

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 $i=1, \dots, H$

For all states $s \in S$:

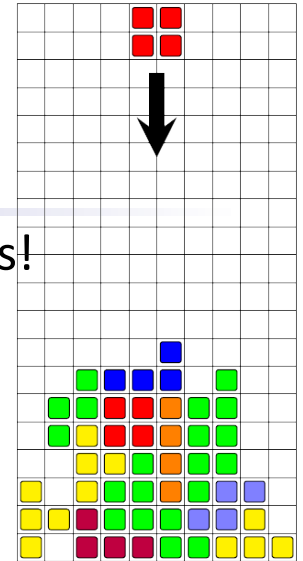
$$V_{i+1}^*(s) \leftarrow \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_i^*(s')]$$

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

- $V_i^*(s)$ = 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

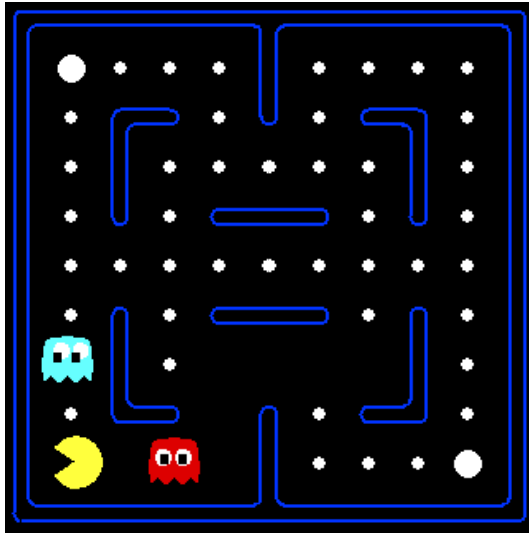
Example: tetris



- 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 ϕ_i
 - Ten basis functions, $0, \dots, 9$, mapping the state to the height $h[k]$ of each of the ten columns.
 - Nine basis functions, $10, \dots, 18$, each mapping the state to the absolute difference between heights of successive columns: $|h[k+1] - h[k]|$, $k = 1, \dots, 9$.
 - 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.

