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

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

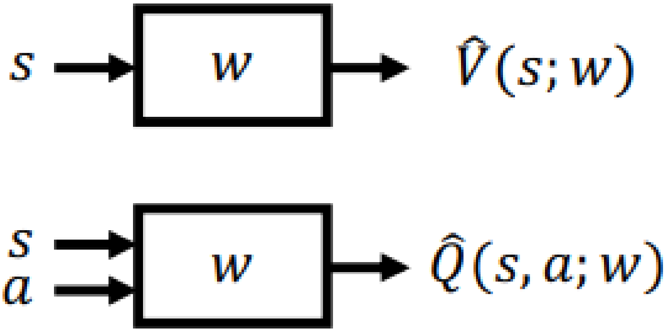
- Optimization
- Delayed consequences
- Exploration
- Generalization

Recall: RL Involves

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

Value Function Approximation

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



Motivation for VFA

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

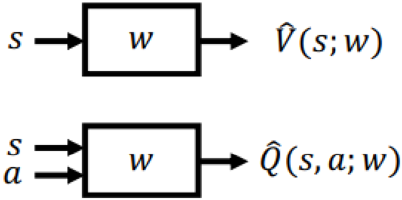
Benefits of generalization

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

Novosibirsk State University *THE REAL SCIENCE

Value Function Approximation

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



- **Which function approximation ?**

Function Approximatiors

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

Gradient Descent

- $J(w)$ - a differentiable function
- Find w that minimizes J
- The gradient

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

$$w \leftarrow w - \alpha \nabla_w J(w)$$

Stochastic Gradient Descent

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

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

- use gradient descent to find local minumum

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

- SGD samples the gradient

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

$$\Delta w = \alpha(V^{\pi}(s) - \hat{V}(s;w))\nabla_w \hat{V}(s;w)$$

Model Free VFA Prediction / Evaluation

- 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

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

- $$\hat{V}(s; w) = \sum_{j=1}^{n} x_j(s) w_j = x(s)^T w$$
- Objective function
$$J(w) = \mathbb{E}_\pi[(V^\pi(s) - \hat{V}(s; w))^2]$$
- Weight update:
$$\nabla w \sim -\alpha \nabla_w J(w)$$
- Update = step-size × prediction error × feature value

MC VFA

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

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

- Update with linear VFA

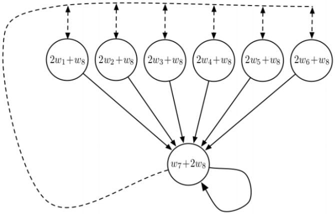
$$\begin{aligned} \Delta w &= \alpha(G_t - \hat{V}(s_t; w)\nabla_w \hat{V}(s_t; w) \\ &= \alpha(G_t - \hat{V}(s_t; w)x(s_t) \\ &= \alpha(G_t - x(s_t)^T w)x(s_t) \end{aligned}$$

- Note: G_t can be very noisy

MC linear VFA

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 π
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



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

Convergence Guarantees for linear VFA

- 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

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

Convergence Guarantees for linear VFA [1]

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

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

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

Batch Monte-Carlo VFA

- logged episodes from a policy
- Solve analytically

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

- result

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

- Note: computationally costly
- Note: no Markov assumptions

Recall: TD with tables

- 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 $V^\pi(s)$
- 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'):

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

- 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:

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'):

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

- 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

[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

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

- Minimize

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

- Update:

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

TD(0) with VFA

```
Initialize w = 0, k = 1
loop
    Sample tuple (s_k, a_k, r_k, s_{k+1}) given π
    Update weights:
        w = w + α(r + γ x(s')^T w − x(s)^T w) x(s)
    k = k + 1
end loop
```

Initialize $\mathbf{w} = \mathbf{0}$, $k = 1$

loop

  Sample tuple (s_k, a_k, r_k, s_{k+1}) given π

  Update weights:

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

  $k = k + 1$

end loop

Convergence Guarantees for linear VFA [1]

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

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

[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
  - Bootstrapping
  - Off-policy learning

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

- Freatures represet both state and 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

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

- 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

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

- SARSA

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

- Q-learning

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

Convergence of TD with VFA

- Sutton and Barto Ch. 11
- TD is not following gradient of $J(w)$
- Bellman Operator are contractions, but VFA fitting can be an expansion

Convergence of Prediction Algorithms

On/Off-Policy	Algorithm	Table Lookup	Linear	Non-Linear
On-Policy	MC	✓	✓	✓
	TD	✓	✓	✗
	Gradient TD	✓	✓	✓
Off-Policy	MC	✓	✓	✓
	TD	✓	✗	✗
	Gradient TD	✓	✓	✓

Convergence of Control Algorithms

Algorithm	Table Lookup	Linear	Non-Linear
Monte-Carlo Control	✓	(✓)	✗
Sarsa	✓	(✓)	✗
Q-learning	✓	✗	✗
Gradient Q-learning	✓	✓	✗

(✓) = chatters around near-optimal value function

What You Should Understand

- 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
    - Bootstrapping
    - Off-policy learning

Explain project details