Lecture 5: Value Function Approximation

Emma Brunskill

CS234 Reinforcement Learning.

Winter 2022

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

L5 Refresh Your Knowledge

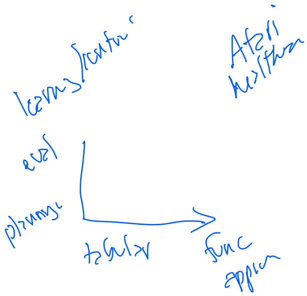
- The basic idea of TD methods are to make state-next state pairs fit the constraints of the Bellman equation on average (question by: Phil Thomas)
  1. True
  2. False
  3. Not sure
- In tabular MDPs, if using a decision policy that visits all states an infinite number of times, and in each state randomly 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. Not sure

L5 Refresh Your Knowledge

- 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)
  True
- In tabular MDPs, if using a decision poilcy 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 (True)
  2. SARSA will converge to the optimal Q-values (False)
  3. Q-learning is learning off-policy (True)
  4. SARSA is learning off-policy (False)
- 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 (True)
  3. True if the MDP has deterministic state transitions (False)
  4. Not sure

Break

-


learning/control
Atari
helicopter
eval
planning
tabular
func
approx

Maximization Bias[1] Proof

- Consider single-state MDP ($|S| = 1$) with 2 actions, and both actions have 0-mean random rewards, ($\mathbb{E}(r|a = a_1) = \mathbb{E}(r|a = a_2) = 0$).
- Then $Q(s, a_1) = Q(s, a_2) = 0 = V(s)$
- Assume there are prior samples of taking action a_1 and a_2
- Let $\hat{Q}(s, a_1), \hat{Q}(s, a_2)$ be the finite sample estimate of Q
- Use an unbiased estimator for Q: e.g. $\hat{Q}(s, a_1) = \frac{1}{n(s,a_1)} \sum_{i=1}^{n(s,a_1)} r_i(s, a_1)$
- Let $\hat{\pi} = \arg\max_a \hat{Q}(s, a)$ be the greedy policy w.r.t. the estimated $\hat{Q}$
- *Even though each estimate of the state-action values is unbiased*, the estimate of $\hat{\pi}$'s value $\hat{V}^{\hat{\pi}}$ can be biased:
  $\hat{V}^{\hat{\pi}}(s) = \mathbb{E}[\max \hat{Q}(s, a_1), \hat{Q}(s, a_2)]$
  $\geq \max[\mathbb{E}[\hat{Q}(s, a_1)], [\hat{Q}(s, a_2)]]$
  $= max[0, 0] = V^{\pi}$,
  where the inequality comes from Jensen's inequality.

[1] Example from Mannor, Simester, Sun and Tsitsiklis. Bias and Variance Approximation in Value Function Estimates. Management Science 2007

Double Q-Learning

- The greedy policy w.r.t. estimated Q values can yield a maximization bias during finite-sample learning
- Avoid using max of estimates as estimate of max of true values
- Instead split samples and use to create two independent unbiased estimates of $Q_1(s_1, a_i)$ and $Q_2(s_1, a_i)$ $\forall a$.
    - Use one estimate to select max action: $a^* = \arg\max_a Q_1(s_1, a)$
    - Use other estimate to estimate value of a^*: $Q_2(s, a^*)$
    - Yields unbiased estimate: $\mathbb{E}(Q_2(s, a^*)) = Q(s, a^*)$

Double Q-Learning

- The greedy policy w.r.t. estimated Q values can yield a maximization bias during finite-sample learning
- Avoid using max of estimates as estimate of max of true values
- Instead split samples and use to create two independent unbiased estimates of $Q_1(s_1, a_i)$ and $Q_2(s_1, a_i)$ $\forall a$.
  - Use one estimate to select max action: $a^* = \arg\max_a Q_1(s_1, a)$
  - Use other estimate to estimate value of a^*: $Q_2(s, a^*)$
  - Yields unbiased estimate: $\mathbb{E}(Q_2(s, a^*)) = Q(s, a^*)$
- Why does this yield an unbiased estimate of the max state-action value?
  Using independent samples to estimate the value
- If acting online, can alternate samples used to update Q_1 and Q_2, using the other to select the action chosen
- Next slides extend to full MDP case (with more than 1 state)