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

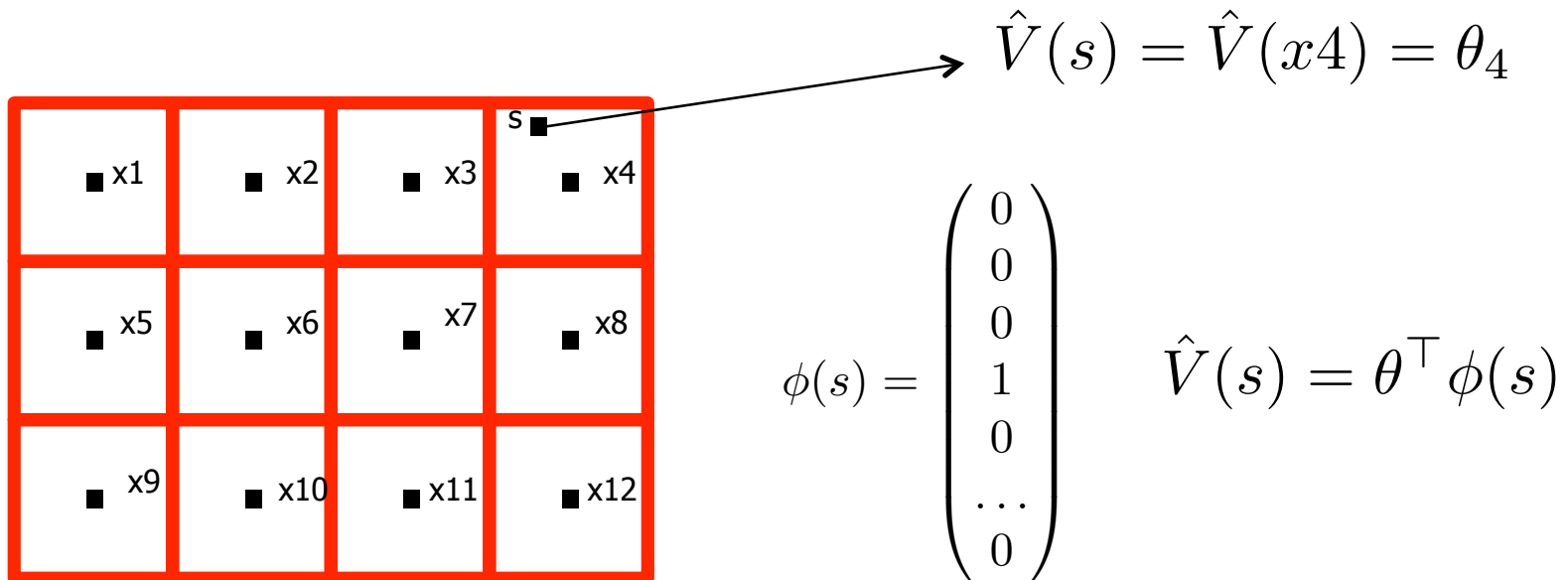
Function Approximation



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

Function Approximation

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



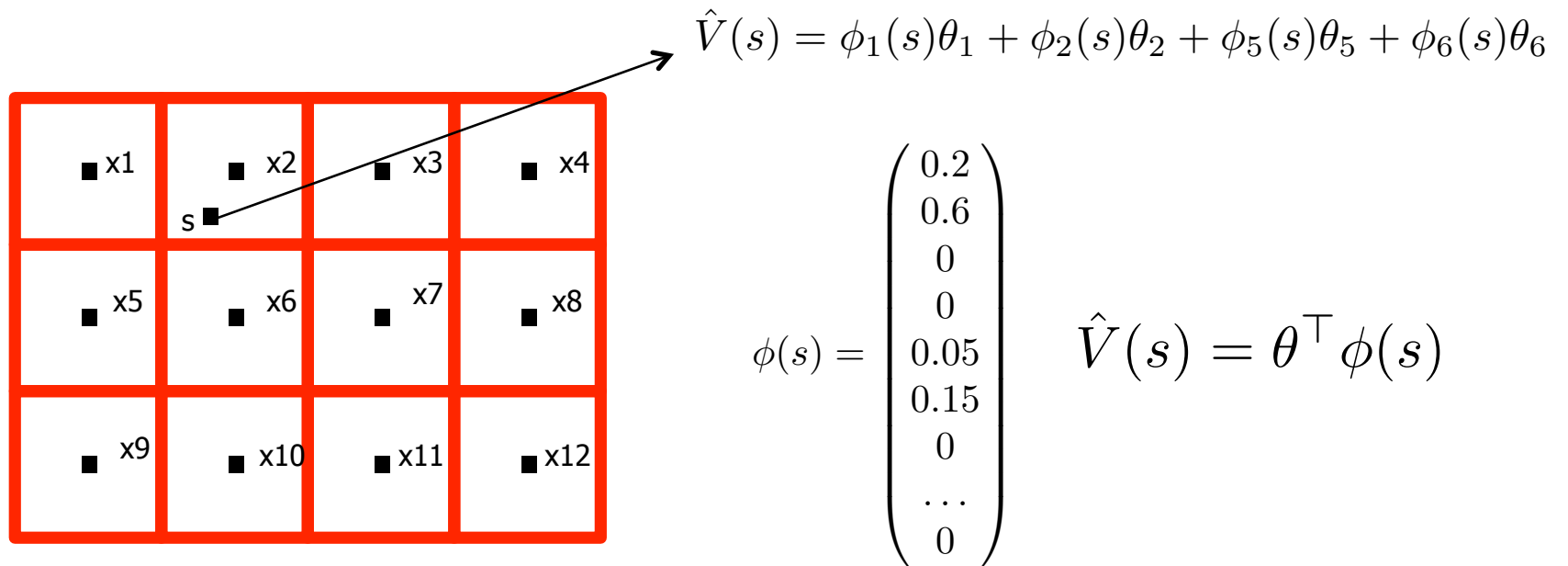
Only store values for x_1, x_2, \dots, x_{12}

– call these values $\theta_1, \theta_2, \dots, \theta_{12}$

Assign other states value of nearest “x” state

Function Approximation

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



Only store values for x_1, x_2, \dots, x_{12}

– call these values $\theta_1, \theta_2, \dots, \theta_{12}$

Assign other states interpolated value of nearest 4 “x” states

Function Approximation

- Examples:

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

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

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

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

Function Approximation

- Main idea:

- Use approximation \hat{V}_θ of the true value function V ,

- θ is a free parameter to be chosen from its domain Θ

- Representation size: $|S| \rightarrow$ down to: $|\Theta|$

+ : less parameters to estimate

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

Supervised Learning

- Given:

- set of examples

$$(s^{(1)}, V(s^{(1)}), (s^{(2)}, V(s^{(2)}), \dots, (s^{(m)}, V(s^{(m)}))$$

- Asked for:

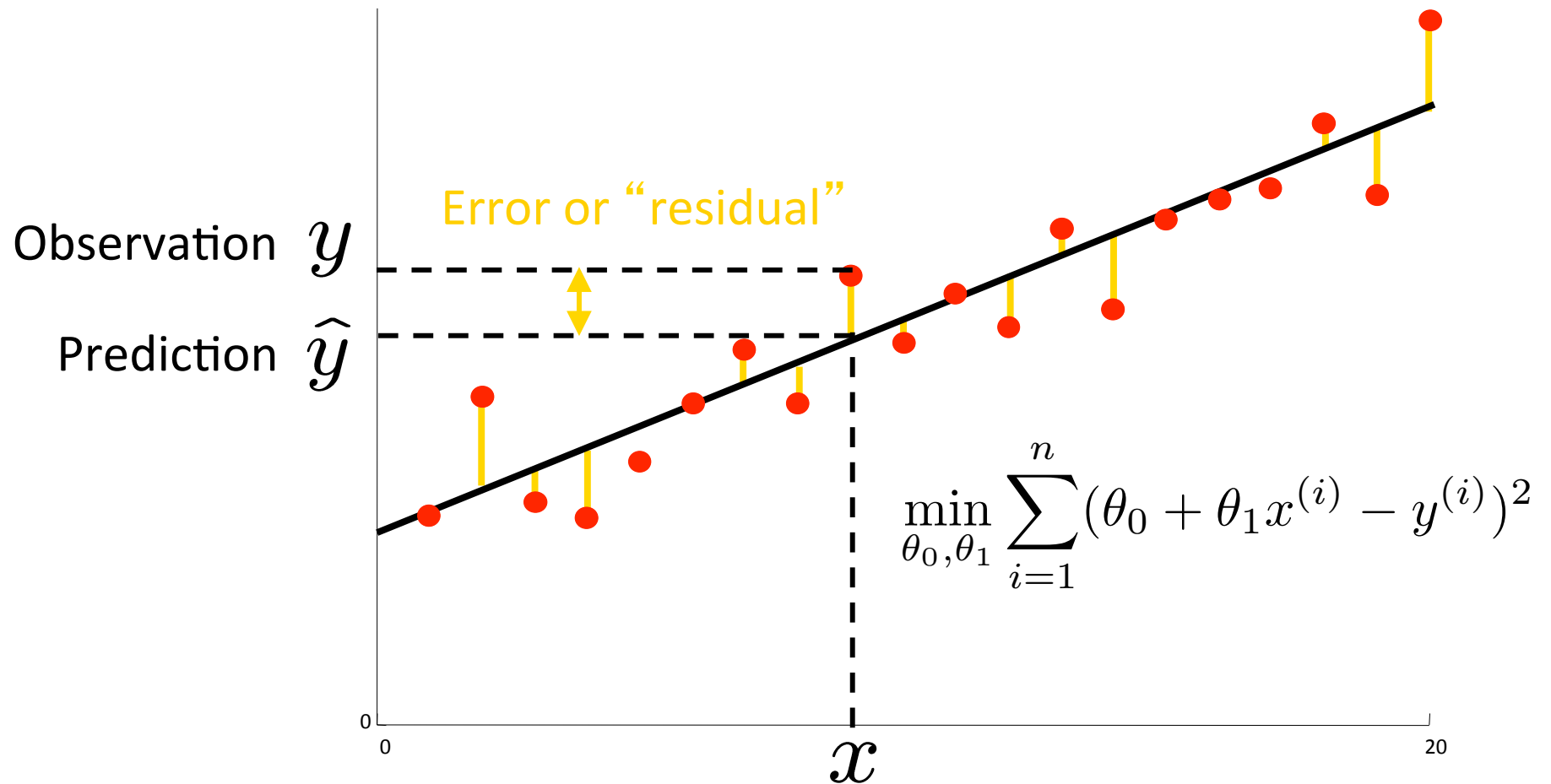
