FGV EMAp

Thalis Ambrosim Falqueto

Aprendizado por Reforço

Resumo

Rio de Janeiro

2025

Sumário

Prefácio

Esse resumo é e será completamente baseado no livro “Reinforcement Learning An Introduction - 2ed” de Richard S. Sutton e Andrew G. Barto e nas aulas e do github do professor Flávio Codeço Coelho.

Se alguém for ler, considere que eu estou aprendendo a matéria, e não a cursei totalmente ainda, provavelmente terão erros.

1. Introdução

1.1 Reinforcement Learning

Em vez de ser teórico, o livro gostaria de começar ensinando diretamente como simular o aprendizado computacionalmmente, sem ficar dando exemplos da vida real, como o aprendizado de bebês, etc. “O agente não terá a certeza de qual ação fazer, mas descobrirá qual a ação trará a maior recompensa por tentá-la. Nos desafios mais interessantes, as ações não irão afetar apenas o momento atual, mas sim todas as ações subsequentes”.

O problema de aprendizado por reforço será formalizado usando ideias da Teoria de Sistemas Dinâmicos, especificamente, controle ótimo do processo de decisão incompleto de Markov.

“Um agente de aprendizagem deve ser capaz de sentir o estado de seu ambiente até certo ponto e deve ser capaz de tomar ações que afetem o estado. O agente também deve ter um ou mais objetivos relacionados ao estado do ambiente. Os processos de decisão markovianos visam incluir apenas esses três aspectos — sensação, ação e objetivo — em suas formas mais simples possíveis, sem banalizar nenhum deles. “

Vale lembrar que Aprendizado por Reforço é diferente do Aprendizado Supervisionado, já que o supervisionado é um aprendizado feito com um conjunto de dados já rotulados dado por um supervisor externo (basicamente o que foi feito em Técnicas e Algoritmos para Ciência de Dados, com o Pacannaro). Além disso, Aprendizado por reforço também não é o mesmo que Aprendizado Não Supervisionado, já que essa busca encontrar estruturas escondidas em dados não rotulados.

Aprendizado por reforço é considerado como o terceiro paradigma do Aprendizado de Máquina, ao lado do Aprendizado Supervisionado e Não Supervisionado, e outros paradigmas.

Uma característica específica do Aprendizado por reforço, é o trade-off da exploração de novas possibilidades e da exploração do que já se sabe. “Para obter uma grande recompensa, um agente de aprendizagem por reforço deve preferir ações que já tentou no passado e que se mostraram eficazes na produção de recompensa. Mas, para descobrir tais ações, ele precisa tentar ações que não selecionou antes. O agente precisa explorar o que já experimentou para obter recompensa, mas também precisa explorar para fazer melhores seleções de ações no futuro.”

“Outra característica fundamental do aprendizado por reforço é que ele considera explicitamente todo o problema de um agente orientado a um objetivo interagindo com um ambiente incerto. Isso contrasta com muitas abordagens que consideram subproblemas

sem abordar como eles podem se encaixar em um cenário mais amplo. Por exemplo, mencionamos que grande parte da pesquisa em aprendizado de máquina se concentra no aprendizado supervisionado sem especificar explicitamente como tal habilidade seria útil."

O aprendizado por reforço começa com um agente completo, interativo e objetivo. Todos os agentens tem metas explícitas, podem sentir os aspectos do seu ambiente e conseguem escolher suas ações para influenciar seus resultados.

1.2 Exemplos

1. Um mestre em xadrez faz um movimento. A escolha é baseada tanto no planejamento — antecipando possíveis respostas e contra-respostas — quanto em julgamentos imediatos e intuitivos sobre a conveniência de determinadas posições e movimentos.
2. Um controlador adaptativo ajusta os parâmetros da operação de uma refinaria de petróleo emtempo real. O controlador otimiza a relação rendimento/custo/qualidade com base nos custos marginais especificados, sem se ater estritamente aos pontos de ajuste originalmente sugeridos pelos engenheiros.
3. Uma gazela luta para ficar de pé após acabar de nascer. 20 minutos depois, ela consegue correr a 20 milhas por hora.
4. Phil prepara seu café da manhã. Abrir o armário, selecionar o cereal, pegá-lo, fechar o armário, pegar uma tigela, encontrar o leite, misturar... Uma série de julgamentos são feitos, ele deve se locomover, acessar informações do seu corpo, como força, preferência de alimentação, nível de fome, etc.

Todos esses exemplos envolvem interação entre um agente tomador de decisão e seu ambiente, com o agente buscando atingir uma meta mesmo sem saber tudo sobre seu ambiente.

1.3 Elementos do Aprendizado de Máquina

Por trás do agente e do ambiente, podemos identificar quatro subelementos de um sistema de Aprendizado por reforço:

1. A **política** define o aprendizado do agente e o comportamento que ele deverá ter em dado momento. Grossamente falando, a política é um mapeamento de estados percebidos do ambiente para as ações que podem ser tomadas quando nesse estado. Em geral, pode ser estocástica, determinando probabilidades de cada ação.

2. Um sinal de **recompensa** define a meta do problema de aprendizado. A cada passo, o ambiente envia ao agente um número chamado de recompensa. O objetivo do agente é maximizar essa recompensa por todo o caminho que ele percorre. Logo, o sinal de recompensa mostra o que são eventos ruins e bons para o agente.

    - O sinal de recompensa é a base principal para alterar a política; se uma ação selecionada pela política for seguida de uma recompensa baixa, a política pode ser alterada para selecionar alguma outra ação naquela situação no futuro. Em geral, os sinais de recompensa podem ser funções estocásticas do estado do ambiente e das ações tomadas.

