Linear Value Function Approximation: Example

Alina Vereshchaka

CSE4/510 Reinforcement Learning
Fall 2019

avereshc@buffalo.edu

October 1, 2019

*Slides are based on CS 287: Advanced Robotics, Fall 2015, Berkeley, Professor Pieter Abbeel

Overview

Table of Contents

Large-Scale Reinforcement Learning

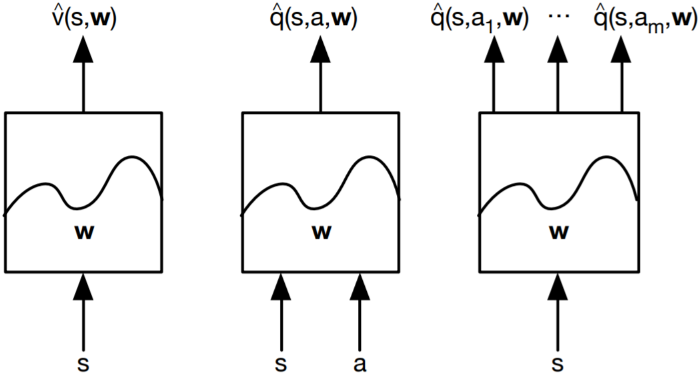
- Solution for large MDPs:
  - Estimate value function with *function approximation*

$$\hat{v}(s, \mathbf{w}) \approx v_\pi(s)$$
$$\hat{q}(s, a, \mathbf{w}) \approx q_\pi(s, a)$$

  - *Generalise* from seen states to unseen states
  - Update parameter $\mathbf{w}$ using MC or TD learning

Value Function Approximation (VFA)

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



Which function approximator?

Linear combinations of features

Feature Vectors

- Represent state by a *feature vector*

$$x(S) = \begin{bmatrix} x_1(S) \\ \vdots \\ x_n(S) \end{bmatrix}$$

- For example:
  - Distance of robot from landmarks
  - Trends in the stock market
  - Piece and pawn configurations in chess

Linear Value Function Approximation

- Represent value function by a linear combination of features

$$\hat{V}(S, \mathbf{w}) = x(S)^T \mathbf{w} = \sum_{j=1}^{n} x_j(S) \mathbf{w}_j$$

Linear Value Function Approximation

- Represent value function by a linear combination of features

$$\hat{V}(S, \mathbf{w}) = x(S)^T \mathbf{w} = \sum_{j=1}^{n} x_j(S)\mathbf{w}_j$$

- Objective function is quadratic in parameters $\mathbf{w}$

$$J(\mathbf{w}) = E_\pi\left[(V_\pi(S) - x(S)^T \mathbf{w})^2\right]$$

Linear Value Function Approximation

- Represent value function by a linear combination of features

$$\hat{V}(S,\mathbf{w}) = x(S)^T\mathbf{w} = \sum_{j=1}^{n} x_j(S)\mathbf{w}_j$$

- Objective function is quadratic in parameters $\mathbf{w}$

$$J(\mathbf{w}) = E_\pi\left[(V_\pi(S) - x(S)^T\mathbf{w})^2\right]$$

- Stochastic gradient descent converges on global optimum

Linear Value Function Approximation

- Represent value function by a linear combination of features

$$\hat{V}(S, \mathbf{w}) = x(S)^T \mathbf{w} = \sum_{j=1}^{n} x_j(S)\mathbf{w}_j$$

- Objective function is quadratic in parameters **w**

$$J(\mathbf{w}) = E_\pi\left[(V_\pi(S) - x(S)^T \mathbf{w})^2\right]$$

- Stochastic gradient descent converges on global optimum
- Update rule is particularly simple

$$\nabla_{\mathbf{w}} \hat{V}(S, \mathbf{w}) = x(S)$$

$$\Delta \mathbf{w} = \alpha(V_\pi(S) - \hat{V}(S, \mathbf{w}))x(S)$$

Linear Value Function Approximation

- Represent value function by a linear combination of features

$$\hat{V}(S, \mathbf{w}) = x(S)^T \mathbf{w} = \sum_{j=1}^{n} x_j(S) \mathbf{w}_j$$

