Function Approximation

Pieter Abbeel
UC Berkeley EECS
Outline

- Value iteration with function approximation
- Linear programming with function approximation
Value Iteration

Algorithm:

- Start with $V_0^*(s) = 0$ for all $s$.
- For $i=1, \ldots, H$

  For all states $s \in S$:

  $$V_{i+1}^*(s) \leftarrow \max_a \sum_{s'} T(s, a, s') \left[ R(s, a, s') + \gamma V_i^*(s') \right]$$

  $$\pi_{i+1}^*(s) \leftarrow \arg \max_{a \in A} \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_i^*(s')]$$

- $V_i^*(s)$ = the expected sum of rewards accumulated when starting from state $s$ and acting optimally for a horizon of $i$ steps
- $\pi_i^*(s)$ = the optimal action when in state $s$ and getting to act for a horizon of $i$ steps

Impractical for large state spaces
Example: tetris

- state: board configuration + shape of the falling piece \( \sim 2^{200} \) 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 of the ten columns.
  - 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.

\[
\hat{V}_\theta(s) = \sum_{i=0}^{21} \theta_i \phi_i(s) = \theta^T \phi(s)
\]

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

\[ V(s) = \theta_0 + \theta_1 \text{ “distance to closest ghost”} \]
\[ + \theta_2 \text{ “distance to closest power pellet”} \]
\[ + \theta_3 \text{ “in dead-end”} \]
\[ + \theta_4 \text{ “closer to power pellet than ghost is”} \]
\[ + \ldots \]
\[ = \sum_{i=0}^{n} \theta_i \phi_i(s) = \theta^T \phi(s) \]
Function Approximation

- 0’th order approximation (1-nearest neighbor):

\[ \hat{V}(s) = \hat{V}(x_4) = \theta_4 \]

\[ \phi(s) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ \ldots \\ 0 \end{pmatrix} \quad \hat{V}(s) = \theta^\top \phi(s) \]

Only store values for \(x_1, x_2, \ldots, x_{12}\)
- call these values \(\theta_1, \theta_2, \ldots, \theta_{12}\)

Assign other states value of nearest “x” state
Function Approximation

- 1'th order approximation (k-nearest neighbor interpolation):

\[ \hat{V}(s) = \phi_1(s)\theta_1 + \phi_2(s)\theta_2 + \phi_5(s)\theta_5 + \phi_6(s)\theta_6 \]

\[ \phi(s) = \begin{pmatrix} 0.2 \\ 0.6 \\ 0 \\ 0 \\ 0.05 \\ 0.15 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \]

\[ \hat{V}(s) = \theta^\top \phi(s) \]

Only store values for x1, x2, ..., x12
- call these values \( \theta_1, \theta_2, \ldots, \theta_{12} \)

Assign other states interpolated value of nearest 4 "x" states
Function Approximation

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

Main idea:

- Use approximation $\hat{V}_\theta$ of the true value function $V$,

  - $\theta$ is a free parameter to be chosen from its domain $\Theta$

  - Representation size: $|S| \rightarrow$ downto: $|\Theta|$

  +: less parameters to estimate

  -: less expressiveness, typically there exist many $V$ for which there is no $\theta$ such that $\hat{V}_\theta = V$
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”

$$\min_{\theta_0, \theta_1} \sum_{i=1}^{n} \left( \theta_0 + \theta_1 x^{(i)} - y^{(i)} \right)^2$$
Overfitting

- To avoid overfitting: reduce number of features used

- Practical approach: leave-out validation
  - Perform fitting for different choices of feature sets using just 70% of the data
  - Pick feature set that led to highest quality of fit on the remaining 30% of data
Status

- Function approximation through supervised learning

BUT: where do the supervised examples come from?
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
    \[
    \forall s \in S' : \quad \bar{V}_{i+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') \left[ R(s, a, s') + \gamma \hat{V}_{\theta^{(i)}}(s') \right]
    \]
  - Step 2: Supervised learning
    find $\theta^{(i+1)}$ as the solution of:
    \[
    \min_{\theta} \sum_{s \in S'} \left( \hat{V}_{\theta^{(i+1)}}(s) - \bar{V}_{i+1}(s) \right)^2
    \]
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

\[ S' = \{ \} \]
Value Iteration with Function Approximation --- Example

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

- 10 features aka basis functions \( \hat{\alpha}_i \)
  - 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, 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.

