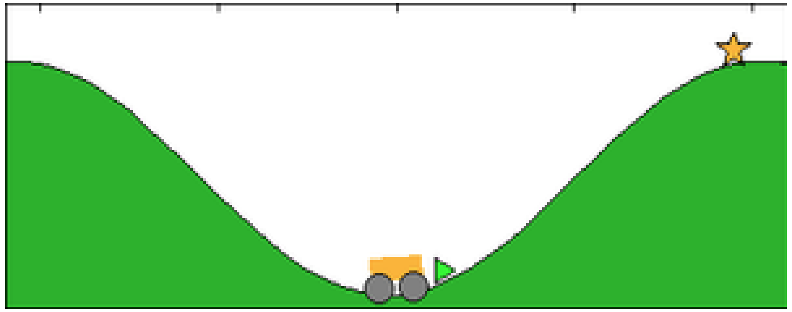


Figure 8.10. Two-dimensional simplex interpolation over a 3×7 grid.

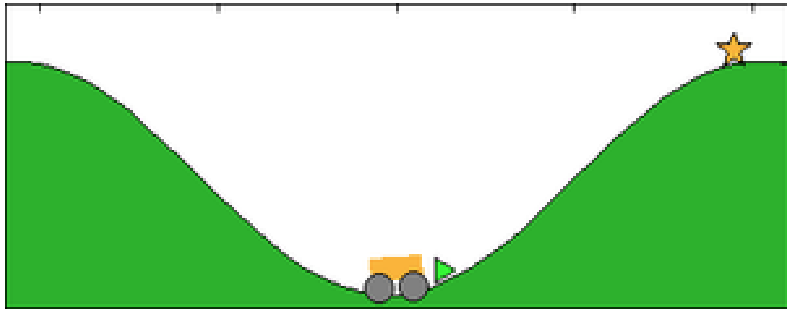
Weighting of only $d + 1$ points!

