REINFORCEMENT LEARNING

AULA 3

Prof. Gian Carlo Brustolin

CONVERSA INICIAL

Sabemos como cada política de um agente inteligente pode ser descrita em função de sua utilidade. Encontrar a política ótima que leva esse agente do *estado i* para o *estado j* com maior utilidade possível: é o problema que enfrentamos empiricamente em nosso último capítulo. Agora vamos dar mais formalismo a essa técnica e entender suas aplicações.

TEMA 1 – SOLUÇÃO DO MDP

Do ponto de vista de implementação matemática ou computacional, a solução de uma cadeia de decisão de Marcov para que se encontre a política ótima não é simples. A aproximação mais comum se dá pela programação dinâmica, embora soluções lineares existam. Vamos agora concluir a descrição formal do cálculo da política ótima para em seguida entendermos os algoritmos possíveis.

1.1 Valor do estado

Em nossos estudos anteriores, conceituamos o valor do estado pelo viés da utilidade, mas também nos será útil pensar no valor do estado com base no valor do estado atual e na probabilidade de que o agente passe para um estado (s'), dado que o processo estava em um estado (s) e o agente decidiu executar uma ação "a". A notação desta probabilidade condicional será T(s'|s,a). A demonstração dessa equação nos ofertada por Pellegrini e Wainer (2007, p. 138) e terá o seguinte formato:

$$V_k^{\pi}(s) = R(s, d_k(s)) + \gamma \sum_{s' \in S} T(s, d_k(s), s') V_{k+1}^{\pi}(s')$$

onde d_k designa a k-ésima decisão de mudança de estado, tomada pelo agente, seguindo a política π. Comumente o ciclo entre tomada de decisão, ação e amostragem do meio para verificar o efeito da ação (recompensa) é chamado de *época de decisão*.

1.2 Valor da ação

Definimos de duas formas diferentes o valor de um estado. Nesta última aproximação, utilizamos a decisão do agente “d” como variável independente no cálculo do valor. De fato, a ação “a” motivada pela decisão “d” pode também receber um valor em função do resultado que dela teremos. Se o agente está no estado s, pode-se definir o valor da ação “a”, considerando a recompensa imediata de “a” e a recompensa esperada nas outras épocas de decisão, seguida a mesma política π. A esta função denotamos Q, definida como (Pellegrini; Wainer, 2007, p. 139):

$$Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s'|s,a) V^{\pi}(s').$$

Nesse ponto, convém distinguirmos objetivamente a V^{π} e Q^{π}. A função que define o valor do estado V^{π} nos informa quão bom é este estado, ao passo que Q^{π} mede quão boa é uma determinada ação neste estado. A notação V^{π} e Q^{π} indica que esses valores são obtidos se seguida a política π.

Podemos imaginar a apresentação dos valores de Q conforme uma tabela:

Tabela 1 – Apresentação dos valores de Q

Estado s	Ação	Q(a)
S_1	$a_{1,s1}$	0,05
	$a_{2,s1}$	0,3
S_2	$a_{1,s2}$	0,1
	$a_{2,s2}$	0,01

Interpretaremos a tabela da seguinte maneira: se o agente está no estado S_1, tem duas opções de ação $a_{1,s1}$ e $a_{2,s1}$. A ação $a_{1,s1}$ tem valor 0,05; já a opção $a_{2,s1}$ tem valor 0,3.

A tabela nos fornece, por hipótese, todos os valores de Q para os estados S_i. Assim, o agente escolheria a ação 2 no estado 1 e posteriormente a ação 1 no estado 2, colhendo os melhores valores de Q, que, pela formulação que vimos acima, implica as melhores recompensas R esperadas.

É possível dizer que a política ótima desse agente é aquela que faz essa escolha de ações em cada estado.

