Lecture 2: Sampling-based Approximations And Function Fitting

Yan (Rocky) Duan
Berkeley AI Research Lab

Many slides made with John Schulman, Xi (Peter) Chen and Pieter Abbeel

Quick One-Slide Recap

- Optimal Control

  =

  given an MDP (S, A, P, R, γ, H)

  find the optimal policy π^*

- Exact Methods:
  - ✓ ***Value Iteration***
  - ✓ ***Policy Iteration***

Limitations:

- Update equations require access to dynamics model
- Iteration over / Storage for all states and actions: requires small, discrete state-action space

-> sampling-based approximations

-> Q/V function fitting

Sampling-Based Approximation

- Q Value Iteration
- Value Iteration?
- Policy Iteration
  - Policy Evaluation
  - Policy Improvement?

Recap Q-Values

$Q^*(s, a)$ = expected utility starting in s, taking action a, and (thereafter) acting optimally

Bellman Equation:

$$Q^*(s,a) = \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q^*(s',a'))$$

Q-Value Iteration:

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q_k(s',a'))$$

(Tabular) Q-Learning

- Q-value iteration: $Q_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q_k(s',a'))$
- Rewrite as expectation: $Q_{k+1} \leftarrow \mathbb{E}_{s' \sim P(s'|s,a)} \left[R(s,a,s') + \gamma \max_{a'} Q_k(s',a') \right]$
- (Tabular) Q-Learning: replace expectation by samples
  - For an state-action pair (s,a), receive: $s' \sim P(s'|s,a)$
  - Consider your old estimate: $Q_k(s,a)$
  - Consider your new sample estimate:

$$\text{target}(s') = R(s,a,s') + \gamma \max_{a'} Q_k(s',a')$$

  - Incorporate the new estimate into a running average:

$$Q_{k+1}(s,a) \leftarrow (1-\alpha) Q_k(s,a) + \alpha \left[\text{target}(s')\right]$$

(Tabular) Q-Learning

Algorithm:

Start with $Q_0(s, a)$ for all s, a.

Get initial state s

For k = 1, 2, … till convergence

Sample action a, get next state s’

If s’ is terminal:

$$\text{target} = R(s, a, s')$$

Sample new initial state s’

else:

$$\text{target} = R(s, a, s') + \gamma \max_{a'} Q_k(s', a')$$

$$Q_{k+1}(s, a) \leftarrow (1 - \alpha) Q_k(s, a) + \alpha \left[\text{target}\right]$$

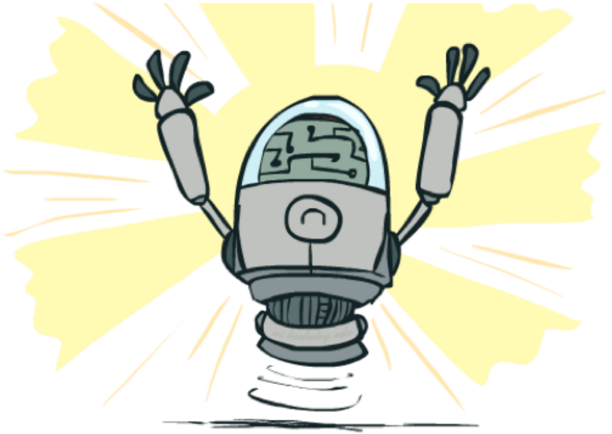
$$s \leftarrow s'$$

How to sample actions?

- Choose random actions?
- Choose action that maximizes $Q_k(s, a)$ (i.e. greedily)?
- ε-Greedy: choose random action with prob. ε, otherwise choose action greedily

Q-Learning Properties

- Amazing result: Q-learning converges to optimal policy -- even if you're acting suboptimally!
- This is called off-policy learning
- Caveats:
  - You have to explore enough
  - You have to eventually make the learning rate small enough
  - … but not decrease it too quickly



