

# CS 747, Autumn 2022: Lecture 18

Shivaram Kalyanakrishnan

Department of Computer Science and Engineering  
Indian Institute of Technology Bombay

Autumn 2022

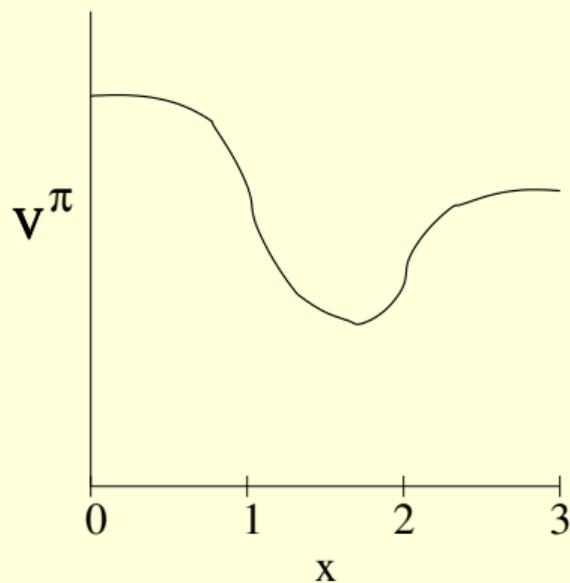
# Reinforcement Learning

1. Tile coding
2. Issues in control with function approximation
3. The case for policy search

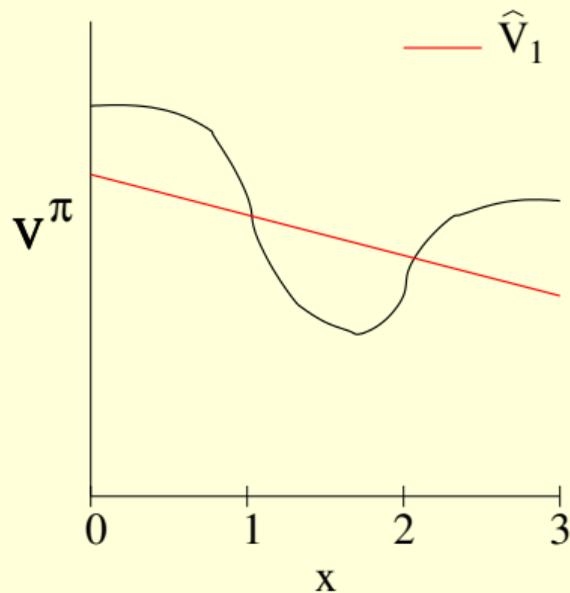
# Reinforcement Learning

1. Tile coding
2. Issues in control with function approximation
3. The case for policy search

# How Good is Linear Function Approximation?

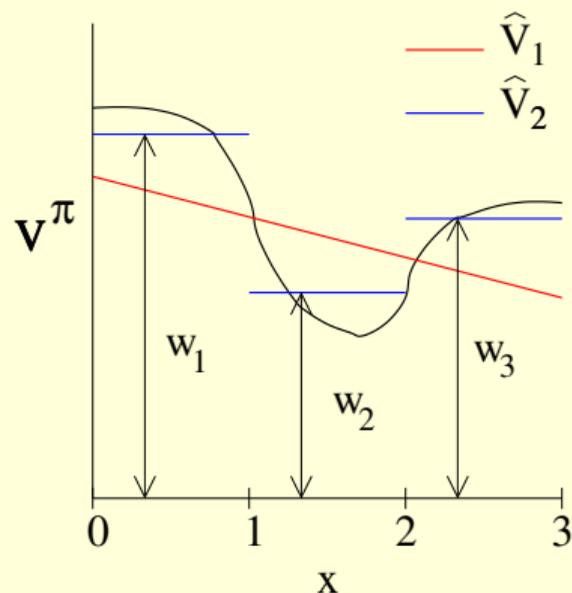


# How Good is Linear Function Approximation?



$$\hat{V}_1(x) = w_1 x + w_2.$$

# How Good is Linear Function Approximation?



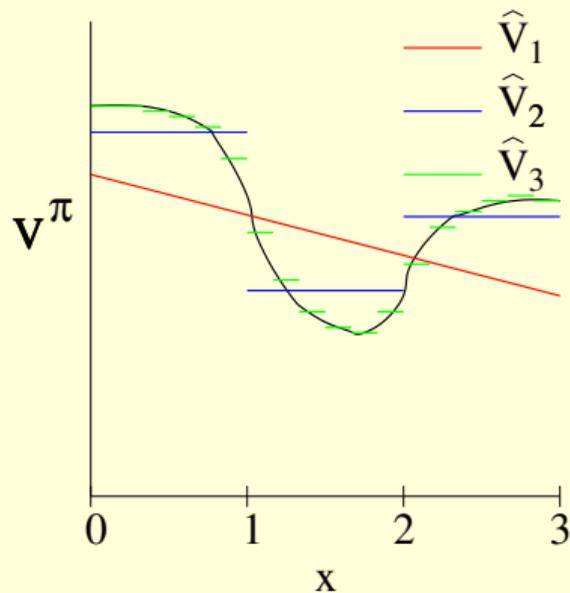
$$b_1 = \begin{cases} 1 & \text{if } 0 \leq x < 1, \\ 0 & \text{otherwise.} \end{cases}$$

$$b_2 = \begin{cases} 1 & \text{if } 1 \leq x < 2, \\ 0 & \text{otherwise.} \end{cases}$$

$$b_3 = \begin{cases} 1 & \text{if } 2 \leq x < 3, \\ 0 & \text{otherwise.} \end{cases}$$