Podemos também conceituar o valor Q de uma ação em um estado s como a soma das probabilidades de mudança para o estado s_i (probabilidade de transição s - s_i) multiplicadas pelo valor descontado desses estados somados à recompensa.

$$Q(s,a)= \sum T(s_i|s)*(\gamma V_i + R_i)$$

Tomemos a tabela de valores das possíveis transições de um estado hipotético A:

Tabela 2 – Possíveis transições

Estado atual	Ação	Estado prox.	T(s|A)	R
A	a_1	A	0,3	0
A	a_1	B	0,2	2
A	a_1	C	0,4	3
A	a_2	A	0,1	0
A	a_2	B	0,5	-1
A	a_2	C	0,15	-2

Se tivermos V(A)= 1; V(B)=2 e V(C)=3, supondo o desconto unitário, o valor de $Q(A, a_1) = 0,3*(1*1+0) + 0,2*(1*2+2)+0,4(1*3+3)=3,5$

$Q(A, a_2)= 0,1*(1*1+0) + 0,5*(1*2-1)+0,15(1*3-2)=0,75$

Com esse cálculo, é possível escolher a ação a_1, uma vez que possui Q maior.

1.3 Equação de Bellman

Conceituamos o valor do estado V^{π} como a esperança de obtenção de recompensas daquele estado. Se para o mesmo estado $V^{\pi 1} > V^{\pi 2}$ a política π1 é melhor que a política π2, então, para esse estado, se seguirmos π1, obteremos uma somatória maior de recompensas.

Podemos assim concluir que a melhor política, que chamaremos de π*, é obtida para o maior V(s), ou seja, V^{π^*} ou simplesmente V*(s) denotará o valor ótimo, de máximo retorno. Escrevemos:

$$V^*(s) = max_\pi V^\pi(s)$$

Como V*(s) é o valor ótimo e identifica a maior sequência de recompensas possível para “s”, então V*(s) maximiza Q(s,a). Essa é a conclusão de Bellman, ou seja:

$$V^*(s) = max_a Q^*(s,a)$$

Neste ponto, poderíamos substituir os valore V*(s) e Q(s,a) pelas equações que vimos em 1.1 e 1.2 acima formando as duas equações clássicas de Bellman para o MDP as quais são conhecidas por serem de solução inviável. Esse fato levou Bellman a propor um método iterativo para se obterem os valores ótimos, ao qual ele nomeou *Programação Dinâmica* (DP).

1.4 Programação Dinâmica

Problemas matemáticos muito complexos podem demandar recursos indisponíveis para a solução. Bellman propôs que o problema seja quebrado em segmentos de problema. Dessa forma, resolvemos cada segmento de maneira independente e armazenamos o algoritmo de solução do segmento e a cada nova aparição de um problema com o mesmo segmento podemos reutilizar o algoritmo.

Para a solução das equações de Bellman, podemos segmentar o problema em dois: *iteração de* V^π e *iteração das várias políticas*.

TEMA 2 – SOLUÇÃO DAS EQUAÇÕES DE BELLMAN POR DP

Considerando que o equacionamento matemático do modelo ótimo, que soluciona o processo decisório para ambientes estocásticos proposto por Markov, passa pela maximização das funções de valor de estado ou de valor de ação, impossíveis de redução algébrica, Bellman propõe o uso de programação

dinâmica. Como já comentamos, duas são as possíveis dimensões de iteração, que serão examinadas a seguir.

2.1 Função de valor do estado

Chamada genericamente de *interação de valor*, essa técnica buscará a função de valor ótima e, com base nela, encontraremos a melhor política do agente.

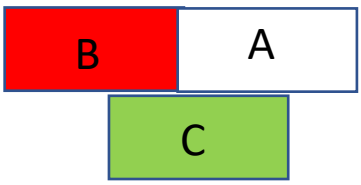
Para encontrarmos a função de valor ótima interativamente, iniciamos por uma função qualquer. A cada iteração, encontraremos uma função com melhor valor de estado até pararmos na função com maior valor, que indicará a política ótima.

Neste ponto você deve estar se perguntando como saberemos essa função inicial, já que não temos uma equação precisa para defini-la. Essa função será composta por uma escolha aleatória de valores para cada estado.

Vamos iniciar o processo recursivo com esses estados aleatórios. Calculamos, então, o valor de Q para cada estado/ação. O máximo valor de Q identificará a melhor função de valor. Atualizamos a função de valor para o máximo Q e repetimos os passos. Quando a variação do valor de Q entre duas iterações for pequeno, encerramos o processo.