- \( \text{Init } \theta^{(0)} = ( -1, -1, -1, -1, -2, -2, -2, -3, -2, 10 ) \)
Value Iteration with Function Approximation --- Example

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

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

$$0.5 \times (1 + \gamma V(\text{state}))) + 0.5 \times (1 + \gamma V(\text{state}))) ,$$

$$0.5 \times (1 + \gamma V(\text{state}))) + 0.5 \times (1 + \gamma V(\text{state}))) ,$$

$$0.5 \times (1 + \gamma V(\text{state}))) + 0.5 \times (1 + \gamma V(\text{state}))) ,$$

$$0.5 \times (1 + \gamma V(\text{state}))) + 0.5 \times (1 + \gamma V(\text{state}))) ,$$

...
Value Iteration with Function Approximation --- Example

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

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

$$0.5 \times (1 + \gamma V( \text{state} )) + 0.5 \times (1 + \gamma V( \text{state} )) ,$$

$$0.5 \times (1 + \gamma V( \text{state} )) + 0.5 \times (1 + \gamma V( \text{state} )) ,$$

$$0.5 \times (1 + \gamma V( \text{state} )) + 0.5 \times (1 + \gamma V( \text{state} )) ,$$
Value Iteration with Function Approximation --- Example

\[ S' = \{ \text{\includegraphics{image1}}, \text{\includegraphics{image2}}, \text{\includegraphics{image3}}, \text{\includegraphics{image4}} \} \]

- 10 features aka basis functions \( \hat{A}_i \)
  - 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, 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.

- \textbf{Init} \( \theta^{(0)} = (-1, -1, -1, -1, -2, -2, -2, -3, -2, 20) \)
Value Iteration with Function Approximation --- Example

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

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

\[ (0,0,2,2, 0,2,0, 2, 0, 1) \]

\[ (0,0,2,2, 0,2,0, 2, 0, 1) \]
Value Iteration with Function Approximation --- Example

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

$$V(\gamma) = \max \{0.5 \times (1+\gamma -30) + 0.5 \times (1+\gamma -30),$$

$$0.5 \times (1+\gamma -30) + 0.5 \times (1+\gamma -30),$$

$$0.5 \times (1+\gamma 0) + 0.5 \times (1+\gamma 0),$$

$$0.5 \times (1+\gamma 6) + 0.5 \times (1+\gamma 6),$$

$$= 6.4 \quad \text{(for } \gamma = 0.9)$$
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 \( S' \):

\[
V(\begin{array}{c}
\text{sink-state, } V=0
\end{array}) = \max \left\{ 0.5 \times (1 + \gamma \theta^T \phi(\begin{array}{c}
\text{sink-state, } V=0
\end{array})) + 0.5 \times (1 + \gamma \theta^T \phi(\begin{array}{c}
\text{sink-state, } V=0
\end{array})) \right\},
\]

\[
= 19
\]

\[
\begin{array}{c}
0.5 \times (1 + \gamma \theta^T \phi(\begin{array}{c}
\text{sink-state, } V=0
\end{array})) + 0.5 \times (1 + \gamma \theta^T \phi(\begin{array}{c}
\text{sink-state, } V=0
\end{array}))
\end{array}
\]

\[
(0,0,0,0, 0,0,0, 0, 0, 1) \rightarrow V = 20
\]

\[
(0,0,0,0, 0,0,0, 0, 0, 1) \rightarrow V = 20
\]
Value Iteration with Function Approximation --- Example

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

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

\[
V(\begin{array}{c}
\end{array}) = \max \{ 0.5 \times (1+\gamma \theta^T \phi(\begin{array}{c}
\end{array})) + 0.5 \times (1 + \gamma \theta^T \phi(\begin{array}{c}
\end{array})) ,
\]
\[
(4,4,0,0, 0,0, 0, 0, 1) \quad \rightarrow V = -8
\]

\[
0.5 \times (1+\gamma \theta^T \phi(\begin{array}{c}
\end{array})) + 0.5 \times (1 + \gamma \theta^T \phi(\begin{array}{c}
\end{array})) ,
\]
\[
(2,4,0,0, 0,4, 0, 0, 1) \quad \rightarrow V = -14
\]

\[
0.5 \times (1+\gamma \theta^T \phi(\begin{array}{c}
\end{array})) + 0.5 \times (1 + \gamma \theta^T \phi(\begin{array}{c}
\end{array})) ,
\]
\[
(0,0,0,0, 0,0, 0, 0, 1) \quad \rightarrow V = 20
\]

\[
= 19
\]
Value Iteration with Function Approximation --- Example

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

\[ V(\begin{array}{c} 4 \\ 0 \\ 6 \\ 6 \\ 4 \\ 6 \\ 0 \\ 6 \\ 4 \\ 1 \end{array}) = \max \left\{ 0.5 \left( 1 + \gamma \theta^T \phi \left( \begin{array}{c} 6 \\ 6 \\ 4 \\ 0 \\ 0 \\ 2 \\ 4 \\ 6 \\ 4 \\ 1 \end{array} \right) \right) + 0.5 \left( 1 + \gamma \theta^T \phi \left( \begin{array}{c} 6 \\ 6 \\ 4 \\ 0 \\ 0 \\ 2 \\ 4 \\ 6 \\ 4 \\ 1 \end{array} \right) \right), \right. \]

