Aula 3

MDPs como modelo para Aprendizado por Reforço

Por que modelar RL com MDPs

Em Aprendizado por Reforço (RL), o agente toma decisões em sequência, influenciando não apenas a recompensa imediata, mas também os *próximos estados* e, por consequência, recompensas futuras. Os Processos de Decisão de Markov (MDPs) fornecem uma formalização matemática simples e geral para esse cenário de decisão sequencial com *recompensa atrasada*, permitindo definir precisamente o objetivo do agente e o comportamento do ambiente.

Interface agente–ambiente e cronologia dos sinais

Consideramos uma sequência de instantes discretos $t = 0, 1, 2, \ldots$. Em cada t, o agente observa um estado $S_t \in \mathcal{S}$, escolhe uma ação $A_t \in \mathcal{A}(S_t)$, e o ambiente devolve uma recompensa $R_{t+1} \in \mathcal{R}$ e o próximo estado S_{t+1}. A interação gera a trajetória

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, \ldots$$

A notação R_{t+1} destaca que *próximo estado e próxima recompensa são determinados conjuntamente* pelo par (S_t, A_t).

Definição de MDP finito

Um **MDP finito** é definido por conjuntos finitos de estados $\mathcal{S}$, recompensas $\mathcal{R}$ e ações admissíveis $\mathcal{A}(s)$ para cada $s \in \mathcal{S}$, e pela **dinâmica** do ambiente dada por

$$p(s', r \mid s, a) \;\triangleq\; \Pr\{S_t = s',\, R_t = r \mid S_{t-1} = s,\, A_{t-1} = a\}, \tag{1}$$

para todos $s, s' \in \mathcal{S}$, $a \in \mathcal{A}(s)$ e $r \in \mathcal{R}$. Para cada par (s, a), $p(\cdot, \cdot \mid s, a)$ é uma distribuição de probabilidade válida:

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) \;=\; 1. \tag{2}$$

Propriedade de Markov. Dizemos que a representação de *estado* possui a *propriedade de Markov* quando as distribuições de (S_t, R_t) dependem apenas do par imediatamente anterior (S_{t-1}, A_{t-1}). Em termos práticos: o estado deve condensar toda a informação do passado relevante para prever a evolução futura. Assumiremos essa propriedade ao longo do texto.

Quantidades derivadas e notações úteis

A partir de (1), é comum usar três funções derivadas:

$$p(s' \mid s, a) \triangleq \Pr\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in \mathcal{R}} p(s', r \mid s, a), \tag{3}$$

$$r(s, a) \triangleq \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a] = \sum_{r \in \mathcal{R}} \sum_{s' \in \mathcal{S}} r \; p(s', r \mid s, a), \tag{4}$$

$$r(s, a, s') \triangleq \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s'] = \frac{\sum_{r \in \mathcal{R}} r \; p(s', r \mid s, a)}{p(s' \mid s, a)}. \tag{5}$$

A função $p(s' \mid s, a)$ resume as probabilidades de transição entre estados; $r(s, a)$ é a recompensa esperada ao executar a em s; e $r(s, a, s')$ é a recompensa esperada *condicionada* em transitar para s'.

Exemplo: Navegação 1D até a estação de recarga (com ruído de ação)

Estados $\mathcal{S} = \{0, 1, 2, 3, 4, 5\}$, onde 5 é terminal (estação). Opcionalmente, incluir um componente de bateria $\{\text{alto}, \text{baixo}\}$: então o estado é o par (posição, bateria).
Ações $\mathcal{A} = \{\text{esquerda}, \text{direita}, \text{ficar}\}$.
Recompensas R: $+1$ ao alcançar o estado terminal ($i = 5$); $-0{,}01$ por passo de tempo; opcionalmente, -1 se a bateria zerar.
Transições $p(s', r \mid s, a)$: com probabilidade $0{,}8$, o movimento pretendido ocorre; com probabilidade $0{,}2$, a ação falha e o agente permanece. Se houver bateria, a cada passo o nível pode cair de "alto" para "baixo" com probabilidade q. O episódio termina ao alcançar 5.
Observações: (i) a propriedade de Markov é satisfeita ao incluir (posição e, se usada, bateria) no estado; (ii) é uma tarefa *episódica*; (iii) o desconto $\gamma \in (0, 1]$ incentiva trajetórias mais curtas até a estação.