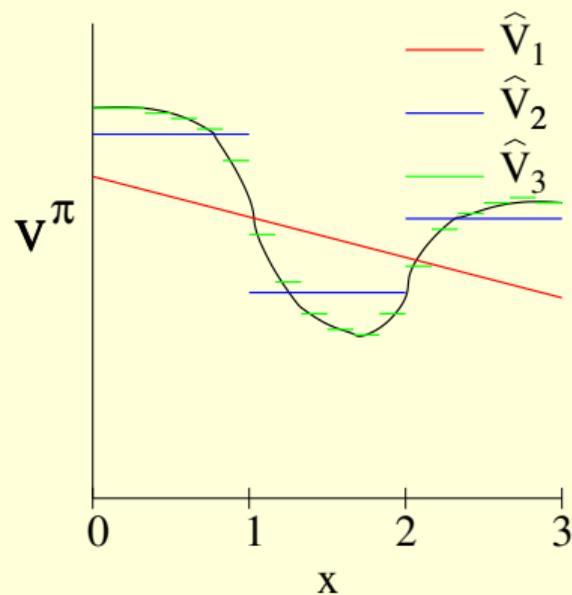
$$\hat{V}_2(x) = w_1 b_1 + w_2 b_2 + w_3 b_3.$$

# How Good is Linear Function Approximation?



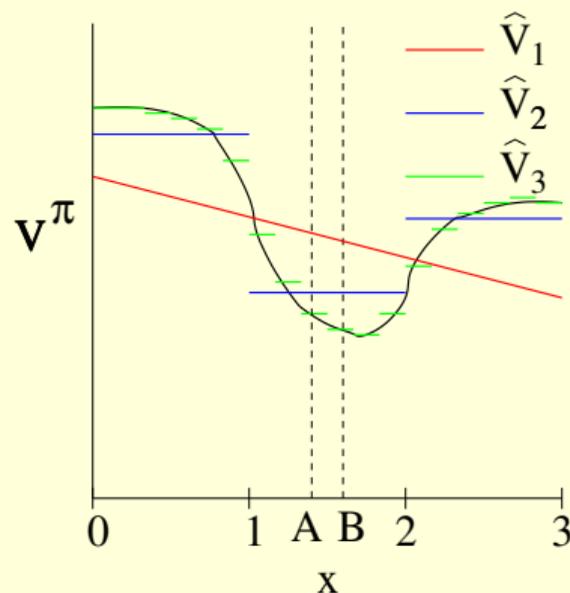
$\hat{V}_3(x)$ : 18 piece-wise constants.

# How Good is Linear Function Approximation?



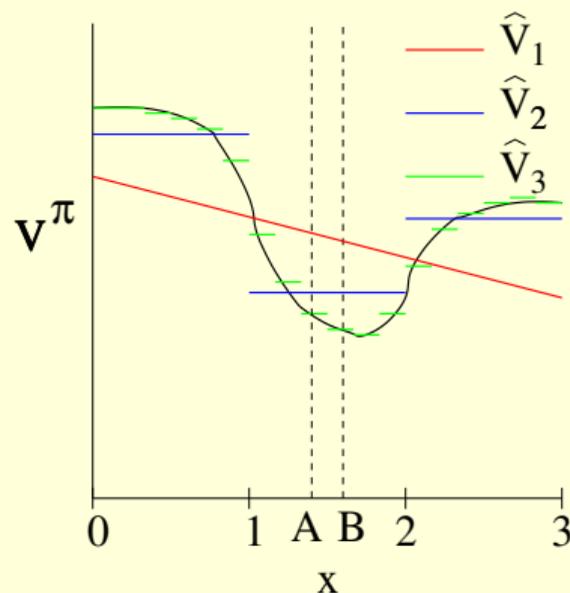
- Is  $\hat{V}^3$  the obvious choice?

# How Good is Linear Function Approximation?



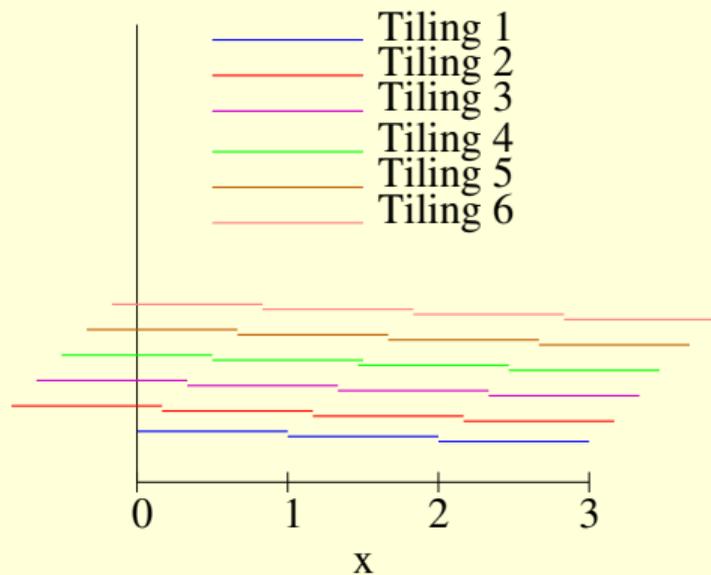
- Is  $\hat{V}^3$  the obvious choice?
- $\hat{V}^3$  has the highest **resolution**, but does not **generalise** well.

# How Good is Linear Function Approximation?



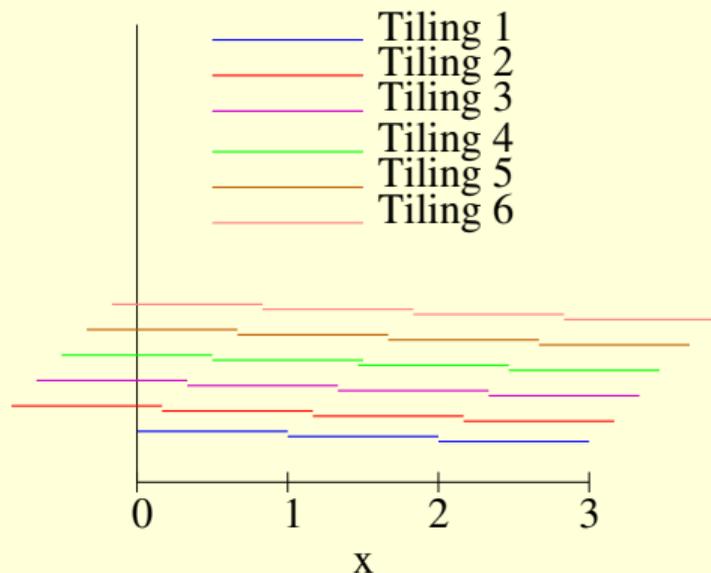
- Is  $\hat{V}^3$  the obvious choice?
- $\hat{V}^3$  has the highest **resolution**, but does not **generalise** well.
- How to achieve high resolution along with generalisation?

