对恋爱建模：马尔可夫决策过程、Bellman方程、价值迭代和策略迭代

Modeling love: Markov decision process, Bellman equation, value iteration and strategy iteration


璟明 Jingming

2025-03-23 08:03:27 已编辑 美国
2025-03-23 08:03:27 Edited United States

前言 Preface

马尔可夫决策过程（Markov Decision Process, MDP）是一个帮助我们在不确定环境中决策的工具，也是强化学习（Reinforcement Learning, RL）的基石。通常教学时以游戏为例，如吃豆人（CS188 Intro to AI）。但这容易让学生困惑：既然游戏规则可以随意设定，我们为什么需要一个工具来对一种特定的游戏建模？如果游戏规则变了，这个工具还有什么用?

Markov Decision Process (MDP) is a tool that helps us make decisions in uncertain environments and is also the cornerstone of Reinforcement Learning (RL). Usually, games are used as examples in teaching, such as Pac-Man (CS188 Intro to AI). But this can easily confuse students: since the rules of the game can be set arbitrarily, why do we need a tool to model a specific game? If the rules of the game change, what is the use of this tool?

补充：实际上MDP所做的类似于用y=kx+b对一元一次方程建模，无论具体方程如何改变，都可以用y=kx+b描述（尽管有时b=0，aka尽管有时游戏规则改变了）。

Addendum: In fact, what MDP does is similar to modeling a linear equation with y=kx+b. No matter how the specific equation changes, it can be described by y=kx+b (even though sometimes b=0, aka even though sometimes the rules of the game change).

但为了方便理解它的现实意义，这篇文章以恋爱为例，从零开始一步步讲解MDP是如何建模的，并分别用价值迭代（Value Iteration）和策略迭代 （Policy Iteration）求解恋爱的最优策略 。

But in order to make it easier to understand its practical significance, this article takes love as an example, explains step by step from scratch how MDP is modeled, and uses value iteration and policy iteration to solve the optimal strategy for love.

一，对恋爱建模：逐步构建MDP

1. Modeling Love: Building the MDP Step by Step

如果整篇文章只能记住一句话，那就是这句话。 一个问题可用MDP建模，要求符合 **马尔可夫性（Markov property）**，即，未来状态只依赖于当前状态，而与过去的历史状态无关。这里我们假设恋爱符合马尔可夫性，然后我们开始建模。

If there is only one sentence to remember from the entire article, this is it. A problem can be modeled using MDP, requiring **Markov property** , that is, the future state depends only on the current state and has nothing to do with the past historical state. Here we assume that love conforms to the Markov property, and then we start modeling.

1. **状态集合（State, S）**。首先，恋爱中存在很多种状态：陌生、暧昧、恋爱、结婚.....我们可以在这些状态之间互相跳转。为了简化，我们只考虑三种状态： 陌生、暧昧、恋爱。**其中，起始状态是陌生**。

1. **State set (State, S)** . First of all, there are many states in love: stranger, ambiguous, in love, married... We can jump between these states. For simplicity, we only consider three states: stranger, ambiguous, in love. **Among them, the starting state is stranger** .

2. **行动集合（Action, S)**。其次，在每种状态下，我们都可以做很多种行动：吃饭、看电影、牵手、拥抱、接吻.....为了简化，我们只考虑两种行动：看电影、接吻。

2. **Action set (Action, S)** . Secondly, in each state, we can do many kinds of actions: eating, watching movies, holding hands, hugging, kissing... For simplicity, we only consider two actions: watching movies and kissing.

3. **奖励函数or反馈函数（Reward, R)**。然后，每当我们在某种状态下，做出某种行动，我们都会得到对方的一些反馈，暂且认为这个反馈意味着亲密度吧。例如：陌生状态下看电影，可能会得到一些正反馈；恋爱状态下接吻，也可能得到一些正反馈；但陌生状态下接吻，大概率会得到一些负反馈.....于是我们知道，奖励函数至少和当前状态s，以及行动a有关，即**R(s,a)**，且假设奖励函数已知。

3. **Reward function or feedback function (Reward, R)** . Then, every time we take an action in a certain state, we will get some feedback from the other party. Let's assume that this feedback means intimacy. For example, if you watch a movie in a strange state, you may get some positive feedback; if you kiss in a relationship, you may also get some positive feedback; but if you kiss in a strange state, you will most likely get some negative feedback... So we know that the reward function is at least related to the current state s and the action a, that is, **R(s,a)** , and assume that the reward function is known.

