Foundations and Trends® in
Machine Learning
Vol. 1, No. 4 (2008) 403–565
© 2009 S. Mahadevan
DOI: 10.1561/2200000003

now
the essence of knowledge

Learning Representation and Control
in Markov Decision Processes: New Frontiers

By Sridhar Mahadevan

【强化学习 105】RPI

张楚珩
清华大学 交叉信息院博士在读

这里想讲一篇长论文，这篇论文提出一种 representation policy iteration （RPI） 的框架；同时还在强化学习表示学习方面做了比较详细的讲述。

原文传送门

Mahadevan, Sridhar. "Learning representation and control in Markov decision processes: New frontiers." Foundations and Trends® in Machine Learning 1.4 (2009): 403-565.

特色

提出 representation policy iteration （RPI），一边学习好的状态空间表示，一边利用所学习到的状态表示来学习策略。把所有的状态看做一个graph，学习状态空间的表示就是要找到这个graph上比较少的几个基函数，使得定义在这个graph上的任意一个函数都尽量可以用这几个基函数来表示。当学习到这样一个状态空间的表示之后，原来较为复杂的MDP就被转化为了一个更简单的MDP，我们可以在这个简单的MDP上求解最优策略。

当然，注意到这篇文章写在十多年前，不是很新；RPI 收敛的条件不是很清楚（表示是依赖于策略的，策略学习到最优的时候，表示可能只适用于该策略和奖励函数，which is a 1D subspace；关于这一点可以看完之后倒回来思考）；RPI 在复杂问题上的效果不是很清楚。

过程

1. 强化学习中的表示学习

动机

状态数目较少的时候，可以分别处理每个不同的状态（tabular case），这些强化学习算法的复杂度通常和状态的总数有关；当状态空间特别大的时候就不能再把每个不同的状态分别处理了，而是需要利用一些特征来表示这些状态，从而更有效地解决强化学习问题。

一些常用的状态表示方法（比如 polynomials、RBFs、NNs）通常从原始的特征出发，构建新的状态表示；这一类表示方法通常依赖于一个先验知识：各种函数相对于原始特征是比较平滑的。本文的方法则把各个状态之间的关系看做一个 graph，其依赖的先验知识则是各种函数在这个 graph 上是比较平滑的。这里提到的各种函数包括：reward function、transition function、value function 和 policy 等。

MDP 中的基底构造问题

基底构造的目标是找到一组基函数 $\Phi \in \mathbb{R}^{|S| \times k}$ ，它包括 k 个基函数；我们希望这一组基函数能够以较小的代价近似该 MDP 上的各种函数。

Definition 1.1. *Basis Construction Problem in MDPs*: Given a Markov decision process M, find an "optimal" basis matrix Φ that provides a "low-dimensional" representation of M, and enables solving M as "accurately" as possible with the "least" computational effort.

因此该问题中涉及到一些权衡：基底数目越少，在近似的 MDP 上的问题求解就会更加容易，但同时求解就会更加不精确。因此，我们的目标就是用尽可能少的基底来准确表示 MDP 上的各种函数。

特别地，在上述提到的各种函数中我们最为关心的是价值函数（value function）的函数拟合问题。我们注意到，即使没有任何先验知识（比如函数需要平滑），也有可能找出一组满足 $k<|S|$ 的基函数，使得 approximation error 为零。比如，假设这样一组基函数构成了某个空间，该空间内任何函数在 Bellman 算子 T^π 的作用后，仍然还在该空间内；我们称这样的空间为 invariant subspace。在这样的空间中，一定存在待求解的价值函数 V^π 。

$$T^\pi(V^\pi)(x) = V^\pi(x).$$

因此，在这篇文章中，基底构造问题也可以看做是构造这样的 invariant subspace。文章主要介绍了两大类方法来实现这件事情：diagonalization 和 dilation。前者在本专栏里面有提到过，用它来构造状态空间基函数比较著名的方法就是 PVF（张楚珩：【强化学习 67】Proto-value Function）。后者我们还不太熟悉，dilation 的方法分为两种：一种基于 Krylov space；一种基于 Drazin inverse 和 diffusion wavelet。名字听起来有点复杂，但是我们会在后面展开讲。

2. Average-reward MDP 以及 MDP 的结构

我们比较熟悉的是 finite horizon 和 infinite horizon + discount 的设定。这篇文章里面还提到了 infinite horizon + average-reward 的设定，这种设定对于马科夫过程的结构关系比较紧密。我们下面考虑一个固定策略下的 MDP，即 Markov reward process （MRP），记做 M=(P, R)。

Limiting matrix P*

Definition 2.8. The long-run transition or *limiting matrix* P^* is defined as

$$P^* = \lim_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n-1} P^k, \tag{2.4}$$

which can be shown to exist when S is finite or countable.

如果 MRP 是遍历和非周期的，那么该矩阵的每一行都一样，都等于稳态分布（invariant distribution），即

$$P^* = \lim_{k\to\infty} P^k = \mathbf{1}\rho, \tag{2.6}$$

Invariant distribution ρ

注意到如果 MRP 遍历，那么 P 的最大特征值（即，spectral radius）为 1，相应的左特征向量就是该稳态分布。

$$\rho P = \rho. \tag{2.5}$$

Gain g and bias h

当存在 discount rate 的时候，优化的目标为 discounted cumulative reward，它综合考虑了长远利益和近期的利益，通过一个 discount rate 来把长远利益和近期利益综合为一个指标。而在 average-reward 的设定下，这两种利益则会被分别考虑：其中 gain 表征长期平均利益，bias 表示实际收到利益相对于长期利益的差值的累计。

$$g(s) = P^* R(s). \tag{2.7}$$

$$h(s) = E\left(\sum_{t=1}^{\infty}(R(s_t) - g(s_t))\right), \tag{2.8}$$

比如 g(s) 表示从状态 s 出发，走无穷多步之后，平均每一步能够获取的平均利益；而在达到这个“无穷多步”之前，每一步获取利益的期望都可能并不等于极限平均利益，把这一部分差值累积起来就得到了 h(s)。当 MRP 遍历的时候，bias 部分还可以写作

$$h = \sum_{t=0}^{\infty}(P^t - P^*)R. \tag{2.9}$$

在 average-reward 的设定下，我们不仅需要最大化 gain 也希望最大化 bias。

Bellman equation

该设定下同样有 Bellman equation

Definition 2.11. Let M be an ergodic (or unichain) average-reward MDP. The *bias* or average-adjusted value of any policy π can be found by solving the equations:

$$h^\pi = R^\pi - g^\pi + P^\pi h^\pi, \tag{2.10}$$

where g^π is the gain associated with policy π. $R^\pi(s)$ is the expected one step reward received for taking action $\pi(s)$.

