

# Lecture 5: Value Function Approximation

Emma Brunskill

CS234 Reinforcement Learning.

Winter 2020

The value function approximation structure for today closely follows much of David Silver's Lecture 6.

# Refresh Your Knowledge 4

- The basic idea of TD methods are to make state-next state pairs fit the constraints of the Bellman equation on average (question by: Phil Thomas)
  - ① True
  - ② False
  - ③ Not sure
- In tabular MDPs, if using a decision policy that visits all states an infinite number of times, and in each state randomly selects an action, then (select all)
  - ① Q-learning will converge to the optimal Q-values
  - ② SARSA will converge to the optimal Q-values
  - ③ Q-learning is learning off-policy
  - ④ SARSA is learning off-policy
  - ⑤ Not sure
- A TD error  $> 0$  can occur even if the current  $V(s)$  is correct  $\forall s$ : [select all]
  - ① False
  - ② True if the MDP has stochastic state transitions
  - ③ True if the MDP has deterministic state transitions
  - ④ True if  $\alpha > 0$
  - ⑤ Not sure

# Table of Contents

- 1 Introduction
- 2 VFA for Prediction
- 3 Control using Value Function Approximation

# Class Structure

- Last time: Control (making decisions) without a model of how the world works
- **This time: Value function approximation**
- Next time: Deep reinforcement learning

# Last time: Model-Free Control

- Last time: how to learn a good policy from experience
- So far, have been assuming we can represent the value function or state-action value function as a vector/ matrix
  - Tabular representation
- Many real world problems have enormous state and/or action spaces
- Tabular representation is insufficient

# Today: Focus on Generalization

- Optimization
- Delayed consequences
- Exploration
- **Generalization**

# Table of Contents

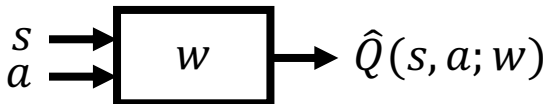
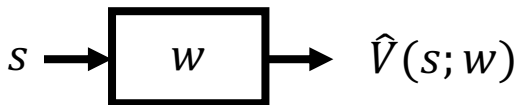
1 Introduction

2 VFA for Prediction

3 Control using Value Function Approximation

# Value Function Approximation (VFA)

- Represent a (state-action/state) value function with a parameterized function instead of a table





# Motivation for VFA

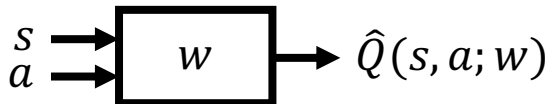
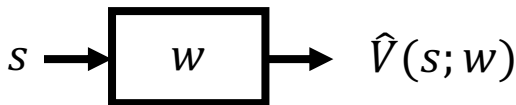
- Don't want to have to explicitly store or learn for every single state a
  - Dynamics or reward model
  - Value
  - State-action value
  - Policy
- Want more compact representation that generalizes across state or states and actions

# Benefits of Generalization

- Reduce memory needed to store  $(P, R)/V/Q/\pi$
- Reduce computation needed to compute  $(P, R)/V/Q/\pi$
- Reduce experience needed to find a good  $P, R/V/Q/\pi$

# Value Function Approximation (VFA)

- Represent a (state-action/state) value function with a parameterized function instead of a table



- Which function approximator?

# Function Approximators

- Many possible function approximators including
  - Linear combinations of features
  - Neural networks
  - Decision trees
  - Nearest neighbors
  - Fourier/ wavelet bases
- In this class we will focus on function approximators that are differentiable (Why?)
- Two very popular classes of differentiable function approximators
  - Linear feature representations (Today)
  - Neural networks (Next lecture)

# Review: Gradient Descent

- Consider a function  $J(\mathbf{w})$  that is a differentiable function of a parameter vector  $\mathbf{w}$
- Goal is to find parameter  $\mathbf{w}$  that minimizes  $J$
- The gradient of  $J(\mathbf{w})$  is