Double Q-Learning

```
1: Initialize $Q_1(s,a)$ and $Q_2(s,a), \forall s \in S, a \in A$ $t = 0$, initial state $s_t = s_0$
2: loop
3:     Select $a_t$ using $\epsilon$-greedy $\pi(s) = \arg\max_a Q_1(s_t,a) + Q_2(s_t,a)$
4:     Observe $(r_t, s_{t+1})$
5:     if (with 0.5 probability) then
6:         $Q_1(s_t,a_t) \leftarrow Q_1(s_t,a_t) + \alpha(\underline{r_t + \gamma \boxed{Q_2}}(s_{t+1}, \underline{\arg\max_a Q_1(s_{t+1},a)}) - Q_1(s_t,a_t))$
7:     else
8:         $\rightarrow Q_2(s_t,a_t) \leftarrow Q_2(s_t,a_t) + \alpha(r_t + \gamma Q_1(s_{t+1}, \arg\max_a Q_2(s_{t+1},a)) - Q_2(s_t,a_t))$
9:     end if
10:    $t = t + 1$
11: end loop
```

2x same unclear

Compared to Q-learning, how does this change the: memory requirements, computation requirements per step, amount of data required?

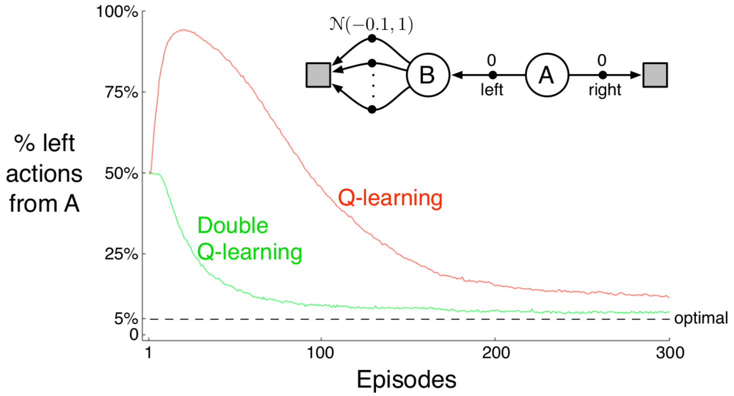
Double Q-Learning

```
1: Initialize $Q_1(s,a)$ and $Q_2(s,a)$,$\forall s \in S, a \in A$ $t = 0$, initial state $s_t = s_0$
2: loop
3:     Select $a_t$ using $\epsilon$-greedy $\pi(s) = \arg\max_a Q_1(s_t,a) + Q_2(s_t,a)$
4:     Observe $(r_t, s_{t+1})$
5:     if (with 0.5 probability) then
6:         $Q_1(s_t,a_t) \leftarrow Q_1(s_t,a_t) + \alpha(r_t + \gamma Q_2(s_{t+1}, \arg\max_a Q_1(s_{t+1},a)) - Q_1(s_t,a_t))$
7:     else
8:         $Q_2(s_t,a_t) \leftarrow Q_2(s_t,a_t) + \alpha(r_t + \gamma Q_1(s_{t+1}, \arg\max_a Q_2(s_{t+1},a)) - Q_2(s_t,a_t))$
9:     end if
10:    $t = t + 1$
11: end loop
```

Compared to Q-learning, how does this change the: memory requirements, computation requirements per step, amount of data required?

Doubles the memory, same computation requirements, data requirements are subtle– might reduce amount of exploration

Double Q-Learning (Figure 6.7 in Sutton and Barto 2018)



Due to the maximization bias, in this case Q-learning spends much more time selecting suboptimal actions than double Q-learning.

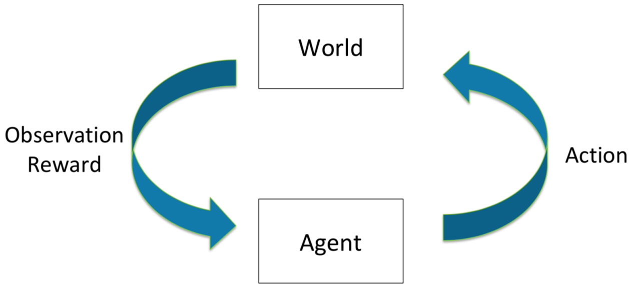
Class Structure

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

Outline for Today

- Value function approximation
- Monte Carlo policy evaluation with linear function approximation
- TD policy evaluation with linear function approximation
- Control methods with linear value function approximation