# Tile coding



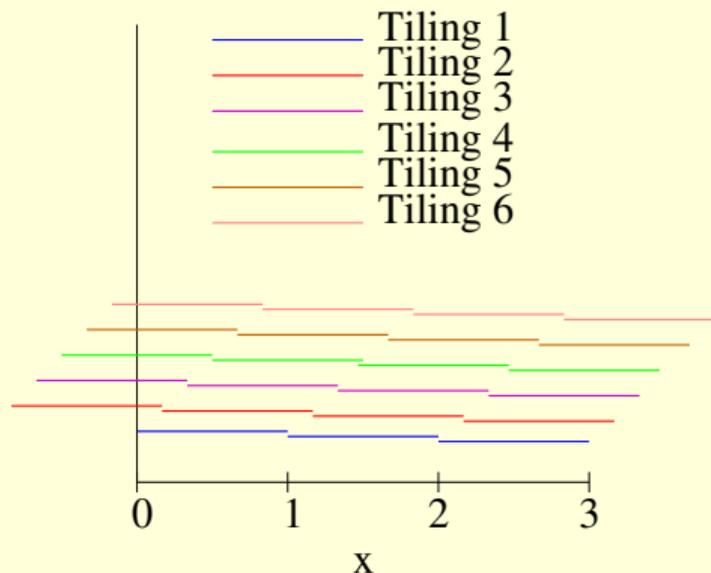
- A **tiling** partitions  $x$  into equal-width regions called **tiles**.

# Tile coding



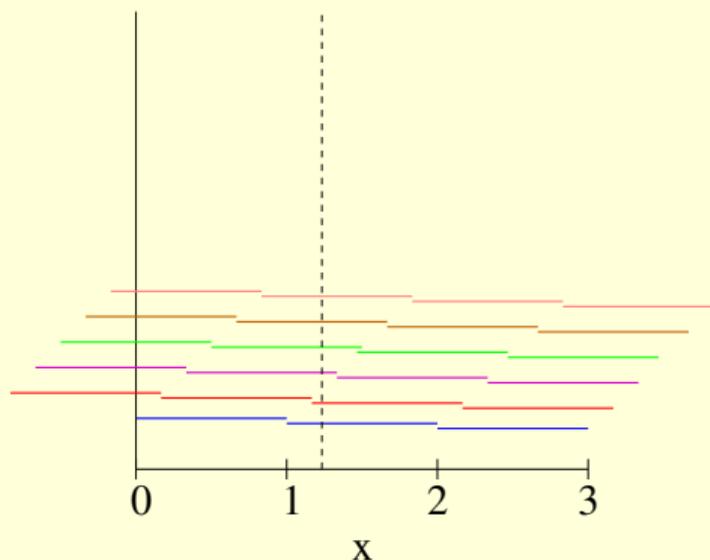
- A **tiling** partitions  $x$  into equal-width regions called **tiles**.
- Multiple tilings (say  $m$ ) are created, each with an **offset** ( $1/m$  tile width) from the previous.

# Tile coding



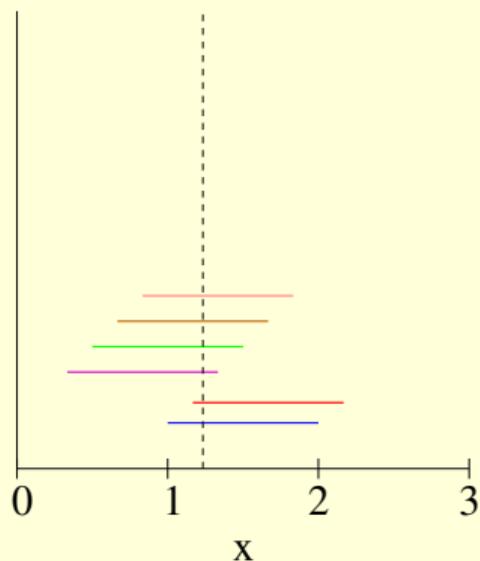
- A **tiling** partitions  $x$  into equal-width regions called **tiles**.
- Multiple tilings (say  $m$ ) are created, each with an **offset** ( $1/m$  tile width) from the previous.
- Each tile has an associated **weight**.

# Tile coding



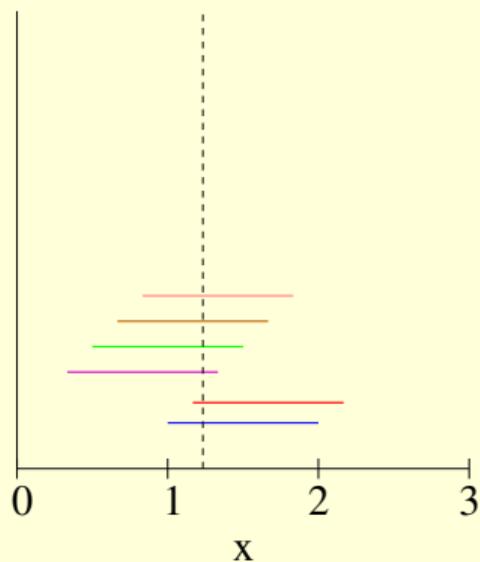
- A **tiling** partitions  $x$  into equal-width regions called **tiles**.
- Multiple tilings (say  $m$ ) are created, each with an **offset** ( $1/m$  tile width) from the previous.
- Each tile has an associated **weight**.
- The function value of a point is the sum of the weights of the tiles intersecting it (one per tiling).

# Tile coding



- A **tiling** partitions  $x$  into equal-width regions called **tiles**.
- Multiple tilings (say  $m$ ) are created, each with an **offset** ( $1/m$  tile width) from the previous.
- Each tile has an associated **weight**.
- The function value of a point is the sum of the weights of the tiles intersecting it (one per tiling).

# Tile coding



- Each tile is a **binary** feature.
- **Tile width** and the **number of tilings** determine generalisation, resolution.
- Observe that two points more than  $(\text{tile width} / \text{number of tilings})$  apart can be given arbitrary function values.

# Representing $\hat{Q}$

- Given a feature value  $x$  as input, the corresponding set of tilings  $F : \mathbb{R} \rightarrow \mathbb{R}$  returns the sum of the weights of the tiles activated by  $x$ .

