A Non-Parametric Approach to Dynamic Programming

Nips 2011 - Dec. 13

Oliver Kroemer
Jan Peters

Technische Universitaet Darmstadt
Max Planck Institute for Intelligent Systems
Motivation

Focus has been on learning in episodic scenarios...

Developing methods for chaining these learned skills

[Muelling et al 2011]
[Kober & Peters NIPS 2008]
Outline

I. Introduction and Background
II. Non-Parametric Dynamic Programming
III. Numerical Evaluation
IV. Conclusion
Reinforcement Learning

Agent

Environment

[Sutton & Barto 1998]
Reinforcement Learning

Agent \( S_t \)

Environment

state \( \Rightarrow S_t \)

[Sutton & Barto 1998]
Reinforcement Learning

\[ \pi(a|s) \]

Agent

Environment

state \( \Rightarrow S_t \)

action \( \Rightarrow a_t \)

[Sutton & Barto 1998]
Reinforcement Learning

next state $\Rightarrow s'_t = s_{t+1}$

Agent

$\pi(a|s)$

Environment

$p(s'|a,s)$

state $\Rightarrow s_t$

action $\Rightarrow a_t$

[Sutton & Barto 1998]
Reinforcement Learning

\[ s'_t = s_{t+1} \]

Agent

\[ \pi(a|s) \]

Environment

\[ p(s'|a,s), r(s,a) \]

state \[ \Rightarrow S_t \]

action \[ \Rightarrow a_t \]

reward \[ \Rightarrow r_t \]

next state \[ \Rightarrow s'_t = s_{t+1} \]
**Reinforcement Learning**

Objective: Maximize accumulation of rewards

\[ \pi(a | s) \]

\[ p(s' | a, s) \]

\[ r(s, a) \]

\[ s'_t = s_{t+1} \]

\[ r_t \]

\[ s_t \]

\[ a_t \]
Value Functions

Value Function:

\[ V(s) = \mathbb{E} \left\{ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, \pi(a|s) \right\} \]

[Sutton & Barto 1998]
Value Functions

Value Function:

\[ V(s) = \mathbb{E} \left\{ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \Big| s_0 = s, \pi(a|s) \right\} \]

Rewards Initial state

[Sutton & Barto 1998]
Value Functions

Value Function:

\[ V(s) = \mathbb{E} \left\{ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, \pi(a|s) \right\} \]

- **Rewards**
- **Initial state**
- **Discount factor**
Value Functions

Value Function:

\[ V(s) = \mathbb{E} \left\{ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, \pi(a|s) \right\} \]

- Rewards
- Initial state
- Discount factor
- Policy

[Sutton & Barto 1998]
Value Functions

Value Function:

\[ V(s) = \mathbb{E} \left\{ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, \pi(a|s) \right\} \]

Policy Iteration:

- High value indicates a “good” state to visit
- Improved policy: action with high reward and next value

[Sutton & Barto 1998]
Value Functions

Bellman Equation:
[Bellman 1953]

\[ V(s) = \int_A \int_S \pi(a|s) p(s'|s,a) [r(s,a) + \gamma V(s')] \, ds' \, da \]
Value Functions

Bellman Equation: [Bellman 1953]

\[ V(s) = \int_A \int_S \pi(a|s) p(s'|s, a) \left[ r(s, a) + \gamma V(s') \right] ds' da \]
Bellman Equation:
[Bellman 1953]

\[ V(s) = \int_A \int_S \pi(a|s) p(s'|s, a) \left[ r(s, a) + \gamma V(s') \right] ds' da \]

- Value function is invariant under this recursion
- Unique value function always exists
Bellman Equation:

\[ V(s) = \int_A \int_S \pi(a|s) p(s'|s, a) \left[ r(s, a) + \gamma V(s') \right] ds'da \]

- Value function is invariant under this recursion
- Unique value function always exists

Focus on computing values for continuous systems
Reinforcement Learning Approaches
Reinforcement Learning Approaches

- Reinforcement Learning
- Policy Search

Data \Rightarrow \pi'$
Reinforcement Learning Approaches

- **Reinforcement Learning**
  - **Policy Search**: $\text{Data} \Rightarrow \pi'$
  - **Value Function Methods**: $\text{Data} \Rightarrow V(s) \Rightarrow \pi'$
Reinforcement Learning Approaches

Policy Search

Value Function Methods

Dynamic Programming

Data $\Rightarrow \pi'$

Data $\Rightarrow V(s) \Rightarrow \pi'$

Data $\Rightarrow$ Model $\Rightarrow V(s) \Rightarrow \pi'$
Reinforcement Learning Approaches

