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?

Guiding Questions

Guiding Questions

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

Current Tool-Belt

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!

Today: Four Tools

1. Linear Dynamics, Quadratic Reward

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)

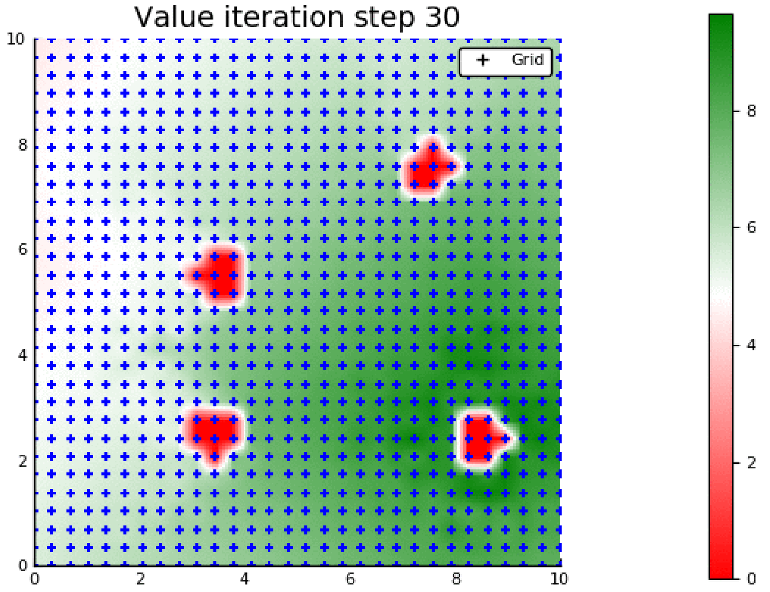
Fitted Value Iteration

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)

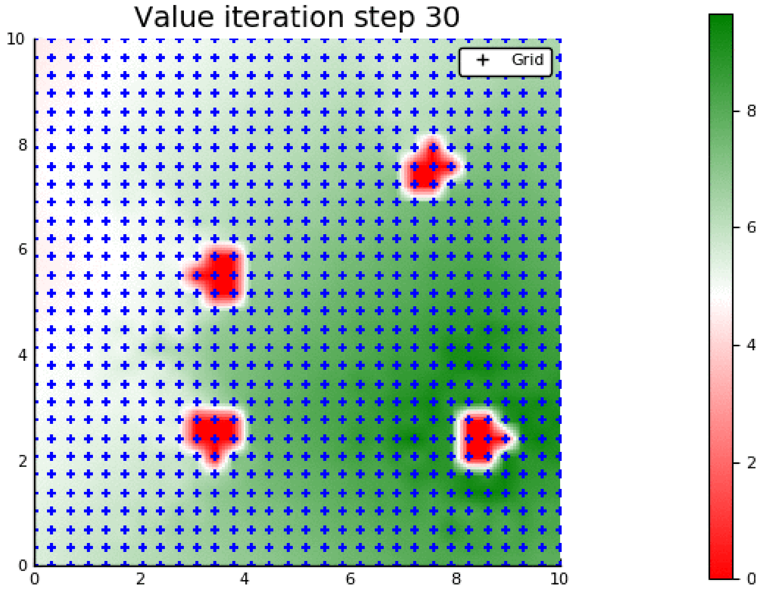
Fitted Value Iteration

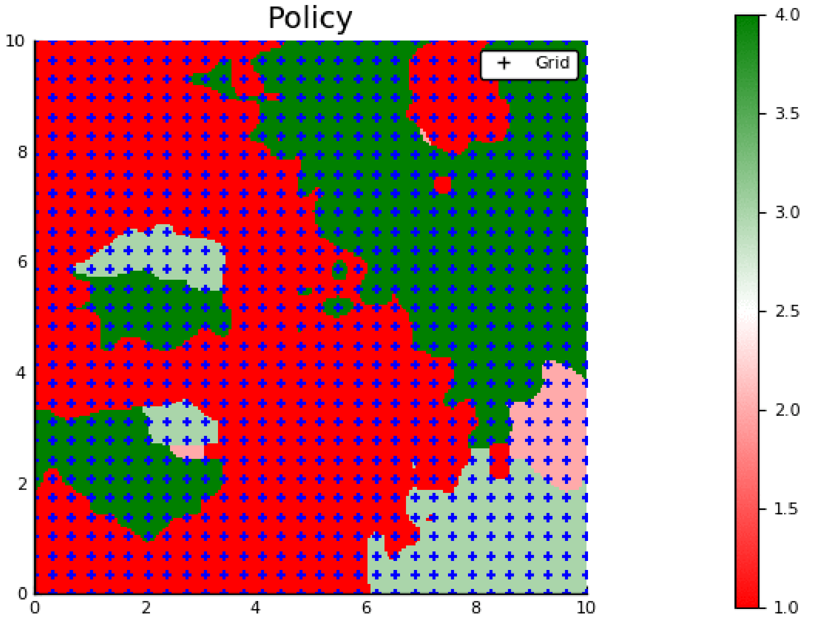
while not converged

$\theta \leftarrow \theta'$

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

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





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

Function Approximation

Weighting of 2^d points

Weighting of only $d + 1$ points!