至此，我们已知三种环境的属性： So far, we know the properties of three environments:

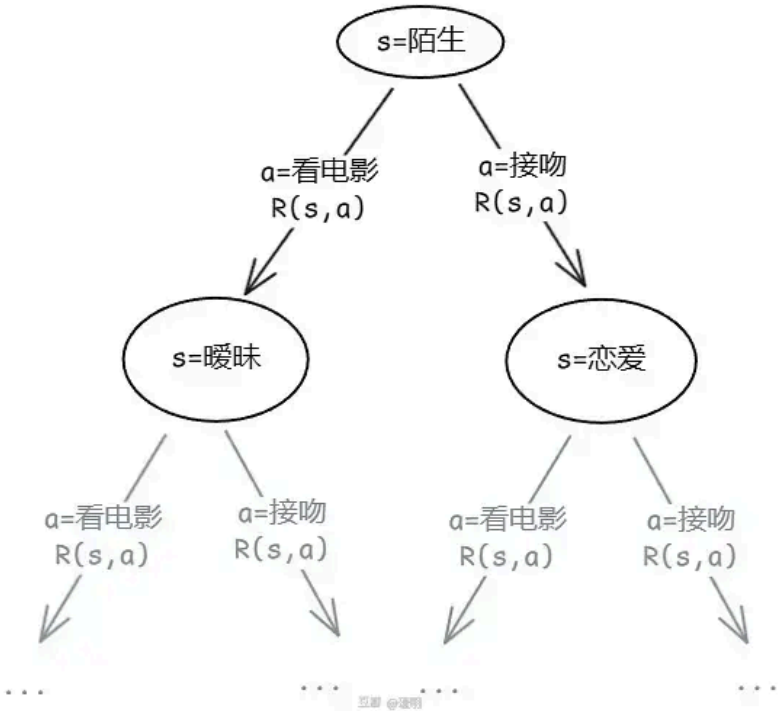
S = { s | 陌生、暧昧、恋爱}, s∈S。 start state = 陌生

S = { s | Strange, Ambiguous, Love}, s∈S. start state = Strange

A = { a | 看电影、接吻}, a∈A

A = { a | watching a movie, kissing }, a∈A

R(s,a)



引入s,a,R(s,a) Introduce s, a, R(s, a)

现在我们定义V*函数 V*(s)： 以状态s为起点，我们最多共能得到多少total Reward。

Now we define the V* function V*(s): starting from state s, how much total reward can we get at most.

如果要问：在恋爱游戏中，以某一个s为起点并做行动a，我们最多能得到多少total Reward呢?

If you ask: In the dating game, starting from a certain s and taking action a, what is the maximum total reward we can get?

此时就可以定义Q*函数：**Q*(s,a) = R(s,a) + V*(s')**，即当前这一步的奖励R(s,a) + 以s'为起点能获得的最多的total reward。

At this point, we can define the Q* function: **Q*(s,a) = R(s,a) + V*(s')** , which is the reward R(s,a) of the current step + the maximum total reward that can be obtained starting from s'.

那么，**V*(s) = max_a [Q*(s,a)] = max_a [R(s,a) + V*(s')],** 即在上面的基础上，在当前这一步，选择能获得最大R(s,a)的那个a去行动。

Then, **V*(s) = max_a [Q*(s,a)] = max_a [R(s,a) + V*(s')],** that is, based on the above, at the current step, choose the a that can obtain the maximum R(s,a) to act.