为了便于大家理解，给出一个简单的 MRP 的算例。

```
P = np.array([[0.3, 0.7], [0.6, 0.4]])
P
```

```
array([[0.3, 0.7],
       [0.6, 0.4]])
```

```
Pstar = P.copy()
for i in range(100):
    Pstar = np.matmul(Pstar, Pstar)
Pstar
```

```
array([[0.46153846, 0.53846154],
       [0.46153846, 0.53846154]])
```

```
R = np.array([[1.0], [2.0]])
R
```

```
array([[1.],
       [2.]])
```

```
g = np.matmul(Pstar, R)
g
```

```
array([[1.53846154],
       [1.53846154]])
```

```
H = np.zeros((2, 2))
Pt = np.eye(2)
for i in range(1000):
    H += Pt - Pstar
    Pt = np.matmul(Pt, P)
h = np.matmul(H, R)
```

```
H
```

```
array([[ 0.41420118, -0.41420118],
       [-0.35502959,  0.35502959]])
```

```
h
```

```
array([[-0.41420118],
       [ 0.35502959]])
```

```
R - g + np.matmul(P, h)
```

```
array([[-0.41420118],
       [ 0.35502959]])
```


3. Generalized Inverse

如果一个方阵不是满秩的，那么它就不可逆；不过此时我们可以定义广义逆（generalized

inverse）。注意到一个真正的逆满足以下所有条件，但当方阵不是满秩时，不能找出一个逆满足这所有的条件，根据逆所满足的不同条件可以定义不同的广义逆。

Definition 3.4. A generalized inverse X of a matrix $A \in \mathbb{C}^n \times \mathbb{C}^n$ is a matrix that satisfies one or more of the following properties:

(1) $AXA = A$.
(2) $XAX = X$.
(3) $(AX)^* = AX$.
(4) $(XA)^* = XA$.
(5) $AX = XA$.
(6) $A^{k+1}X = A^k$.

where $()^*$ denotes the conjugate transpose.

A generalized inverse matrix X of A that satisfies a subset $C \subset \{1,2,3,4,5,6\}$ of these properties is labeled a A^C inverse.

非常著名的 Moore-Penrose pseudo-inverse 满足前四条，即 $A^\dagger = A^{\{1,2,3,4\}}$ ；group inverse 满足其中的三条，即 $A^\# = A^{\{1,2,5\}}$ ；Drazin inverse 也满足其中的三条，即 $A^D = A^{\{2,5,6\}}$ 。

3. Laplacian 算子

在专栏前面的文章里面我们看到可以在图上定义若干种不同的 Laplacian：

Table 1.1 Some Laplacian operators on undirected graphs. W is a symmetric weight matrix reflecting pairwise similarities. D is a diagonal matrix whose entries are row sums of W. All these operators are represented by matrices whose row sums are 0 and have non-positive off-diagonal entries.

Operator	Definition	Spectrum
Combinatorial Laplacian	$L = D - W$	$\lambda \in [0, 2\max_v d_v]$
Normalized Laplacian	$\mathcal{L} = I - D^{-1/2}WD^{-1/2}$	$\lambda \in [0,2]$
Random Walk Laplacian	$L_r = I - D^{-1}W$	$\lambda \in [0,2]$

如果把不同的状态看做图上的节点，当给定一个概率转移矩阵 P 时，就对应了一个 MC；由此，可以定义相应的 random walk Laplacian 算子 $\mathbf{L} = I - P$ 。同时注意到价值函数可以由相应的 Laplacian 算子表示。

Definition 3.3. The *interest rate* $\rho \equiv (1-\gamma)\gamma^{-1}$ or equivalently, $\gamma = \frac{1}{1+\rho}$. The interpretation of ρ as an interest rate follows from the property that, if a reward of 1 is "invested" at the first time step, then $1+\rho$ is the amount received at the next time step. The interest rate

formulation of the discounted value function associated with a policy π can be written as

$$V^\pi = (I - \gamma P^\pi)^{-1} R^\pi = (1+\rho)(\rho I + \mathbb{L}^\pi)^{-1} R^\pi, \tag{3.3}$$

where $\mathbb{L}^\pi = I - P^\pi$ is the Laplacian matrix associated with policy π.

Laplacian 算子的 group inverse 可以被写为

Definition 3.6. The *group inverse* of the Laplacian $\mathbb{L} = I - P$ of a Markov chain with transition matrix P is defined as

$$\mathbb{L}^\# = (I - P + P^*)^{-1} - P^*. \tag{3.7}$$

其中，中间求逆的部分被称作 fundamental matrix，它是可逆的：

Theorem 3.3. The eigenvalues of the matrix $P - P^*$ of an ergodic Markov chain lie within the unit circle. Consequently, the *fundamental matrix* [110, 121] associated with P

$$\mathbb{L} + P^* = I - P + P^* \tag{3.6}$$

is invertible.

Laplacian 算子的 Drazin inverse 的计算稍微复杂一点，假设 transition matrix 可以做如下分解

Definition 3.10. Let a transition matrix P of a Markov chain be defined on a finite state space S, and suppose P induces m recurrent classes on S. Then, P can be decomposed into the following form:

$$P = W \begin{pmatrix} I & 0 \\ 0 & Q \end{pmatrix} W^{-1}, \tag{3.13}$$

where W is nonsingular, I is an $m \times m$ identity matrix, and Q is an $|S| - m \times |S| - m$ times matrix. The inverse $(I - Q)$ exists since 1 is not an eigenvalue of Q. Also $\lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n-1} Q^k = 0$, since Q is nilpotent.

[5] N is a nilpotent matrix if for some nonnegative integer k, $N^k = 0$.

那么相应的 Drazin inverse 可以被写为：

Definition 3.11. Given an arbitrary transition matrix P on a finite state space S, the Drazin inverse $\mathbb{L}^D$ of $\mathbb{L}$ is given by

$$\mathbb{L}^D = (I - P)^D = W \begin{pmatrix} 0 & 0 \\ 0 & (I - Q)^{-1} \end{pmatrix} W^{-1}. \quad (3.14)$$

文章给出了相应的计算 Drazin inverse 的方法，输入 Laplacian 矩阵，输出相应的 Drazin inverse。这里不再贴出来了。

那么我们为什么要计算 Laplacian 的逆呢？因为价值函数可以被表示为 Laplacian 逆的多项式和，而高阶多项式的系数又衰减地比较快，因此可以用 Laplacian 逆对应较大特征值的特征向量来作为我们需要的基底，从而对图上的函数有较好的近似。

Theorem 3.5. Given a discounted MDP M and policy π, the value function V^π can be expressed in terms of the *gain* and *bias* of the associated Markov reward process $M_\pi = (P^\pi, R^\pi)$, and the Drazin inverse of the Laplacian, as follows

$$V^\pi = (1 + \rho) \left(\rho^{-1} g^\pi + h^\pi + \sum_{n=0}^{\infty} (-\rho)^n ((\mathbb{L}^\pi)^D)^{n+1} R^\pi \right). \quad (3.21)$$

4. 希尔伯特空间

在无向图上定义 RKHS

通过在无向图上定义 RKHS，说明从数学上应该如何描述函数在图上的平滑性。

对于一个无向图，其邻接矩阵为 W。Laplacian 算子在图上定义了半范数，“半”是因为它对于任意的常值函数，其范数都为零。

Definition 3.12. The Laplacian $L = D - W$ defines a *semi-norm* over all functions on an undirected graph G

$$\langle f, g \rangle = f^T L g. \quad (3.25)$$

Furthermore, the length of a function is defined as $\|f\| = \sqrt{\langle f, f \rangle}$.