Reinforcement Learning



- Goal: Learn to select actions to maximize total expected future reward

Last Time: Tabular Representations for 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

Motivation for Function Approximation

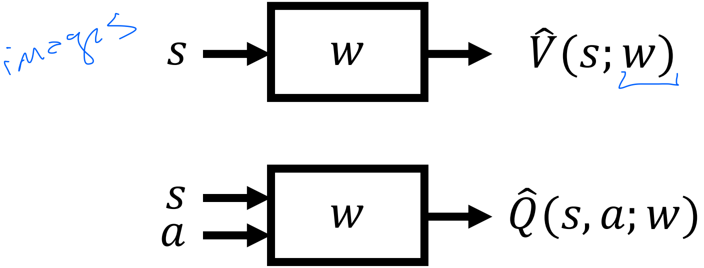
- Don't want to have to explicitly store or learn for every single state a
  - Dynamics or reward model
  - Value
  - State-action value
  - Policy
- Want more compact representation that generalizes across state or states and actions

Benefits of Function Approximation

- 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)

Outline for the Rest of Today: Policy Evaluation to Control

- Given known dynamics and reward models, and a tabular representation
  - Discussed how to do policy evaluation and then control (value iteration and policy iteration)
- Given no models, and a tabular representation
  - Discussed how to do policy evaluation (MC/TD) and then control (MC, SARSA, Q-learning)
- **Given no models, and function approximation**
  - **Today will discuss how to do policy evaluation and then control**

Review: Gradient Descent

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

$$\nabla J(w) = \left[\frac{\partial J}{\partial w_1} \quad \frac{\partial J}{\partial w_2} \quad \cdots \quad \frac{\partial J}{\partial w_n} \right]$$

$$w = w - \Delta w$$

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 $V^{\pi}(s)$
- Similar to supervised learning: assume given $(s, V^{\pi}(s))$ pairs
- The objective is to find the best approximate representation of V^{π} given a particular parameterized function $\underline{\hat{V}(s; w)}$

Stochastic Gradient Descent

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

$$J(\boldsymbol{w}) = \mathbb{E}_{\overset{s\sim\pi}{\boxed{\pi}}}[(\underbrace{V^\pi}_{\text{true}}(s) - \hat{V}(s;\boldsymbol{w}))^2]$$

- Can use gradient descent to find a local minimum $(V^\pi - \hat{V})\nabla_w V(s,w)$

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

- Stochastic gradient descent (SGD) uses a finite number of (often one) samples to compute an approximate gradient:

$$\nabla_w J(w) = \nabla_w E_\pi [V^\pi(s) - \hat{V}(s,w)]^2$$
$$= E_\pi\, 2(V^\pi(s) - \hat{V}(s,w)] \underline{\nabla_w V(s,w)}$$

- Expected SGD is the same as the full gradient update

Stochastic Gradient Descent

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

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

- Can use gradient descent to find a local minimum

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

- Stochastic gradient descent (SGD) uses a finite number of (often one) samples to compute an approximate gradient:

$$\begin{aligned}\Delta_w J(w) &= \Delta_w E_{\pi}[V^{\pi}(s) - \hat{V}(s;w)]^2 \\ &= E_{\pi}[2(V^{\pi}(s) - \hat{V}(s;w)]\Delta_w\hat{V}(s,w)\end{aligned}$$

- 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

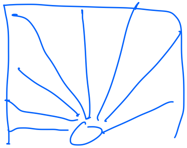
- Recall model-free policy evaluation (Lecture 3)
    - Following a fixed policy π (or had access to prior data)
    - Goal is to estimate V^π and/or Q^π
- Maintained a lookup table to store estimates V^π and/or Q^π
- Updated these estimates after each episode (Monte Carlo methods) or after each step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

Break

Outline for Today

- Value function approximation
- **Monte Carlo policy evaluation with linear function approximation**
- TD policy evaluation with linear function approximation
- Control methods with linear value function approximation

Feature Vectors



- Use a feature vector to represent a state s

$$\mathbf{x}(s) = \begin{pmatrix} x_1(s) \\ x_2(s) \\ \dots \\ x_n(s) \end{pmatrix} \begin{matrix} 9 \\ 8 \\ - \\ - \\ - \\ - \end{matrix}$$

