Last Time

Last Time

Tree Search

VI, PI

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

Guiding Questions

Guiding Questions

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

Current Tool-Belt

- VI
- PI
- MCTS

Why won't these work with continuous S, A?

Value Function is no longer a Vector

Continuous S and A

Continuous S and A

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

Continuous S and A

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

The old rules still work!

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

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

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

$$\underbrace{\int T(s' \mid s,a)\, U(s')\, ds'}_{\text{hard! (but less than maximization)}}$$

Nonlinear Optimization

Monte Carlo

Today: Four Tools

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

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

1. Linear Dynamics, Quadratic Reward

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

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

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

"optimal h-step utility"

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

$\pi_h^*(s)$

Show that

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

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

$\int p(w) w^T V_h T_s s \, dw = 0$

$U_{h+1}(s) = \max_a \left(R(s,a) + \underset{s' \sim T(s,a)}{E}[U(s')]\right)$

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

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

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

∇_a (max term) $= 0$

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

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

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

$$\pi_h(s) = -K_h s$$

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

$$U_{h+1} = \underline{s^T V_{h+1} s} + \underline{q_{h+1}}$$

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

as $h \to \infty$

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

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

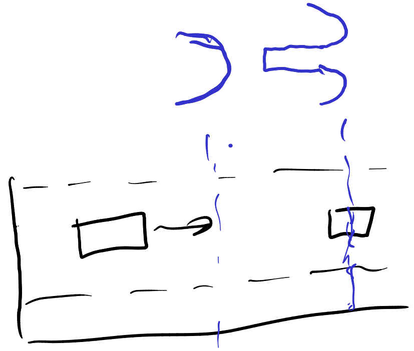
Does $U_\infty(s)$ depend on Σ

$U_\infty(s) = \infty$

U_h does depend on Σ through q

K_∞ has no dependence on Σ

Certainty-Equivalence Principle



Optimal Value is Quadratic

Optimal policy is linear and doesn't depend on noise

2. Value Function Approximation

2. Value Function Approximation

V_θ

$V_\theta(s) = f_\theta(s)$ (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

$\theta \leftarrow \theta'$

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

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

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

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

2. Value Function Approximation

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

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

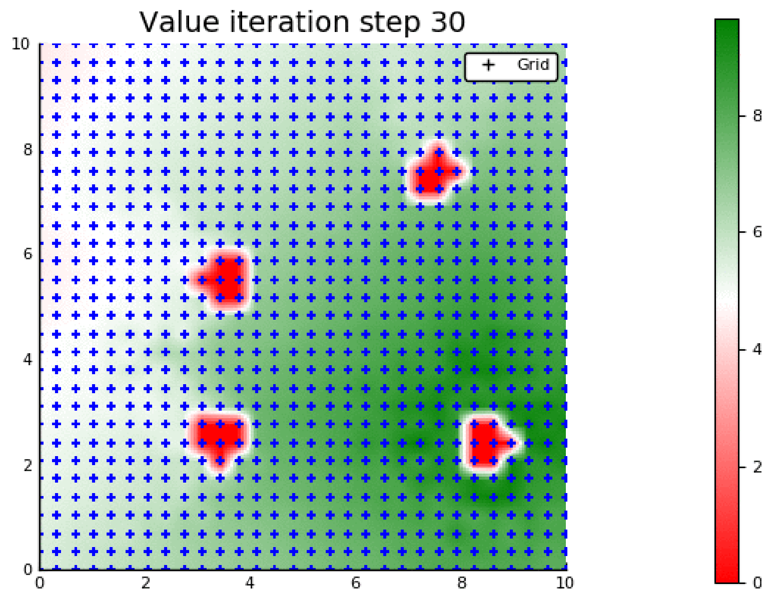
Fitted Value Iteration

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)