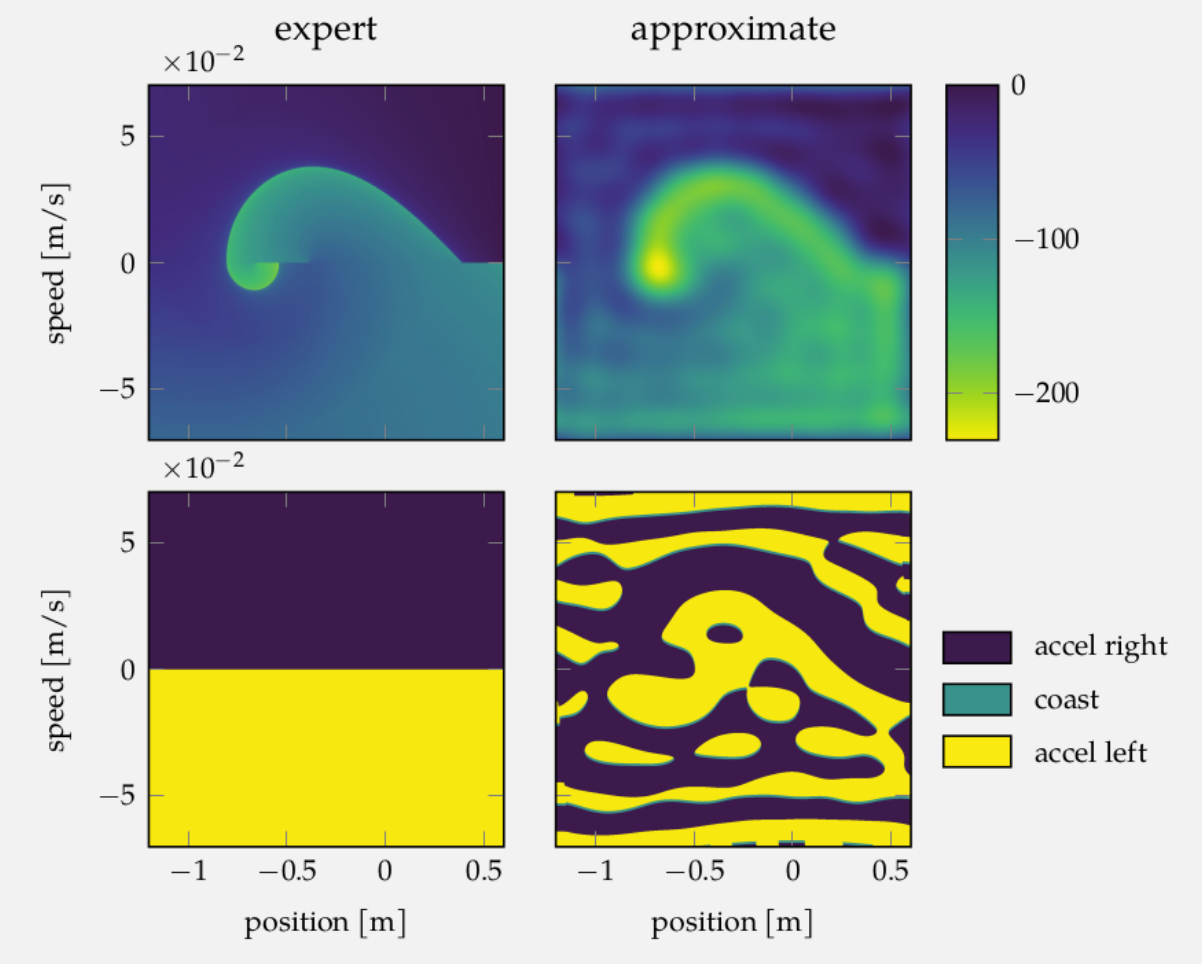
Function Approximation: Mountain Car

Function Approximation: Mountain Car



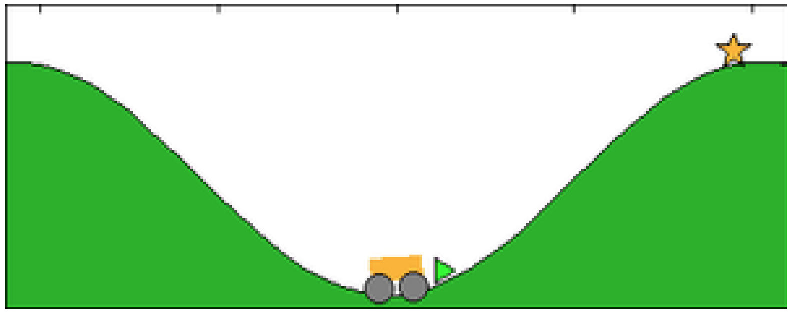
Function Approximation: Mountain Car





(Fourier, 17 params)