3. Enquanto a recompensa indica o que é bom no senso imediato do agente, a **função de valor** mostra o que é bom no longo prazo. Por exemplo, um estado pode dar uma recompensa baixa de imediato, mas ter um alto valor pois os estados que se seguem trazem recompensas maiores.
   - Podemos dizer que a recompensa são de senso primário, enquanto valores são secundários. No entanto, são os valores que mais nos preocupam ao tomar e avaliar decisões. As escolhas de ação são feitas com base em julgamentos de valor. Por isso, buscamos ações que gerem estados de maior valor, não de maior recompensa, porque essas ações nos proporcionam a maior recompensa a longo prazo.
4. O elemento final é o **modelo do ambiente**. Isso é algo que imita o comportamento do ambiente ou, de forma mais geral, que permite fazer inferências sobre como o ambiente se comportará. Por exemplo, dado um estado e uma ação, o modelo do ambiente deveria prever o próximo estado e a próxima recompensa. Os métodos para resolver problemas de aprendizagem por reforço que utilizam modelos e planejamento são chamados de métodos baseados em modelos, e são o contrário de métodos mais simples sem modelos que são explicitamente aprendizes de tentativa e erro.

1.4 Limitações e escopo

O estado é um conceito central em aprendizado por reforço, servindo como entrada para a política, a função de valor e o modelo. Informalmente, pode ser entendido como o sinal que descreve “como o ambiente está” em um dado momento.

Embora muitos métodos de aprendizado por reforço se baseiem em funções de valor, isso não é obrigatório. Métodos evolutivos (como algoritmos genéticos e programação genética) não usam funções de valor: eles testam várias políticas em paralelo, selecionam as mais recompensadas e geram novas políticas a partir delas, de forma análoga à evolução biológica. Esses métodos podem ser eficazes em certos contextos (por exemplo, quando o espaço de políticas é pequeno ou quando há bastante tempo disponível), e podem lidar bem com situações em que o agente não percebe o estado completo do ambiente.

1.5 Tic-Tac-Toe (jogo da velha)

Vamos considerar que empates e perdas são igualmente ruins. Como construir um jogador que encontre as falhas nas jogadas de um oponente(considerando um que consiga perder), que aprenda a maximizar as chances de vitória? Basicamente, esse jogo pode ser resolvido de outra forma, mas não de forma satisfatória sem o Aprendizado por Reforço. Enfim, vamos resolver usando um método de função de valor.

Primeiro fazemos uma tabela de números, um para cada estado possível do jogo. O estado A será escolhido se for maior que o estado B, e, assumindo que sempre jogaremos com os X’s, se tivermos 3 X’s em uma diagonal, linha ou coluna, então nossa probabilidade de ganhar é 1. No caso inverso, onde tem 3 bolas nessas posições, então a probabilidade de vencer é 0. Definimos o restante dos estados, inicialmente valendo 0.5, representando 50% de chance.

Continuamos jogando várias partidas contra o oponente, e para, selecionarmos os estados que iremos escolher, normalmente nos moveremos gananciosamente (greedy), buscando

o maior valor. Por vezes, mudaremos isso e escolheremos um valor aleatório, significando movimentos exploratórios, com o objetivo de experienciar outros estados.

Durante o jogo, os valores dos estados são atualizados continuamente para refletirem melhor a probabilidade de vitória. Isso é feito por meio de um processo de backup: o valor de um estado anterior $V(S_t)$ é ajustado para ficar mais próximo do valor do estado seguinte $V(S_{t+1})$

$$V(S_t) \leftarrow V(S_t) + \alpha[V(S_{t+1}) - V(S_t)] \tag{1}$$

onde α é uma pequena fração positiva chamada de parâmetro de taxa de passo(step-size), que influencia a velocidade do aprendizado. Essa regra de atualização é um exemplo de um método de aprendizado por diferença temporal (temporal-difference learning), assim chamado porque suas mudanças se baseiam em uma diferença, $V(S_{t+1}) - V(S_t)$, entre estimativas em dois instantes sucessivos.