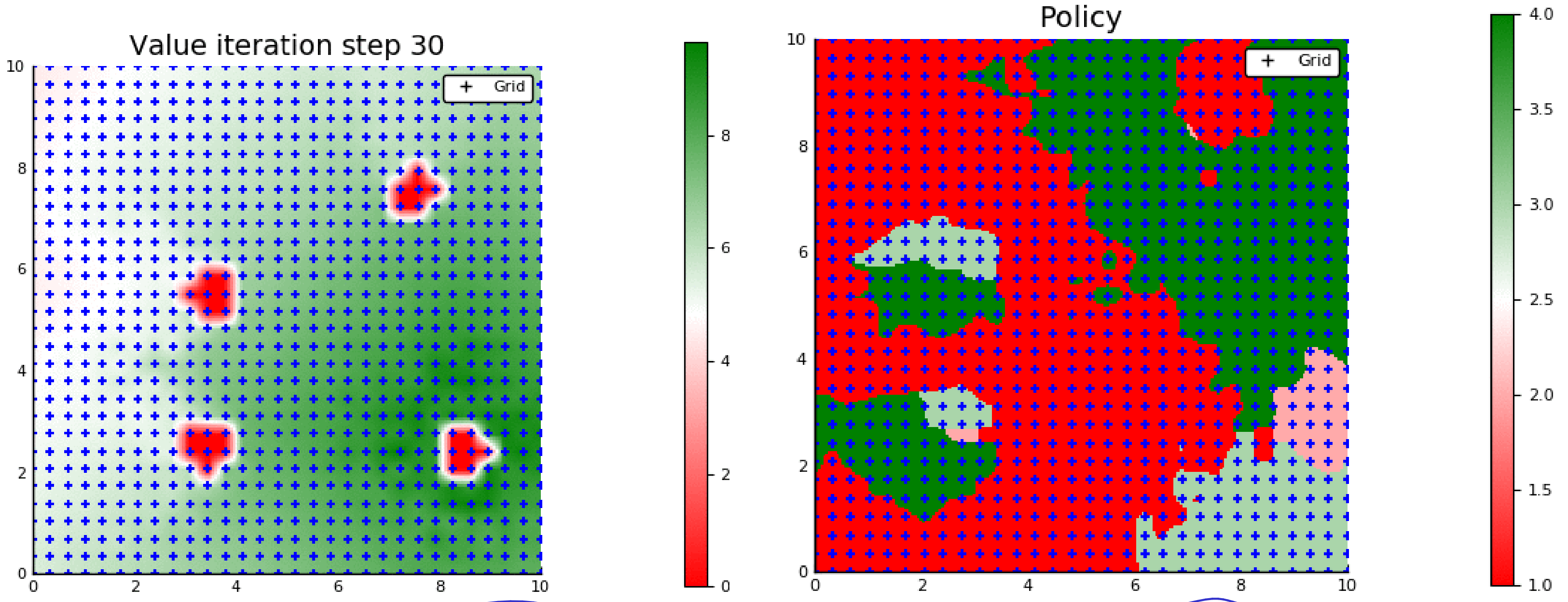
Fitted Value Iteration

while not converged

$\theta \leftarrow \theta'$

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

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



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

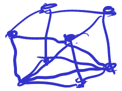
Function Approximation

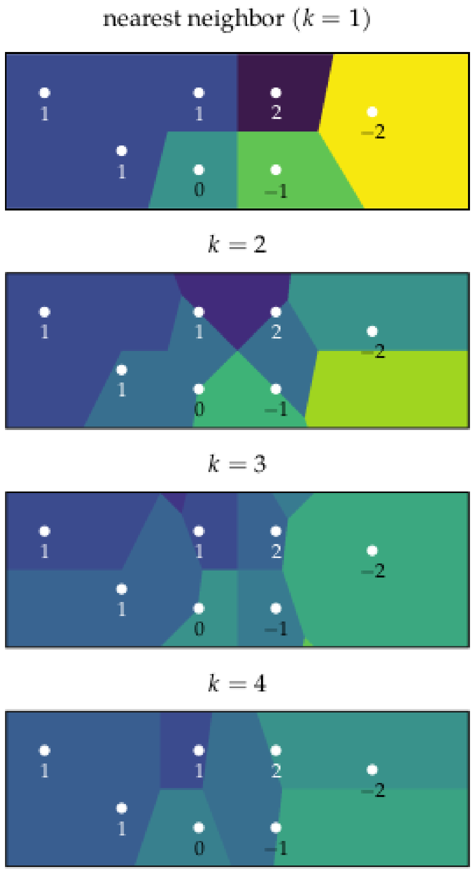
Function Approximation

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

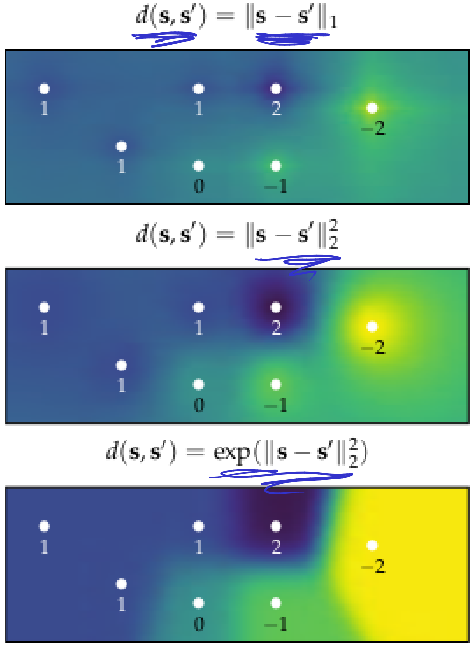
Function Approximation

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





Kernel



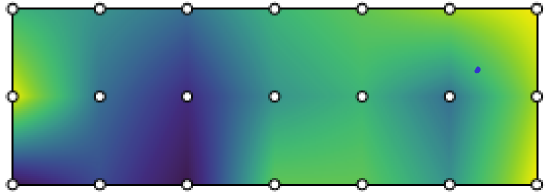
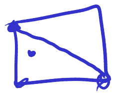


Figure 8.9. Two-dimensional linear interpolation over a 3×7 grid.