# Representing $\hat{Q}$

- Given a feature value  $x$  as input, the corresponding set of tilings  $F : \mathbb{R} \rightarrow \mathbb{R}$  returns the sum of the weights of the tiles activated by  $x$ .
- The usual practice is to have a **separate** set of tilings  $F_{aj} : \mathbb{R} \rightarrow \mathbb{R}$  for each action  $a$  and state feature  $j \in \{1, 2, \dots, d\}$ . Hence

$$\hat{Q}(s, a) = \sum_{j=1}^d F_{aj}(x_j(s)).$$

# Representing $\hat{Q}$

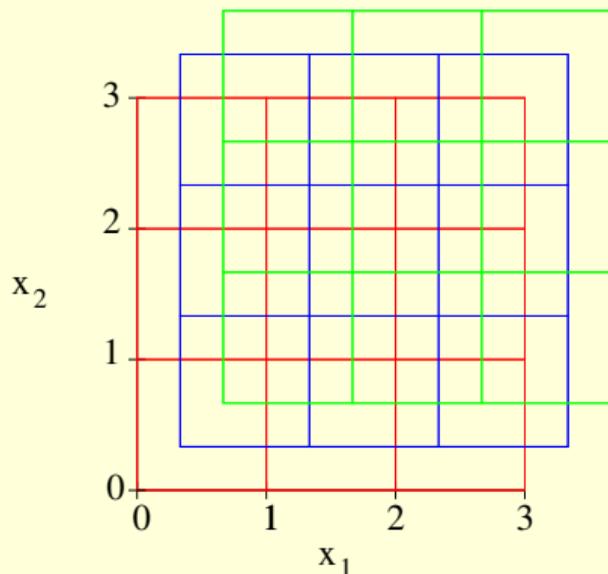
- Given a feature value  $x$  as input, the corresponding set of tilings  $F : \mathbb{R} \rightarrow \mathbb{R}$  returns the sum of the weights of the tiles activated by  $x$ .
- The usual practice is to have a **separate** set of tilings  $F_{aj} : \mathbb{R} \rightarrow \mathbb{R}$  for each action  $a$  and state feature  $j \in \{1, 2, \dots, d\}$ . Hence

$$\hat{Q}(s, a) = \sum_{j=1}^d F_{aj}(x_j(s)).$$

- Usually, tile widths and the number of tilings are configured specifically for each feature. For example, in soccer, could use  $2m$  as tile width for “distance” features, and  $10^\circ$  as tile width for “angle” features.

## 2-d Tile coding

- For representing more complex functions, can also have tilings on **conjunctions of features** (see below for 2 features).



- Introduces more parameters—which could help or hurt.

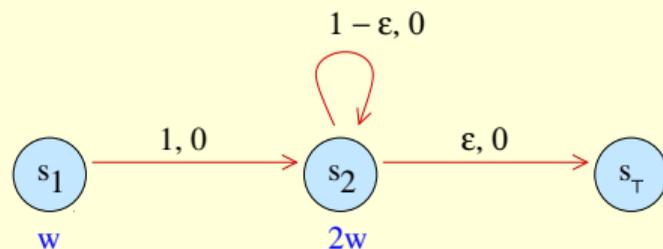
# Tile Coding: Summary

- Linear function approximation does not restrict us to a representation that is linear in the given/raw features.
- Tile coding a standard approach to **discretise** input features and tune both resolution and generalisation.
- **Many empirical successes**, especially in conjunction with Linear Sarsa( $\lambda$ ).
- Common to store weights in a **hash table** (collisions don't seem to hurt much), whose size is set based on practical constraints.
- 1-d tilings most common; rarely see conjunction of 3 or more features.

# Reinforcement Learning

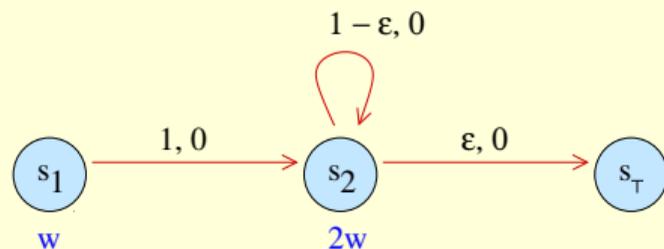
1. Tile coding
2. Issues in control with function approximation
3. The case for policy search

# A Counterexample (Tsitsiklis and Van Roy, 1996)



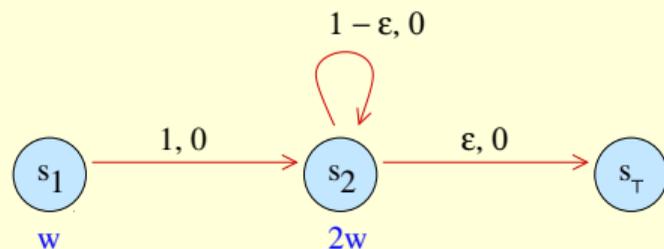
- Prediction problem (policy  $\pi$ ).
- Episodic, start state is  $s_1$ .
- Observe that  $V^\pi(s_1) = V^\pi(s_2) = 0$ .
- Linear function approximation with single parameter  $w$ :  
 $x(s_1) = 1, x(s_2) = 2$ ; hence  $\hat{V}(s_1) = w, \hat{V}(s_2) = 2w$ .

# A Counterexample (Tsitsiklis and Van Roy, 1996)



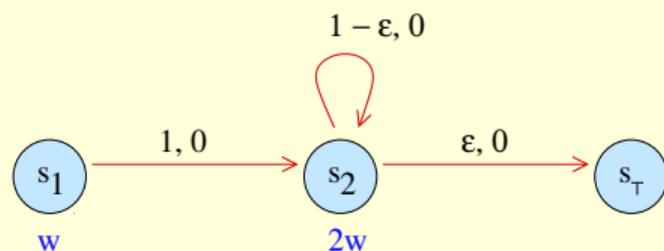
- Prediction problem (policy  $\pi$ ).
- Episodic, start state is  $s_1$ .
- Observe that  $V^\pi(s_1) = V^\pi(s_2) = 0$ .
- Linear function approximation with single parameter  $w$ :  
 $x(s_1) = 1, x(s_2) = 2$ ; hence  $\hat{V}(s_1) = w, \hat{V}(s_2) = 2w$ .
- What's the optimal setting of  $w$ ?