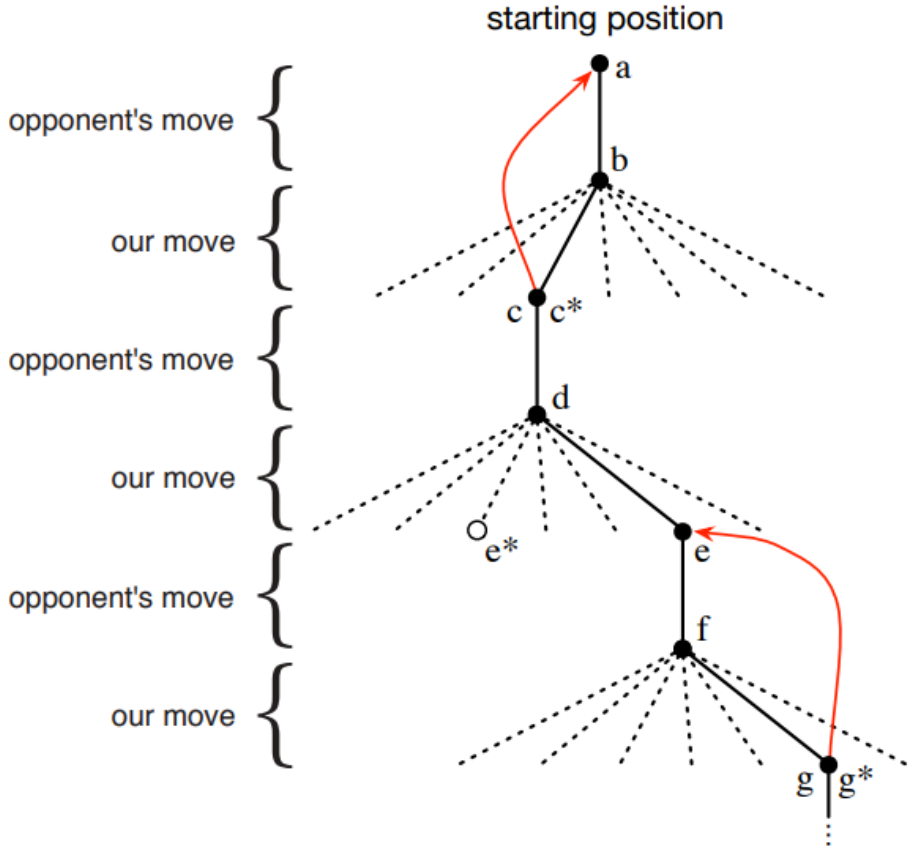

Figura 1: Uma sequência de movimentos no tic-tac-toe. As linhas pretas contínuas representam os movimentos feitos durante uma partida; as linhas tracejadas representam movimentos que nós (o agente) consideramos, mas não realizamos. Nosso segundo movimento foi um movimento exploratório, ou seja, foi realizado mesmo que outro movimento alternativo, aquele que leva a e^* tivesse classificação mais alta. Movimentos exploratórios não resultam em aprendizado, mas cada um dos outros movimentos resulta, causando atualizações conforme sugerido pelas setas vermelhas, nas quais os valores estimados são propagados para cima na árvore, de nós posteriores para nós anteriores.

- talvez um código vai ser colocado em algum lugar simulando isso??

Esse exemplo mostra a diferença entre métodos evolutivos e métodos função de valor(no caso, a solução desse problema aborda o método de função de valor, pois nós atualizamos o valor do estado a cada jogada com base na diferença temporal), onde para evoluir a política usando o método evolutivo nós usamos a mesma política e jogamos diversas vezes contra o oponente, analisando apenas a saída final (vitória/derrota). No fim, ambos os métodos buscam o espaço das políticas, mas aprender a função de valor faz com que se ganhe vantagem das informações disponíveis durante cada jogo (aparentemente, o método evolutivo não faz parte do aprendizado por reforço).

Vale lembrar que o Aprendizado por reforço é mais geral que apenas esse exemplo e que abrange vários problemas maiores com características diferentes, como os que não tem um adversário, desafios sem uma separação exata(sem começo e fim), com recompensas a qualquer momento e que também pode ser aplicado a um cenário contínuo.

“O quão bem um sistema de aprendizado por reforço pode funcionar em problemas com conjuntos de estados tão grandes está intimamente ligado à sua capacidade de generalizar adequadamente a partir de experiências passadas. É nesse papel que temos a maior necessidade de métodos de aprendizado supervisionado com aprendizado por reforço.”

Por fim, existem métodos que não precisam de nenhum tipo de modelo de ambiente em geral. O jogo da velha por si próprio é model-free no contexto de jogador, já que não há nenhum tipo de modelo para o oponente do agente. “Como os modelos precisam ser razoavelmente precisos para serem úteis, os métodos sem modelo podem ter vantagens sobre métodos mais complexos quando o verdadeiro gargalo na resolução de um problema é a dificuldade de construir um modelo de ambiente suficientemente preciso.”

1.6 Sumário

O Aprendizado por reforço é uma abordagem computacional para entender e automatizar o aprendizado e a tomada de decisão orientados a objetivos, diferenciando-se por enfatizar o aprendizado de um agente a partir da interação direta com o ambiente, sem supervisão explícita nem modelos completos. Baseia-se no formalismo de Processos de Decisão de Markov, que descrevem a interação em termos de estados, ações e recompensas, capturando causa e efeito, incerteza e a presença de metas. Um conceito central é a função de valor, fundamental para guiar a busca eficiente no espaço de políticas, sendo justamente o uso dessas funções o que distingue os métodos de aprendizado por reforço dos métodos evolutivos, que avaliam apenas políticas inteiras.

1.7 Pequena história do Aprendizado por Reforço

Desculpe, mas eu sei tão pouco sobre isso que resumir seria um pecado, pois não sei nada de história.

Parte I - Soluções de métodos tabulares

Nessa parte do livro iremos falar sobre a resolução de problemas que podem ser representados por arrays e tabelas(por isso o nome métodos tabulares). Ou seja, o espaço das ações e os estados são pequenos o suficiente para serem armazenados, cada espaço possível. Nesse caso, os métodos podem achar o valor exato das soluções, ou seja, o valor exato do valor ótimo da função e a política exata.

O primeiro capítulo dessa parte vai falar sobre problemas tabulares onde há apenas um único estado, chamado de bandit problems(ou bandidos de um braço só, ou multi-armed bandits). O segundo capítulo descreve problemas mais gerais onde iremos falar sobre processos de Markov finitos e suas principais ideias, etc.

2. Bandidos de muitos braços (Multi-armed bandits)

A característica mais importante do Aprendizado por Reforço que a difere de outros tipos de aprendizados é que ela utiliza informações de treino que avalia as ações já tomadas, ou seja, enquanto o Aprendizado Supervisionado dá um feedback instrutivo, isto é, o feedback não depende da ação tomada, no Aprendizado por Reforço, o feedback é instrutivo, ou seja, o feedback depende inteiramente da ação tomada.

Nesse capítulo vamos ver o aspecto do feedback avaliativo simplificado, que não envolve aprender a agir em mais de uma situação, ou seja, teremos sempre apenas um estado, e depois generalizaremos.

2.1 O problema do bandido k-armado

Considere o seguinte problema: você seguidamente tem que escolher entre k opções, ou ações, e depois de cada escolha você recebe uma recompensa de uma distribuição de probabilidade estacionária que depende da sua escolha.