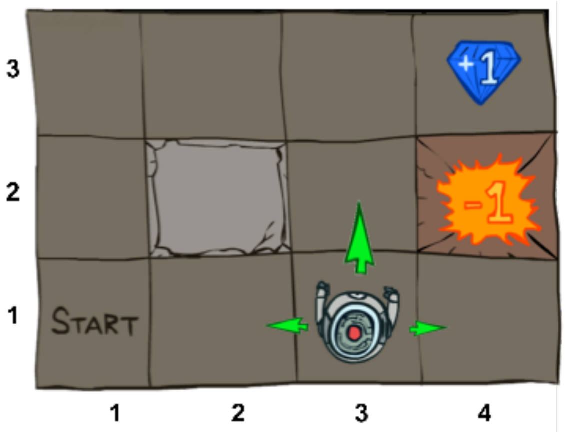
Q-Learning Properties

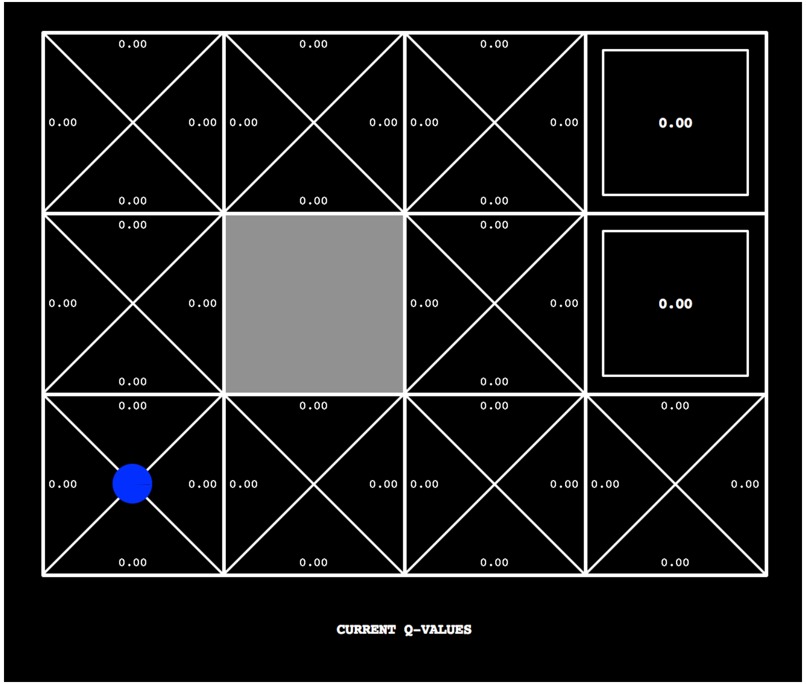
- Technical requirements.
  - All states and actions are visited infinitely often
    - Basically, in the limit, it doesn't matter how you select actions (!)
  - Learning rate schedule such that for all state and action pairs (s,a):

$$\sum_{t=0}^{\infty} \alpha_t(s,a) = \infty \qquad \sum_{t=0}^{\infty} \alpha_t^2(s,a) < \infty$$

For details, see Tommi Jaakkola, Michael I. Jordan, and Satinder P. Singh. On the convergence of stochastic iterative dynamic programming algorithms. Neural Computation, 6(6), November 1994.

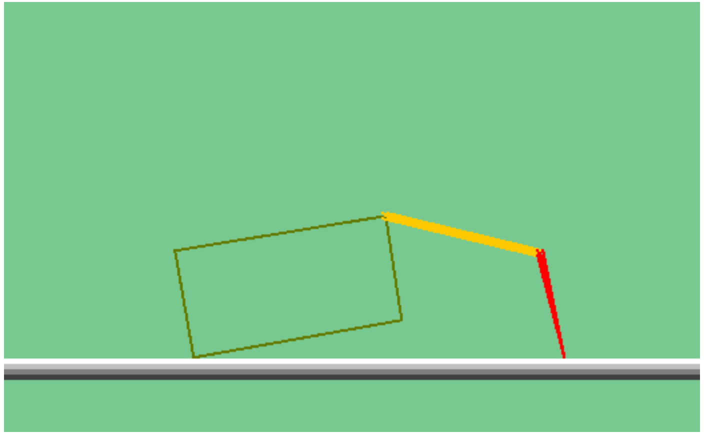
Q-Learning Demo: Gridworld





- States: 11 cells
- Actions: {up, down, left, right}
- Deterministic transition function
- Learning rate: 0.5
- Discount: 1
- Reward: +1 for getting diamond, -1 for falling into trap

Q-Learning Demo: Crawler



- States: discretized value of 2d state: (arm angle, hand angle)
- Actions: Cartesian product of {arm up, arm down} and {hand up, hand down}
- Reward: speed in the forward direction

Sampling-Based Approximation

- Q Value Iteration → (Tabular) Q-learning
- Value Iteration?
- Policy Iteration
  - Policy Evaluation
  - Policy Improvement?

Value Iteration w/ Samples?

- Value Iteration

$$V_{i+1}^*(s) \leftarrow \max_a \mathbb{E}_{s' \sim P(s'|s,a)} \left[R(s,a,s') + \gamma V_i^*(s') \right]$$

- unclear how to draw samples through max......

