Function Approximation

Pieter Abbeel
UC Berkeley EECS

Value Iteration

Algorithm:

Start with $V_0^*(s) = 0$ for all s.

Impractical for large state spaces

For i = 1, … , H

For all states s in S:

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

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

This is called a value update or Bellman update/back-up

$V_i^*(s)$ = expected sum of rewards accumulated starting from state s, acting optimally for i steps

$\pi_i^*(s)$ = optimal action when in state s and getting to act for i steps

Similar issue for policy iteration and linear programming

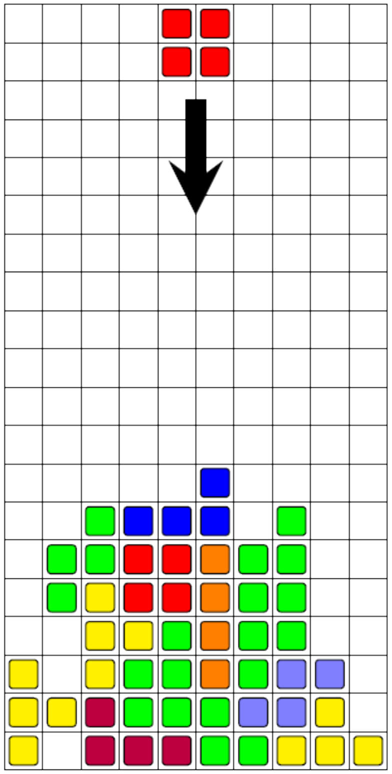
Outline

- Function approximation
- Value iteration with function approximation
- Policy iteration with function approximation
- Linear programming with function approximation

Function Approximation Example1 : 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, . . . , 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.

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



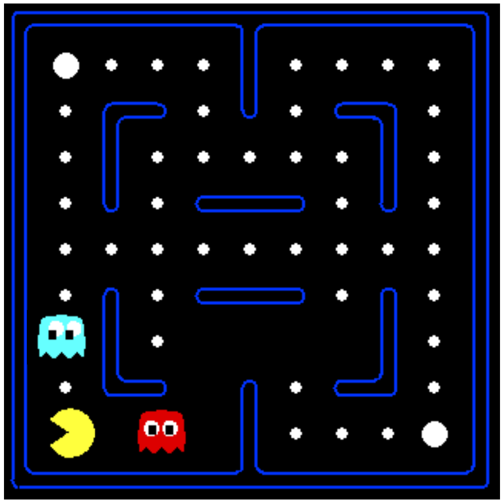
[Bertsekas & Ioffe, 1996 (TD); Bertsekas & Tsitsiklis 1996 (TD); Kakade 2002 (policy gradient); Farias & Van Roy, 2006 (approximate LP)]

Function Approximation Example 2: Pacman

V(s) = θ_0

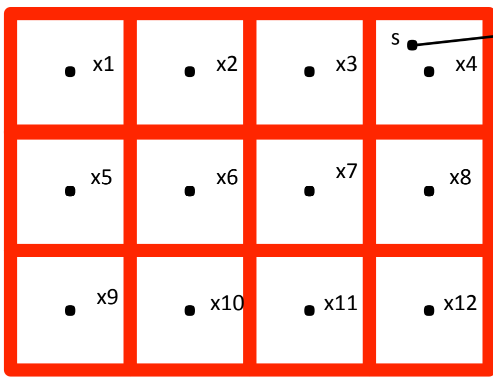
+ θ_1 "distance to closest ghost"
+ θ_2 "distance to closest power pellet"
+ θ_3 "in dead-end"
+ θ_4 "closer to power pellet than ghost"
+ ...