Nota: o jogo da velha, explicado no capítulo passado, não é um problema que se enquadra como bandido k-armado, já que cada estado depende dos anteriores, e cada jogada muda o estado do tabuleiro. Como exemplo simples de um problema do bandido 1-armado, pense apenas num caça níquel, onde só existe uma ação(puxar o braço), as recompensas são sempre diferentes, e só existe um estado possível.

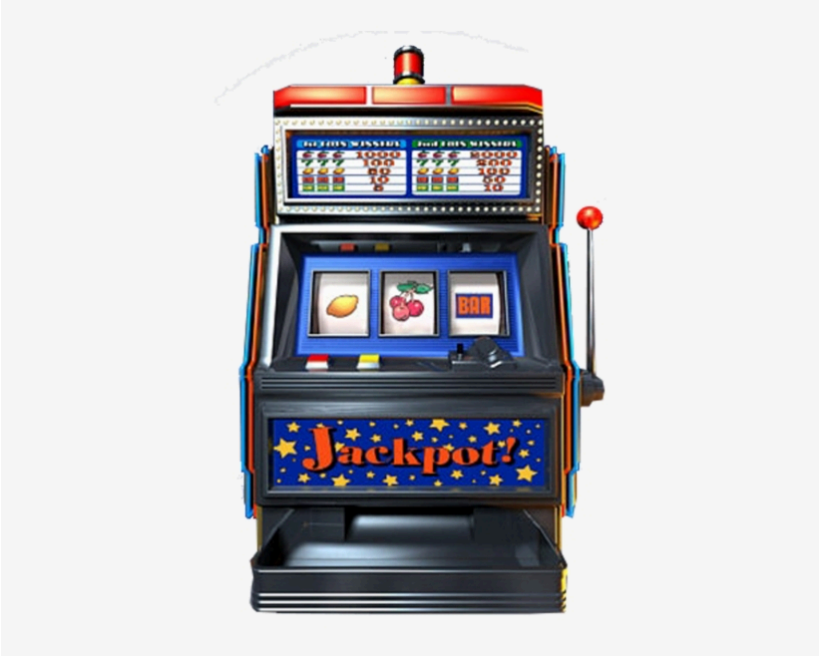

Figura 2: Caça-níquel

Nesse problema do bandido k-armado, cada uma das k-ações tem uma média esperada dada a ação selecionada(vamos chamar isso de valor da função). Nós iremos denotar a ação selecionada no tempo t como A_t, e a sua respectiva recompensa como R_t. Então, o

valor de uma ação arbitrária a, denotado $q_*(a)$, é o valor esperado da recompensa dado que a foi escolhido:

$$q_*(a) \doteq \mathbb{E}[R_t \mid A_t = a] \tag{2}$$

Se soubermos o valor de cada ação, então seria trivial para resolver o problema do bandido k-armado: basta selecionar a ação com maior recompensa. Porém, em geral, não sabemos o valor exato da ação, embora possamos ter estimadores. Denotamos o valor estimado do valor de uma ação a no tempo t como $Q_t(a)$. Nós claramente gostaríamos que $Q_t(a)$ fosse próximo de $q_*(a)$.

Se mantivermos estimativas dos valores das ações, então, em qualquer tempo t, existe pelo menos uma ação cujo valor estimado é o maior. Por isso, chamamos essas de ações gananciosas (greedy actions). Quando selecionamos uma dessas ações, dizemos que estamos explorando(exploiting) o conhecimento atual dos valores das ações.

Se, em vez disso, selecionarmos uma das ações não gananciosas, então dizemos que estamos explorando(exploring), porque permite melhorar sua estimativa do valor dessa ação não gananciosa. A exploração (exploitation) é a escolha correta para maximizar a recompensa esperada em um único passo, mas a exploração (exploration) pode gerar uma recompensa total maior a longo prazo.

Por exemplo, suponha que o valor de uma ação gananciosa seja conhecido com certeza, enquanto várias outras ações são estimadas como quase tão boas, mas com bastante incerteza. A incerteza é tal que pelo menos uma dessas outras ações provavelmente é, na verdade, melhor que a gananciosa, mas o agente não sabe qual, pois não explorou ainda.

Se o agente ainda tiver muitos passos futuros para escolher ações, pode ser melhor explorar as ações não gananciosas e descobrir quais delas são melhores que a gananciosa. A recompensa será menor no curto prazo, durante a exploração, mas maior no longo prazo porque, depois de descobrir as melhores ações, você poderá explorá-las repetidamente. Como não é possível explorar e explorar ao mesmo tempo em uma única escolha de ação, fala-se frequentemente no “conflito entre exploração(exploitation) e exploração(exploration)”.

O livro enfatiza que esse problema de balanceamento entre exploitation e exploration é recorrente, já que não podemos escolher duas ações diferentes ao mesmo tempo. Em geral, existem métodos especificos para rebalancear isso, mas normalmente são necessários fortes afirmações sobre conhecimentos do modelo que são impossíveis de verificar em aplicações completas de Aprendizado por Reforço.

2.2 Métodos baseados em valores de ações

Qual seria uma forma natural de estimar o valor de uma ação selecionada $Q_t(a)$? Intuitivamente, uma boa resposta seria a média das recompensas de quando a ação a foi escolhida, ou seja:

$$Q_t(a) \doteq \frac{\text{soma das recompensas quando a é tomada antes de t}}{\text{número de vezes que a foi tomada antes de t}} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}} \tag{3}$$

onde $\mathbb{1}_{\text{ação}}$ denota a variável aleatória que é 1 se ação é verdadeiro e 0 caso contrário. Se o denominador for zero, então definimos $Q_t(a)$ como quisermos, normalmente 0. Se o denominador tender à infinito, pela Lei dos Grandes Números, $Q_{t(a)}$ converge a $q_*(a)$. É claro que essa não é a única abordagem para estimar o valor de uma ação, e muito menos a melhor métrica.