\[ \left. \begin{array}{c} 0.5 \left( 1 + \gamma \theta^T \phi \left( \begin{array}{c} 4 \\ 6 \\ 6 \\ 0 \\ 2 \\ 0 \\ 6 \\ 6 \\ 4 \\ 1 \end{array} \right) \right) + 0.5 \left( 1 + \gamma \theta^T \phi \left( \begin{array}{c} 4 \\ 6 \\ 6 \\ 0 \\ 2 \\ 0 \\ 6 \\ 6 \\ 4 \\ 1 \end{array} \right) \right), \right. \]

\[ \left. \begin{array}{c} 0.5 \left( 1 + \gamma \theta^T \phi \left( \begin{array}{c} 4,0,6,6 \\ 4,6,0,6,4,1 \end{array} \right) \right) + 0.5 \left( 1 + \gamma \theta^T \phi \left( \begin{array}{c} 4,0,6,6 \\ 4,6,0,6,4,1 \end{array} \right) \right), \right. \]

\[ \left. \begin{array}{c} \rightarrow V = -34 \\ \rightarrow V = -38 \\ \rightarrow V = -42 \end{array} \right. \]

\[ = -29.6 \]
Value Iteration with Function Approximation --- Example

- After running the Bellman back-ups 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 $\theta$:

$$\min_{\theta} \left( 6.4 - \theta^\top \phi(2,2,4,0, 0,2,4, 4, 0, 1) \right)^2$$

$$+ \left( 19 - \theta^\top \phi(4,4,4,0, 0,0,4, 4, 0, 1) \right)^2$$

$$+ \left( 19 - \theta^\top \phi(2,2,0,0, 0,2,0, 2, 0, 1) \right)^2$$

$$+ \left( -29.6 - \theta^\top \phi(4,0,4,0, 4,4,4, 4, 0, 1) \right)^2$$

$\Rightarrow$ Running least squares gives new $\theta$

$$\theta^{(1)} = (0.195, 6.24, -2.11, 0, -6.05, 0.13, -2.11, 2.13, 0, 1.59)$$
Potential guarantees?
Simple example**

Function approximator: $[1 \ 2] \ast \theta$
Simple example**

\[ \tilde{J}_\theta = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \theta \]

\[ \tilde{J}^{(1)}(x_1) = 0 + \gamma \tilde{J}_{\theta^{(0)}}(x_2) = 2\gamma \theta^{(0)} \]
\[ \tilde{J}^{(1)}(x_2) = 0 + \gamma \tilde{J}_{\theta^{(0)}}(x_2) = 2\gamma \theta^{(0)} \]

Function approximation with least squares fit:

\[ \begin{bmatrix} 1 \\ 2 \end{bmatrix} \theta^{(1)} \approx \begin{bmatrix} 2\gamma \theta^{(0)} \\ 2\gamma \theta^{(0)} \end{bmatrix} \]

Least squares fit results in:

\[ \theta^{(1)} = \frac{6}{5\gamma} \theta^{(0)} \]

Repeated back-ups and function approximations result in:

\[ \theta^{(i)} = \left( \frac{6}{5\gamma} \right)^i \theta^{(0)} \]

which diverges if \( \gamma > \frac{5}{6} \) even though the function approximation class can represent the true value function.]
Composing operators**

- **Definition.** An operator G is a *non-expansion* with respect to a norm \( ||\cdot|| \) if

\[
||GJ_1 - GJ_2|| \leq ||J_1 - J_2||
\]

- **Fact.** If the operator F is a \( \gamma \) contraction with respect to a norm \( ||\cdot|| \) and the operator G is a non-expansion with respect to the same norm, then the sequential application of the operators G and F is a \( \gamma \)-contraction, i.e.,

\[
||GFJ_1 - GFJ_2|| \leq \gamma ||J_1 - J_2||
\]

- **Corollary.** If the supervised learning step is a non-expansion, then iteration in value iteration with function approximation is a \( \gamma \)-contraction, and in this case we have a convergence guarantee.
Averager function approximators are non-expansions**

**DEFINITION:** A real-valued function approximation scheme is an averager if every fitted value is the weighted average of zero or more target values and possibly some predetermined constants. The weights involved in calculating the fitted value $Y_i$ may depend on the sample vector $X_0$, but may not depend on the target values $Y$. More precisely, for a fixed $X_0$, if $Y$ has $n$ elements, there must exist $n$ real numbers $k_i$, $n^2$ nonnegative real numbers $\beta_{ij}$, and $n$ nonnegative real numbers $\beta_i$, so that for each $i$ we have $\beta_i + \sum_j \beta_{ij} = 1$ and $\hat{Y}_i = \beta_i k_i + \sum_j \beta_{ij} Y_j$.

