Reinforcement Learning with Soft State Aggregation

Reinforcement Learning with Soft State Aggregation (http://blogs.cuit.columbia.edu/zp2130/files/2019/02/Reinforcement-Learning-with-Soft-State-Aggregation.pdf)

Math Analysis

– Present A New Approach Based On Bayes' Theorem: Apply Clustering π Rather than State Lookup Table for Computing Q Value

Problem Definition and Summary of Notation

We consider the problem of solving large Markovian decision processes (MDPs) (https://blogs.cuit.columbia.edu/zp2130/policy_gradient/) using RL algorithms and compact **function approximation**. The objective is to **maximize the expected,** infinite horizon, discounted sum of payoffs.

S: state space

A: action space

$P^a(s,\ s')$: transition probability

$R^a(s)$: payoff

γ: discount factor

χ: cluster space

$P(x \mid s)$: clustering probability, each state *s* belongs to cluster *x* with the probabilities.

$x,\ y$: represent individual **cluster**s

$s\ s'$: represent individual states

$< s_t,\ a_t,\ s_{t+1}, r_t >$: An **online RL algorithm** essentially sees a sequence of quadruples, representing a transition from current state s_t to next state s_{t+1} on current action a_t with an associated **payoff $\mathbf{r_t}$，所谓在线强化学习算法，就是指执行算法后得到数据集，而不是一开始获得所有数据集。**

$q_*(a)$: true value (expected reward) of action a

α: learning rate

In a Markov Decision Process:

s, s': states

a: an action

r: a reward

A_t: action at time t

S_t: state at time t, typically due, scholastically, to S_{t-1} and A_{t-1}

R_t: reward at time t, typically due, scholastically, to S_{t-1} and A_{t-1}

π: policy (decision-making rule)

$\pi(s)$: action taken in state s under *deterministic* policy π

$\pi(a \mid s)$: probability of taking action a **in state s** under *stochastic* policy π

r(s,a): expected immediate reward from state s after a

$v_\pi(s)$: value of state s under policy π (expected return)

$v_*(s)$: value of state s under the optimal policy

$q_\pi(s,a)$: value of **taking action a** in state s under policy π,

$q_*(s,a)$: value of taking action a in state s under the **optimal policy**

v(s) and q(s,a): MDP and Policy Gradient (https://blogs.cuit.columbia.edu/zp2130/policy_gradient/)

V, V_t: array estimates of state-value function v_π or v_*（表明当前状态的值，可能只研究状态对价值函数的影响，比如马尔可夫决策过程特殊的Q-learning问题TD(0)每个状态只有一个动作，不存在多个动作产生多个Q值而需要取某状态下的Q值最大值的问题）

Q, Q_t: array estimates of action-value function q_π or q_*（也表明下个状态的值，可能同时研究状态和动作对价值函数的影响）

G_t: return following time t

h: horizon, the time step one looks up to in a forward view

1. A general Convergence Theorem

Consider any stochastic process that generates a sequence of random quadruples,

任何随机过程，它产生随机四元组的一个序列：

$$\Psi = \{< x_i,\ a_i,\ y_i, r_i >\}_i$$

即该序列ψ为

$$\Psi = \{\Psi_1, \Psi_2, \Psi_3, ..., \Psi_i, ...\} = \{< x_1, a_1, y_1, r_1 >,\ < x_2, a_2, y_2, r_2 >,\ < x_3, a_3, y_3, r_3 >, ... < x_i, a_i, y_i, r_i >, ...\}$$

以<当前状态，动作，**某聚集内**的下一个状态，相关回报**>四元组为元素的序列，即所有随机状态。**

可以类比**一个在线强化学习**见证的**一个四元组序列**：

< s_t , a_t , s_{t+1} , r_t >

< current state, current action, next state, associated payoff >

这两个四元组的区别只在于，论文中讨论的下一个状态不一定在当前的聚集里，也有可能在**别的聚集**里。

where

$x_i, y_i \in Y$ represent individual clusters,

$$x_{i+1} \neq or = y_i \ (*)$$

* 关于文中强调的这点，我个人理解，从数学角度来看，可能是因为在这篇有关"软"状态聚合的论文中状态是以**聚集概率(Clustering Probability)**从属于某个"聚集"(Cluster)，也就是说，论文中当说到随机四元组(quadruple)中的 "下一个状态" 不一定是对应于当前状态所在的状态聚合中（也许可以用*x*表示，也许就不行）的必然发生的那个"下一个状态"(因为不一定，所以不一定等于x_{i+1})，也有可能下一个状态在另外一个**聚空间**，因为某个状态从属于某个"聚集"(cluster)是有某个概率的，并不是100%从属某cluster，为了避免混淆State Cluster和State Aggregation各自概念中的"下一个状态"，所以在描述状态聚集时用y_i表示下一个状态，而不是x_{i+1}。我这里想到一个方便理解的例子，有位学生背包里装了若干课本坐校车，课本A1, A2, A3 . . . 相当于某些状态，校车B1, B2, B3, . . . 相当于聚，当要求学生必须坐校车B1时，这相当于聚合，而如果允许学生自由选择，既可以坐B1也可以坐B2, B3, . . .，这就相当于某课本（某状态）以聚集概率从属于某校车聚集(有关聚集概率本文有解释说明，请搜索关键

字"聚集概率")。假设书包里可能有至少一本课本：A1, A2, A3, . . .，如果研究哪本课本被带到哪辆校车的概率，则可看做书包里装了某课本的条件下的坐上某校车**条件概率**，可以发现一个有趣的研究方向，既然可以研究某课本在书包里的条件下（书包里有A1的课本的概率不为0）被带上某校车的概率，反过来，也可以研究学生坐上某校车的条件下（坐上某校车的概率不为0）某课本被带上校车的概率，这也是**贝叶斯公式**的应用之一，而该论文也正是利用类似的逆向思维研究计算动作值函数的Q值。有关软状态聚合或状态聚集与状态聚合的概念和区别，我在本小结后面做了具体说明。

$$a_i \in A$$

$$r_i: \text{bounded real number}$$

Let $|Y|$ and $|A|$ be finite, and define indicator variables

$$\chi_i(x,a,y) = \begin{cases} 1 & when\ \Psi_i = <x,a,y,.> \ (for\ any\ r) \\ 0 & otherwise, \end{cases}$$

and

$$\chi_i(x,a) = \begin{cases} 1 & \Psi_i = <x,\ a,.\ ,.\ > (for\ any\ y,\ and\ any\ r)) \\ 0 & otherwise. \end{cases}$$

随机过程产生的以随机四元组为元素的序列中的某个元素即四元组ψ_i，如果属于聚集则χ_i 为1，不属于聚集则χ_i 为0。换句话说，就是**看在某随机过程产生的所有随机四元组里是否有某些四元组（状态）属于某状态聚集<x, a, y, . > 或 <x, a, . , . >。**

Define

$$P^a_{i,j}(x,y) = \frac{\sum_{k=i}^{j} \chi_k(x,a,y)}{\sum_{k=i}^{j} \chi_k(x,a)}$$

$$P^a_{i,j}(x,y) = \frac{\chi_i(x,a,y) + \chi_{i+1}(x,a,y) + \chi_{i+2}(x,a,y) + ... + \chi_j(x,a,y)}{\chi_i(x,a) + \chi_{i+1}(x,a) + \chi_{i+2}(x,a) + ... + \chi_j(x,a)}$$