A ação mais simples normalmente é apenas selecionar a ação de maior valor, e em caso de empate, sortear, ou ainda selecionar alguma arbitrariamente. Nós escrevemos essa seleção de ação ganansiosa como:

$$A_t \doteq \underset{a}{\operatorname{argmax}}\, Q_{t(a)} \tag{4}$$

onde argmax_a denota a ação a para a qual a expressão acima é maximizada. Ok, mas como explicado na seção anterior, normalmente escolher apenas a melhor ação sempre não é a melhor ideia.

Uma alternativa simples para escolher é se comportar de maneira gananciosa na maior parte do tempo, mas de vez em quando, com uma pequena probabilidade ε, selecionarmos aleatoriamente entre todas as ações com probabilidade igual, independentemente das estimativas de valor das ações. Nós chamamos métodos que usam essa regra de seleção quase-greedy de métodos ε-greedy.

Uma vantagem desses métodos é que, no limite, à medida que o número de passos aumenta, toda ação será amostrada um número infinito de vezes, garantindo assim que todas as estimativas convirjam para o valor verdadeiro. Isso implica que a probabilidade de selecionar a ação ótima convirja para maior que $1 - \varepsilon$ ou seja, para quase certeza.(como?)

2.3 O banco de teste de 10 braços

Para avaliar o efeito de um método totalmente ganancioso de um método ε-ganancioso, nós os compararemos numericamente em um conjunto de problemas. Esse foi um conjunto de 2000 problemas do bandido k-armado com $k = 10$. Para cada valor da ação, $q_*(a) \sim \mathbb{N}(0, 1)$, $a = 1, ..., 10$(lembre que como $k = 10$, temos 10 ações possíveis). Então, quando um método de aprendizado era aplicado e selecionava a ação A_t no instante t, a recompensa real $R_t \sim \mathbb{N}(q_*(a), 1)$, ou seja, uma distribuição normal centrada em $q_*(a)$ e variância 1.

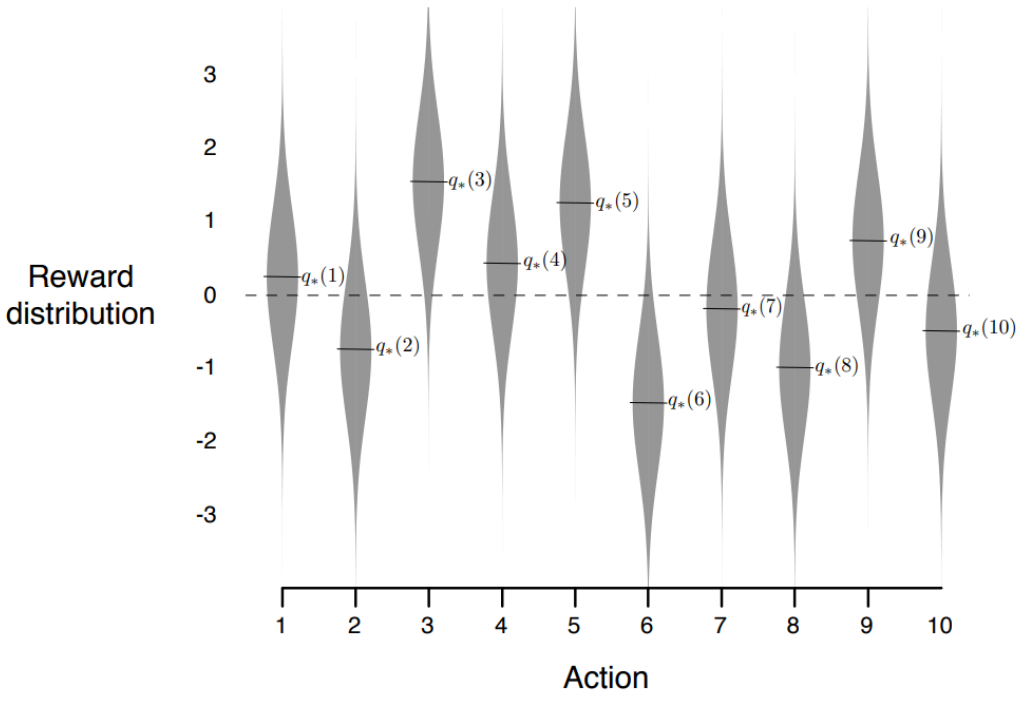

Figura 3: Um exemplo do problema do bandido 10-armado. O valor real $q_*(a)$ de cada uma das dez ações foi selecionado de acordo com uma distribuição normal com média zero e variância unitária e as recompensas reais foram selecionadas de acordo com uma distribuição normal de média $q_*(a)$ e variância unitária, conforme sugerido por essas distribuições em cinza.

Para qualquer método de aprendizado, nós podemos mensurar a performance e o comportamento realizando 1000 ações para cada bandido 10-armado. Isso é uma execução, faremos 2000 delas, pois temos 2000 bandidos 10-armados diferentes e tiraremos a média como estimativa.

A Figura 4 compara um método totalmente ganancioso com outros dois métodos ε-ganancioso($\varepsilon = 0.01$ e $\varepsilon = 0.1$).Todos os métodos performaram usando a média amostral. o gráfico de cima mostra o crescimento dá recompensa média com a experiência dos passos e o de baixo a porcentagem de ações ótimas.