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

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

$$\nabla_w X(s)^T w = X(s)$$

- Objective function is

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

- Recall weight update is

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

prediction error

feature value

- Update is: $\Delta w = \frac{-1}{2}\alpha \left(\underbrace{V^\pi(s)} - X(s)^T w \right) X(s)$

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: $\langle s_1, G_1 \rangle, \langle s_2, G_2 \rangle, \ldots, \langle s_T, G_T \rangle$
  - Substitute G_t for the true $V^\pi(s_t)$ when fit function approximator

$$\Delta w = \alpha \left(V^\pi(s) - \hat{V}(s,w) \right) x(s_t)$$
$$\approx \alpha \left(G_t(s) - x(s_t) w \right) x(s_t)$$

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: $\langle s_1, G_1\rangle, \langle s_2, G_2\rangle, \ldots, \langle s_T, G_T\rangle$
  - Substitute G_t for the true $V^\pi(s_t)$ when fit function approximator
- Concretely when using linear VFA for policy evaluation

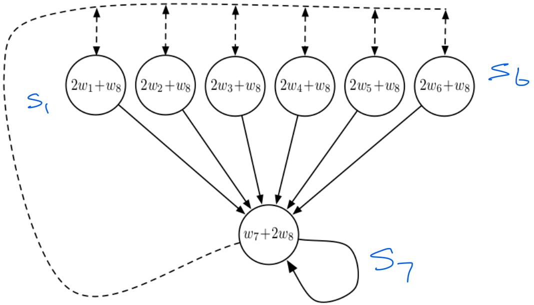
$$\begin{aligned}\Delta \boldsymbol{w} &= \alpha(G_t - \hat{V}(s_t; \boldsymbol{w}))\nabla_{\boldsymbol{w}} \hat{V}(s_t; \boldsymbol{w}) \\ &= \alpha(G_t - \hat{V}(s_t; \boldsymbol{w}))\boldsymbol{x}(s_t) \\ &= \alpha(G_t - \boldsymbol{x}(s_t)^T \boldsymbol{w})\boldsymbol{x}(s_t)\end{aligned}$$

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

MC Linear Value Function Approximation for Policy Evaluation

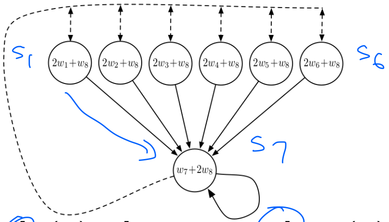
```
1: Initialize w = 0, $k = 1$
2: loop
3:     Sample $k$-th episode $(s_{k,1}, a_{k,1}, r_{k,1}, s_{k,2}, \dots, s_{k,L_k})$ given $\pi$
4:     for $t = 1, \dots, L_k$ do
5:         if First visit to $(s)$ in episode $k$ then
6:             $G_t(s) = \sum_{j=t}^{L_k} r_{k,j}$   now could weight by $\gamma$
7:             Update weights:
                 $w = \alpha (G_t(s) - x(s)^T w) x(s)$
8:         end if
9:     end for
10:    $k = k + 1$
11: end loop
```

Baird (1995)-Like Example with MC Policy Evaluation[2]



- $x(s_1) = [2\ 0\ 0\ 0\ 0\ 0\ 0\ 1]$ $x(s_2) = [0\ 2\ 0\ 0\ 0\ 0\ 0\ 1]$ $\ldots x(s_6) = [0\ 0\ 0\ 0\ 0\ 2\ 0\ 1]$ $x(s_7) = [0\ 0\ 0\ 0\ 0\ 0\ 1\ 2]$ $r(s) = 0\ \forall s$ 2 actions a_1 solid line, a_2 dotted
- Small prob s_7 goes to terminal state s_T

Baird (1995)-Like Example with MC Policy Evaluation[3]



- $\mathbf{x}(s_1) = [2\ 0\ 0\ 0\ 0\ 0\ 0\ \boxed{1}]$ $\mathbf{x}(s_2) = [0\ 2\ 0\ 0\ 0\ 0\ 0\ \boxed{1}] \ldots \mathbf{x}(s_6) = [0\ 0\ 0\ 0\ 0\ 2\ 0\ \boxed{1}]$ $\mathbf{x}(s_7) = [0\ 0\ 0\ 0\ 0\ 0\ 1\ \boxed{2}]$ $r(s) = 0\ \forall s$ 2 actions a_1 solid line, a_2 dotted
- Small prob s_7 goes to terminal state s_T
- Consider trajectory $(s_1, a_1, 0, \underline{s_7}, a_1, 0, \underline{s_7}, a_1, 0, \underline{s_T})$. $G(s_1) = 0$
- Let $w_0 = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]$. MC update: $\Delta \mathbf{w} = \alpha\underbrace{(G_t - \mathbf{x}(s_t)^T \mathbf{w})\mathbf{x}(s_t)}$