这个公式是以概率收敛方式，从宏观角度考虑，求某随机过程中产生随机四元组为元素的一个序列中的所有四元组元素中属于某个状态聚集(x,y)的概率，也就是说这**随机的四元组**中有多大的比例是属于**某个状态聚集(x,y)**的。或者说，把**所有的随机状态（四元组）以概率形式划分出某些个状态聚集，这些状态聚集以某概率属于所有的随机状态。**

还以**课本上校车**为例，可以假设B1校车发车时间是对应数学课，B2校车发车时间对应物理课，比如学生总共有8本课本，需要带3本上车，就有C_8^3 种可能，我们可以考虑这些可能中有多大比例是属于3本课本其中有一本是数学的，也就是说多大概率是上B1校车。

而P(x | s)是指每个状态 s 属于某聚集 x 的概率。

这个结论对于看懂文章的公式很重要！！！：所以，综上，某个状态属于某聚集有个概率 P (x | s)，而某聚集属于所有随机状态有个概率

$$P^a_{0,\infty}(x,y) = \bar{P}^a(x,y)$$

and

$$R^a_{i,j}(x) = \frac{\sum_{k=i}^{j} r_k \chi_k(x,a)}{\sum_{k=i}^{j} \chi_k(x,a)} \quad (*)$$

*论文中可能存在编辑错误，论文中计算回报从 k = 1 开始求和，分子可能应该从 k = i 开始求和。

这里根据状态空间分别定义了：在状态聚集cluster角度看，状态空间第 i 个，. . . ， 第 j 个聚集，把状态空间里的这些聚集即元素看成一个状态聚集序列 x_i ，. . . ，x_j ，这个序列中的某个元素即状态聚集x_k 某状态x的下一状态y发生的概率，以及发生下一状态y所带来的回报。

Theorem 1:

If $\forall \epsilon > 0, \exists M_\epsilon < \infty$, such that for all $i \geqslant 0$ for all $x,\ y \in Y$ and for all $a \in A$ the following conditions characterize the infinite sequence Ψ with probability $1-\epsilon$,

$$\left|P^a_{i,i+M_\epsilon}(x,y) - \bar{P}^a(x,y)\right| < \epsilon$$

and

$$\left|R^a_{i,i+M_\epsilon}(x) - \bar{R}^a(x)\right| < \epsilon \quad (1)$$

where for all x, a, and y, with probability one 这里对于所有可能发生的状态空间，从概率论与数理统计的角度看依概率收敛于

$$P^a_{0,\infty}(x,y) = \bar{P}^a(x,y)$$

and

$$R^a_{0,\infty}(x) = \bar{R}^a(x)$$

也就是说，(1)公式表明：以数学的角度看状态聚集(state cluster)，**某状态聚集**属于**所有随机状态**的概率和回报分别**依概率收敛**于$\bar{P}^a(x,y)$ 和 $\bar{R}^a(x,y)$。本小结有从概率论与数理统计的角度**依概率收敛**的解释。

Then, online Q-learning applied to such a sequence will converge with probability one to the solution of the following system of equations:

$$\forall x \in Y, and\ \forall a \in A$$

$$Q(x,a) = \bar{R}^a(x) + \gamma \sum_{y \in Y} \bar{P}^a(x,y) max_{a' \in A} Q(y,a') \quad (2)$$

公式(2)计算状态聚集中在线Q-learning的Q值，它等于那个收敛的回报$\bar{R}^a(x,y)$ 加上衰减 乘以 以下两者乘积的求和： 1. 当前状态以某概率从属于的若干个状态聚集中某动作的对应的四元组中的下一个状态y对应于动作 a' （这里的 a' 可能与当前的状态的动作a相同，也可能不同）的动作值函数Q值的最大值，2.那个收敛概率$\bar{P}^a(x,y)$。

依概率收敛(Convergence in Probability)

依概率收敛比依分布收敛更强，记作$X_n \xrightarrow{p} X$意味着：$P(|X_n - X| < \epsilon) \to 1\ when\ n \to \infty, \forall \epsilon > 0$，也就是说，当n很大的时候，对于任意发生的事件，$X_n$的**值**和X的**值差不多。**依概率收敛在乎的是**随机变量的值。**

Convergence in probability is stronger than convergence in distribution. In particular, for sequence $X_1, X_2, X_3 \ldots$ to converge to a random variable X, we must have that $P(|X_n - X| < \epsilon)\ goes\ to\ 1\ as\ n \to \infty$, for any $\epsilon > 0$. To say that X_n converges in probability to X, we write

$$X_n \xrightarrow{p} X$$

A sequence of random variables $X_1, X_2, X_3 \ldots$ converges in probability to a random variable X. shown by

$X_n \xrightarrow{p} X$, if

$$\lim_{n \to \infty} P(|X_n - X| < 0) = 1,\ for\ all\ \epsilon > 0$$

Norms and Metrics

Download Norms and Metrics (http://blogs.cuit.columbia.edu/zp2130/files/2019/02/NormsMetrics.pdf)

Appendix A Convergence of semi-batch Q-learning (Theorem 1)

首先我们需要了解下semi-batch在这篇论文中是什么含义。

"**Batch mode** means that the **entire data set** for learning is **available from** the start, as opposed to the **online mode** of the algorithms we focus on in this book in which **data are acquired sequentially while the learning algorithm executes**. Batch-mode reinforcement learning algorithms are sometimes necessary when online learning is not practical, and they can use any batch-mode supervised learning regression algorithm, including algorithms known to scale well to high-dimensional spaces." – <*Reinforcement Learning: An introduction*>.

批量模式意味着学习的**整个数据集从一开始就是已知的**，与在（《*强化学习导论*》）这本书中我们主要关注的一些算法的**在线模式**相反，**在线模式即数据是在执行学习算法之后获得数据。**当在线学习不可实现的时候，批量模式的强化学习算法有时候是必要的，这些算法能用任何批量模式监督学习回归算法，包括对于已知适用于高维空间较好的一些算法。–《*强化学习导论*》

Consider a **semi-batch algorithm** that collects the changes to the Q-value function for **M steps** before making the change to the Q-value function.

所以，根据《强化学习导论》里有关批量模式的解释，和论文中以下有关回报和Q-值计算的描述，可以认为**semi-batch算法在本论文中是把算法中的每M个聚集"打包"成1个批量，在第k个批量中一次性执行算法中的M个聚集，（也就是说 聚集总数量 = k * M个），再获得相应数据集，从而改变Q-value function。**所以既不是整个数据集从一开始就已知的，也不是每个迭代步执行学习算法后才获得数据。

$$R^a_k(x) = \sum_{i=(k-1)M}^{kM} r_i \chi_i(x,a)$$

计算在**第k个批量**中，在所有随机状态中从**第(k-1)M个到第kM个属于某状态聚集 (x, a) 的状态的回报的和**

计算第1个批量，即当k = 1 时，i = 0 ~ M， 在所有随机状态中从**第0个到第M个属于某状态聚集 (x, a) 的状态的回报的和**r_0x_0 ~ r_Mx_M

计算第2个批量，即当k = 2 时，i = M ~ 2M，在所有随机状态中从**第M个到第2M个属于某状态聚集 (x, a) 的状态的回报的和**r_Mx_M ~ $r_{2M}x_{2M}$

计算第3个批量，即当k = 3 时，i = 2M ~ 3M，在所有随机状态中从**第2M个到第3M个属于某状态聚集 (x, a) 的状态的回报的和**$r_{2M}x_{2M}$ ~ $r_{3M}x_{3M}$

. . .

