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

Jingming 2025-03-23 08:03:27 Edited United States

Preface

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?

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).

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.

1. Modeling Love: Building the MDP Step by Step

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. **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. **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.

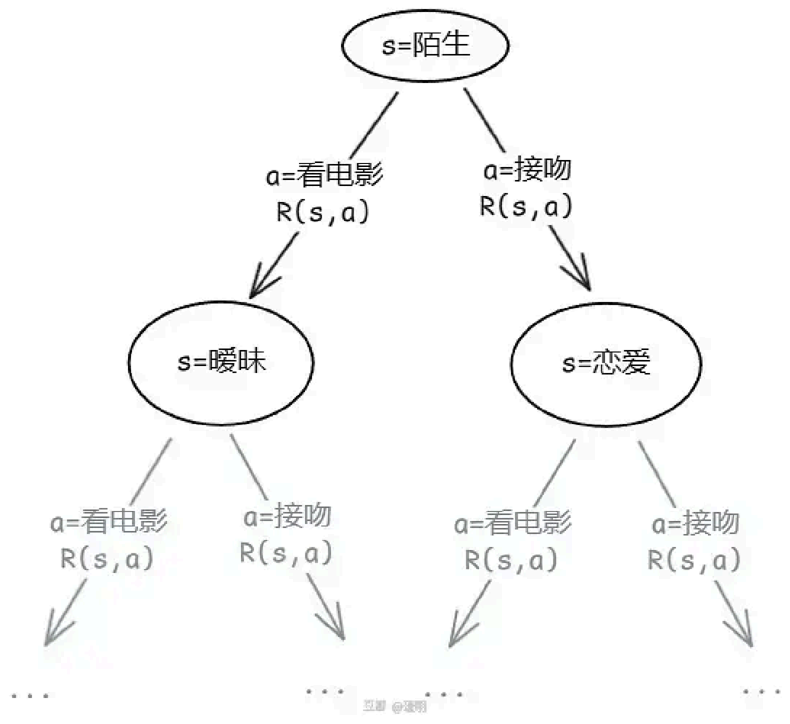
3. **Reward function or feedback function (Reward, R)** . Then, every time we take an action in a certain state, we will get some feedback from the other party. Let's assume that this feedback means intimacy. For example, if you watch a movie in a strange state, you may get some positive feedback; if 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.

So far, we know the properties of three environments:

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

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

R(s,a)



Introduce s, a, R(s, a)

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

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?

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'.

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.

So far, we know two equations:

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

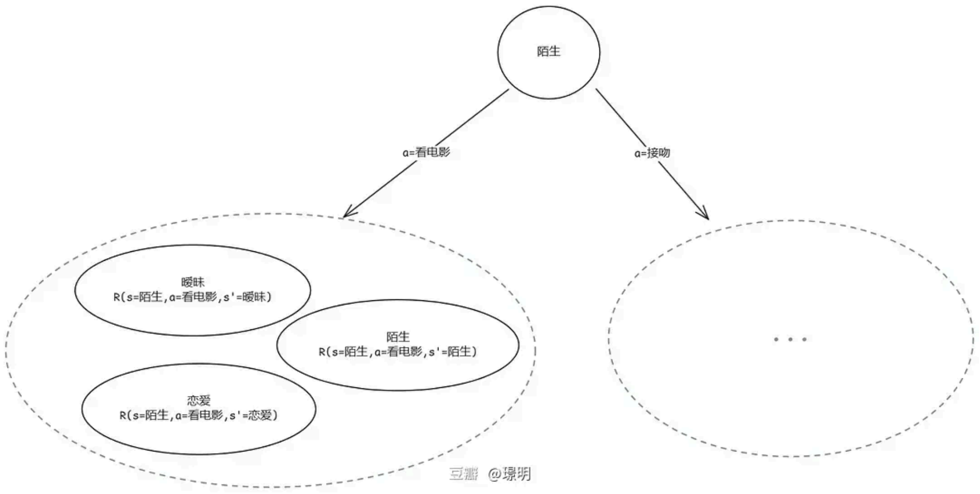
V*(s) = max_a Q*(s,a) = max_a [R(s,a) + V*(s')]

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.

So we can optimize and introduce new properties:

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

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.**


陌生
a=看电影
a=接吻
暧昧
R(s=陌生,a=看电影,s'=暧昧)
陌生
R(s=陌生,a=看电影,s'=陌生)
恋爱
R(s=陌生,a=看电影,s'=恋爱)
...
豆瓣 @璟明

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

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')]

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.

We are almost there! But we are missing something:

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

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')]

