Lecture 5: Value Function Approximation

Emma Brunskill

CS234 Reinforcement Learning.

Winter 2019

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

Table of Contents

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

Recall: Reinforcement Learning Involves

- Optimization
- Delayed consequences
- Exploration
- Generalization

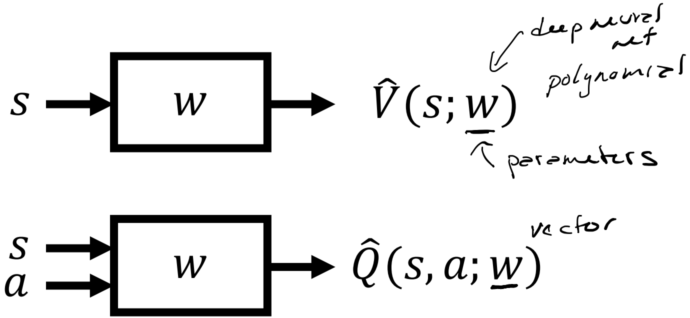
Today: Focus on Generalization

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

Table of Contents

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

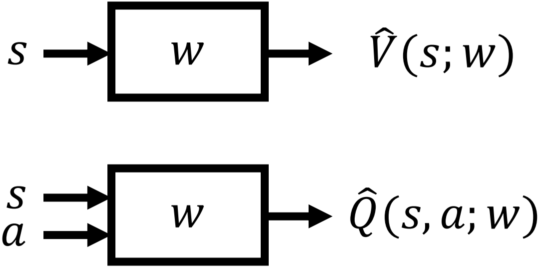
representational capacity vs memory computation data

- 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 $P, R/V/Q/\pi$

how much data is needed

Value Function Approximation (VFA)

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

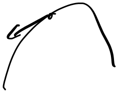


- **Which function approximator?**

Function Approximators

- Many possible function approximators including
  - Linear combinations of features
  - Neural networks
  - Decision trees highly interpretable
  - 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 $J(\boldsymbol{w})$ 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

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

$$\vec{w} \leftarrow \vec{w} - \underbrace{\alpha}_{\text{learning rate}} \nabla_w J(w)$$

<u>local</u> optima

Table of Contents

Value Function Approximation for Policy Evaluation with an Oracle

$\pi(s) \to a$
could be stochastic
$s \to p(a)$

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

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

parameterized

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

- 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) samples the gradient:

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

$$\underbrace{\Delta w}_{\text{updating } w} = \alpha (V^\pi(s) - \hat{V}(s,w)) \nabla_w V(s)$$

single point

- Expected SGD is the same as the full gradient update

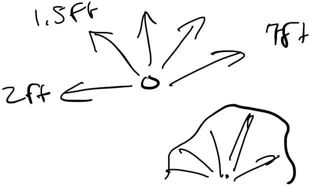
Model Free VFA Policy Evaluation

- 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

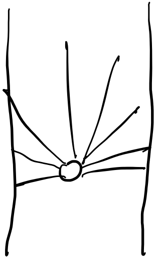
- Recall model-free policy evaluation (Lecture 3)
  - Following a fixed policy π (or had access to prior data)
  - Goal is to estimate V^π and/or Q^π
- Maintained a look up table to store estimates V^π and/or Q^π
- Updated these estimates after each episode (Monte Carlo methods) or after each step (TD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

Feature Vectors

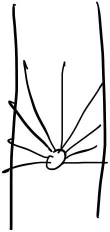


- Use a feature vector to represent a state s

floor 1



floor 2



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

180°

$$\begin{bmatrix} \text{dist at } 1° \\ 2° \\ \vdots \end{bmatrix}$$

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

vector

Q

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

- Objective function is

$s \sim \pi$

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

$x^T w$

- Recall weight update is

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

true

- Update is:

$$\Delta w = -\tfrac{1}{2}\alpha \left(2\left(V^\pi(s) - \hat{V}(s;w)\right) x(s)\right)$$