至此，我们已知两个等式： So far, we know two equations:

Q*(s,a) = R(s,a) + V*(s')

V*(s) = max_a Q*(s,a) = max_a [R(s,a) + V*(s')]

但是等等，我们忽略了什么？既然我们说“ 陌生状态下接吻，**大概率**会得到一些负反馈 ”，说明这是一个概率性事件。大概率R(s,a)很差（被打一顿，然后保持陌生状态），但也有小概率R(s,a) 很好（很开心，然后进入恋爱状态），who knows？也就是说， 对于同样的 s=陌生 和 a=接吻 ，R是可能不同的，现有的R(s,a)已经无法表达这一点了。

But wait, what did we overlook? Since we said "kissing in a strange state will **most likely** get some negative feedback", it means this is a probabilistic event. There is a high probability that R(s,a) is very bad (getting beaten and then remaining in a strange state), but there is also a small probability that R(s,a) is very good (being very happy and then falling in love), who knows? In other words, for the same s=strange and a=kissing, R may be different, and the existing R(s,a) can no longer express this.

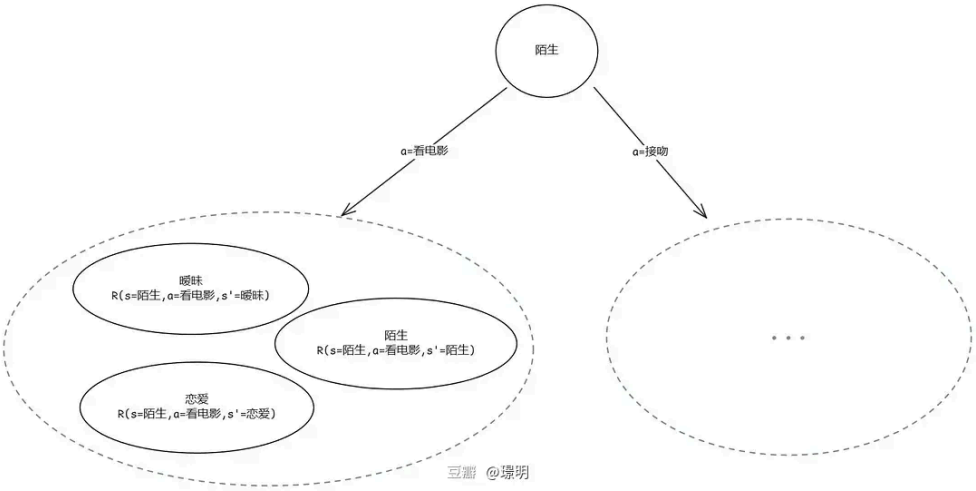
因此我们就可以优化并引入新属性： So we can optimize and introduce new properties:

4. 首先优化R(s,a)为**R(s,a,s')** 。因为它也可以与下一状态s'有关。

4. First optimize R(s,a) to **R(s,a,s')** . Because it can also be related to the next state s'.

5. 引入**状态转移函数也就是概率函数（Probability, P)**。R(s,a,s') 对应 **P(s'|s,a)，即条件s, a下，发生s'的概率。**

5. Introduce **the state transfer function, which is also the probability function (Probability, P).** R(s,a,s') corresponds to **P(s'|s,a), that is, the probability of s' occurring under conditions s and a.**



引入s,a,R(s,a,s'),P(s'|s,a)

s,a,R(s,a,s'),P(s'|s,a)

两个等式就变成了： The two equations become:

Q*(s,a) = ∑_s' P(s'|s,a) * [R(s,a,s') + V*(s')]

V*(s) = max_a Q*(s,a) = max_a ∑_s' P(s'|s,a) * [R(s,a,s') + V*(s')]

因为有了概率，左边高亮部分就是计算期望，右边高亮就是优化后的R。

Because we have the probability, the highlighted part on the left is the calculated expectation, and the highlighted part on the right is the optimized R.

至此我们已经接近完成了！不过我们还漏掉了什么： We are almost there! But we are missing something:

6. γ (念gamma) ！ 这是我们自己定义的参数（其余都是环境本身的客观属性）

6. γ (pronounced gamma) ! This is a parameter we define ourselves (the rest are objective properties of the environment itself)

标准的**Bellman最优方程**，也就是我们的两个等式，是这样的：

The standard **Bellman optimal equation** , which is our two equations, is this:

Q*(s,a) = ∑_s' P(s'|s,a) * [R(s,a,s') + γ * V*(s')]

V*(s) = max_a Q*(s,a) = max_a ∑_s' P(s'|s,a) * [R(s,a,s') + γ * V*(s')]

