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
1 Recap: Linear Function Approximation

2 Example: Tetris
1 Recap: Linear Function Approximation

2 Example: Tetris
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?
Function Approximators

Linear combinations of features
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

\[
\hat{V}(S, w) = x(S)^T w = \sum_{j=1}^{n} x_j(S)w_j
\]
Linear Value Function Approximation

- Represent value function by a linear combination of features

\[ \hat{V}(S, w) = x(S)^T w = \sum_{j=1}^{n} x_j(S)w_j \]

- Objective function is quadratic in parameters \( w \)

\[ J(w) = E_\pi \left[ (V_\pi(S) - x(S)^T w)^2 \right] \]
Linear Value Function Approximation

- Represent value function by a linear combination of features

\[
\hat{V}(S, w) = x(S)^T w = \sum_{j=1}^{n} x_j(S)w_j
\]

- Objective function is quadratic in parameters \( w \)

\[
J(w) = E_\pi [(V_\pi(S) - x(S)^T w)^2]
\]

- Stochastic gradient descent converges on global optimum

\[
\nabla_w \hat{V}(S, w) = x(S) \Delta w = \alpha (V_\pi(S) - \hat{V}(S, w)) x(S)
\]

Update = step-size \times prediction error \times feature value
Linear Value Function Approximation

- Represent value function by a linear combination of features
  \[
  \hat{V}(S, w) = x(S)^T w = \sum_{j=1}^{n} x_j(S)w_j
  \]

- Objective function is quadratic in parameters \( w \)
  \[
  J(w) = E_\pi [(V_\pi(S) - x(S)^T w)^2]
  \]

- Stochastic gradient descent converges on global optimum

- Update rule is particularly simple
  \[
  \nabla_w \hat{V}(S, w) = x(S) \\
  \Delta w = \alpha (V_\pi(S) - \hat{V}(S, w))x(S)
  \]
Linear Value Function Approximation

- Represent value function by a linear combination of features

\[ \hat{V}(S, w) = x(S)^T w = \sum_{j=1}^{n} x_j(S)w_j \]

- Objective function is quadratic in parameters \( w \)

\[ J(w) = E_\pi [(V_\pi(S) - x(S)^T w)^2] \]

- Stochastic gradient descent converges on global optimum

- Update rule is particularly simple

\[ \nabla_w \hat{V}(S, w) = x(S) \]
\[ \Delta w = \alpha (V_\pi(S) - \hat{V}(S, w))x(S) \]

Update = step-size \( x \) prediction error \( x \) feature value
Control with Value Function Approximation

Policy evaluation  Approximate policy evaluation, $\hat{q}(\cdot, \cdot, w) \approx q_\pi$
Policy improvement  $\epsilon$-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 $\phi_i$
  - Ten basis functions, $0, \ldots, 9$, mapping the state to the height $h[k]$ of each column.
  - Nine basis functions, $10, \ldots, 18$, 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)]
V(s) = \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"}
+ ... \\
= \sum_{i=0}^{n} \theta_i \phi_i(s) = \theta^\top \phi(s)
Examples:

- $S = \mathbb{R}, \quad \hat{V}(s) = \theta_1 + \theta_2 s$

- $S = \mathbb{R}, \quad \hat{V}(s) = \theta_1 + \theta_2 s + \theta_3 s^2$

- $S = \mathbb{R}, \quad \hat{V}(s) = \sum_{i=0}^{n} \theta_i s^i$

- $S, \quad \hat{V}(s) = \log\left(\frac{1}{1 + \exp(\theta^T \phi(s))}\right)$
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 \(\theta\) 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

Observation $y$
Prediction $\hat{y}$

Error or “residual”

$$\min_{\theta_0, \theta_1} \sum_{i=1}^{n} (\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2$$
Value Iteration with Function Approximation