Vamos tentar um exemplo elementar, para o meio a seguir:

Figura 1 – Exemplo 1



O agente está no estado A e precisa ir para C, evitando B. Vamos supor que temos a tabela de transições para o estado A, como abaixo:

Tabela 3 – Tabela de transições

Estado atual	Ação	Novo Estado	T(s|A)	R
A	baixo	A	0,3	0
A	baixo	B	0,2	2

A	baixo	C	0,4	3
A	esquerda	A	0,1	0
A	esquerda	B	0,5	-1
A	esquerda	C	0,15	-2

Vamos iniciar a primeira interação fazendo todos os valores de estado iguais a zero, V(A)=V(B)=V(C)=0. Supondo por facilidade γ=1, o cálculo dos valores de Q será:

Q(A, baixo)= 0,3*(1*0+0) + 0,2*(1*0+2)+0,4(1*0+3)=1,6

Q(A, esquerda)= 0,1*(1*0+0) + 0,5*(1*0-1)+0,15(1*0-2)=-0,8

Como Q(A, baixo) > Q(A, esquerda), vamos assumir o novo valor de V(A) como sendo 1,6 (máximo Q(A,a)).

Seguindo nosso fluxo, faremos agora os cálculos de Q(B,a) e atualizaremos V(B). Para isso, necessitamos da tabela de transições. Vamos apenas considerar, em prol da brevidade desta explicação, que fazendo esses cálculos da mesma forma que fizemos para A, cheguemos em V(B)=-1. Procedendo identicamente com C, obtivéssemos o novo V(C)=2.

Agora faremos a segunda iteração, utilizando os novos valores de V(s) obtidos na primeira iteração. Assim:

Q(A, baixo)= 0,3*(1*1,6+0) + 0,2*(1*-1+2)+0,4(1*2+3)=0,88

Q(A, esquerda)= 0,1*(1*1,6+0) + 0,5*(1*-1-1)+0,15(1*2-2)=0,16

Como Q(A, baixo) > Q(A, esquerda), vamos assumir o novo valor de V(A) como sendo 0,88 (máximo Q(A,a)). Faremos agora os cálculos de Q(B,a) e atualizaremos V(B) da mesma forma que fizemos antes, vamos supor V(B)=-0,9. Procedendo identicamente com C, o novo V(C)=0,7. E seguiríamos iterando até que as variações de Q para cada estado sejam pequenas.

A pergunta natural é: quão pequena deve ser essa diferença? De fato, não há um número preestabelecido, mas supondo que os valores de Q sejam próximos da unidade, o que é aceitável dada a multiplicação pelas probabilidades que são menores que um, variações menores que 0,01% podem ser ditas pequenas. Esse valor, nos algoritmos computacionais, é, via de regra, chamado de *valor limite* ou *threshold* em inglês e pode ser estabelecido livremente, guardado o que acima comentamos.

Vamos supor que, após n iterações, chegamos a um valor estável para V(s), segundo o *threshold* escolhido. Nesta última iteração, tivemos os seguintes resultados hipotéticos:

Tabela 4 – Relação entre estado e ação segundo o *threshold* escolhido

Estado	Ação	Valor Q
A	baixo	0,99
A	esquerda	-0,43
B	baixo	-0,31
B	esquerda	-0,21
C	baixo	0,3
C	esquerda	0,1

O maior valor de Q, que é Q(A, baixo) indica a melhor política, ou seja, com base em A podemos ir a C evitando B.

2.2 Política ótima

Uma possibilidade alternativa para o cálculo da melhor política é nos utilizarmos de uma política aleatória e testarmos sua otimicidade e então iteramos novas políticas até encontrar a melhor delas. Esse método, como veremos a seguir, é um pouco mais complexo e utiliza boa parte do que já estudamos acima. Por esse motivo, é preterido ao anterior.

