Aula 2

Elementos do Reinforcement Learning

O RL é estruturado em torno de quatro elementos principais: política, recompensa, função valor e modelo. Esses componentes são fundamentais para o funcionamento e o aprendizado do agente dentro de um ambiente.

Policy

A policy define a estratégia de comportamento do agente, ou seja, determina qual ação deve ser tomada em cada interação com o ambiente. Ela indica o resultado do agente ao interagir com o ambiente. Através da policy, o agente busca maximizar as recompensas acumuladas ao longo do tempo. Ela pode ser estocástica (representada com uma função de densidade de probabilidade, em que a ação é amostrada) ou determinística (a ação com maior chance sempre é a escolhida).

- $\pi(a|s_t)$ (estocástica)
- $\pi(s_t)$ (determinística)

Recompensa

A recompensa é o resultado da policy recebido pelo agente após cada interação com o ambiente. Ela indica, de forma imediata, o quão boa ou ruim foi a ação tomada em relação ao objetivo final. Em outras palavras, a recompensa orienta o agente no curto prazo, servindo como base para avaliar seu desempenho em cada etapa. É importante citar também o retorno, que é a soma descontada das recompensas futuras esperadas dado que está em um estado e seguindo uma policy (será utilizado no próximo elemento).

- $r_t \in \mathbb{R}$; a recompensa nunca é infinita
- $G_t = \sum_{k=0}^{T-t-1}(\gamma^k R_{t+k+1})$; como o modelo geralmente tem ideias mais distorcidas dos estados futuros, o retorno geral deve incluir descontos ($0 < \gamma < 1$) para estimulá-lo a valorizar mais as recompensas próximas, as quais ele tem mais certeza sobre o valor exato

Função valor

Diferentemente da recompensa, que reflete apenas o resultado imediato de uma ação, a função valor estima o retorno esperado a longo prazo. Ela representa o valor de um estado, considerando as recompensas futuras que o agente pode obter a partir dali. Essa função é essencial para evitar que o agente fique preso em "mínimos locais"— soluções aparentemente boas, mas que impedem o alcance de recompensas maiores no futuro. Assim, a função valor guia o agente na busca por recompensas cumulativas mais altas, aproximando-se do chamado "ótimo global". Matematicamente: $V^{\pi}(s_t) = \mathbb{E}[G_t|s_t]$.

Modelo

O modelo é um componente opcional no RL. Ele consiste em uma representação interna do ambiente, permitindo ao agente simular interações antes de realizá-las efetivamente. Um modelo pode prever as próximas recompensas e estados resultantes de determinadas ações, auxiliando na tomada de decisão de forma mais eficiente. Embora não seja essencial, o uso de um modelo pode acelerar o aprendizado, economizando tempo e recursos computacionais.

N-Armed-Bandit Problem

Trata-se de um problema clássico em reinforcement que introduz um conceito muito importante: o trade-off exploration-exploitation (traduzido grosseiramente para "dilema exploração-aproveitamento"). Se introduzirmos nosso agente em um novo ambiente totalmente desconhecido, ele terá varias opções diferentes para tomar, e então deverá explorá-las; mas como ele também deseja maximizar sua recompensa, quando ele deveria parar de explorar e focar em apenas se aproveitar da melhor ação que ele conhece até então? O que garante que a outra opção que ele explorou pouco/nada não teria uma recompensa maior que a atual? Os 4 algoritmos mostrados a seguir exibem algumas maneiras de lidar com isso de uma forma geral, sendo o exemplo dado na aula o do "n-armed-bandit".

O problema especificamente consiste no seguinte: suponha que o agente se depare com uma máquina de cassino ("bandit") que tem ’n’ alavancas atreladas a ela ("arms"); cada alavanca tem uma distribuição que dita sua possível recompensa quando puxada (denotemos a média de cada uma como μ_i). O papel do nosso agente é inicialmente puxar cada alavanca várias vezes (explorar), montando uma estimativa da média de cada alavanca ($\hat{\mu_i}$), até que, em determinado ponto (em que as estimativas provavelmente estão próximas das médias reais), ele pare de puxar todas e puxe apenas a com maior média ($\arg\max_a(\hat{\mu_i})$), aproveitando-se do que já sabe para maximizar sua recompensa.

Um último adendo: esse não é um problema "verdadeiramente"de reinforcement learning, já que há apenas um estado (qual alavanca puxar), sem que as ações do agente afetem isso. Nas próximas aulas problemas mais complexos serão abordados, que englobam esses problemas e, desse modo, a totalidade do RL.