其中，γ ∈ [0,1] **where γ ∈ [0,1]**

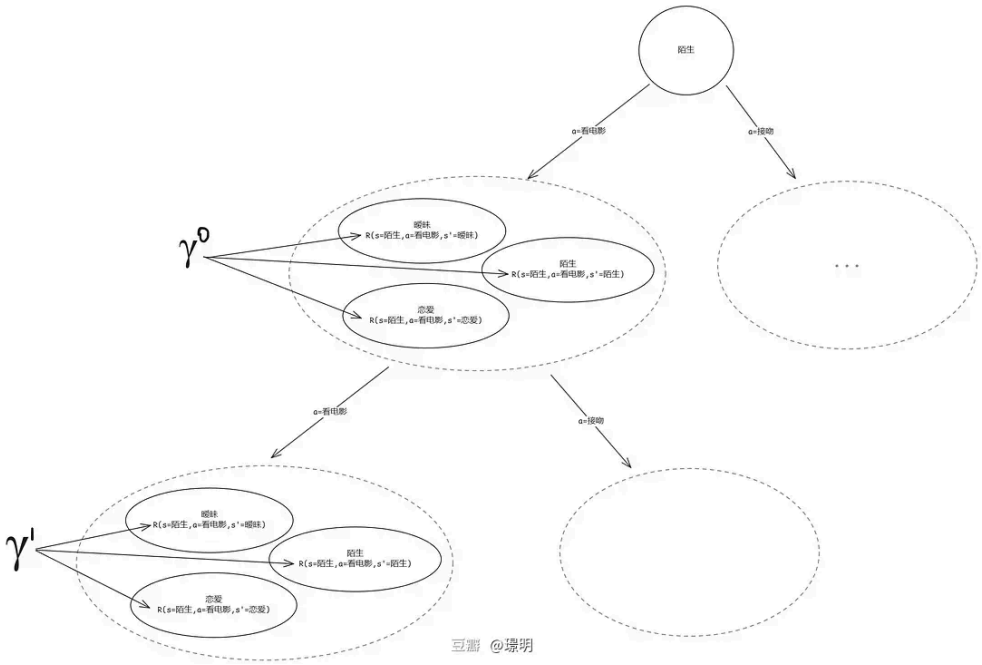
γ 的作用有两个： γ has two functions:

1. 计算上，在后续通过价值迭代or策略迭代求解最优策略时，若γ∈[0,1)能让计算收敛。如果γ=1，虽然计算不会收敛，但我们也可以使用有限时间（Finite Horizon）的方法比较不同策略哪个好。

1. Computationally, when solving the optimal strategy through value iteration or strategy iteration, if γ∈[0,1) can make the calculation converge. If γ=1, although the calculation will not converge, we can also use the finite horizon method to compare different strategies.

2. 现实意义上，由于未来有不确定性，我们可以通过控制γ的大小，平衡当前奖励和未来奖励。当γ=1时，等价于没有γ，当前奖励和未来奖励同样重要（长期主义）；当0<γ<1时，当前奖励比未来奖励更重要（一般观念）；当γ=0时，只关心当前奖励（极端及时行乐）。大家照照镜子，反思自己在恋爱关系中的γ是多少（逃

2. In practical terms, since the future is uncertain, we can balance current rewards and future rewards by controlling the size of γ. When γ=1, it is equivalent to no γ, and current rewards and future rewards are equally important (long-termism); when 0<γ<1, current rewards are more important than future rewards (general concept); when γ=0, only current rewards are concerned (extreme instant gratification). Everyone, look in the mirror and reflect on what your γ is in your relationship (escape



引入s,a,R(s,a,s'),P(s'|s,a),γ

现在我们已经完成了对恋爱的建模。 Now we have completed modeling the relationship.