De forma análoga à iteração de valor, devemos iniciar o processo propondo uma política aleatória. Uma política é de fato uma decisão de ação. Assim, o que se faz é escolher aleatoriamente um movimento para o agente em cada estado. De forma a ficar mais claro, retornemos ao nosso exemplo rudimentar. Podemos propor uma ação aleatória para os estados A, B e C. Por exemplo (A, baixo); (B, direita); (C, baixo). Essa política indica o que o agente deve fazer se estiver em cada um desses estados.

Feita a escolha aleatória, devemos calcular o valor de cada estado para essa política. Para tanto, utilizamos as recompensas obtidas por cada ação. Feito esse cálculo, podemos obter Q, como fizemos na iteração de valor. Com

os valores de Q para cada ação de um estado, podemos montar uma nova política baseados na maximização de Q.

Se a nova política for igual à política anterior, encontramos a política correta; caso contrário, atualizamos a política e calculamos novamente o valor e Q.

Como observamos, essa técnica de iteração por política introduz mais um passo computacional em relação à iteração por valor e, por esse motivo, não é a mais utilizada.

TEMA 3 – MODELOS ALTERNATIVOS DE MARKOV

Nos tópicos que estudamos até o momento, fizemos algumas suposições implícitas sobre as variáveis disponíveis para o MDP. Algumas dessas suposições podem tornar difícil a solução de problemas reais. Apresentaremos a seguir algumas modelagens que permitem o uso do processo de Markov mesmo em ambientes multivariados e ruidosos.

3.1 Modelos ocultos de Markov – HMM

Quando um agente enfrenta um meio estocástico, muitas variáveis estão naturalmente presentes. Em nosso estudo do MDP, nos baseamos na suposição de que temos conhecimento das recompensas que o meio oferece para cada ação tomada. Isso implica um meio plenamente observável. Também levamos em conta o pressuposto de que a descrição da ação pode ser feita por uma única variável determinística em relação ao resultado da ação, ou seja, nos exemplos que demos, o agente move-se para a direita toda vez que tomamos a decisão de movê-lo para a direita, mas isso pode não ocorrer em um ambiente real.

Os modelos ocultos de Markov (HMM – *Hidden Markov Models*, em inglês) tentam sintetizar essas incertezas em uma única variável estocástica. Dito de outra forma, em um ambiente real, que sempre será estocástico, não podemos ter certeza do resultado de leitura dos sensores do agente (de forma que se obtenha a recompensa) nem da efetividade da ação do agente. Podemos pensar que, mesmo que o agente decida virar para a direita, algo no meio pode mantê-lo imóvel ou produzir um giro menor que o programado pela política, por

exemplo. Dessa forma, é preciso dar à ação “a” uma probabilidade de transição para o estado ao qual se almeja.

Seguindo a mesma linha de pensamento, mesmo que o agente execute o giro pretendido pela ação “a”, os sensores podem sofrer ruído e indicarem um posicionamento não totalmente verdadeiro para o agente, o que pode orientar incorretamente as futuras decisões. Cria-se aqui uma probabilidade da recompensa, ou seja, receber uma recompensa também não é um fato objetivo e determinado. Várias aproximações matemáticas tentam solucionar essas variações estatísticas.

Uma aproximação possível é compor todas as incertezas em uma *supervariável* que expresse, de maneira única, o total das incertezas do meio. Outra maneira de ver a supervariável é dizer que a característica não determinística do meio é resultado da ação de outras variáveis parcialmente, ou totalmente, desconhecidas (Norvig, 2013, p. 506). Em nosso exemplo, do agente relutante, podemos dizer que a topografia do terreno é uma variável que influencia diretamente na efetividade da ação. Se o agente estiver em um declive, a decisão de ir em frente, por exemplo, pode resultar em uma mudança abrupta de posição (estado) não prevista pela política. Nesse caso, pode-se dizer que a característica estocástica do resultado da ação não levou em conta a variável inclinação do terreno.

De qualquer forma, em ambientes reais não é possível conhecermos e amostrarmos todas as variáveis envolvidas na determinação do resultado de uma ação qualquer. Naturalmente é mais provável que um agente chegue a um estado por determinação de uma ação que tem esse objetivo do que logre outro estado qualquer como resultado da mesma ação. Esse fato justifica uma aproximação estatística do resultado das ações.