$x(s_1)^T w = 3$

$w = w + \alpha\,(0 - 3)\,[2\ 0\ 0\ 0\ 0\ 0\ 0\ 1]$

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 π relative to the true value as

$$MSVE_{\mu}(\boldsymbol{w}) = \sum_{s \in S} \mu(s)(V^{\pi}(s) - \hat{V}^{\pi}(s; \boldsymbol{w}))^2$$

- where
  - $\mu(s)$: probability of visiting state s under policy π . Note $\sum_s \mu(s) = 1$
  - $\hat{V}^{\pi}(s; \boldsymbol{w}) = \boldsymbol{x}(s)^T \boldsymbol{w}$, a linear value function approximation

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 π relative to the true value as

$$MSVE_{\mu}(\boldsymbol{w}) = \sum_{s \in S} \mu(s)(V^{\pi}(s) - \hat{V}^{\pi}(s; \boldsymbol{w}))^2$$

- where
  - $\mu(s)$: probability of visiting state s under policy π . Note $\sum_s \mu(s) = 1$
  - $\hat{V}^{\pi}(s; \boldsymbol{w}) = \boldsymbol{x}(s)^T \boldsymbol{w}$, a linear value function approximation
- Monte Carlo policy evaluation with VFA converges to the weights $\boldsymbol{w}_{MC}$ which has the minimum mean squared error possible with respect to the distribution μ:

$$MSVE_{\mu}(\boldsymbol{w}_{MC}) = \min_{\boldsymbol{w}} \sum_{s \in S} \mu(s)(V^{\pi}(s) - \hat{V}^{\pi}(s; \boldsymbol{w}))^2$$

Break

Today: Focus on Generalization using Linear Value Function

- Preliminaries
- Monte Carlo policy evaluation with linear function approximation
- **TD policy evaluation with linear function approximation**
- Control methods with linear value function approximation

Recall: Temporal Difference Learning w/ Lookup Table

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

$$V^\pi(s) \;=\; V^\pi(s) + \alpha(r + \gamma V^\pi(\underline{s'}) - V^\pi(s))$$

- 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^π
- Updates estimate $V^\pi(s)$ after each transition (s, a, r, s'):

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

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

  - sampling (vs expectation over s')
  - bootstrapping
  - VFA

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

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

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

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

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

- In value function approximation, target is $r + \gamma \hat{V}^\pi(s'; \boldsymbol{w})$, 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: TD targets
  - $\langle s_1, \underbrace{r_1 + \gamma \hat{V}^\pi(s_2; \boldsymbol{w})}\rangle, \langle s_2, r_2 + \gamma \underbrace{\hat{V}}(s_3; \boldsymbol{w})\rangle, \ldots$
- Find weights to minimize mean squared error

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

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

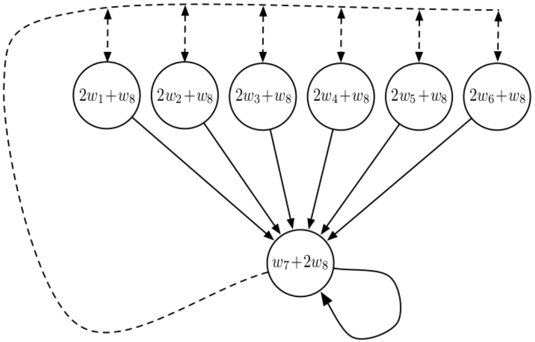
- In value function approximation, target is $r + \gamma \hat{V}^{\pi}(s'; \boldsymbol{w})$, a biased and approximated estimate of the true value $V^{\pi}(s)$
- Supervised learning on a different set of data pairs: $\langle s_1, r_1 + \gamma \hat{V}^{\pi}(s_2; \boldsymbol{w})\rangle, \langle s_2, r_2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