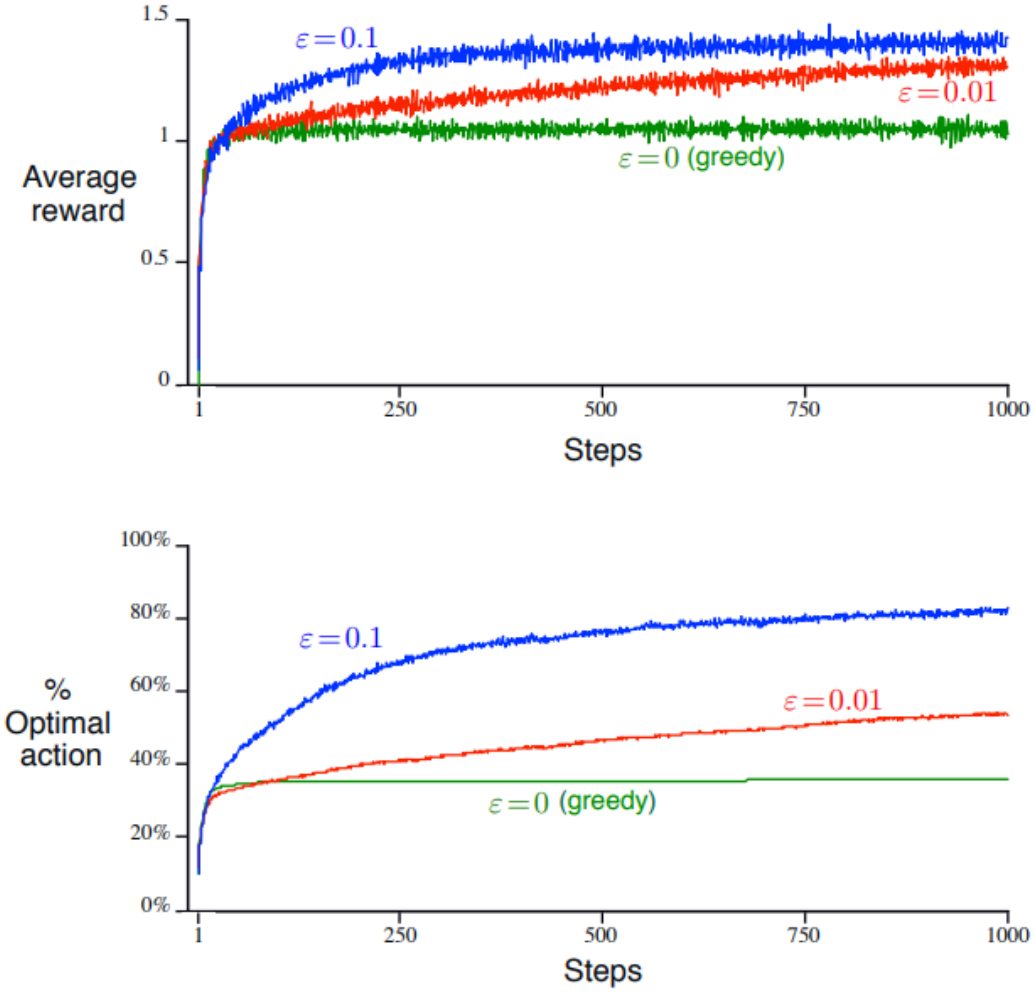

Figura 4:

Note como acontece basicamente tudo o que falamos até agora, ou seja, no extremo começo, a ação completamente gananciosa ganha, porém, depois, perde para todos os dois métodos ε-greedy. Além, note que a recompensa média do método completamente ganancioso se estagna em 1, enquanto o método menos ganancioso tem um valor médio de $1,5$, muito próximo do valor máximo da figura 3. Ainda, perceba como a porcentagem de ações ótimas do método ganancioso fica completamente estagnada, enquanto a do $0,1$-greedy chega em valores maiores que 80%.

O autor termina dizendo que em casos determinísticos, ou seja, quando a variância é 0, significa que o valor de todas as ações tem o mesmo valor, logo, o método completamente ganancioso funciona melhor. No caso contrário(não determinístico), temos uma variância(considere uma maior que 1) então usar um método ε-greedy será melhor para otimizar o valor das ações.

Ainda, em casos não estacionários, ou seja, quando o valor das ações pode mudar, significa que o ε-greedy é ainda mais importante, já que fixar-se numa ação que muda de valor é pior, sabendo que ela pode decair ainda mais, e o agente nunca saberá qual é a maior recompensa.

2.4 Implementação incremental

Como computar de maneira eficiente todos esse valores de ações com menos cálculo e memória constante?

Vamos nos concentrar em somente uma ação. Considere que R_i denota a recompensa recebida após a $i$ésima seleção dessa ação, e chame de Q_n a estimativa do valor da ação depois de ela ser selecionada $n-1$ vezes. Logo

$$Q_n \doteq \frac{R_1 + R_2 + ...R_{n-1}}{n-1}. \tag{5}$$

Parando para pensar, isso não resolveria o problema, já que mesmo assim, ao escolher a ação, teríamos que somar tudo novamente, guardar o valor novamente, etc... Mas, não precisamos disso, já que

$$\begin{aligned} Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\ &= \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) \\ &= \frac{1}{n}\left(R_n + (n-1)\cdot\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right) \\ &= \frac{1}{n}(R_n + (n-1)Q_n) \\ &= \frac{1}{n}(R_n + nQ_n - Q_n) \\ &= Q_n + \frac{1}{n}[R_n - Q_n] \end{aligned} \tag{6}$$

Legal! Conseguimos chegar em uma equação pequena que depende apenas das estimativa da ação passada e da recompensa passada.

