



Approximate Policy Iteration

Emma Brunskill

Many thanks to Alan Fern for the majority of the LSPI slides.

<https://web.engr.oregonstate.edu/~afern/classes/cs533/notes/lspi.pdf>

Fitted Value Iteration: Treatment Decision Making

The CATIE study: a sequentially randomized trial

- The Clinical Antipsychotic Trials of Intervention and Effectiveness (CATIE) study was an 18 month sequential, multiple assignment, randomized trial of 1460 patients with schizophrenia.
- CATIE was a practical clinical trial, thus had a broad entry criteria and a protocol designed to mimic real life.
Specifically, participants choose *when* to switch treatment
- Two major treatment phases, with monthly follow-ups from baseline
- Two measures of symptoms collected during CATIE
 1. Positive and Negative Syndrome Scale (PANSS) total score
 2. Quality of Life (QOL) scale
- We distinguish between two different types of variables in CATIE:
Scheduled collected on everyone at pre-specified visits
end-of-phase collected when a patient entered a new treatment phase

Informing sequential clinical decision-making through reinforcement learning: an empirical study
Shortreed et al. *Machine Learning Journal* 2011



Patient State Variables

Variables with no missing information.

Time independent variables:

Age (continuous), Sex (binary), Race (categorical), Tardive dyskinesia status at baseline (binary), Marital status (binary), Patient education (categorical), Hospitalization history in 3 months prior to CATIE (binary), Clinical setting at which patient received CATIE treatment (categorical), Treatment prior to CATIE enrollment (categorical), Phase 1 treatment assignment (categorical), Time in study on phase 1 treatment assignment (continuous).

Variables with missing information.

Time independent variables:

Employment status (categorical), Years since first prescribed anti-psychotic medication at baseline (continuous), Neurocognitive composite score at baseline (continuous), Phase 2 treatment assignment (categorical), Phase 2 randomization arm (binary), Reason for discontinuing phase 1 and 2 (categorical), Reason for discontinuing the CATIE study early (categorical), Total time spent in the CATIE study (continuous).

Variables collected at months 1-18 and at end-of-phase:

Treatment adherence, the proportion of capsules taken since last visit (continuous)

Variables collected at months 0, 1, 3, 6, 9, 12, 15, 18 and at end-of-phase:

Body mass index (continuous), Clinical drug use scale (ordinal), Clinical alcohol use scale (ordinal), Clinical Global Impressions Severity of illness score (ordinal), Positive and Negative Syndrome Scale total score (continuous), Calgary Depression total Score (continuous), Simpson-Angus EP mean scale (continuous), Barnes Akathisia scale (continuous), Total movement severity score (continuous)

Variables collected at months 0, 6, 12, 18 and at end-of-phase:

Quality of Life total score (continuous), SF-12 Mental health summary (continuous), SF-12 Physical health summary (continuous), Illicit drug use (binary)



Used Linear FQI

$$Q_t(s_t, a_t; \beta_t) = \beta_t^\top x(s_t, a_t) = \sum_{k=1}^{|\mathcal{A}_t|} \beta_{t,k}^\top s_t \mathbb{1}_{a_t=k}.$$

Challenges

- Missing data for patients
- How much better is the best computed policy from any other? Can we trust Q estimates?



Fitted V/Q Iteration Summary

- Model free, value function based technique
- Performs supervised learning problem at each iteration to fit the V/Q function
- Due to iterative nature, error can compound (worse than linear)
- Input data sampling distribution has a crucial impact
- Poor performance (divergence) possible, choice of function class/regressor important (for convergence and accuracy of resulting estimate)



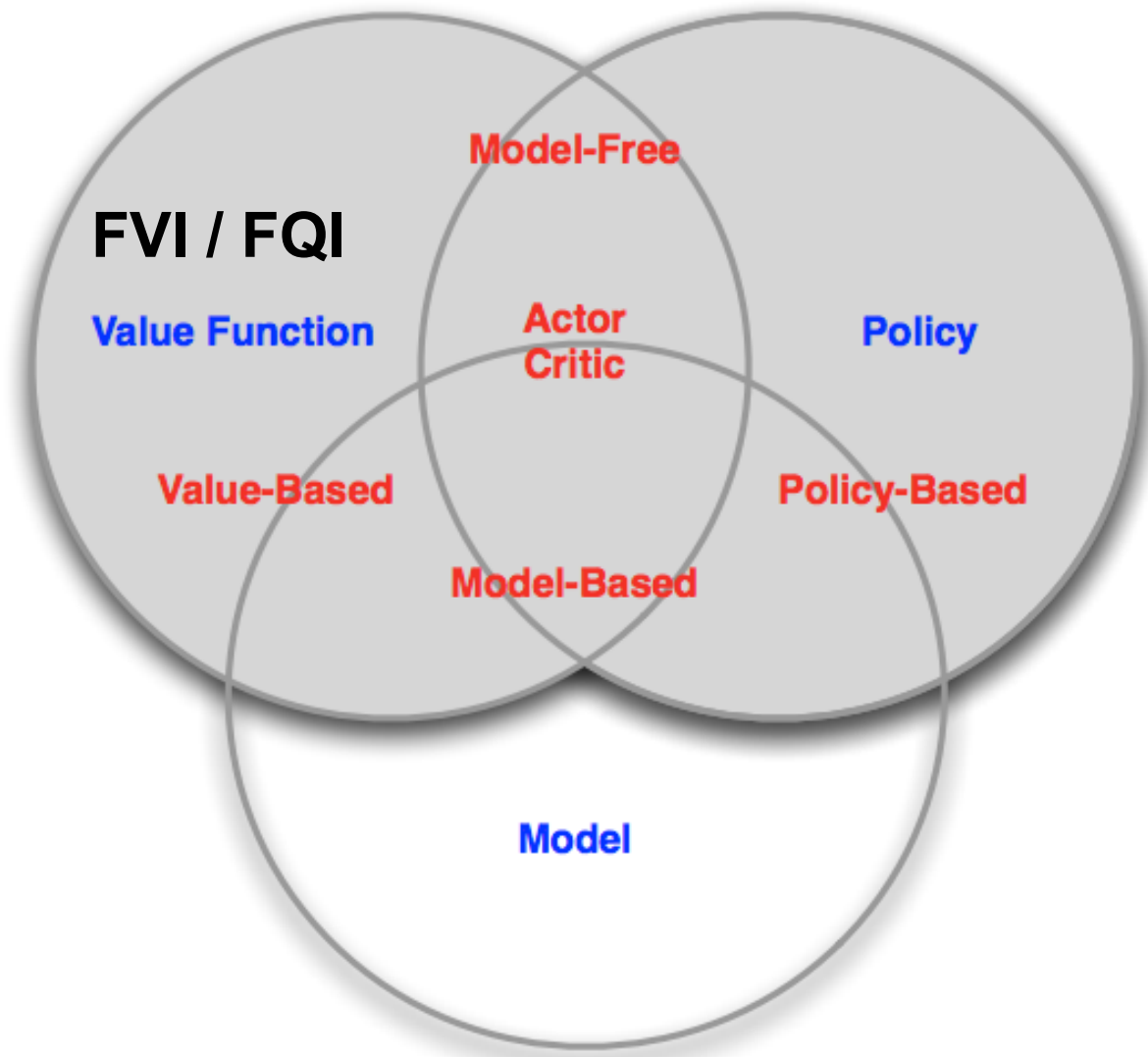
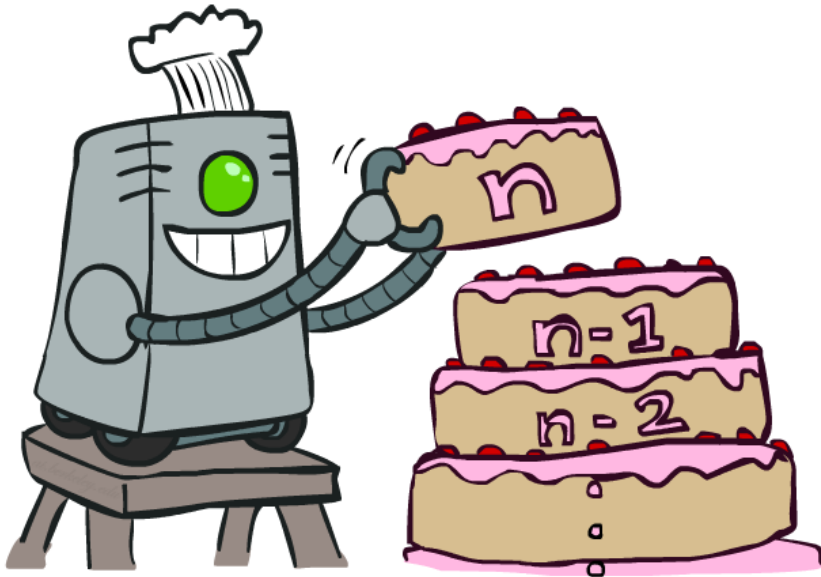