= $$\sum_{i=0}^{n} \theta_i \phi_i(s) = \theta^\top \phi(s)$$



Function Approximation Example 3: Nearest Neighbor

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



$$\hat{V}(s) = \hat{V}(x4) = \theta_4$$

$$\phi(s) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ \dots \\ 0 \end{pmatrix} \qquad \hat{V}(s) = \theta^\top \phi(s)$$

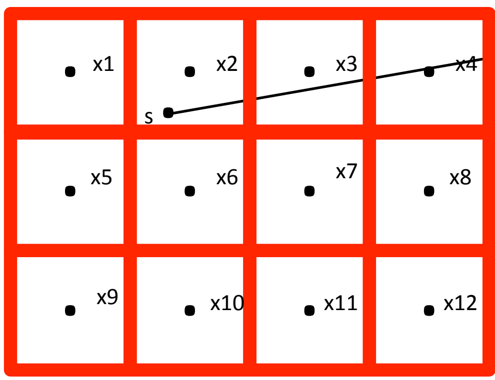
Only store values for x1, x2, …, x12

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

Assign other states value of nearest "x" state

Function Approximation Example 4: k-Nearest Neighbor

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



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

$$\phi(s) = \begin{pmatrix} 0.2 \\ 0.6 \\ 0 \\ 0 \\ 0.05 \\ 0.15 \\ 0 \\ \dots \\ 0 \end{pmatrix} \qquad \hat{V}(s) = \theta^\top \phi(s)$$

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

More Function Approximation Examples

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

Function Approximation

- Main idea:
  - Use approximation $\hat{V}_\theta$ of the true value function V,
    - θ is a free parameter to be chosen from its domain Θ
    - Representation size: $|S|$ → downto: $|\Theta|$