Sampling-Based Approximation

- Q Value Iteration → (Tabular) Q-learning
- ~~Value Iteration?~~
- Policy Iteration
  - Policy Evaluation
  - Policy Improvement?

Recap: Policy Iteration

One iteration of policy iteration:

- Policy evaluation for current policy π_k :
  - Iterate until convergence

$$V_{i+1}^{\pi_k}(s) \leftarrow \mathbb{E}_{s' \sim P(s'|s,\pi_k(s))}[R(s,\pi_k(s),s') + \gamma V_i^{\pi_k}(s')]$$

Can be approximated by samples
This is called Temporal Difference (TD) Learning

- Policy improvement: find the best action according to one-step look-ahead

$$\pi_{k+1}(s) \leftarrow \arg\max_a \mathbb{E}_{s' \sim P(s'|s,a)}[R(s,a,s') + \gamma V^{\pi_k}(s')]$$

Unclear what to do with the max (for now)

Sampling-Based Approximation

- ✓ Q Value Iteration → (Tabular) Q-learning
- ~~Value Iteration?~~
- Policy Iteration
  - ✓ Policy Evaluation → (Tabular) TD-learning
  - ~~Policy Improvement (for now)~~

Quick One-Slide Recap

- Optimal Control

  =

  given an MDP (S, A, P, R, γ, H)

  find the optimal policy π^*

- Exact Methods:
  - ✓ ***Value Iteration***
  - ✓ ***Policy Iteration***

Limitations:

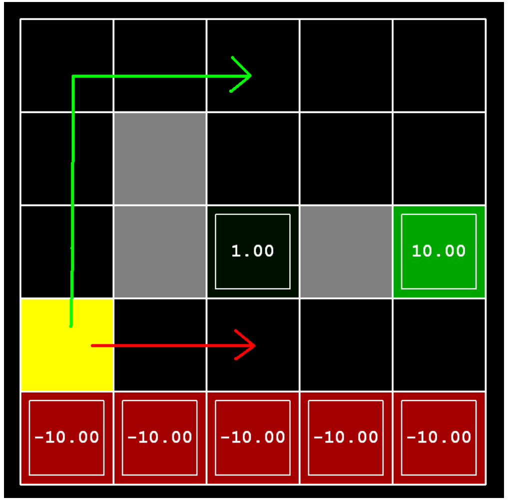
- Update equations require access to dynamics model
- Iteration over / Storage for all states and actions: requires small, discrete state-action space

✓ **sampling-based approximations**

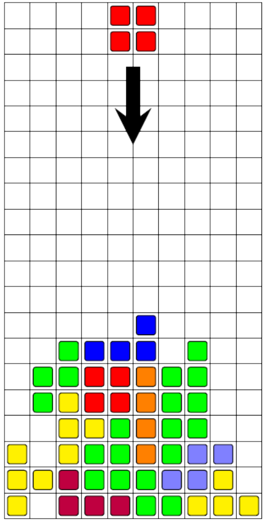
-> Q/V function fitting

Can tabular methods scale?

- Discrete environments



Gridworld
10^1

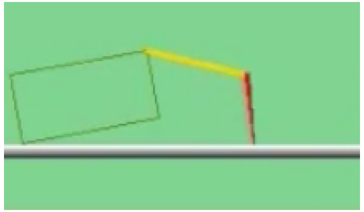


Tetris
10^60

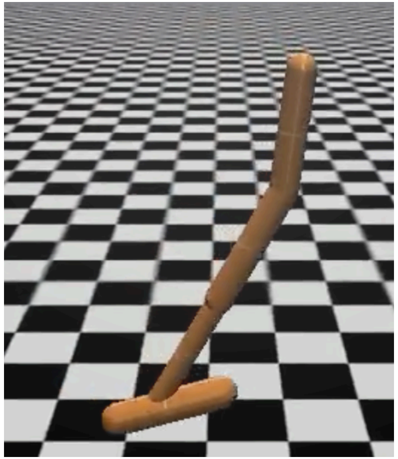
Atari
10^308 (ram) 10^16992 (pixels)

Can tabular methods scale?