- **Examples:**
  - nearest neighbor (aka state aggregation)
  - linear interpolation over triangles (tetrahedrons, ...)

---
Averager function approximators are non-expansions**

**Proof:** Let $J_1$ and $J_2$ be two vectors in $\mathbb{R}^n$. Consider a particular entry $s$ of $\Pi J_1$ and $\Pi J_2$:

$$
\left| (\Pi J_1)(s) - (\Pi J_2)(s) \right| = \left| \beta_{s0} + \sum_{s'} \beta_{ss'} J_1(s') - \beta_{s0} + \sum_{s'} \beta_{ss'} J_2(s') \right|
$$

$$
= \left| \sum_{s'} \beta_{ss'} (J_1(s') - J_2(s')) \right|
$$

$$
\leq \max_{s'} \left| J_1(s') - J_2(s') \right|
$$

$$
= \| J_1 - J_2 \|_{\infty}
$$

This holds true for all $s$, hence we have

$$
\| \Pi J_1 - \Pi J_2 \|_{\infty} \leq \| J_1 - J_2 \|_{\infty}
$$
Linear regression

Figure 2: The mapping associated with linear regression when samples are taken at the points $x = 0, 1, 2$. In (a) we see a target value function (solid line) and its corresponding fitted value function (dotted line). In (b) we see another target function and another fitted function. The first target function has values $y = 0, 0, 0$ at the sample points; the second has values $y = 0, 1, 1$. Regression exaggerates the difference between the two functions: the largest difference between the two target functions at a sample point is 1 (at $x = 1$ and $x = 2$), but the largest difference between the two fitted functions at a sample point is $\frac{7}{5}$ (at $x = 2$).

[Example taken from Gordon, 1995.]
Guarantees for fixed point**

**Theorem.** Let $J^*$ be the optimal value function for a finite MDP with discount factor $\gamma$. Let the projection operator $\Pi$ be a non-expansion w.r.t. the infinity norm and let $\tilde{J}$ be any fixed point of $\Pi$. Suppose $\|\tilde{J} - J^*\|_\infty \leq \epsilon$. Then $\Pi T$ converges to a value function $\tilde{J}$ such that:

$$\|\tilde{J} - J^*\| \leq 2\epsilon + \frac{2\gamma\epsilon}{1 - \gamma}$$

- I.e., if we pick a non-expansion function approximator which can approximate $J^*$ well, then we obtain a good value function estimate.

- To apply to discretization: use continuity assumptions to show that $J^*$ can be approximated well by chosen discretization scheme
Outline

✓ Value iteration with function approximation

- Linear programming with function approximation
Infinite Horizon Linear Program

\[
\min_V \sum_{s \in S} \mu_0(s)V(s)
\]

s.t. \( V(s) \geq \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V(s')] \), \( \forall s \in S, a \in A \)

\( \mu_0 \) is a probability distribution over \( S \), with \( \mu_0(s) > 0 \) for all \( s \in S \).

**Theorem.** \( V^* \) is the solution to the above LP.
Infinite Horizon Linear Program

\[
\min_{\hat{V}} \sum_{s \in S} \mu_0(s) V(s)
\]

s.t. \( V(s) \geq \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V(s')] \), \( \forall s \in S, a \in A \)

Let: \( \hat{V}(s) = \theta^\top \phi(s) \), and consider \( S' \) rather than \( S \):

\[
\min_\theta \sum_{s \in S'} \mu_0(s) \theta^\top \phi(s)
\]

s.t. \( \theta^\top \phi(s) \geq \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma \theta^\top \phi(s')] \), \( \forall s \in S', a \in A \)

\( \Rightarrow \) Linear program that finds \( \hat{V}_\theta(s) = \theta^\top \phi(s) \)
Approximate Linear Program – Guarantees**

\[
\min_{\theta} \sum_{s \in S'} \mu_0(s) \theta^T \phi(s) \\
\text{s.t.} \quad \theta^T \phi(s) \geq \sum_{s'} T(s, a, s') \left[ R(s, a, s') + \gamma \theta^T \phi(s') \right], \quad \forall s \in S', a \in A
\]

- LP solver will converge
- Solution quality: [de Farias and Van Roy, 2002]

Assuming one of the features is the feature that is equal to one for all states, and assuming \( S' = S \) we have that:

\[
\|V^* - \Phi \theta\|_{1, \mu_0} \leq \frac{2}{1 - \gamma} \min_{\theta} \|V^* - \Phi \theta\|_{\infty}
\]

(slightly weaker, probabilistic guarantees hold for \( S' \) not equal to \( S \), these guarantees require size of \( S' \) to grow as the number of features grows)