Greedy/∈-Greedy

Na computação, um algoritmo greedy (chamado frequentemente de guloso em português) se baseia em um algoritmo que, dadas suas opções de ação no momento, sempre opta pela ação que lhe dará maior recompensa imediata. Essa classe abrangente geralmente provê algoritmos bons, mas raramente ideias, para os problemas enfrentados na computação.

Desse modo, um algoritmo puramente greedy exploraria pouquíssimo, já que logo que obtivesse uma recompensa, apenas repetiria a ação que o levou a ela, ignorando totalmente todas as outras opções disponíveis a ele. Isso obviamente é bem ruim, já que queremos um agente que explore um bom tanto antes de começar a ser guloso, já que a partir de um certo grau de exploração, a opção gulosa provavelmente é a melhor mesmo.

Conceitos

Chamamos de ∈-Greedy o algoritmo que se comporta como guloso mas, a cada oportunidade que tem de escolher uma ação, tem uma chance "∈"de escolher uma ação aleatória dentre as disponíveis, explorando-a e atualizando sua concepção a respeito dela ao receber a recompensa. Matematicamente, sendo A_t a ação tomada no momento 't', e $Q_t(a)$ nossa estimativa até então da verdadeira recompensa daquela ação ($q(a)$):

- para o ∈-Greedy: $Q_t(a) = \frac{r_1+r_2+...+r_{N_t(a)}}{N_t(a)}$, $N_t(a)$ = número de vezes que tomamos a ação
- $A_t = \arg\max_a[Q_t(a)]$

Uma virtude desse algoritmo é que há garantia assintótica de que chegará um ponto em que garantidamente será descoberta a melhor ação a ser tomada. Outro ponto interessante também é explorar variar $\in$ e implementar algum modo de diminuí-lo conforme o número de tentativas, refletindo a necessidade reduzida de explorar quando já se conhece bastante o ambiente.

Método do Gradiente

Conceitos

A primeira coisa que devemos pensar para implementar esse método é que nós teremos um vetor H_t que irá guardar a **preferência** que temos por cada ação, sendo todos os seus valores iniciados como 0.

A ideia é que, caso uma ação seja benéfica para o nosso agente, ou seja, ajude ele a acumular o máximo de recompensas possíveis, sua preferência deve aumentar.

A ação será escolhida por meio de uma função *softmax*, que nos da a probabilidade p_a de uma ação **a** ser esolhida:

$$p_a = \frac{e^{H_t(a)}}{\sum_{i=0}^{k} e^{H_t(i)}} \tag{1}$$

Sendo π_t o vetor que junta essas probabilidades, que é a nossa política.

Dessa forma, de acordo com as probabilidades a ação A_t é escolhida e a recompensa é dada ao agente. Então, baseada na recompensa recebida as preferências irão mudar.

Como atualizar as preferências

Para conseguirmos atualizar nossas preferências iremos utilizar da técnica de gradiente ascendente, ficando dessa forma:

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha(R_t - \bar{R}_t)\,(1 - p_{A_t})\,,$$
$$H_{t+1}(a) \doteq H_t(a) - \alpha(R_t - \bar{R}_t)p_a, \quad \text{para todo } a \neq A_t,$$

Sendo $\bar{R}_t$ a média das recompensas, incluindo a recebida no tempo t, R_t o valor da recompensa recebida em t e α um hiperparâmetro para dizer a influência do gradiente.

Analisando essas fórmulas podemos ver que a preferência da ação escolhida aumenta caso a recompensa recebida seja maior do que a média, caso contrário o parcela do gradiente ficará com sinal negativo e a $H_t(A_t)$ irá diminuir.

Também podemos ver que o parcela do gradiente possúi o termo $(1-p_{A_t})$, que representa o quanto a política pode ser ajustada para favorecer mais a ação escolhida, proporcionalmente à diferença de recompensa. Isso significa que, caso a probabilidade de escolher A_t seja baixa, $(1 - p_{A_t})$ será perto de 1, indicando que há grande espaço para aumentar a preferência por essa ação, pois ela é pouco favorecida. Por outro lado, se p_{A_t} já for alto, o termo $(1 - p_{A_t})$ será pequeno, limitando a magnitude da atualização e evitando que a política fique excessivamente concentrada em uma única ação, promovendo maior estabilidade e controle no aprendizado.