Generalizando ainda mais, nós chegamos em uma fórmula interessante:

$$\text{NovaEstimativa} \leftarrow \text{EstimativaAntiga} + \text{PequenoPasso}\ [\text{Alvo} - \text{EstimativaAntiga}].$$

Que se parece bastante com a fórmula que vimos no Tic-Tac-Toe, mas agora entendemos de onde vêm.

- Colocar aqui o pseudocódigo talvez?

2.5 Monitorando um problema não estacionário

Os métodos discutidos anteriormente foram para problemas de bandidos estacionários, ou seja, problemas em que as recompensas não mudam de acordo com o tempo. No caso não estacionário, talvez seja melhor dar um peso maior às recompensas mais recentes. Uma sugestão para fazer isso é mudando o PequenoPasso(step-size) da atualização da estimativa da recompensa, ou seja:

$$Q_{n+1} = Q_n + \alpha[R_n - Q_n] \tag{7}$$

onde $\alpha \in (0, 1]$ e é constante. Note que essa equação é uma média ponderada das recompensas passadas e da estimativa inicial Q_1:

$$\begin{aligned} Q_{n+1} &= Q_n + \alpha[R_n - Q_n] \\ &= \alpha R_n + (1-\alpha)Q_n \\ &= \alpha R_n + (1-\alpha)[\alpha R_{n-1} + (1-\alpha)Q_{n-1}] \\ &= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 Q_{n-1} \\ &= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 \alpha R_{n-2} + \cdots \\ &\quad + (1-\alpha)^{n-1}\alpha R_1 + (1-\alpha)^n Q_1 \\ &= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i \end{aligned} \tag{8}$$

Às vezes vale a pena variar o step-size de ação para ação. Considere que $\alpha_n(a)$ o parâmetro step-size na n-ésina seleção da ação a. Um resultado conhecido na aproximação estocástica nos dá as condições requeridas para garantir convergência para o valor real da ação com probabilidade 1.

$$\sum_{n=1}^{\infty} \alpha_n(a) = \infty \quad \text{e} \quad \sum_{n=1}^{\infty} \alpha_n(a)^2 < \infty \tag{9}$$

A primeira condição garante que os passos sejam grandes o suficiente para eventualmente superar qualquer condição inicial ou flutuações aleatórias. A segunda condição garante que eventualmente os passos sejam pequenos o suficiente para garantir convergência. Geralmente essas condições são pouco usadas na prática pois podem demorar demais.

2.6 Valores Iniciais Ótimos

Todos os métodos discutidos até agora tem uma forte dependência nas estimativas iniciais de recompensa $Q_1(a)$. Na linguagem estatística, esses métodos são chamados de $\mathrm{enviesados}$ por suas estimativas iniciais de recompensa. Para métodos que calculam a estimativa pela (5) usando $\alpha_n(a) = \frac{1}{n}$, chegamos na fórmula (6), e, após a primeira estimativa, nosso $Q_n(a)$ não depende mais diretamente de Q_1, ou seja, ele não é mais enviesado. Já com outro α, no caso, chegamos na fórmula (8), que é diretamente enviesado por Q_1.

Um enviesamento pode ser bom ou ruim. Suponha que em vez de iniciar a estimativa inicial $Q_1(a) = 0$, como fazemos anteriormente, considere que colocaremos todos como $+5$(lembre que estávamos usando $q*(a) \sim \mathbb{N}(0, 1)$, ou seja, uma estimativa desse tamanho é bem otimista). Esse otimismo encoraja o agente a explorar, já que de início, todas as estimativas são mais altas do que qualquer valor real da ação. Então, ao escolher certa ação, o agente recebe uma recompensa menor do que o esperado, ficando desapontado e diminuindo o valor da recompensa esperada. Ao escolher a próxima ação, todas as outras têm uma estimativa maior do que a primeira selecionada. Assim, o sistema fará explorará todas as ações, mesmo se sempre selecionarmos a mais gananciosa sempre.

Nós chamamos essa estratégia de Valores Iniciais Ótimos. Vamos comparar esse método com o método ε-greedy explicados anteriormente:

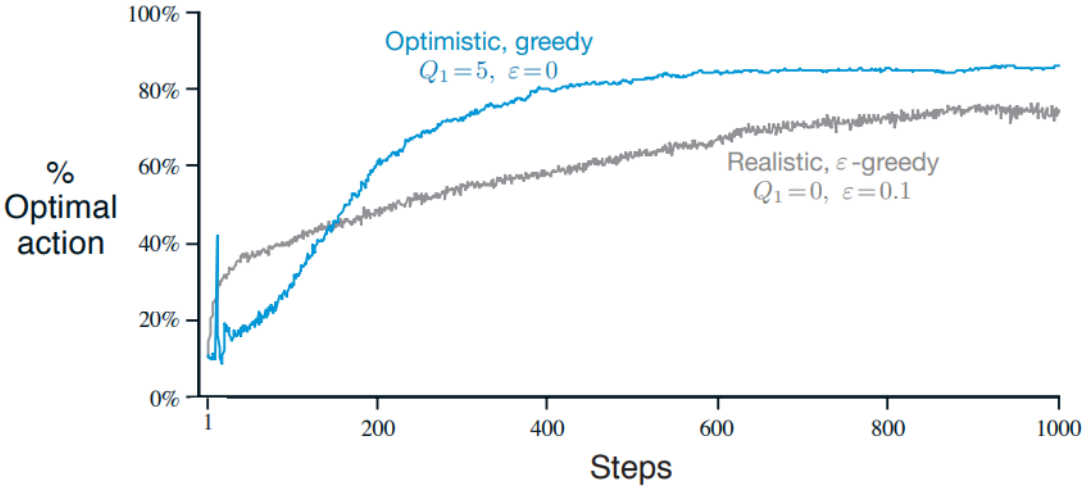

Figura 5: Efeito da inicialização otimista do valor da ação no banco de teste de 10 braços. Ambos usaram o step-size $\alpha = 0.1$

Note como o método otimista começa pior, pois o agente fica inicialmente apenas explorando várias ações, mas eventualmente performa melhor porque acaba parando de explorar. Essa estratégia pode ser boa às vezes, mas não é a mais adequada em casos não estacionários(valor das ações muda) pois a exploração de outras ações é temporário. Em geral, qualquer método que depende fortemente de condições iniciais não são muito bons para métodos não estacionários.