这个半范数反映了一个函数在图上的不平滑程度，可以从下面这个公式看出来

Definition 3.13. The *Dirichlet sum* is defined as

$$f^T L f = \sum_{(u,v) \in E} w_{uv} (f(u) - f(v))^2 . \quad (3.26)$$

其中 w 为其邻接矩阵的元素。我们注意到，Laplacian 矩阵一定会有一个或者多个特征值为 0，这表示该图有多少个连通区域。把不同连通区域之间的函数自由组合的自由度排除之后，可以定义在图上的希尔伯特空间。

Definition 3.14. Let the eigenvalues of L on a graph G with r connected components be ordered as $\lambda_1 = 0, \ldots, \lambda_r = 0$, $\lambda_{r+1} > 0, \ldots, \lambda_n > 0$. Let the associated eigenvectors as $u_i, 1 \leq i \leq n$. The Hilbert space associated with a graph G is defined as

$$\mathbb{H}(G) = \{g : g^T u_i = 0, 1 \leq i \leq r\}. \tag{3.27}$$

由此，定义 Laplacian 矩阵的伪逆，并且该伪逆就是再生希尔伯特空间（RKHS）的再生核。

Definition 3.15. The *pseudo-inverse* of Laplacian $L = D - W$ is defined as

$$L^+ \equiv \sum_{i=r+1}^{n} \frac{1}{\lambda_i} u_i u_i^T, \tag{3.28}$$

where the graph G is assumed to be undirected and unweighted, and having r connected components.

Theorem 3.6. The reproducing kernel associated with the RKHS $\mathbb{H}(G)$ of an undirected unweighted graph G is given by

$$K = L^+. \tag{3.29}$$

回忆一下，再生核的含义

$$g(i) = e_i L^+ L g = K(:, i) L g = \langle K(:, i), g \rangle,$$

当我们利用 Laplacian 算子在图上定义了一个 RKHS 之后，图上的函数拟合问题的原则就是找到一个最光滑的函数，即 minimum-norm interpretation。（可以参考本专栏前面讲的一个平均场方法求解半监督学习的文章）

Definition 3.16. Given an undirected graph G, whose Laplacian $L = D - W$, given samples of a function f on l vertices of G, the minimum-norm interpolant in $\mathbb{H}(G)$ to f is given as follows

$$\min_{g \in \mathbb{H}(G)} \{\|g\| : g_i = f_i, i = 1, \ldots, l\}. \tag{3.31}$$

在 MDP 上定义希尔伯特空间

通过在 MDP 上定义 RKHS，说明把状态空间上的函数定义为一组特征的线性组合的合理性。

对于状态空间的任意两个函数，定义这两个函数的内积如下：

Definition 4.1. The space of all value functions on a discrete MDP forms a Hilbert space under the inner product induced by the invariant distribution ρ^π of a specific policy π, where

$$\langle V_1, V_2 \rangle_{\rho^\pi} = \sum_{s \in S} V_1^\pi(s) V_2^\pi(s) \rho^\pi(s). \tag{4.1}$$

The "length" or norm in this inner product space is defined as $\|V\|_{\rho^\pi} = \sqrt{\langle V, V \rangle_{\rho^\pi}}$.[1]

其中，对于每个点都有一个权重的调整，该权重为在该策略下的稳态状态分布（invariant distribution）。任意一个函数可以往，这个希尔伯特空间中的一个子空间做投影

Definition 4.3. If $\phi_1, \ldots, \phi_k$ is a basis for the subspace $\mathcal{K}$, the projection operator $\Pi_\mathcal{K}$ operator can be written as

$$\Pi_\mathcal{K}(u) = \sum_{i=1}^{k} \langle u, \phi_i \rangle_\mathcal{H} \phi_i. \tag{4.3}$$

后面文章又讲了 Bellman 算子和该投影算子的复合算子也是 contraction，即在某一个固定的基上可以做 approximated policy evaluation 和 control。

考虑 RKHS 的定义，一个定义在状态空间上的函数可以用再生核表示出来

$$V(x) = \langle V, K(x, .) \rangle_\mathcal{H},$$

我们可以使用一组状态的特征来表示再生核

$$K(x, y) = \langle K(x, .), K(y, .) \rangle_\mathcal{H}.$$

$$K(x, y) = \langle \phi(x), \phi(y) \rangle_\mathcal{H}.$$

由此，任意一个函数都可以用一组特征和相应的系数表示

$$\hat{V}^{\pi}(s) \approx \langle w, \phi(s) \rangle_{\mathcal{H}}, \tag{4.12}$$

5. MDP 上的表示学习问题

状态空间的低维表示使得 MDP 的求解变得容易

给定一个 MDP 状态空间上的低维表示和一个固定的策略之后，我们可以定义一个 approximated MRP：

Definition 5.1. Given a basis matrix Φ and policy π, the induced *approximate reward* function R_{Φ}^{π} and *approximate model* P_{Φ}^{π} are defined as:

$$P_{\Phi}^{\pi} = (\Phi^T \Phi)^{-1} \Phi^T P^{\pi} \Phi,$$
$$R_{\Phi}^{\pi} = (\Phi^T \Phi)^{-1} \Phi^T R^{\pi}.$$

我们可以知道，在这个 MRP 上求解的结果等价于在原 MRP 上求解的 fixed point solution 再投影到该低维表示空间上。

Theorem 5.1. (Parr et al. [104]) Given a basis matrix Φ, the exact solution to the approximate policy evaluation problem defined by P_{Φ}^{π} and R_{Φ}^{π} is the same as that given by the fixed point solution of the exact policy evaluation problem defined by P^{π} and R^{π} onto the basis Φ.

在低维空间上求解的好处是其复杂度更低：

Lemma 5.2. Given an approximate model P_{Φ}^{π} and reward function R_{Φ}^{π} induced by a basis Φ, the exact policy evaluation problem defined by P^{π} and R^{π} is reduced from its original complexity of $O(n^3)$ by the basis Φ to $O(k^3)$.

MDP 表示学习的若干考虑

- 学习过程的复杂度：注意到我们进行表示学习的目的还是为了更好的求解 MDP。因此，如果我们的目标是针对一个特定的 MDP 求解，那么我们显然希望学习到相应表示的复杂度不能高于求解该 MDP 的复杂度；如果我们的表示能够被迁移到其他类似 MDP 上，该条件可以被放宽。
- 表示的复杂度：如果学习到的表示的表征能力比较强，当然使用该表示带来的性能损失会比较少，但预测同时在该表示空间内进行求解的复杂度会相应更高。因此，我们的目标就是以较为简单的表示来实现更小的近似误差。相应的方向有：找到更稀疏的表示；针对状态空间的特殊结构做分解，从而更有效地进行状态的表示；本文后面将会提到的，一些更有效的表示方法，比如多层次的小波基（multiscale wavelet bases）；对于连续状态空间，我们一般用一些空间上的样本来进行 non-parametric 的表示，那么我们希望找到好的采样和拟合方法，使得我们储存较少的样本就能达到较好的精度。