Function Approximation: Mountain Car



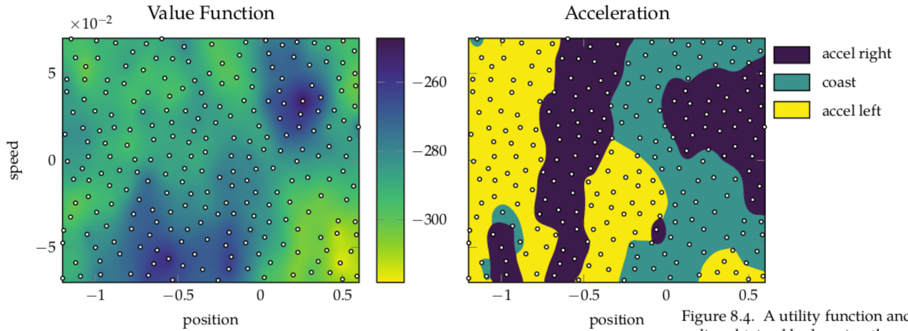
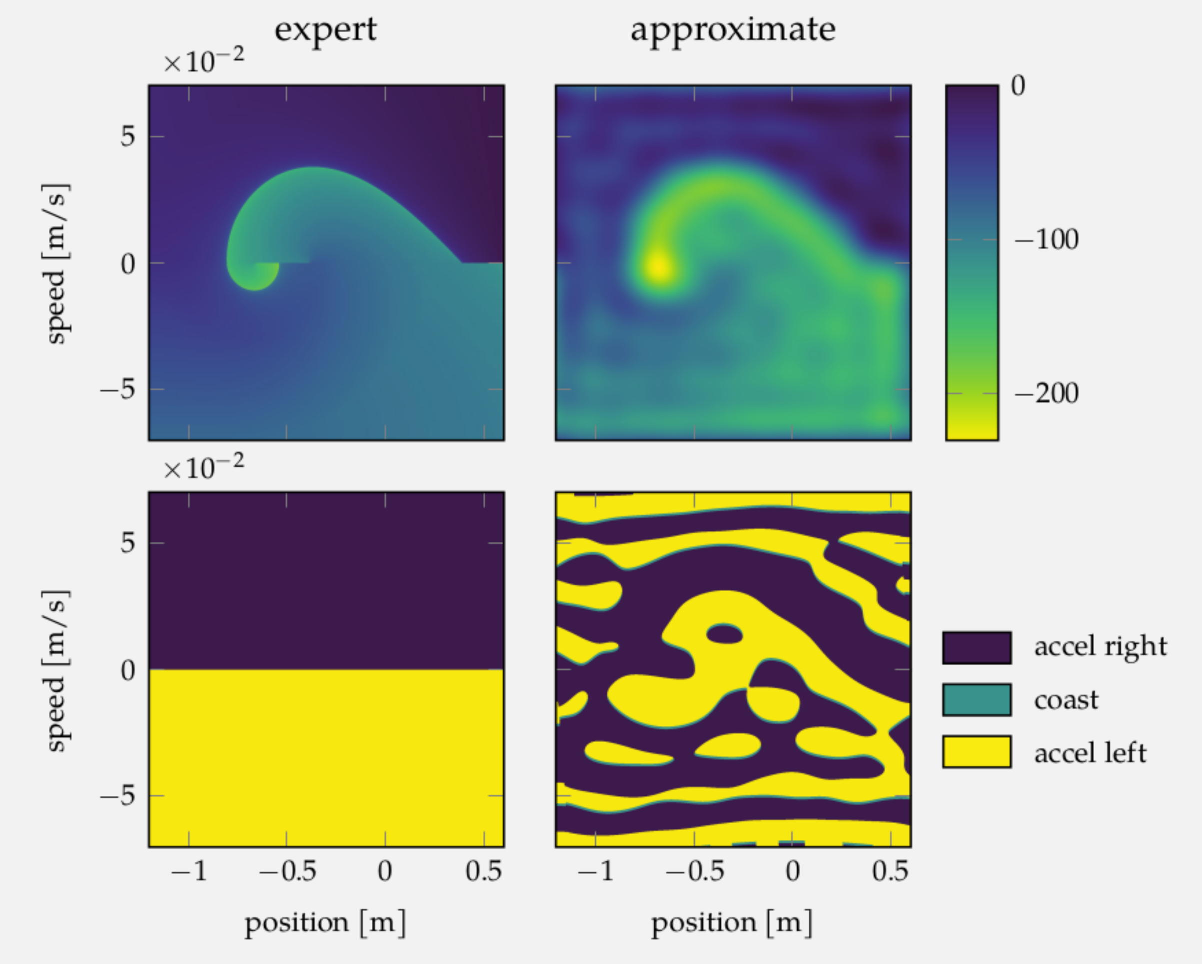


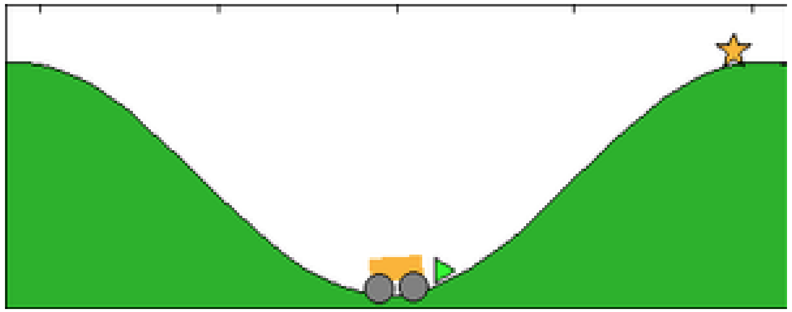
Figure 8.4. A utility function and policy obtained by learning the action values for a finite set of states (white) in the mountain car problem using the distance function $\|\mathbf{s}-\mathbf{s}'\|_2 + 0.1$.