# A Counterexample (Tsitsiklis and Van Roy, 1996)



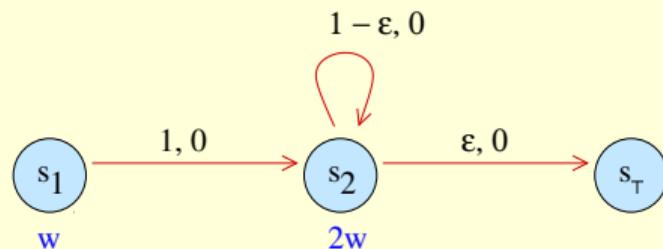
- Prediction problem (policy  $\pi$ ).
- Episodic, start state is  $s_1$ .
- Observe that  $V^\pi(s_1) = V^\pi(s_2) = 0$ .
- Linear function approximation with single parameter  $w$ :  
 $x(s_1) = 1, x(s_2) = 2$ ; hence  $\hat{V}(s_1) = w, \hat{V}(s_2) = 2w$ .
- What's the optimal setting of  $w$ ?
- $w = 0$  gives the exact answer!

# A Counterexample (Tsitsiklis and Van Roy, 1996)



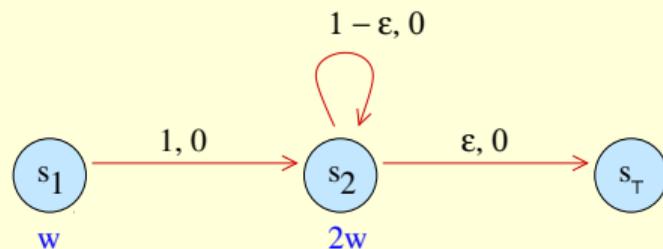
- Prediction problem (policy  $\pi$ ).
- Episodic, start state is  $s_1$ .
- Observe that  $V^\pi(s_1) = V^\pi(s_2) = 0$ .
- Linear function approximation with single parameter  $w$ :  
 $x(s_1) = 1, x(s_2) = 2$ ; hence  $\hat{V}(s_1) = w, \hat{V}(s_2) = 2w$ .
- What's the optimal setting of  $w$ ?
- $w = 0$  gives the exact answer!
- We design an iteration  $w_0 \rightarrow w_1 \rightarrow w_2 \rightarrow \dots$ , and see if it converges to 0.

# A Counterexample (Tsitsiklis and Van Roy, 1996)



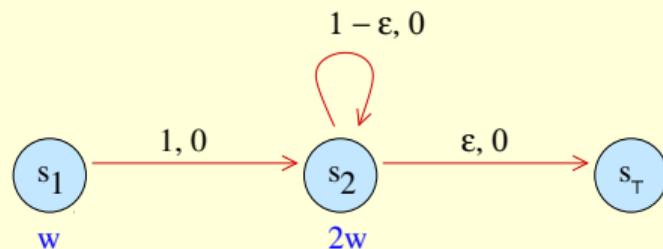
- From state  $s$ , let  $s'$ ,  $r$  be the (random) next state, reward.

# A Counterexample (Tsitsiklis and Van Roy, 1996)



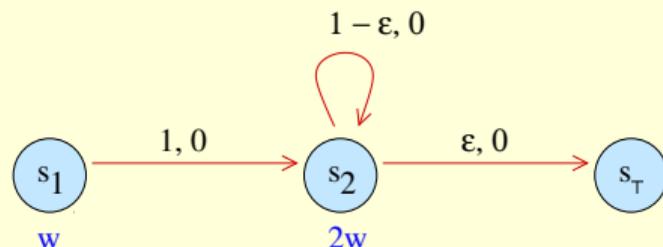
- From state  $s$ , let  $s'$ ,  $r$  be the (random) next state, reward.
- If our current estimate of  $V^\pi$  is  $\hat{V}$ , the bootstrapping idea suggests  $\mathbb{E}_\pi[r + \gamma \hat{V}(s')]$  as a “better estimate” of  $V^\pi(s)$ .

# A Counterexample (Tsitsiklis and Van Roy, 1996)



- From state  $s$ , let  $s'$ ,  $r$  be the (random) next state, reward.
- If our current estimate of  $V^\pi$  is  $\hat{V}$ , the bootstrapping idea suggests  $\mathbb{E}_\pi[r + \gamma \hat{V}(s')]$  as a “better estimate” of  $V^\pi(s)$ .
- Starting with  $w = w_0$ , we update  $w$  so it **best-fits** the bootstrapped estimate in terms of squared error on the states.

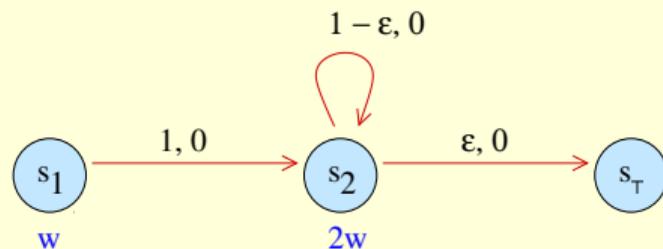
# A Counterexample (Tsitsiklis and Van Roy, 1996)



- From state  $s$ , let  $s'$ ,  $r$  be the (random) next state, reward.
- If our current estimate of  $V^\pi$  is  $\hat{V}$ , the bootstrapping idea suggests  $\mathbb{E}_\pi[r + \gamma \hat{V}(s')]$  as a “better estimate” of  $V^\pi(s)$ .
- Starting with  $w = w_0$ , we update  $w$  so it **best-fits** the bootstrapped estimate in terms of squared error on the states. For  $k \geq 0$ :

$$w_{k+1} \leftarrow \operatorname{argmin}_{w \in \mathbb{R}} \sum_s \left( \mathbb{E}_\pi[r + \gamma \hat{V}(w_k, x(s'))] - \hat{V}(w, x(s)) \right)^2.$$