- Policy Search
  - Data $\Rightarrow \pi'$

- Value Function Methods
  - Data $\Rightarrow V(s) \Rightarrow \pi'$

- Dynamic Programming
  - Data $\Rightarrow$ Model $\Rightarrow V(s) \Rightarrow \pi'$
Value Function Methods

Value Function Methods
Value Function Methods

Monte Carlo
Value Function Methods

Monte Carlo
High Variance

[Schoknecht 2002]
Value Function Methods

- Monte Carlo
  - High Variance
- Temporal Difference
Value Function Methods

- Monte Carlo
  - High Variance
- Temporal Difference
  - Biased Solution

[Schoknecht 2002]
Value Function Methods

- Monte Carlo
  - High Variance

- Temporal Difference
  - Biased Solution

- Residual Gradient
  - Double Samples

[Schoknecht 2002]
Reinforcement Learning Approaches

Policy Search
- Data $\Rightarrow \pi'$

Value Function Methods
- Data $\Rightarrow V(s) \Rightarrow \pi$

Dynamic Programming
- Data $\Rightarrow$ Model $\Rightarrow$ $V(s) \Rightarrow \pi'$
Model:

Transition Table $P_{ss'}^a$

Reward Table $R_s^a$

Discrete State Dynamic Programming

[Bellman 1953]
Discrete State Dynamic Programming

Model:

Transition Table $P_{ss'}^a$

Reward Table $R_s^a$

Value function:

Value Table $V_s$

<table>
<thead>
<tr>
<th>$V(s)$</th>
<th>$s = 1$</th>
<th>$s = 2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V(1)$</td>
<td>19</td>
<td>20</td>
</tr>
</tbody>
</table>

[Bellman 1953]
Discrete State Dynamic Programming

Model:

- Transition Table \( P_{ss'}^a \)
- Reward Table \( R_s^a \)

Value function:

- Value Table \( V_s \)

Value Table:

<table>
<thead>
<tr>
<th>( V_s )</th>
<th>( s = 1 )</th>
<th>( s = 2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V(s) )</td>
<td>19</td>
<td>20</td>
</tr>
</tbody>
</table>

Requires discretization

[Bellman 1953]
Model:

\[ s_{t+1} = A s_t + B a_t \]

\[ a_t = K s_t \]

\[ r(s, a) = -s^T Q s - a^T R a \]

Linear-Quadratic Optimal Control

[Kalman 1960]
Linear-Quadratic Optimal Control

Model:
\[ s_{t+1} = A s_t + B a_t \]
\[ a_t = K s_t \]
\[ r(s, a) = -s^T Q s - a^T R a \]

Value function:
\[ V(s) = -s^T P s \]
Linear-Quadratic Optimal Control

Model:

\[ s_{t+1} = A s_t + B a_t \]

\[ a_t = K s_t \]

\[ r(s, a) = -s^T Q s - a^T R a \]

Value function:

\[ V(s) = -s^T P s \]

Requires linearization

[Kalman 1960]
I. Introduction and Background

II. Non-Parametric Dynamic Programming
   I. System modeling
   II. Form of the value function
   III. Policy evaluation

III. Numerical Evaluation

IV. Conclusion
Non-Parametric Dynamic Programming

Knowledge of system obtained from \( n \) samples:

| \( S_i \) | \( A_i \) | \( S'_i \) | \( r_i \) |
| State | Action | Next State | Reward |
Non-Parametric Dynamic Programming

Knowledge of system obtained from $n$ samples:

- $S_i$: State
- $a_i$: Action
- $s'_i$: Next State
- $r_i$: Reward

Model the system’s joint distribution $p(s, a, s')$

$$p(s, a, s') = p(s'|a, s)p(a|s)p(s)$$
Kernel density estimate

\[ p(s, a, s') = n^{-1} \sum_{i=1}^{n} \psi(s', s'_i) \varphi(a, a_i) \phi(s, s_i) \]

- Convergence to true distribution [Weid & Weissbach 2010]
Kernel density estimate

\[ p(s, a, s') = n^{-1} \sum_{i=1}^{n} \psi (s', s_i) \varphi (a, a_i) \phi (s, s_i) \]

- Convergence to true distribution [Weid & Weissbach 2010]