Recompensa e Retorno

Aprofundando mais em outro aspecto importante da teoria de Reinforcement Learning, temos a recompensa, denotada por R_t. A recompensa é dada pelo ambiente para o agente, e define precisamente o *objetivo* do modelo, e não o *método* para atingi-lo. Por exemplo: caso recompensemos um robô por determinadas partes do jogo de xadrez (comer peças, por exemplo), ele pode focar nesses aspectos ao invés de desenvolver sua própria estratégia para ganhar de fato o jogo. Um dos aspectos mais valiosos dos modelos de RL é fazê-los desenvolverem seus próprios estrategemas para ganharem, e, para tal, é necessário atenção com o modo de recompensá-los.

Reward Hypothesis: estabelece-se em Reinforcement Learning que os objetivos de um modelo nada mais são na prática que a maximização do valor esperado ($\mathbb{E}$) de uma soma cumulativa (G_t/retorno) de um sinal escalar recebido do ambiente, chamado de 'recompensa'. Isto é, matematicamente, o propósito do modelo é encapsulado por:

$$\text{MAX}(\mathbb{E}[G_t]) \tag{6}$$

A questão é: como criar essa soma, G_t, de modo que possamos utilizar e determiná-la matematicamente? Em um primeiro instante, pode parecer natural afirmar que, para um cenário no qual há um *estado terminal* (isto é, que finda a simulação do agente quando este o alcança), a soma seja dada por:

$$G_t = R_1 + R_2 + R_3 + ... + R_T = \sum_{i=1}^{T} R_i \tag{7}$$

O problema é que nem sempre há estado terminal; chamamos esses eventos de contínuos, em contraste aos episódicos, discutidos no parágrafo anterior. A fim de impedir que o modelo "pense para sempre"no futuro (G_t incalculável e infinita), introduz-se um coeficiente γ que impede isso. Quando $0 < \gamma < 1$, o retorno torna-se calculável. Escrevemos então:

$$G_t = \gamma^0 R_1 + \gamma^1 R_2 + \gamma^2 R_3 + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, 0 < \gamma < 1 \tag{8}$$

Destaca-se que um γ mais próximo de 0 estimula um pensamento mais curto-prazo, enquanto um γ perto de 1 tem o efeito contrário. É possível escrever uma equação de retorno que sirva para eventos contínuos e episódicos simultaneamente, bastando considerar o evento contínuo como um caso particular do episódico, em que o estado terminal "prende"o agente em um loop dele de recompensa 0. Logo, escreve-se:

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}, \gamma = 1 \text{ ou } T = \infty \tag{9}$$

Funções Valor e Política

Para terminar de discutir mais a fundo os componentes filosófico-matemáticos iniciais do aprendizado por reforço, é necessário explicar funções valor (value-functions) e políticas (policies).

Função Valor $v_\pi(s)$

Como explicado na apostila passada, a função valor representa quão bom é estar em um determinado estado seguindo uma política específica. Dessa vez, também definiremos a função ação-valor (action-value), que nada mais é que uma versão da função valor que também leva em conta a tomada de uma ação específica dado um estado. Matematicamente, expressamos ambas como:

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi[\sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}|S_t = s] \tag{10}$$

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] = \mathbb{E}_\pi[\sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}|S_t = s, A_t = a] \tag{11}$$

Sabendo a definição matemática do valor esperado, é possível fazer algumas manipulações algébricas e expressar a função valor em um formato academicamente denominado de equação Bellmann para $v_\pi(s)$. Considerando ação a, próximo estado s', e recompensa r, tudo que está sendo feito é: para cada trio específico desses valores, multiplica-se a probabilidade dele ocorrer ($\pi(a|s)p(s', r|a, s)$) pelo valor da recompensa dada neste caso somada ao retorno futuro, sendo o somatório de todos o valor esperado. A equação Bellman será utilizada nas seções seguintes nos métodos DP.