Image from David Silver

Carnegie Mellon University

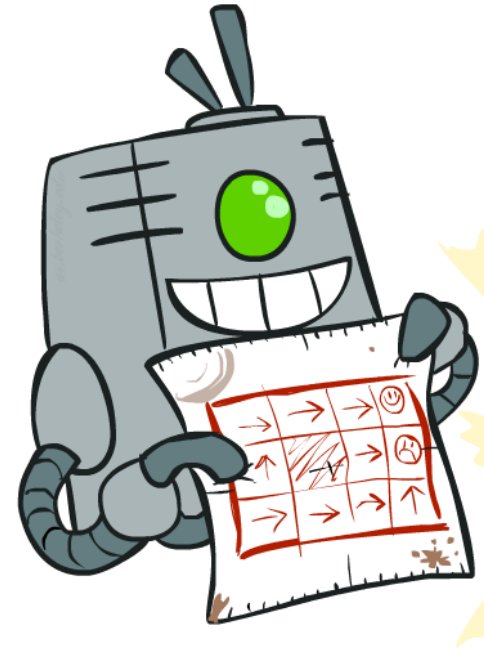
Value Iteration

Maintain optimal values if
have n more actions



Policy Iteration

Maintain value of following a
particular policy forever



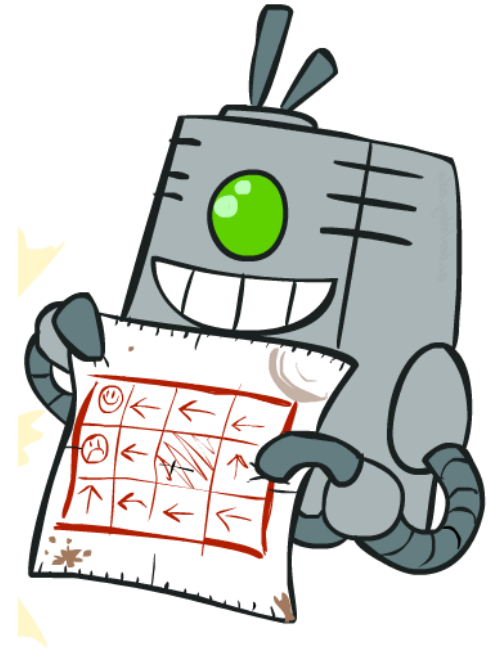
Drawings by Ketrina Yim



Policy Iteration for Infinite Horizon

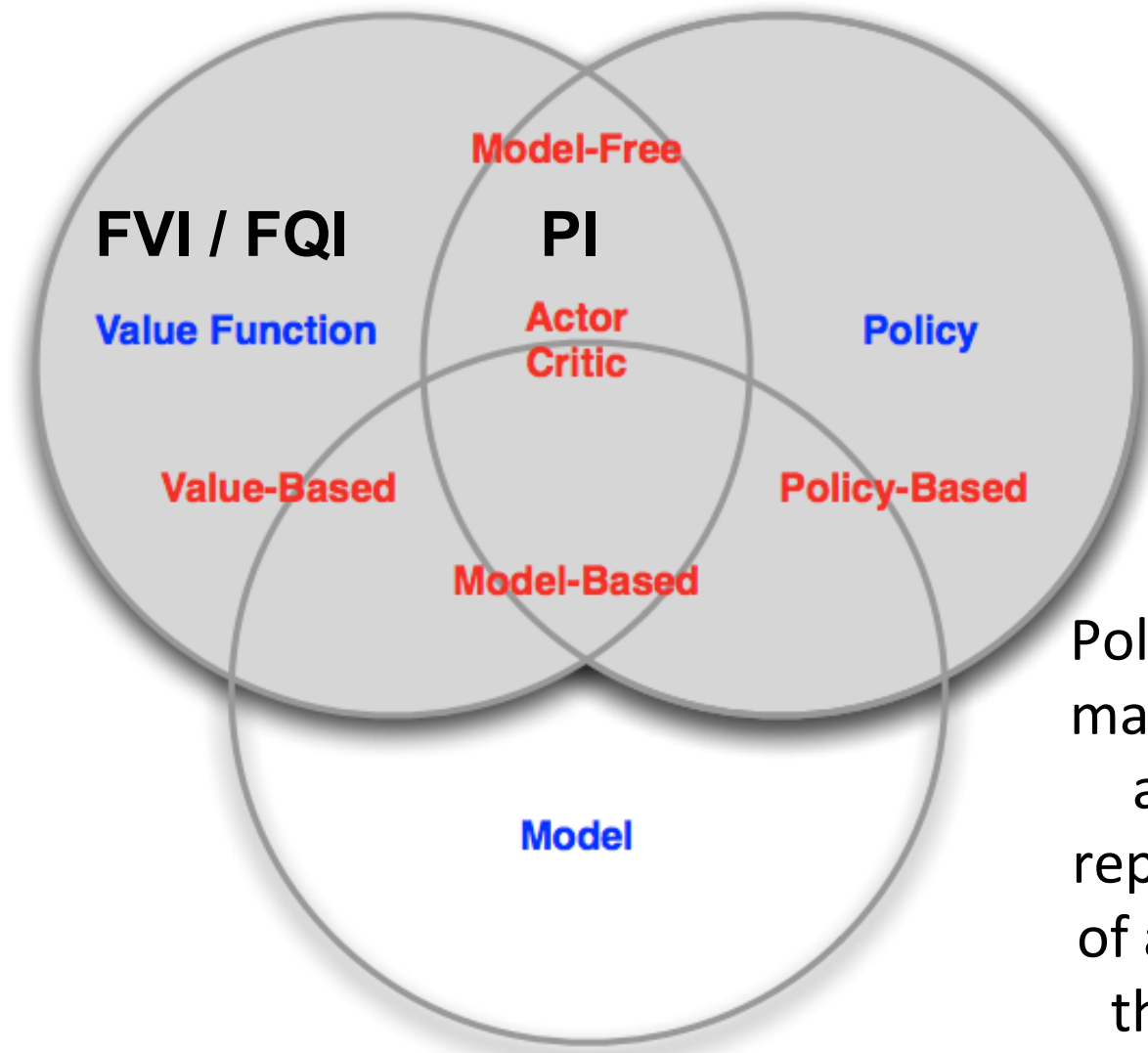
Given MDP

1. **Policy Evaluation:** Calculate exact value of acting in infinite horizon for a particular policy
2. **Improve policy**
3. Repeat 1 & 2 until policy doesn't change



Drawing by Ketrina Yim





Policy Iteration maintains both an explicit representation of a policy and the value of that policy