Nadaraya-Watson reward function

[Nadaraya 1964][Watson 1964]

\[ r(s, a) = \mathbb{E}[r | s, a] = \frac{\sum_{k=1}^{n} r_k \varphi (a, a_k) \phi (s, s_k)}{\sum_{i=1}^{n} \varphi (a, a_i) \phi (s, s_i)} \]
NPDP

Form of Value Function

Determine form of value function for this model

- Invariance under Bellman equation [Bellman 1953]
Determine form of value function for this model

- Invariance under Bellman equation [Bellman 1953]

Bellman equation expanded for model:

\[ V(s) = \sum_{i=1}^{n} \phi(s, s_i) \left[ r_i + \gamma \int_{\mathcal{S}} \psi(s', s_i') V(s') \, ds' \right] \]

\[ \sum_{j=1}^{n} \phi(s, s_j) \]
Form of Value Function

Determine form of value function for this model

- Invariance under Bellman equation [Bellman 1953]

Bellman equation expanded for model:

\[ V(s) = \sum_{i=1}^{n} \phi(s, s_i) \left[ r_i + \gamma \int_{\mathcal{S}} \psi(s', s_i') V(s') \, ds' \right] / \sum_{j=1}^{n} \phi(s, s_j) \]

Constants \( \theta_i \)
Form of Value Function

Determine form of value function for this model

- Invariance under Bellman equation [Bellman 1953]

Bellman equation expanded for model:

\[
V(s) = \sum_{i=1}^{n} \phi(s, s_i) \left[ r_i + \gamma \int_{\mathcal{S}} \psi(s', s_i') V(s') \, ds' \right] \frac{\sum_{j=1}^{n} \phi(s, s_j)}{\sum_{j=1}^{n} \phi(s, s_j)}
\]

- Invariant iff value function has Nadaraya-Watson form
Resulting value function:

\[
V(s) = \frac{\sum_{k=1}^{n} \theta_k \phi(s, s_k)}{\sum_{i=1}^{n} \phi(s, s_i)}
\]

- basis functions match observed system dynamics
NPDP
Policy Evaluation

Resulting value function:

\[ V(s) = \frac{\sum_{k=1}^{n} \theta_k \phi(s, s_k)}{\sum_{i=1}^{n} \phi(s, s_i)} \]

- basis functions match observed system dynamics

Policy evaluation:

\[ \theta = (I - \gamma \lambda)^{-1} r \]

\[ [\lambda]_{i,j} = \int_{S} \frac{\phi(s', s_j) \psi(s', s'_i)}{\sum_{k=1}^{n} \phi(s', s_k)} ds' \]

- Only requires inversion of a sparse matrix
Algorithm Overview

<table>
<thead>
<tr>
<th>NON-PARAMETRIC DYNAMIC PROGRAMMING</th>
</tr>
</thead>
</table>

**INPUTS:**
\[ n \text{ samples:} \]
\[ s_i, s'_i, r_i \]

**kernel functions:**
\[ \phi(\cdot, \cdot), \psi(\cdot, \cdot) \]

**COMPUTATIONS:**
\[ \theta = (I - \gamma \lambda)^{-1} r \]
\[ [\lambda]_{i,j} = \int_{S} \frac{\phi(s', s_j) \psi(s', s'_i)}{\sum_{k=1}^{n} \phi(s', s_k)} ds' \]

**OUTPUT:**
\[ V(s) = \sum_{k=1}^{n} \theta_k \phi(s, s_k) \]
\[ \sum_{i=1}^{n} \phi(s, s_i) \]
I. Introduction and Background

II. Non-Parametric Dynamic Programming

III. Numerical Evaluation

IV. Conclusion
Numerical Evaluation

- Sinusoidal transition and reward functions
- Gaussian noise on transitions
- Monte Carlo baseline - 500,000 sampled trajectories
Value Function

Monte Carlo
100 Samples Evaluation

M.S.E. 3.8300
200 Samples Evaluation

M.S.E. 0.0244
300 Samples Evaluation

Monte Carlo
NPDP

M.S.E. 0.0135
Discussion

Promising initial results

- Accurately predicts true value function using limited samples
- Requires policy improvement for robot application
Conclusion

A Non-Parametric Dynamic Programming Approach

- Approximate system rather than value function directly
- Model system in a flexible manner using KDE
- Precisely compute value function for this model
- Ensures that basis functions match observed dynamics
- Positive initial results in numerical evaluations

Paper also includes a common framework for deriving NPDP, LSTD, and Kernelized TD methods; e.g. GPTD.
References