Function Approximation

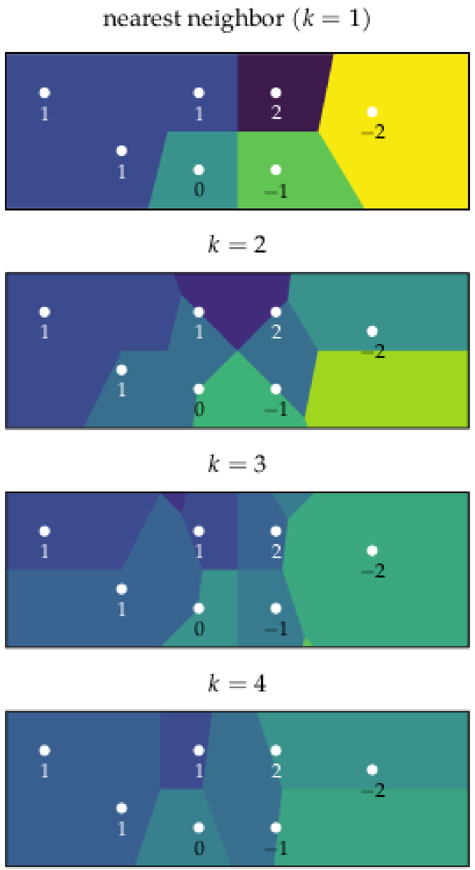
- Global: (e.g. Fourier, neural network)
- Local: (e.g. simplex interpolation)

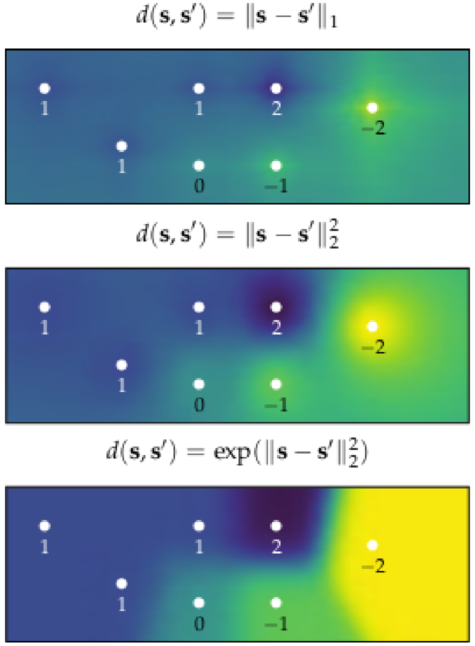
Weighting of 2^d points

Weighting of only $d + 1$ points!

Function Approximation

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





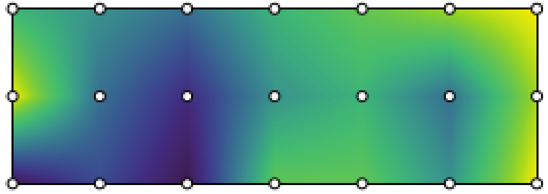


Figure 8.9. Two-dimensional linear interpolation over a 3×7 grid.

Weighting of 2^d points

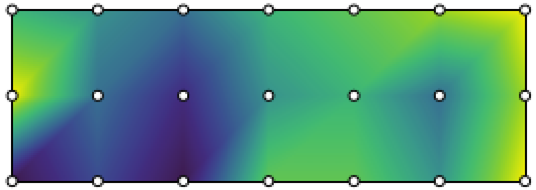
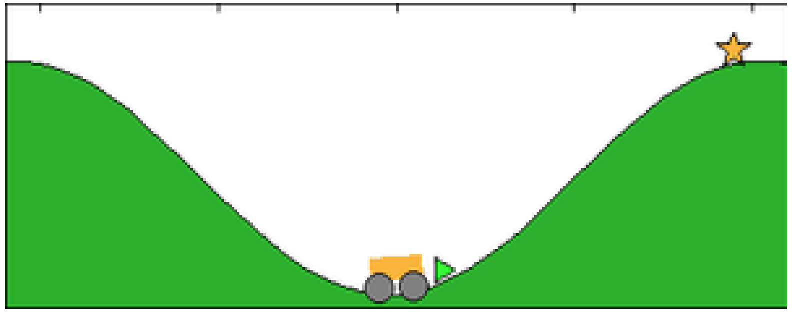


Figure 8.10. Two-dimensional simplex interpolation over a 3×7 grid.