- 表示适用于单一奖励函数还是多种奖励函数：如果学习到的表示是针对单一奖励函数的，那么该表示可能能够对于特定的子空间做到更有效的表示；如果学习到的表示能够针对多个（或者任意）奖励函数，那么学习到的表示就能够适应于一族学习任务。
- 表示是针对单一策略还是多个策略：当策略给定之后，MDP 就退化为 MRP，针对该策略就更容易学习到具有理论保证的状态表示；但与此同时，在 RPI 中，策略是在迭代的，这样学习到的状态表示的生命周期就比较短。
- 逐步学习或者是一次性学习（incremental or batch）：文章里面讲的是一组基底中的每一个基底是一个一个构造出来的还是一次性全部构造出来的；而我这里想说的是，表示是一次性就学习到，还是通过一个循环迭代逐渐学习到。比如，在 policy iteration 中，策略就是通过迭代逐步学习。

6. 一个古老的算法：adaptive state aggregation

说它古老是因为该算法发表于 1988 年 [1]。考虑对状态做 aggregation，即把整个状态空间分为互不相交的 k 个集合，这样基底表示矩阵 Φ 的每行都是 one-hot 向量。该算法的目标是在给定完整的 transition function、 reward function 和固定策略的情况下，找到一个最优的 state aggregation，使得价值函数能够在该 aggregation 基底下较准确地被表示出来。算法的迭代过程如下：

1. 按照当前的 Bellman residual $T^\pi V^k - V^k$ 的数值大小，把状态分到不同的集合中，得到表示矩阵 Φ；
2. 按照该表示矩阵 Φ，把当前的价值函数做投影 $V^k \leftarrow \Pi_\Phi(V^k)$；
3. 调整价值函数，将它往奖励更高的方向移动： $V^{k+1} \leftarrow V^k + \Phi y,\ y = (I - \gamma P_\Phi^\pi)^{-1} R_\Phi^\pi$，其中

$$
\begin{aligned}
P_\Phi^\pi &= \Phi^\dagger P^\pi \Phi, \\
R_\Phi^\pi &= \Phi^\dagger (T(V^k) - V^k).
\end{aligned}
$$

注意到，这里 R_Φ^π 是 Bellman residual 的往基底上的投影，MDP 原始的奖励函数隐含在 Bellman 算子 T 中。在第二步中，价值函数在 Φ 张成的空间上；而第三步的修正仍然也在该空间中，因此第三步得到的价值函数仍然在 Φ 张成的空间上。这里比较特别的是，第一步中根据不同点上 Bellman residual 的大小来做 state aggregation 的做法。其原因如下：每步迭代后的误差可以被写作以下两项的和

$$
\begin{aligned}
E_1 &= (I - \Pi)(T(V^k) - V^k), \\
E_2 &= \gamma (I - \Pi) P^\pi \Phi y,
\end{aligned}
$$

where $\Pi = \Phi\Phi^\dagger$ is the orthogonal projection onto the range of Φ.

其中，如果 Φ 张成了相对于 P 的 invariant subspace 的话，第二项将会为零，即 $(I - \Pi) P \Phi x = 0, \forall x$，当然，对于 state aggregation 来说很难找到一个 invariant subspace。而第一项可以看做是每一个集合内对应状态的 Bellman residual 相比于其集合内平均的差值；为了减小这一项，自然考虑到把 Bellman residual 相近的状态划分到同一类中。

7. Diagonalization

在价值函数表示中，拉普拉斯矩阵的最小特征值对应最重要的特征向量

考虑拉普拉斯矩阵的特征值分解

$$
\mathbb{L}^\pi = \sum_{i=1}^{n} \lambda_i^\pi \phi_i^\pi (\phi_i^\pi)^T,
$$

同时价值函数也做相应的分解

$$R^\pi = \Phi^\pi \alpha^\pi, \tag{6.1}$$

不难推导出，价值函数可以被分解为相应特征向量的加权和，而特征值最小的特征向量对应的系数更小，这说明我们可以选用特征值最小的那一些特征向量来作为价值函数的基底

$$\begin{aligned} V^\pi &= \sum_{i=0}^{\infty} (\gamma P^\pi)^i \Phi^\pi \alpha^\pi \\ &= \sum_{k=1}^{n} \sum_{i=0}^{\infty} \gamma^i (1-\lambda_k^\pi)^i \phi_k^\pi \alpha_k^\pi \\ &= \sum_{k=1}^{n} \frac{1}{1-\gamma(1-\lambda_k^\pi)} \phi_k^\pi \alpha_k^\pi \end{aligned}$$

从图论的角度来看，拉普拉斯矩阵的最小特征值对应函数在图上最平滑的分量

考虑拉普拉斯算子的特征值 λ_i 和特征向量 e_i，定义一个函数在图上的 norm

$$\|f\|_2^2 = \sum_{v \in V} |f(v)|^2 d(v).$$

其中，d 为相应 MC 在图上的稳态分布；一个函数在图上的光滑程度可以通过如下 Sobolev norm 来衡量

$$\begin{aligned} \|f\|_{\mathcal{H}^2}^2 &= \|f\|_2^2 + \|\nabla f\|_2^2 \\ &= \sum_{v \in V} |f(v)|^2 d(v) + \sum_{u \sim v} |f(u) - f(v)|^2 w(u,v). \end{aligned} \tag{6.3}$$

注意到第一项衡量了这个函数的绝对数值大小；第二项衡量这个函数在图上的平滑程度，注意到对于特征向量 e_i，有 $\|\nabla e_i\|_2^2 = \lambda_i$，由此可见，对于同样的 ϵ-approximation

$$\left\| f - \sum_{i \in S(\epsilon)} \alpha_i e_i \right\| \leq \epsilon$$

显然应该尽量让对应特征值更小的特征向量前面的系数更大，这样的拟合产生更平滑的函数。文章中还提到，可以一个一个地来找到最平滑的特征函数，具体做法就是把特征值问题通过 Rayleigh quotient 表示为优化问题，比如

$$\lambda_1 = \inf_{f \perp \sqrt{D}\mathbf{1}} \frac{\sum_{u \sim v} (f(u) - f(v))^2 w_{uv}}{\sum_u f^2(u) d_u}.$$

这一节后面还讲述了对于具有特定结构的图，可以把图做分解（factor），然后其总的 Laplacian 也可以最相应的分解，从而简化表示；这里我暂时不是很感兴趣，就不写了。

8. Dilation

Krylov space

一个 Krylov subspace 是由一个算子 T 和一个函数 f 来定义的，它是如下基函数的 span

Definition 7.1. The jth *Krylov* subspace $\mathcal{K}_j$ generated by an operator T and a function f is the space spanned by the vectors:

$$\mathcal{K}_j = \{f, Tf, T^2 f, \ldots, T^{j-1} f\}. \tag{7.1}$$

Clearly, $\mathcal{K}_j \subset \mathbb{C}^N$. Note that $\mathcal{K}_1 \subseteq \mathcal{K}_2 \subseteq \cdots$, such that for some $m, \mathcal{K}_m = \mathcal{K}_{m+1} = \mathcal{K}$. Thus, $\mathcal{K}$ is the T-invariant Krylov space generated by T and f. When T is completely diagonalizable, the projections of f onto the eigenspaces of T form a basis for the Krylov space $\mathcal{K}$ [88].

