Feature Selection for Approximate Offline RL

Emma Brunskill
Image from David Silver
Spoken Dialogue System Example
(Pietquin, Geist, Chandramohan & Frezza-Buet)

• Form-filling, task oriented information system for restaurants
• Goal: determine the value of 3 slots
  • Restaurant location
  • Restaurant cuisine
  • Restaurant price range
• Information state of slot represents confidence in the value (from 0 to 1) → State space is 3 dim continuous vector
• Action space:
  • Ask-A-Slot (one for each slot), ExplicitConfirm-Slot (one for each slot), Implicit-Confirm-And-Ask-A-Slot (6 actions, in combination of 2 slots) and Close-Dialogue action.
Spoken Dialogue System Example
(Pietquin, Geist, Chandramohan & Frezza-Buet)

• Form-filling, task oriented information system for restaurants
• Goal: determine the value of 3 slots
  • Restaurant location
  • Restaurant cuisine
  • Restaurant price range
• Information state of slot represents confidence in the value (from 0 to 1) → State space is 3 dim continuous vector
• Q-function representation?
  • $351 = 3^3 \times 13$ Radial basis functions
  • 3 Gaussian kernels for each state dimension
  • 13 actions
What if We Have Very Little Data? What is the Danger?

- Form-filling, task oriented information system for restaurants
- Goal: determine the value of 3 slots
  - Restaurant location
  - Restaurant cuisine
  - Restaurant price range
- Information state of slot represents confidence in the value (from 0 to 1) → State space is 3 dim continuous vector
- Q-function representation?
  - $351 = 3^3 \times 13$ Radial basis functions
  - 3 Gaussian kernels for each state dimension
  - 13 actions
What if We Have Very Little Data?

What is the Danger? Overfitting

- Form-filling, task oriented information system for restaurants
- Goal: determine the value of 3 slots
  - Restaurant location
  - Restaurant cuisine
  - Restaurant price range
- Information state of slot represents confidence in the value (from 0 to 1) → State space is 3 dim continuous vector
- Q-function representation?
  - $351 = 3^3 \times 13$ Radial basis functions
  - 3 Gaussian kernels for each state dimension
  - 13 actions
Feature-Based Approximate RL

- Where do features come from?
- Does it matter?
  - Yes!
  - Impacts computation
  - Impacts performance
    - Changes feature class, representational power
    - Changes finite sample (finite dataset) performance (can lead to overfitting, changes estimation error)
Overview of Selecting Features for Feature-Based Approximate RL

1. Feature selection
   Input: Big feature set
   Output: Subset of original features

2. Feature compression/projection
   Input: Big feature set
   Output: Projected (dimensionality reduc) features

3. Feature construction
   Input: Small feature set
   Output: Superset of original feature set
Feature Selection

- Input: Big feature set
- Output: Subset of features
- Techniques build strongly on supervised learning regularization
- L2 norm (Ridge regularization)
  - $\min_w ||Y - Xw||_2 + b ||w||_2$
- L1 norm (Lasso)
  - $\min_w ||Y - Xw||_2 + b ||w||_1$
Feature Selection for Approximate RL

<table>
<thead>
<tr>
<th>Objective of Fitting Q/V</th>
<th>L2 Regularization (Ridge)</th>
<th>L1 Regularization (LASSO)</th>
<th>Orthogonal Matching Pursuit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fitted V/Q Iteration</td>
<td>X</td>
<td>LASSO on FQI</td>
<td>Value pursuit iteration</td>
</tr>
</tbody>
</table>

Comparisons across AVI (approximate value iteration) & API (approximate policy iteration) are rare
Feature Dimensionality Reduction

Take a set of features, and project down to a lower dimensional basis

Can use any form of dimensionality reduction (Principle component analysis, …)
Feature Construction

- Protovalue function construction (Mahadevan & colleagues)
- Bellman Error Basis Function (BEBF) (Parr et al. 2007)
- Incremental Feature Dependency Discovery (Geramifard & colleagues)
Proto-Value Functions

Proto-value functions are **reward-independent**
global (or local) basis functions, **customized**
to a state (action) space
Value Function Approximation using Fourier and Wavelet Bases

These bases are automatically learned from a set of transitions \((s,a,s')\)
Overview for Protovalue Function Basis Invention

(Mahadevan, AAAI, ICML, UAI 2005; Mahadevan & Maggioni, NIPS 2005; Maggioni and Mahadevan, ICML 2006)
Overview of Selecting Features for Feature-Based Approximate RL

1. Feature selection
   Input: Big feature set
   Output: Subset of original features

2. Feature compression/projection
   Input: Big feature set
   Output: Projected (dimensionality reduc) features

3. Feature construction
   Input: Small feature set
   Output: Superset of original feature set
Evaluation of Methods for Designing Features (for RL)

1. Empirical quality of resulting solution
   Mean square error relative to true value function
   Output: Subset of original features

2. Computational complexity
   As a function of features, data set size, ...
   Output: Projected (dimensionality reduc) features

3. Formal guarantees on performance
   Is the method stable? (Converge to a fixed set of features)
   If a small set of features is sufficient to represent V, can find that set?