- Pick some $S' \subseteq S$ (typically $|S'| << |S|$)
- Initialize by choosing some setting for $\theta^{(0)}$
- Iterate for $i = 0, 1, 2, \ldots, H$:
  - Step 1: Bellman back-ups
    \[ \forall s \in S': \quad \tilde{V}_{i+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') \left[ R(s, a, s') + \gamma \tilde{V}_{\theta(i)}(s') \right] \]
  - Step 2: Supervised learning
    find $\theta^{(i+1)}$ as the solution of:
    \[ \min_{\theta} \sum_{s \in S'} \left( \tilde{V}_{\theta(i+1)}(s) - \tilde{V}_{i+1}(s) \right)^2 \]
1 Recap: Linear Function Approximation

2 Example: Tetris
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
S' = \{ \begin{array}{ccc}
\text{cell1}, & \text{cell2}, & \text{cell3}, \\
\text{cell4}, & \text{cell5}, & \text{cell6}
\end{array} \}
Value Iteration with Function Approximation: Example

\[ S' = \{ \text{tiles} \} \]

- 10 features (also called basis functions) \( \varphi_i \)
  - Four basis functions, 0, \ldots, 3, \textit{mapping the state to the height} \( h[k] \) \textit{of each of the four columns}.
  - Three basis functions, 4, \ldots, 6, \textit{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, \textit{that maps state to the maximum column height} \( \max_k h[k] \)
  - One basis function, 8, \textit{that maps state to the number of ‘holes’} in the board.
  - One basis function, 9, \textit{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 $S'$:

\[ V(s) = \max \{ 0.5 \times (1 + \gamma V(s_{next})), 0.5 \times (1 + \gamma V(s_{next})), 0.5 \times (1 + \gamma V(s_{next})), 0.5 \times (1 + \gamma V(s_{next})) \} \]
Bellman back-ups for the states in $S'$:

$$V(\text{[state 1]}) = \max \{0.5 \times (1 + \gamma V(\text{[state 2]})) + 0.5 \times (1 + \gamma V(\text{[state 3]})),$$

$$0.5 \times (1 + \gamma V(\text{[state 4]})) + 0.5 \times (1 + \gamma V(\text{[state 5]})),$$

$$0.5 \times (1 + \gamma V(\text{[state 6]})) + 0.5 \times (1 + \gamma V(\text{[state 7]})),$$

$$0.5 \times (1 + \gamma V(\text{[state 8]})) + 0.5 \times (1 + \gamma V(\text{[state 9]})) \}$$
Value Iteration with Function Approximation: Example

\[ S' = \{ \text{state 1}, \text{state 2}, \text{state 3}, \text{state 4} \} \]

- 10 features aka basis functions \( \varphi_i \):
  - Four basis functions, 0, \ldots, 3, mapping the state to the height \( h[k] \) of each of the four columns.
  - Three basis functions, 4, \ldots, 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 $S'$:

\[
V(\begin{array}{c}
\text{(sink-state, V=0)}
\end{array}) = \max \left\{ 0.5 \ast (1 + \gamma) \theta^T \phi(\begin{array}{c}
6, 2, 4, 0, 4, 2, 4, 6, 0, 1
\end{array}) + 0.5 \ast (1 + \gamma) \theta^T \phi(\begin{array}{c}
2, 6, 4, 0, 4, 2, 4, 6, 0, 1
\end{array}) ,
0.5 \ast (1 + \gamma) \theta^T \phi(\begin{array}{c}
2, 6, 4, 0, 4, 2, 4, 6, 0, 1
\end{array}) + 0.5 \ast (1 + \gamma) \theta^T \phi(\begin{array}{c}
0, 0, 2, 2, 0, 2, 0, 2, 0, 1
\end{array}),
0.5 \ast (1 + \gamma) \theta^T \phi(\begin{array}{c}
0, 0, 2, 2, 0, 2, 0, 2, 0, 1
\end{array}) \right\}
\]
Bellman back-ups for the states in $S'$:

\[
V(\text{state}) = \max \{ \ 0.5 \times (1 + \gamma \times (-30)) + 0.5 \times (1 + \gamma \times (-30)) \}\]

\[
0.5 \times (1 + \gamma \times (-30)) + 0.5 \times (1 + \gamma \times (-30)) \}
\]

\[
0.5 \times (1 + \gamma \times 0) + 0.5 \times (1 + \gamma \times 0) \}
\]

\[
0.5 \times (1 + \gamma \times 6) + 0.5 \times (1 + \gamma \times 6) \}
\]

\[= 6.4 \quad \text{(for } \gamma = 0.9)\]
Value Iteration with Function Approximation: Example