- Objective function is quadratic in parameters $\mathbf{w}$

$$J(\mathbf{w}) = E_\pi\left[(V_\pi(S) - x(S)^T \mathbf{w})^2\right]$$

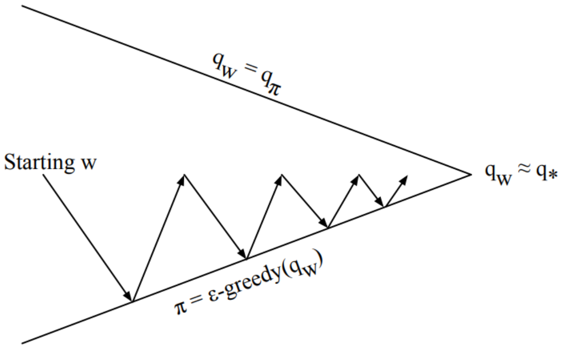
- Stochastic gradient descent converges on global optimum
- Update rule is particularly simple

$$\nabla_{\mathbf{w}} \hat{V}(S, \mathbf{w}) = x(S)$$

$$\Delta \mathbf{w} = \alpha(V_\pi(S) - \hat{V}(S, \mathbf{w}))x(S)$$

Update = step-size x prediction error x feature value

Control with Value Function Approximation



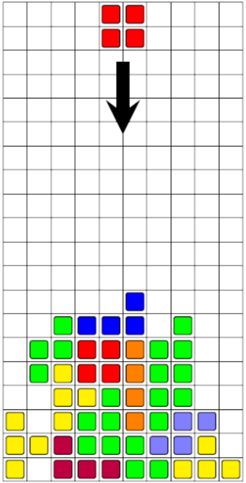
Policy evaluation **Approximate** policy evaluation, $\hat{q}(\cdot, \cdot, \mathbf{w}) \approx q_\pi$

Policy improvement ϵ-greedy policy improvement

Function Approximation: Tetris

- state: board configuration + shape of the falling piece ~2^{200} states!
- action: rotation and translation applied to the falling piece
- 22 features aka basis functions ϕ_i
  - Ten basis functions, 0, . . . , *9, mapping the state to the height h[k] of each* column.
  - Nine basis functions, 10, . . . , *18, each mapping the state to the absolute difference* between heights of successive columns: $|h[k+1] - h[k]|$, $k = 1, \ldots, 9$.
  - One basis function, 19, that maps state to the maximum column height: $\max_k h[k]$
  - One basis function, 20, that maps state to the number of 'holes' in the board.
  - One basis function, 21, that is equal to 1 in every state.

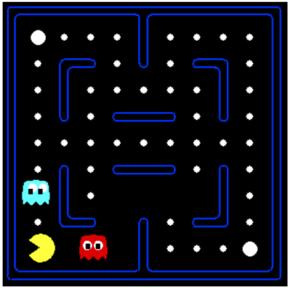
$$\hat{V}_\theta(s) = \sum_{i=0}^{21} \theta_i \phi_i(s) = \theta^\top \phi(s)$$



[Bertsekas & Ioffe, 1996 (TD); Bertsekas & Tsitsiklis 1996 (TD); Kakade 2002 (policy gradient); Farias & Van Roy, 2006 (approximate LP)]

Function Approximation: Pacman

$$
\begin{aligned}
V(s) = \quad & \theta_0 \\
& + \theta_1 \text{“distance to closest ghost”} \\
& + \theta_2 \text{“distance to closest power pellet”} \\
& + \theta_3 \text{“in dead-end”} \\
& + \theta_4 \text{“closer to power pellet than ghost”} \\
& + \ldots \\
= \quad & \sum_{i=0}^{n} \theta_i \phi_i(s) = \theta^\top \phi(s)
\end{aligned}
$$



- Examples:
  - $S = \mathbb{R}, \quad \hat{V}(s) = \theta_1 + \theta_2 s$
  - $S = \mathbb{R}, \quad \hat{V}(s) = \theta_1 + \theta_2 s + \theta_3 s^2$
  - $S = \mathbb{R}, \quad \hat{V}(s) = \sum_{i=0}^{n} \theta_i s^i$
  - $S, \quad \hat{V}(s) = \log(\frac{1}{1 + \exp(\theta^\top \phi(s))})$