  + : less parameters to estimate

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

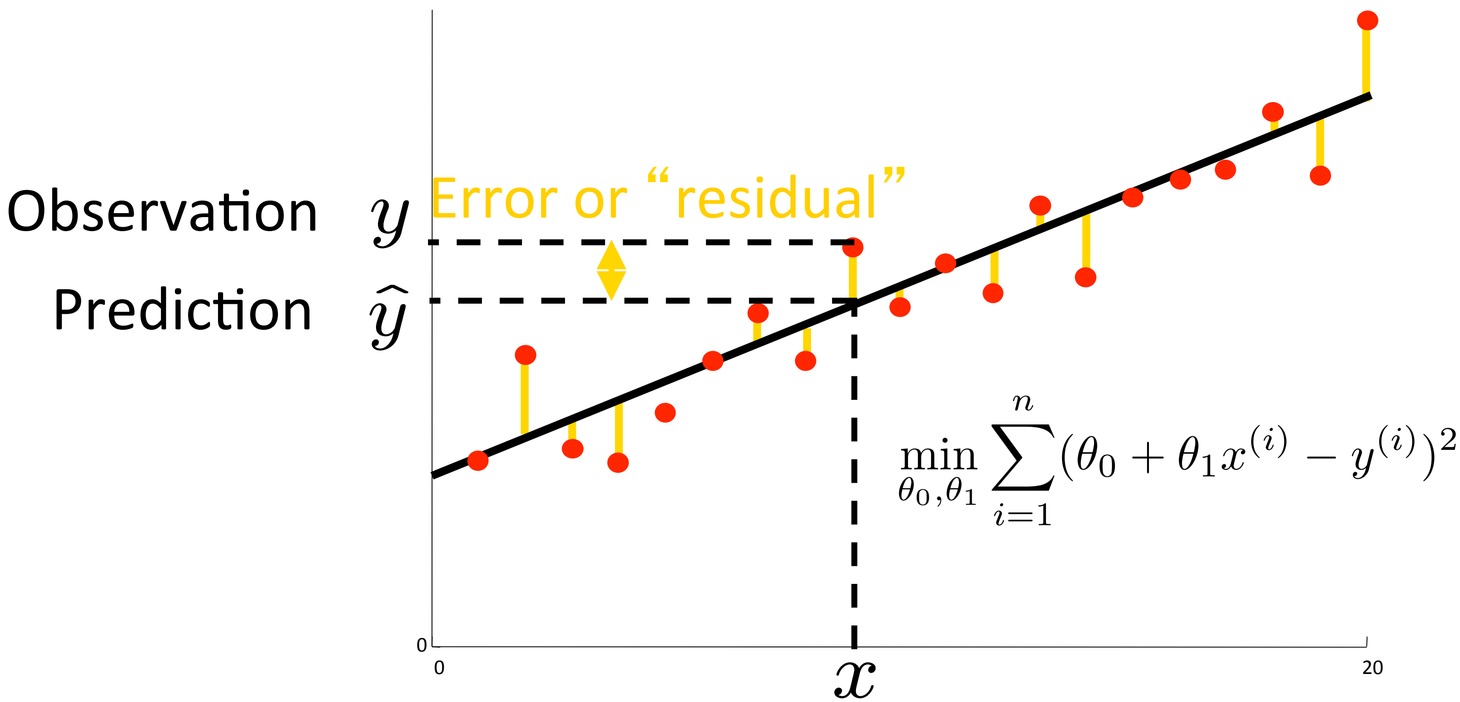
Supervised Learning

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

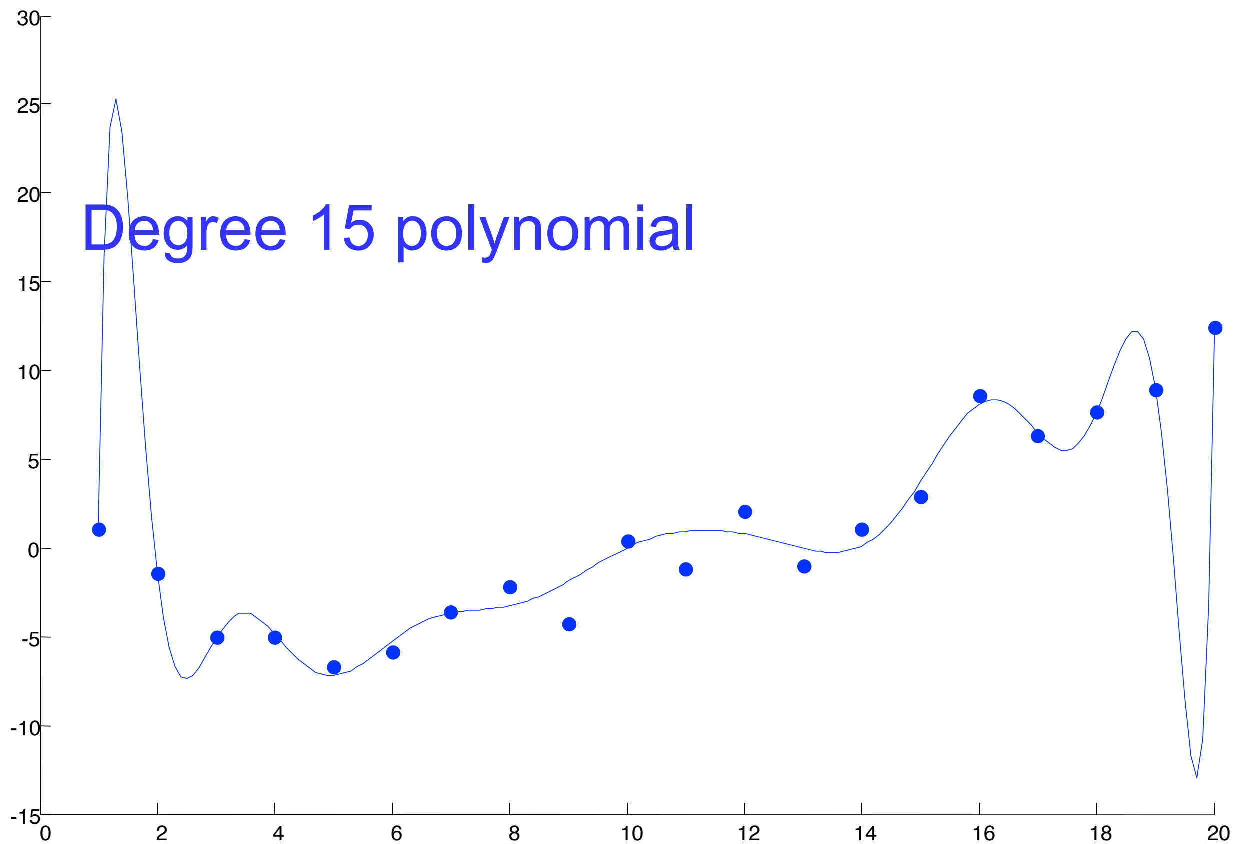
$$\min_{\theta \in \Theta} \sum_{i=1}^{m} (\hat{V}_\theta(s^{(i)}) - V(s^{(i)}))^2$$

Supervised Learning Example

- Linear regression



Overfitting



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?

