Value Function Approximation [2]

DOROZHKO Anton

Novosibirsk State University

May 27, 2020

Novosibirsk State University *THE REAL SCIENCE

[2] David Silver's Lecture 6, Stanford CS231 lecture 5

Outline

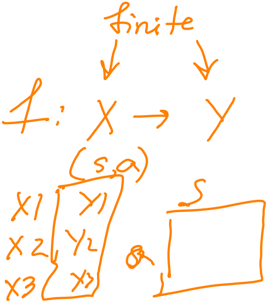
1. Introduction
2. VFA for prediction
3. Control using VFA

Novosibirsk State University
*THE REAL SCIENCE

- Previous lecture : Control (makind decision) in Model-Free case
- **This time: Value function approximation**

Last time: Model-Free Control

- How to learn policy from experience ?
- Tabular representation of $Q(s, a)$ and $V(s)$
- In real world:
  - Backgammon $\sim 10^{20}$
  - Go $\sim 10^{100}$
  - Robotic control - continuous
- Tabular is insufficient



Recall: RL Involves

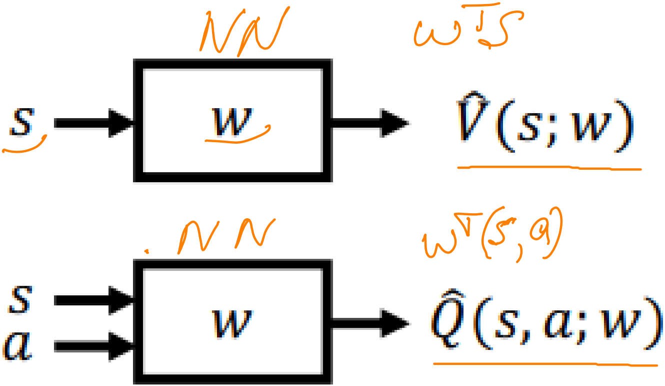
- Optimization — target, opt alg
- Delayed consequences — $o \to \cdot \to O \to \dots \rightsquigarrow R$ $\qquad G_t = R_t + \gamma \cdot$ $\downarrow$ $R_{t+2} + \dots$
- Exploration — $\lim_{t \to \infty} N(s, a) \to \infty \quad \forall s, a$
- Generalization

Recall: RL Involves

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

Value Function Approximation

Represent a (state-action / state) value function with parametrized function



Motivation for VFA

✓ model based ✗ model free

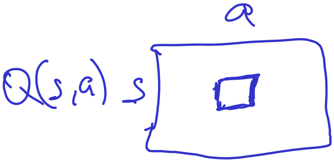
$$MDP\,(S,\ A,\ \overbrace{P,\ R},\ \gamma)$$

dynamics

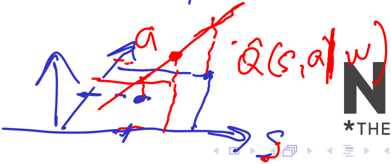
- Don’t want to explicitly store/learn for every single state
  - Dynamics or reward model $\hat{P}$, $\hat{R}$
  - Value V
  - State-action Q
  - Policy $\pi(a|s) \sim \pi_\theta(a|s)$
- Want a compact representation that generalizes across states, states-actions

Benefits of generalization

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

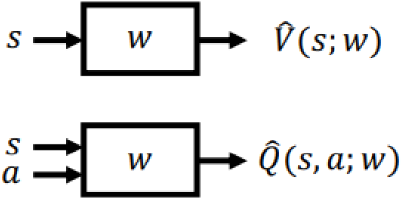


visit s, and take a



Value Function Approximation

- Represent a (state-action / state) value function with parametrized function



- **Which function approximation ?**

linear

deep RL
NN

| ML alg
RF, SVM |

Function Approximatiors

- Linear combinations of features
- Neural networks
- Decision trees
- Nearest neighbors kNN
- Fourier / wavelets

Gradient Descent

- $\underline{J(w)}$ - a differentiable function
- Find $\underline{w}$ that minimizes J
- The gradient

$$\nabla_w J(w) = \left[\frac{\partial J(w)}{\underline{\partial w_1}}, \ldots, \frac{\partial J(w)}{\partial w_N} \right]$$

$$w \leftarrow \underline{w} - \underset{\substack{\Uparrow \\ \text{learning rate}}}{\alpha} \nabla_w J(w)$$

Stochastic Gradient Descent

sample

FA $Q^\pi(s,a)$

- Find w that minimizes loss between $\underline{V^\pi(s)}$ and $\underline{\hat{V}(s;\overset{\downarrow}{w})}$
- MSE