(Kernel, > 100 params)



(Fourier, 17 params)

Function Approximation: Mountain Car



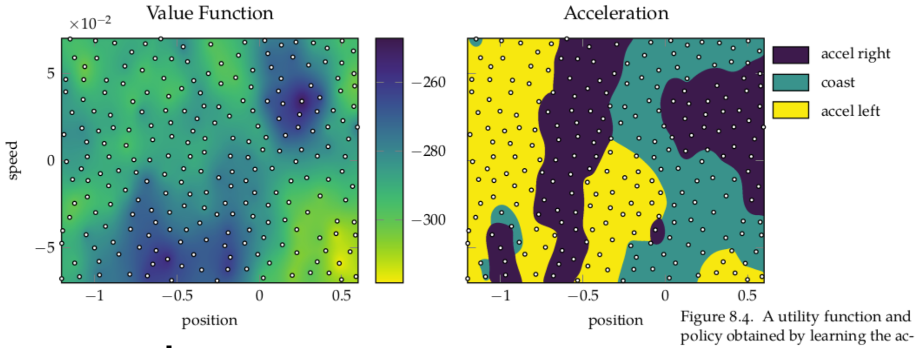
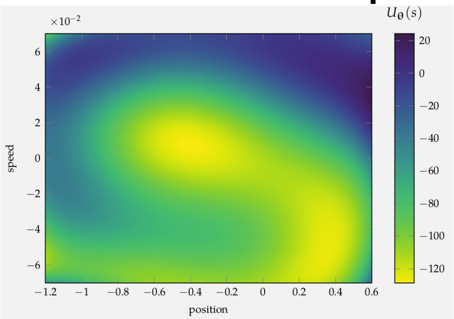
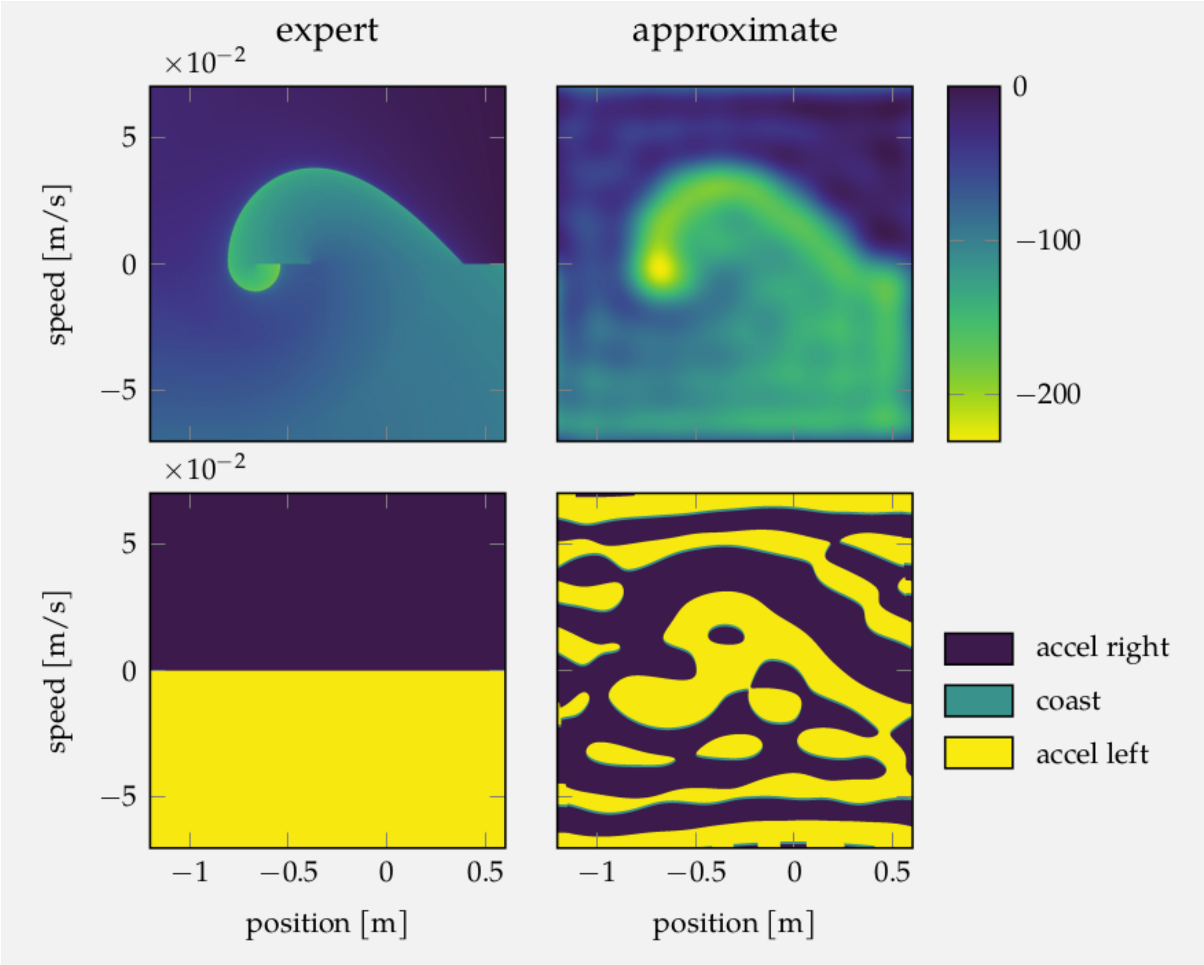