计算第k个批量，i = (k-1)M ~ kM，在所有随机状态中从**第(k – 1) M个到第 kM 个属于某状态聚集 (x, a) 的状态的回报的和**$r_{(k-1)M}x_{(k-1)M}$ ~ $r_{kM}x_{kM}$

$$M_k(x,a) = \sum_{i=(k-1)M}^{kM} \chi_i(x,a)$$

计算**第k个批量**中，在所有随机状态中从**第(k-1)M个到第kM个状态里属于某状态聚集 (x, a) 的个数**

计算第1个批量，即当k = 1 时，i = 0 ~ M， 在所有随机状态中从**第0个到第M个属于某状态聚集 (x, a) 的状态的个数**

计算第2个批量，即当k = 2 时，i = M ~ 2M， 在所有随机状态中从**第M个到第2M个属于某状态聚集 (x, a) 的状态的个数**

计算第3个批量，即当k = 3 时，i = 2M ~ 3M， 在所有随机状态中从**第2M个到第3M个属于某状态聚集 (x, a) 的状态的个数**

. . .

计算第k个批量，即当i = (k-1)M ~ kM，在所有随机状态中从**第(k – 1) M个到第 kM 个属于某状态聚集 (x, a) 的状态的个数**

$$M_k(x,a,y) = \sum_{i=(k-1)M}^{kM} \chi_i(x,a,y)$$

计算**第k个批量**中，在所有随机状态中从**第(k-1)M个到第kM个状态里属于某状态聚集 (x, a, y) 的个数**

计算第1个批量，即当k = 1 时，i = 0 ~ M， 在所有随机状态中从**第0个到第M个属于某状态聚集 (x, a, y) 的状态的个数**

计算第2个批量，即当k = 2 时，i = M ~ 2M， 在所有随机状态中从**第M个到第2M个属于某状态聚集 (x, a, y) 的状态的个数**

计算第3个批量，即当k = 3 时，i = 2M ~ 3M， 在所有随机状态中从**第2M个到第3M个属于某状态聚集 (x, a, y) 的状态的个数**

. . .

计算第k个批量，即当i = (k-1)M ~ kM，在所有随机状态中从**第(k – 1) M个到第 kM 个属于某状态聚集 (x, a, y) 的状态的个数**

Then the Q-value of (x, a) after the k^{th} batch is given by:

$$Q_{k+1}(x,a) = (1 - M_k(x,a)\alpha_k(x,a))Q_k(x,a) + M_k(x,a)\alpha_k(x,a)\left[\frac{R_k^a(x)}{M_k(x,a)} + \gamma\sum_{y\in Y}\frac{M_k(x,a,y)}{M_k(x,a)}max_{a'}Q_k(y,a')\right]$$

$$= Q_k(x,a) + M_k(x,a)\alpha_k(x,a)\left[\frac{R_k^a(x)}{M_k(x,a)} + \gamma\sum_{y\in Y}\frac{M_k(x,a,y)}{M_k(x,a)}max_{a'}Q_k(y,a') - Q_k(x,a)\right]$$

在第k个批量中，learning rate等于$M_k(x,a)\alpha_k(x,a)$，因为相当于一次同时**处理了所有随机状态里面从(k-1)M到kM属于某聚集状态的多个随机状态**，效率提高了$M_k(x,a)$倍，所以学习率乘以$M_k(x,a)$，而作为第k个批量"打包"来说，这个**"包"**里的**每个聚集**的**回报**需要除以$M_k(x,a)$，而a'动作对应的下个状态的聚集(y, a')有M_k(x, a, y)个，所以需要乘以$M_k(x,a)$，即乘以某个聚集中a'动作对应的下个状态聚集y发生的概率 。根据**下面经典的Q-learning update rule**和前面有关的说明，可以推导出上面**有关批量Q值计算**的结果。

Q-learning: Off-policy TD Control

One of the early breakthroughs in reinforcement learning was the development of an off-policy TD control algorithm known as Q-learning (Watkins, 1989), defined by

$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha\left[R_{t+1} + \gamma max_a Q(S_{t+1},a) - Q(S_t,A_t)\right]$$

关于Q-learning，核心是采取某种策略使得Q值越大越好。在当前状态下更新当前状态的Q值，预判接下来发生的某个动作导致的下一个状态（这个动作可能与当前动作相同，也可能与当前动作不同，所以下一个状态也有可能不同）所带来的回报，从估计角度看，Q(S_t , A_t) 是当前状态的Q值；从现实角度看，回报在当前状态并没有得到，但是如果达到下个状态了，回报就是实际存在的，既然是"预判"，所以下个状态的Q最大值需要乘以一个衰减系数再加上回报 $R_{t+1} + \gamma max_a Q(S_{t+1},a)$， 估计与现实的差距与学习效率有关，学习效率快，则更新当前状态的Q值速度越快，所以差距需要乘以学习效率。

In this case, the learned action-value function, Q, directly approximates q_*, the optimal action-value function, independent of the policy being followed. This dramatically simplifies the analysis of the algorithm and enabled early convergence proofs. The policy still has an effect in that it determines which state–action pairs are visited and updated. However, all that is required for correct convergence is that all pairs continue to be updated. This is a minimal requirement in the sense that any method guaranteed to find optimal behavior in the general case must require it. Under this assumption and a variant of the usual stochastic approximation conditions on the sequence of step-size parameters, Q has been shown to converge with probability 1 to q_*.

Q-learning (off-policy TD control) for estimating π≈π∗

Algorithm parameters: step size $\alpha \in (0,1]$, small ε > 0

Initialize Q(s,a), for all $s \in S^+, a \in A(s)$, arbitrarily except that Q(*terminal*,.) = 0

Loop for each episode:

  Initialize S

  Loop for each step of episode:

    Choose *A* form S using policy derived from Q (e.g., ε-greedy)

    Take action *A*, observe *R*, *S'*

    $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma max_a\, Q(S',a) - Q(S,A)\right]$

    $S \leftarrow S'$

    until *S* is terminal

<Reinforcement Learning, An Introduction> Richard S. Sutton and Andrew G. Barto

---

令$\bar{Q}$是公式（2）的解：

$$Q(x,a) = \bar{R}^a(x) + \gamma\sum_{y\in Y}\bar{P}^a(x,y)max_{a'\in A}Q(y,a') \quad (2)$$

也就是说

$$\bar{Q}(x,a) = \bar{R}^a(x) + \gamma\sum_{y\in Y}\bar{P}^a(x,y)max_{a'\in A}\bar{Q}(y,a')$$