- “best” \hat{V}_θ

- 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



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

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

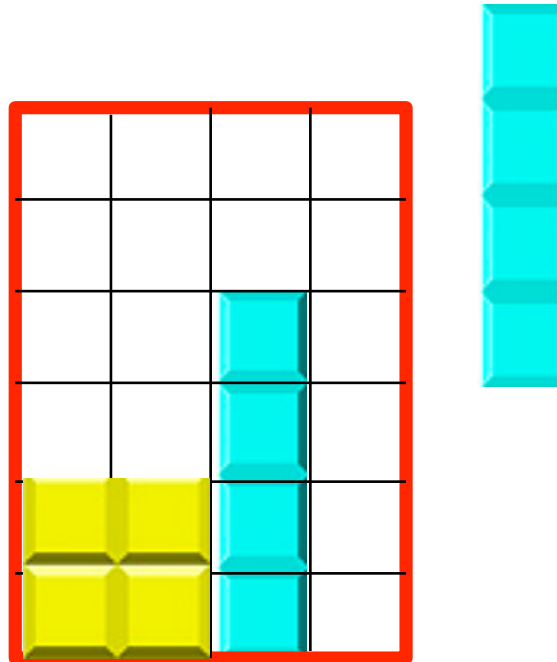
- Step 2: Supervised learning

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

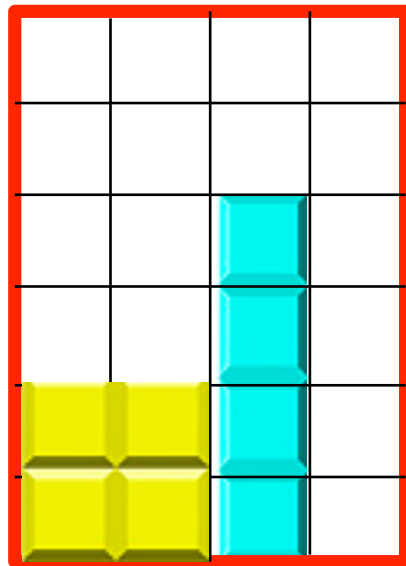
- Mini-tetris: two types of blocks, can only choose translation (not rotation)
 - Example state:



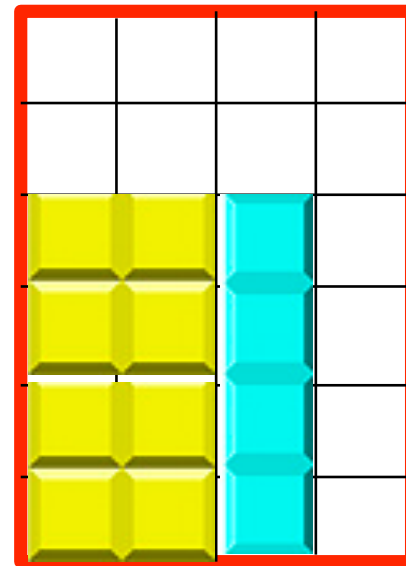
- Reward = 1 for placing a block
- Sink state / Game over is reached when block is placed such that part of it extends above the red rectangle
- If you have a complete row, it gets cleared