4. Sample efficiency
   How well does it use the available data to find good features?
Rest of Today

1. Feature selection
   Input: Big feature set
   Output: Subset of original features
   Idea: Greedily select features.

2. Feature compression/projection
   Input: Big feature set
   Output: Projected (dimensionality reduc) features

3. Feature construction (may get to)
   Input: Small feature set
   Output: Superset of original feature set
OMP Overview: On the board
OMP Empirical Comparison

- LARS-TD: LSTD + L1 regularization
- LARS-BRM: BRM + L1 regularization
- OMP-TD
- OMP-BRM
Empirical Setup

<table>
<thead>
<tr>
<th>Problem</th>
<th>State space</th>
<th>Features</th>
<th>Samples</th>
<th>Trials</th>
<th>LARS-TD $L_2$?</th>
<th>BRM double samples?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chain</td>
<td>Discrete, 50 states</td>
<td>208</td>
<td>500</td>
<td>1000</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Pendulum</td>
<td>Continuous, 2d</td>
<td>268</td>
<td>200</td>
<td>1000</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Blackjack</td>
<td>Discrete, 203 states</td>
<td>219</td>
<td>1600</td>
<td>1000</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Mountain Car</td>
<td>Continuous, 2d</td>
<td>1366</td>
<td>5000</td>
<td>100</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Puddleworld</td>
<td>Continuous, 2d</td>
<td>570</td>
<td>2000</td>
<td>500</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Two Room</td>
<td>Continuous, 2d</td>
<td>2227</td>
<td>5000</td>
<td>1000</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

Size of dataset used to fit $V^*$

Number of trials used to evaluate resulting solution/weights
OMP Results: TD generally better than BRM, OMP Generally better than L1 Regularization

Figures from Painter-Wakefield & Parr
*Important Notes on Empirical Comparison*

- **LARS-TD**
  - Sometimes added small amount of L2 regularization on top of L1 regularization
- **OMP-BRM**
  - Sometimes added in small amount of L2 regularization
- **OMP-TD**
  - Added small amount of L2 regularization when computing final solution for a given beta
  - Seemed critical to get stable performance for harder problems
  - When # samples very small, more unstable
OMP-BRM and OMP-TD Summary

Takes in a set of features
Greedily adds features to set
OMP-TD has better empirical performance than OMP-BRM, but OMP-BRM has stronger theoretical guarantees
OMP-BRM/TD Limitation

Scalability

Required to compute residual with all (remaining) features at every iteration
Rest of Today

1. Feature selection
   Input: Big feature set
   Output: Subset of original features
   Idea: Greedily select features.

2. Feature compression/projection

3. Feature construction
   Input: Small feature set
   Output: Superset of original feature set
   Idea: Greedily add conjunctions of features
Alternative: Generate Features
The original algorithm was an online algorithm.
Batch-iFDD

Run iFDD in Batch: Add new feature (conjunction) with highest error reduction (akin to OMP-TD).

**Theorem:** iFDD in batch approximately finds the feature with the best guaranteed error reduction.

\[
\| \tilde{\mathbf{V}} - \Pi T(\tilde{\mathbf{V}}) \| \\
\]

\[
\tilde{f}_1^* = \arg\max_{f \in \text{pair} (\mathbf{x})} \frac{\sum_{i \in \{1, \ldots, m\}, \phi_f(s_i) = 1} \delta_i}{\sqrt{\sum_{i \in \{1, \ldots, m\}, \phi_f(s_i) = 1} 1}} \\
iFDD^+
\]

\[
\tilde{f}_2^* = \arg\max_{f \in \text{pair} (\mathbf{x})} \sum_{i \in \{1, \ldots, m\}, \phi_f(s_i) = 1} |\delta_i| \\
iFDD^{\text{[Geramifard et al. 2012]}}
\]
Batch-iFDD

Loop

1. Run LSTD \[\text{Bradtke & Barto 1996}\]

2. Expand feature sets

Remaining

Used
Batch-iFDD

Loop

1) Run LSTD [Bradtke & Barto 1996]

2) Expand feature sets

Remaining

Used
Batch-iFDD

Loop

1. Run LSTD [Bradtke & Barto 1996]

2. Expand feature sets
Feature Generation with OMP

Batch IFDD+ sample efficient and (computationally) scalable
Still relies on decent input set of features
Requires input features are binary
Also limits type of features can create
OMP-TD can handle general features
Summary

1. Feature selection
   Should be able to characterize OMP-BRM & OMP-TD
   (computational complexity, strengths/limitations)
   Should be able to implement both

2. Feature compression/projection (know these exist)

3. Feature construction
   Should understand (at a high level) how Batch iFDD works
   Be able to list benefits over OMP