Image from David Silver

Carnegie Mellon University

No max in Bellman eqn so linear set of equations... Analytic Solution!

$$V^\pi(s) = \sum_{s' \in S} p(s' | s, \pi(s)) [R(s, \pi(s), s') + \gamma V^\pi(s')]$$

Requires taking an inverse of a S by S matrix: $O(S^3)$



Policy Improvement

- Have $V^\pi(s)$ for all s
- First compute

$$Q^\pi(s, a) = \sum_{s' \in S} p(s' | s, a) [R(s, a, s') + \gamma V^\pi(s')]$$

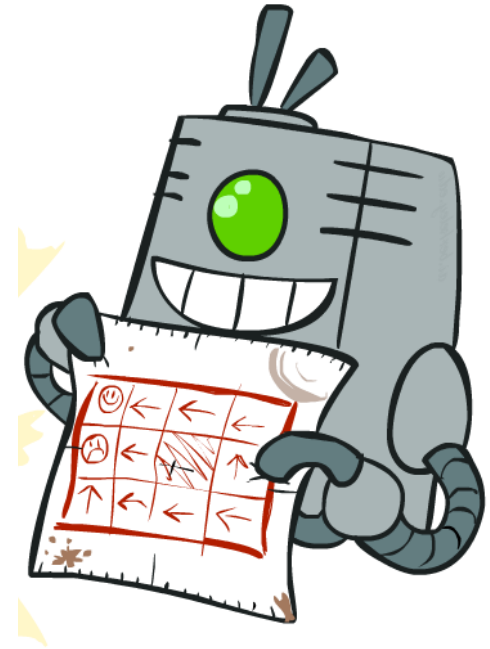
- Then extract new policy.
For each s ,

$$\pi'(s) = \arg \max_a Q^\pi(s, a)$$



Policy Iteration for Infinite Horizon

1. **Policy Evaluation:** Calculate exact value of acting in infinite horizon for a particular policy
2. **Policy Improvement**
3. Repeat 1 & 2 until policy doesn't change

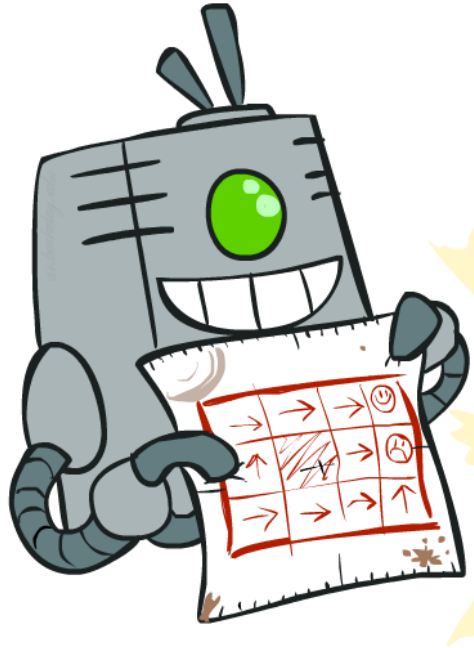


Drawing by Ketrina Yim



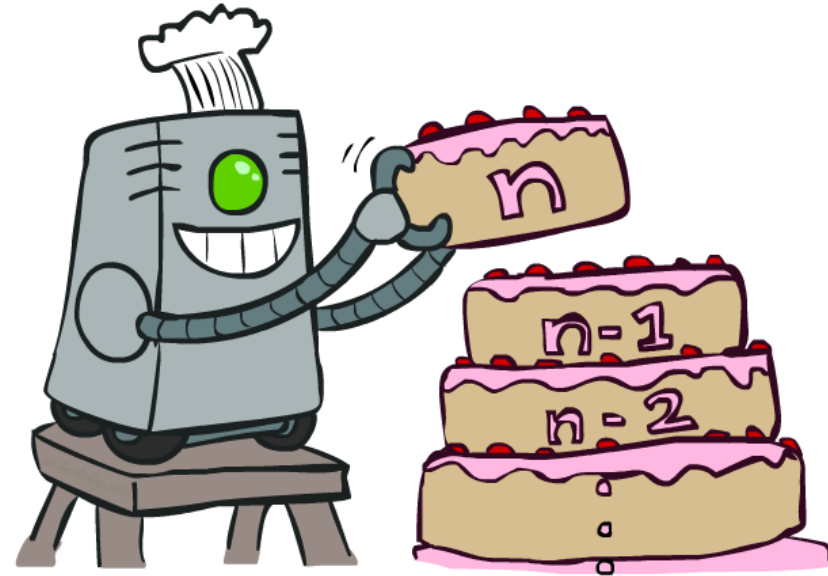
Policy Iteration

Maintain value of policy
Improve policy



Value Iteration

Keep optimal value for
finite steps, increase steps

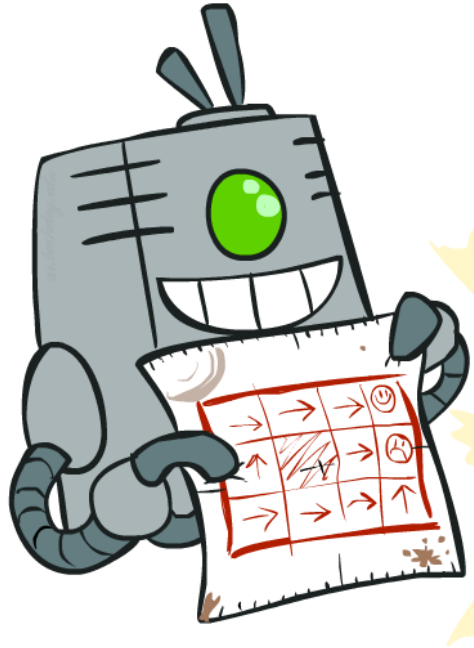


Drawings by Ketrina Yim

Policy Iteration

Fewer Iterations

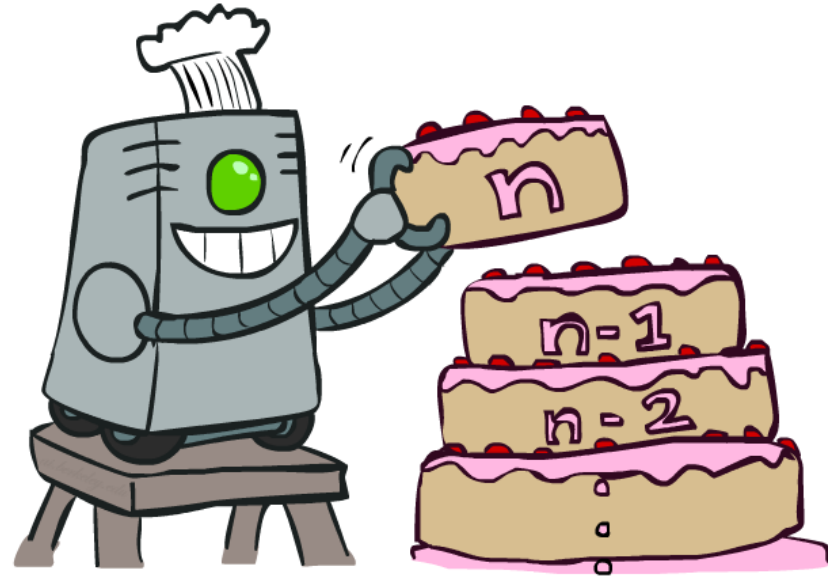
More expensive per iteration



Value Iteration

More iterations

Cheaper per iteration



Drawings by Ketrina Yim

Back to Offline, Batch Setting

- Don't know MDP transition or reward models
- Have a set of samples
- Huge or infinite state space
- Want to compute good policy to use in the future