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

- use gradient descent to find local minumum

$$\Delta w = \underline{-\frac{1}{2}\alpha\nabla_w (J(w))}$$

- SGD samples the gradient

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

$$\Delta w = \underline{\overset{\downarrow}{\alpha}\overbrace{(V^\pi(s) - \hat{V}(s;w))}^{\delta\ \text{error}}}\nabla_w \hat{V}(s;w)$$

Model Free VFA Prediction / Evaluation

~~P, R~~

- Model-free policy evaluation
  - Follow a fixed policy π
  - Estimate $V^{\pi}(s)$ and/or Q^{π}
- Maintain lookup table for $V^{\pi}(s)$ and/or Q^{π}
- Update with MC or TD
- **With VFA: update is fitting of the VFA**

Feature Vectors

RGB picture

Ra4
c4

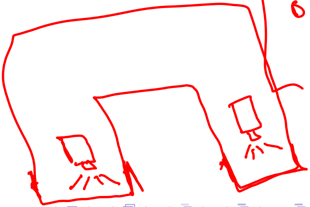
↓

0.1
0.5

Use feature vector to represent state s

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

Example: laser sensor, state aliasing



Linear VFA

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

$$\frac{\partial \hat{V}(s;w)}{\partial w} =$$

$$= x(s)$$

- Objective function

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

- Weight update:

$$\nabla w \sim -\alpha \underline{\nabla_w J(w)}$$

- Update = step-size × prediction error × feature value

α $[V^\pi - \hat{V}]$ $x(s)$

MC VFA

cum reward at the end of the episode

- G_t is unbiased noisy sample of $\underline{V^\pi(s_t)}$
- Can be used in updates with pairs

$$\underline{(s_1, G_1), \quad (s_2, G_2), \ldots, (s_T, G_T)}$$

- Update with linear VFA

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

- Note: $\underline{G_t}$ can be very noisy

MC linear VFA

```
Initialize $\mathbf{w} = \mathbf{0}$, $k = 1$
loop
    Sample $k$-th episode $(s_{k,1}, a_{k,1}, r_{k,1}, s_{k,2}, \dots, s_{k,L_k})$ given $\pi$
    for $t = 1, \dots, L_k$ do
        if First visit to $(s)$ in episode $k$ then
            $G_t(s) = \sum_{j=t}^{L_k} r_{k,j}$
            Update weights:
            $w = w - \alpha \left[ \underbrace{G_t(s) - x(s)^T w}_{error} \right] x(s)$
        end if
    end for
    $k = k + 1$
end loop
```

Baird(1995)-like example with MC policy Evaluation

$w_0 = [1, 1, 1, \dots, 1]$

$\Delta w = [-3, 0, \dots, 0, -1{,}5]$

all $r = 0$

$S1\ [2, 0\ 0\ 0\ 0\ 0\ 0\ 1] \leftarrow$

$S_1\ a_1\ r_1 = 0$

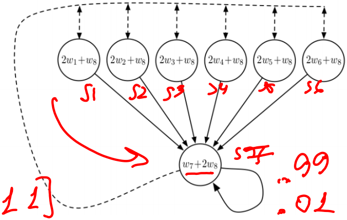
$S_7\ a_1\ 0$

$S_7\ a_1\ 0$

terminate

a_1 ——

a_2 - - -



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

.99

.01 Terminate

$\alpha = \frac{1}{2}$

$G_t = 0$

- MC update $\Delta w = \alpha(G_t - x(s_t)^T w)x(s_t)$
- With samll prob s_7 goes to terminal. $x(s_7)^T = [0, 0, 0, 0, 0, 0, 1, 2]$

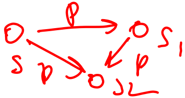
w1 w2 w3 w7 w8

$V(S_1) = X(S_1)^T w_0 =$

$= 3$

$0.5 (0 - 3) [2, 0 .. 0 1]$

Convergence Guarantees for linear VFA



fixed

- Markov Chain defined by MDP + policy will converge to some distribution over states $d(s)$
- $d(s)$ - stationary distribution of π
- $\sum_s d(s) = 1$
- $d(s)$ satisfies

$d(s)$ = p(being in state s)

$$d(s') = \sum_s \left[\sum_a \underbrace{\pi(a|s)}_{\text{fixed}} p(s'|s,a) \right] d(s)$$

$p(s \rightarrow s')$

$p(s'|s)$

$$d(\bar{s}) = P\, d(\bar{s})$$

Convergence Guarantees for linear VFA [1]