Theorem 7.1. If a matrix $T \in \mathbb{C}^n \times \mathbb{C}^n$ is diagonalizable, and has n distinct eigenvalues, the nontrivial projections of f onto the eigenspaces of T form a basis for the Krylov space generated by T and f.

文章里面主要讲了两种方法，一种令 $T=\mathbb{L}$ ，即直接把 Laplacian 作为这里的算子 T；另外一种令 $T=\mathbb{L}^p$ ，即把 Laplacian 的 Drazin inverse 作为这里的算子 T。当然，我们一般不会一直重复直到找到一个 invariant subspace，因此一般来说 invariant subspace 的基底数量是非常多的。我们希望只需要前几项就能够较好的近似整个空间中可能的价值函数。为什么能做这样的近似呢？其主要的原因还是在于价值函数可以做如下的级数展开，因此基底中有几个基，就能够准确拟合前几项的结果。

$$V^\pi = (I - \gamma P^\pi)^{-1} R^\pi = (1+\rho)(\rho I + \mathbb{L}^\pi)^{-1} R^\pi, \tag{3.3}$$

$$V^\pi = (1+\rho)\left(\rho^{-1} g^\pi + h^\pi + \sum_{n=0}^{\infty} (-\rho)^n ((\mathbb{L}^\pi)^D)^{n+1} R^\pi\right). \tag{3.21}$$

注意到 $(a+x)^{-1} = a^{-1} - a^{-1}x + a^{-3}x^2 + \cdots$ ，也可以被写作级数形式，但是注意到当 a<1 时，前几项级数并不能很好的进行拟合；这意味着用 Laplacian 的效果会不如用 Laplacian 的逆的效果，这一点在后面的实验中也可以看到。

Dilation using Laplacian

这种方法就是按照如下顺序构建基

Definition 7.2. Given the Laplacian matrix $\mathbb{L}^\pi$ and reward function R^π, the jth *Krylov* subspace $\mathcal{K}_j$ is defined as the space spanned by the vectors:

$$\mathcal{K}^\pi{}_j = \{R^\pi, \mathbb{L}^\pi R^\pi, (\mathbb{L}^\pi)^2 R^\pi, \ldots, (\mathbb{L}^\pi)^{j-1} R^\pi\}. \tag{7.2}$$

Note that $\mathcal{K}^\pi{}_1 \subseteq \mathcal{K}^\pi{}_2 \subseteq \cdots$, such that for some $m, \mathcal{K}^\pi{}_m = \mathcal{K}^\pi{}_{m+1} = \mathcal{K}^\pi$. Thus, $\mathcal{K}^\pi$ is the $\mathbb{L}^\pi$-invariant Krylov space generated by $\mathbb{L}^\pi$ and R^π.

显然，这样构建的各个基底之间不正交。可以利用 Bellman error basis function（BEBF），每次把 Bellman error 作为新的基底

$$\phi_{k+1} = \mathrm{BE}(\Phi).$$

$$\mathrm{BE}(\Phi) = T(V^\pi_\Phi) - V^\pi_\Phi = R^\pi + \gamma P^\pi \Phi w^\pi_\Phi - \Phi w^\pi_\Phi.$$

不难看出，每次构造的新基底都和已有基底张成的空间正交；同时如果这里也从 R^π 出发，可以证明上述方法构造出来的基底和用 Laplacian 构造出来的 Krylov space 一致。（另外可以参见本人数学专栏里面讲的 Lanczos 和 conjugate gradient 方法，想法上和这个很类似）

为了给大家有一个直观的感受，举一个例子：考虑一个一维链式 MRP，有 50 个状态，在第 10 和第 41 个状态上有奖励，策略为最优策略。在这种情况下，基底 Krylov space 的前几个基函数画出来长下面这个样子

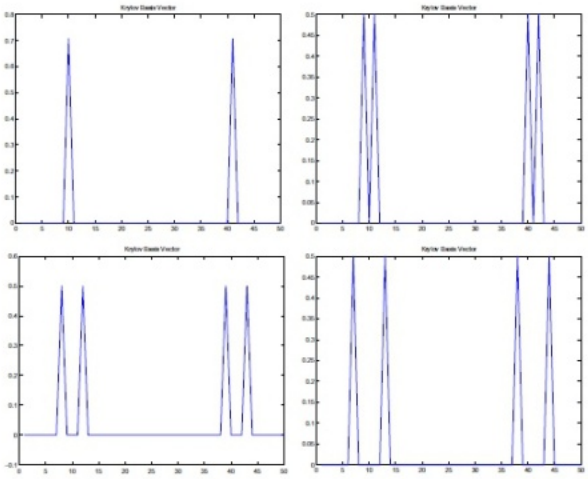


Fig. 7.1 The first four Krylov bases for a chain MDP with 50 states. Rewards of +1 are given in states 10 and 41. Actions are to go left or right, each succeeds with probability 0.9. The Laplacian associated with the optimal policy is used to dilate the reward function.

可以看出，这个基底就是每次按照 Laplacian 把奖励函数往外扩散。直观看起来效率是比较低的。

再举一个二维的例子，感觉上也是类似的。

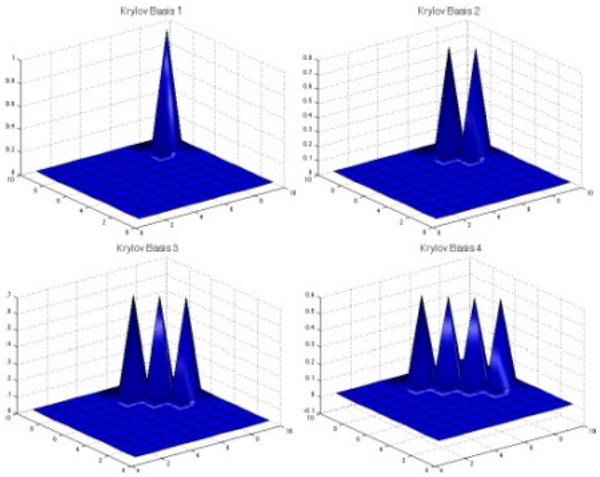


Fig. 7.2 The first four Krylov bases for the "two-room" environment based on dilation of the reward function using the Laplacian of a policy.

Dilation using Drazin inverse of Laplacian

下面，我们介绍如何使用 Laplacian 的 Drazin inverse 来构造 Krylov space（也叫 Drazin space）。

Definition 7.3. The *Drazin space* associated with a policy π and reward function R^π is defined as the space spanned by the set of Drazin vectors:

$$\mathbb{D}_m^\pi = \{P^* R^\pi, (\mathbb{L}^\pi)^D R^\pi, ((\mathbb{L}^\pi)^D)^2 R^\pi, \ldots, ((\mathbb{L}^\pi)^D)^{m-1} R^\pi\}. \quad (7.3)$$

Note that $\mathcal{D}^\pi{}_1 \subseteq \mathcal{D}^\pi{}_2 \subseteq \cdots$, such that for some $m, \mathcal{D}^\pi{}_m = \mathcal{D}^\pi{}_{m+1} = \mathcal{D}^\pi$. Thus, $\mathcal{D}^\pi$ is the $(\mathbb{L}^\pi)^D$-invariant Krylov space generated by $\mathbb{L}^\pi$ and R^π.