[redacted]

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

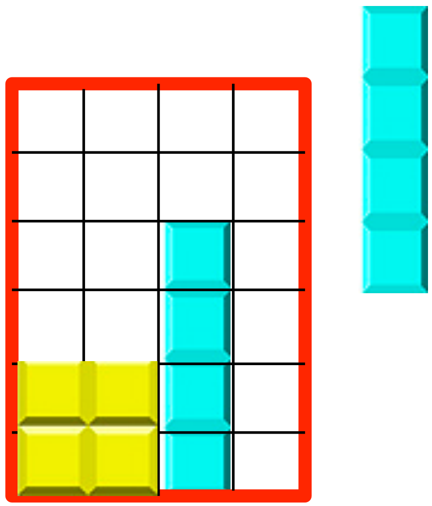
    $$\forall s \in S' : \quad \bar{V}_{i+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \hat{V}_{\theta^{(i)}}(s') \right]$$

  - Step 2: Supervised learning

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

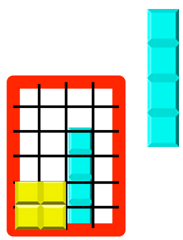
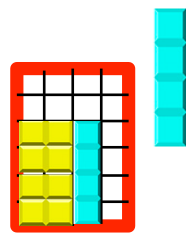
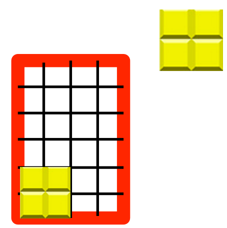
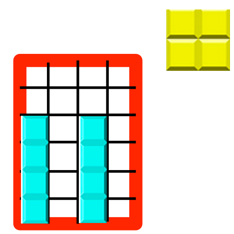
Value Iteration w/Function Approximation --- Example

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

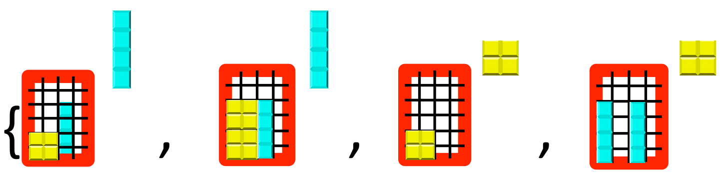


  - Reward = 1 for placing a block
  - Sink state / Game over is reached when block is placed such that part of it extends above the red rectangle
  - If you have a complete row, it gets cleared

Value Iteration w/Function Approximation --- Example

S’ = { , , ,  }

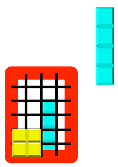
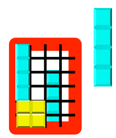
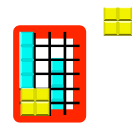
Value Iteration w/Function Approximation --- Example

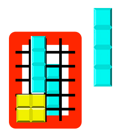
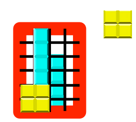
S’ = {  }

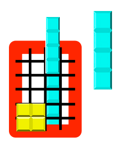
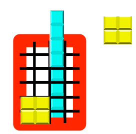
- 10 features (also called basis functions) φ_i
  - Four basis functions, 0*, . . . , 3, mapping the state to the height h[k] of each of the four* columns.
  - Three basis functions, 4*, . . . , 6, each mapping the state to the absolute difference* between heights of successive columns: $|h[k+1] - h[k]|$, $k = 1, . . . , 3$.
  - 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)

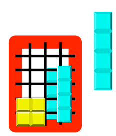
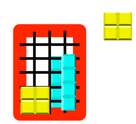
Value Iteration w/Function Approximation --- Example

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

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

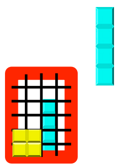
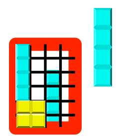
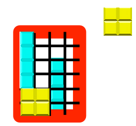
0.5 *(1+ γ V()) + 0.5*(1 + γ V()) ,