A ideia da supervariável muito se assemelha à preparação de dados que se faz em ML, recompondo o *dataset*. Trata-se aqui de buscar uma representação composta das variações estatísticas, tornando então viável a solução do problema pelo MDP, conforme estudamos.

Um exemplo interessante, do uso do HMM, é o reconhecimento de fala. Nesse processo, o reconhecimento de fonemas pode ser descrito como a variável independente. Ocorrem, entretanto, variações nesses fonemas em função dos fonemas próximos. Da mesma forma, para que se possa seguir no

algoritmo de busca de significado, a identificação do término de uma palavra não é clara. Um humano, ao falar, pode dar intervalos maiores ou menores entre palavras ou mesmo omiti-los. Assim, há outras variáveis além dos fonemas. A ideia clássica para contornar o problema é criar fonemas dependentes de contexto à esquerda, à direita e trifones (bidependentes). Essa aproximação cria uma supervariável que contorna em boa parte um problema multivariável. É, portanto, uma técnica HMM (Ynoguti, 1999).

3.2 Markov para ambientes ruidosos

Markov desvinculou relativamente o passado do futuro mantendo somente o vínculo do futuro com o presente para permitir as conclusões que estudamos. Para tanto, pressupôs um ambiente completamente observável. Neste ambiente o estado do agente é absolutamente conhecido. Em ambientes parcialmente observáveis, o agente não sabe necessariamente em que estado se encontra. Essa incerteza pode impedir o cumprimento da política para o estado atual e dificultar a determinação da ação ótima para esse estado. Nesses meios, devemos adaptar o método de Markov. O MDP voltado a ambientes parcialmente observáveis é chamado POMDP.

Norvig (2013, p. 574) afirma que é possível utilizar os mesmos conceitos de um MDP clássico em um POMDP, bastando acrescentar um modelo de ruído para os sensores, ou seja, uma probabilidade de que os sensores nos informem o posicionamento incorreto. Assim, o estado “s” será descrito por uma probabilidade de se encontrar a evidência “e” de se estar no estado “s” estando neste estado, dita P(e | s).

O estado do agente, em referência às estruturas bayesianas, passa a ser referenciado por estado de crença “b”. O estado de crença pode ser descrito como uma distribuição de probabilidades de todos os estados possíveis no meio. Norvig (2013) descreve o estado de crença futuro b’(s’) alcançando a partir do estado de crença anterior b(s) pela ação “a”, como a seguir:

$$P(e|a,b) = \sum_{s'} P(e|a,s',b)P(s'|a,b)$$

Dessa forma, tomada uma ação em um estado de crença, sobre o qual não temos certeza, podemos ir para um novo estado de crença cuja

probabilidade de ser o estado “s” desejado é composta pela incerteza de se estar em “s” multiplicado pela incerteza do estado de crença anterior.

Um POMDP pode ser descrito em relação à previsibilidade como um estado de crença futuro depende apenas do estado de crença atual.

A solução de POMDPs por DP via iteração de valor é possível, mas devemos considerar que o resultado não é definitivo. Para cada sequência de estados de crença escolhidos teremos uma política ótima, assim devemos pensar em planos condicionais e não em política ótima, uma vez que não a encontraremos. Poderemos apenas encontrar a plano condicional mais provável.

O estudo completo da solução do POMDP não será possível em nosso curso, por isso basta-nos neste momento compreender a sua existência e a diferença que a observabilidade parcial do meio insere nos algoritmos.

TEMA 4 – *FROZEN LAKE*: O PROBLEMA DO LAGO CONGELADO

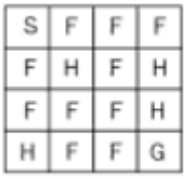
Processos de aprendizagem por reforço, como vimos no estudo até o momento, dependem do ambiente externo para que o agente possa aprender. Dessa forma, alguns ambientes de teste, baseados na simplificação de problemas reais, foram criados. Cada ambiente se adequa melhor a dado algoritmo de solução e por esse motivo se torna o campo perfeito de aprendizado daquela técnica.