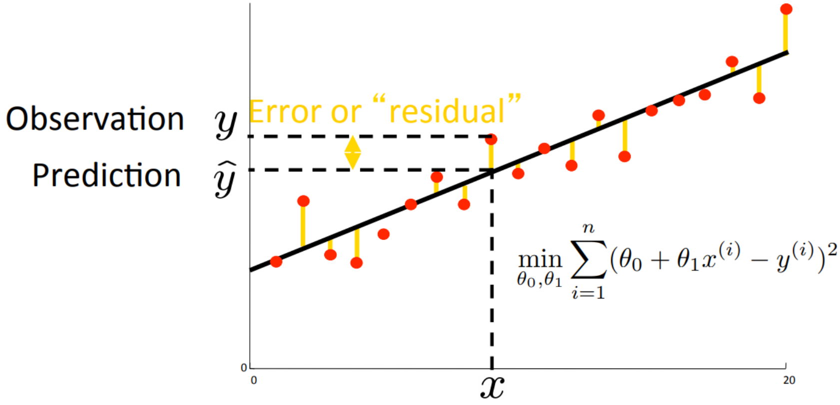
Supervised Learning

- Given:
  - set of examples $(s^{(1)}, V(s^{(1)})), (s^{(2)}, V(s^{(2)})), \ldots, (s^{(m)}, V(s^{(m)})),$
- Asked for:
  - "best" $\hat{V}_\theta$
- Representative approach: find θ through least squares

$$\min_{\theta \in \Theta} \sum_{i=1}^{m} (\hat{V}_\theta(s^{(i)}) - V(s^{(i)}))^2$$

Supervised Learning: Example

- Linear regression



Value Iteration with Function Approximation

- Pick some $S' \subseteq S$ (typically $|S'| << |S|$)
- Initialize by choosing some setting for $\theta^{(0)}$
- Iterate for i = 0, 1, 2, ..., H:
  - Step 1: Bellman back-ups

$$\forall s \in S' : \quad \bar{V}_{i+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \hat{V}_{\theta^{(i)}}(s') \right]$$

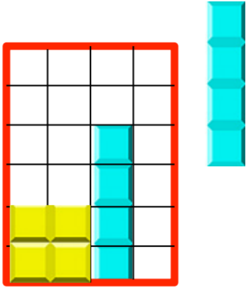
  - Step 2: Supervised learning

find $\theta^{(i+1)}$ as the solution of: $\min_\theta \sum_{s \in S'} \left(\hat{V}_{\theta^{(i+1)}}(s) - \bar{V}_{i+1}(s) \right)^2$

Table of Contents

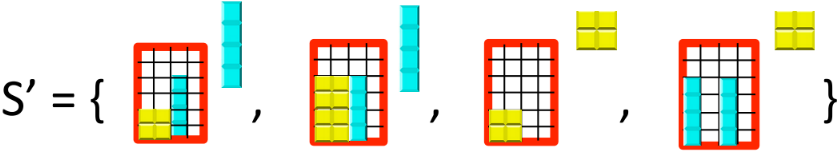
Value Iteration with Function Approximation: Example

- Mini-tetris: two types of blocks, can only choose translation (not rotation)
  - Example state:

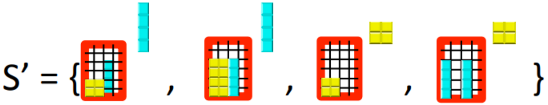


  - Reward = 1 for placing a block
  - Sink state / Game over is reached when block is placed such that part of it extends above the red rectangle
  - If you have a complete row, it gets cleared