Figure 8.4. A utility function and policy obtained by learning the action values for a finite set of states (white) in the mountain car problem using the distance function $\|\mathbf{s} - \mathbf{s}'\|_2 + 0.1$.

(Kernel, > 100 params)

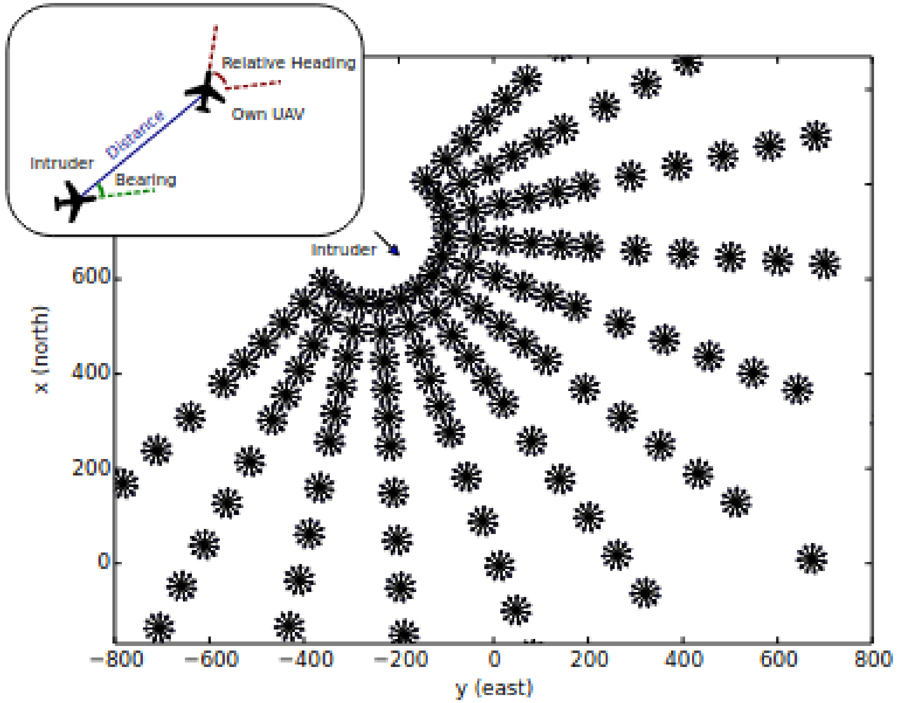


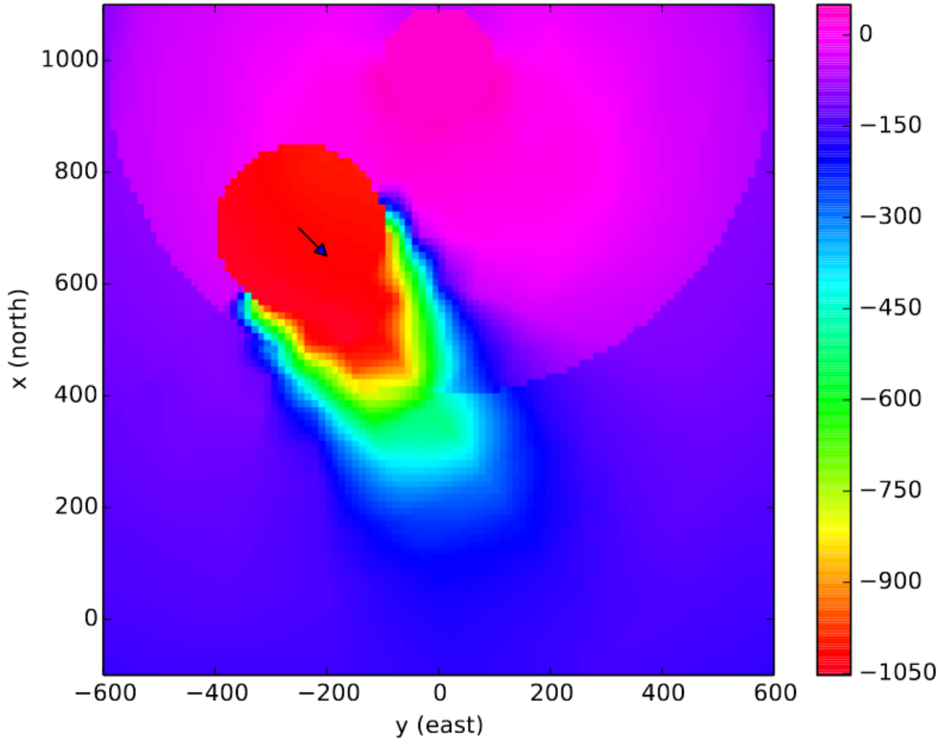
(Polynomial, 28 params)



