# Towards Function Approximation Methods

Easwar Subramanian

TCS Innovation Labs, Hyderabad

Email : cs5500.2020@iith.ac.in

September 16, 2023

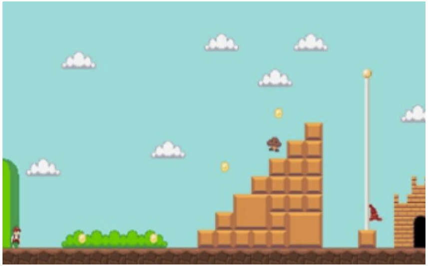
# Overview

1. Function Approximation Methods
2. Convergence of Approximation Methods

# 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

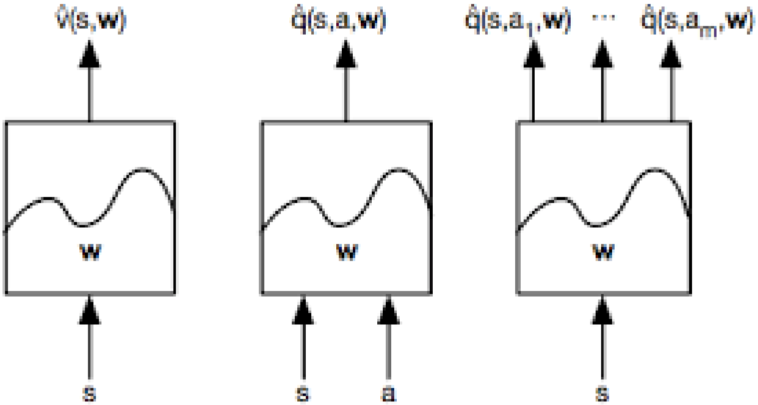


Slide Credit: David Silver : UCL lectures

# 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
  - $\dots$

# Neural Network Approximators

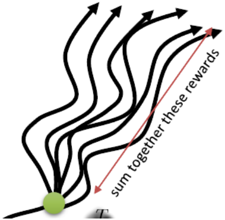


# Policy Evaluation Using Neural Networks

The value of a policy $\pi$ 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

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

**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,

$$V^{\pi}_{\phi}(s) \approx r + \gamma V^{\pi}_{\phi}(s')$$

- Directly use the previous fitted value function $V^{\pi}_{\phi}$
- Collect training data,

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

- Perform supervised regression

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

# Fitted V Iteration : Algorithm

**Algorithm** Fitted V Iteration

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

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

    Form input-output pairs $(s_i, y_i)$ ($K$ datapoints in total)
  **end for**
  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 $\phi_{j+1}$
**end for**

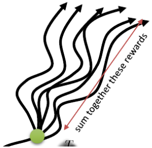
# Optimal Value Function : Control

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 $Q^*_\phi$ 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

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

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

    Form input-output pairs $(s_i, , y_i)$ ($K$ Datapoints in total)
  **end for**
  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 $\phi_{j+1}$
**end for**

# Convergence of Approximation Methods

# On the Convergence of Fitted Iterations

**Question :** What can we say about the convergence of fitted iteration methods ?

- Does fitted $V$ iteration converge to $V^\pi$ ?
- Does neural fitted iteration converge to $Q_*$ ?

**Convergence in DP setup**

- Use the fixed point equation below to define a **contraction** operator $\mathcal{L}$ (contraction in $L_\infty$ norm)

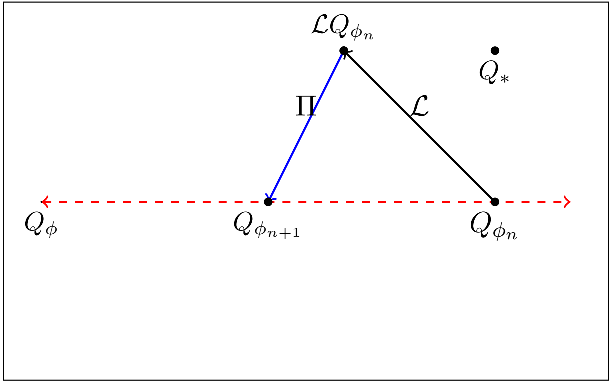
$$Q_*(s,a) \leftarrow \left[ \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left( \mathcal{R}^a_{ss'} + \gamma \max_{a'} Q_*(s',a') \right) \right]$$

**Convergence in TD setup**

- State and action spaces are finite
- All state-action pairs are visited infinitely often
- Robbins-Monroe condition: $\sum_t \alpha_t = \infty$, $\sum_t \alpha_t^2 < \infty$

# Projections and Convergence

Space of $Q$ Functions



# Convergence Guarantee For Fitted Iteration Methods

- Define operator $\mathcal{L} : \mathcal{Q} \to \mathcal{Q}$ such that

$$\mathcal{L}Q = r + \gamma \max_{a'} Q(s', a')$$

- Backup operator $\mathcal{L}$ is a contraction in $L_\infty$ norm
- Projection operator $(\Pi)$ are contractions in $L_2$ norm
- What about the composition $(\Pi \circ \mathcal{L})Q$ ?
  - ★ Need not be a contraction with respect to any norm

**Sad Corollary**

No guarantees on convergence to optimal value functions (on the manifold) exist for fitted iteration methods

# Convergence of Monte Carlo 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

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

    Form input-output pairs $(s_i, y_i)$ ($N$ datapoints in total)
**end for**
Perform supervised regression with loss function

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

# Convergence of Monte Carlo Based Algorithm

- Step 7 is gradient descent and it will converge at least local optimum
- Important : **Convergence guarantee is in the parameter space ($\phi$) and not in value function space**

# Fitted Q Iteration

**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 $\mu$
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 $\phi_{j+1}$
9: **end for**

# Online Q learning / Incremental Q learning

**Question :** Can we do the gradient update for every transition $(s, a, r, s')$ ?

- We use the fitted Q iteration and set $K = 1$
- This is also the Watkins Q-learning update (used with function approximators)

**Algorithm** Online Q Learning

**for** $n = 1$ to $N$ **do**
  Take an action $a$ and obtain the transition $(s, a, r, s')$ using $\epsilon$-greedy policy
  Calculate target $y$ using one step TD approximation

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

  Compute $g^{(n)} = \nabla_\phi (Q_{\phi_n}(s, a) - y)^2$
  Set $\phi_{n+1} = \phi_n - \alpha g^{(n)}$
**end for**

# Convergence Guarantee on Online Q learning

**Algorithm** Online Q Learning

**for** $n = 1$ to $N$ **do**
  Take an action $a$ and obtain the transition $(s, a, r, s')$ using $\epsilon$-greedy policy
  Calculate target $y$ using one step TD approximation

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

  Compute $g^{(n)} = \nabla_\phi (Q_{\phi_n}(s, a) - y)$
  Set $\phi_{n+1} \leftarrow \underbrace{\phi_n - \alpha g^{(n)}}_{\text{Is this GD ?}}$
**end for**

- Take a closer look at the one step gradient

$$g^{(n)} \leftarrow \phi_n - \alpha \nabla_\phi (Q_\phi(s, a) - \underbrace{\boxed{r + \gamma \max_{a'} Q_\phi(s', a')}}_{\text{moving target}})$$

# Summary : Convergence Discussion

- Projection ($\Pi$) of the backup operator ($\mathcal{L}$) of optimal $Q$ function need not be a contraction in any norm
- Fitted $V$ iteration or fitted $Q$ iteration need not converge because of the moving target problem
- In online $Q$ learning algorithm,
  - ★ Samples obtained are sequentially correlated
  - ★ Moving target problem
- **Convergence guarantees exist only in tabular case**