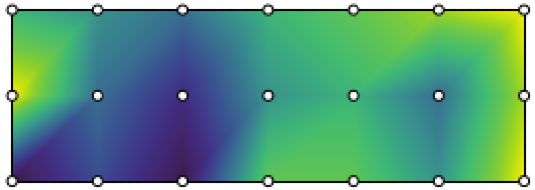
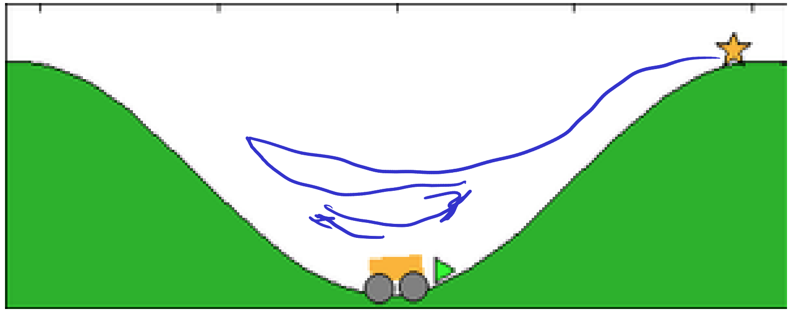


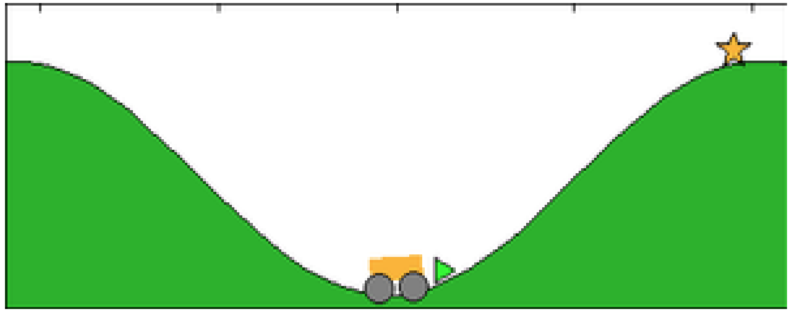
Figure 8.10. Two-dimensional simplex interpolation over a 3×7 grid.