Existem algumas aplicações clássicas do método de Markov equacionado por Bellman, como já aprendemos. Talvez o mais elementar seja o problema do lago congelado baseado em um jogo de labirinto simples. É um problema de posicionamento e movimento sobre um ambiente que possui penalidades e recompensas. Isso o torna perfeito para RL com uso do MDP. A maioria das bibliotecas de RL disponibilizam esse ambiente para testes, como a OpenAI Gym, que indicaremos para seus testes de algoritmos. Vamos então conhecê-lo.

A ideia é cruzar um lago congelado composto por diversas áreas. Existem áreas nas quais o gelo é seguro e o agente pode por elas transitar sem riscos, chamadas de F (*frozen areas*, em inglês). Existem também regiões de gelo fino nas quais o agente, se transitar, cairia em um buraco, denominadas H (*holes* em inglês). O início do aprendizado se dará na região de largada S (start), e o objetivo estará na área G (*goal*). Os movimentos permitidos são para cima ou

para baixo, direita e esquerda, sendo que a transição diagonal não faz parte das regras (abstrações) do meio. O número de áreas poderia ser qualquer, mas o uso é que se configure um ambiente com 4x4 ou 8X9 áreas. A disposição das áreas pode ser aleatória, mas há um ambiente clássico para testes iniciais com 16 estados, descrito a seguir:

Figura 2 – Disposição clássica do ambiente *Frozen Lake*



Por se tratar de uma superfície congelada, há uma probabilidade de que o agente escorregue, ou seja, ao tentar tomar uma direção de deslocamento, de fato vá em outra direção. Essa probabilidade será escolhida, a princípio, aleatoriamente pelo algoritmo do ambiente. Tradicionalmente a probabilidade de andar na direção e sentido escolhidos é de 33% e nos sentidos perpendiculares ao escolhido 33% em cada sentido.

Saiba mais

O algoritmo dá ao jogador uma *recompensa unitária* se ele caminhar corretamente sobre o lago e *recompensa nula* se ele cair em um buraco. Um exemplo deste algoritmo, codificado em Python, que simula o problema do lago, pode ser visto na biblioteca OpenIA Gyn e está acessível diretamente no endereço a seguir:

OPENAI GYN. Frozen Lake. **OpenAI Gym**, 1 abr. 2022. Disponível em: <https://github.com/openai/gym/blob/master/gym/envs/toy_text/frozen_lake.py>. Acesso em: 12 abr. 2022.

Podemos agora enfrentar o problema do lago congelado com a aproximação de RL por MDP, treinando um agente para que ele vença o desafio de transpor o lago. Quando o agente chegar a seu destino, terá aprendido uma política correta para o meio, mas queremos mais que uma política exitosa; desejamos que o agente encontre a política ótima, que será aquela que coleta a maior recompensa.

Vamos descrever o problema do lago, da Figura 2, segundo a aproximação do MDP. Conforme já comentamos, o conjunto de ações será a={L,R,U,D} e teremos 16 estados possíveis, correspondentes às 16 áreas do problema clássico da figura. Utilizaremos também os conceitos V(s) valor do estado, Q(s,a) valor da ação em dado estado e a decisão de qual ação tomar, dita, política.

A solução do MDP nos indicará direta ou indiretamente o melhor caminho entre o estado S e o estado G. A solução pode ser obtida por programação dinâmica pela iteração de valor ou pela iteração de políticas, ambas chegarão ao mesmo resultado.

Deixaremos ao leitor a tarefa de aplicar o algoritmo a esse caso concreto.

TEMA 5 – ALGORITMO DE MONTE CARLO

Em nosso tópico anterior, entendemos como avaliar e treinar um agente para encontrar a maior recompensa e consequentemente a política ótima em um ambiente estocástico. Fato é que na aplicação do MPD presumimos conhecido o ambiente, ou seja, sabemos quais os estados possíveis e as recompensas correspondentes mesmo que estatísticos, como no caso de POMDPs. Se o ambiente não for conhecido, a aplicação de Markov será impossível ou extremamente complexa.