(Fourier, 17 params)

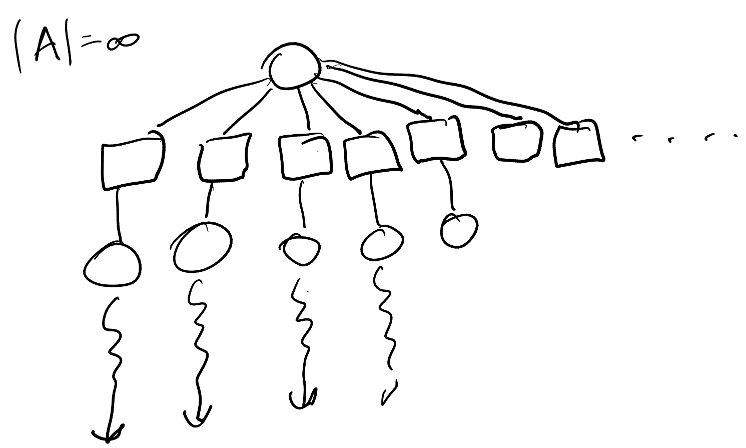
Function Approximation

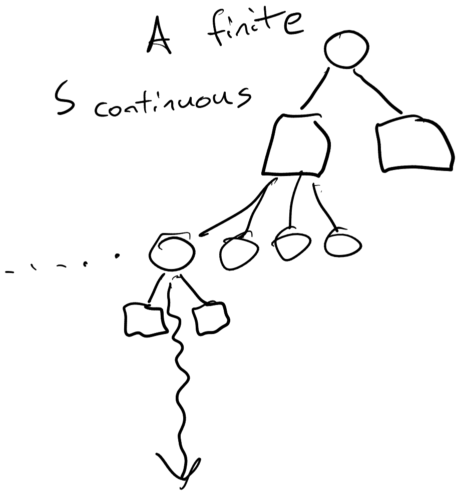




Break

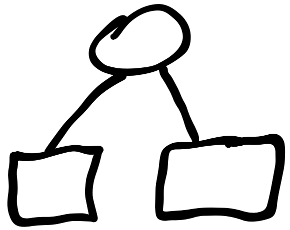
What will a Monte Carlo Tree Search tree look like if run on a problem with continuous spaces?



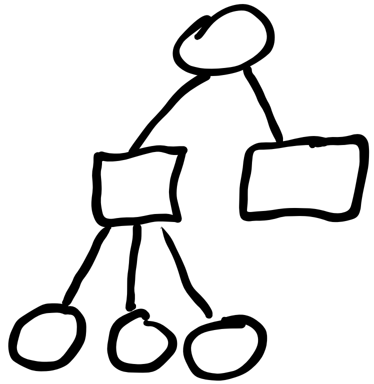


3. Sparse Tree Search/Progressive Widening

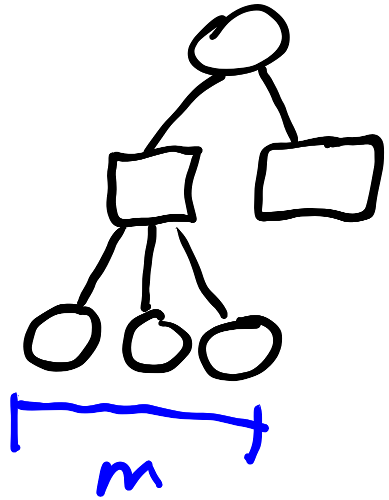
3. Sparse Tree Search/Progressive Widening