Value Iteration with Function Approximation: Example

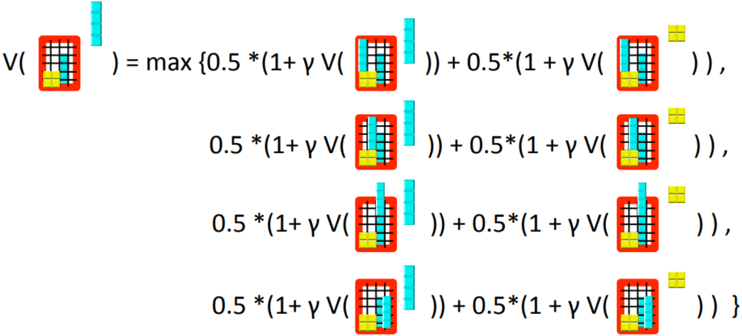


Value Iteration with Function Approximation: Example

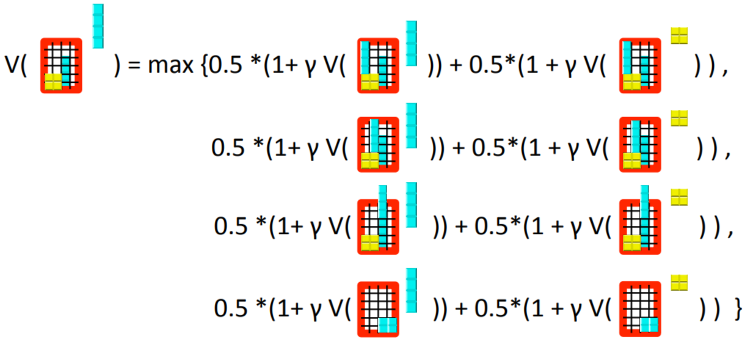


- 10 features (also called basis functions) φ_i
  - Four basis functions, 0, . . . , 3, *mapping the state to the height h[k] of each of the four* columns.
  - Three basis functions, 4, . . . , 6, *each mapping the state to the absolute difference* between heights of successive columns: $|h[k+1] - h[k]|$, $k = 1, \ldots, 3$.
  - One basis function, 7, that maps state to the maximum column height: $\max_k h[k]$
  - One basis function, 8, that maps state to the number of 'holes' in the board.
  - One basis function, 9, that is equal to 1 in every state.
- Init with $\theta^{(0)} = (-1, -1, -1, -1, -2, -2, -2, -3, -2, 10)$

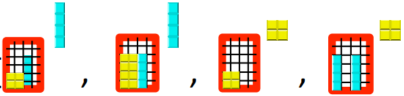
- Bellman back-ups for the states in S':



- Bellman back-ups for the states in S':



Value Iteration with Function Approximation: Example

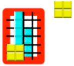
S' = {  }

- 10 features aka basis functions φ_i
  - Four basis functions, 0, . . . , 3, *mapping the state to the height h[k] of each of the four* columns.
  - Three basis functions, 4, . . . , 6, *each mapping the state to the absolute difference* between heights of successive columns: $|h[k+1] - h[k]|$, $k = 1, \dots, 3$.
  - One basis function, 7, that maps state to the maximum column height: $\max_k h[k]$
  - One basis function, 8, that maps state to the number of 'holes' in the board.
  - One basis function, 9, that is equal to 1 in every state.
- Init with $\theta^{(0)} = (-1, -1, -1, -1, -2, -2, -2, -3, -2, 10)$

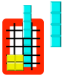
- Bellman back-ups for the states in S':

V() = max { 0.5 *(1 + γ $\theta^\top \phi$ ()) + 0.5 *(1 + γ $\theta^\top \phi$ ()) ,

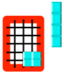
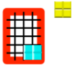
(6,2,4,0, 4, 2, 4, 6, 0, 1) (6,2,4,0, 4, 2, 4, 6, 0, 1)

0.5 *(1 + γ $\theta^\top \phi$ ()) + 0.5 *(1 + γ $\theta^\top \phi$ ()) ,

(2,6,4,0, 4, 2, 4, 6, 0, 1) (2,6,4,0, 4, 2, 4, 6, 0, 1)

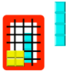
0.5 *(1 + γ V ()) + 0.5*(1 + γ V ()) ,

(sink-state, V=0) (sink-state, V=0)

0.5 *(1 + γ $\theta^\top \phi$ ()) + 0.5*(1 + γ $\theta^\top \phi$ ()) }

(0,0,2,2, 0,2,0, 2, 0, 1) (0,0,2,2, 0,2,0, 2, 0, 1)

- Bellman back-ups for the states in S':

V() = max { 0.5 *(1 + γ (-30)) + 0.5 *(1 + γ (-30)),

0.5 *(1 + γ (-30)) + 0.5 *(1 + γ (-30)),

0.5 *(1 + γ (0)) + 0.5*(1 + γ (0)),

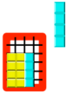
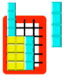
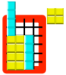
0.5 *(1 + γ (6)) + 0.5*(1 + γ (6)) }

= 6.4 (for $\gamma = 0.9$)

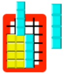
Value Iteration with Function Approximation: Example

$\theta^{(0)} = (-1, -1, -1, -1, -2, -2, -2, -3, -2, 20)$

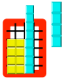
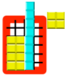
- Bellman back-ups for the second state in S':

V() = max { 0.5 *(1 + γ V ()) + 0.5 *(1 + γ V ()) ,

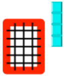
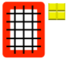
(sink-state, V=0) (sink-state, V=0)