# Table of Contents

- 1 Introduction
- 2 VFA for Prediction
- 3 Control using Value Function Approximation

# Value Function Approximation for Policy Evaluation with an Oracle

- First assume we could query any state  $s$  and an oracle would return the true value for  $V^\pi(s)$
- The objective was to find the best approximate representation of  $V^\pi$  given a particular parameterized function

# Stochastic Gradient Descent

- Goal: Find the parameter vector  $\mathbf{w}$  that minimizes the loss between a true value function  $V^\pi(s)$  and its approximation  $\hat{V}(s; \mathbf{w})$  as represented with a particular function class parameterized by  $\mathbf{w}$ .
- Generally use mean squared error and define the loss as

$$J(\mathbf{w}) = \mathbb{E}_\pi[(V^\pi(s) - \hat{V}(s; \mathbf{w}))^2]$$

- Can use gradient descent to find a local minimum

$$\Delta \mathbf{w} = -\frac{1}{2}\alpha \nabla_{\mathbf{w}} J(\mathbf{w})$$

- Stochastic gradient descent (SGD) uses a finite number of (often one) samples to compute an approximate gradient:
- Expected SGD is the same as the full gradient update



# Model Free VFA Policy Evaluation

- Don't actually have access to an oracle to tell true  $V^\pi(s)$  for any state  $s$
- Now consider how to do model-free value function approximation for prediction / evaluation / policy evaluation without a model

# Model Free VFA Prediction / Policy Evaluation

- Recall model-free policy evaluation (Lecture 3)
  - Following a fixed policy  $\pi$  (or had access to prior data)
  - Goal is to estimate  $V^\pi$  and/or  $Q^\pi$
- Maintained a lookup table to store estimates  $V^\pi$  and/or  $Q^\pi$
- Updated these estimates after each episode (Monte Carlo methods) or after each step (TD methods)
- **Now: in value function approximation, change the estimate update step to include fitting the function approximator**

# Feature Vectors

- Use a feature vector to represent a state  $s$

$$\mathbf{x}(s) = \begin{pmatrix} x_1(s) \\ x_2(s) \\ \dots \\ x_n(s) \end{pmatrix}$$

# Linear Value Function Approximation for Prediction With An Oracle

- Represent a value function (or state-action value function) for a particular policy with a weighted linear combination of features

$$\hat{V}(s; \mathbf{w}) = \sum_{j=1}^n x_j(s) w_j = \mathbf{x}(s)^T \mathbf{w}$$

- Objective function is

$$J(\mathbf{w}) = \mathbb{E}_{\pi}[(V^{\pi}(s) - \hat{V}(s; \mathbf{w}))^2]$$

- Recall weight update is

$$\Delta \mathbf{w} = -\frac{1}{2} \alpha \nabla_{\mathbf{w}} J(\mathbf{w})$$

- Update is:

- Update = step-size  $\times$  prediction error  $\times$  feature value

# Monte Carlo Value Function Approximation

- Return  $G_t$  is an unbiased but noisy sample of the true expected return  $V^\pi(s_t)$
- Therefore can reduce MC VFA to doing supervised learning on a set of (state,return) pairs:  $\langle s_1, G_1 \rangle, \langle s_2, G_2 \rangle, \dots, \langle s_T, G_T \rangle$ 
  - Substitute  $G_t$  for the true  $V^\pi(s_t)$  when fit function approximator
- Concretely when using linear VFA for policy evaluation

$$\begin{aligned}\Delta \mathbf{w} &= \alpha(G_t - \hat{V}(s_t; \mathbf{w})) \nabla_{\mathbf{w}} \hat{V}(s_t; \mathbf{w}) \\ &= \alpha(G_t - \hat{V}(s_t; \mathbf{w})) \mathbf{x}(s_t) \\ &= \alpha(G_t - \mathbf{x}(s_t)^T \mathbf{w}) \mathbf{x}(s_t)\end{aligned}$$