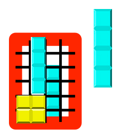
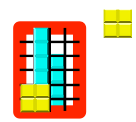
0.5 *(1+ γ V()) + 0.5*(1 + γ V()) ,

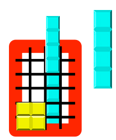
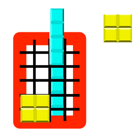
0.5 *(1+ γ V()) + 0.5*(1 + γ V()) }

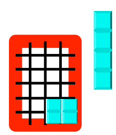
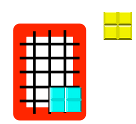
Value Iteration w/Function Approximation --- Example

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

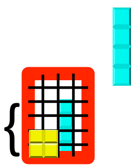
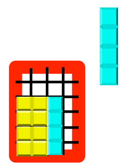
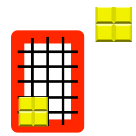
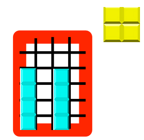
V() = max {0.5 *(1+ γ V()) + 0.5*(1 + γ V()) ,

0.5 *(1+ γ V()) + 0.5*(1 + γ V()) ,

0.5 *(1+ γ V()) + 0.5*(1 + γ V()) ,

0.5 *(1+ γ V()) + 0.5*(1 + γ V()) }

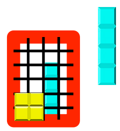
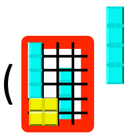
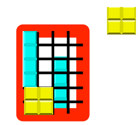
Value Iteration w/Function Approximation --- Example

S' = {  ,  ,  ,  }

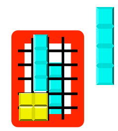
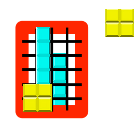
- 10 features aka basis functions φ_i
  - Four basis functions, 0*, . . . , 3, mapping the state to the height h[k] of each of the four* columns.
  - Three basis functions, 4*, . . . , 6, each mapping the state to the absolute difference* between heights of successive columns: $|h[k+1] - h[k]|$, $k = 1, . . . , 3$.
  - 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)

Value Iteration w/Function Approximation --- Example

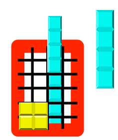
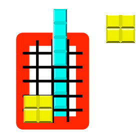
- Bellman back-ups for the states in S':

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

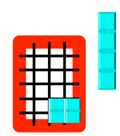
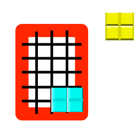
(6,2,4,0, 4, 2, 4, 6, 0, 1) (6,2,4,0, 4, 2, 4, 6, 0, 1)

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 ()) ,

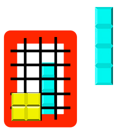
(sink-state, V=0) (sink-state, V=0)

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)

Value Iteration w/Function Approximation --- Example

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

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

$0.5 * (1 + \gamma\ (\quad -30 \quad)) + 0.5 * (1 + \gamma\ (\quad -30 \quad))$,

$0.5 * (1 + \gamma\ (\quad 0 \quad)) + 0.5 * (1 + \gamma\ (\quad 0 \quad))$,

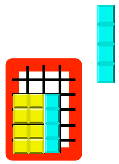
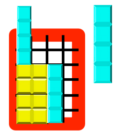
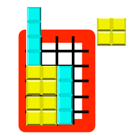
$0.5 * (1 + \gamma\ (\quad 6 \quad)) + 0.5 * (1 + \gamma\ (\quad 6 \quad))$ }

= 6.4 (for $\gamma = 0.9$)

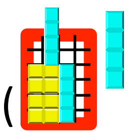
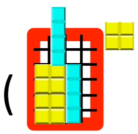
Value Iteration w/Function Approximation --- Example

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

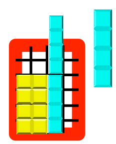
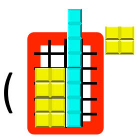
- Bellman back-ups for the second state in S':

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

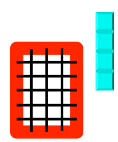
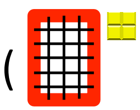
(sink-state, V=0) (sink-state, V=0)