Analisando a atualização das preferências para as ações diferentes de A_t, vemos que a parcela do gradiente nesse caso é negativa, ou seja, caso a recompensa recebida pela escolha da outra ação seja menor que a média, a preferência dessas ações irá aumentar. Essa mudança é ajustada pelo termo p_a, que beneficia ações que já são bem escolhidas, seguindo o conceito de "exploit", ou prejudica elas, caso a diferença de recompensa seja positiva, para dar espaço para a ação A_t, que teve um bom desempenho

UCB

Conceitos

Não conhecemos de fato o valor da função valor para as ações ($q(a)$), pois ela diz respeito com base nas recompensas futuras. Entretanto podemos estimar essa função com base na função $Q(a)$. A ideia do UCB é balancear o tradeoff entre o quanto sabemos da função valor de ação (Q_t) e o quanto devemos explorar da função de ação, que chamaremos de de função Upper Confidence (U_t), em determinado instante t. Dito isso, vamos tentar estimar o valor da Upper Confidence para determinada ação tal que:

$$q(a) \leq Q_t(a) + U_t(a) \tag{2}$$

Então selecionamos a ação que maximiza esse valor (assim como fizemos nos algoritmos Greedy e ϵ-Greedy:

$$a_t = \arg\max_{a \in \mathcal{A}} Q_t(a) + U_t(a) \tag{3}$$

A função $U_t(a)$ pode ser entendido como a incerteza que temos sobre determinada ação a, nesse algoritmo isso será levado em conta. Além disso podemos relacionar a contagem da ação a ($N_t(a)$)com a incerteza $U_t(a)$:

- Se $N_t(a)$ é pequeno, ou seja, selecionamos/exploramos pouco a ação, então sua incerteza $U_t(a)$ é alta;
- Se $N_t(a)$ é grande, ou seja, selecionamos/exploramos bastante a ação, então sua incerteza $U_t(a)$ é baixa e sabemos mais sobre $Q_t(a)$;

Então, a ação a será escolhida se:

- ... $Q_t(a)$ é grande (sabemos então que a ação a é boa), ou
- ... $U_t(a)$ é alto (grande incerteza sobre determinada ação)
- ... ou ambos os casos

Desigualdade de Hoeffding

Para calcularmos U_t utilizaremos da Desigualdade de Hoeffding. Esse teorema diz: Suponha as variáveis aleatórias e independentes e identicamente distribuídas $X_1, X_2, ..., X_n$ no intervalo $[0, 1]$ com média $= E[X]$ e seja $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ ser a média amostral. Então:

$$p(\overline{X} + u \leq \mu) \leq e^{-2nu^2} \tag{4}$$

Com esse teorema podemos pensar nas variáveis aleatórias como R_t, a média amostral $\overline{X}$ como a função $Q_t(a)$, u como a função $U_t(a)$ e μ como $q(a)$. Note que estamos considerando recompensas no intervalo $[0, 1]$ mas podemos estender isso para outros intervalos. então temos:

$$p(Q_t(a) + U_t(a) \leq q(a)) \leq e^{-2N_t(a)U_t(a)^2} \tag{5}$$

E por simetria:

$$p(Q_t(a) - U_t(a) \geq q(a)) \leq e^{-2N_t(a)U_t(a)^2} \tag{6}$$

A ideia desse teorema é sabermos o quão longe estamos da média. Obviamente que, com mais amostras (maior $N_t(a)$), mais perto estaremos de $q(a)$ ou μ. E com isso, podemos estimar $U_t(a)$ como sendo o maior u que esteja dentro do limite dessa probabilidade. Então sabemos que nosso $q(a)$ está dentro desse intervalo. Dessa forma, selecionamos o valor p, sendo o maior possível que podemos alcançar com esse teorema:

$$e^{-2N_t(a)U_t(a)^2} = p$$

$$U_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}}$$

A ideia então é que essa probabilidade seja reduzida conforme nós temos mais recompensas (observações). Podemos então trocar p por $1/t$:

$$U_t(a) = \sqrt{\frac{\log t}{2N_t(a)}}$$

Isso garante que continuamos explorando as outras ações, conforme não sabemos sobre ela. Caso já escolhemos ela diversas vezes, sua incerteza diminui. Portanto, nosso algoritmo UCB pode ser escrito da seguinte maneira:

$$a_t = \arg\max_{a \in \mathcal{A}} Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}} \tag{7}$$

Em que c é um hiperparâmetro e no caso demonstrado, é $c = \frac{1}{\sqrt{2}}$. OBS: esse algoritmo tem valor de arrependimento que cresce logariticamente.