Approximate Policy Iteration

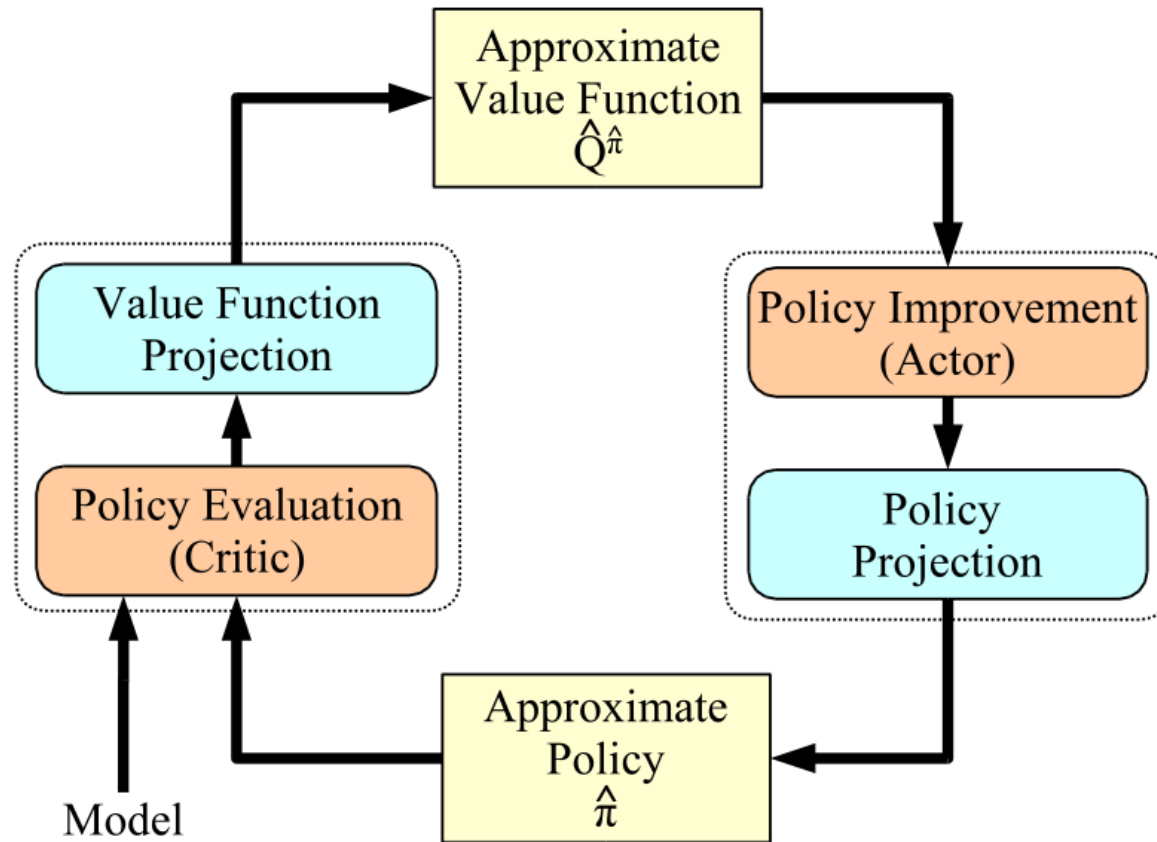


Figure modified from
Lagoudakis & Parr 2003

API Guarantees

(Bertsekas & Tsitsiklis; Lagoudakis & Parr)

Theorem 3.1 *Let $\hat{\pi}_0, \hat{\pi}_1, \hat{\pi}_2, \dots, \hat{\pi}_m$ be the sequence of policies generated by an approximate policy-iteration algorithm and let $\hat{Q}^{\hat{\pi}_0}, \hat{Q}^{\hat{\pi}_1}, \hat{Q}^{\hat{\pi}_2}, \dots, \hat{Q}^{\hat{\pi}_m}$ be the corresponding approximate value functions. Let ϵ and δ be positive scalars that bound the error in all approximations (over all iterations) to value functions and policies respectively. If*

$$\forall m = 0, 1, 2, \dots, \|\hat{Q}^{\hat{\pi}_m} - Q^{\hat{\pi}_m}\|_{\infty} \leq \epsilon, \quad \text{error in policy evaluation}$$

and ³

$$\forall m = 0, 1, 2, \dots, \|T_{\hat{\pi}_{m+1}}\hat{Q}^{\hat{\pi}_m} - T_*\hat{Q}^{\hat{\pi}_m}\|_{\infty} \leq \delta. \quad \text{error in policy improvement}$$

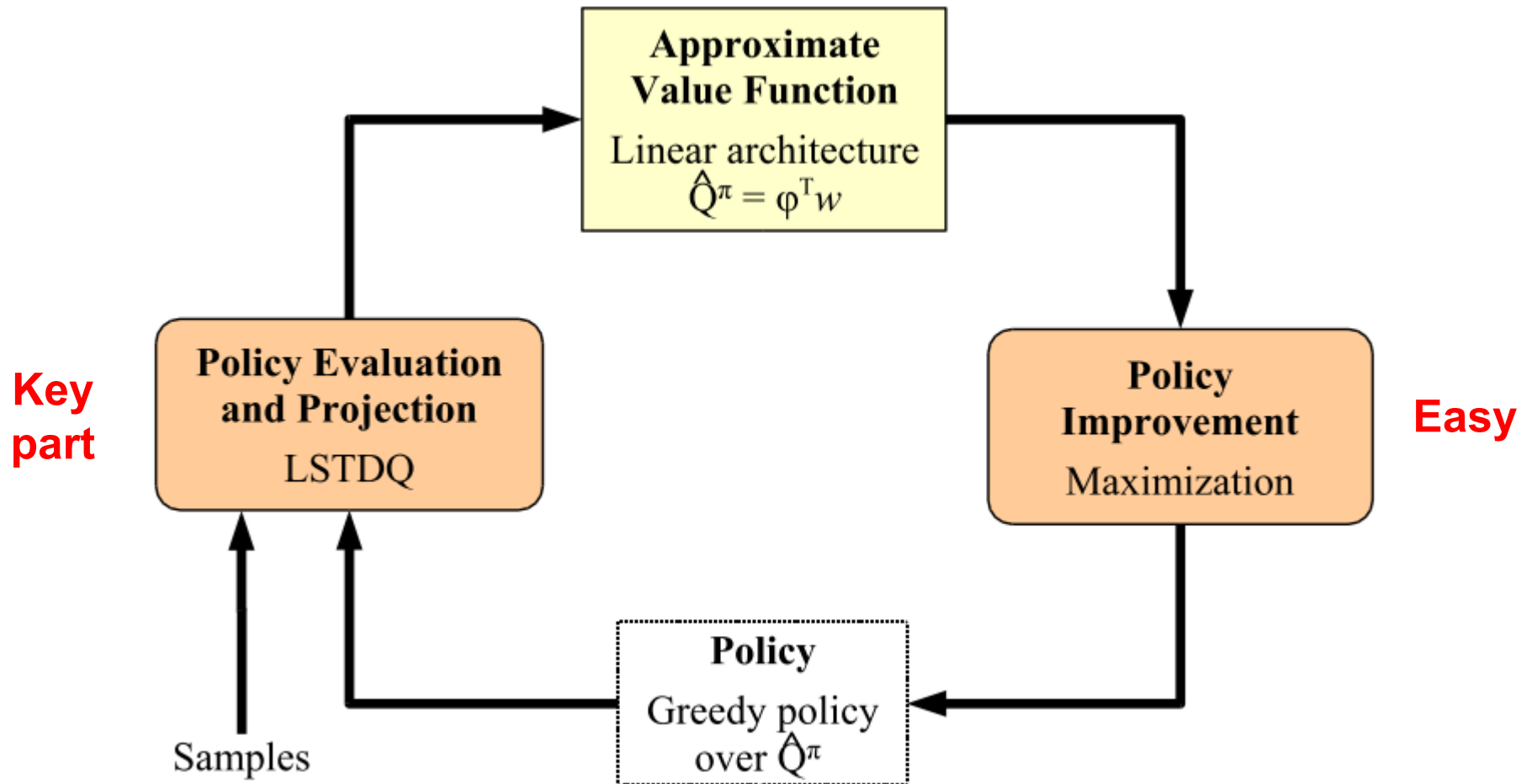
Then, this sequence eventually produces policies whose performance is at most a constant multiple of ϵ and δ away from the optimal performance:

$$\limsup_{m \rightarrow \infty} \|\hat{Q}^{\hat{\pi}_m} - Q^*\|_{\infty} \leq \frac{\delta + 2\gamma\epsilon}{(1 - \gamma)^2}.$$