0.5 *(1 + γ V ()) + 0.5 *(1 + γ V ()) ,

(sink-state, V=0) (sink-state, V=0)

0.5 *(1 + γ V ()) + 0.5*(1 + γ V ()) ,

(sink-state, V=0) (sink-state, V=0)

0.5 *(1 + γ $\theta^\top \phi$()) + 0.5*(1 + γ $\theta^\top \phi$()) }

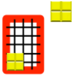
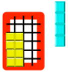
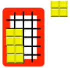
(0,0,0,0, 0,0,0, 0, 0, 1) (0,0,0,0, 0,0,0, 0, 0, 1)

-> V = 20 -> V = 20

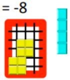
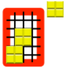
= 19

$\theta^{(0)} = (-1, -1, -1, -1, -2, -2, -2, -3, -2, 20)$

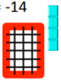
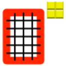
- Bellman back-ups for the third state in S':

V() = max {0.5 * (1 + γ $\theta^\top \phi$ ()) + 0.5 * (1 + γ $\theta^\top \phi$ ()) ,

(4,4,0,0, 0,4,0, 4, 0, 1) -> V = -8 | (4,4,0,0, 0,4,0, 4, 0, 1) -> V = -8

0.5 *(1 + γ $\theta^\top \phi$ ()) + 0.5 * (1 + γ $\theta^\top \phi$ ()) ,

(2,4,4,0, 2,0,4, 4, 0, 1) -> V = -14 | (2,4,4,0, 2,0,4, 4, 0, 1) -> V = -14

0.5 *(1 + γ $\theta^\top \phi$ ()) + 0.5 * (1 + γ $\theta^\top \phi$ ()) }

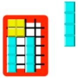
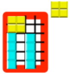
(0,0,0,0, 0,0,0, 0, 0, 1) -> V = 20 | (0,0,0,0, 0,0,0, 0, 0, 1) -> V = 20

= 19

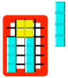
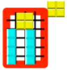
Value Iteration with Function Approximation: Example

$\theta^{(0)} = (-1, -1, -1, -1, -2, -2, -2, -3, -2, 20)$

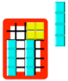
- **Bellman back-ups for the fourth state in S':**

V() = max { 0.5 * (1 + γ $\theta^\top \phi$ ()) + 0.5 * (1 + γ $\theta^\top \phi$ ()) ,

(6,6,4,0, 0,2,4, 6, 4, 1) -> V = -34   (6,6,4,0, 0,2,4, 6, 4, 1) -> V = -34

0.5 * (1 + γ $\theta^\top \phi$ ()) + 0.5 * (1 + γ $\theta^\top \phi$ ()) ,

(4,6,6,0, 2,0,6, 6, 4, 1) -> V = -38   (4,6.6,0, 2,0,6, 6, 4, 1) -> V = -38

0.5 * (1 + γ $\theta^\top \phi$ ()) + 0.5 * (1 + γ $\theta^\top \phi$ ()) }

(4,0,6,6, 4,6,0, 6, 4, 1) -> V = -42   (4,0,6,6, 4,6,0, 6, 4, 1) -> V = -42

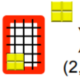
= -29.6

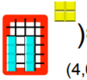
Value Iteration with Function Approximation: Example

- After running the Bellman back-ups for all 4 states in S' we have:

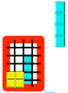
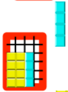
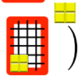
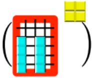
V()= 6.4
(2,2,4,0, 0,2,4, 4, 0, 1)

V()= 19
(4,4,4,0, 0,0,4, 4, 0, 1)

V()= 19
(2,2,0,0, 0,2,0, 2, 0, 1)

V()= -29.6
(4,0,4,0, 4,4,4, 4, 0, 1)

- We now run supervised learning on these 4 examples to find a new θ:

$$\min_\theta (6.4 - \theta^\top \phi($$  $$))^2$$
$$+(19 - \theta^\top \phi($$  $$))^2$$
$$+(19 - \theta^\top \phi($$  $$))^2$$
$$+((-29.6) - \theta^\top \phi($$  $$))^2$$

Running least squares gives:

$$\theta^{(1)} = (0.195, 6.24, -2.11, 0, -6.05, 0.13, -2.11, 2.13, 0, 1.59)$$