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

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

\[
V(\text{state}) = \max \{ 0.5 \cdot (1 + \gamma) \cdot V(\text{sink-state, V=0}),
0.5 \cdot (1 + \gamma) \cdot V(\text{sink-state, V=0}),
0.5 \cdot (1 + \gamma) \cdot V(\text{sink-state, V=0}),
0.5 \cdot (1 + \gamma) \cdot \theta^T \phi(x), \theta^T \phi(x) \} \]

\[
= 19 \quad \rightarrow V = 20
\]
Value Iteration with Function Approximation: Example

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

$$V(\begin{array}{c}
\text{state} \\
\end{array}) = \max \{0.5 \ast (1 + \gamma \theta^T \phi (\begin{array}{c}
4,4,0,0,0,4,0,1 \\
\end{array})) + 0.5 \ast (1 + \gamma \theta^T \phi (\begin{array}{c}
4,4,0,0,0,4,0,1 \\
\end{array})), \}
$$

$$- \rightarrow V = -8$$

$$0.5 \ast (1 + \gamma \theta^T \phi (\begin{array}{c}
2,4,4,0,2,0,4,0,1 \\
\end{array})) + 0.5 \ast (1 + \gamma \theta^T \phi (\begin{array}{c}
2,4,4,0,2,0,4,0,1 \\
\end{array})), \}

$$- \rightarrow V = -14$$

$$0.5 \ast (1 + \gamma \theta^T \phi (\begin{array}{c}
0,0,0,0,0,0,0,1 \\
\end{array})) + 0.5 \ast (1 + \gamma \theta^T \phi (\begin{array}{c}
0,0,0,0,0,0,0,1 \\
\end{array})), \}

$$- \rightarrow V = 20$$

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

$$= 19$$
Bellman back-ups for the fourth state in $S'$:

\[
V(\begin{array}{c}
\end{array}) = \max \left\{ 0.5 \times (1 + \gamma \theta^T \phi(\begin{array}{c}
\end{array})) + 0.5 \times (1 + \gamma \theta^T \phi(\begin{array}{c}
\end{array})) \right\},
\]

\[
\Rightarrow V = -34
\]

\[
0.5 \times (1 + \gamma \theta^T \phi(\begin{array}{c}
\end{array})) + 0.5 \times (1 + \gamma \theta^T \phi(\begin{array}{c}
\end{array})) ,
\]

\[
\Rightarrow V = -38
\]

\[
0.5 \times (1 + \gamma \theta^T \phi(\begin{array}{c}
\end{array})) + 0.5 \times (1 + \gamma \theta^T \phi(\begin{array}{c}
\end{array})) 
\]

\[
\Rightarrow V = -42
\]

\[
= -29.6
\]
After running the Bellman backups for all 4 states in $S'$ we have:

\[
V(2, 2, 4, 0, 0, 2, 4, 4, 0, 1) = 6.4
\]

\[
V(4, 4, 4, 0, 0, 0, 4, 4, 0, 1) = 19
\]

\[
V(2, 2, 0, 0, 0, 2, 0, 2, 0, 1) = 19
\]

\[
V(4, 0, 4, 0, 4, 4, 4, 4, 0, 1) = -29.6
\]

We now run supervised learning on these 4 examples to find a new $\theta$:

\[
\min_{\theta} (6.4 - \theta^T \phi(2, 2, 4, 0, 0, 2, 4, 4, 0, 1))^2
\]

\[
+ (19 - \theta^T \phi(4, 4, 4, 0, 0, 0, 4, 4, 0, 1))^2
\]

\[
+ (19 - \theta^T \phi(2, 2, 0, 0, 0, 2, 0, 2, 0, 1))^2
\]

\[
+ ((-29.6) - \theta^T \phi(4, 0, 4, 0, 4, 4, 4, 4, 0, 1))^2
\]

Running least squares gives:

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