总结一下，我们现在已知一个五元组（S,A,P,R,γ）：

To summarize, we now know a quintuple (S, A, P, R, γ):

S = { s | 陌生、暧昧、恋爱}, s∈S。 start state = 陌生

S = { s | Strange, Ambiguous, Love}, s∈S. start state = Strange

A = { a | 看电影、接吻}, a∈A

A = { a | watching a movie, kissing }, a∈A

R(s,a,s')

P(s'|s,a)

γ

以及 Bellman最优方程： And the Bellman optimal equation:

Q*(s,a) = ∑_s' P(s'|s,a) * [R(s,a,s') + γ * V*(s')]

V*(s) = max_a Q*(s,a) = max_a ∑_s' P(s'|s,a) * [R(s,a,s') + γ * V*(s')]

二，求解恋爱问题：找到最优策略 2. Solving the love problem: Finding the optimal strategy

策略（Policy，用π表示）是指在每种状态s下，我们分别应该做什么动作a，才能获得最大的reward之和。换句话说，为了获得最大的reward之和，我们在s=陌生时，应该看电影还是接吻，在s=暧昧时呢？在s=恋爱时呢?

Policy (denoted by π) refers to what actions a we should take in each state s to get the maximum reward. In other words, in order to get the maximum reward, when s=strange, should we watch a movie or kiss? When s=ambiguity? When s=love?

为了找到最优policy，我们需要利用Bellman最优方程，并通过迭代的方法求解。

In order to find the optimal policy, we need to use the Bellman optimal equation and solve it through an iterative method.

但在这之前，我们先完善一下已知的五元组（S,A,P,R,γ），这样才能计算：

But before that, let's perfect the known quintuple (S, A, P, R, γ) so that we can calculate:

首先，完善P和R： First, perfect P and R:

a=接吻： a=kiss:

P(s'=陌生|s=陌生,a=接吻)=1, R=-1——被拒

P(s'=strange|s=strange,a=kiss)=1, R=-1——rejected

P(s'=恋爱|s=暧昧,a=接吻)=0.5, R=+2——进步

P(s'=love|s=ambiguity, a=kissing)=0.5, R=+2——progress

P(s'=陌生|s=暧昧,a=接吻)=0.5, R=-1——被拒

P(s'=strange|s=ambiguous, a=kissing)=0.5, R=-1——rejected

P(s'=陌生|s=恋爱,a=接吻)=0.1, R=-1——原因你猜，被分手

P(s'=stranger|s=love,a=kiss)=0.1, R=-1——Guess the reason: breakup

P(s'=恋爱|s=恋爱,a=接吻)=0.9, R=+2——加深亲密

P(s'=love|s=love,a=kiss)=0.9, R=+2——deepen intimacy

a=看电影： a=watch a movie:

P(s'=暧昧|s=陌生,a=看电影)=1, R=+1——进步

P(s'=ambiguous|s=strange,a=watching a movie)=1, R=+1——progress

P(s'=暧昧|s=暧昧,a=看电影)=0.5, R=+1——加深亲密

P(s'=ambiguity|s=ambiguity,a=watching a movie)=0.5, R=+1——deepen intimacy

P(s'=恋爱|s=暧昧,a=看电影)=0.5, R=+2——进步

P(s'=love|s=ambiguity, a=watching a movie)=0.5, R=+2——progress

P(s'=恋爱|s=恋爱,a=看电影)=0.8, R=+1——加深亲密

P(s'=love|s=love,a=watching movies)=0.8, R=+1——deepen intimacy

P(s'=陌生|s=恋爱,a=看电影)=0.2, R=-1——对电影有巨大分歧，被分手

P(s'=strange|s=love,a=watching a movie)=0.2, R=-1——huge disagreement on the movie, breakup

然后，人为设定γ=0.5（为了快速收敛） **Then, we artificially set γ=0.5 (for fast convergence)**

最后，因为没有end state（通常end state的V(s)=0），所以我们必须初始化 V_0(s) = 0。

Finally, since there is no end state (usually the end state has V(s) = 0), we must initialize V_0(s) = 0.

Tips: 只要γ属于[0,1)，除了用0初始化，别的初始化方法也可以（随机初始化等等），只影响收敛速度，最终都会收敛到全局唯一解。

Tips: As long as γ is in [0,1), in addition to initializing with 0, other initialization methods are also acceptable (random initialization, etc.), which only affects the convergence speed and will eventually converge to a globally unique solution.