- Note:  $G_t$  may be a very noisy estimate of true return

# MC Linear Value Function Approximation for Policy Evaluation

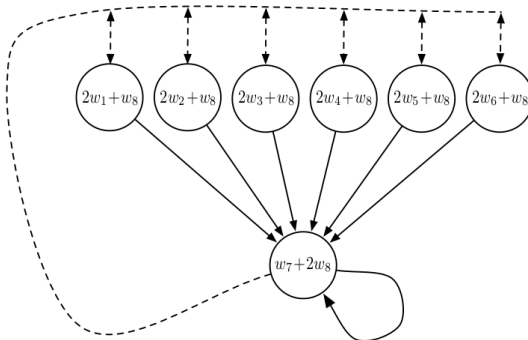
---

```
1: Initialize  $\mathbf{w} = \mathbf{0}$ ,  $k = 1$ 
2: loop
3:   Sample  $k$ -th episode  $(s_{k,1}, a_{k,1}, r_{k,1}, s_{k,2}, \dots, s_{k,L_k})$  given  $\pi$ 
4:   for  $t = 1, \dots, L_k$  do
5:     if First visit to  $(s)$  in episode  $k$  then
6:        $G_t(s) = \sum_{j=t}^{L_k} r_{k,j}$ 
7:       Update weights:

8:     end if
9:   end for
10:   $k = k + 1$ 
11: end loop
```

---

# Baird (1995)-Like Example with MC Policy Evaluation<sup>1</sup>



- MC update:  $\Delta \mathbf{w} = \alpha (G_t - \mathbf{x}(s_t)^T \mathbf{w}) \mathbf{x}(s_t)$
- Small prob  $s_7$  goes to terminal state,  $\mathbf{x}(s_7)^T = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 2]$

# Convergence Guarantees for Linear Value Function Approximation for Policy Evaluation: Preliminaries

- For infinite horizon, the Markov Chain defined by a MDP with a particular policy will eventually converge to a probability distribution over states  $d(s)$
- $d(s)$  is called the stationary distribution over states of  $\pi$
- $\sum_s d(s) = 1$
- $d(s)$  satisfies the following balance equation:

$$d(s') = \sum_s \sum_a \pi(a|s) p(s'|s, a) d(s)$$



# Convergence Guarantees for Linear Value Function Approximation for Policy Evaluation<sup>2</sup>

- Define the mean squared error of a linear value function approximation for a particular policy  $\pi$  relative to the true value as

$$MSVE(\mathbf{w}) = \sum_{s \in S} d(s)(V^\pi(s) - \hat{V}^\pi(s; \mathbf{w}))^2$$

- where
  - $d(s)$ : stationary distribution of  $\pi$  in the true decision process
  - $\hat{V}^\pi(s; \mathbf{w}) = \mathbf{x}(s)^T \mathbf{w}$ , a linear value function approximation

---

<sup>2</sup>Tsitsiklis and Van Roy. An Analysis of Temporal-Difference Learning with Function Approximation. 1997. <https://web.stanford.edu/~bvr/pubs/td.pdf>

# Convergence Guarantees for Linear Value Function Approximation for Policy Evaluation<sup>1</sup>

- Define the mean squared error of a linear value function approximation for a particular policy  $\pi$  relative to the true value as

$$MSVE(\mathbf{w}) = \sum_{s \in S} d(s)(V^\pi(s) - \hat{V}^\pi(s; \mathbf{w}))^2$$

- where
  - $d(s)$ : stationary distribution of  $\pi$  in the true decision process
  - $\hat{V}^\pi(s; \mathbf{w}) = \mathbf{x}(s)^T \mathbf{w}$ , a linear value function approximation
- Monte Carlo policy evaluation with VFA converges to the weights  $\mathbf{w}_{MC}$  which has the minimum mean squared error possible:

$$MSVE(\mathbf{w}_{MC}) = \min_{\mathbf{w}} \sum_{s \in S} d(s)(V^\pi(s) - \hat{V}^\pi(s; \mathbf{w}))^2$$