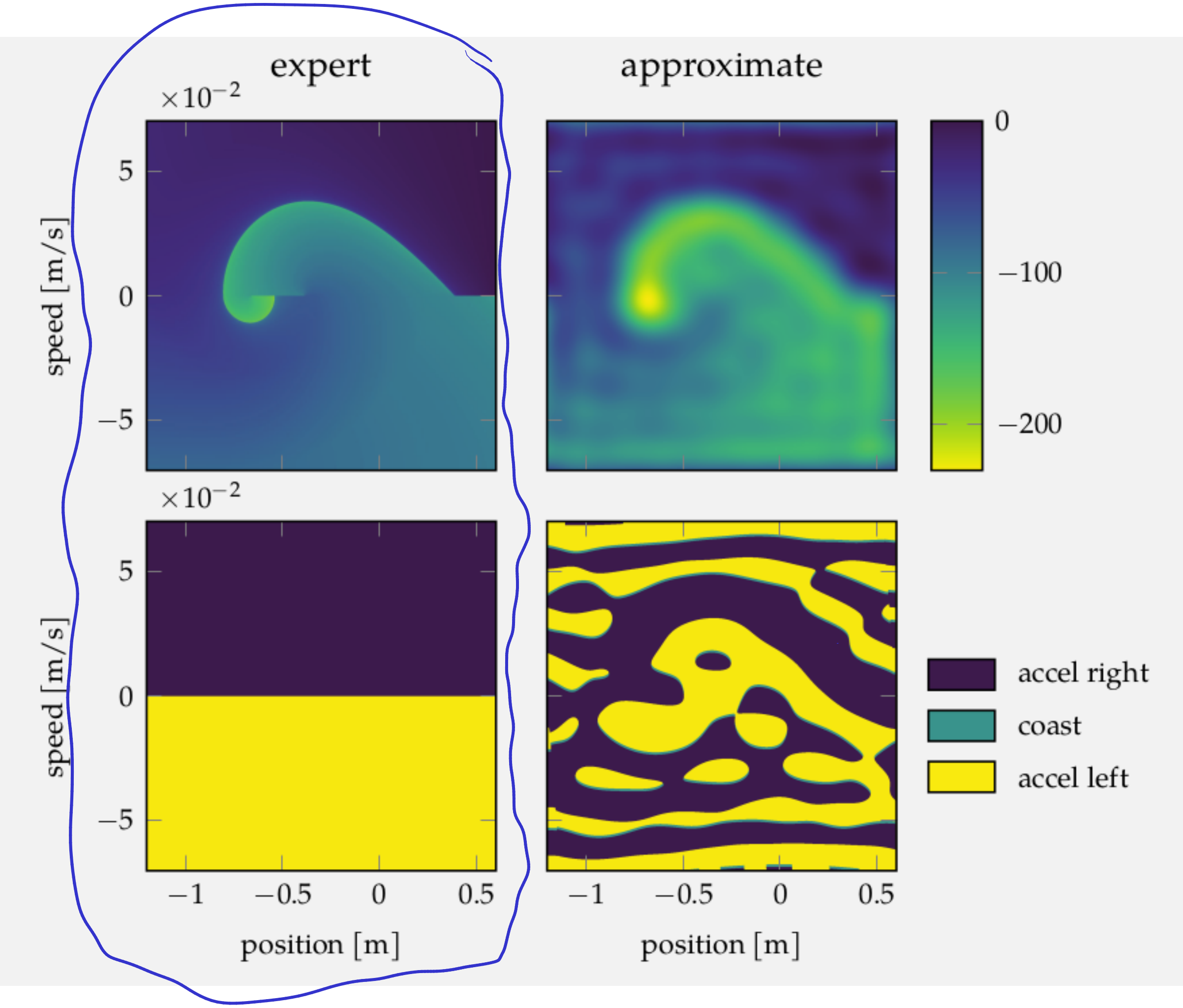
Function Approximation: Mountain Car

Function Approximation: Mountain Car



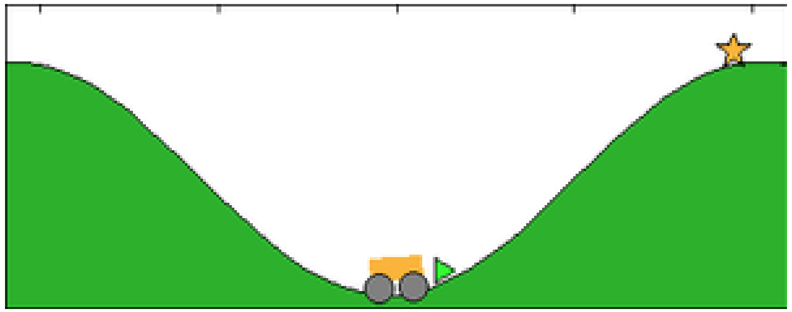