- MSE for linear VFA for π $\quad \mathbb{E}_{\pi}\left[\left(V^{\pi}(s) - \hat{V}^{\pi}(s;w)\right)^2\right]$

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

- where
  - $d(s)$ - stationary distribution
  - $\hat{V}^p i(s;w) = x(s)^T w$ linear VFA
- MC will converge to minimum MSE

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

[1] Tsitsiklis and Van Roy. An Analysis of Temporal-Difference Learning with Funciton Approximation 1997

Batch Monte-Carlo VFA

- logged episodes from a policy
- Solve analytically

$$\arg\min_{w} \sum_{i=1}^{N} (\overbrace{\underline{G(s_i)} - x(s_i)^T w}^{\text{MC error}})^2$$

- result

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

$$G \begin{bmatrix} G(s_1) \\ \vdots \\ G(s_N) \end{bmatrix}$$

$$\begin{bmatrix} x(s_1)^T \\ \vdots \\ x(s_N) \end{bmatrix}$$

- Note: computationally costly
- Note: no Markov assumptions

Recall: TD with tables

min err to prediction $G_t :\to r + \gamma V^{\pi}(s')$

play some games

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

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

- Target is $r + \gamma V^\pi(s')$ - baised estimate of $V^\pi(s)$
- Represent $V^\pi(s)$ as table

Recall: TD(0) with VFA

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

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

- Target is $r + \gamma V^\pi(s')$ - baised estimate of $V^\pi(s)$
- In VFA target is $r + \gamma \underline{\hat{V}^\pi(s'; \boxed{w)}}$
- 3 forms of approximation:

Recall: TD(0) with VFA

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

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

- Target is $r + \gamma V^\pi(s')$ - baised estimate of $V^\pi(s)$
- In VFA target is $r + \gamma \hat{V}^\pi(s'; w)$
- 3 forms of approximation:
    - function approximation

[26] David Silver's Lecture 6, Stanford CS231 lecture 5

Recall: TD(0) with VFA

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

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

- Target is $r + \gamma V^\pi(s')$ - baised estimate of $V^\pi(s)$
- In VFA target is $r + \gamma \hat{V}^\pi(s'; w)$
- 3 forms of approximation:
  - function approximation
  - bootstrapping

[26] David Silver's Lecture 6, Stanford CS231 lecture 5

Recall: TD(0) with VFA

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

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

- Target is $r + \gamma V^\pi(s')$ - baised estimate of $V^\pi(s)$
- In VFA target is $r + \gamma \underline{\hat{V}^\pi(s'; w)}$
- 3 forms of approximation:
  - function approximation ←
  - bootstrapping
  - sampling

deadly triad

← some games

[26] David Silver's Lecture 6, Stanford CS231 lecture 5

Recall: TD(0) with VFA

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

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

- Target is $r + \gamma V^\pi(s')$ - baised estimate of $V^\pi(s)$
- In VFA target is $r + \gamma \hat{V}^\pi(s'; w)$
- 3 forms of approximation:
  - function approximation
  - bootstrapping
  - sampling
- Note: we are still on-policy

[26] David Silver's Lecture 6, Stanford CS231 lecture 5

TD(0) with VFA

- Reduce TD(0) to supervised learning with

$$(s_1, \underbrace{r_1 + \gamma \hat{V}^{\pi}(s_2; w)}_{Gt}), (s_2, r_2 + \gamma \hat{V}^{\pi}(s_3; w))...$$

- Minimize

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

- Update:

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

TD(0) with VFA

```
Initialize $\mathbf{w} = \mathbf{0}$, $k = 1$
loop
    Sample tuple $(s_k, a_k, r_k, s_{k+1})$ given $\pi$        1 step
    Update weights:
        $\mathbf{w} = \mathbf{w} + \alpha(r + \gamma \underbrace{\mathbf{x}(s')^T\mathbf{w}}_{NN} - \underbrace{\mathbf{x}(s)^T\mathbf{w}}_{NN})\mathbf{x}(s)$
    $k = k + 1$
end loop
```

Convergence Guarantees for linear VFA [1]

MDP + π ⇒ Markov chain

↓

d(s)

- MSE for linear VFA for π

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

- where
  - $d(s)$ - stationary distribution
  - $\hat{V}^p i(s;w) = \underline{x(s)^T w \text{ linear VFA}}$
- ~~MC~~ TD will converge to minimum MSE

$\gamma = 0$

$G_t = R_t$

$$\underline{MSVE(w_{TD})} \boxed{\leq} \frac{1}{\cancel{1-\gamma}} min_w \sum_{s \in S} d(s)(V^\pi(s) - \hat{V}^\pi(s;w))^2$$