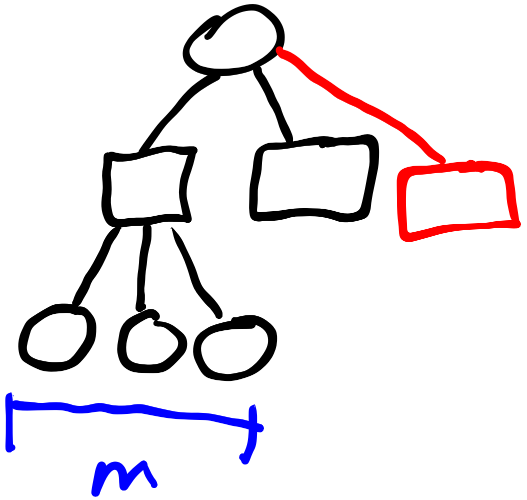
3. Sparse Tree Search/Progressive Widening



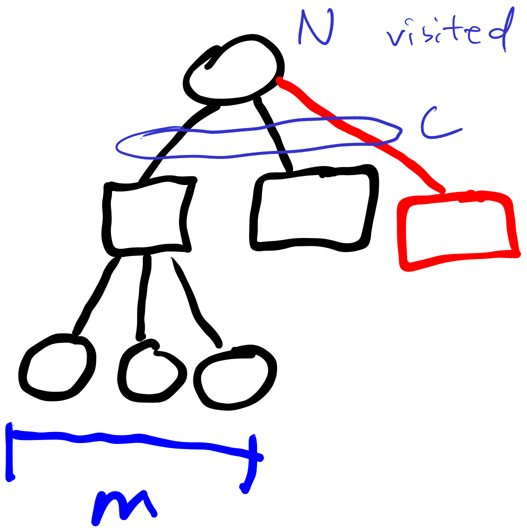
3. Sparse Tree Search/Progressive Widening



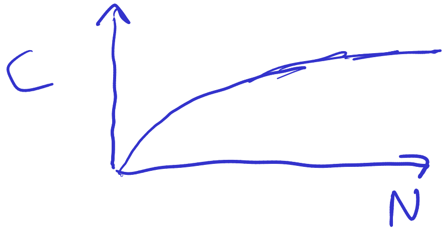
3. Sparse Tree Search/Progressive Widening