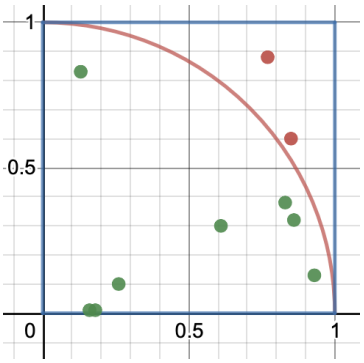
Nessas situações, métodos de inferência aproximada são mais simples e eficientes. Os algoritmos de Monte Carlo são métodos de amostragem aleatória capazes de fornecer respostas aproximadas para um problema. Da mesma forma que na estatística preditiva, a exatidão de Monte Carlo dependerá do número de amostras utilizadas.

Os algoritmos de Monte Carlo são utilizados em computação para estimar resultados cuja complexidade os torna incalculáveis, ao menos por ora. A melhor forma de compreender esse tipo de algoritmo é por meio de um exemplo de aplicação. Vejamos o cálculo de Pi (não utilizaremos a letra grega tradicional π para evitar a confusão com a variável que usamos até o momento para identificar dada política).

5.1 Cálculo do valor de Pi

A aplicação do método de Monte Carlo para a aproximação de um cálculo qualquer leva em conta a amostra aleatória do espaço de eventos. Vamos então criar um ambiente como ilustrado na Figura 3. Nesse ambiente, temos um quadrado de aresta “a” e um quarto de círculo de mesmo raio “r”. Veja que a área de um círculo é dada por πr^2. A área do quarto de círculo será então $\pi r^2 /4$. A área do quadrado será calculada por a^2.

Figura 3 – Quarto de círculo inserido em quadrado



Fonte: Ichi.Pro, [S.d.].

A probabilidade de um ponto aleatório estar dentro do quarto de círculo em relação ao quadrado será:

P(p)= área do quarto de círculo / área do quadrado

Se substituirmos a área das figuras pela sua formulação matemática, considerando que, nesse caso, r=a teremos:

$$P(p) = \frac{\frac{(\pi r^2)}{4}}{r^2}$$

$$P(p) = \frac{\pi}{4}$$

Agora, se fizermos uma amostragem de pontos aleatórios dentro do quadrado, alguns ficarão fora do quarto de círculo (chamados de “f”) e outros dentro (chamados de “d”). A probabilidade de que um ponto aleatório caia dentro do quarto de círculo será calculada também como:

$$P(p) = \frac{n(d)}{(n(d) + n(f))}$$

Portanto:

$$\frac{n(d)}{\left(n(d) + n(f)\right)} = \frac{\pi}{4}$$

Observamos no exemplo de pontos aleatórios gerados na figura anterior que: n(d) = 8 e n(f)=2, assim com esta quantidade pequena de amostras calcularíamos π como:

π = 4* (8/10) = 3,2

Uma aproximação interessante para apenas 10 amostras. Para 1.000 pontos aleatórios, o valor se torna 3,144 bem mais próximo do valor real.

Podemos concluir que, com um número suficientemente grande de amostras, obteríamos um valor muito próximo do calculado atualmente por métodos algébricos tradicionais. Essa é a ideia do método de Monte Carlo: aproximar o valor de um cálculo por amostragens aleatórias no espaço de eventos. Do ponto de vista formal, podemos escrever:

$$\lim_{N \to \infty} \frac{N(x_1, \ldots, x_n)}{N} = P(x_1, \ldots, x_n)$$

5.2 Amostragem de Gibbs

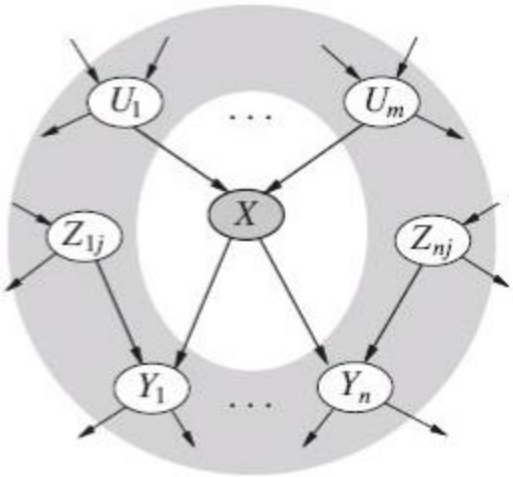
Já entendemos como o algoritmo de Monte Carlo age para solucionar um problema. Vamos agora ver uma aplicação em que se deseja calcular a probabilidade de um estado futuro em uma rede bayesiana.