Thompson Sampling

Bayes (muito brevemente)

A estatística tem várias interpretações distintas; dentre elas, há a bayesiana, que vê a probabilidade de algo ocorrer como dependente do nossas crenças prévias sobre a situação e os resultados de sua amostragem. Isso é exemplificado pela equação de Bayes:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)} \tag{8}$$

- $P(A|B)$ é a posterior: chance da proposição A estar correta dada a evidência de B
- $P(B|A)$ é a likelihood: chance de ver a evidência supondo a proposição A factual
- $P(A)$ é o prior: expressa nossa crença sobre A antes de levar em conta as evidências
- $P(B)$ é a chance da evidência ocorrer

Na prática, trabalhar com uma perspectiva bayesiana envolve determinar nossos priors (nosso modelo estatístico inicial) e ir incorporando as amostragens com a likelihood, atualizando nossa modelagem até que o que acreditamos reflita bem o que está ocorrendo.

Conceitos

O método de amostragem de Thompson consiste em criar diversos priors não-informativos ("flat") sobre cada ação disponível e amostrar r_i de todas; daí, a ação escolhida na prática é a que retornou

a maior recompensa nessa "simulação"que o modelo fez da realidade. Esse ciclo de amostrar das n alavancas "teóricas", escolher puxar a que te retornou maior recompensa/retorna recompensa mais frequentemente, receber um resultado e modificar sua distribuição original, é a base desse algoritmo.

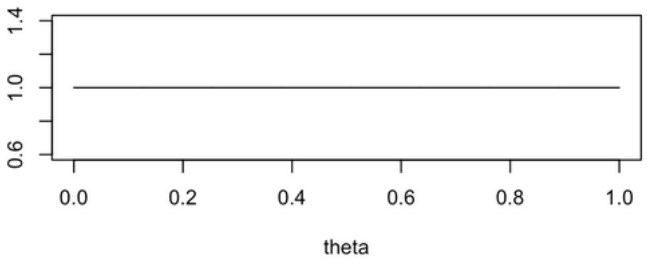

Figura 1: exemplo de flat prior: todos os possíveis valores de θ tem a mesma chance de ocorrerem

O método ficará mais claro com um exemplo: considere o 'n-armed-bandits' em que cada recompensa r_i pode ser +1 ou 0 (o caso contínuo pode ser aproximado com uma função gaussiana, como fazemos na prática exibida em aula). Nesse caso, o ideal para inicializar o algoritmo seria inicializar como prior a distribuição $Beta(\alpha_i, \beta_i)$ com $\alpha = 1, \beta = 1$. Esse prior assume como equiprovável qualquer chance entre 0 e 1 da recompensa em cada braço surgir.

A likelihood pode ser bem modelada como uma distribuição do tipo $Bernoulli(\theta)$, considerando sua natureza de recompensa. Dessa forma, o posterior também será uma função na forma $Beta(\alpha_i, \beta_i)$ (para saber o por quê, pesquise por "prior conjugado"). Deste modo, sempre que o agente estiver explorando o mundo e puxando as alavancas, o output será utilizado para atualizar o posterior, incrementando o conhecimento do ambiente e auxiliando a tomada óptima de decisões. Para uma alavanca específica, cada vez que ela retornou $+1$, aumentamos um pouco α, e caso ela não retorne nada, aumentamos um pouco o β, refletindo sua tendência de nos recompensar ou não.

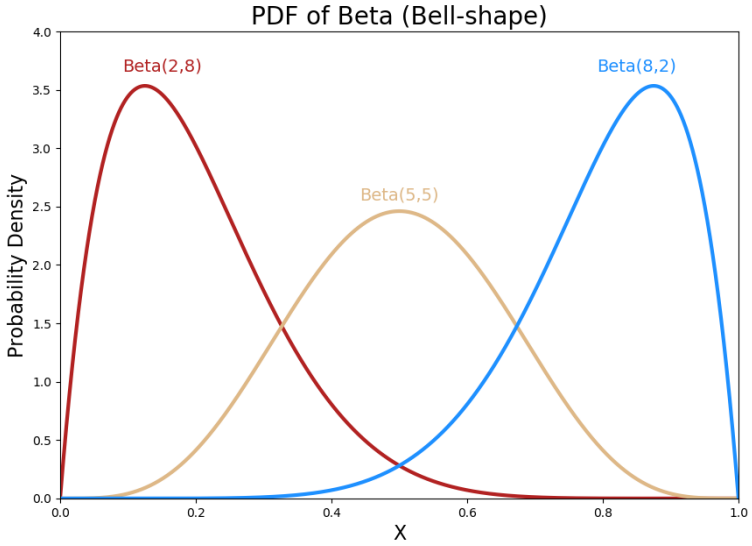

Figura 2: Beta distribution: é possível ver que aumentar β ou α influencia diretamente quão provável o agente crê que aquela ação retornará recompensas; para ambos = 1, a distribuição é idêntica à imagem anterior