Least Squares Policy Iteration

Lagoudakis & Parr (2003)



LSPI Guarantees

Theorem 7.1 *Let $\pi_0, \pi_1, \pi_2, \dots, \pi_m$ be the sequence of policies generated by LSPI and let $\hat{Q}^{\pi_1}, \hat{Q}^{\pi_2}, \dots, \hat{Q}^{\pi_m}$ be the corresponding approximate value functions as computed by LSTDQ. Let ϵ be a positive scalar that bounds the errors between the approximate and the true value functions over all iterations:*

$$\forall m = 1, 2, \dots, \|\hat{Q}^{\pi_m} - Q^{\pi_m}\|_{\infty} \leq \epsilon .$$

Then, this sequence eventually produces policies whose performance is at most a constant multiple of ϵ away from the optimal performance:

$$\limsup_{m \rightarrow \infty} \|\hat{Q}^{\pi_m} - Q^*\|_{\infty} \leq \frac{2\gamma\epsilon}{(1-\gamma)^2} .$$



LSPI Guarantees

Theorem 7.1 *Let $\pi_0, \pi_1, \pi_2, \dots, \pi_m$ be the sequence of policies generated by LSPI and let $\hat{Q}^{\pi_1}, \hat{Q}^{\pi_2}, \dots, \hat{Q}^{\pi_m}$ be the corresponding approximate value functions as computed by LSTDQ. Let ϵ be a positive scalar that bounds the errors between the approximate and the true value functions over all iterations:*

$$\forall m = 1, 2, \dots, \|\hat{Q}^{\pi_m} - Q^{\pi_m}\|_{\infty} \leq \epsilon .$$

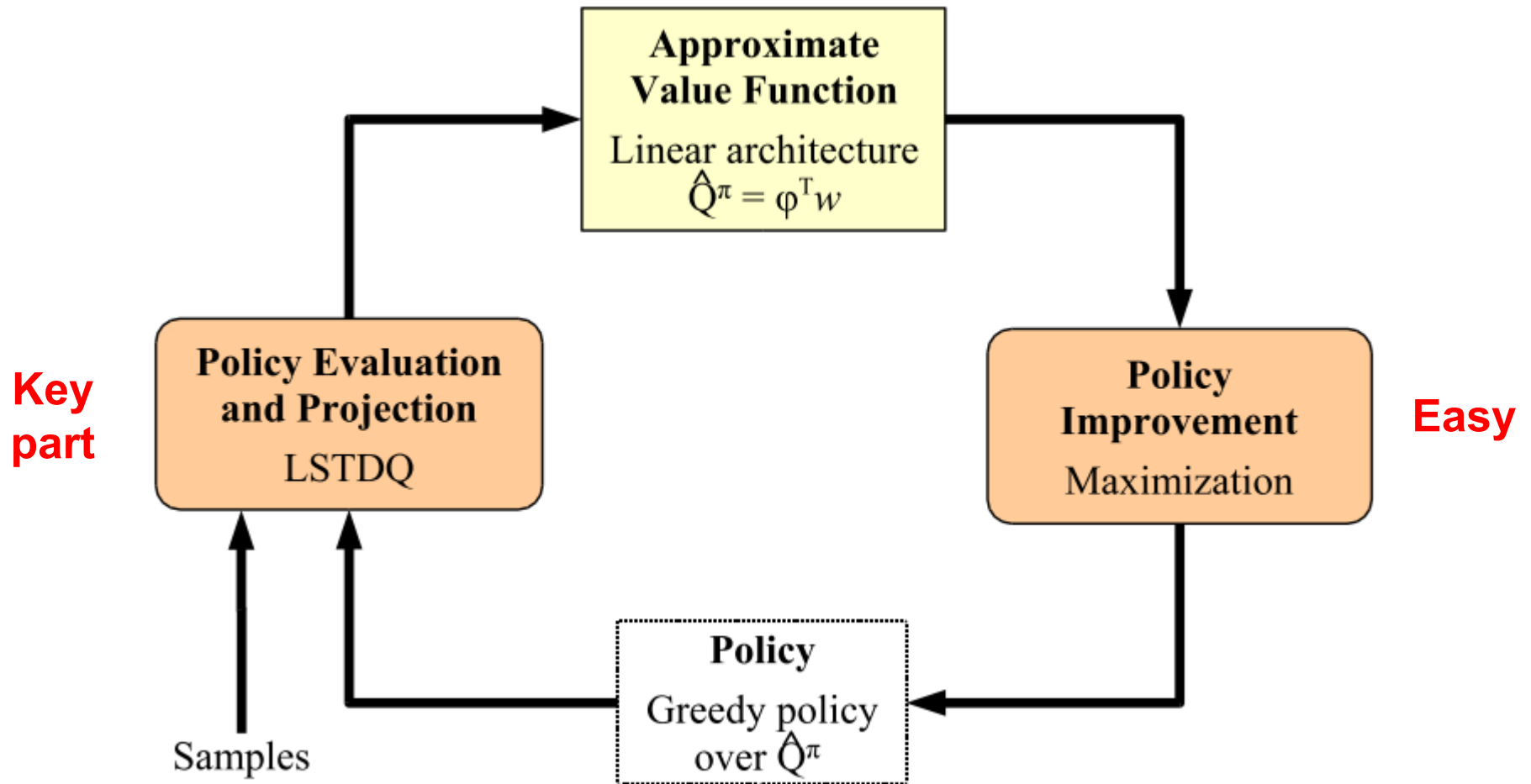
Then, this sequence eventually produces policies whose performance is at most a constant multiple of ϵ away from the optimal performance:

$$\begin{aligned} \limsup_{m \rightarrow \infty} \|\hat{Q}^{\pi_m} - Q^*\|_{\infty} &\leq \frac{2\gamma\epsilon}{(1-\gamma)^2} . && \text{LSPI bound (no error in} \\ &&& \text{policy improvement)} \\ &\leq \frac{\delta + 2\gamma\epsilon}{(1-\gamma)^2} . && \text{generic approx policy} \\ &&& \text{iteration bound} \end{aligned}$$



Least Squares Policy Iteration

Lagoudakis & Parr (2003)



Projection Approach to Approximation

- Recall the standard Bellman equation:

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

or equivalently $V^* = T[V^*]$ where $T[.]$ is the Bellman operator

- Recall from value iteration, the sub-optimality of a value function can be bounded in terms of the Bellman error:

$$\|V - T[V]\|_{\infty}$$

- This motivates trying to find an approximate value function with small Bellman error

Projection Approach to Approximation

- Suppose that we have a space of representable value functions
 - ▲ E.g. the space of linear functions over given features
- Let Π be a *projection* operator for that space
 - ▲ Projects any value function (in or outside of the space) to “closest” value function in the space
- “Fixed Point” Bellman Equation with approximation
$$\hat{V}^* = \Pi(T[\hat{V}^*])$$
 - ▲ Depending on space this will have a small Bellman error
- LSPI will attempt to arrive at such a value function
 - ▲ Assumes linear approximation and least-squares projection



How does LSPI fix these?