Value Iteration with Function Approximation --- Example

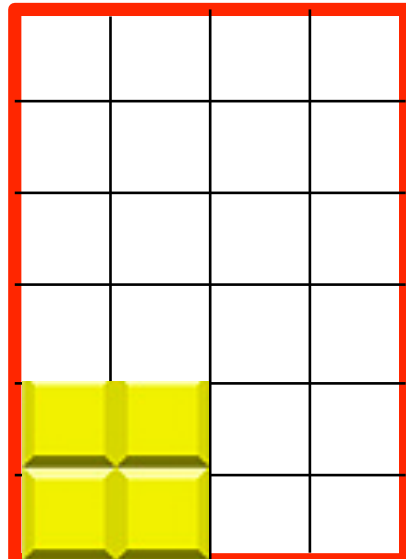
$S' = \{$



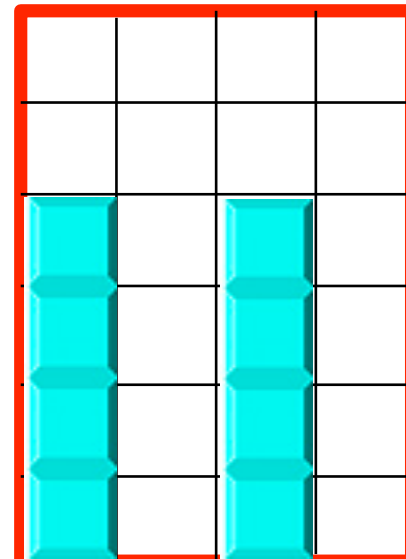
,



,

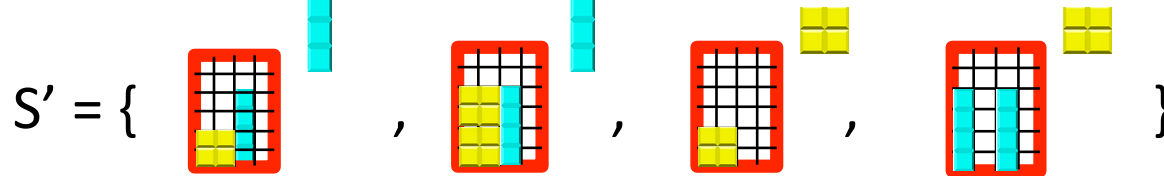


,



}

Value Iteration with Function Approximation --- Example



- 10 features aka basis functions \hat{A}_i
 - Four basis functions, $0, \dots, 3$, mapping the state to the height $h[k]$ of each of the four columns.
 - Three basis functions, $4, \dots, 6$, 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 $\theta^{(0)} = (-1, -1, -1, -1, -2, -2, -2, -3, -2, 10)$

Value Iteration with Function Approximation --- Example

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

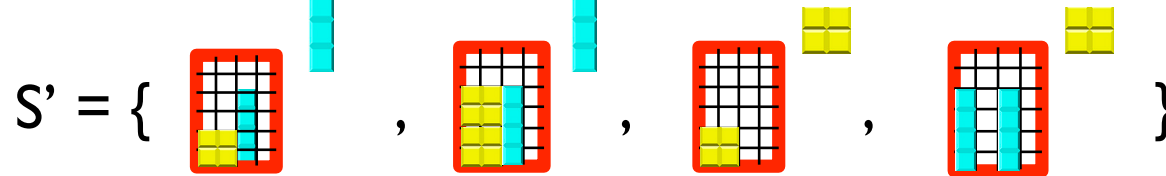
$$\begin{aligned}
 V(\text{state}) &= \max \{ 0.5 * (1 + \gamma V(\text{state}_1)) + 0.5 * (1 + \gamma V(\text{state}_2)) , \\
 & 0.5 * (1 + \gamma V(\text{state}_3)) + 0.5 * (1 + \gamma V(\text{state}_4)) , \\
 & 0.5 * (1 + \gamma V(\text{state}_5)) + 0.5 * (1 + \gamma V(\text{state}_6)) , \\
 & 0.5 * (1 + \gamma V(\text{state}_7)) + 0.5 * (1 + \gamma V(\text{state}_8)) ,
 \end{aligned}$$

Value Iteration with Function Approximation --- Example

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

$$\begin{aligned}
 V(\text{state}) &= \max \{ 0.5 * (1 + \gamma V(\text{state}_1)) + 0.5 * (1 + \gamma V(\text{state}_2)) , \\
 & 0.5 * (1 + \gamma V(\text{state}_3)) + 0.5 * (1 + \gamma V(\text{state}_4)) , \\
 & 0.5 * (1 + \gamma V(\text{state}_5)) + 0.5 * (1 + \gamma V(\text{state}_6)) , \\
 & 0.5 * (1 + \gamma V(\text{state}_7)) + 0.5 * (1 + \gamma V(\text{state}_8)) ,
 \end{aligned}$$