$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_\pi(s')\right], \tag{12}$$

Funções Valor Óptimas

Para os casos de MDPs finitas, podemos definir políticas óptimas ($\pi_*(a|s)$) como aquelas que compartilham a mesma função-valor óptima ($v_*(s)$), isto é, aquela função valor definida como $v_*(s) = \max_\pi[v_\pi(s)]$. Se aplicarmos a equação de Bellman para a função-valor óptima, obtemos a equação de Bellman óptima (um nome muito criativo de fato). Esta equação implica que o valor esperado de um estado, quando seguimos uma dentre as melhores políticas possíveis, equivale ao valor esperado do retorno considerando termos tomado a melhor ação possível. Podemos expressá-la de duas formas convenientes:

$$\begin{aligned} v_*(s) &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \max_{a \in \mathcal{A}(s)} \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_*(s')\right] \end{aligned} \tag{13}$$

Há também uma função ação-valor óptima, que pode ser expressa de forma similar; ela considera o fato de tomarmos no momento uma ação que terá maior retorno, dado que no futuro sempre tomemos as melhores decisões:

$$\begin{aligned} q_*(s,a) &= \mathbb{E}\Big[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \;\Big|\; S_t = s, A_t = a\Big] \\ &= \sum_{s',r} p(s', r \mid s, a) \Big[r + \gamma \max_{a'} q_*(s', a')\Big] \end{aligned} \tag{14}$$

A equação de Bellman óptima ((13)) pode ser expandida em um sistema de N equações com N incógnitas, sendo, portanto, possível solucioná-la exatamente sempre (para MDPs finitos). Se você obteve $v_*(s)$, é trivial obter uma π_*: para qualquer estado s_t, observe qual o próximo estado s_{t+1}

disponível que tenha maior valor, e escolha-o; se sua política é não determinística, basta levar em conta a ação que maximiza a equação abaixo (é basicamente o que foi dito antes, mas considerando as probabilidades de transição de estado e de recompensa, dado que pode ser muito raro chegar nesses estados "óptimos").

$$\pi_*(s) \in \arg\max_a \sum_{s'} p(s', r \mid s, a) \left[r + \gamma v_*(s')\right]. \tag{15}$$

Em suma, se conhecemos tudo sobre nosso ambiente (ou seja, temos $p(s', r|s, a)$), é possível utilizar as equações de optimalidade de Bellman para chegar garantidamente na melhor política possível. O problema? Montar este sistema de equações e resolvê-lo requer uma busca exaustiva por todo o espaço de exploração, e isso é, para praticamente todos os problemas reais, computacionalmente infazível. A família de algoritmos a seguir (DP) busca solucionar essa limitação, mesmo que ainda assuma que tenhamos $p(s', r|s, a)$, o que também é, realisticamente, quase nunca o caso.

Programação Dinâmica

Avaliação de Políticas

A avaliação de políticas (*policy evaluation*) é o primeiro passo fundamental na programação dinâmica. Dado um ambiente conhecido e uma política fixa π, o objetivo desta etapa é calcular a função valor de estado $v_\pi(s)$, que estima o valor esperado do retorno ao seguir a política π a partir de cada estado s.