- LSPI performs approximate policy iteration
 - ▲ PI involves policy evaluation and policy improvement
 - ▲ Uses a variant of least-squares temporal difference learning (LSTD) for **approx. policy evaluation** [Bratdke & Barto '96]
- Stability:
 - ▲ LSTD directly solves for the fixed point of the approximate Bellman equation for policy values
 - ▲ With singular-value decomposition (SVD), this is always well defined
- Data efficiency
 - ▲ LSTD finds best approximation for any finite data set
 - ▲ Makes a single pass over the data for each policy
 - ▲ Can be implemented incrementally



OK, What's LSTD?

- Least Squares Temporal Difference Learning
- Assumes linear value function approximation of K features

$$\hat{V}(s) = \sum_k w_k \phi_k(s)$$

- The ϕ_k are arbitrary feature functions of states
- Some vector notation

$$\hat{V} = \begin{bmatrix} \hat{V}(s_1) \\ \vdots \\ \hat{V}(s_n) \end{bmatrix} \quad w = \begin{bmatrix} w_1 \\ \vdots \\ w_k \end{bmatrix} \quad \phi_k = \begin{bmatrix} \phi_k(s_1) \\ \vdots \\ \phi_k(s_n) \end{bmatrix} \quad \Phi = [\phi_1 \quad \dots \quad \phi_K]$$



Suppose we know value of policy

- Want: $\Phi \mathcal{W} \approx V^\pi$
- Least squares weights minimizes squared error

$$\mathcal{W} = \underbrace{(\Phi^T \Phi)^{-1} \Phi^T}_{\text{Sometimes called pseudoinverse}} V^\pi$$

Sometimes called pseudoinverse

- Least squares projection is then

$$\hat{V} = \Phi \mathcal{W} = \underbrace{\Phi (\Phi^T \Phi)^{-1} \Phi^T}_{\text{Textbook least squares projection operator}} V^\pi$$

Textbook least squares projection operator



But we don't know V ...

- Recall fixed-point equation for policies

$$V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s'} P(s' | s, \pi(s)) V^\pi(s')$$

- Will solve a projected fixed-point equation:

$$\hat{V}^\pi = \Pi(R + \gamma P \hat{V}^\pi)$$

$$R = \begin{bmatrix} R(s_1, \pi(s_1)) \\ \vdots \\ R(s_n, \pi(s_n)) \end{bmatrix}, \quad P = \begin{bmatrix} P(s_1 | s_1, \pi(s_1)) & \cdots & P(s_n | s_1, \pi(s_1)) \\ \vdots & \ddots & \vdots \\ P(s_1 | s_n, \pi(s_n)) & \cdots & P(s_n | s_n, \pi(s_n)) \end{bmatrix}$$

- Substituting least squares projection into this gives:

$$\Phi w = \Phi(\Phi^T \Phi)^{-1} \Phi^T (R + \gamma P \Phi w)$$

- Solving for w : $w = (\Phi^T \Phi - \gamma \Phi^T P \Phi)^{-1} \Phi^T R$



Almost there...

$$w = (\Phi^T \Phi - \gamma \Phi^T P \Phi)^{-1} \Phi^T R$$

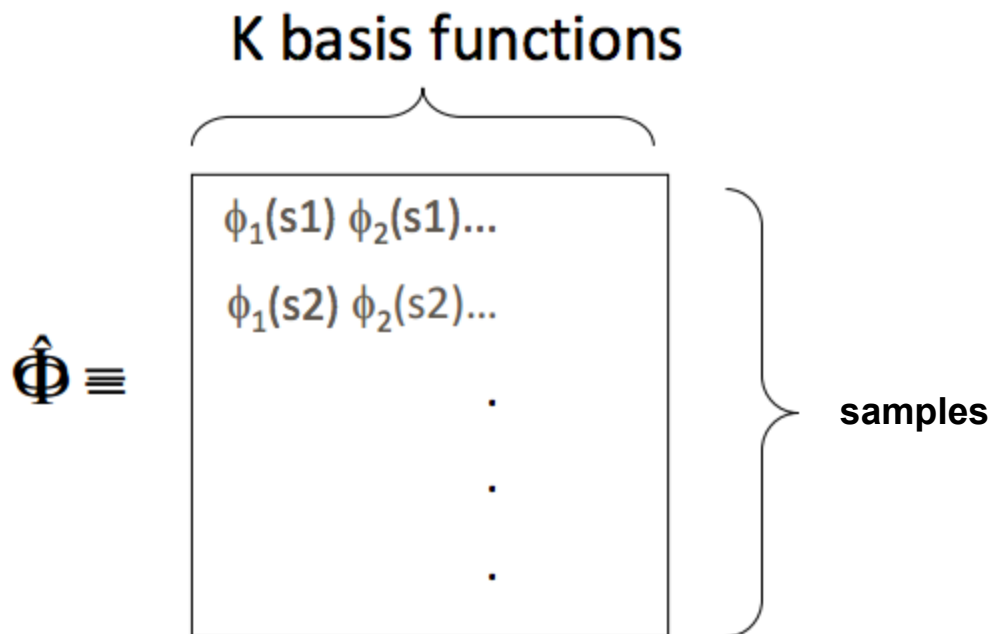
- Matrix to invert is only $K \times K$
- But...
 - ▲ Expensive to construct matrix (e.g. P is $|S| \times |S|$)
 - ▲ We don't know P
 - ▲ We don't know R



Using Samples for Φ

Suppose we have state transition samples of the policy running in the MDP: $\{(s_i, a_i, r_i, s'_i)\}$

Idea: Replace enumeration of states with sampled states



Using Samples for R

Suppose we have state transition samples of the policy running in the MDP: $\{(s_i, a_i, r_i, s_i')\}$

Idea: Replace enumeration of reward with sampled rewards

$$R = \begin{array}{|c|} \hline r_1 \\ r_2 \\ \cdot \\ \cdot \\ \cdot \\ \hline \end{array} \left. \vphantom{\begin{array}{|c|} \hline r_1 \\ r_2 \\ \cdot \\ \cdot \\ \cdot \\ \hline \end{array}} \right\} \text{samples}$$

Using Samples for $P\Phi$

Idea: Replace expectation over next states with sampled next states.

$$P\Phi \approx \left\{ \begin{array}{l} \phi_1(s1') \phi_2(s1')... \\ \phi_1(s2') \phi_2(s2')... \\ \cdot \\ \cdot \\ \cdot \end{array} \right\} \quad \left\{ \begin{array}{l} s' \text{ from } (s,a,r,s') \end{array} \right.$$

K basis functions



Putting it Together

- LSTD needs to compute:

$$w = (\Phi^T \Phi - \gamma \Phi^T P \Phi)^{-1} \Phi^T R = B^{-1} b$$

$$B = \Phi^T \Phi - \gamma \Phi^T \underbrace{(P \Phi)}$$

$$b = \Phi^T R$$

from previous slide



Putting it Together

- LSTD needs to compute:

$$w = (\Phi^T \Phi - \gamma \Phi^T P \Phi)^{-1} \Phi^T R = B^{-1} b$$

$$B = \Phi^T \Phi - \gamma \Phi^T \underbrace{(P \Phi)}$$

$$b = \Phi^T R$$

from previous slide

How would this work on the example from Tsitsiklis & Van Roy we did last time? What w is computed?



Different Fitting Objectives:

What is FVI minimizing? What is LSTD minimizing?