Value Iteration with Function Approximation --- Example



- 10 features aka basis functions \hat{A}_i
 - Four basis functions, $0, \dots, 3$, mapping the state to the height $h[k]$ of each of the four columns.
 - Three basis functions, $4, \dots, 6$, 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 $\theta^{(0)} = (-1, -1, -1, -1, -2, -2, -2, -3, -2, 20)$

Value Iteration with Function Approximation --- Example

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

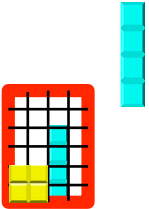
$$\begin{aligned}
 V(\text{state}) = & \max \{ 0.5 * (1 + \gamma \theta^T \phi(\text{state}_1)) + 0.5 * (1 + \gamma \theta^T \phi(\text{state}_2)), \\
 & 0.5 * (1 + \gamma \theta^T \phi(\text{state}_3)) + 0.5 * (1 + \gamma \theta^T \phi(\text{state}_4)), \\
 & 0.5 * (1 + \gamma \theta^T \phi(\text{state}_5)) + 0.5 * (1 + \gamma \theta^T \phi(\text{state}_6)), \\
 & 0.5 * (1 + \gamma \theta^T \phi(\text{state}_7)) + 0.5 * (1 + \gamma \theta^T \phi(\text{state}_8)) \}
 \end{aligned}$$

The states are represented by 4x4 grids with a red border and a cyan bar on the right. The state vectors are:

- State 1: (6,2,4,0, 4, 2, 4, 6, 0, 1)
- State 2: (6,2,4,0, 4, 2, 4, 6, 0, 1)
- State 3: (2,6,4,0, 4, 2, 4, 6, 0, 1)
- State 4: (2,6,4,0, 4, 2, 4, 6, 0, 1)
- State 5: (sink-state, V=0)
- State 6: (sink-state, V=0)
- State 7: (0,0,2,2, 0,2,0, 2, 0, 1)
- State 8: (0,0,2,2, 0,2,0, 2, 0, 1)

Value Iteration with Function Approximation --- Example

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



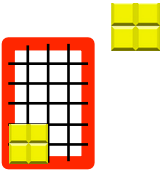
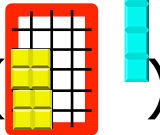
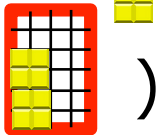
$$\begin{aligned}
 V(\text{state}) = & \max \{ 0.5 * (1 + \gamma * -30) + 0.5 * (1 + \gamma * -30), \\
 & 0.5 * (1 + \gamma * -30) + 0.5 * (1 + \gamma * -30), \\
 & 0.5 * (1 + \gamma * 0) + 0.5 * (1 + \gamma * 0), \\
 & 0.5 * (1 + \gamma * 6) + 0.5 * (1 + \gamma * 6), \\
 & = 6.4 \quad (\text{for } \gamma = 0.9)
 \end{aligned}$$

Value Iteration with Function Approximation --- Example

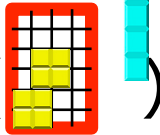
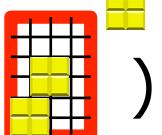
$$\theta^{(0)} = (-1, -1, -1, -1, -2, -2, -2, -3, -2, 20)$$

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

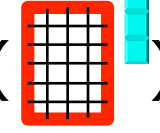
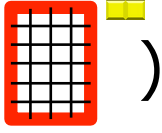
$$\begin{aligned}
 V(\text{grid}) = & \max \{ 0.5 * (1 + \gamma \theta^T \phi(\text{grid}_1)) + 0.5 * (1 + \gamma \theta^T \phi(\text{grid}_2)) , \\
 & 0.5 * (1 + \gamma \theta^T \phi(\text{grid}_3)) + 0.5 * (1 + \gamma \theta^T \phi(\text{grid}_4)) , \\
 & 0.5 * (1 + \gamma \theta^T \phi(\text{grid}_5)) + 0.5 * (1 + \gamma \theta^T \phi(\text{grid}_6)) ,
 \end{aligned}$$

 $(4,4,0,0, 0,4,0, 4, 0, 1)$ $(4,4,0,0, 0,4,0, 4, 0, 1)$
 $\rightarrow V = -8$ $\rightarrow V = -8$

