Towards Function Approximation Methods

Easwar Subramanian

TCS Innovation Labs, Hyderabad

Email : easwar.subramanian@tcs.com / cs5500.2020@iith.ac.in

October 07, 2021

Overview

1. Review
2. Function Approximation Methods

Review

Prediction and Control Problems

- Given a MDP $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$ and a policy π, the **prediction problem** involves computing
  - ★ State-value function :

$$V^\pi(s) = \mathbb{E}_\pi \left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s \right)$$

  - ★ Action value function :

$$Q^\pi(s, a) = \mathbb{E}_\pi \left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right)$$

- Given a MDP $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$, the **control problem** involves finding optimal value functions V_* and Q_* or equivalently finding optimal policy π_*.

Model Based Techniques

Model based implies, we know $\mathcal{P}$ and $\mathcal{R}$

- Iterative Policy Evaluation (for prediction problem)

$$V_{k+1}(s) \leftarrow \sum_{a} \pi(s,a) \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma V_k(s')\right]$$

- Value Iteration (for control problem)

$$V_*(s) \leftarrow \max_{a} \left[\sum_{s' \in \mathcal{S}} \mathcal{P}^{a}_{ss'} \left(\mathcal{R}^{a}_{ss'} + \gamma V_*(s')\right)\right]$$

Drawbacks of DP Algorithms

- Requires full prior knowledge of the dynamics of the environment
- Can be implemented only on small, discrete state spaces

Model Free Techiques for Prediction

- Key idea is to estimate the following expectations using samples

$$
\begin{aligned}
V^\pi(s) &= \mathbb{E}_\pi\left(\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s\right) \\
&= \mathbb{E}_\pi\left[r_{t+1} + \gamma V^\pi(s_{t+1})|s_t = s\right]
\end{aligned}
$$

- Monte-Carlo and TD Methods
- **One Step TD for V** : If the *transition* (s, r, s') is observed at time t under policy π, then

$$V(s) \leftarrow V(s) \ + \ \alpha_t[r + \gamma V(s') - V(s)]$$

- **One step TD for Q** : Given the transition (s, a, r, s') from π, sample $a' \sim \pi(s')$ and update

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$$

Model Free Techniques for Control

- Policy is always ε-greedy with ε decay
- **SARSA Update** : Given a trajectory segment (s, a, r, s', a') generated by the ε-greedy policy, the policy evaluation steps involves the follwing update

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$$

  and policy improvement is done using ϵ-greedy with respect to current Q
- **Q-learning update (Watkins)** : Given a trajectory segment (s, a, r, s') generated by the ε-greedy policy, update

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

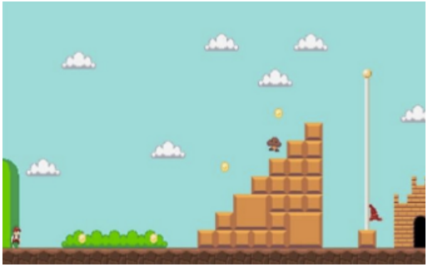
Drawback

- Not scalable to large state and action spaces

Function Approximation Methods

On the need for Function Approximators

- To solve large scale RL problems
  - ★ Game of Backgammon : 10^{20} states
  - ★ Game of Go : 10^{170} states
  - ★ Even Atari games have large state space



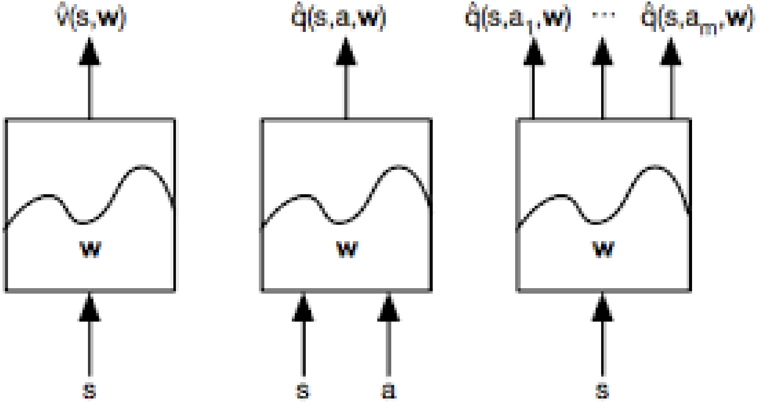
$|\mathcal{S}|$ is very large : Curse of Dimensionality

Slide Credit: David Silver : UCL lectures

Value Function Approximators

- Value function have been basically lookup tables.
- Solution for large MDP's is to use function approximators
  - ★ Generalize from seen to unseen states
- Function approximators could be
  - ★ Linear function approximator
  - ★ Neural networks
  - ★ Decision tree
  - ★ ...

Neural Network Approximators



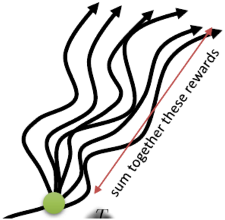
 Slide Credit: David Silver : UCL

Policy Evaluation Using Neural Networks

The value of a policy π is given by

$$
\begin{aligned}
V^{\pi}(s) &= \mathbb{E}_{\pi}\left(\sum_{k=0}^{\infty} \gamma^{k} r_{t+k+1} | s_t = s\right) \\
&= \mathbb{E}_{\pi}\left[r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s\right]
\end{aligned}
$$

Question : How do we compute the above expectations using neural networks ?