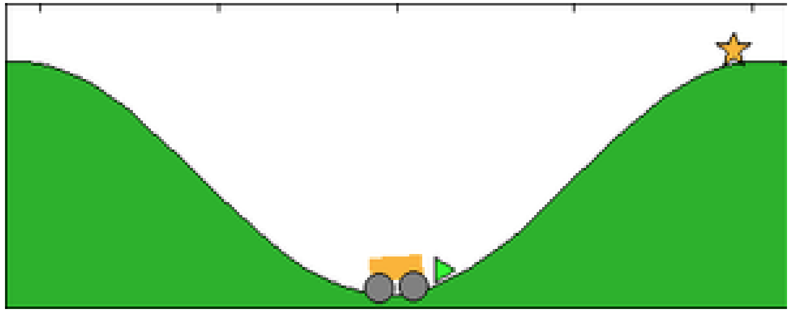
Weighting of only $d + 1$ points!

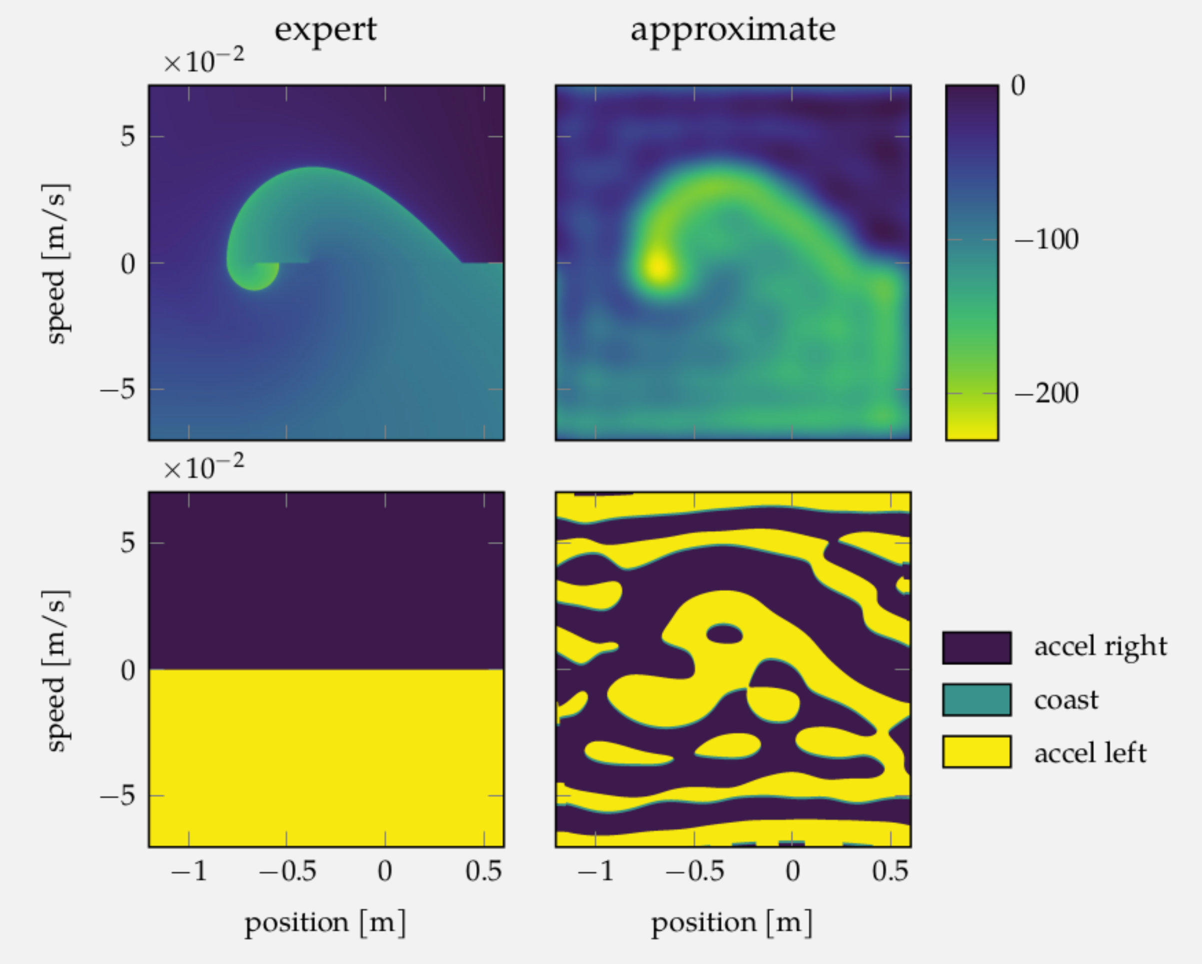
Function Approximation: Mountain Car

Function Approximation: Mountain Car



