Adaptive Aggregation for Approximate Dynamic Programming Methods

Workshop on restless bandits, index policies and applications in reinforcement learning

Université Grenoble Alpes, November 20–21 2023

Orso Forghieri

Erwan Le Pennec (École Polytechnique)

Hind Castel (Télécom SudParis)

Emmanuel Hyon (Sorbonne Université)

November 20, 2023
Markov Decision Processes and Four Rooms instance

Four rooms:
- $S = [0 ; 100]$, $\mathcal{A} = \{N, S, E, W\}$
- Reward: $-1$ until exit is reached, $0$ otherwise.
Increasing complexity
Table of Contents

1. State Abstraction and Approximate Dynamic Programming
2. Adaptive Aggregation Value Iteration algorithm
3. Runtimes comparison
4. Conclusion
Table of Contents

1 State Abstraction and Approximate Dynamic Programming

2 Adaptive Aggregation Value Iteration algorithm

3 Runtimes comparison

4 Conclusion
Litterature context

Our objectives:

- Find a good approximation of a large MDP
- Find the optimal policy on this approximation

To this end, we need:

- State Abstraction context to have intuition on how to build a final “good” abstraction
- Approximate Dynamic Programming to solve the new simplified problem iterating a contracting operator
Hierarchical Reinforcement Learning\(^1\)

Two types of learnable hierarchy to divide a MDP:

- Temporal Abstraction: apply a series of actions that skip timesteps (like a given skill)

- Spatial Abstraction: Divide state space into regions and jump from one to another

\(^1\)[Abel, 2022] gives a good insight of it.
State Abstraction for Markov Decision Processes

State Abstraction characteristics:

- Gather similar states\(^2\) into regions to form a new simpler MDP\(^3\)
- Reasonable loss of information\(^4\)
- Fastly built abstraction

\(^2\) same value, same \(Q\)-value or same policy
\(^3\) [Abel et al., 2016]
\(^4\) [Abel et al., 2019]
State Abstraction
buildings[Tsitsiklis and Van Roy, 1996]

Let $S = \{s_1, s_2, s_3\} = S_1 \sqcup S_2 = \{s_1, s_2\} \sqcup \{s_3\}$. We define:

$$\phi := (1_{s \in S_k})_{s \in S, 1 \leq k \leq K} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

and

$$\omega := (\phi^T \cdot \phi)^{-1} \cdot \phi^T = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Then

$$V \in \mathbb{R}^S \quad \Pi: = \phi \cdot \omega \quad \tilde{V} \in \mathbb{R}^S \quad \overset{\omega}{\leftrightarrow} \quad V \in \mathbb{R}^K$$

$$V = \begin{pmatrix} 3 \\ 4 \\ 5 \end{pmatrix} \quad \Pi: = \phi \cdot \omega \quad \begin{pmatrix} 3.5 \\ 3.5 \\ 5 \end{pmatrix} \quad \overset{\omega}{\leftrightarrow} \quad \begin{pmatrix} 3.5 \\ 5 \end{pmatrix}$$
Abstract MDP definition

For an original MDP

\[ \mathcal{M} = (S, A, T, R, \gamma) \]

Abstract transition and reward depending on the original:

\[ R_{abs} = \omega \cdot R, \quad T_{abs} = \omega \cdot T \cdot \phi \]

and its optimal value function \( V^* \) is the solution of

\[ V^* = T_{abs} V^* \]

\[ [\text{Abel et al., 2016}] \]
State Abstraction application

\[ V^* = \begin{pmatrix} -96 \\ -96.96 \\ -96.96 \\ -97.37 \end{pmatrix} \in \mathbb{R}^4 \]
Approximate Value Iteration

VI:

\[ V_{t+1} \leftarrow \mathcal{T}^* V_t \]

\[ = \max_{a \in \mathcal{A}} (R_a + \gamma T_a \cdot V) \]

AVI\(^6\):

\[ V_{t+1} \leftarrow \Pi \mathcal{T}^* V_t \]

where \( \Pi \) is a projector on a subspace of \( \mathbb{R}^S \).

For State Aggregation:

\[ \Pi = \phi \cdot \omega := \phi \cdot (\phi^T \cdot \phi)^{-1} \cdot \phi^T \]

so we iterate

\[ V_{t+1} \leftarrow \phi \cdot \omega \cdot \mathcal{T}^* V_t \]

\(^6\)[Powell, 2007]
General context and work

In this work, we achieve to

- Aggregate states *having close value*
- *Approximate optimal value* function
- *Adapt the method* through Q-Value Iteration and Policy Iteration algorithms
Table of Contents

1 State Abstraction and Approximate Dynamic Programming

2 Adaptive Aggregation Value Iteration algorithm

3 Runtimes comparison

4 Conclusion
Lemma (Optimal Error Bound with arbitrary partition, O.F.)

For any piecewise constant value function $\tilde{V}$

$$
\|\tilde{V} - V^*\|_{\infty} \leq \frac{1}{1 - \gamma} \left( \max_{1 \leq k \leq K} \text{Span}_{S_k} \left( \mathcal{T}^* \tilde{V} \right) + \|\tilde{V} - \Pi \mathcal{T}^* \tilde{V}\|_{\infty} \right)
$$

where $\text{Span}_{S_k} V := \max_{s \in S_k} V(s) - \min_{s \in S_k} V(s)$, and $V^*$ is the optimal value function.
Sketch of proof