 $(2,4,4,0, 2,0,4, 4, 0, 1)$ $(2,4,4,0, 2,0,4, 4, 0, 1)$
 $\rightarrow V = -14$ $\rightarrow V = -14$

 $(0,0,0,0, 0,0,0, 0, 0, 1)$ $(0,0,0,0, 0,0,0, 0, 0, 1)$
 $\rightarrow V = 20$ $\rightarrow V = 20$

$$= 19$$

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' :

$$V(\text{grid}) = \max \left\{ \begin{aligned} &0.5 * (1 + \gamma \theta^T \phi(\text{grid}_1)) + 0.5 * (1 + \gamma \theta^T \phi(\text{grid}_2)), \\ &0.5 * (1 + \gamma \theta^T \phi(\text{grid}_3)) + 0.5 * (1 + \gamma \theta^T \phi(\text{grid}_4)), \\ &0.5 * (1 + \gamma \theta^T \phi(\text{grid}_5)) + 0.5 * (1 + \gamma \theta^T \phi(\text{grid}_6)) \end{aligned} \right. ,$$

$(6,6,4,0, 0,2,4, 6, 4, 1) \rightarrow V = -34$ $(6,6,4,0, 0,2,4, 6, 4, 1) \rightarrow V = -34$

$(4,6,6,0, 2,0,6, 6, 4, 1) \rightarrow V = -38$ $(4,6,6,0, 2,0,6, 6, 4, 1) \rightarrow V = -38$

$(4,0,6,6, 4,6,0, 6, 4, 1) \rightarrow V = -42$ $(4,0,6,6, 4,6,0, 6, 4, 1) \rightarrow V = -42$

$$= -29.6$$

Value Iteration with Function Approximation --- Example

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

$$V(\text{state 1}) = 6.4$$

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

$$V(\text{state 2}) = 19$$

(4,4,4,0, 0,0,4, 4, 0, 1)

$$V(\text{state 3}) = 19$$

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

$$V(\text{state 4}) = -29.6$$

(4,0,4,0, 4,4,4, 4, 0, 1)

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

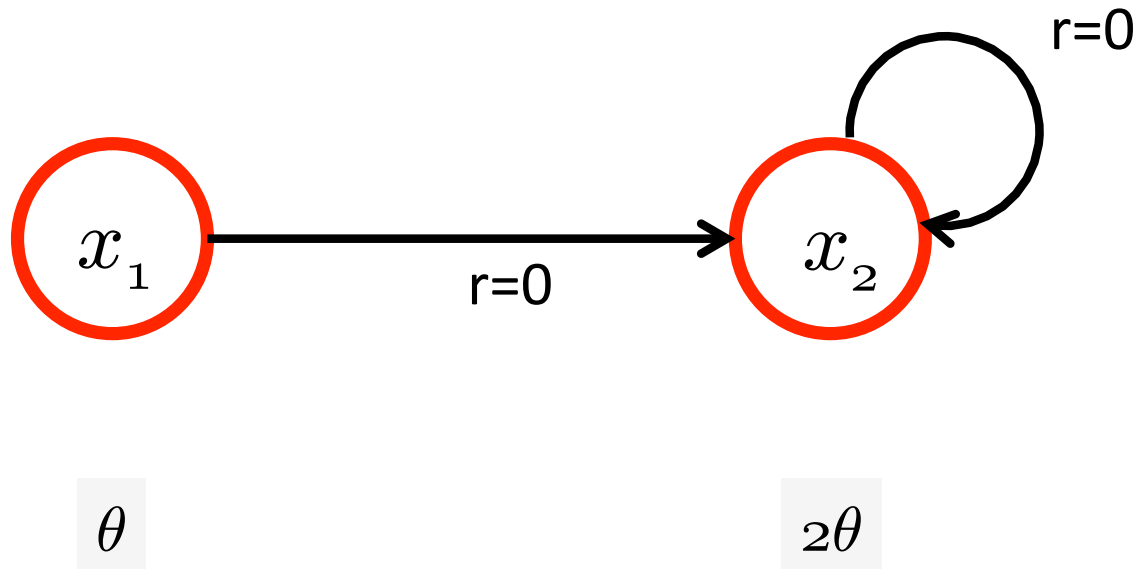
$$\min_{\theta} (6.4 - \theta^T \phi(\text{state 1}))^2 + (19 - \theta^T \phi(\text{state 2}))^2 + (19 - \theta^T \phi(\text{state 3}))^2 + ((-29.6) - \theta^T \phi(\text{state 4}))^2$$

→ Running least squares gives new θ

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

Potential guarantees?