# A Counterexample (Tsitsiklis and Van Roy, 1996)

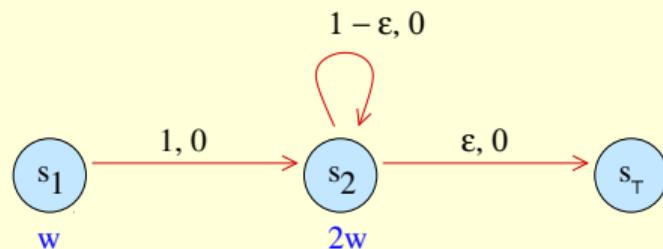


- From state  $s$ , let  $s'$ ,  $r$  be the (random) next state, reward.
- If our current estimate of  $V^\pi$  is  $\hat{V}$ , the bootstrapping idea suggests  $\mathbb{E}_\pi[r + \gamma \hat{V}(s')]$  as a “better estimate” of  $V^\pi(s)$ .
- Starting with  $w = w_0$ , we update  $w$  so it **best-fits** the bootstrapped estimate in terms of squared error on the states. For  $k \geq 0$ :

$$w_{k+1} \leftarrow \operatorname{argmin}_{w \in \mathbb{R}} \sum_s \left( \mathbb{E}_\pi[r + \gamma \hat{V}(w_k, x(s'))] - \hat{V}(w, x(s)) \right)^2.$$

- Is  $\lim_{k \rightarrow \infty} w_k = 0$ ?

# A Counterexample (Tsitsiklis and Van Roy, 1996)

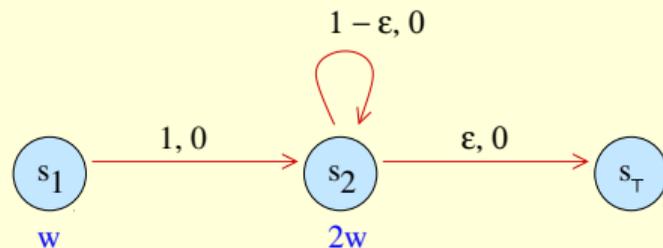


- From state  $s$ , let  $s'$ ,  $r$  be the (random) next state, reward.
- If our current estimate of  $V^\pi$  is  $\hat{V}$ , the bootstrapping idea suggests  $\mathbb{E}_\pi[r + \gamma \hat{V}(s')]$  as a “better estimate” of  $V^\pi(s)$ .
- Starting with  $w = w_0$ , we update  $w$  so it **best-fits** the bootstrapped estimate in terms of squared error on the states. For  $k \geq 0$ :

$$w_{k+1} \leftarrow \operatorname{argmin}_{w \in \mathbb{R}} \sum_s \left( \mathbb{E}_\pi[r + \gamma \hat{V}(w_k, x(s'))] - \hat{V}(w, x(s)) \right)^2.$$

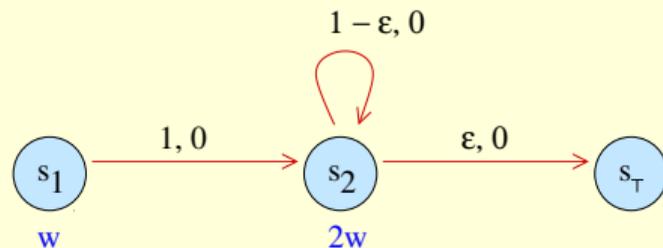
- Is  $\lim_{k \rightarrow \infty} w_k = 0$ ? Let's see.

# A Counterexample (Tsitsiklis and Van Roy, 1996)



$$\begin{aligned} w_{k+1} &= \operatorname{argmin}_{w \in \mathbb{R}} \sum_s \left( \mathbb{E}_{\pi} [r + \gamma \hat{V}(w_k, x(s'))] - \hat{V}(w, x(s)) \right)^2 \\ &= \operatorname{argmin}_{w \in \mathbb{R}} \left( (2\gamma w_k - w)^2 + (2\gamma(1 - \epsilon)w_k - 2w)^2 \right) = \gamma \frac{6 - 4\epsilon}{5} w_k. \end{aligned}$$

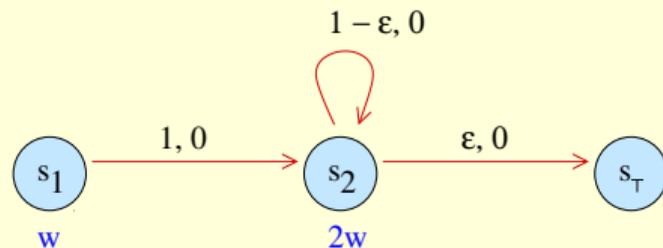
# A Counterexample (Tsitsiklis and Van Roy, 1996)



$$\begin{aligned} w_{k+1} &= \operatorname{argmin}_{w \in \mathbb{R}} \sum_s \left( \mathbb{E}_{\pi} [r + \gamma \hat{V}(w_k, x(s'))] - \hat{V}(w, x(s)) \right)^2 \\ &= \operatorname{argmin}_{w \in \mathbb{R}} \left( (2\gamma w_k - w)^2 + (2\gamma(1 - \epsilon)w_k - 2w)^2 \right) = \gamma \frac{6 - 4\epsilon}{5} w_k. \end{aligned}$$

- For  $w_0 = 1$ ,  $\epsilon = 0.1$ ,  $\gamma = 0.99$ ,  $\lim_{k \rightarrow \infty} w_k = \infty$ ; **divergence!**

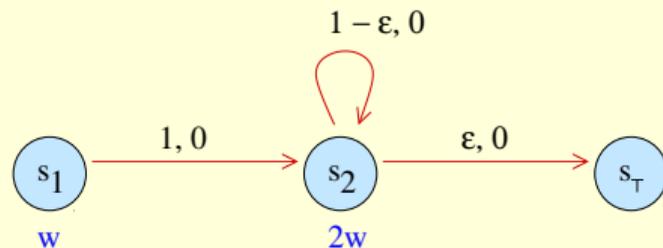
# A Counterexample (Tsitsiklis and Van Roy, 1996)



$$\begin{aligned} w_{k+1} &= \operatorname{argmin}_{w \in \mathbb{R}} \sum_s \left( \mathbb{E}_{\pi} [r + \gamma \hat{V}(w_k, x(s'))] - \hat{V}(w, x(s)) \right)^2 \\ &= \operatorname{argmin}_{w \in \mathbb{R}} \left( (2\gamma w_k - w)^2 + (2\gamma(1 - \epsilon)w_k - 2w)^2 \right) = \gamma \frac{6 - 4\epsilon}{5} w_k. \end{aligned}$$

- For  $w_0 = 1$ ,  $\epsilon = 0.1$ ,  $\gamma = 0.99$ ,  $\lim_{k \rightarrow \infty} w_k = \infty$ ; **divergence!**
- The failure owes to the combination of three factors: **off-policy updating**, **generalisation**, **bootstrapping**.