- Roll-out m trajectories from state s and observe rewards

Figure Source: Sergey Levine : UCB Course

Value Function Fitting using Monte Carlo

- Consider a MDP with a finite horizon H

$$V^{\pi}(s) \approx \frac{1}{m}\left[\sum_{j=1}^{m}\left[\sum_{k=0}^{H}\left(\gamma^{k} r^{i}_{t+k+1} | s_t = s\right)\right]\right]$$

- Need to reset the simulator back to state s (Not always possible)
- <u>Alternative</u> : Roll-out single sample estimate (high variance, but OK)
- Collect training data for as many states as possible and regress thereafter

$$\left(s_i, \underbrace{\left[\sum_{k=t}^{H}\left(\gamma^{k} r_{t+k+1} | s_t = s\right)\right]}_{=y_i}\right)$$

MC Based Algorithm

Algorithm Monte Carlo Based Value Function Fitting

Initialize number of iterations N

for $i = 1$ to N **do**

Perform a roll-out from an initial state s_i (could be any state from $\mathcal{S}$)

Calculate targets y_i using Monte-Carlo roll outs

$$y_i = \left[\sum_{k=0}^{H} \left(\gamma^k r^i_{t+k+1} | s_t = s_i \right) \right]$$

Form input-output pairs (s_i, y_i) (N datapoints in total)

end for

Perform supervised regression with loss function

$$L(\phi) = \frac{1}{2} \sum_{i=1}^{N} \left[V^{\pi}_{\phi}(s_i) - y_i \right]^2$$

Policy Evaluation : MC Based Algorithm

- Needs complete sequences, suitable only for episodic tasks

Fitted V Iteration

We observe transition (s, a, r, s') at time t; Using one step look-ahead,

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_\pi \left[r + \gamma V^\pi(s') | s_t = s \right] \\ &\approx r + \gamma V^\pi(s') \text{ (Bootstrap } V^\pi) \end{aligned}$$

Using function approximators, we get,

$$V_\phi^\pi(s) \approx r + \gamma V_\phi^\pi(s')$$

- Directly use the previous fitted value function V_ϕ^π
- Collect training data,

$$\left(s_i, \underbrace{r + V_\phi^\pi(s_i')}_{=y_i} \right)$$

- Perform supervised regression

$$L(\phi) = \frac{1}{2} \sum_{i=1}^{N} \left[V_\phi^\pi(s_i) - y_i \right]^2$$

Fitted V Iteration : Algorithm

Algorithm Fitted V Iteration

1: Initialize number of iterations N
2: **for** $j = 1$ to N **do**
3: Sample K transitions (s, a, r, s') using policy π
4: **for** $i = 1$ to K **do**
5: Calculate targets y_i using one step TD approximation

$$y_i = \left[r + V^{\pi}_{\phi_j}(s'_i)\right]$$

6: Form input-output pairs (s_i, y_i) (K datapoints in total)
7: **end for**
8: Perform supervised regression (Optimizer : RProp) using loss function

$$L(\phi_j) = \frac{1}{2}\sum_{i=1}^{K}\left[V^{\pi}_{\phi_j}(s_i) - y_i\right]^2$$

and get a new function approximator with new weights ϕ_{j+1}
9: **end for**

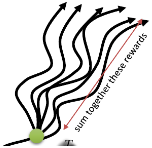
Optimal Value Function : Control

Bellman optimality equation for V_* is given by,

$$V_*(s) \leftarrow \max_a \left[\sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V_*(s') \right) \right] \approx \max_a E\left[r_{t+1} + \gamma V_*(s_{t+1}) | s_t = s) \right]$$

Question : How do we get a sample estimate for transition (s, a, r, s') for V_* ?

$$V(s) \approx \max_a [r + \gamma V(s')]$$



- To compute max over a, we need to know the outcome of all actions starting from s. Mostly not possible and costly as well.
- For model free control, we use approximators for Q and not V

Fitted Q Iteration

Bellman optimality equation for Q_*

$$Q_*(s,a) = \left[\sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q_*(s',a')\right)\right] \approx \mathbb{E}\left[r_{t+1} + \gamma \max_{a'} Q_*(s_{t+1},a') | s_t = s, a_t = a\right]$$

- Max is inside the expectation; that's ok
- For transitions (s,a,r,s') we can compute $r + \gamma \max_{a'} Q(s',a')$
- Does not require simulating over actions
- Use the previous fitted optimal Q function Q^*_ϕ like in fitted V iteration
- Collect training data,

$$\left(s_i, \underbrace{r + \gamma \max_{a'} Q_\phi(s'_i, a'_i)}_{=y_i} \right)$$

- Perform supervised regression

$$L(\phi) = \frac{1}{2} \sum_{i=1}^{N} \left[Q_\phi(s_i,a_i) - y_i\right]^2$$

Fitted Q Iteration : Algorithm

Algorithm Fitted Q Iteration

1: Initialize number of iterations N
2: **for** $j = 1$ to N **do**
3: Sample K transitions (s, a, r, s') using any behaviour policy μ
4: **for** $i = 1$ to K **do**
5: Calculate targets y_i using one step TD approximation

$$y_i = \left[r + \gamma \max_{a'} Q_{\phi_j}(s'_i, a') \right]$$

6: Form input-output pairs $(s_i, , y_i)$ (K Datapoints in total)
7: **end for**
8: Perform supervised regression (Optimizer : RProp) using loss function

$$L(\phi_j) = \frac{1}{2} \sum_{i=1}^{K} \left[Q_{\phi_j}(s_i, a_i) - y_i \right]^2$$

and get a new function approximator with new weights ϕ_{j+1}
9: **end for**