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 $\theta^\top \phi$()) }

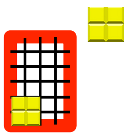
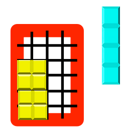
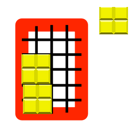
(0,0,0,0, 0,0,0, 0, 0, 1) -> V = 20 (0,0,0,0, 0,0,0, 0, 0, 1) -> V = 20

= 19

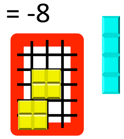
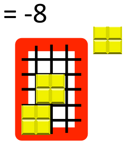
Value Iteration w/Function Approximation --- Example

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

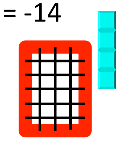
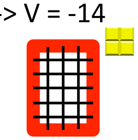
- Bellman back-ups for the third state in S':

V() = 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 $\theta^\top \phi$ ()) }

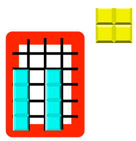
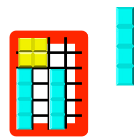
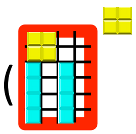
(0,0,0,0, 0,0,0, 0, 0, 1) -> V = 20   (0,0,0,0, 0,0,0, 0, 0, 1) -> V = 20

= 19

Value Iteration w/Function Approximation --- Example

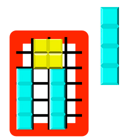
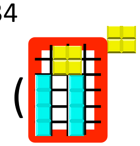
$$\theta^{(0)} = (-1, -1, -1, -1, -2, -2, -2, -3, -2, 20)$$

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

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

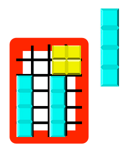
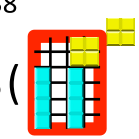
(6,6,4,0, 0,2,4, 6, 4, 1) (6,6,4,0, 0,2,4, 6, 4, 1)

-> V = -34 -> 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) (4,6.6,0, 2,0,6, 6, 4, 1)

-> V = -38 -> V = -38

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

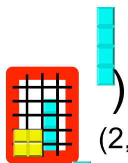
(4,0,6,6, 4,6,0, 6, 4, 1) (4,0,6,6, 4,6,0, 6, 4, 1)

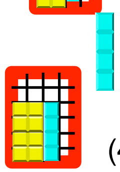
-> V = -42 -> V = -42

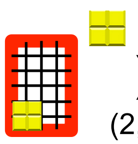
= -29.6

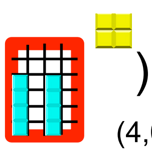
Value Iteration w/Function Approximation --- Example

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

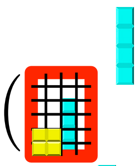
V()= 6.4
(2,2,4,0, 0,2,4, 4, 0, 1)

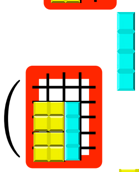
V()= 19
(4,4,4,0, 0,0,4, 4, 0, 1)

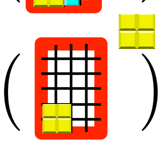
V()= 19
(2,2,0,0, 0,2,0, 2, 0, 1)

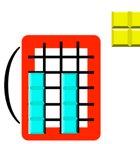
V()= -29.6
(4,0,4,0, 4,4,4, 4, 0, 1)