定义

$$F_k(x,a) = \frac{R_k^a(x)}{M_K(x,a)} + \gamma\sum_{y\in Y}\frac{M_k(x,a,y)}{M_k(x,a)}max_{a'}Q_k(y,a') - \bar{Q}(x,a) \quad \textbf{(Define\_F}_k\textbf{)}$$

把上面推导**第k个批量**的**Q值**的公式复制到这里方便对比看定义的**F_k(x, a)**有什么特点：**和下面公式后面方括号里的内容几乎一样，当然，从强化学习概念的角度看，严格来说，因为是第K个批量，所以下面公式里Q值并不是所有随机状态序列里收敛的那个Q值，但**可以基于两者高度相似性作为推导公式的入手点。

事实上，**F_k 即是TD error.**

$$\delta_t \doteq R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

"Finally, note that the quantity in brackets in the **TD(0)** update is a sort of error, measuring the difference between the estimated value of S_t and the better estimate $R_{t+1} + \gamma V(S_{t+1})$. This quantity, called the ***TD error***, arises in various forms throughout reinforcement learning.

$$V(S_t) \leftarrow V(S_t) + \alpha\left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right]$$

TD(0), or one-step TD, because it is a special case of the TD(λ) and n-step TD methods developed in Chapter 12 and Chapter 7.

Notice that the TD error at each time is the error in the estimate *made at that time.* Because the TD error depends on the next state and next reward, it is not actually available until one time step later. That is, δ_t is the error in $V(S_t)$, available at time t+1. "

$$Q_{k+1}(x,a) = (1 - M_k(x,a)\alpha_k(x,a))Q_k(x,a) + M_k(x,a)\alpha_k(x,a)\left[\frac{R_k^a(x)}{M_k(x,a)} + \gamma\sum_{y\in Y}\frac{M_k(x,a,y)}{M_k(x,a)}max_{a'}Q_k(y,a')\right]$$

$$= Q_k(x,a) + M_k(x,a)\alpha_k(x,a)\left[\frac{R_k^a(x)}{M_k(x,a)} + \gamma\sum_{y\in Y}\frac{M_k(x,a,y)}{M_k(x,a)}max_{a'}Q_k(y,a') - Q_k(x,a)\right]$$

如果能够证明**方括号里面的值是极小的**，即

$$\forall\epsilon > 0, \exists M_\epsilon < \infty\ such\ that\ \epsilon_k^{M_\epsilon} < \epsilon\ with\ probability\ 1{-}\epsilon. Q_k(x,a) \to Q_\infty(x,a), where\ \left|Q_\infty(x,a) - \bar{Q}(x,a)\right| \le C\epsilon$$

则能**证明Q值收敛于某个值。**

为了说明**F_k(x, a)有个极小值，论文推导出了如下形式，我把这形式称为(Quantity_F_k)**

$$F_k(x,a) = \gamma\sum_y\frac{M_k(x,a,y)}{M_k(x,a)}\left[V_k(y) - \bar{V}(y)\right] + \left[\frac{R_k^a(x)}{M_k(x,a)} - R_{0,\infty}(x)\right] + \gamma\sum_y\left[\left(\frac{M_k(x,a,y)}{M_k(X,a)} - P_{0,\infty}^a(x,y)\right)\bar{V}(y)\right]$$

(Quantity_F_k)

上面公式中可能有排版错误，分母中X应该是x。

我在这里采用与论文中相反的办法，即从**(Quantity_F_k)推导到(Define_F_k)**

$$F_k(x,a) = \gamma\sum_y\frac{M_k(x,a,y)}{M_k(x,a)}\left[V_k(y) - \bar{V}(y)\right] + \left[\frac{R_k^a(x)}{M_k(x,a)} - R_{0,\infty}(x)\right] + \gamma\sum_y\left[\left(\frac{M_k(x,a,y)}{M_k(X,a)} - P_{0,\infty}^a(x,y)\right)\bar{V}(y)\right]$$

$$= \gamma\sum_y\frac{M_k(x,a,y)}{M_k(x,a)}\left[V_k(y)\right] + \left[\frac{R_k^a(x)}{M_k(x,a)} - R_{0,\infty}(x)\right] + \gamma\sum_y\left[\left(-P_{0,\infty}^a(x,y)\right)\bar{V}(y)\right]$$

$$= \frac{R_k^a(x)}{M_k(x,a)} + \gamma\sum_y\frac{M_k(x,a,y)}{M_k(x,a)}\left[V_k(y)\right] - R_{0,\infty}(x) + \gamma\sum_y\left[\left(-P_{0,\infty}^a(x,y)\right)\bar{V}(y)\right]$$

$$= A - B$$

$$A = \frac{R_k^a(x)}{M_k(x,a)} + \gamma\sum_y\frac{M_k(x,a,y)}{M_k(x,a)}\left[V_k(y)\right]$$

$$B = R_{0,\infty}(x) + \gamma\sum_y\left[\left(P_{0,\infty}^a(x,y)\right)\bar{V}(y)\right]$$

因为在随机过程产生的所有随机状态组成的序列中存在某聚集(x,y)的概率和回报都收敛：

$$P^a_{0,\infty}(x,y) = \bar{P}^a(x,y)$$

and

$$R^a_{0,\infty}(x) = \bar{R}^a(x)$$

根据前面$\bar{Q}$是公式（2）的解：

$$\bar{Q}(x,a) = \bar{R}^a(x) + \gamma \sum_{y \in Y} \bar{P}^a(x,y) max_{a' \in A} \bar{Q}(y,a')$$

如果

$V_k(x) = max_a Q_k(x,a)$：**所有随机状态中的某聚集中的第k批量的Q值最大值**

则可知：$V(y) = max_{a'} Q(y,a')$

$\bar{V}(x) = max_a \bar{Q}(x,a)$：**所有随机状态中的某聚集的Q值的最大值**

则可知：$\bar{V}(y) = max_{a'} \bar{Q}(y,a')$

推导得：

$$F_k(x,a) = \frac{R^a_k(x)}{M_k(x,a)} + \gamma \sum_y \frac{M_k(x,a,y)}{M_k(x,a)} max_a Q(y,a') - \bar{Q}(x,a) \text{ (Define\_F}_k\text{)}$$

当然，在论文中从公式(Define_F_k)到(Quantity_F_k)的顺序正好与这里的推导相反。这里是从(Quantity_F_k)推导到(Define_F_k)。

所以我们可以接着分析F_k的大小

$$F_k(x,a) = \gamma \sum_y \frac{M_k(x,a,y)}{M_k(x,a)} \left[V_k(y) - \bar{V}(y)\right] + \left[\frac{R^a_k(x)}{M_k(x,a)} - R_{0,\infty}(x)\right] + \gamma \sum_y \left[\left(\frac{M_k(x,a,y)}{M_k(X,a)} - P^a_{0,\infty}(x,y)\right)\bar{V}(y)\right]$$

(Quantity_F_k)

Norms and Metrics

http://www.u.arizona.edu/~mwalker/econ519/Econ519LectureNotes/Norms&Metrics.pdf
(http://www.u.arizona.edu/~mwalker/econ519/Econ519LectureNotes/Norms&Metrics.pdf)