# A Counterexample (Tsitsiklis and Van Roy, 1996)



$$\begin{aligned} w_{k+1} &= \operatorname{argmin}_{w \in \mathbb{R}} \sum_s \left( \mathbb{E}_{\pi} [r + \gamma \hat{V}(w_k, x(s'))] - \hat{V}(w, x(s)) \right)^2 \\ &= \operatorname{argmin}_{w \in \mathbb{R}} \left( (2\gamma w_k - w)^2 + (2\gamma(1 - \epsilon)w_k - 2w)^2 \right) = \gamma \frac{6 - 4\epsilon}{5} w_k. \end{aligned}$$

- For  $w_0 = 1$ ,  $\epsilon = 0.1$ ,  $\gamma = 0.99$ ,  $\lim_{k \rightarrow \infty} w_k = \infty$ ; **divergence!**
- The failure owes to the combination of three factors: **off-policy updating**, **generalisation**, **bootstrapping**.
- But these are almost always **used together in practice!**

# Summary of Theoretical Results\*

Method	Tabular	Linear FA	Non-linear FA
TD(0)	C, O	C	NK
TD( $\lambda$ ), $\lambda \in (0, 1)$	C, O	C	NK
TD(1)	C, O	C, "Best"	C, Local optimum
Sarsa(0)	C, O	Chattering	NK
Sarsa( $\lambda$ ), $\lambda \in (0, 1)$	NK	Chattering	NK
Sarsa(1)	NK	NK	NK
Q-learning(0)	C, O	NK	NK

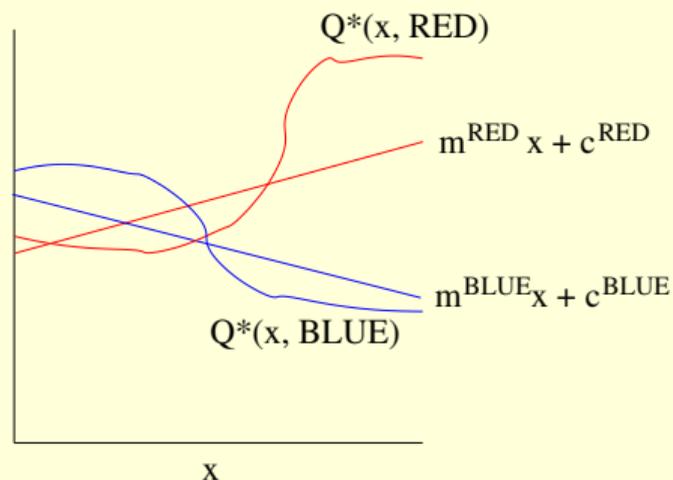
(C: Convergent; O: Optimal; NK: Not known.)

\*: to the best of your instructor's knowledge.

# Reinforcement Learning

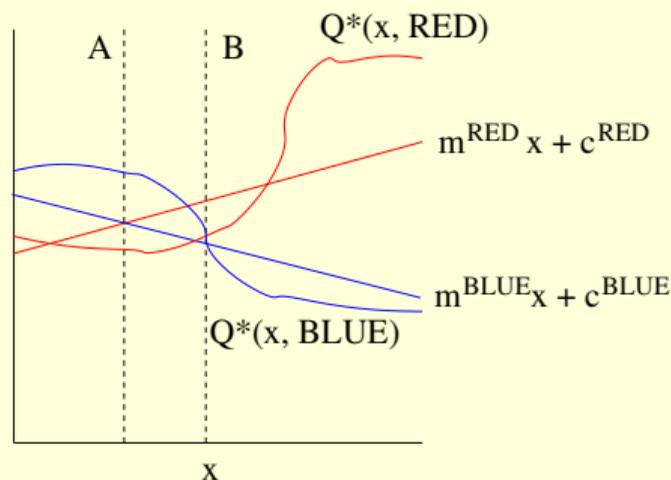
1. Tile coding
2. Issues in control with function approximation
3. The case for policy search

# So Near, Yet So Far



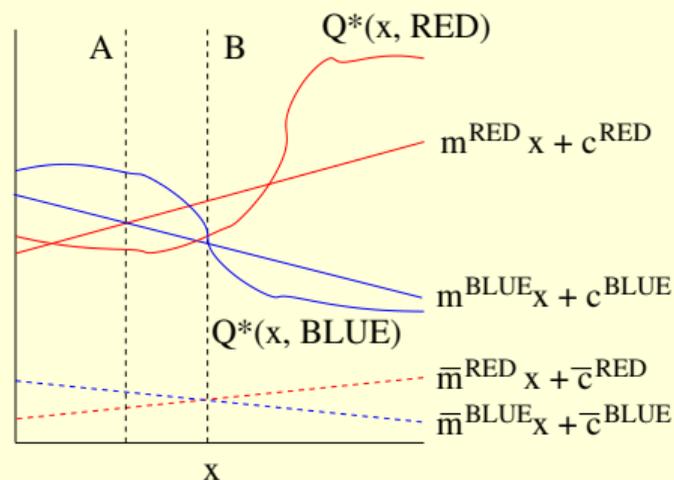
- $(m^{\text{RED}}, c^{\text{RED}}, m^{\text{BLUE}}, c^{\text{BLUE}})$  a “good” approximation of  $Q^*$ .