Em outras palavras, determina-se o quão bom é seguir a política π em cada estado, levando em conta todas as recompensas futuras esperadas com base nas transições estocásticas do ambiente. Esse cálculo pode ser feito de forma iterativa, atualizando sucessivamente os valores de v_π até convergência, através da equação de Bellman para uma política fixa:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_\pi(s')\right]$$

onde $\gamma \in [0, 1]$ é o fator de desconto, e $p(s', r \mid s, a)$ é a probabilidade de transitar para o estado s' e receber recompensa r ao tomar a ação a no estado s.

Melhoramento de Políticas

Uma vez avaliada a função valor para uma política π, o próximo passo é o melhoramento de políticas (*policy improvement*). A ideia é derivar uma nova política π', que seja pelo menos tão boa quanto π, utilizando a função valor v_π obtida na etapa anterior. Isso é feito escolhendo, para cada estado s, a ação que maximiza o retorno esperado segundo v_π:

$$\pi'(s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_\pi(s')\right]$$

Se essa nova política π' for estritamente melhor que π, ou seja, se $v_{\pi'}(s) \geq v_\pi(s)$ para todo s e $v_{\pi'}(s) > v_\pi(s)$ para pelo menos um estado, então ocorreu uma melhoria de política. Esse processo pode ser repetido para obter políticas cada vez melhores.

Iteração de Políticas

A iteração de políticas (*policy iteration*) combina os dois passos anteriores — avaliação e melhoramento — de forma iterativa, até que a política ótima π^* seja encontrada. O processo começa com uma política arbitrária π_0 e segue repetidamente alternando entre avaliar a política atual e melhorar com base em sua função valor:

$$\pi_0 \xrightarrow{A} v_{\pi_0} \xrightarrow{M} \pi_1 \xrightarrow{A} v_{\pi_1} \xrightarrow{M} \pi_2 \xrightarrow{A} \cdots \xrightarrow{M} \pi_n \xrightarrow{A} v_{\pi_n}$$

Nesta notação:

- $\xrightarrow{A}$ representa a avaliação de política (Policy Evaluation),
- $\xrightarrow{M}$ representa o melhoramento de política (Policy Improvement).

O algoritmo termina quando a política deixa de mudar após uma iteração de melhoramento, ou seja, quando $\pi_{k+1} = \pi_k$, indicando que uma política ótima foi alcançada. A função valor correspondente, v_{π^*}, será então a função valor ótima v^*.

A iteração de políticas é garantida de convergir em um número finito de passos em MDPs com espaços de estados e ações finitos, e produz uma política ótima determinística. Esse processo é uma das formas mais diretas e eficientes de se resolver MDPs quando o modelo do ambiente é conhecido.

Programação Dinâmica Assíncrona (Asynchronous DP)

Problema: A limitação de varreduras completas

Os métodos de Programação Dinâmica (DP) que vimos até agora possuem uma grande desvantagem: eles operam sobre todo o conjunto de estados do ambiente, exigindo varreduras completas (sweeps) a cada iteração. Para ambientes com um espaço de estados gigantesco, isso é computacionalmente inviável (por exemplo, o jogo de gamão, que possui mais de 10^{20} estado)

Mesmo que um computador pudesse atualizar um milhão de estados por segundo, levaria mais de mil anos para completar uma única varredura. Fica claro que precisamos de uma abordagem mais inteligente.

A Solução Assíncrona

Os algoritmos de DP assíncronos resolvem esse problema eliminando a necessidade de varreduras sistemáticas. Eles são algoritmos de DP iterativos e in-place (atualizam os valores diretamente, sem criar cópias) que funcionam da seguinte maneira:

- Os valores dos estados são atualizados em qualquer ordem, sem uma sequência fixa.

- A atualização de um estado utiliza os valores mais recentes disponíveis dos outros estados, mesmo que alguns deles não tenham sido atualizados há muito tempo.
- Alguns estados podem ser atualizados várias vezes antes que outros sejam atualizados uma única vez.

Para que o método possa convergir para a política ótima, existe apenas uma condição fundamental: o algoritmo deve continuar atualizando todos os estados eventualmente. Nenhum estado pode ser permanentemente ignorado.