Euclidean Norm

欧几里得范数

$$\|x\| = \sqrt{x_1^2 + x_2^2 + ... + x_n^2}$$

Euclidean metric (l^2 metric)

$$d_{l^2}((x_1,...,x_n),(y_1,...,y_n)) := \sqrt{(x_1-y_1)^2 + ... + (x_n-y_n)^2}$$
$$= \left(\sum_{i=1}^n (x_i - y_i)^2\right)^{1/2}$$

if n=2, then $d_{l^2}((1,6),(4,2)) = \sqrt{(3^2 + 4^2)} = 5$. This metric corresponds to the geometric distance between the two points (x_1, x_2, . . . , x_n), (y_1, y_2 , . . . , y_n) as given by **Pythagoras' theorem**.毕达哥拉斯定理

Euclidean distance between two points x, x' ∈ R^n

$$d(x,x') := \|x,x'\|$$

Taxicab metric (l^1 metric)

$$d_{l^1}((x_1,x_2,...,x_n),(y_1,y_2,...,y_n)) := |x_1 - y_1| + ... + |x_n - y_n|$$
$$= \sum_{i=1}^n |x_i - y_i|$$

Because it models the distance a taxi-cab would have to traverse to get from one point to another if the cab was only allowed to move in cardinal directions (north, south, east, west) and not diagonally. **出租车只能沿着东西走向或者南北走向的街道开，不能从房子上或者从房子里开过去（虽然两点间直线最短）。**

$$d_{l^2}(x,y) \leq d_{l^1}(x,y) \leq \sqrt{n} d_{l^2}(x,y)$$

最简单形式的理解可以是**直角三角形两直角边之和**大于**斜边。**

$$\|F_k(x,a)\| \leq \gamma \|V_k - \bar{V}\| + \left\|\frac{R^a_k}{M_k(x,a)} - R^a_{0,\infty}(x)\right\| + \gamma \left\|\sum_y \left[\frac{M_k(x,a,y)}{M_k(x,a)} - P^a_{0,\infty}\right]\bar{V}(y)\right\|$$
$$\leq \gamma \|V_k - \bar{V}\| + C\epsilon^M_k$$

where

$$\epsilon^M_k = max\left\{\left|\frac{R^a_k(x)}{M_k(x,a)} - R^a_{0,\infty}(x)\right|, \gamma \left|\sum_y (\frac{M_k(x,a,y)}{M_k(x,a)} - P^a_{0,\infty}(x,y))\right|\right\}$$

ϵ^M_k 是 l^1 metric大于等于 norm (l^1 metric)

也就是说

$\|F_k(x,a)\|$**小于一个极小值**

根据前面有关定义F_k 的原因，为什么要证明F_k小于一个极小值的原因的说明，**也就证明了Q值收敛于某个值。**

简单介绍有关基于贝叶斯理论推导的依靠采样策略π的聚集化的Q值计算方法及

软状态聚合（或状态聚集）与状态聚合分别在概率论与数理统计和强化学习的概念和区别

现实世界里，以机械手臂（甚至具有传感功能的"八爪鱼"）为例，可能有无数状态(State)和动作(Action)（我个人理解"八爪鱼"触手可能具有的传感功能包括：温度感应，压力感应，等等），实际工作生活中要求机械手臂的灵活度越高越好，例如1000+自由度，从强化学习(Reinforcement Learning, RL)中数学角度看，状态空间(State Space)非常大，所以普遍认为，用更简洁的表达方式而不是查找表(lookup table)来标定(Scaling)强化学习到现实问题是非常关键的。在这片论文发表之前，几乎所有的强化学习理论用查找表研究该领域的问题，而这篇论文针对结合**近似函数(function approximation)**[1]和强化学习时所产生的问题，提出解决方案，比如从依靠采样策略**(sampling policy π)**产生**聚集(clustering)端**来计算Q值，而不是从**状态端出发**的查找表方法，因为作者提出有别于聚合的聚集理论（软聚合，即某状态通过聚集概率而非完全从属于某聚），用**贝叶斯理论**将有关**"状态""条件下的"聚"概率**计算Q值转换为有关**"聚"条件下的"状态"概率**计算Q值，也许可以这样理解：这样就解决了从**状态端出发**的查找表来计算Q值的计算量过大的问题，也就是说可以从**聚集端出发**的策略计算Q值，降低了计算量，从而提高了计算效率，节省计算时间和成本，为此作者通过缜密的逻辑思路和精准的用词，层层推进，教科书式地提出了以下几个研究成果：

1. 一个近似者函数(function approximator, FA)（在这个post中权且翻译成"近似者函数"，与通常的近似函数区分），这个函数是基于状态聚合(state aggregation)[2]的一个简单的扩展，也就是说状态以**聚集概率(clustering probability)**[3]属于某个聚合，故论文名：**"软"状态聚合(*soft* state aggregation)**[4]；
2. 一个对于任意的但是固定的软状态聚合的强化学习的收敛理论。既然提出一个函数，而且在"聚集"的层面上强化学习的问题可能属于非马尔可夫过程，所以证明这个近似者函数的收敛性是必要的，本文**依概率收敛**说明提出的软状态聚合的强化学习从数学概率论与数理统计的角度看也是正确的；
3. 关于这种软状态聚合对在线强化学习的影响的一种新颖的直观理解，这里利用了**贝叶斯公式的扩展形式**，即**贝叶斯公式简单形式**和**全概率公式**的结合形式，**将有关聚集概率（状态发生的条件下聚集的概率）的计算动作值函数Q值转换为有关在聚集的条件下状态发生的概率的计算动作值函数Q值，也可以将这种思路理解为是某种逆向概念；正是因为通过采样策略(sampling policy π)产生聚集(clustering)来计算Q值，而不是查找表(lookup table case)，（在本文搜关键字"贝叶斯公式"，有多个地方解释了该术语及其在论文中的应用）**
4. 一种新的探索式的适应性(adaptive)软状态聚合算法，这个算法通过探索软状态聚合的非离散本质，找到改良的简洁的软状态聚合的表达方式。

总之，这4步内容互相关联，教科书式地展开，层层递进，用词精准，逻辑缜密。首先提出一个函数，接着证明其收敛性，最后说明此函数的应用。

前面提到海量状态空间，为了研究方便且有效率，有必要简化表达它，**状态聚合**[2](State Aggregation)的概念就被提出来了。这篇论文提出了状态以聚集概率从属于多个聚集的形式的"软"状态聚合，可以理解为是状态聚合的一种升级版本。为了区分这种 – 某状态以某概率从属某聚的 状态"聚" 与 通常常用的 状态聚合，论文中用**cluster**，而非**aggregation**来表示'聚'。而我在这篇post中将state aggregation翻译成状态聚合，state cluster翻译成状态聚集，也是为了避免在数学上混淆两者。