这第一个向量看起来有点奇怪，注意到在 average-reward 的设定下，gain $g^\pi = P^* R^\pi$ ，我们把它作为起始的“种子”向量；而接下来的一项 $\mathbb{L}^D g = \mathbb{L}^D P^* R = 0$ ，因此就把这一项跳过去了。

Theorem 7.2. The product $(P^\pi)^*(\mathbb{L}^\pi)^D = 0$.

还是前面的例子，把相应的 Drazin space 中前几个基底的图如下所示。可以看出相应的前几个基底更快地“扩散”到状态空间中。后面会看到，这样的基底近似效果会更好

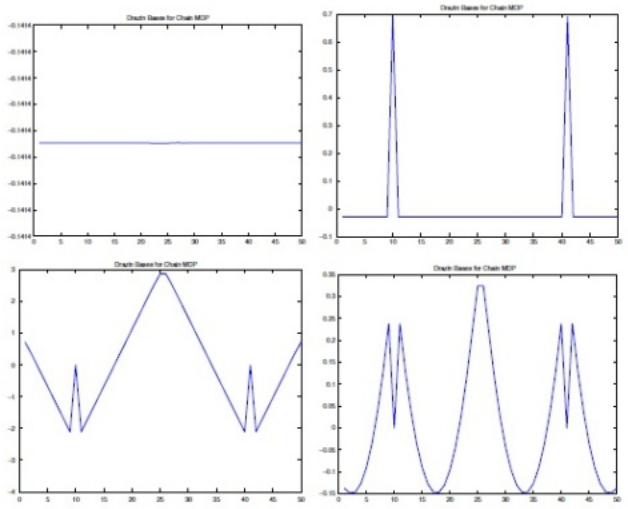


Fig. 7.3 The first four Drazin bases for a chain MDP with 50 states. Rewards of +1 are given in states 10 and 41. Actions are to go left or right, each succeeds with probability 0.5. The Laplacian associated with the optimal policy is used to dilate the reward function.

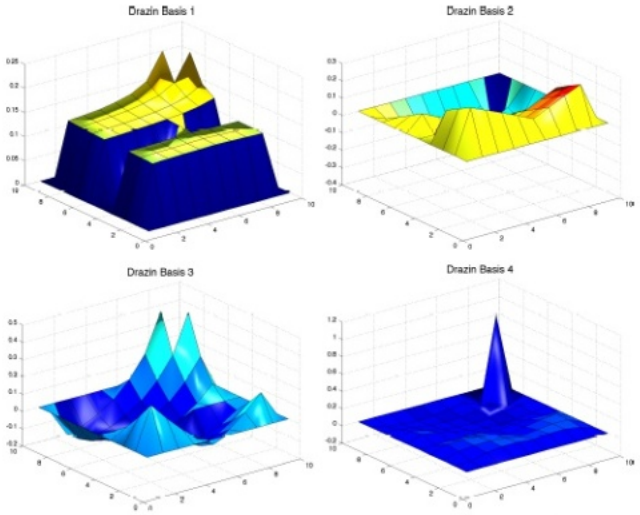


Fig. 7.4 Drazin bases for the "two-room" environment based on dilation of the reward function using the Drazin inverse of the Laplacian of a policy.

Diffusion wavelet

文章接着后面还讲了使用 diffusion wavelet 的做法来获得多尺度上更好的近似，大致思想上是我们现在不要再一个一个地构造 Drazin space 了，而是要 $\mathbf{L}^D R, (\mathbf{L}^D)^2 R, (\mathbf{L}^D)^4 R, (\mathbf{L}^D)^8 R \ldots$ 地构造。但是 wavelet 看得我很头痛，并且看懂之后发现这个解决问题的设定我就不是很感兴趣，因此就不再写了。关于 wavelet 的简要介绍见下一篇吧。

下面贴一下这种方法的实验效果，文章把这种方法和前面的 eigenvalue 的方法进行对比，我们发现只使用 5 个基底，该方法不仅能够很好得近似比较不光滑的函数（比如第一行中的函数），也能够很好地近似比较光滑的函数（比如第二行中的函数）。与此同时，随着基底数量的增加，拟合误差下降地也更快，即这种方法的近似效率更高。

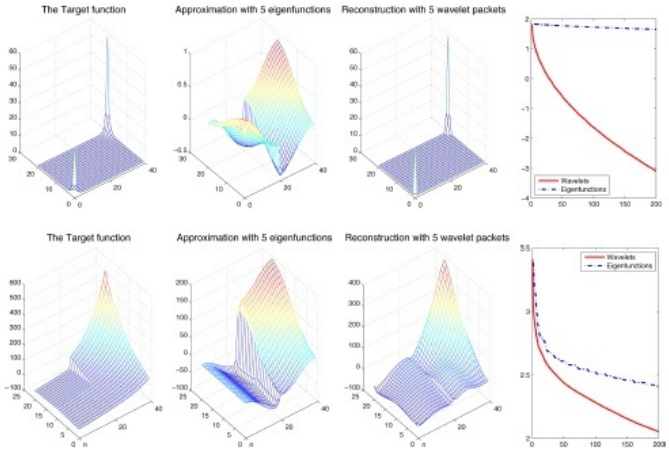


Fig. 7.9 *Left column*: Target functions. *Middle two columns*: Approximations produced by 5 diffusion wavelet bases and Laplacian eigenfunctions. *Right column*: Least-squares approximation error (log scale) using up to 200 basis functions (*Bottom curve*: Diffusion wavelets; *Top curve*: Laplacian eigenfunctions).

9. Continuous MDP

这里讲一下如何处理连续状态空间的表示学习问题（假设行动还是离散的）。状态空间连续的时候，可以定义相应的 Bellman 方程，称作 Hamilton-Bellman-Jacobi （HBJ）方程：

$$Q^*(s,a) = \int_{s'} P^a_{ss'} \left(R^a_{ss'} + \max_{a'} \gamma Q^*(s',a') \right) \mathrm{d}s'.$$

回顾一下，在状态空间离散的时候，可以把每个状态当做图上的节点；我们在前面看到，这个图上对应 Laplacian 的前几个特征向量表示了最平滑的若干个定义在该图上的函数。当状态连续的时候，就不能再把它们看做图上的节点了；这时候我们把它们建模为一个 Riemannian manifold，在这个 manifold 上也可以定义类似的 Laplacian，我们称之为 Laplace-Beltrami 算子，该算子的前几个特征向量同样对应最平滑的函数。下面将简要介绍 Riemannian manifold 和 Laplace-Beltrami 算子。