Um sistema de inferência probabilístico tem por objetivo calcular a distribuição de probabilidade posterior de um dado evento. Em redes bayesianas *largas*, ou seja, cuja árvore de decisão tem muitas derivações a cada nível que descemos, o cálculo exato das probabilidades condicionadas se torna intratável (Norvig, 2013, p. 464). Nesses casos, uma aproximação do resultado pelo método de Monte Carlo é aceitável e desejada.

Selecionada uma variável de análise “v”, a amostragem de Gibbs escolhe um estado qualquer “X” e, com base nele, no espaço markoviano do entorno,

amostra os estados vizinhos em busca dos valores de “v”. Após “n” visitas aleatórias, é possível calcular qual a probabilidade da variável “v” transitar para um valor “v_1” pela relação estatística entre o número de visitas nos quais $v=v_1$ em relação ao total de visitas.

Figura 4 – Espaço marcoviano de entorno ao estado X



Fonte: Norvig, 2013, p. 454.

Estabelece-se que amostragem de Gibbs é *pertencente* à classe CMMC de algoritmos por usar o princípio da cadeia de Marcov associado ao algoritmo de Monte Carlo. A presença da hipótese de Marcov limita a busca na rede bayesiana aos pais e filhos do estado “s”, focando o processo.

5.3 Aprendizado sem modelo

Quando em um MDP não conhecemos plenamente as probabilidades de transição e de recompensas, a solução por DP se torna impossível. Nesse caso, o algoritmo de Monte Carlo é uma solução viável. Esse algoritmo fará amostragens de estados, ações e recompensas e, como inferimos do que estudamos até o momento, será capaz de aproximar o valor destas distribuições de probabilidade. Uma vez que o algoritmo não exige nenhuma modelagem prévia ou escolha de método de aprendizagem, é também chamado de *algoritmo de aprendizagem sem modelo*. Muito se teria que avançar ainda para entendermos como efetivamente aplicar Monte Carlo na solução de uma cadeia

de Markov, mas já é possível perceber seu significativo potencial de solução desse problema.

FINALIZANDO

Aqui, entendemos como solucionar um problema de aprendizado em ambiente estocástico modelado pela cadeia de Markov pelo uso de programação dinâmica. A técnica de DP para solução iterativa das equações de Bellman, entretanto, depende da disponibilidade de diversas informações estatísticas do meio e do resultado das ações que nem sempre estão disponíveis. Nesses casos de indisponibilidade de dados, o algoritmo de Monte Carlo foi apresentado como possibilidade de solução. Em nosso próximo capítulo estudaremos algumas abordagens alternativas de solução do problema de aprendizado estocástico.

REFERÊNCIAS

COPPIN, B. **Inteligência artificial**. São Paulo: Grupo Gen, 2010.

FACELI, K. et al. **Inteligência artificial** – Uma abordagem de aprendizado de máquina. 2. ed. São Paulo: Grupo Gen, 2021.

HAYKIN, S. **Redes neurais**. São Paulo: Grupo A, 2011.

ICHI.PRO. Calculando o valor de Pi: uma simulação de Monte Carlo. **Ichi.Pro**, S.d. Disponível em: <https://ichi.pro/pt/calculando-o-valor-de-pi-uma-simulacao-de-monte-carlo-233886969834463>. Acesso em: 12 abr. 2022.

NORVIG, P. **Inteligência artificial**. 3. ed. São Paulo: Grupo Gen, 2013.

PELLEGRINI, J.; WAINER, J. Processos de decisão de Markov: um tutorial. **Revista de Informática Teórica e Aplicada**, v. 14, n. 2, p. 133-179, 2007.

YNOGUTI, C. A. **Reconhecimento de fala contínua usando modelos ocultos de Markov.** Tese (Doutorado em Engenharia Elétrica). Universidade Estadual de Campinas, Campinas, São Paulo, 1999.