Function Approximation: Mountain Car





(Fourier, 17 params)

Function Approximation: Mountain Car



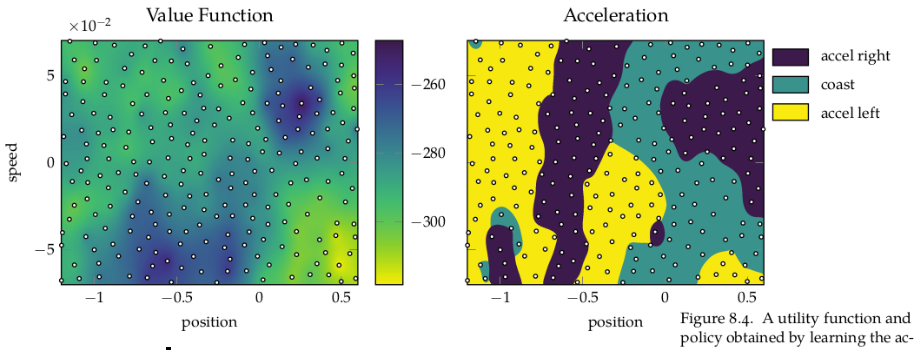
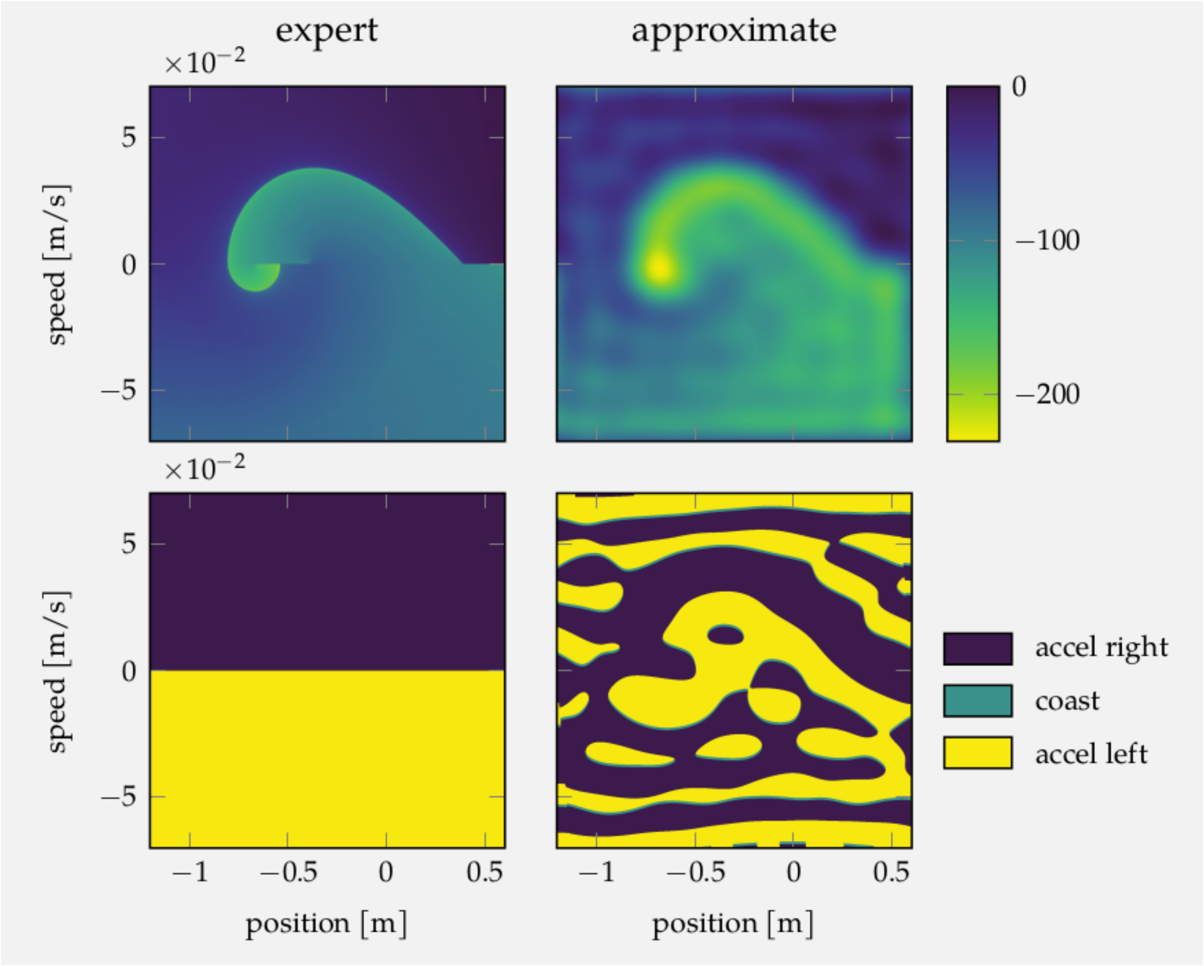