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

TD(0) Linear Value Function Approximation for Policy Evaluation

```
1: Initialize w = 0, $k = 1$
2: loop
3:     Sample tuple $(s_k, a_k, r_k, s_{k+1})$ given $\pi$
4:     Update weights:
           $\boldsymbol{w} = \boldsymbol{w} + \alpha(r + \gamma \overbrace{\boldsymbol{x}(s')^T}^{V(s')} \boldsymbol{w} - \boldsymbol{x}(s)^T \boldsymbol{w})\boldsymbol{x}(s)$
5:     $k = k + 1$
6: end loop
```

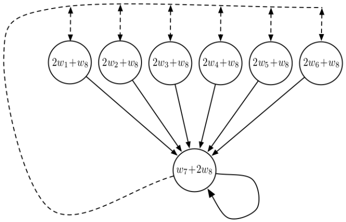
Baird Example with TD(0) On Policy Evaluation [1]



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

[1]Figure from Sutton and Barto 2018

Baird Example with TD(0) On Policy Evaluation [1]



- $\mathbf{x}(s_1) = [2\ 0\ 0\ 0\ 0\ 0\ 0\ 1]$ $\mathbf{x}(s_2) = [0\ 2\ 0\ 0\ 0\ 0\ 0\ 1]$ $\ldots \mathbf{x}(s_6) = [0\ 0\ 0\ 0\ 0\ 2\ 0\ 1]$ $\mathbf{x}(s_7) = [\underbrace{0\ 0\ 0\ 0\ 0\ 0\ 1\ 2}]$ $\quad r(s) = 0\ \forall s$ $\quad$ 2 actions a_1 solid line, a_2 dotted
- Small prob s_7 goes to terminal state s_T
- Consider tuple $(\boxed{s_1}, a_1, 0, s_7)$.
- Let $w_0 = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]$. TD update: $\Delta \mathbf{w} = \alpha(r + \gamma \underbrace{\mathbf{x}(s')^T \mathbf{w}}_{} - \underbrace{\mathbf{x}(s)^T \mathbf{w}}_{})\mathbf{x}(s)$ $\overset{0}{}$

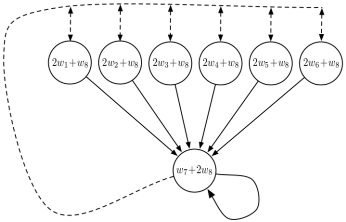
$V^\pi(s_1) = x(s_1)w = 3 \qquad V^\pi(s_7) = 3$

$\Delta w = \alpha(\gamma \cdot 3 - 3)\,[2\ 0\ 0\ 0\ 0\ 0\ 0\ 1]$

vs MC $\qquad (0 - 3)$

[1] Figure from Sutton and Barto 2018

Baird Example with TD(0) On Policy Evaluation [1]



- $\boldsymbol{x}(s_1) = [2\ 0\ 0\ 0\ 0\ 0\ 0\ 1]$ $\boldsymbol{x}(s_2) = [0\ 2\ 0\ 0\ 0\ 0\ 0\ 1]$ $\ldots \boldsymbol{x}(s_6) = [0\ 0\ 0\ 0\ 0\ 2\ 0\ 1]$ $\boldsymbol{x}(s_7) = [0\ 0\ 0\ 0\ 0\ 0\ 1\ 2]$ $\quad r(s) = 0\ \forall s$ $\quad$ 2 actions a_1 solid line, a_2 dotted
- Small prob s_7 goes to terminal state s_T
- Consider tuple $(s_1, a_1, 0, s_7)$.
- Let $w_0 = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]$. TD update: $\Delta\boldsymbol{w} = \alpha(r + \gamma\boldsymbol{x}(s')^T\boldsymbol{w} - \boldsymbol{x}(s)^T\boldsymbol{w})\boldsymbol{x}(s)$
- TD target is $r + \gamma\boldsymbol{x}(s')^T\boldsymbol{w}$. $r = 0$ $\quad \boldsymbol{x}(s')^T\boldsymbol{w} = 3$.
- $\boldsymbol{x}(s)^T\boldsymbol{w} = 3$
- $\Delta\boldsymbol{w} = \alpha(3\gamma - 3)\boldsymbol{x}(s_1)$

[1] Figure from Sutton and Barto 2018

Convergence Guarantees for TD Linear VFA 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 π
- $\sum_s d(s) = 1$
- $d(s)$ satisfies the following balance equation:

Markov systems

$$d(s') = \sum_s \sum_a \underbrace{\pi(a|s)}_{} \underbrace{p(s'|s,a)}_{} \underbrace{d(s)}_{}$$