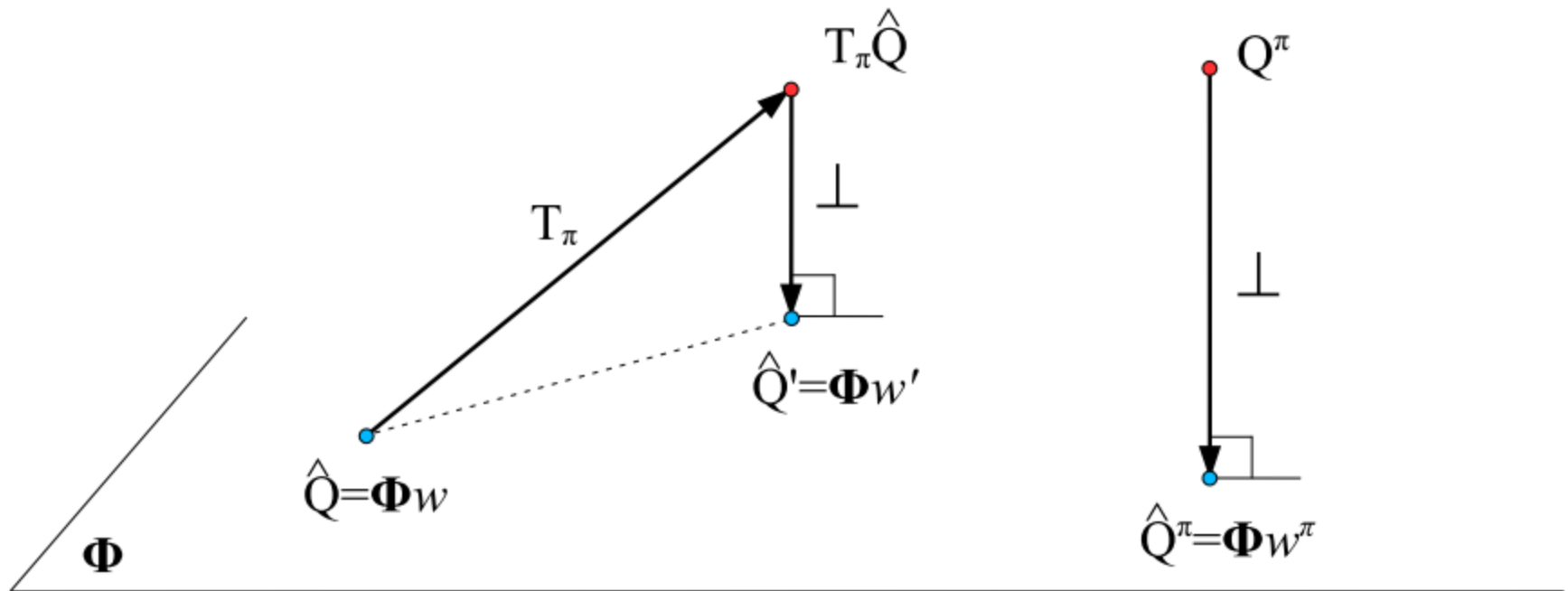


Image from Lagoudakis & Parr 2003

Carnegie Mellon University

Putting it Together

- LSTD needs to compute:

$$w = (\Phi^T \Phi - \gamma \Phi^T P \Phi)^{-1} \Phi^T R = B^{-1} b$$

$$B = \Phi^T \Phi - \gamma \Phi^T \underbrace{(P \Phi)}$$

$$b = \Phi^T R$$

from previous slide

- The hard part of which is B the $k \times k$ matrix:
- Both B and b can be computed incrementally for each (s, a, r, s') sample: (initialize to zero)

$$B_{ij} \leftarrow B_{ij} + \phi_i(s) \phi_j(s) - \gamma \phi_i(s) \phi_j(s')$$

$$b_i \leftarrow b_i + r \cdot \phi_i(s)$$



LSTD Algorithm

- Collect data by executing trajectories of current policy
- For each (s,a,r,s') sample:

$$B_{ij} \leftarrow B_{ij} + \phi_i(s)\phi_j(s) - \gamma\phi_i(s)\phi_j(s')$$

$$b_i \leftarrow b_i + r \cdot \phi_i(s,a)$$

$$w \leftarrow B^{-1}b$$



LSTD Summary

- Does $O(k^2)$ work per datum
 - ▲ Linear in amount of data.
- Approaches model-based answer in limit
- Finding fixed point requires inverting matrix
- Fixed point almost always exists
- Stable; efficient



LSTD Summary

- Does $O(k^2)$ work per datum
 - ▲ Linear in amount of data.
- Approaches model-based answer in limit
- Finding fixed point requires inverting matrix
- Fixed point almost always exists
- Stable; efficient
- Note: LSTD as just described assumes estimating the value of a fixed policy
- Samples generated from this fixed policy



Approximate Policy Iteration with LSTD

Policy Iteration: iterates between policy improvement and policy evaluation

Idea: use LSTD for approximate policy evaluation in PI

Start with random weights \mathbf{w} (i.e. value function)

Repeat Until Convergence

$\pi(s) = \text{greedy}(\hat{V}(s, \mathbf{w}))$ // policy improvement

Evaluate π using LSTD

- Generate sample trajectories of π
- Use LSTD to produce new weights \mathbf{w}
(\mathbf{w} gives an approx. value function of π)



What Breaks?

Policy Iteration: iterates between policy improvement and policy evaluation

Idea: use LSTD for approximate policy evaluation in PI

Start with random weights \mathbf{w} (i.e. value function)

Repeat Until Convergence

$\pi(s) = \text{greedy}(\hat{V}(s, \mathbf{w}))$ // policy improvement

Evaluate π using LSTD

- Generate sample trajectories of π
- Use LSTD to produce new weights \mathbf{w}
(\mathbf{w} gives an approx. value function of π)



What Breaks?

- No way to execute greedy policy without a model
- Approximation is biased by current policy
 - ▲ We only approximate values of states we see when executing the current policy
 - ▲ LSTD is a *weighted* approximation toward those states
- Can result in Learn-forget cycle of policy iteration
 - ▲ Drive off the road; learn that it's bad
 - ▲ New policy never does this; forgets that it's bad
- Not truly a batch method
 - ▲ Data must be collected from current policy for LSTD



LSPI

- LSPI is similar to previous loop but replaces LSTD with a new algorithm LSTDQ
- LSTD: produces a value function
 - ▲ Requires sample from policy under consideration
- LSTDQ: produces a Q-function for current policy
 - ▲ Can learn Q-function for policy from any (reasonable) set of samples---sometimes called an off-policy method
 - ▲ No need to collect samples from current policy
- Disconnects policy evaluation from data collection
 - ▲ Permits reuse of data across iterations!
 - ▲ Truly a batch method.