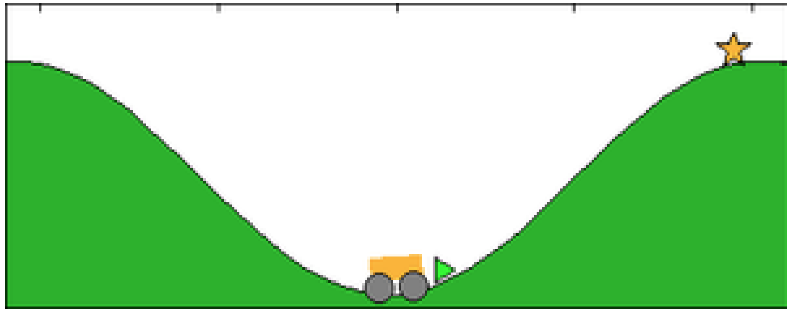
Function Approximation: Mountain Car





(Fourier, 17 params)

Function Approximation: Mountain Car



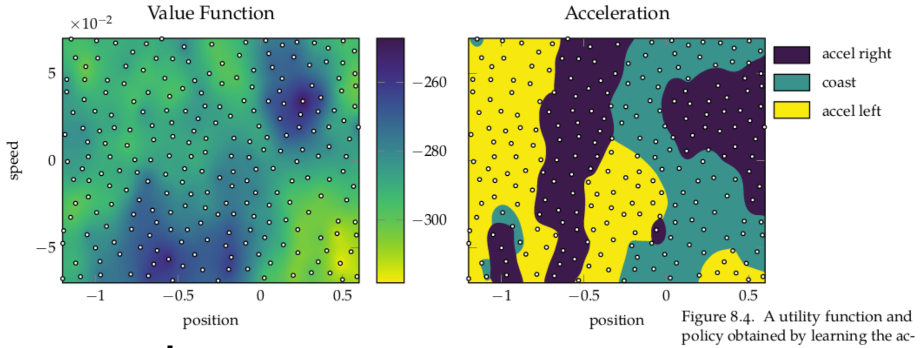
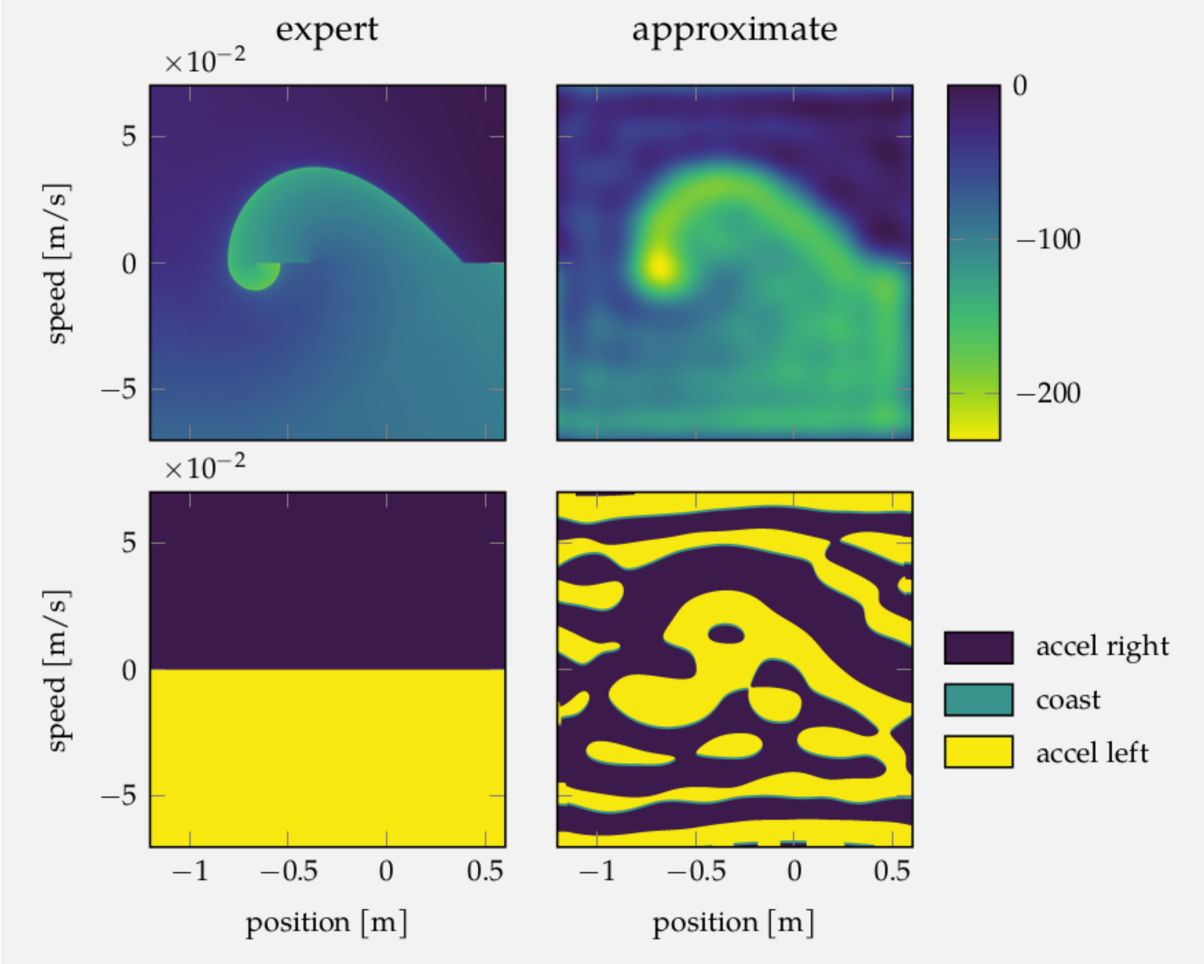