Proof.

\[
\|V^* - V\|_\infty \leq \frac{1}{1 - \gamma} \|V - T^*V\|_\infty
\]

\[
\leq \frac{1}{1 - \gamma} (\|\Pi T^*V - T^*V\|_\infty + \|V - \Pi T^*V\|_\infty)
\]

\[
\leq \frac{1}{1 - \gamma} \left( \max_{k} \text{Span}_{S_k} (T^*V) + \|V - \Pi T^*V\|_\infty \right)
\]

→ Also true for \( T_Q^* \) and \( T^\pi \) for any \( \pi \)
Remark

Compared to $\mathcal{T}^*$, we lose in complexity:

- $\frac{|S|}{K}$ computing $\Pi \mathcal{T}^*$
- $\left(\frac{|S|}{K}\right)^3$ computing $\Pi \mathcal{T}_Q^*$
- $\left(\frac{|S|}{K}\right)^2$ computing $\Pi \mathcal{T}^\pi$

where $K$ is the number of regions.
Adaptive Aggregation algorithm

From

$$\|\tilde{V} - V^*\|_\infty \leq \frac{1}{1 - \gamma} \left( \max_{1 \leq k \leq K} \text{Span}_{S_k} \mathcal{T}^*\tilde{V} + \|\tilde{V} - \Pi \mathcal{T}^*\tilde{V}\|_\infty \right)$$

we propose the following process:

1. Approximate the original state space by a unique trivial region
2. Then alternate between
   1. Progressively refine the partitioning along $\mathcal{T}^*\tilde{V}$ to reduce $\max_{1 \leq k \leq K} \text{Span}_{S_k} \mathcal{T}^*\tilde{V}$
   2. Iterate the contracting operator $\Pi \mathcal{T}^*$ to reduce Projected Bellman Residual $\|\tilde{V} - \Pi \mathcal{T}^*\tilde{V}\|_\infty$
3. Finish if the two terms are each bounded by $\epsilon$
Theoretical guarantee

Corollary (Final precision and aggregation criterion)

The algorithm result \((\tilde{V}, \{S_k\})\) checks:

\[
\|V - V^*\|_\infty \leq \frac{2\epsilon}{1 - \gamma}
\]

and

\[
\forall k \in [1; K], \forall s, s' \in S_k, \ |V^*(s) - V^*(s')| \leq \frac{4\epsilon}{1 - \gamma}
\]
Adaptive Aggregation Value Iteration
Adaptive Aggregation Value Iteration
Adaptive Aggregation Value Iteration
Adaptive Aggregation Value Iteration

![Adaptive Aggregation Value Iteration Diagram]
Final Partition
Average runtime for variable $|S| \in [50 ; 1000]$ 
($A = 10, 10 \text{ exp/point}, \text{random MDP with 95\% non-zero, precision } 10^{-2}, \gamma = 0.99$)
Runtime comparison — Random MDPs

<table>
<thead>
<tr>
<th>Sparsity</th>
<th>Agg PI</th>
<th>Agg Value</th>
<th>Agg Q-Value</th>
<th>VI</th>
<th>PI</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>1.52</td>
<td>3.78</td>
<td>8.31</td>
<td>216.28</td>
<td>2.35</td>
</tr>
<tr>
<td>40%</td>
<td>2.72</td>
<td>4.06</td>
<td>21.94</td>
<td>783.85</td>
<td>4.36</td>
</tr>
<tr>
<td>70%</td>
<td>3.82</td>
<td>4.59</td>
<td>34.91</td>
<td>1311.30</td>
<td>6.28</td>
</tr>
<tr>
<td>100%</td>
<td>3.18</td>
<td>4.59</td>
<td>31.08</td>
<td>1048.93</td>
<td>8.85</td>
</tr>
</tbody>
</table>

Average runtime (s) for variable sparsity.

\( A = \{10, 50, 100\}, S \in [50 ; 1000], 5 \text{ exp/point}, \text{ random MDP, precision } 10^{-2}, \gamma = 0.99 \)
Discussion:

- Low sparsity $\Rightarrow$ slow classical Value Iteration
- Higher accuracy and greater discount $\Rightarrow$ our algorithm struggle
- Value Iteration $\approx$ Adaptive Aggregation Value Iteration for sparse MDPs?
Table of Contents

1. State Abstraction and Approximate Dynamic Programming

2. Adaptive Aggregation Value Iteration algorithm

3. Runtimes comparison

4. Conclusion
Conclusion on State Abstraction

We provided:

- An efficient Approximate algorithm to compute optimal Policy/Value Function
- Useful State Abstractions

Perspective:

- More simulations needed (more models, impact of $A$, bigger state space, lower $\epsilon$)
- Improve speed of the Policy Iteration-like algorithm
- Generalization over model-free problems
Thank you for your attention!


Adaptive aggregation methods for infinite horizon dynamic programming.


Hierarchical approaches.

State abstraction discovery from irrelevant state variables.
In *IJCAI*, volume 8, pages 752–757.

Towards a unified theory of state abstraction for mdps.
In *AI&M*.

*Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703.
John Wiley & Sons.

Reinforcement learning with soft state aggregation.
*Advances in neural information processing systems*, 7.

Feature-based methods for large scale dynamic programming. 