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 π relative to the true value given the distribution d as

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

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

$\gamma = .9$

$$MSVE_d(\boldsymbol{w}_{TD}) \le \boxed{\frac{1}{1-\gamma}} \min_{\boldsymbol{w}} \sum_{s \in S} d(s)(V^\pi(s) - \hat{V}^\pi(s; \boldsymbol{w}))^2$$

Check Your Understanding L5N1: Poll

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

$$MSVE_d(\boldsymbol{w}_{TD}) \leq \frac{1}{1-\gamma} \min_{\boldsymbol{w}} \sum_{s \in S} d(s)(\overbrace{V^\pi(s) - \hat{V}^\pi(s;\boldsymbol{w})}^{=0})^2$$

- If the VFA is a tabular representation (one feature for each state), what is the $MSVE_d$ for TD?

1. Depends on the problem
2. MSVE $= 0$ for TD   0
3. Not sure

Check Your Understanding L5N1 : Poll

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

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

- If the VFA is a tabular representation (one feature for each state), what is the $MSVE_d$ for TD?

MSVE $=$ 0 for TD

Break

Outline for Today

- Value function approximation
- Monte Carlo policy evaluation with linear function approximation
- TD policy evaluation with linear function approximation
- Control methods with linear value function approximation

Table of Contents

Control using Value Function Approximation

- Use value function approximation to represent state-action values $\hat{Q}^\pi(s, a; \boldsymbol{w}) \approx Q^\pi$
- Interleave
    - Approximate policy evaluation using value function approximation
    - Perform ϵ-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

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

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

- Use stochastic gradient descent to find a local minimum

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

- Stochastic gradient descent (SGD) samples the gradient

Check Your Understanding L5N2: Predict Control Updates

- The weight update for control for MC and TD-style methods will be near identical to the policy evaluation steps. Try to see if you can predict which are the right weight update equations for the different methods (select all that are true)
- (1) is the SARSA control update
- (2) is the MC control update
- (3) is the Q-learning control update
- (4) is the MC control update
- (5) is the Q-learning control update

(s a r s' a')

$$
\begin{aligned}
\Delta \boldsymbol{w} &= \alpha(r + \gamma \hat{Q}(\underline{s'}, \underline{a'}; \boldsymbol{w}) - \hat{Q}(s, a; \boldsymbol{w}))\nabla_{\boldsymbol{w}} \hat{Q}(s, a; \boldsymbol{w}) (1) \quad \text{true} \\
\Delta \boldsymbol{w} &= \alpha(G_t + \gamma \hat{Q}(s', a'; \boldsymbol{w}) - \hat{Q}(s, a; \boldsymbol{w}))\nabla_{\boldsymbol{w}} \hat{Q}(s, a; \boldsymbol{w}) (2) \\
\Delta \boldsymbol{w} &= \alpha(r + \gamma \underline{\max_{a'}} \hat{Q}(s', a'; \boldsymbol{w}) - \hat{Q}(s, a; \boldsymbol{w}))\nabla_{\boldsymbol{w}} \hat{Q}(s, a; \boldsymbol{w}) (3) \quad \text{true} \\
\Delta \boldsymbol{w} &= \alpha(G_t - \hat{Q}(s_t, a_t; \boldsymbol{w}))\nabla_{\boldsymbol{w}} \hat{Q}(s_t, a_t; \boldsymbol{w}) (4) \quad \text{true} \\
\Delta \boldsymbol{w} &= \alpha(r + \gamma \max_{s'} \hat{Q}(s', a; \boldsymbol{w}) - \hat{Q}(s, a; \boldsymbol{w}))\nabla_{\boldsymbol{w}} \hat{Q}(s, a; \boldsymbol{w}) (5) \quad \times
\end{aligned}
$$