Vantagens e Implicações para o Aprendizado por Reforço

Essa abordagem "sem varreduras"nos dá uma flexibilidade enorme e introduz uma ideia muito poderosa em Aprendizado por Reforço.

Primeiramente, podemos priorizar as atualizações. Em vez de gastar tempo com o espaço de estados inteiro, podemos focar o esforço computacional em estados que são mais relevantes ou mais importantes para o comportamento ótimo do agente.

A principal vantagem, no entanto, é a possibilidade de intercalar o cálculo com a interação em tempo real. Imagine um agente explorando um ambiente. Com um método assíncrono, podemos fazer o seguinte:

1. O agente visita um estado.
2. A experiência do agente (o estado que ele visitou) é usada para guiar o algoritmo de DP, que realiza uma atualização focada *apenas naquele estado* ou em seus vizinhos.
3. A política atualizada pelo algoritmo de DP pode, então, ser usada para guiar as próximas decisões do agente.

Isso permite focar o aprendizado nas partes do ambiente que o agente de fato experiencia. Este conceito de focar o esforço computacional em áreas relevantes é um tema central e recorrente em todo o campo de Aprendizado por Reforço.

Iteração de Política Generalizada (Generalized Policy Iteration - GPI)

Os Dois Processos Interativos

Quase todos os algoritmos de Aprendizado por Reforço podem ser descritos como uma interação entre dois processos que ocorrem simultaneamente:

- **Avaliação de Política (Policy Evaluation):** É o processo que tenta responder à pergunta: "o quão boa é a minha política atual?". Ele busca tornar a função de valor, $V(s)$, consistente com a política que o agente está seguindo no momento.

- **Melhoria de Política (Policy Improvement):** Este processo busca responder: "com base no que sei, como posso agir de uma forma melhor?". Ele ajusta a política para que ela se torne (*greedy*) em relação à função de valor atual, ou seja, para que ela passe a escolher as ações que parecem ser as melhores neste momento.

O conceito central da **Iteração de Política Generalizada (GPI)** é a ideia de deixar esses dois processos interagirem livremente para, juntos, encontrarem a solução ótima.

O que significa "Generalizada"?

Nos métodos mais simples, como o Policy Iteration, esses dois processos se alternam de forma rígida: primeiro, uma avaliação completa da política é executada até convergir; depois, uma melhoria é feita, e o ciclo se repete.

O termo ”Generalizada” significa que essa interação não precisa ser tão rígida. A granularidade dessa interação pode variar:

- No Value Iteration, por exemplo, apenas um passo de avaliação é realizado entre cada passo de melhoria.
- Nos métodos de DP Assíncrono, a granularidade é ainda mais fina: a avaliação do valor de um único estado pode ser seguida imediatamente pela melhoria da política para aquele mesmo estado.

O resultado final é o mesmo. Desde que ambos os processos continuem a ser executados, o sistema como um todo irá convergir para a função de valor ótima (v^*) e para a política ótima (π^*).

Competição e Cooperação: A Dinâmica da Convergência

Podemos pensar nesses dois processos como forças que, ao mesmo tempo, competem e cooperam.

Eles competem no curto prazo. Quando melhoramos a política (Melhoria de Política), a função de valor que tínhamos se torna incorreta para esta nova política. Por outro lado, quando atualizamos a função de valor para refletir a política atual (Avaliação de Política), a política que tínhamos deixa de ser a melhor possível (greedy). Uma força ”puxa” a outra para direções opostas.

No entanto, eles cooperam no longo prazo. Essa ”dança” entre os dois processos, onde um constantemente busca o outro, é o que move o sistema na direção da solução ótima. A estabilidade só é alcançada quando não há mais mudanças a serem feitas, ou seja, quando a política já é gananciosa em relação à sua própria função de valor. Neste ponto, encontramos a solução ótima, que satisfaz a equação de otimalidade de Bellman.