- Update = step-size × prediction error × 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: $\langle s_1, G_1\rangle, \langle s_2, G_2\rangle, \ldots, \langle s_T, G_T\rangle$ ← estimate of $V^\pi(s_T)$
  - Substitute G_t for the true $V^\pi(s_t)$ when fit function approximator
- Concretely when using linear VFA for policy evaluation

episodic

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

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

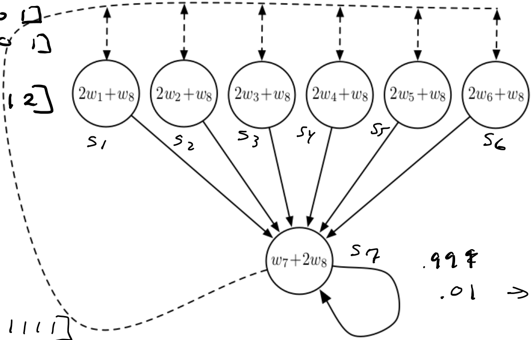
MC Linear Value Function Approximation for Policy Evaluation

$\gamma = 1$

```
Initialize w = 0, k = 1
loop
    Sample k-th episode (s_{k,1}, a_{k,1}, r_{k,1}, s_{k,2}, ..., s_{k,L_k}) given π
    for t = 1, ..., L_k do
        if First visit to (s) in episode k then        could also do every visit
            G_t(s) = Σ_{j=t}^{L_k} r_{k,j} γ^{j-t}       x^{(s)}ω
            Update weights:
              ω = ω - α(G_t(s) - V̂(s, ω)) x(s)
        end if
    end for
    k = k + 1
end loop
```

Baird (1995)-Like Example with MC Policy Evaluation[1]

s_1 [2 0 0 0 0 0 0 1]
s_2 [0 2 0 0 0 0 0 1]
s_3 [0 0 2 0 0 0 0 1]
⋮
s_7 [0 0 0 0 0 0 1 2]



rewards 0

a_1 —
a_2 - - -
w/prob 1/6 s_i
$i = 1:6$

.99
.01 → terminate

$w_0 = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]$

$\alpha = .5$

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

s_1 a_1 0 s_7 a_1 0 s_7 a_1 0 terminate

s_1 $G = 0$ $V(s_1) = 3 = x^T w$

$\Delta w = .5\,(0 - 3)\,x = -1.5\,[2 \ldots 1] = [-3\ 0\ 0\ 0\ 0\ 0\ 0\ -1.5]$

$w = w + \Delta w = [-2\ 1\ 1\ 1\ 1\ 1\ 1\ -.5]$

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

- 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:

$$\cancel{d(s) = \sum_{s'} \sum_{a} \pi(s'|a) p(s'|s,a) d(s')}$$

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

Convergence Guarantees for Linear Value Function Approximation for Policy Evaluation[2]

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

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

- 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

[2]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[1]

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

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

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

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

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

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 $V^\pi(s_i)$

set of data

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

- 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

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'):

bootstrap

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

sampling

- 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 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 of the true value $V^\pi(s)$
- 3 forms of approximation:

  - function approx
  - bootstrapping
  - sampling

  but still on policy

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:
  - $\langle s_1, r_1 + \gamma \hat{V}^{\pi}(s_2; \boldsymbol{w})\rangle, \langle s_2, r_2 + \gamma \hat{V}(s_3; \boldsymbol{w})\rangle, \ldots$
- 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)

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

TD(0) Linear Value Function Approximation for Policy Evaluation

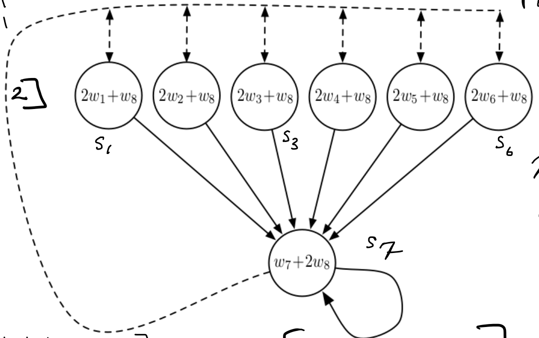
Initialize $\mathbf{w} = \mathbf{0}$, $k = 1$
loop
  Sample tuple (s_k, a_k, r_k, s_{k+1}) given π
  Update weights:

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

  $k = k + 1$
end loop

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

s_1 2 0 0 0 0 0 0 1
0 2 0 0 0 0 0 1
s_3 0 0 2 0 0 0 0 1
⋮
s_7 [0 0 0 0 0 0 1 2]



rewards all 0

$\alpha = .5$

$\gamma = .9$

a_1 —

a_2 - - -

$w_0 = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]$

$[s_1, a_1, 0, s_7]$

- TD update: $\Delta \boldsymbol{w} = \alpha(r + \gamma \underbrace{\boldsymbol{x}(s')}_{s_7}{}^T \boldsymbol{w} - \underbrace{\boldsymbol{x}(s)^T \boldsymbol{w}}_{x(s_1)w = 3})\boldsymbol{x}(s)$

new w? vs MC?

$\Delta w = \alpha(0 + .9 \cdot 3 - 3) = \alpha \cdot -.3$

[1]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 π relative to the true value as

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

- 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 minimum mean squared error possible:

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

Check Your Understanding

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

$$MSVE(\mathbf{w}_{MC}) = \min_{\mathbf{w}} \sum_{s \in S} d(s) \underbrace{(V^{\pi}(s) - \hat{V}^{\pi}(s; \mathbf{w}))^2}_{=0}$$

- 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:

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

- If the VFA is a tabular representation (one feature for each state), what is the MSVE for MC and TD? 0

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

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**

sampling

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

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

- Stochastic gradient descent (SGD) samples the gradient

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 $\underset{(s,a)}{G_t}$ 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 $r + \gamma \underbrace{\hat{Q}(s', a'; \boldsymbol{w})}_{}$ which leverages the current function approximation value

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

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

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

Convergence of TD Methods with VFA

Sutton & Barto chp 11

- TD with value function approximation is not following the gradient of an objective function
- 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

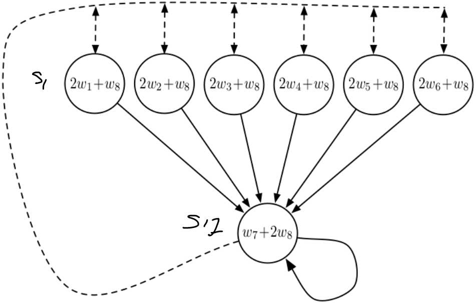
$$|OV - OV'|_\infty \leq |V - V'|$$

operator

$$\cancel{\$}\ \Phi O V$$

Geoff Gordon 1995

Challenges of Off Policy Control: Baird Example [1]



a_1 evaluate
$\pi(\text{solid}|\cdot) = 1$
a_2 behavior
$\mu(\text{dashed}|\cdot) = 6/7$
$\mu(\text{solid}|\cdot) = 1/7$
a_1
$\gamma = 0.99$

generate data under behavior policy μ

- Behavior policy and target policy are not identical
- Value can diverge (s, a, r, s')

throw away data if $a \neq \pi(s)$

Convergence of Control Methods with VFA

chatter

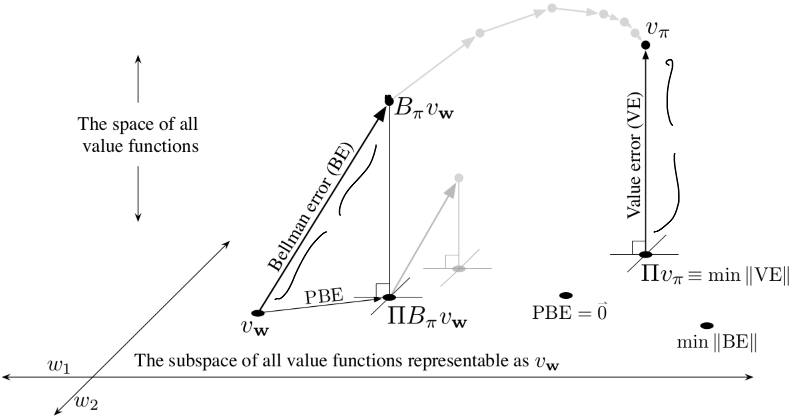
Algorithm	Tabular	Linear VFA	Nonlinear VFA
Monte-Carlo Control	✓	(✓)	✗
Sarsa	✓	(✓)	✗
Q-learning	✓	✗	✗

off policy

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 S& B
- 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[3]



[3] 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 qualitiatively: 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