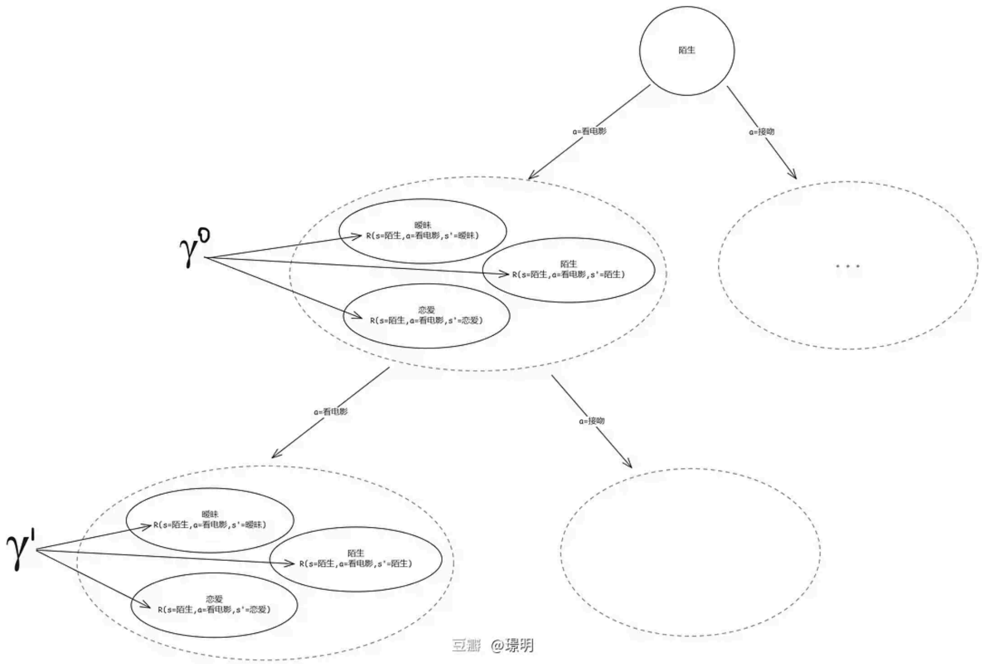
V*(s) = max_a Q*(s,a) = max_a ∑_s' P(s'|s,a) * [R(s,a,s') + γ * V*(s')]

where γ ∊ [0,1]

γ has two functions:

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. 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


陌生
a=看电影
a=接吻
γ⁰
暧昧
R(s=陌生,a=看电影,s'=暧昧)
陌生
R(s=陌生,a=看电影,s'=陌生)
恋爱
R(s=陌生,a=看电影,s'=恋爱)
. . .
a=看电影
a=接吻
γ'
暧昧
R(s=陌生,a=看电影,s'=暧昧)
陌生
R(s=陌生,a=看电影,s'=陌生)
恋爱
R(s=陌生,a=看电影,s'=恋爱)
豆瓣 @璟明

引入s,a,R(s,a,s'),P(s'|s,a),γ

Now we have completed modeling the relationship.

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

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

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

R(s,a,s')

P(s'|s,a)

γ

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')]

---

2. Solving the love problem: Finding the optimal strategy

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?

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

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

First, perfect P and R:

a=kiss:

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

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

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

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

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

a=watch a movie:

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

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

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

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

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

Then, we artificially set γ=0.5 (for fast convergence)

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

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.

Solution 1: Value Iteration

step:

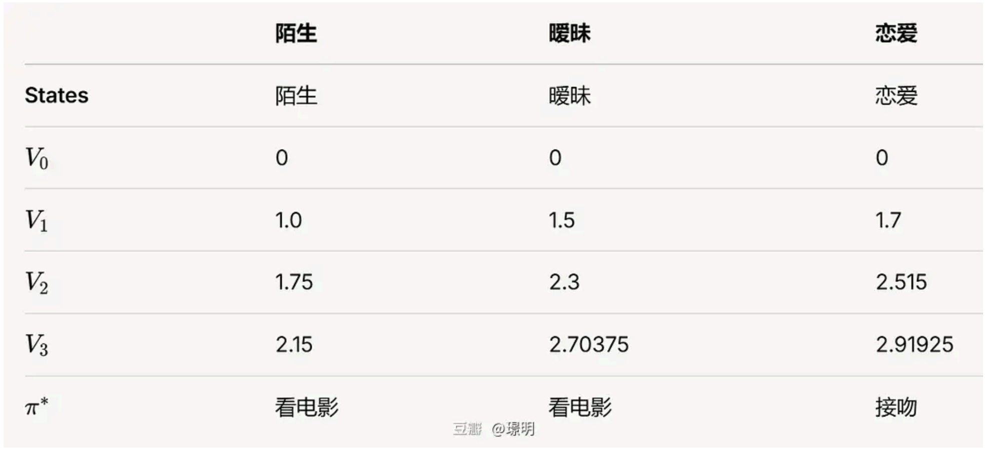
1. Initialize V_k(s)=0, k=0.

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. 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

The difference between V_3 and V_2 is not large and close to convergence.

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

Solution 2: Policy Iteration

step:

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

2. Policy evaluation

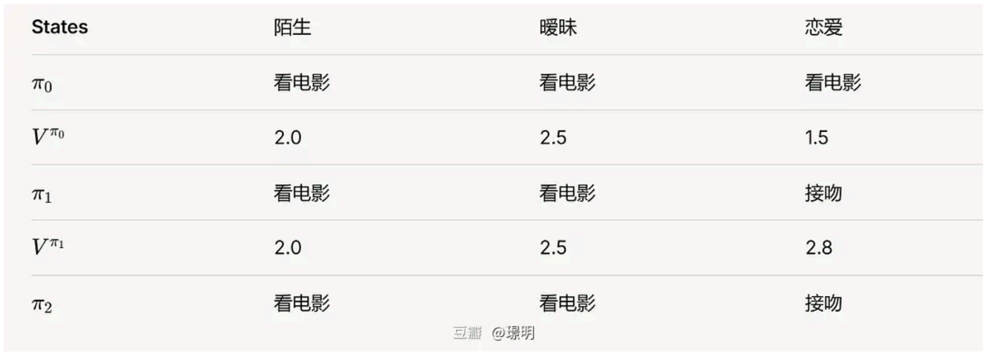
V ^π (s) = ∑_s′ P(s′|s, π(s)) * [R(s,π (s) ,s′) + γ * V ^π (s′)]

Get V^π_k(s)

3. policy improvement

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

At this time, π_2=π_1, which has converged.

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

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

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.

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

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.

Advanced RL learning materials:

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)

complaint