Riemannian manifold

这一部分只是看了一下定义，可能个人理解不是特别准确。个人理解，manifold 用来被描述一个在

局部和（低维）欧式空间有一些对应关系的结构。除此之外，Riemannian manifold 还要求 manifold 和欧式空间的对应关系是可导的（相当于规定了平滑性），同时还能够通过 Riemannian metric 来定义距离度量。具体地，对于 n 维空间上的 manifold 上的一点 $p \in \mathcal{M}$ ，考虑过这一点的 tangent space $T_p(\mathcal{M})$ ，在该空间上可以定义内积 $g_p : T_p(\mathcal{M}) \times T_p(\mathcal{M}) \to \mathbb{R}$ ，当然也可以把它写出一个 nxn 的矩阵 G_p （Riemannian metric）。对于 tangent space 上的一点 q，可以写出 p、q 两点在 manifold 上的距离度量 $(q-p)^T G_p (q-p)$ 。比如对于 $\mathbb{R}^n$ 本身也可以看做一个 manifold，其在任意点上的 $G_p = I_n$ ，因此两点之间的距离就是欧氏距离；对于一个 probability distribution $P(X|\theta), \theta \in \mathbb{R}^n$ ，它在某一点的距离度量 $G_\theta = \mathcal{I}(\theta)$ 为 Fisher information matrix。回忆一下，TRPO 里面需要限制参数变化带来的策略空间的变化大小，实际上限制了其 manifold 上的距离 $(\theta' - \theta)^T \mathcal{I}(\theta)(\theta' - \theta)$ 。

Laplace-Beltrami 算子

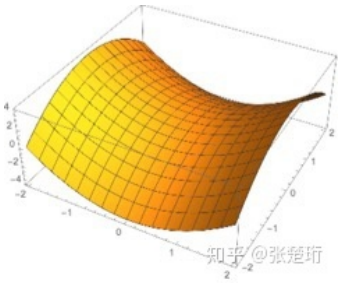
可以在 manifold 上定义 Laplacian-Beltrami 算子

$$\Delta = \operatorname{div} \operatorname{grad} = \frac{1}{\sqrt{\det g}} \sum_{ij} \partial_i \left(\sqrt{\det g}\, g^{ij} \partial_j \right),$$

其中 g 就代表前面提到的 Riemannian metric。这个看起来挺复杂的，不过没事，这就是一个定义而已，在 $\mathbb{R}^n$ 空间中，它可以被写作

$$\Delta = \sum_i \frac{\partial^2}{\partial x_i^2}$$

满足 $\Delta f = 0$ 的函数被称为 harmonic functions，比如在 $\mathbb{R}^2$ 空间中，如下函数就是一个 harmonic function。



同样，该算子也有相应的特征值和特征向量，即

$$\Delta \phi = \lambda \phi.$$

在 manifold 上也可以定义相应的 smoothness functional $\mathcal{H}^1(\mathcal{M})$ -norm

$$\mathcal{H}^1(\mathcal{M}) = \left\{ f \in \mathbb{L}^2(\mathcal{M}) : \|f\|_{\mathcal{H}^1(\mathcal{M})} := \|f\|_{\mathbb{L}^2(\mathcal{M})} + S(f) \right\}. \qquad (9.1)$$

其中

$$S(f) \equiv \int_{\mathcal{M}} |\nabla f|^2 \, d\mu = \int_{\mathcal{M}} f \Delta f \, d\mu = \langle \Delta f, f \rangle_{\mathbb{L}^2(\mathcal{M})},$$

不难发现，对于归一化的特征向量 $S(\phi)=\lambda$ 。因此，相应地，要找到 manifold 上平滑的函数就是要找 Laplace-Beltrami 算子对应较小特征值的特征向量们张成的空间。

Hodge theorem

Hodge theorem 说的是 Laplace-Beltrami 算子对应的特征们能够张成的空间就是 manifold 上所有函数构成的空间，这说明如果我们选用该算子的特征向量来作为状态空间的基，当选用的特征向量足够多的时候，其张成的空间就是状态空间中的所有函数空间，即 asymptotic approximation error = 0。

Theorem 9.1 (Hodge [115]). Let $(\mathcal{M}, g)$ be a smooth compact connected oriented Riemannian manifold. The spectrum $0 \leq \lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_k \leq \cdots, \lambda_k \to +\infty$, of Δ is discrete, and the corresponding eigenfunctions $\{\phi_k\}_{k\geq 0}$ form an orthonormal basis for $\mathbb{L}^2(\mathcal{M})$.

Manifold 的表示方法

回顾状态是离散的时候，我们是如何表示学习到的基底的呢？我们使用了一个矩阵 $\Phi \in \mathbb{R}^{|\mathcal{S}|\times k}$ 来表示每一个状态上这 k 个基函数的数值。当状态空间变为连续的时候，我们不能再用这样一个矩阵来表示所学习到的基底了，因为 $|\mathcal{S}| \to \infty$ 。这个时候，我们需要在有限个样本上表示这 k 个基底，并且当出现新的一个状态的时候，我们通过外拓来找到新的状态下各个基底的数值。这涉及到两个问题：如何选择这有限个支撑状态样本；通过何种方式来外拓得到未见过状态的各个基底的数值。

对于第一个问题，文章说可以纯粹在已有的样本中做 random sample，也可以构造 ϵ -net。对于第二个问题，文章提出使用 Nystrom extension

$$\hat{\phi}_m(x) = \frac{1}{\hat{\lambda}} \sum_{i=1}^{n} w_i k(x, s_i) \hat{\phi}_m(s_i). \tag{9.5}$$

其中，k(x,s) 衡量了这个新数据点到已有数据点的相似性（kernel function），w 为quadrature weight，文章也没说清楚这个东西到底咋算，不过大致不难理解，因为用了很多数据点做 support，那么各个数据点的重要性肯定还是区别的，需要用一个权重来作相应的调整。

Remark：在离散状态空间中选择一组基底，会优先选择在 Laplacian 描述的这张图上比较光滑的基底，这相当于用到了一个“图上光滑”的先验；当状态空间连续的时候，不仅用到了“图上光滑”的先验，其实还需要用到“原始度量空间中光滑”的先验（考虑上一个公式中的 $k(x,s_i)$ ）。

10. Representation policy iteration (RPI)

Model-based RPI

在 model-based 的设定中，假设当给定一个策略 π 之后， P^π, R^π 都已知。针对一个给定的 MDP ，可以通过如下循环来进行求解：

- Representation construction：对于当前策略，构造相应的低维表示，生成一个低维的 MRP；
- Policy evaluation：估计当前策略下的价值函数；
- Policy improvement：在当前价值函数的估计上做一步学习。

注意到，由于假设模型已知，因此这里只需要对状态价值函数 V 进行低维空间的表示即可；在策略改进的时候，可以利用已知的状态转移矩阵来对计算 after-state 的价值函数，从而实现策略的更新。在模型未知的时候，一般需要学习行动价值函数 Q 的低维表示，这种情况就要再更头疼一

些。

Model-Based RPI (M, π, k):

// $M = (S, A_s, P, R)$: Input (discrete) MDP
// π: Initial policy
// **Flag** : Convergence condition for policy iteration
// k: Number of basis functions to use

- **Repeat**

Representation Construction Phase

  - Define Laplacian operator $\mathbb{L}^\pi = I - P^\pi$.
  - Construct an $|S| \times k$ basis matrix Φ by orthogonalizing the vectors:
    - **Krylov basis**:

$$\mathcal{K}_k^\pi = \{R^\pi, \mathbb{L}^\pi R^\pi, (\mathbb{L}^\pi)^2 R^\pi, \ldots, (\mathbb{L}^\pi)^{k-1} R^\pi\}$$