---

<sup>1</sup>Tsitsiklis and Van Roy. An Analysis of Temporal-Difference Learning with Function Approximation. 1997. <https://web.stanford.edu/~bvr/pubs/td.pdf>

# Recall: Temporal Difference Learning w/ Lookup Table

- Uses bootstrapping and sampling to approximate  $V^\pi$
- Updates  $V^\pi(s)$  after each transition  $(s, a, r, s')$ :

$$V^\pi(s) = V^\pi(s) + \alpha(r + \gamma V^\pi(s') - V^\pi(s))$$

- Target is  $r + \gamma V^\pi(s')$ , a biased estimate of the true value  $V^\pi(s)$
- Represent value for each state with a separate table entry

# Temporal Difference (TD(0)) Learning with Value Function Approximation

- Uses bootstrapping and sampling to approximate true  $V^\pi$
- Updates estimate  $V^\pi(s)$  after each transition  $(s, a, r, s')$ :

$$V^\pi(s) = V^\pi(s) + \alpha(r + \gamma V^\pi(s') - V^\pi(s))$$

- Target is  $r + \gamma V^\pi(s')$ , a biased estimate of the true value  $V^\pi(s)$
- In value function approximation, target is  $r + \gamma \hat{V}^\pi(s'; \mathbf{w})$ , a biased and approximated estimate of the true value  $V^\pi(s)$
- 3 forms of approximation:

# Temporal Difference (TD(0)) Learning with Value Function Approximation

- In value function approximation, target is  $r + \gamma \hat{V}^\pi(s'; \mathbf{w})$ , a biased and approximated estimate of the true value  $V^\pi(s)$
- Can reduce doing TD(0) learning with value function approximation to supervised learning on a set of data pairs:
  - $\langle s_1, r_1 + \gamma \hat{V}^\pi(s_2; \mathbf{w}) \rangle, \langle s_2, r_2 + \gamma \hat{V}^\pi(s_3; \mathbf{w}) \rangle, \dots$
- Find weights to minimize mean squared error

$$J(\mathbf{w}) = \mathbb{E}_\pi[(r_j + \gamma \hat{V}^\pi(s_{j+1}, \mathbf{w}) - \hat{V}(s_j; \mathbf{w}))^2]$$

# Temporal Difference (TD(0)) Learning with Value Function Approximation

- In value function approximation, target is  $r + \gamma \hat{V}^\pi(s'; \mathbf{w})$ , a biased and approximated estimate of the true value  $V^\pi(s)$
- Supervised learning on a different set of data pairs:  
 $\langle s_1, r_1 + \gamma \hat{V}^\pi(s_2; \mathbf{w}) \rangle, \langle s_2, r_2 + \gamma \hat{V}^\pi(s_3; \mathbf{w}) \rangle, \dots$
- In linear TD(0)

$$\begin{aligned}\Delta \mathbf{w} &= \alpha(r + \gamma \hat{V}^\pi(s'; \mathbf{w}) - \hat{V}^\pi(s; \mathbf{w})) \nabla_{\mathbf{w}} \hat{V}^\pi(s; \mathbf{w}) \\ &= \alpha(r + \gamma \hat{V}^\pi(s'; \mathbf{w}) - \hat{V}^\pi(s; \mathbf{w})) \mathbf{x}(s) \\ &= \alpha(r + \gamma \mathbf{x}(s')^T \mathbf{w} - \mathbf{x}(s)^T \mathbf{w}) \mathbf{x}(s)\end{aligned}$$

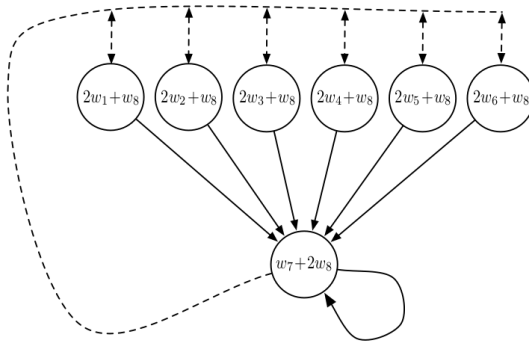
# TD(0) Linear Value Function Approximation for Policy Evaluation