Implementing LSTDQ

- Both LSTD and LSTDQ compute: $B = \Phi^T \Phi - \lambda \Phi^T (P \Phi)$
- But LSTDQ basis functions are indexed by actions

$$\hat{Q}_w(s, a) = \sum_k w_k \cdot \phi_k(s, a)$$


defines greedy policy: $\pi_w(s) = \arg \max_a \hat{Q}_w(s, a)$

- For each (s, a, r, s') sample:

$$B_{ij} \leftarrow B_{ij} + \phi_i(s, a) \phi_j(s, a) - \lambda \phi_i(s, a) \phi_j(s', \pi_w(s'))$$

$$b_i \leftarrow b_i + r \cdot \phi_i(s, a)$$

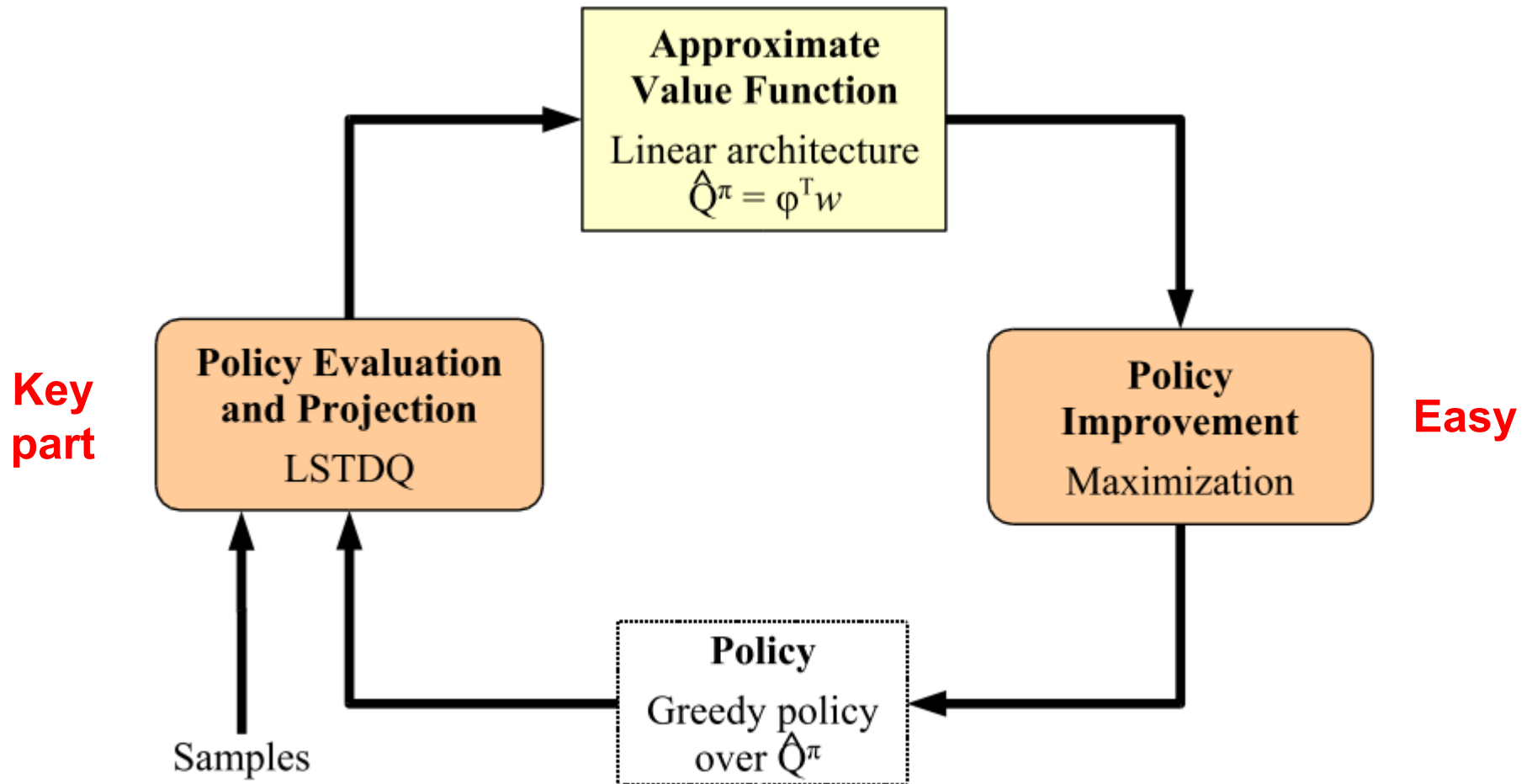
$$w \leftarrow B^{-1} b$$

$$\arg \max_a \hat{Q}_w(s', a)$$




Least Squares Policy Iteration

Lagoudakis & Parr (2003)

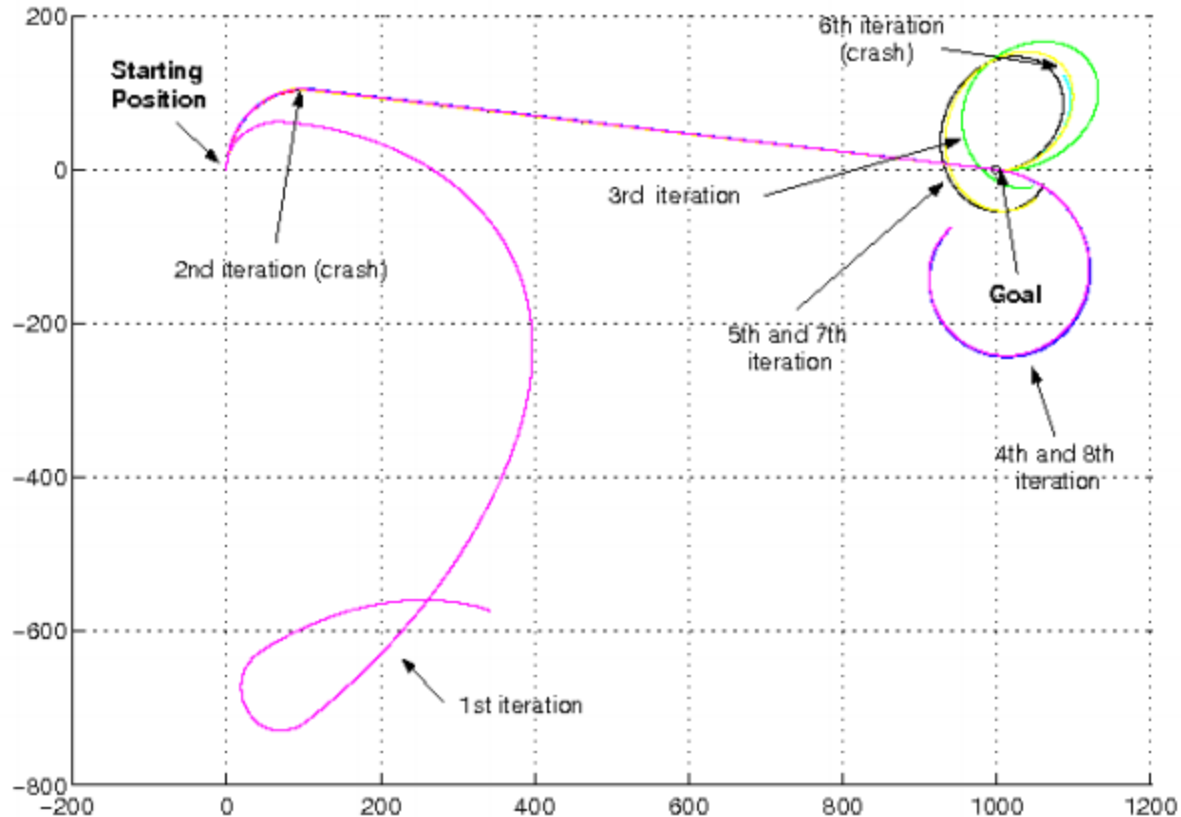


Results: Bicycle Riding

- Watch random controller operate bike
- Collect $\sim 40,000$ (s, a, r, s') samples
- Pick 20 simple feature functions ($\times 5$ actions)
- Make 5-10 passes over data (PI steps)
- Reward was based on distance to goal + goal achievement
- Result:
Controller that balances and rides to goal



Bicycle Trajectories



Convergence of Control Algorithms with Different Representations

Algorithm	Table Lookup	Linear	Non-Linear
Monte-Carlo Control	✓	(✓)	✗
(on-policy Q-learning) Sarsa	✓	(✓)	✗
Q-learning	✓	✗	✗
LSPI	✓	(✓)	-

(✓) = chatters around near-optimal value function



Image from David Silver

Carnegie Mellon University

Open Questions

- Sample efficiency?
- Feature representation?
- Which objective function (fixed point, minimizing projected error, etc) is best?
- Homework 1 is a chance to explore some of these issues