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

$\min_{\theta}(6.4 - \theta^{\top}\phi($  $))^2$

$+(19 - \theta^{\top}\phi($  $))^2$

$+(19 - \theta^{\top}\phi($  $))^2$

$+((-29.6) - \theta^{\top}\phi($  $))^2$

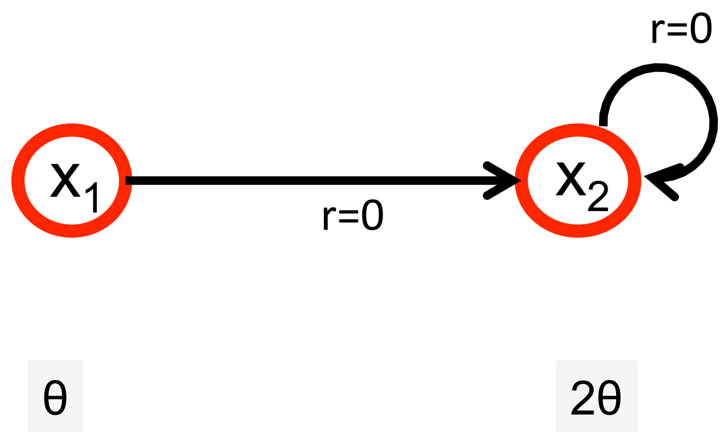
Running least squares gives:

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

Potential Guarantees?

[redacted]

Simple Example**



Function approximator: [1 2] * θ

Simple Example**

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

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

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 || . || if $\|GJ_1 - GJ_2\| \leq \|J_1 - J_2\|$

- **Fact.** If the operator F is a γ-contraction with respect to a norm || . || 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 $\Re^n$. Consider a particular entry s of ΠJ_1 and ΠJ_2:

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

This holds true for all s, hence we have

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

Linear Regression ☹ **

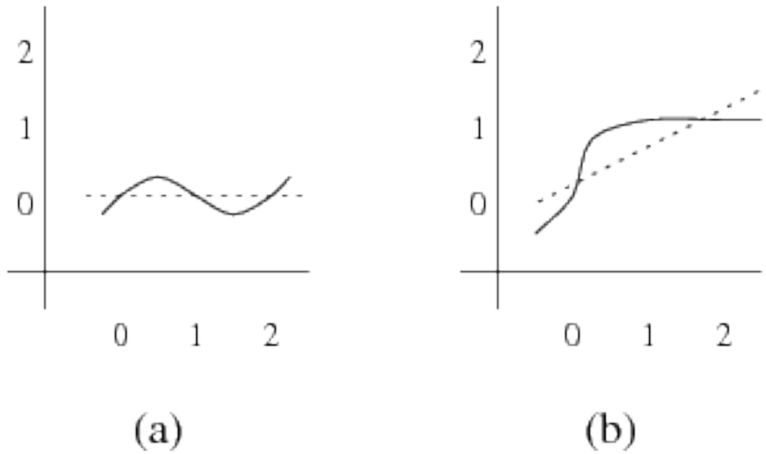


Figure 2: The mapping associated with linear regression when samples are taken at the points $x = 0, 1, 2$. In (a) we see a target value function (solid line) and its corresponding fitted value function (dotted line). In (b) we see another target function and another fitted function. The first target 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

Outline

- ✓ Function approximation
- ✓ Value iteration with function approximation
- Policy iteration with function approximation
- Linear programming with function approximation

Policy Iteration

One iteration of policy iteration:

- Policy evaluation: with fixed current policy π, find values with simplified Bellman updates:
  - Iterate until values converge

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

- Policy improvement: with fixed utilities, find the best action according to one-step look-ahead

$$\pi_{k+1}(s) = \arg\max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V^{\pi_k}(s')\right]$$

Insert Function Approximation Here

- Repeat until policy converges
- At convergence: optimal policy; and converges faster under some conditions

Policy Evaluation Revisited

- *Idea 1:* modify Bellman updates

$$V_0^\pi(s) = 0$$

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

Insert Function Approximation Here

- Idea 2: it is just a linear system, solve with Matlab (or whatever)

  variables: $V^\pi(s)$

  constants: T, R

$$\forall s \quad V^\pi(s) = \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V^\pi(s')]$$

Insert Function Approximation Here

And Here

Outline

- ✓ Function approximation
- ✓ Value iteration with function approximation
- ✓ Policy iteration with function approximation
- Linear programming with function approximation

[redacted]

Infinite Horizon Linear Program

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

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

Theorem. V^* is the solution to the above LP.

μ_0 is a probability distribution over S, with $\mu_0(s) > 0$ for all s in S.

Infinite Horizon Linear Program

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

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

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

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

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

We find approximate value function $\hat{V}_\theta(s) = \theta^\top \phi(s)$

Approximate Linear Program – Guarantees**

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

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

- 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)