- 
- 1: Initialize  $\mathbf{w} = \mathbf{0}$ ,  $k = 1$
  - 2: **loop**
  - 3:   Sample tuple  $(s_k, a_k, r_k, s_{k+1})$  given  $\pi$
  - 4:   Update weights:

$$\mathbf{w} = \mathbf{w} + \alpha(r + \gamma \mathbf{x}(s')^T \mathbf{w} - \mathbf{x}(s)^T \mathbf{w}) \mathbf{x}(s)$$

- 5:    $k = k + 1$
  - 6: **end loop**
-

# Baird Example with TD(0) On Policy Evaluation <sup>1</sup>



- TD update:  $\Delta \mathbf{w} = \alpha(r + \gamma \mathbf{x}(s')^T \mathbf{w} - \mathbf{x}(s)^T \mathbf{w}) \mathbf{x}(s)$

<sup>1</sup>Figure from Sutton and Barto 2018



# Convergence Guarantees for Linear Value Function Approximation for Policy Evaluation

- Define the mean squared error of a linear value function approximation for a particular policy  $\pi$  relative to the true value as

$$MSVE(\mathbf{w}) = \sum_{s \in S} d(s)(V^\pi(s) - \hat{V}^\pi(s; \mathbf{w}))^2$$

- where
  - $d(s)$ : stationary distribution of  $\pi$  in the true decision process
  - $\hat{V}^\pi(s; \mathbf{w}) = \mathbf{x}(s)^T \mathbf{w}$ , a linear value function approximation
- TD(0) policy evaluation with VFA converges to weights  $\mathbf{w}_{TD}$  which is within a constant factor of the minimum mean squared error possible:

$$MSVE(\mathbf{w}_{TD}) \leq \frac{1}{1 - \gamma} \min_{\mathbf{w}} \sum_{s \in S} d(s)(V^\pi(s) - \hat{V}^\pi(s; \mathbf{w}))^2$$

# Check Your Understanding: Poll

- Monte Carlo policy evaluation with VFA converges to the weights  $\mathbf{w}_{MC}$  which has the minimum mean squared error possible:

$$MSVE(\mathbf{w}_{MC}) = \min_{\mathbf{w}} \sum_{s \in S} d(s) (V^\pi(s) - \hat{V}^\pi(s; \mathbf{w}))^2$$

- TD(0) policy evaluation with VFA converges to weights  $\mathbf{w}_{TD}$  which is within a constant factor of the minimum mean squared error possible:

$$MSVE(\mathbf{w}_{TD}) \leq \frac{1}{1-\gamma} \min_{\mathbf{w}} \sum_{s \in S} d(s) (V^\pi(s) - \hat{V}^\pi(s; \mathbf{w}))^2$$

- If the VFA is a tabular representation (one feature for each state), what is the MSVE for MC and TD? [select all]

- 1 MSVE=0 for MC
- 2 MSVE > 0 for MC
- 3 MSVE = 0 for TD
- 4 MSVE > 0 for TD
- 5 Not sure

# Convergence Rates for Linear Value Function Approximation for Policy Evaluation

- Does TD or MC converge faster to a fixed point?
- Not (to my knowledge) definitively understood
- Practically TD learning often converges faster to its fixed value function approximation point

# Table of Contents

1 Introduction

2 VFA for Prediction

3 Control using Value Function Approximation

# Control using Value Function Approximation

- Use value function approximation to represent state-action values  
 $\hat{Q}^{\pi}(s, a; \mathbf{w}) \approx Q^{\pi}$
- Interleave
  - Approximate policy evaluation using value function approximation
  - Perform  $\epsilon$ -greedy policy improvement
- Can be unstable. Generally involves intersection of the following:
  - Function approximation
  - Bootstrapping
  - **Off-policy learning**