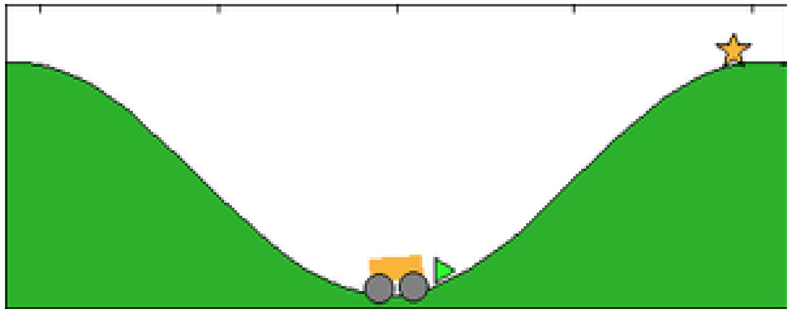
Figure 8.4. A utility function and policy obtained by learning the action values for a finite set of states (white) in the mountain car problem using the distance function $\|\mathbf{s} - \mathbf{s}'\|_2 + 0.1$.

(Kernel, > 100 params)



(Fourier, 17 params)

Function Approximation: Mountain Car



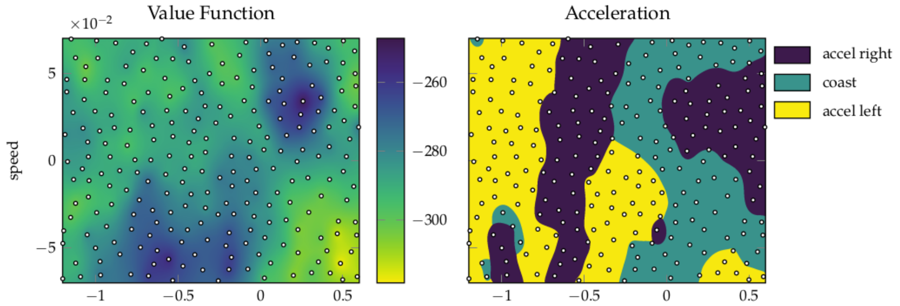
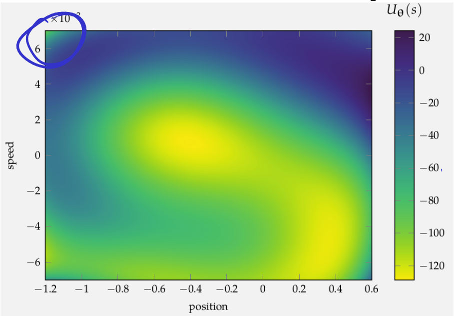
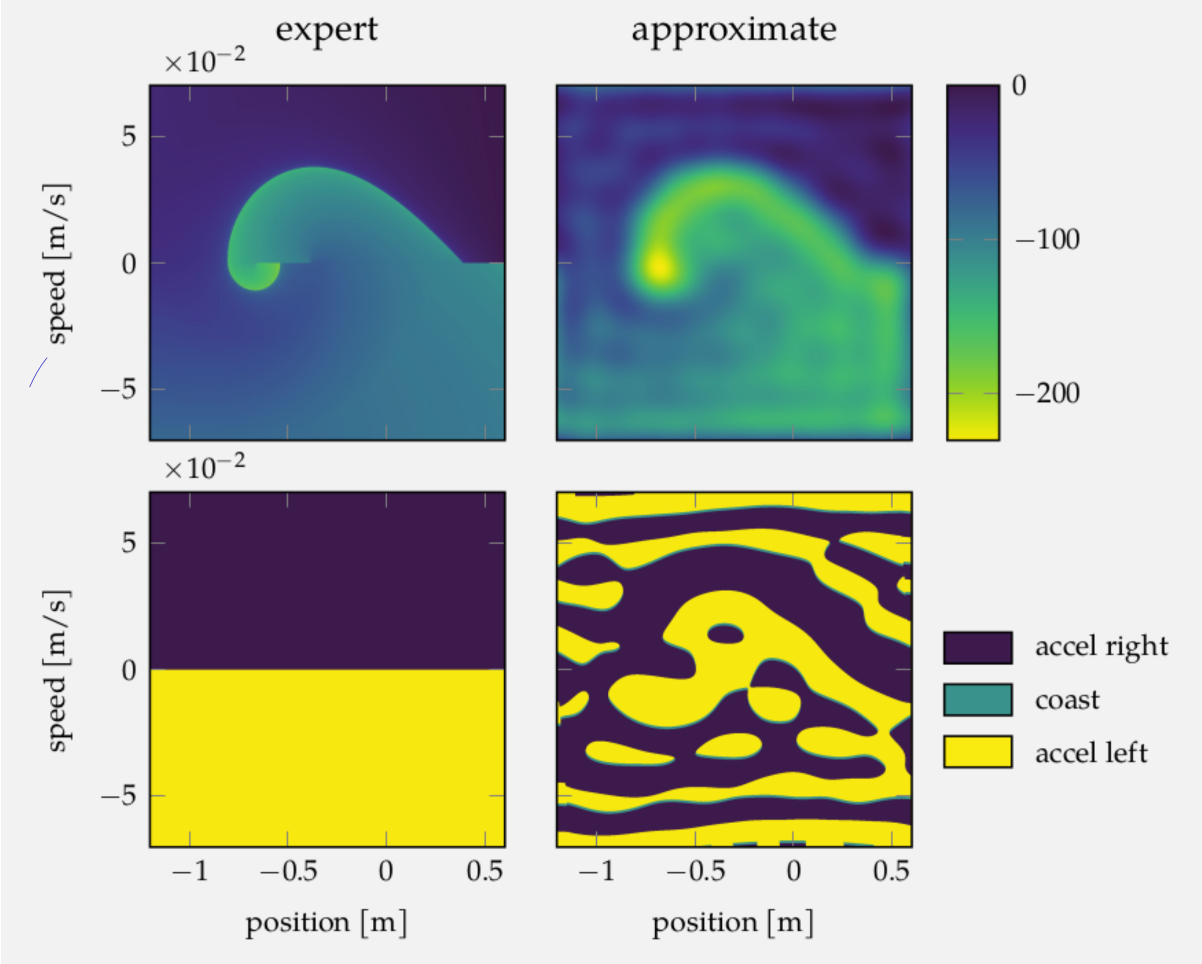


Figure 8.4. A utility function and policy obtained by learning the action values for a finite set of states (white) in the mountain car problem using the distance function $\|\mathbf{s} - \mathbf{s}'\|_2 + 0.1$.

(Kernel, > 100 params)



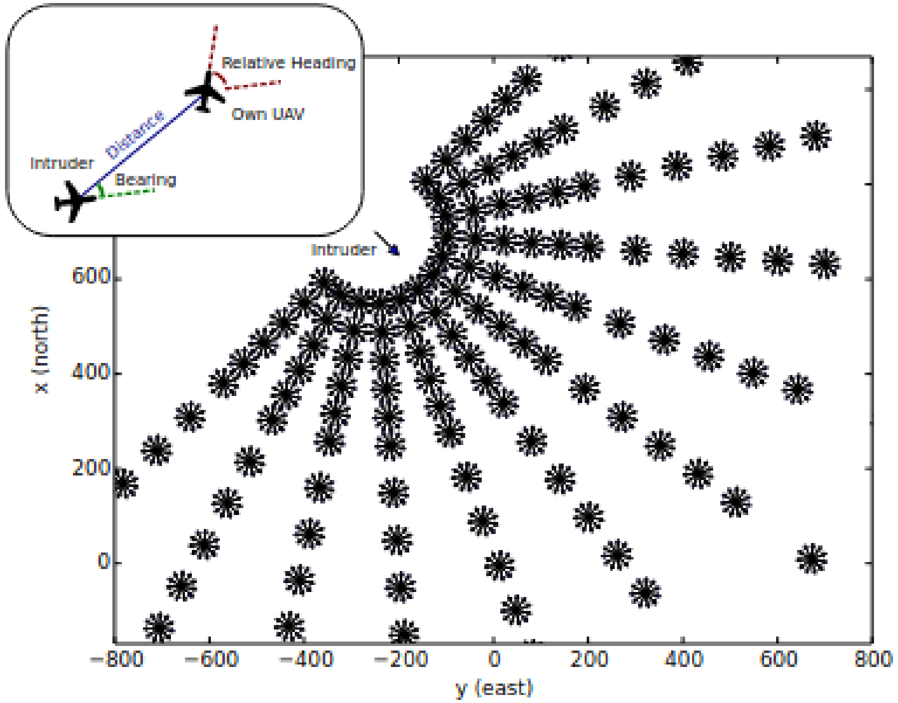
(Polynomial, 28 params)