求解方法一：价值迭代（Value Iteration）

Solution 1: Value Iteration

步骤： step:

1. 初始化V_k(s)=0，k=0。 1. Initialize V_k(s)=0, k=0.

2. 然后迭代V_k+1(s)。 2. Then iterate V_k+1(s).

V_k+1(s) = max_a ∑_s′ P(s′|s,a) * [R(s,a,s′) + γ * V_k(s′)]

3. 最后提取最优策略 3. Finally, extract the optimal strategy

π*(s) = argmax_a ∑_s′ P(s′|s,a) * [R(s,a,s′) + γ * V*(s′)]

	陌生	暧昧	恋爱
States	陌生	暧昧	恋爱
V_0	0	0	0
V_1	1.0	1.5	1.7
V_2	1.75	2.3	2.515
V_3	2.15	2.70375	2.91925
π^*	看电影	看电影	接吻

V_3和V_2差值不大，接近收敛。 The difference between V_3 and V_2 is not large and close to convergence.

提取最优策略：π*(s=陌生)=看电影， π*(s=暧昧)=看电影， π*(s=恋爱)=接吻。

Extract the optimal strategy: π*(s=stranger)=watching a movie, π*(s=ambiguous)=watching a movie, π*(s=love)=kissing.

求解方法二：策略迭代（Policy Iteration）

Solution 2: Policy Iteration

步骤： step:

1. 初始化policy，即π_k(s)，k=0 1. Initialize the policy, i.e. π_k(s), k=0

2. 策略评估（policy evaluation）

2. Policy evaluation

V^π(s) = ∑_s′ P(s′|s,π(s)) * [R(s,π(s),s′) + γ * V^π(s′)]

V ^π (s) = ∑_s′ P(s′|s, π(s)) * [R(s,π (s) ,s′) + γ * V ^π (s′)]

得到V^π_k(s) Get V^π_k(s)

3. 策略改进（policy improvement）

3. policy improvement

π_k+1(s) = argmax_a ∑_s' P(s'|s,a) * [R(s,a,s') + γ * V^π_k(s')]

得到π_k+1(s) Get π_k+1(s)

States	陌生	暧昧	恋爱
π_0	看电影	看电影	看电影
V^{π_0}	2.0	2.5	1.5
π_1	看电影	看电影	接吻
V^{π_1}	2.0	2.5	2.8
π_2	看电影	看电影	接吻

此时π_2=π_1，已经收敛。 At this time, π_2=π_1, which has converged.

得到最优策略：π*(s=陌生)=看电影， π*(s=暧昧)=看电影， π*(s=恋爱)=接吻。

The optimal strategy is: π*(s=stranger)=watching a movie, π*(s=ambiguous)=watching a movie, π*(s=love)=kissing.

三，如果信息不充分怎么办： P和R未知

3. What to do if the information is insufficient: P and R are unknown

以上是MDP的建模和求解。此时P和R都已知（称为**model-based**），所以才能迭代求解。

The above is the modeling and solution of MDP. At this time, both P and R are known (called **model-based**), so it can be solved iteratively.

如果P和R未知（称为**model-free**），该怎么办?

What if P and R are unknown (called **model-free**)?

我们就需要通过与环境的交互不断试错（trial-and-error），学习策略，这被称为**强化学习（Reinforcement Learning，RL）**，这就是下一篇的内容了。

We need to learn strategies through trial-and-error interactions with the environment, which is called **reinforcement learning (RL)** . This is the content of the next article.

RL进阶学习资料： Advanced RL learning materials:

播客：[89.逐句讲解DeepSeek-R1、Kimi K1.5、OpenAI 01技术报告——“最优美的算法最干净”](https://www.xiaoyuzhoufm.com/episode/67a1b697247d51713c868367)

Podcast: [89. Explain DeepSeek-R1, Kimi K1.5, OpenAI 01 Technical Report sentence by sentence - "The most beautiful algorithm is the cleanest"](https://www.xiaoyuzhoufm.com/episode/67a1b697247d51713c868367)

投诉 complaint

293人浏览 编辑 I 设置 I 删除

293 people viewed Edit I Settings I delete

回应 转发 赞 收藏

Reply Retweet Like Collection