# Action-Value Function Approximation with an Oracle

- $\hat{Q}^\pi(s, a; \mathbf{w}) \approx Q^\pi$
- Minimize the mean-squared error between the true action-value function  $Q^\pi(s, a)$  and the approximate action-value function:

$$J(\mathbf{w}) = \mathbb{E}_\pi[(Q^\pi(s, a) - \hat{Q}^\pi(s, a; \mathbf{w}))^2]$$

- Use stochastic gradient descent to find a local minimum

$$\begin{aligned} -\frac{1}{2}\nabla_{\mathbf{w}}J(\mathbf{w}) &= \mathbb{E}\left[(Q^\pi(s, a) - \hat{Q}^\pi(s, a; \mathbf{w}))\nabla_{\mathbf{w}}\hat{Q}^\pi(s, a; \mathbf{w})\right] \\ \Delta(\mathbf{w}) &= -\frac{1}{2}\alpha\nabla_{\mathbf{w}}J(\mathbf{w}) \end{aligned}$$

- Stochastic gradient descent (SGD) samples the gradient

# Linear State Action Value Function Approximation with an Oracle

- Use features to represent both the state and action

$$\mathbf{x}(s, a) = \begin{pmatrix} x_1(s, a) \\ x_2(s, a) \\ \dots \\ x_n(s, a) \end{pmatrix}$$

- Represent state-action value function with a weighted linear combination of features

$$\hat{Q}(s, a; \mathbf{w}) = \mathbf{x}(s, a)^T \mathbf{w} = \sum_{j=1}^n x_j(s, a) w_j$$

- Stochastic gradient descent update:

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \nabla_{\mathbf{w}} \mathbb{E}_{\pi} [(Q^{\pi}(s, a) - \hat{Q}^{\pi}(s, a; \mathbf{w}))^2]$$

# Incremental Model-Free Control Approaches

- Similar to policy evaluation, true state-action value function for a state is unknown and so substitute a target value
- In Monte Carlo methods, use a return  $G_t$  as a substitute target

$$\Delta \mathbf{w} = \alpha(G_t - \hat{Q}(s_t, a_t; \mathbf{w})) \nabla_{\mathbf{w}} \hat{Q}(s_t, a_t; \mathbf{w})$$

- For SARSA instead use a TD target  $r + \gamma \hat{Q}(s', a'; \mathbf{w})$  which leverages the current function approximation value

$$\Delta \mathbf{w} = \alpha(r + \gamma \hat{Q}(s', a'; \mathbf{w}) - \hat{Q}(s, a; \mathbf{w})) \nabla_{\mathbf{w}} \hat{Q}(s, a; \mathbf{w})$$



# Incremental Model-Free Control Approaches

- Similar to policy evaluation, true state-action value function for a state is unknown and so substitute a target value
- In Monte Carlo methods, use a return  $G_t$  as a substitute target

$$\Delta \mathbf{w} = \alpha(G_t - \hat{Q}(s_t, a_t; \mathbf{w})) \nabla_{\mathbf{w}} \hat{Q}(s_t, a_t; \mathbf{w})$$

- For SARSA instead use a TD target  $r + \gamma \hat{Q}(s', a'; \mathbf{w})$  which leverages the current function approximation value

$$\Delta \mathbf{w} = \alpha(r + \gamma \hat{Q}(s', a'; \mathbf{w}) - \hat{Q}(s, a; \mathbf{w})) \nabla_{\mathbf{w}} \hat{Q}(s, a; \mathbf{w})$$

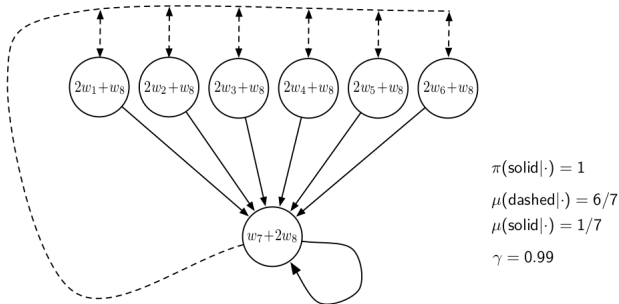
- For Q-learning instead use a TD target  $r + \gamma \max_{a'} \hat{Q}(s', a'; \mathbf{w})$  which leverages the max of the current function approximation value

$$\Delta \mathbf{w} = \alpha(r + \gamma \max_{a'} \hat{Q}(s', a'; \mathbf{w}) - \hat{Q}(s, a; \mathbf{w})) \nabla_{\mathbf{w}} \hat{Q}(s, a; \mathbf{w})$$