# So Near, Yet So Far



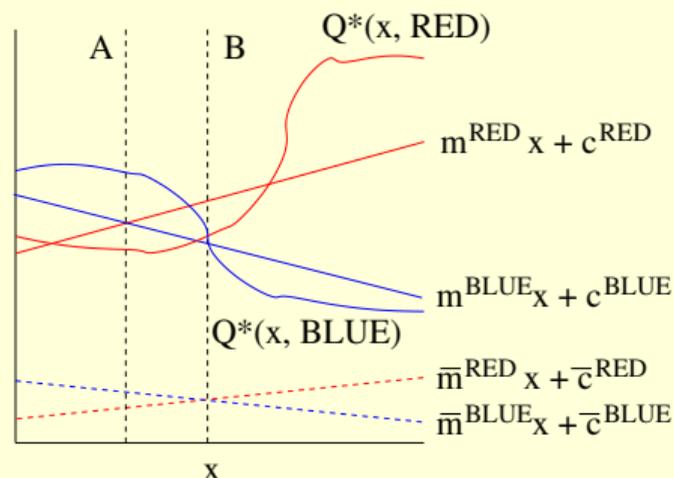
- $(m^{\text{RED}}, c^{\text{RED}}, m^{\text{BLUE}}, c^{\text{BLUE}})$  a “good” approximation of  $Q^*$ .  
But induces non-optimal actions for  $x \in (A, B)$ .

# So Near, Yet So Far



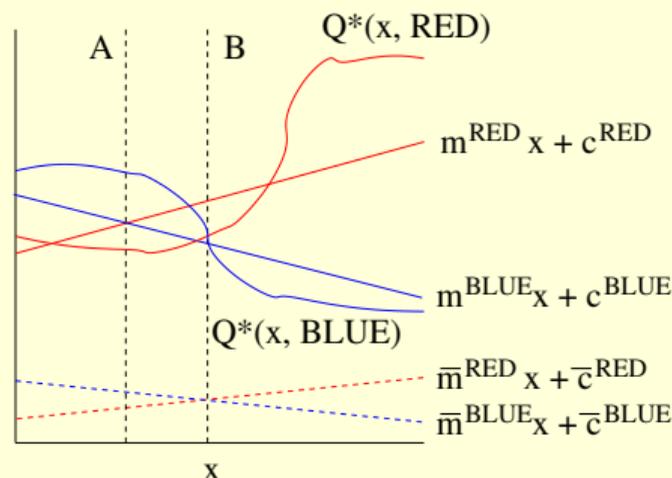
- $(m^{\text{RED}}, c^{\text{RED}}, m^{\text{BLUE}}, c^{\text{BLUE}})$  a “good” approximation of  $Q^*$ .  
But induces non-optimal actions for  $x \in (A, B)$ .
- $(\bar{m}^{\text{RED}}, \bar{c}^{\text{RED}}, \bar{m}^{\text{BLUE}}, \bar{c}^{\text{BLUE}})$  a “bad” approximation of  $Q^*$ .  
But induces optimal actions for all  $x$ !

# So Near, Yet So Far



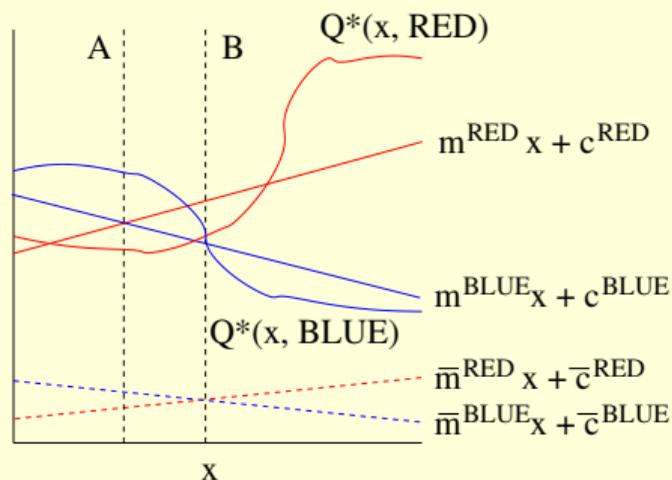
- $(m^{\text{RED}}, c^{\text{RED}}, m^{\text{BLUE}}, c^{\text{BLUE}})$  a “good” approximation of  $Q^*$ .  
But induces non-optimal actions for  $x \in (A, B)$ .
- $(\bar{m}^{\text{RED}}, \bar{c}^{\text{RED}}, \bar{m}^{\text{BLUE}}, \bar{c}^{\text{BLUE}})$  a “bad” approximation of  $Q^*$ .  
But induces optimal actions for all  $x$ !
- Perhaps we found  $(m^{\text{RED}}, c^{\text{RED}}, m^{\text{BLUE}}, c^{\text{BLUE}})$  by Q-learning.

# So Near, Yet So Far



- $(m^{\text{RED}}, c^{\text{RED}}, m^{\text{BLUE}}, c^{\text{BLUE}})$  a “good” approximation of  $Q^*$ .  
But induces non-optimal actions for  $x \in (A, B)$ .
- $(\bar{m}^{\text{RED}}, \bar{c}^{\text{RED}}, \bar{m}^{\text{BLUE}}, \bar{c}^{\text{BLUE}})$  a “bad” approximation of  $Q^*$ .  
But induces optimal actions for all  $x$ !
- Perhaps we found  $(m^{\text{RED}}, c^{\text{RED}}, m^{\text{BLUE}}, c^{\text{BLUE}})$  by Q-learning.
- How to find  $(\bar{m}^{\text{RED}}, \bar{c}^{\text{RED}}, \bar{m}^{\text{BLUE}}, \bar{c}^{\text{BLUE}})$ ?

# So Near, Yet So Far



- $(m^{\text{RED}}, c^{\text{RED}}, m^{\text{BLUE}}, c^{\text{BLUE}})$  a “good” approximation of  $Q^*$ .  
But induces non-optimal actions for  $x \in (A, B)$ .
- $(\bar{m}^{\text{RED}}, \bar{c}^{\text{RED}}, \bar{m}^{\text{BLUE}}, \bar{c}^{\text{BLUE}})$  a “bad” approximation of  $Q^*$ .  
But induces optimal actions for all  $x$ !
- Perhaps we found  $(m^{\text{RED}}, c^{\text{RED}}, m^{\text{BLUE}}, c^{\text{BLUE}})$  by Q-learning.
- How to find  $(\bar{m}^{\text{RED}}, \bar{c}^{\text{RED}}, \bar{m}^{\text{BLUE}}, \bar{c}^{\text{BLUE}})$ ? **Next class: policy search.**