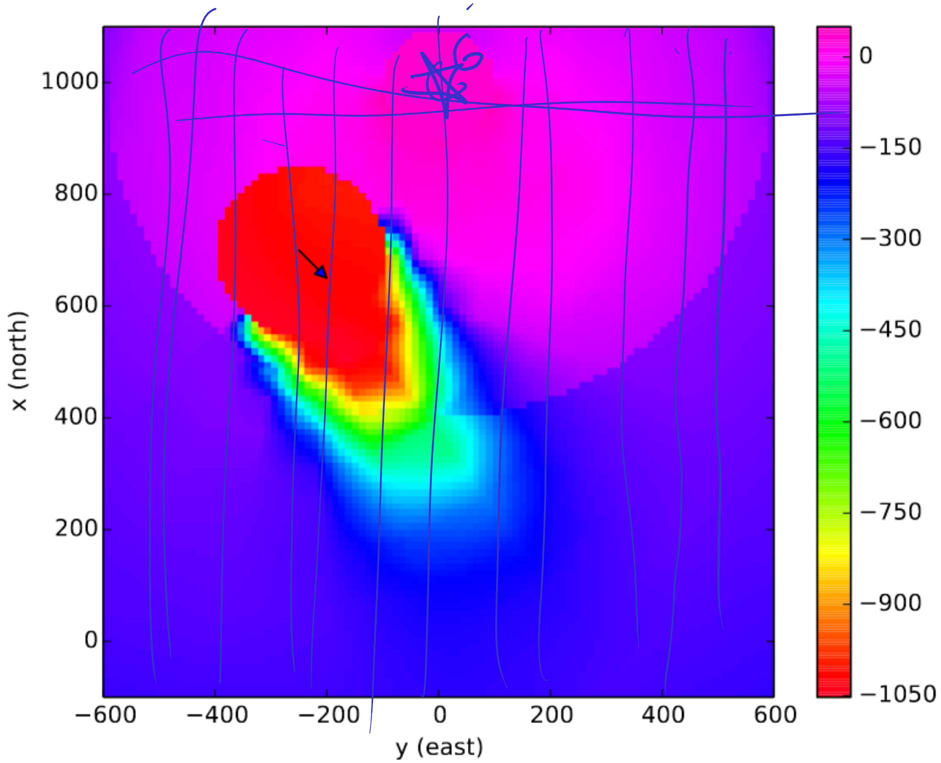


(Fourier, 17 params)

Function Approximation

(x, y, θ)
(x, y, θ)

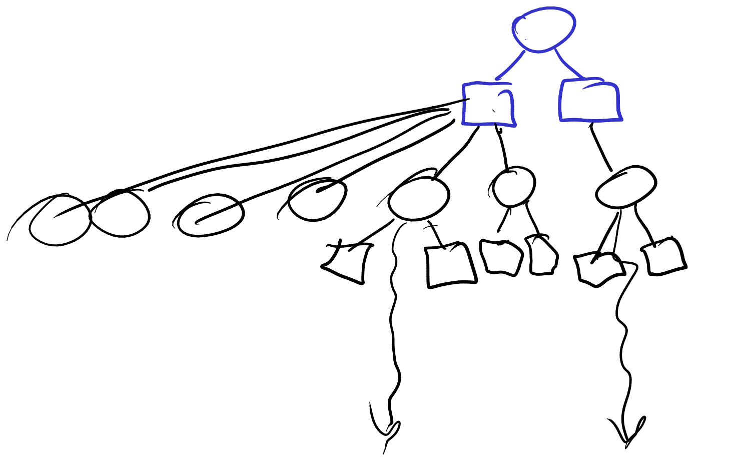




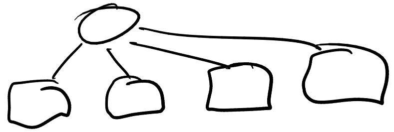
Break

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

discrete A, continuous S

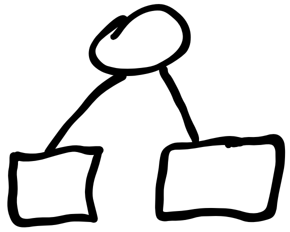


continuous

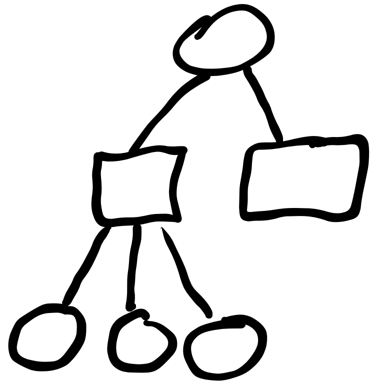


3. Sparse Tree Search/Progressive Widening

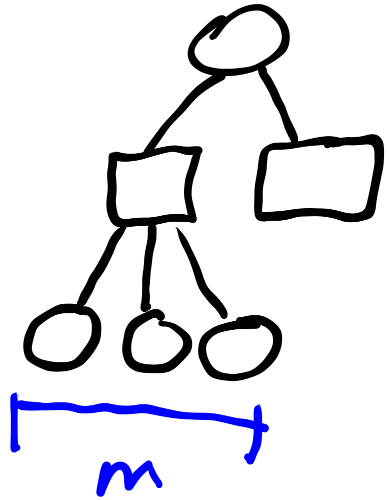
3. Sparse Tree Search/Progressive Widening