Simple example**



Function approximator: $[1 \ 2] * \theta$

Simple example**

$$\bar{J}_\theta = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \theta$$

$$\begin{aligned} \bar{J}^{(1)}(x_1) &= 0 + \gamma \hat{J}_{\theta^{(0)}}(x_1) = 2\gamma\theta^{(0)} \\ \bar{J}^{(1)}(x_2) &= 0 + \gamma \hat{J}_{\theta^{(0)}}(x_2) = 2\gamma\theta^{(0)} \end{aligned}$$

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 γ 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 γ -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 γ -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 \hat{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 β_{ij} , and n nonnegative real numbers β_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 \mathfrak{R}^n . Consider a particular entry s of ΠJ_1 and ΠJ_2 :

$$\begin{aligned} |(\Pi J_1)(s) - (\Pi J_2)(s)| &= |\beta_{s0} + \sum_{s'} \beta_{ss'} J_1(s') - \beta_{s0} + \sum_{s'} \beta_{ss'} J_2(s')| \\ &= \left| \sum_{s'} \beta_{ss'} (J_1(s') - J_2(s')) \right| \\ &\leq \max_{s'} |J_1(s') - J_2(s')| \\ &= \|J_1 - J_2\|_\infty \end{aligned}$$

This holds true for all s , hence we have

$$\|\Pi J_1 - \Pi J_2\|_\infty \leq \|J_1 - J_2\|_\infty$$

Linear regression ☹️ **

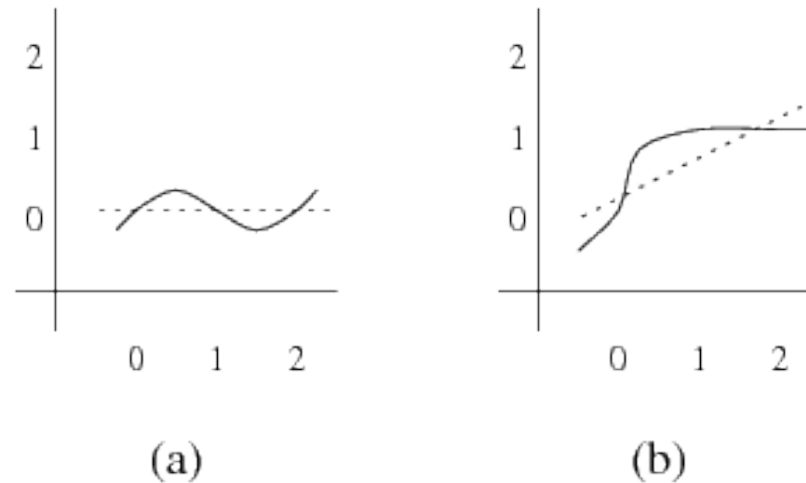


Figure 2: The mapping associated with linear regression when samples are taken at the points $x = 0, 1, 2$. In (a) we see a target value function (solid line) and its corresponding fitted value function (dotted line). In (b) we see another target function and another fitted function. The first target 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 $\frac{7}{6}$ (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 γ . Let the projection operator Π be a non-expansion w.r.t. the infinity norm and let \tilde{J} be any fixed point of Π . Suppose $\|\tilde{J} - J^*\|_\infty \leq \epsilon$. Then ΠT converges to a value function \bar{J} such that:

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

- 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

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

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

μ_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

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

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

- Let: $V(s) = \theta^\top \phi(s)$, and consider S' rather than S :

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

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

→ Linear program that finds $\hat{V}_\theta(s) = \theta^\top \phi(s)$

Approximate Linear Program – Guarantees**

$$\begin{aligned} & \min_{\theta} \sum_{s \in S'} \mu_0(s) \theta^\top \phi(s) \\ \text{s.t. } & \theta^\top \phi(s) \geq \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma \theta^\top \phi(s')], \quad \forall s \in S', a \in A \end{aligned}$$

- 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)