[1] Tsitsiklis and Van Roy. An Analysis of Temporal-Difference Learning with Funciton Approximation 1997

Control Using VFA

- use VFA to represent state-action values $\hat{Q}^{\pi}(s, a; w) \approx Q^{\pi}$
- Interleave:
  - Approximate policy eval using VFA
  - ϵ-greedy improvement ?
- Can be unstable (Deadly Triad)
  - Function approximation – DRL
  - Bootstrapping –
  - Off-policy learning

$(s, a, r, s') \sim \beta$

$a \sim \beta \quad a' \text{ at } s' \sim \pi$

greedy policy impr.

$\pi = \underset{a}{\arg\max}\, Q(s,a)$

ε-greedy

$$\pi = \begin{cases} \underset{a}{\arg\max}\, Q & p = 1-\varepsilon \\ \text{random } a & p = \varepsilon \end{cases}$$

Action-Value approximation

- Find w that minimizes loss between $Q^{\pi}(s,a)$ and $\hat{Q}(s,a;w)$
- MSE

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

- SGD samples the gradient

$$-\frac{1}{2}\nabla_w J(w) = \mathbb{E}_{\pi}[(Q^{\pi}(s,a) - \hat{Q}(s,a;w))\nabla_w \hat{Q}^{\pi}(s,a;w)]$$

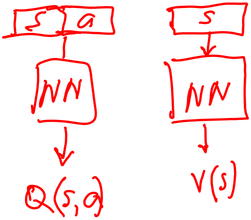
$$\Delta w = -\frac{1}{2}\alpha\nabla_w J(w)$$

Linear $Q(s, a)$ function approximation

$V(s)$

- Freatures represet both $\overset{S}{\text{state}}$ and $\overset{A}{\text{action}}$

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



- State-action function as weighted linear combination of features

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

- SGD

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

Incremental Model-Free Control

- MC

$$\Delta w = \alpha(\overset{\text{MC target}}{\overset{\downarrow}{G_t}} - \hat{Q}^{\pi}(s_t, a_t; w))\nabla_w \hat{Q}(s_t, a_t; w)$$

- SARSA on-policy

$$\Delta w = \alpha(\underbrace{r + \gamma \hat{Q}(s', \overset{\downarrow}{a'}; w)} - \hat{Q}^{\pi}(s_t, a_t; w))\nabla_w \hat{Q}(s_t, a_t; w)$$

- Q-learning off-policy

$$\Delta w = \alpha(\underbrace{r + \gamma \max_{a'} \hat{Q}(s', a'; w)}_{\text{TD target / bootstrapped}} - \hat{Q}^{\pi}(s_t, a_t; w))\nabla_w \hat{Q}(s_t, a_t; w)$$

Convergence of TD with VFA

$$\hat{Q}(s,a) = \overset{\overset{\text{reward signal from env}}{\downarrow}}{r} + \gamma \max_{a'} \hat{Q}(s',a)$$

- Sutton and Barto Ch. 11
- TD is not following gradient of $\boxed{J(w)} = \mathbb{E}\left[\left(\overset{\overset{\text{target}}{\downarrow}}{Q} - \hat{Q}(s,a|w)\right)^2\right]$
- Bellman Operator are contractions, but VFA fitting can be an expansion

Convergence of Prediction Algorithms

$\hat{V}(s)$

form of function $\hat{V}$

On/Off-Policy	Algorithm	Table Lookup	Linear	Non-Linear
On-Policy	MC	✓	✓	✓
	TD	✓	✓	✗
	Gradient TD	✓	✓	✓
Off-Policy	MC	✓	✓	✓
	TD	✓	$\boxed{✗}$	✗
	Gradient TD	✓	✓	✓

$X(s)^T w$

Convergence of Control Algorithms

$Q(s,a) = x(s,a)^T w$

Algorithm	Table Lookup	Linear	Non-Linear
Monte-Carlo Control	✓	(✓)	✗
Sarsa	✓	(✓)	✗
Q-learning	✓	$\boxed{\text{✗}}$	✗
Gradient Q-learning	✓	✓	✗

(✓) = chatters around near-optimal value function

off policy

What You Should Understand

Tabular V, Q → func approx

- Implement TD(0) and MC on policy evaluation with linear function approximation
- Be able to implement Q-learning, SARSA and MC control with function approximation
- List the 3 issues that can cause instability
  - Function approximation NN
  - Bootstrapping fit on prediction
  - Off-policy learning

deadly triad

easier to collect more experience

Explain project details