    - **Drazin basis**:

$$\mathbb{D}_k^\pi = \{(P^\pi)^* R^\pi, (\mathbb{L}^\pi)^D R^\pi, ((\mathbb{L}^\pi)^D)^2 R^\pi, \ldots, ((\mathbb{L}^\pi)^D)^{k-1} R^\pi\}$$

  - Form the Markov reward process $M_\phi = (P_\phi^\pi, R_\Phi^\pi)$:

$$P_\Phi^\pi = \Phi^T P^\pi \Phi$$
$$R_\Phi^\pi = \Phi^T R^\pi$$

Policy Evaluation Phase

  - Find compressed solution: $(I - \gamma P_\Phi^\pi) w_\Phi = R_\Phi^\pi$.
  - Project solution back to original state space: $V_\Phi^\pi = \Phi w_\Phi$

Policy Improvement Phase

  - Find the "greedy" policy π' associated with V_Φ^π:

$$\pi'(s) \in \operatorname*{argmax}_a \left(\sum_{s'} P_{ss'}^a \left(R_{ss'}^a + \gamma V_\Phi^\pi(s') \right) \right)$$

  - If $\pi' \neq \pi$ set $\pi \leftarrow \pi'$, set **Flag** to false, return to Step 2
  - Else set **Flag** to true.

- Until **Flag**
- Return π.

Fig. 8.1 This figure shows a model-based algorithm for jointly learning representation and control in discrete MDPs.

DWT-Based RPI $(\mathrm{M}, k, \pi, \epsilon)$:

// $M = (S, A_s, P, R)$: Input (discrete) MDP
// π: Initial policy
// **Flag** : Convergence condition for policy iteration
// k: Number of levels for diffusion wavelet tree construction
// ϵ: Desired resolution for fast inversion
// Φ_0: Initial basis (e.g., unit vectors).
// **SpQR**: Sparse QR decomposition routine

- **Repeat**

Representation Construction Phase

- Build a diffusion wavelet tree as described in Section 7.6.

$$\mathcal{T}_\pi = DWT(P^\pi, \Phi_0, k, \mathbf{SpQR}, \epsilon)$$

Policy Evaluation Phase

- Compute a fast direct inverse using the Schultz expansion as described in Section 7.4

$$\hat{V}^\pi = \text{FastDirectInverse}(I - \gamma P^\pi, R^\pi)$$

Policy Improvement Phase

- Find the "greedy" policy π' associated with $\hat{V}^\pi$:

$$\pi'(s) \in \underset{a}{\text{argmax}} \left(\sum_{s'} P^a_{ss'} \left(R^a_{ss'} + \gamma \hat{V}^\pi \right) \right)$$

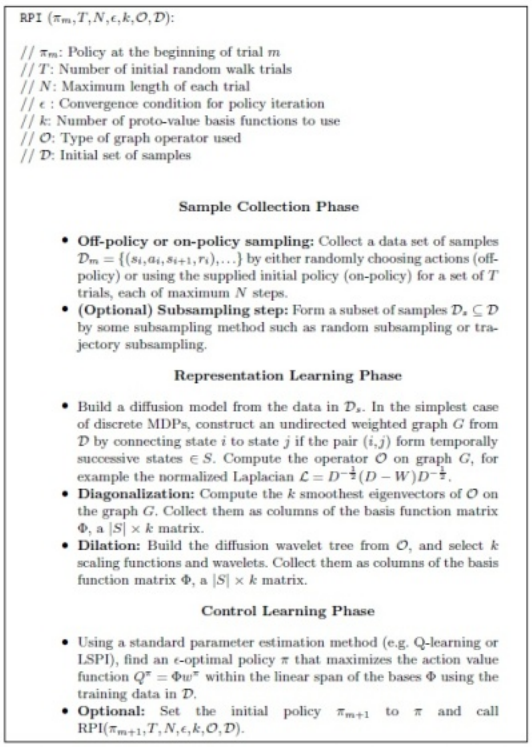
- If $\pi' \neq \pi$ set $\pi \leftarrow \pi'$, set **Flag** to false
- Else set **Flag** to true.
- **Until Flag**

Return π.

Fig. 8.2 This figure shows a variant of RPI using fast inversion with diffusion wavelets.

Model-free RPI

这个时候 transition function 不再已知，而是需要通过采集样本来得到。



RPI $(\pi_m, T, N, \epsilon, k, \mathcal{O}, \mathcal{D})$:

// π_m: Policy at the beginning of trial m
// T: Number of initial random walk trials
// N: Maximum length of each trial
// ϵ : Convergence condition for policy iteration
// k: Number of proto-value basis functions to use
// $\mathcal{O}$: Type of graph operator used
// $\mathcal{D}$: Initial set of samples

Sample Collection Phase

- **Off-policy or on-policy sampling:** Collect a data set of samples $\mathcal{D}_m = \{(s_i, a_i, s_{i+1}, r_i), \ldots\}$ by either randomly choosing actions (off-policy) or using the supplied initial policy (on-policy) for a set of T trials, each of maximum N steps.
- **(Optional) Subsampling step:** Form a subset of samples $\mathcal{D}_s \subseteq \mathcal{D}$ by some subsampling method such as random subsampling or trajectory subsampling.

Representation Learning Phase

- Build a diffusion model from the data in $\mathcal{D}_s$. In the simplest case of discrete MDPs, construct an undirected weighted graph G from $\mathcal{D}$ by connecting state i to state j if the pair (i, j) form temporally successive states $\in S$. Compute the operator $\mathcal{O}$ on graph G, for example the normalized Laplacian $\mathcal{L} = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$.
- **Diagonalization:** Compute the k smoothest eigenvectors of $\mathcal{O}$ on the graph G. Collect them as columns of the basis function matrix Φ, a $|S| \times k$ matrix.
- **Dilation:** Build the diffusion wavelet tree from $\mathcal{O}$, and select k scaling functions and wavelets. Collect them as columns of the basis function matrix Φ, a $|S| \times k$ matrix.

Control Learning Phase

- Using a standard parameter estimation method (e.g. Q-learning or LSPI), find an ϵ-optimal policy π that maximizes the action value function $Q^\pi = \Phi w^\pi$ within the linear span of the bases Φ using the training data in $\mathcal{D}$.
- **Optional:** Set the initial policy π_{m+1} to π and call RPI$(\pi_{m+1}, T, N, \epsilon, k, \mathcal{O}, \mathcal{D})$.

Fig. 10.1 This figure shows a generic algorithm for jointly learning representation and control, where representations are specifically constructed by diagonalizing (or dilating an initial basis with) a graph operator, such as the normalized graph Laplacian.

Remark：这里的 model-based 和 model-free 和我们强化学习里面的设定不一样，这里讲的 model-based 其实是我们现在经常讲的 known transition，它就不是 RL 问题；而这里讲的 model-free 其实是我们现在经常讲的 model-based RL。

参考文献

[1] Bertsekas, Dimitri P., and David A. Castanon. "Adaptive aggregation methods for infinite horizon dynamic programming." (1988).

编辑于 2020-03-02

英文论文 学术论文 强化学习 (Reinforcement Learning)

文章被以下专栏收录