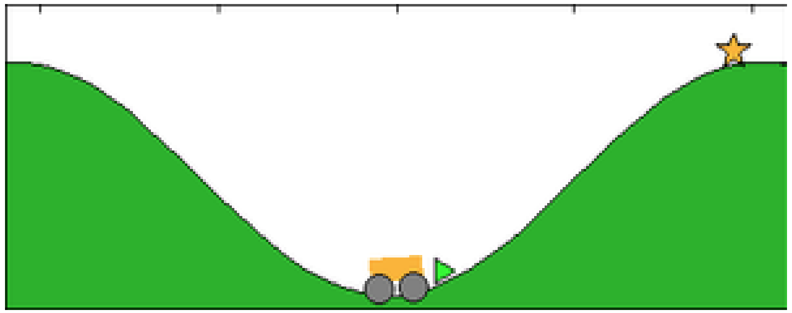
Figure 8.4. A utility function and policy obtained by learning the action values for a finite set of states (white) in the mountain car problem using the distance function $\|\mathbf{s} - \mathbf{s}'\|_2 + 0.1$.

(Kernel, > 100 params)



(Fourier, 17 params)

Function Approximation: Mountain Car



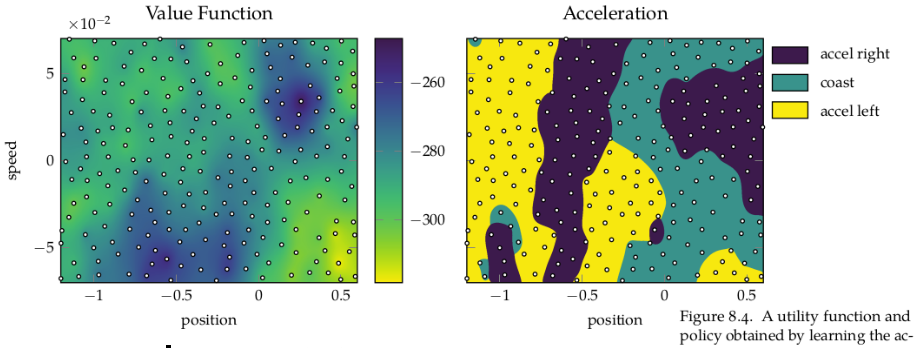
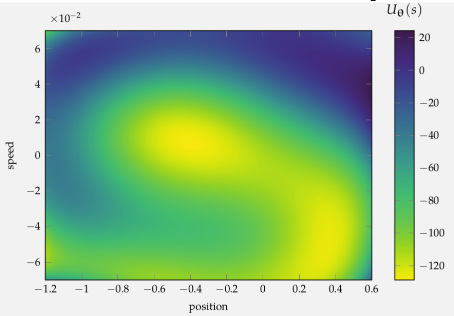
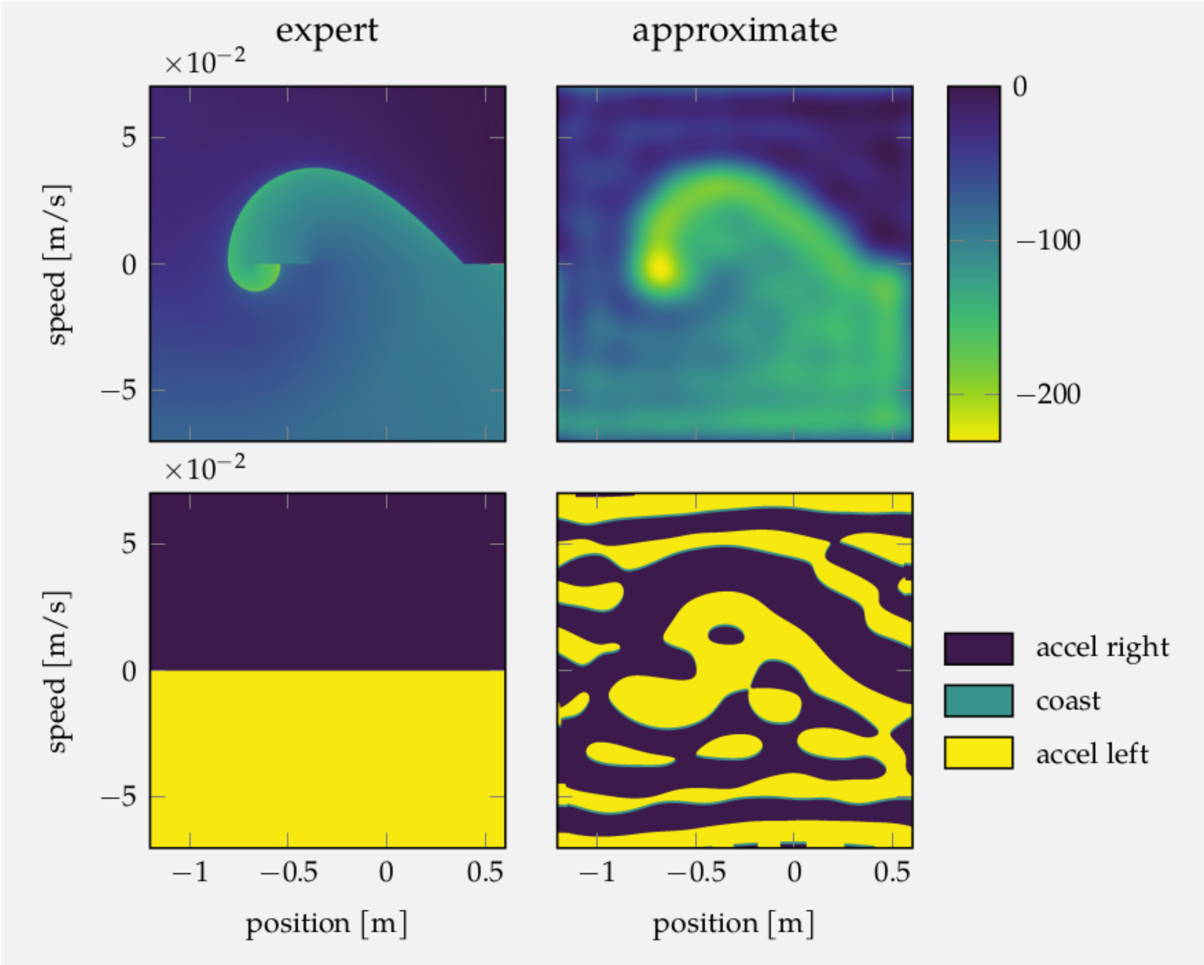


Figure 8.4. A utility function and policy obtained by learning the action values for a finite set of states (white) in the mountain car problem using the distance function $\|\mathbf{s} - \mathbf{s}'\|_2 + 0.1$.

(Kernel, > 100 params)