3. Sparse Tree Search/Progressive Widening

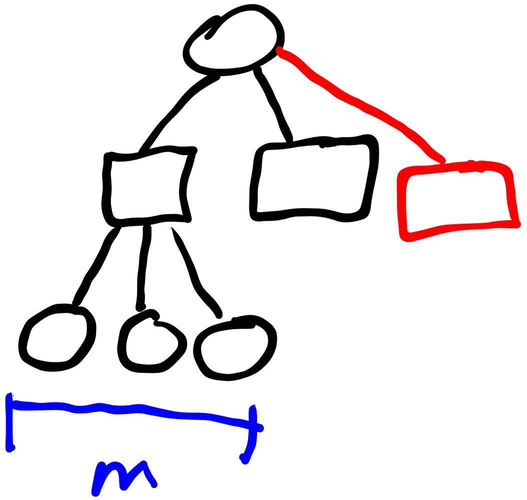


add new branch if $C < kN^{\alpha}$ $(\alpha < 1)$

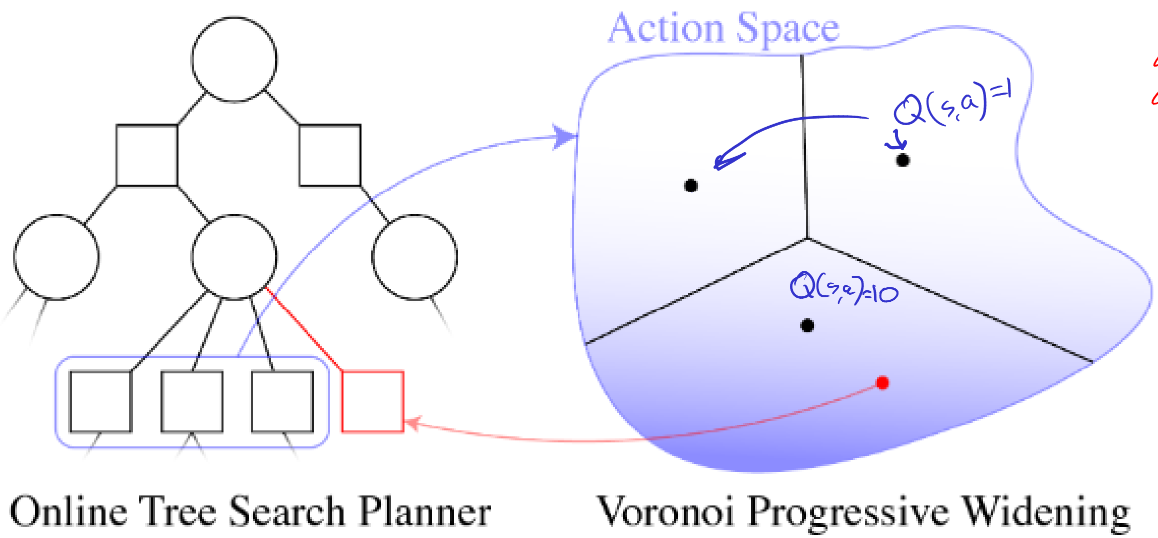


$\alpha < \frac{1}{2}$

3. Sparse Tree Search/Progressive Widening



add new branch if $C < kN^{\alpha} \quad (\alpha < 1)$



4. Model Predictive Control

(Use off-the-shelf optimization software, e.g. Ipopt)

4. Model Predictive Control

(Use off-the-shelf optimization software, e.g. Ipopt)

Certainty-Equivalent

$$\begin{aligned} \underset{a_{1:d}, s_{1:d}}{\text{maximize}} \quad & \sum_{t=1}^{d} \gamma^t R(s_t, a_t) \\ \text{subject to} \quad & s_{t+1} = \mathrm{E}[T(s_t, a_t)] \quad \forall t \end{aligned}$$

4. Model Predictive Control

(Use off-the-shelf optimization software, e.g. Ipopt)

Certainty-Equivalent

$$\begin{aligned} \underset{a_{1:d}, s_{1:d}}{\text{maximize}} \quad & \sum_{t=1}^{d} \gamma^t R(s_t, a_t) \\ \text{subject to} \quad & s_{t+1} = \mathrm{E}[T(s_t, a_t)] \quad \forall t \end{aligned}$$

Open-Loop

$$\begin{aligned} \underset{a_{1:d}, s_{1:d}^{(1:m)}}{\text{maximize}} \quad & \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{d} \gamma^t R(s_t^{(i)}, a_t) \\ \text{subject to} \quad & s_{t+1} = G(s_t^{(i)}, a_t, w_t^{(i)}) \quad \forall t, i \end{aligned}$$

4. Model Predictive Control

(Use off-the-shelf optimization software, e.g. Ipopt)

Certainty-Equivalent

$$\begin{aligned} \underset{a_{1:d}, s_{1:d}}{\text{maximize}} \quad & \sum_{t=1}^{d} \gamma^t R(s_t, a_t) \\ \text{subject to} \quad & s_{t+1} = \mathrm{E}[T(s_t, a_t)] \quad \forall t \end{aligned}$$

Open-Loop

$$\begin{aligned} \underset{a_{1:d}, s_{1:d}^{(1:m)}}{\text{maximize}} \quad & \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{d} \gamma^t R(s_t^{(i)}, a_t) \\ \text{subject to} \quad & s_{t+1} = G(s_t^{(i)}, a_t, w_t^{(i)}) \quad \forall t, i \end{aligned}$$

Hindsight Optimization

$$\begin{aligned} \underset{a_{1:d}^{(1:m)}, s_{1:d}^{(1:m)}}{\text{maximize}} \quad & \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{d} \gamma^t R(s_t^{(i)}, a_t^{(i)}) \\ \text{subject to} \quad & s_{t+1} = G(s_t^{(i)}, a_t^{(i)}, w_t^{(i)}) \quad \forall t, i \\ & a_1^{(i)} = a_1^{(j)} \quad \forall i, j \end{aligned}$$

Guiding Questions

- What tools do we have to solve MDPs with continuous S and A?