Access models

Full probabilistic description.

Costs $P_s$

Can't store

Information theoretically hard

Deterministic Generative Model

$f(x, u) \rightarrow X$

Set the random seed
to recreate same scenario

Generative model.

Programmatic access
- Rested model.
  Execute \( f(x, u) \) sequences and anytime reset to \( x_0 \).

- Trace model.
  Can't ever go back.

Approximate Dynamic Programming:
(Neuro-Dynamic Programming)
- Too expensive to compute and store value iteration \( V(x, t) \)
- Need weaker access to model \( \rightarrow \) generative model.

Fitted Value Iteration - (FVI)
Regression Algorithm as oracle.

\[
D = \sum_{i=1}^{m} \alpha_i y_i z_i \quad \alpha_i \rightarrow \text{vector of features}
\]

\( z_i \rightarrow \text{representation of state space} \quad \text{and q-value} \)
\[ f : x \rightarrow y \]

\[ \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - y_i)^2 \text{ is small} \]

- Squared error, sensitive to outliers
- Overfitting
- How to fix?
  - by minimizing
    \[ \mathbb{E} \left[ (f(x) - y)^2 \right] \leq \varepsilon \]
    
    and regularize

What regression algos?

- Linear Regression
- Locally Weighted Regression
- Gaussian Processes
- Random forests
- Neural Network
- Splines
\[ V_k(s) = \min_a \text{cost}(s, a) + \gamma \sum_{s'} \pi(s' | s, a) V_{k+1}(s') \]

**General Batch ADP Data Flow:**

1. Sample estimated model
2. Needs only the reset model
3. Data = \( \delta \times \text{States} \) minimum.

- Initially yet by
  1. Watch human play
  2. Watch random play
  3. Randomly populate board.

But keep in mind that some states are not reachable.

Regular Value Iteration also has that problem.
Pattern Set
\[
\exists \begin{array}{c}
\{ \, ? \, ? \, ? \, \} \\
\text{States} \\
(x) \quad \min \text{cost}(s,a) + \sum_{s'} \frac{1}{\rho(s' / s, a)} N_{K_1}(s') \\
\text{to get } Y \text{ value.}
\end{array}
\]

This is set to 0 for the 1st iteration.

(Generative model)

General Batch ADP Data Flow:
Fitted Q-Iteration (FQI)

Quality Function.
Action-Value Function.

\( Q(s, a) \) = Value you get if you choose \( a \) at \( s \) and then follow the optimal policy.

\( V(s) \) = \( \min_a Q(s, a) \)

\( \pi^* = \arg\min_a Q^*(s, a) \)

\( Q(s, a, t) = \text{cost}(s, a) + \min_{a'} Q(s', a', t+1) \)

Stochastic policy requires more space: \( \text{cost}(s, a) + \frac{1}{\varepsilon} \min_{s', a'} Q(s', a') \)
Pattern Set

\[ \sum_{S} \min_{a} Q^k(S, a) \]

Just store the samples input.
No model.

Fitted Policy Iteration (FPI)

Recover Policy Iteration

Policy Iteration

(a) Evaluate Policy
Calculate \( V^k(S) \)

\[ V^k(S) = \text{cost}(S, \pi^k(S)) + \gamma \sum_{S'} P(S'|S, \pi^k(S)) \max_{a} Q^k(S', a) \]

(b) Improve Policy

\( \pi^k' \) is argmin \( a \)

(Note: Don't need to use model)
Pattern Set

\[ \{ (s, a), c(s, a) + \gamma Q^\pi(s', \pi(s')) \} \]

(Data Flow Diagrams is same as FAI)

Advantages (vs. FU)

* Won't Error Blowup.

* Stable than FU. (as it is linear)
  in the Evaluate Policy equation.

Toy Examples

(Eggen and Moore)

Cam on the Hill problem

Empty grid with goal at one corner.

You have to go the wrong way to go the right way.
Convergence results?

- Relatively weak
  "The Averages"
  (Gordon et al.)

- "On policy" temporal difference methods will have stronger results.
  \[ \Rightarrow \text{Hang on, still coming.} \]

- Error analysis?
  Horrendous.

Why does F/I go horribly wrong sometimes?

I suggest just not enough and your

\[ \max_a r(s, a) + \beta \sum_{s'} p(s', a) V_{T-1}(s') \]

Amplifying error? 

Problem

1) Where should sample set be?

2) Varn in value iteration amplifies errors.

In policy iteration, the inner loop has no max or min term and hence is more stable.

Recap

- Fitted Dynamic Programming (FDP)
- FQI

1) Bootstrapping increases error (feedback). We can fix this.

2) Where should the sample set be?
For problem 2

the problem in FPI case happens in the policy improvement
but (in the outer loop)

\[ \sqrt{N} \]

It wants to change the actions for every state such that the next state you land in has a higher value.

But you don't have values in that area.

Fix

High level idea:

Cache policies and value functions.

An optimal policy has the property that no matter what you do now, hereafter
you must act optimally.

Alg: $\triangledown$ to fix

Policy Search by Dynamic Prag.

for each $t \in T-1, \ldots, 0$:

$\Delta_t = \text{Generate Samples} \left( \{ x_t^{(i)}, \ldots, x_{t-1}^{(i)} \} \right)$

$h_t = \text{Learn} (\Delta_t)$

Generate Samples

$\Delta^F (a_0, s, T-4)$

Choose the $h$ as the higher reward

$\Delta (a_1, s, T-4)$
1. We can say that this is a classification problem. Find which state at $T-1$ maps to which action $a_0$.

Weight the classification problem as:

$$Q^*(a_1, s_1, T-1) = Q^*(a_0, s, T-1)$$

For multiple weights make this the best action.

2. Regression on $Q(s)$

$$R(s_t) = \arg \max_a Q(s, a, t)$$

This is different than FBE, FQI.

This is more expensive as we have to do $T$ times more work so $(T^2)$ but no bootstrapping stable.
* Errors in bootstrapping are propagated but rollouts are not.

* Performance guarantee

- $V$ learned $\geq V^{\text{true}}$ how well any policy you are clarifying?
where $a_k = \frac{1}{T}$

$\pi_{\text{new}} = \pi_{\text{old}} + (1-\rho) \Delta$

Policy Iteration

$Q(s,a) = \max_{\pi} \pi(s') [R(s,a) + \gamma \pi(s')]$

$Q^*(s,a) = \max_{\pi} \pi(s') [R(s,a) + \gamma \pi(s')]$

$\pi^{*} = \pi_{\text{argmax}} Q^*(s,a)$
with \((1-\gamma)\) probability you stop and try out actions; \(\gamma\) is the discount factor.

**Temporal Difference Methods:**

Upto now we have seen batch methods: (FVI) \(\rightarrow\) SSA and \(\epsilon\)-Learning (FAI) \(\rightarrow\) TD (EPI) \(\rightarrow\) \(\epsilon\)TD

**Overview**

\[ s' \]

\[ \Delta \]

Choose Action

Update V/A

Simulator

**PD:** policy evaluation.

given \(\pi\), estimate \(V_\pi(s), V_\pi\)

Goal: find \(s^*\). \(V_{\pi^*}(s) = \max_{a \in A(s)} Q_\pi(s,a)\)

Find \(s^*\): \(V_{\pi}(s) = R(s, \pi(s)) + \gamma \mathbb{E}_{s' \sim P} V_{\pi}(s') \)