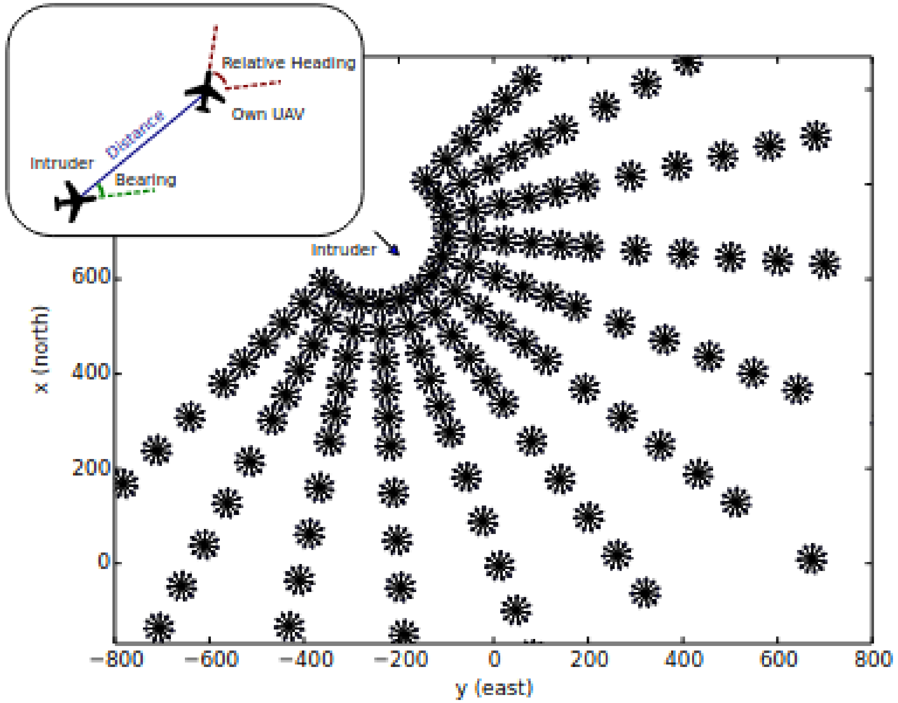


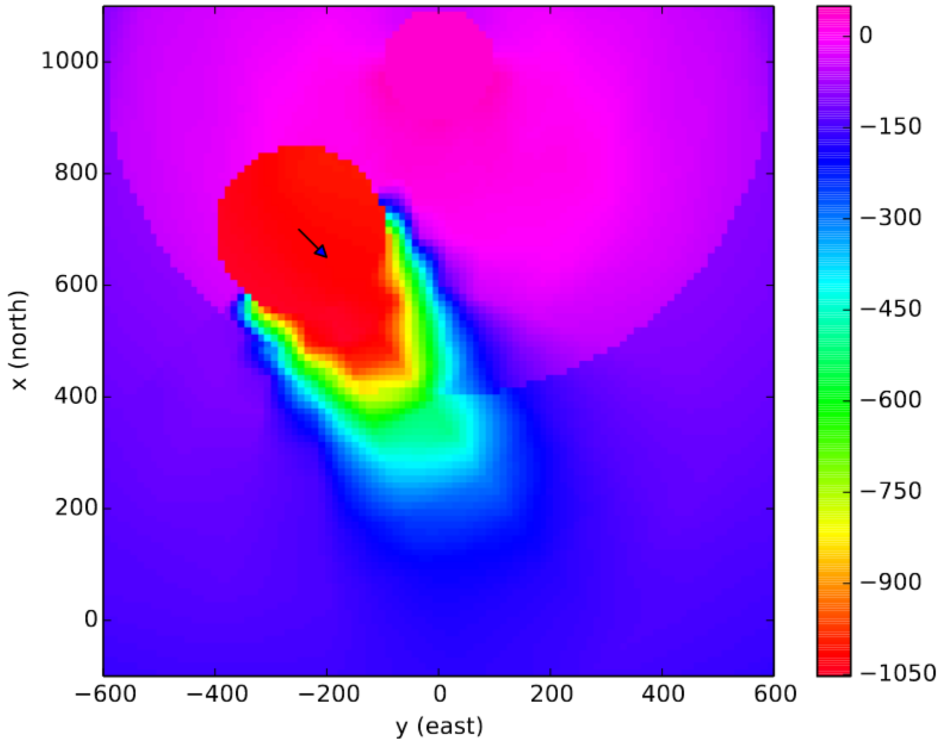
(Polynomial, 28 params)



(Fourier, 17 params)

Function Approximation



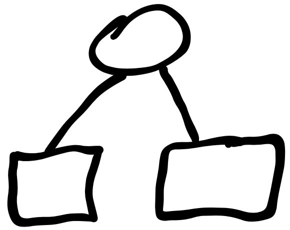


Break

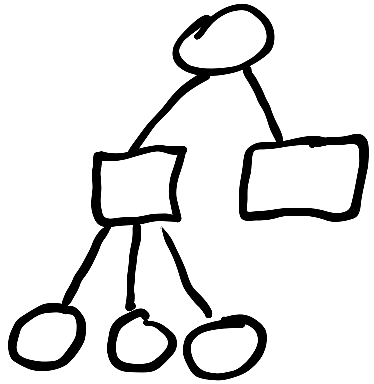
What will a Monte Carlo Tree Search tree look like if run on a problem with continuous spaces?

3. Sparse Tree Search/Progressive Widening

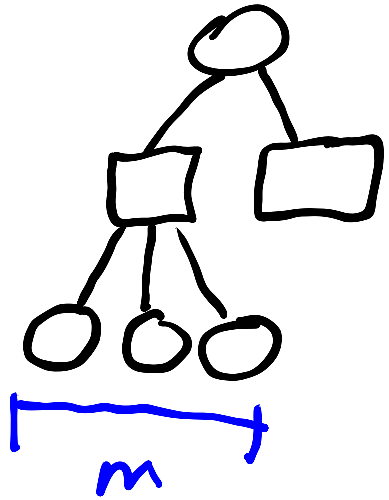
3. Sparse Tree Search/Progressive Widening