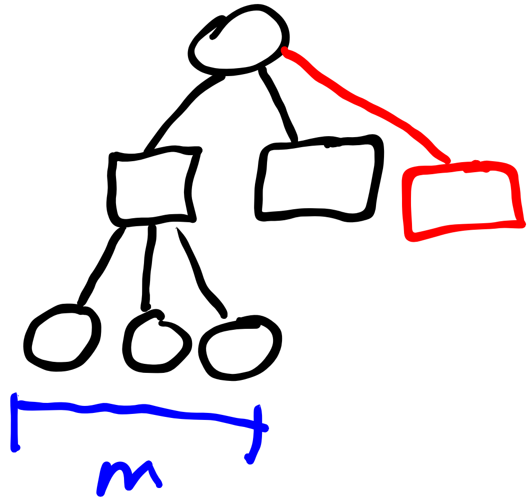
3. Sparse Tree Search/Progressive Widening



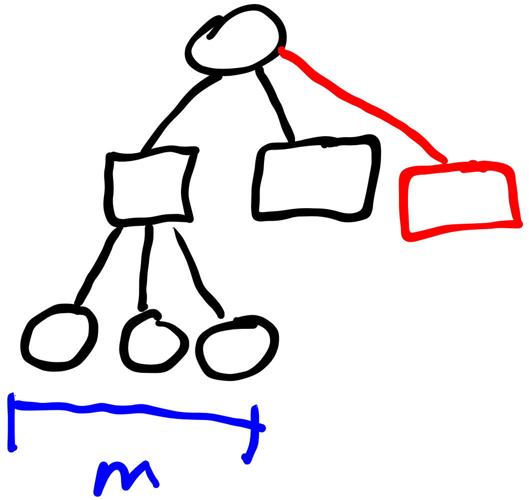
3. Sparse Tree Search/Progressive Widening



3. Sparse Tree Search/Progressive Widening

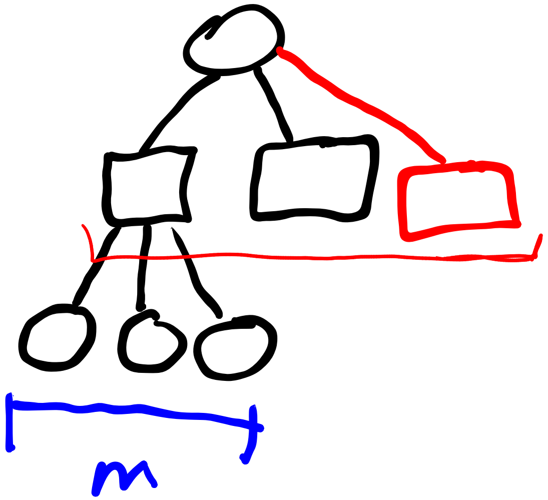


3. Sparse Tree Search/Progressive Widening



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

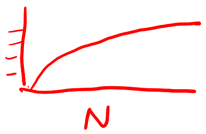
3. Sparse Tree Search/Progressive Widening



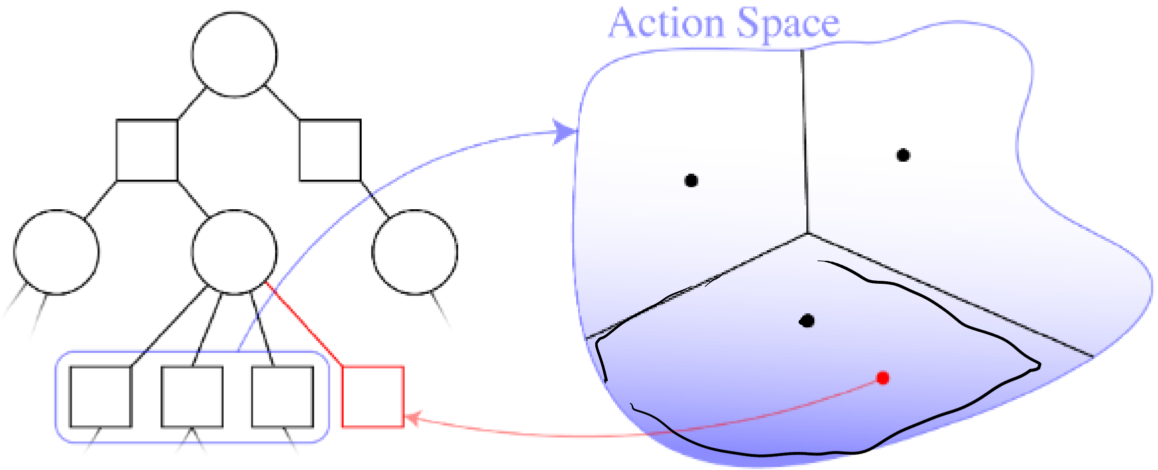
number children of state node

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

times node has been visited



Voronoi Progressive Widening



Online Tree Search Planner Voronoi Progressive Widening

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?

1. LQR
2. Value Function Approx
3. Sparse Sampling / P.W.