# Convergence of TD Methods with VFA

- Informally, updates involve doing an (approximate) Bellman backup followed by best trying to fit underlying value function to a particular feature representation
- Bellman operators are contractions, but value function approximation fitting can be an expansion

# Challenges of Off Policy Control: Baird Example <sup>1</sup>



- Behavior policy and target policy are not identical
- Value can diverge

# Convergence of Control Methods with VFA

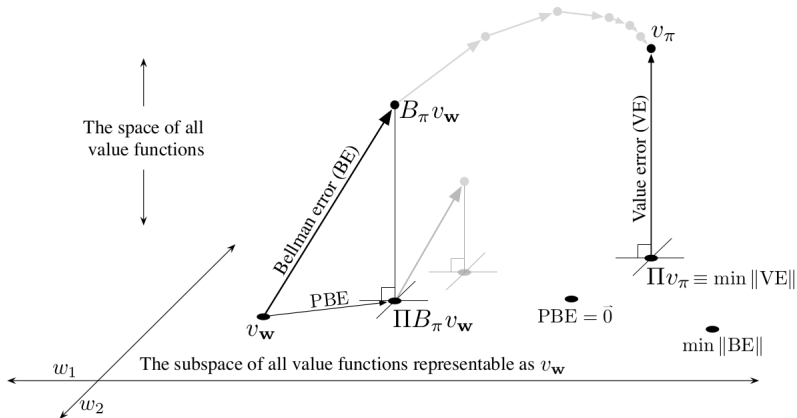
Algorithm	Tabular	Linear VFA	Nonlinear VFA
Monte-Carlo Control			
Sarsa			
Q-learning			

# Hot Topic: Off Policy Function Approximation

## Convergence

- Extensive work in better TD-style algorithms with value function approximation, some with convergence guarantees: see Chp 11 SB
- Exciting recent work on batch RL that can converge with nonlinear VFA (Dai et al. ICML 2018): uses primal dual optimization
- An important issue is not just whether the algorithm converges, but **what** solution it converges too
- Critical choices: **objective function and feature representation**

# Linear Value Function Approximation<sup>3</sup>



<sup>3</sup>Figure from Sutton and Barto 2018

# What You Should Understand

- Be able to implement TD(0) and MC on policy evaluation with linear value function approximation
- Be able to define what TD(0) and MC on policy evaluation with linear VFA are converging to and when this solution has 0 error and non-zero error.
- Be able to implement Q-learning and SARSA and MC control algorithms
- List the 3 issues that can cause instability and describe the problems qualitatively: function approximation, bootstrapping and off policy learning

# Class Structure

- Last time: Control (making decisions) without a model of how the world works
- This time: Value function approximation
- **Next time:** Deep reinforcement learning



# Batch Monte Carlo Value Function Approximation

- May have a set of episodes from a policy  $\pi$
- Can analytically solve for the best linear approximation that minimizes mean squared error on this data set
- Let  $G(s_i)$  be an unbiased sample of the true expected return  $V^\pi(s_i)$

$$\arg \min_{\mathbf{w}} \sum_{i=1}^N (G(s_i) - \mathbf{x}(s_i)^T \mathbf{w})^2$$

- Take the derivative and set to 0

$$\mathbf{w} = (X^T X)^{-1} X^T \mathbf{G}$$

- where  $\mathbf{G}$  is a vector of all  $N$  returns, and  $X$  is a matrix of the features of each of the  $N$  states  $\mathbf{x}(s_i)$
- Note: not making any Markov assumptions