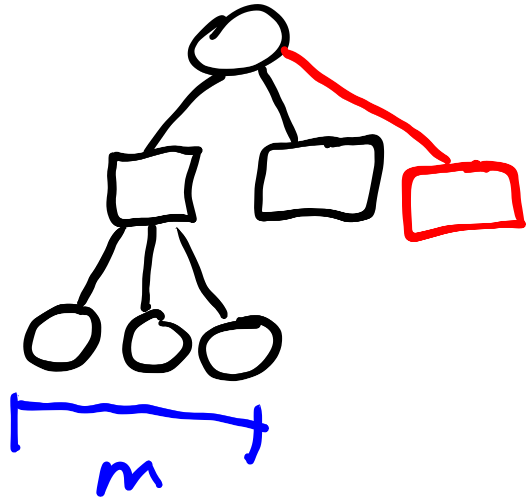
3. Sparse Tree Search/Progressive Widening



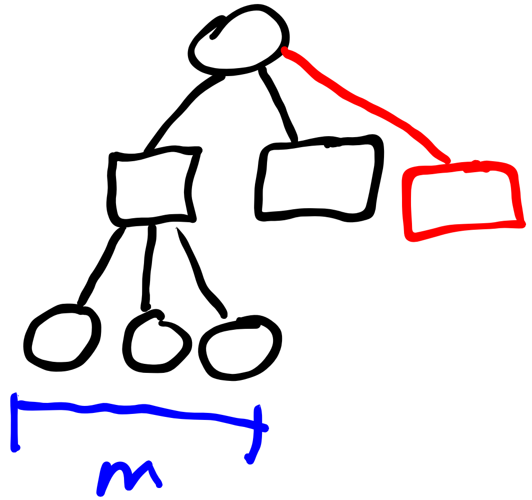
3. Sparse Tree Search/Progressive Widening



3. Sparse Tree Search/Progressive Widening

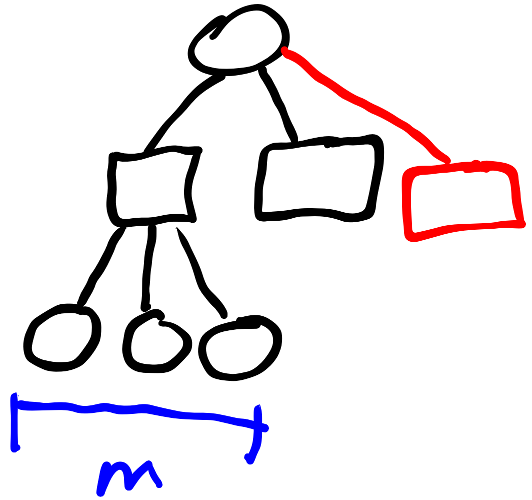


3. Sparse Tree Search/Progressive Widening

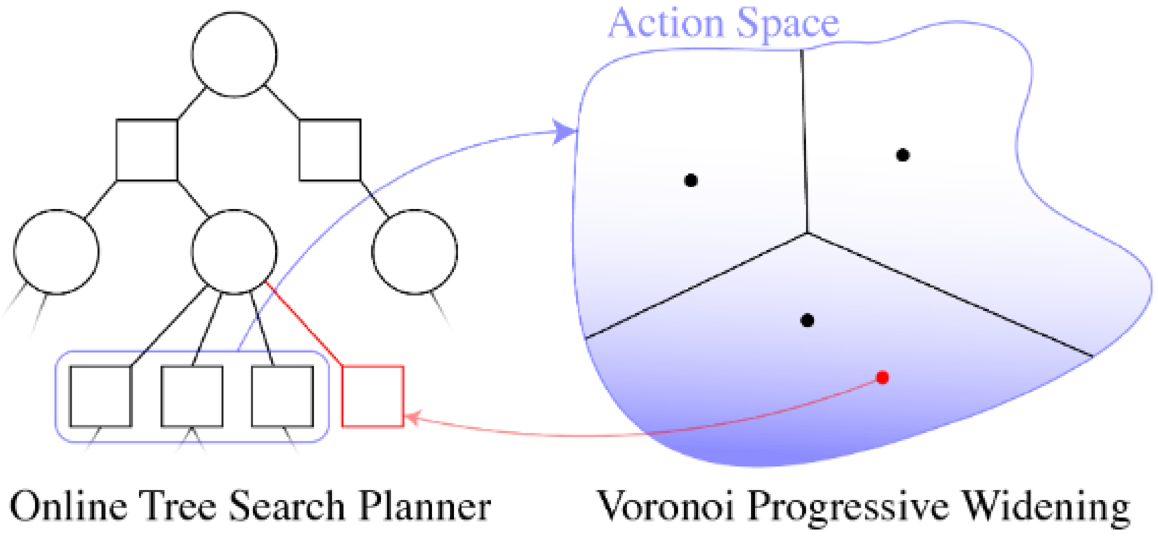


add new branch if $C < kN^{\alpha} \quad (\alpha < 1)$

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?