- Continuous environments (by crude discretization)



Crawler
10^2



Hopper
10^10

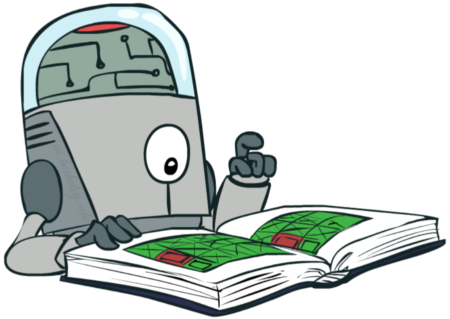
Humanoid
10^100

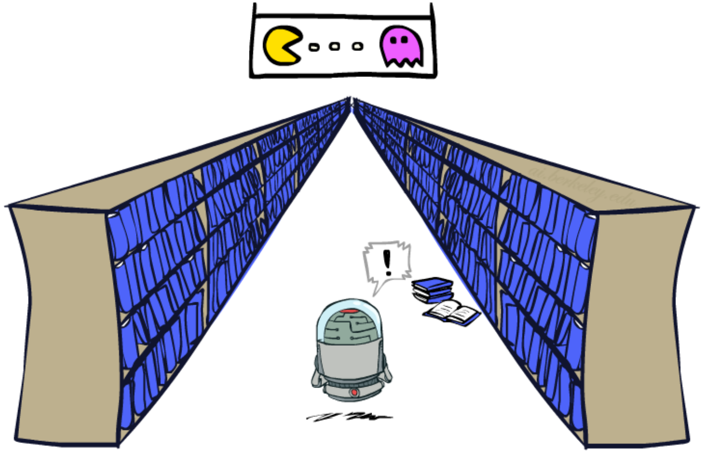
Generalizing Across States

- Basic Q-Learning keeps a table of all q-values

- In realistic situations, we cannot possibly learn about every single state!
  - Too many states to visit them all in training
  - Too many states to hold the q-tables in memory

- Instead, we want to generalize:
  - Learn about some small number of training states from experience
  - Generalize that experience to new, similar situations
  - This is a fundamental idea in machine learning, and we'll see it over and over again





Approximate Q-Learning

- Instead of a table, we have a parametrized Q function: $Q_\theta(s,a)$
  - Can be a linear function in features:

$$Q_\theta(s,a) = \theta_0 f_0(s,a) + \theta_1 f_1(s,a) + \cdots + \theta_n f_n(s,a)$$

  - Or a complicated neural net

- Learning rule:
  - Remember: $\text{target}(s') = R(s,a,s') + \gamma \max_{a'} Q_{\theta_k}(s',a')$
  - Update:

$$\theta_{k+1} \leftarrow \theta_k - \alpha \nabla_\theta \left[\frac{1}{2} (Q_\theta(s,a) - \text{target}(s'))^2 \right] \bigg|_{\theta=\theta_k}$$

Connection to Tabular Q-Learning

- Suppose $\theta \in \mathbb{R}^{|S|\times|A|}, \quad Q_\theta(s,a) \equiv \theta_{sa}$

$$\nabla_{\theta_{sa}}\left[\frac{1}{2}(Q_\theta(s,a)-\text{target}(s'))^2\right]$$
$$=\nabla_{\theta_{sa}}\left[\frac{1}{2}(\theta_{sa}-\text{target}(s'))^2\right]$$
$$=\theta_{sa}-\text{target}(s')$$

- Plug into update: $\theta_{sa} \leftarrow \theta_{sa} - \alpha(\theta_{sa} - \text{target}(s'))$

$$= (1-\alpha)\theta_{sa} + \alpha[\text{target}(s')]$$

- Compare with Tabular Q-Learning update:

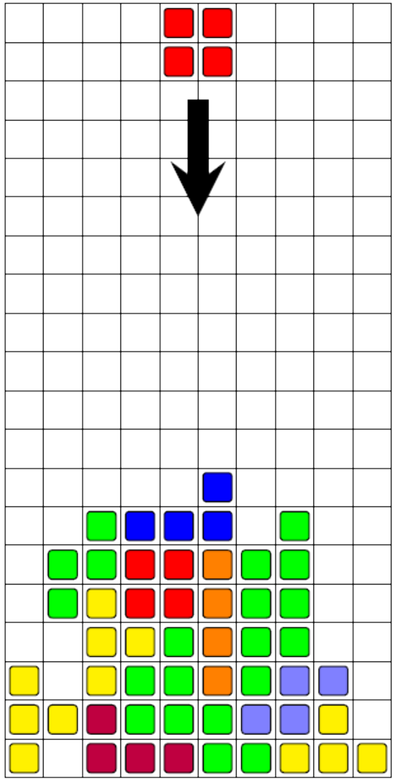
$$Q_{k+1}(s,a) \leftarrow (1-\alpha)Q_k(s,a) + \alpha\left[\text{target}(s')\right]$$

Engineered Approximation Example: Tetris

- state: naïve board configuration + shape of the falling piece ~10^{60} states!
- action: rotation and translation applied to the falling piece