1. **近似函数(function approximation)**：将状态空间（和所有的模型参数）以一种更简洁的方式近似化，比如线性（向量化），神经网络，决定树，等等。
2. **状态聚合(state aggregation)**：一个普遍使用的简洁表达状态的方式，它是一种简单的规范化近似函数的形式，也就是把状态分组打包，每个组有一个估计值（权重向量里的一个分量），一个状态的值也就是组里的分量，此状态更新，只要单独更新那个分量。
3. **聚集概率(clustering probability)**：每个状态以某一个概率从属于某一个"聚"，而这个概率叫聚集概率(clustering probability)。我觉得可能因为作者不想在数学层面混淆两类概念，为了区分aggregation，而另外取了名字cluster，**其二者区别在于cluster更强调状态以某个聚集概率(clustering probability)而非100%完全从属于某一个"聚"(cluster)，换句话说aggregation没有这个状态从属"聚"的概率，而cluster有。**我把aggregation翻译成聚合，cluster翻译成聚集，也是为了在数学的"概率"层面区别这两者，虽然本质上含义可能差不多。
4. **"软"状态聚合(*soft* state aggregation)**：状态以聚集概率而非完全从属于某一个聚集cluster，也就是说允许每个状态s从属于多个"聚"。一个有趣的特殊的例子是通常的状态聚合也就是每个状态只从属于一个聚集cluster。一个聚集的值可以通过与聚集概率的正比关系推广或规范generalize到所有的状态。也被称作"状态聚集" (State Clustering).

State aggregation is a simple form of generalizing function approximation in which **states are grouped together**, with one estimated value (one component of the weight vector w) for each group. The value of a state is estimated as its group's component, and when the state is updated, that component alone is updated.

Function Approximation: approximate the state space (and all model parameters) with a more compact one!
• Reduction in # of states (computation and space)
• More importantly: generalization to unseen states!

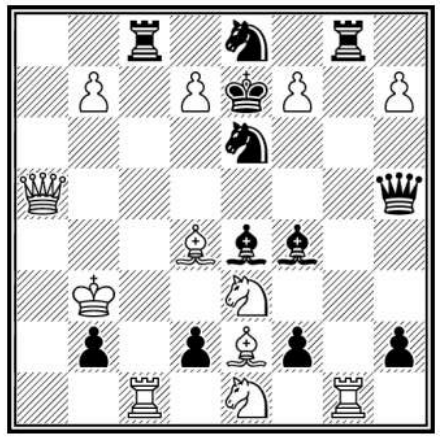
Types of (value / action-value) function approximation:
• Linear
• Neural network
• Decision tree

…

Function approximation
Finding optimal $\theta \longrightarrow$ knowledge of value for ALL states!

$$v_\theta(s) = \theta_1 x_1(s) + \theta_2 x(s) + \ \ldots \ + \theta_n x_n(s) = \theta^T x(s)$$

10^{40} states are mapped to linear function over n "important" features, i.e.
1. Number of white pieces – black pieces
2. Distance between kings
3. Etc.



Function approximation idea – generalization and efficiency
• Gradient descent approximation to estimate value/Q functions
• gradient descent to optimize the optimal Q-function directly

402-lec20 (http://blogs.cuit.columbia.edu/zp2130/files/2019/02/402-lec20.pdf)

Markov Decision Process (MDP)

Markov Reward Process, definition:

- Tuple (S, P, R, A, γ) where
  - S = states, including start state
  - A = set of possible actions
  - P = transition matrix $P_{SS'}^{a} = Pr[S_{t+1} = s' \mid S_t = s, A_t = a]$
  - R = reward function, $R_S^{\ a} = E[R_{t+1} \mid S_t = s, A_t = a]$
  - $\gamma \in [0, 1]$ = discount factor
- Return
  - $G_t = \sum_{i=1 \ to \ \infty} R_{t+i} \gamma^{i-1}$
- Goal: take actions to maximize expected return

Policies

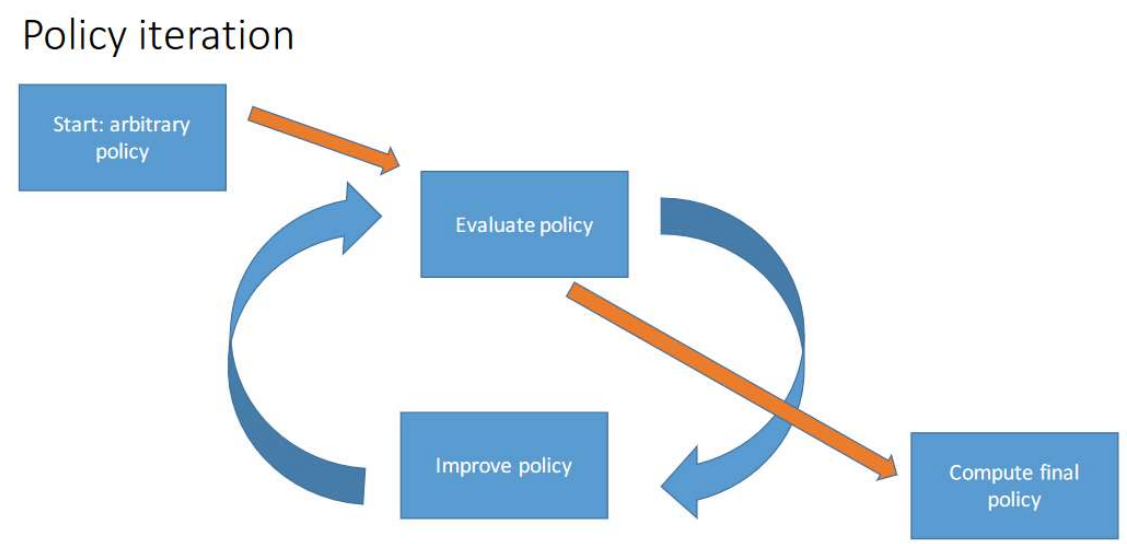
The Markovian structure —> best action depends **only** on current state.

- Policy = mapping from state to distribution over actions.
  - $\pi : Sx \mapsto \Delta(A), \pi(a \mid s) = Pr[A_t = a \mid S_t = s]$
- Given a policy, the MDP reduces to a Markov Reward Process

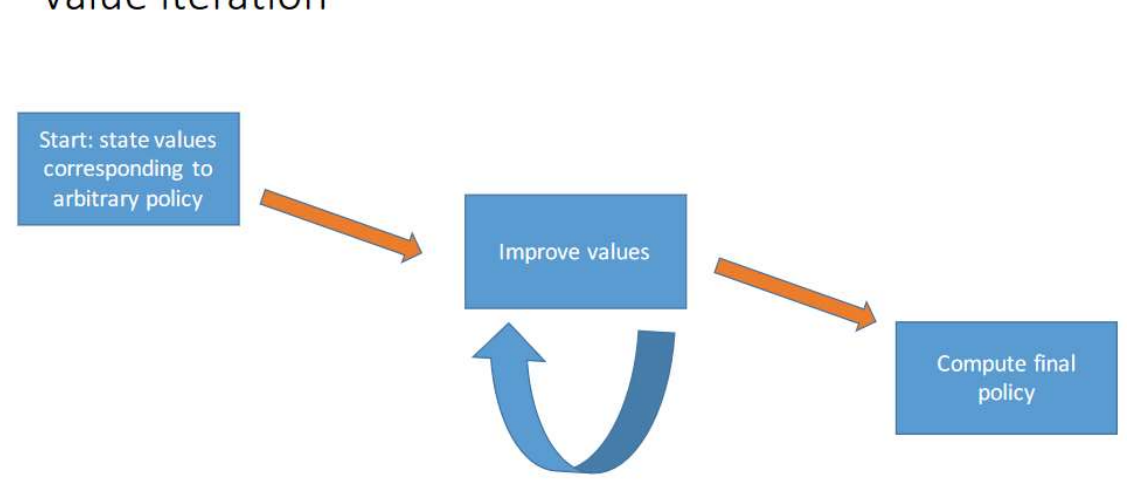
Bellman optimality equations

- Bellman equation:
  - $v_*(s) = max_a q_*(s, a)$
  - implies Bellman optimality equations:

  - $q_*(s, a) = R_S^{\ a} + \gamma \sum_{S'} P_{SS'}^{a} \ max_{a'} \{q_*(s', a')\}$
  - $v_*(s) = max_a \left\{ R_S^{\ a} + \gamma \sum_{S'} P_{SS'}^{a} \ v_x(S') \right\}$
- Iterative methods based on the Bellman equations: dynamic programming
  - Policy iteration



  - Value iteration



Temporal-Difference Learning

A simple every-visit Monte Carl method suitable for nonstationary environments is

$$V(S_t) \leftarrow V(S_t) + \alpha \left[G_t - V(S_t) \right]$$

$$NewEstimate \leftarrow OldEstimate + StepSize \left[Target - OldEstimate \right]$$

The expression [Target – OldEstimate] is an *error* in the estimate.

where G_t is the actual return following time t, and α is a constant step-size parameter

The target for the Monte Carlo update is G_t

TD(0), or *one-step* TD, because it is a special case of the TD(λ) and *n-step* TD methods

The simplest TD method makes the update

$$V(S_t) \leftarrow V(S_t) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right]$$

immediately on transition to S_{t+1} and receiving R_{t+1}.

The target for the TD update is $R_{t+1} + \gamma V(S_{t+1})$

Tabular TD(0) for estimating v_π
Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize V(s), for all $s \in S^+$, arbitrarily except that V(*terminal*) = 0
Loop for each episode:
  Initialize S
  Loop for each step of episode:
    $A \leftarrow$ action given by π for *S*
    Take action *A*, observe *R*, *S'*
    Invalid Equation
    $S \leftarrow S'$
  until *S* is terminal

<Reinforcement Learning, An Introduction> Richard S. Sutton and Andrew G. Barto

2. RL and Soft State Aggregation

这部分应用了定理1 的结论来提供两个例子的收敛性。

1. 用Q-learning和近似者函数(FA)解马尔可夫决策过程。
2. 用TD(0)和近似者函数(FA)决定一个固定决策的价值函数。

在线强化学习，对于learning agent而言，transition probability和payoff function都是未知的。

2.1 Case

Q-learning and Fix Soft State Aggregation

这部分作者开发出了一个在聚集空间级别针对Q-learning的"Bellman equation"。作者假设agent按照稳定的随机策略，也就是给每一个状态指定一个非零概率，即在每一个状态中执行每个动作都是非零概率。

$$P^\pi(s \mid x) = \frac{P(x \mid s)P^\pi(s)}{\sum_{s'} P(x \mid s')P^\pi(s')}$$

where for all s, $P^\pi(s)$: steady-state probability of being in state s.

这是一个**贝叶斯理论Bayes' theorem，全概率公式law of total probability**的结合形式，或者理解成**Bayes' theorem Extended form。**

分子分母都可以理解为有关条件概率的计算，首先，执行某个动作或者说成为某个状态s存在一个概率，然后，某状态s属于某聚集x存在一个聚集概率，所以总体看是先满足状态稳定的条件下再考虑此状态属于某个聚集，也就是说这是有关条件概率的计算。分母是在所有可能的下一步的状态条件下的聚集的条件概率分别与下一步状态概率乘积的总和，也就是聚集的概率，而分子当前状态的概率乘以在当前状态的条件下聚集的概率（条件概率），它们的结果是当前状态与某聚集的联合概率，分子分母的比值就是在聚集的条件下状态的概率，也就是说**某聚集内，存在某状态的概率**，这与**某状态从属于某聚集的概率**（聚集概率，聚集概率也算条件概率，它是在某状态的条件下，属于聚集的概率）的概念互为逆向。接下来简单介绍**贝叶斯公式及其扩展形式方便理解作者提出的"Bellman equation"。**

条件概率(Conditional probability)

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}, \ if \ P(B) \neq 0$$

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)}, \ if \ P(A) \neq 0$$

where $P(A \cap B)$ is 联合概率 joint probability of both A and B being true.

全概率公式(law of total probability):

for some partition {B_i } of the sample space, the event space is given or conceptualized in terms of P(B_i) 若事件B_1, B_2, B_3, . . . , B_n 是样本空间Ω的一个划分，则:

$$P(A) = \sum_{i=1}^{n} P(A \cap B_i)$$

and P(A | B_i). It is then useful to compute P(A) using the law of total probability:又因为条件概率公式，可进一步得:

$$P(A) = \sum_{i=1}^{n} P(A \mid B_i)P(B_i)$$

贝叶斯理论（Bayes' theorem）

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)} = \frac{P(A \mid B)P(B)}{P(A)}$$

贝叶斯理论扩展形式(Bayes' theorem Extended form)

(贝叶斯理论的常规形式与全概率公式的结合形式)

Bayes' Theorem

$$\begin{aligned} P(B \mid A) &= \frac{P(A \cap B)}{P(A)} \\ &= \frac{P(A \mid B)P(B)}{P(A)} \\ &= \frac{P(A \mid B)P(B)}{P(A \cap B_1) + P(A \cap B_2) + P(A \cap B_3) + ... + P(A \cap B_\infty)} \\ &= \frac{P(A \mid B)P(B)}{P(A \mid B_1)P(B_1) + P(A \mid B_2)P(B_2) + P(A \mid B_3)P(B_3) + ... + P(A \mid B_\infty)P(B_\infty)} \\ &= \frac{P(A \mid B)P(B)}{\sum_{B'} P(A \mid B')P(B')} \end{aligned}$$

Corollary 1

利用了**贝叶斯公式的扩展形式**，即**贝叶斯公式简单形式**和**全概率公式**的结合形式，**将有关聚集概率（状态发生的条件下聚集的概率）的计算动作值函数Q值转换为有关在聚集的条件下状态发生的概率的计算动作值函数Q值，也可以将这种思路理解为是某种逆向概念。**

为了对比，将前文提到的计算动作值函数的Q值的公式和由贝叶斯公式转换而来的新的计算动作值函数的Q值的公式放在一起。

$$Q(x, a) = \bar{R}^a(x) + \gamma \sum_{y \in Y} \bar{P}^a(x, y) max_{a' \in A} Q(y, a') \quad (2)$$

where

$$\bar{P}^a(x, y) = P^a_{0,\infty}(x, y)$$

$$\bar{R}^a(x) = R^a_{0,\infty}(x)$$

$$Q(x, a) = \sum_s P^\pi(s \mid x) \left[R^a(x) + \gamma \sum_y P^a(s, y) max_{a'} Q(y, a') \right] \quad (3)$$

where

$$P^a(s, y) = \sum_{s'} P^a(s, s')P(y \mid s')$$

Proof:

将以下两个**全概率公式**带入公式(2)，便得到公式(3)。

$$\bar{P}^{\,a}(x, y) = \sum_s P^\pi(s \mid x)P^a(s, y)$$

$$\bar{R}^a(x) = \sum_s P^\pi(s \mid x)R^a(s)$$

Case 2: TD(0) and Fixed Soft State Aggregation

$$V(x) = \sum P^\pi(s \mid x) \left[R^\pi(s) + \gamma \sum_y P^\pi(s, y)V(y) \right]$$

(4)

Proof:

TD(0)是马尔可夫决策过程Q-learning的特殊情况，也就是说**每个状态只有一个简单（也可能是随机的）动作**，也就是说**不用考虑a的多种可能性**，即不存在比较各种动作下的Q值的最大值的情况，**用V(y)代替公式(3)Max取值部分即可。**

3. Adaptive State Aggregation

在聚集个数固定的前提下，找到好的聚集概率(clustering probabilities)来提高简洁代表(compact representation)，从而获取更好的价值函数的近似值。在推理1和2里表明，RL在聚集空间(cluster space)可以找到零Bellman误差(zero Bellman error)的解。相应地，在状态空间，Bellman误差一般来说不为0。

总之，好的聚集化是以降低在马尔可夫决策过程状态中的Bellman误差为表现形式的。

聚集概率

$$P(x|s;\theta) = \frac{e^{\theta(x,s)}}{\sum_{x'} e^{\theta(x',s)}}$$

where

$$\theta = weight\ between\ state\ s\ and\ cluster\ x.$$

Bellman error at state s given parameter θ (a matrix) is

$$\begin{aligned} J(s|\theta) &= V(s|\theta) - \left[R^\pi(s) + \gamma \sum_{s'} P^\pi(s,s')V(s'|\theta)\right] \\ &= \left[\sum_x P(x|s;\theta)V(x|\theta)\right] - \left[R^\pi + \gamma \sum_{s'} P^\pi(s,s') \sum_x P(x|s';\theta)V(x|\theta)\right] \end{aligned}$$

通过**聚集概率**将**状态端s**转换为**聚集端x**

由原先的从某状态计算Q值，到现在从某聚集计算Q值。

$$\frac{\partial J^2(s|\theta)}{\partial\theta(y,s)} = 2J(s|\theta)[P(y|s;\theta)(1-\gamma P^\pi(s,s'))(V(y|\theta) - V(s|\theta))]$$

$$\frac{\partial J(s|\theta)}{\partial\theta(y,s)} = P(y|s;\theta)(1-\gamma P^\pi(s,s'))(V(y|\theta) - V(s|\theta)) \quad \text{(How do I get this?)}$$

Refer to Policy Gradient Methods for Reinforcement Learning with Function Approximation (https://blogs.cuit.columbia.edu/zp2130/policy_gradient_methods_for_reinforcement_learning_with_function_approximation/)

$$\frac{\partial P(x|s;\theta)}{\partial\theta(y,s)} = \frac{e^{\theta(x,s)}\sum_{x'} e^{(x',s)} - e^{\theta(x,s)}\sum_{x'} e^{(x',s)}}{(\sum_{x'} e^{(x',s)})^2} = 0 \quad \text{(Is it correct?)}$$

useful information online:

https://morvanzhou.github.io/tutorials/machine-learning/ML-intro/4-03-q-learning/ (https://morvanzhou.github.io/tutorials/machine-learning/ML-intro/4-03-q-learning/)

在看论文的过程中，看懂一点论文的状态动作值函数Q值越来越大。:)

edit (https://blogs.cuit.columbia.edu/zp2130/wp-admin/post.php?post=4166&action=edit)

Author: Z Pei (https://blogs.cuit.columbia.edu/zp2130/author/zp2130/) on February 6, 2019

Categories: Bayes' Theorem (https://blogs.cuit.columbia.edu/zp2130/category/bayes-theorem/), Q-learning (https://blogs.cuit.columbia.edu/zp2130/category/q-learning/), Reinforcement Learning (https://blogs.cuit.columbia.edu/zp2130/category/reinforcement-learning/), RL (https://blogs.cuit.columbia.edu/zp2130/category/rl/)

Tags: Bayes' Theorem (https://blogs.cuit.columbia.edu/zp2130/tag/bayes-theorem/), Q-learning (https://blogs.cuit.columbia.edu/zp2130/tag/q-learning/), Reinforcement Learning (https://blogs.cuit.columbia.edu/zp2130/tag/reinforcement-learning/), RL (https://blogs.cuit.columbia.edu/zp2130/tag/rl/)

Other posts

Fourier Transform and Matrix-vector Multiplication (https://blogs.cuit.columbia.edu/zp2130/fourier_transform_and_matrix-vector_multiplication/)
«» IntSim: A CAD tool for Optimization of Multilevel Interconnect Networks (https://blogs.cuit.columbia.edu/zp2130/intsim_a_cad_tool_for_optimization_of_multilevel_interconnect_networks/)

Last posts

- Symbolic Netlist to Innovus-friendly Netlist (https://blogs.cuit.columbia.edu/zp2130/symbolic_netlist_to_innovus-friendly_netlist/)
- Finite-Sample Convergence Rates for Q-Learning and Indirect Algorithms (https://blogs.cuit.columbia.edu/zp2130/finite-sample_convergence_rates_for_q-learning_and_indirect_algorithms/)
- Solving H-horizon, Stationary Markov Decision Problems In Time Proportional To Log(H) (https://blogs.cuit.columbia.edu/zp2130/paul_tseng_1990/)
- Randomized Linear Programming Solves the Discounted Markov Decision Problem In Nearly-Linear (Sometimes Sublinear) Run Time (https://blogs.cuit.columbia.edu/zp2130/randomized_linear_programming_solves_the_discounted_markov_decision_problem_in_nearly-linear_sometimes_sublinear_run_time/)
- KL Divergence (https://blogs.cuit.columbia.edu/zp2130/kl_divergence/)
- The Asymptotic Convergence-Rate of Q-learning (https://blogs.cuit.columbia.edu/zp2130/the_asymptotic_convergence-rate_of_q-learning/)
- Hierarchical Apprenticeship Learning, with Application to Quadruped Locomotion (https://blogs.cuit.columbia.edu/zp2130/hierarchical_apprenticeship_learning_with_application_to_quadruped_locomotion/)
- Policy Gradient Methods (https://blogs.cuit.columbia.edu/zp2130/policy_gradient_methods/)
- Actor-Critic Algorithms for Hierarchical Markov Decision Processes (https://blogs.cuit.columbia.edu/zp2130/actor-critic_algorithms_for_hierarchical_markov_decision_processes/)
- Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation (https://blogs.cuit.columbia.edu/zp2130/hierarchical_deep_reinforcement_learning_integrating_temporal_abstraction_and_intrinsic_motivation/)