Check Your Understanding L5N2: Answers

- The weight update for control for MC and TD-style methods will be near identical to the policy evaluation steps. Try to see if you can predict which are the right weight update equations for the different methods.
- (1) is the SARSA control update
  $\Delta \boldsymbol{w} = \alpha(r + \gamma \hat{Q}(s', a'; \boldsymbol{w}) - \hat{Q}(s, a; \boldsymbol{w}))\nabla_{\boldsymbol{w}} \hat{Q}(s, a; \boldsymbol{w})$
- (3) is the Q-learning control update
  $\Delta \boldsymbol{w} = \alpha(r + \gamma \max_{a'} \hat{Q}(s', a'; \boldsymbol{w}) - \hat{Q}(s, a; \boldsymbol{w}))\nabla_{\boldsymbol{w}} \hat{Q}(s, a; \boldsymbol{w})(3)$
- (4) is the MC control update
  $\Delta \boldsymbol{w} = \alpha(G_t - \hat{Q}(s_t, a_t; \boldsymbol{w}))\nabla_{\boldsymbol{w}} \hat{Q}(s_t, a_t; \boldsymbol{w})$

Linear State Action Value Function Approximation with an Oracle

- Use features to represent both the state and action

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

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

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

- Stochastic gradient descent update:

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

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

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

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

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

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

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

Convergence of TD Methods with VFA

V_1 is 1 value fn

- 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

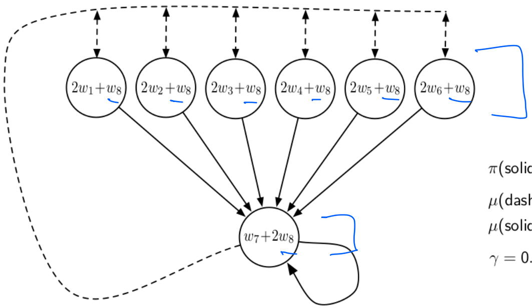
$$|BV_1 - BV_2|_\infty \leq \gamma |V_1 - V_2|_\infty$$

tabular

$$|\underset{\text{VFA}}{O} V_1 - OV_2|_\infty \geq |V_1 - V_2|_\infty$$

Geoff Gordon 1995

Challenges of Off Policy Control: Baird Example [1]



$\pi(\text{solid}|\cdot) = 1$

$\mu(\text{dashed}|\cdot) = 6/7$

$\mu(\text{solid}|\cdot) = 1/7$

$\gamma = 0.99$

- Behavior policy and target policy are not identical
- Value can diverge w can go to ∞

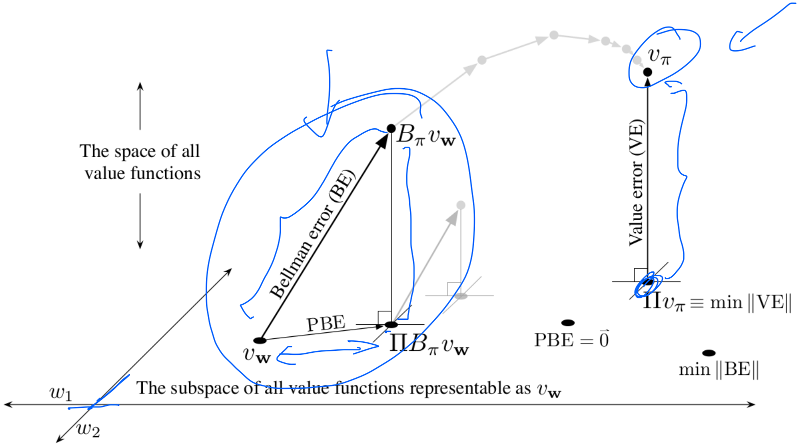
Convergence of Control Methods with VFA

Algorithm	Tabular	Linear VFA
Monte-Carlo Control	✓	✓
Sarsa	✓	✓
Q-learning	✓	✗

Important Open Area: Off Policy Learning with Function Approximation

- Extensive work in better TD-style algorithms with value function approximation, some with convergence guarantees: see Chp 11 SB
- Will come up further later in this course

Linear Value Function Approximation[4]



[4]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 π
- 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 $V^\pi(s_i)$

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

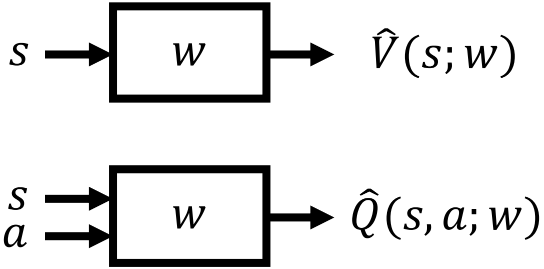
- Take the derivative and set to 0

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

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

Value Function Approximation (VFA)

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


s
w
$\hat{V}(s;w)$
s
a
w
$\hat{Q}(s,a;w)$