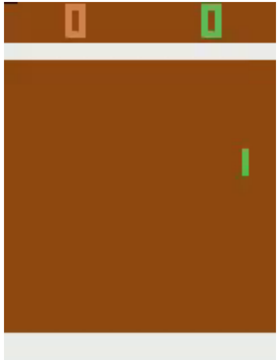
- 22 features aka basis functions ϕ_i
  - Ten basis functions, 0, *. . . , 9, mapping the state to the height h[k] of each* column.
  - Nine basis functions, 10, *. . . , 18, each mapping the state to the absolute difference* between heights of successive columns: $|h[k+1] - h[k]|$, $k = 1, \ldots, 9$.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

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

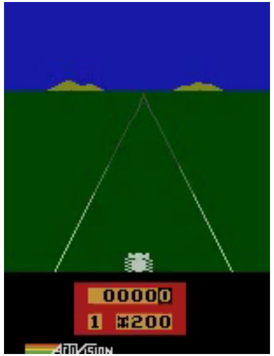


[Bertsekas & Ioffe, 1996 (TD); Bertsekas & Tsitsiklis 1996 (TD); Kakade 2002 (policy gradient); Farias & Van Roy, 2006 (approximate LP)]

Deep Reinforcement Learning

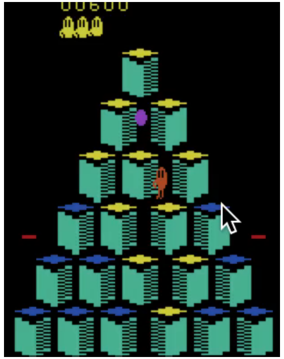


Pong



Enduro

Beamrider



Q*bert

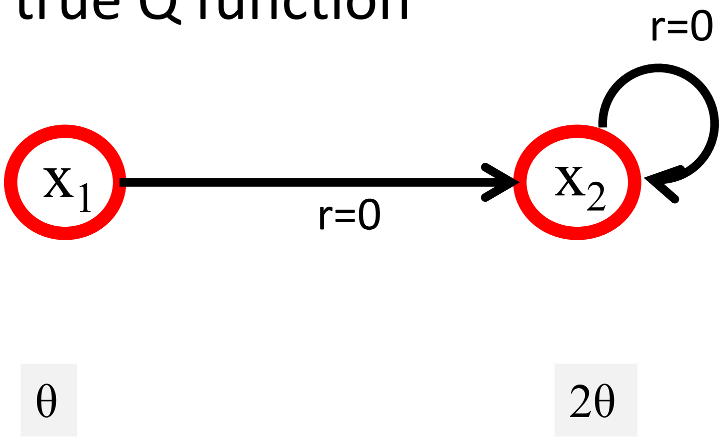
- From pixels to actions
- Same algorithm (with effective tricks)
- CNN function approximator, w/ 3M free parameters

Lab 1

- We have now covered enough materials for Lab 1.
- Will be released on Piazza by this afternoon.
- Covers value iteration, policy iteration, and tabular Q-learning.

Convergence of Approximate Q-Learning

- The bad: it is not guaranteed to converge…
  - Even if the function approximation is expressive enough to represent the true Q function



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 ☹ **

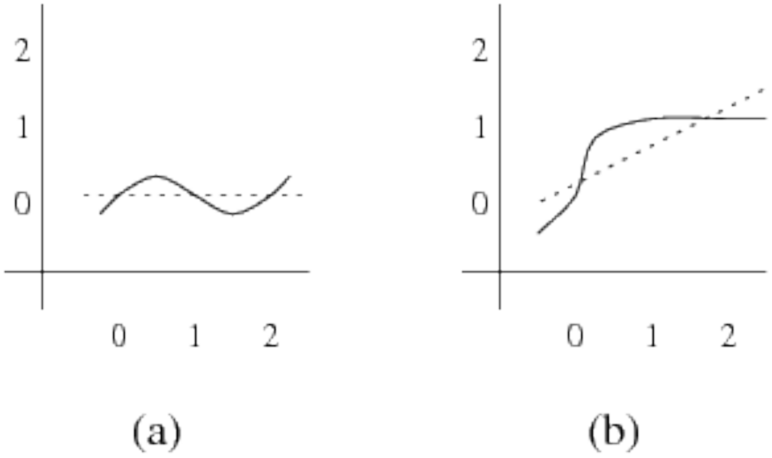


Figure 2: The mapping associated with linear regression when samples are taken at the points $x = 0, 1, 2$. In (a) we see a target value function (solid line) and its corresponding fitted value function (dotted line). In (b) we see another target function and another fitted function. The first target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