2.7 Seleção de Ação por Nível Superior de Confiança

Como dito e reforçado por vezes, exploração é necessário e precisamos que ela aconteça, mas que tal se explorassemos não de forma arbitrária (como no ε-greedy), mas buscando agora selecionar as ações de acordo com a seu potencial de serem ótimas, levando em conta o quão perto a estimativa está perto de ser máxima e também a incerteza de cada estimativa. Um jeito bom de fazer isso é pela fórmula

$$A_t \doteq \underset{a}{\operatorname{argmax}} \left[Q_t(a) + c\sqrt{\frac{\ln(t)}{N_t(a)}} \right], \tag{10}$$

onde $\ln(t)$ significa o logarítmo natural de t, $N_t(a)$ significa o número de vezes que a ação a foi escolhida antes do tempo t, e o número $c > 0$ controla o grau de exploração. Se $N_t(a) = 0$, então a é uma ação maximizada.

de onde vem isso

A ideia do UCB(Upper Confidence Bound), resumidamente, é que o termo da direita é um termo de incerteza que diminui quanto mais você seleciona a ação, e o mesmo termo aumenta quando você não escolhe a ação, fazendo o agente não se esquecer de nenhuma ação.

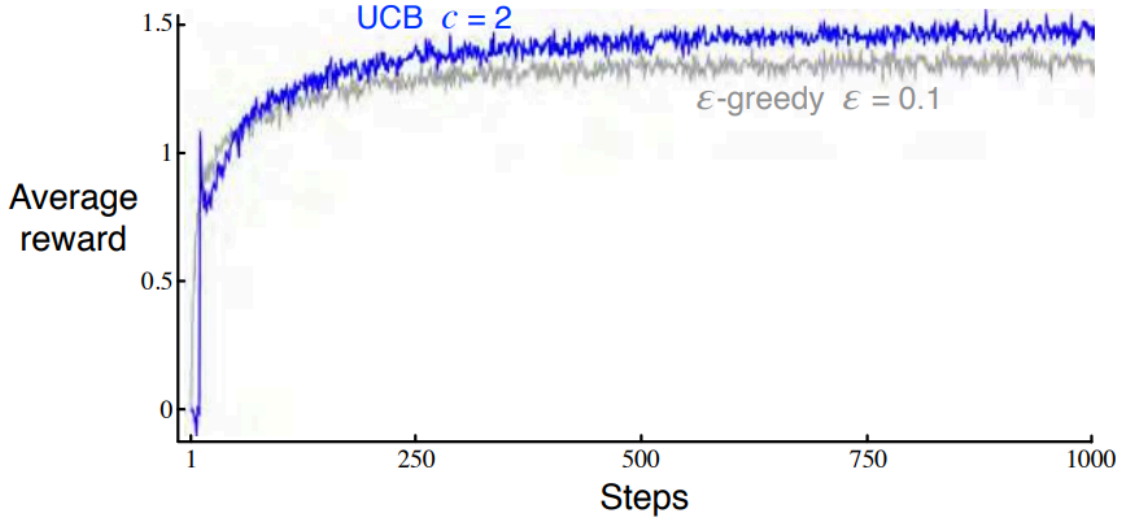

Figura 6: Performance média do método UCB para o problema do bandido 10-armado.

O UCB pode performar bem no testes do bandido 10-armado, mas o autor afirma que em problemas reais com grande espaços de estado ou com problemas não estacionários o método pode não performar bem, porque seria inviável guardar e gerenciar os valoroes de $N_t(a)$ e porque simplesmente não faz sentido usar o UCB como confiança da recompensa se as recompensas mudam, respectivamente.

2.8 Algoritmos do Bandido Baseado em gradiente

Nessa seção, vamos considerar aprender uma preferência numérica para cada ação a, denotada $H_t(a)$. Quanto maior a preferência, mais a ação será tomada, mas a preferência não tem interpretação em termos de recompensa. Perceba que apenas a preferência relativa de uma ação sobre a outra é importante, e ela é definida de acordo com uma $\text{distribuição soft-max}$ como se segue:

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a) \tag{11}$$

onde $\pi_t(a)$ é definido como a probabilidade de tomar a ação a no tempo t. Inicialmente todas as preferências são as mesmas (ou seja, $H_1(a) = 0$, para todo a). Logo todas as ações tem mesma probabilidade. Existe uma fórmula natural de aprender melhor as preferências, baseando-se na ideia do gradiente estocástico ascendente:

$$\begin{aligned} H_{t+1}(A_t) &\doteq H_t(A_t) + \alpha\Big(R_t - \bar{R}_t\Big)(1 - \pi_t(A_t)), \qquad \text{and} \\ H_{t+1}(a) &\doteq H_t(a) - \alpha\Big(R_t - \bar{R}_t\Big)\pi_t(a) \qquad\qquad \text{for all a} \neq A_t \end{aligned} \tag{12}$$

onde $\alpha > 0$ é um step-size, e $\overline{R_t}$ é a média de todas as recompensas incluindo o tempo t. O $\overline{R_t}$ funciona como referência, ou seja, se a recompensa é maior do que a média de recompensas, então a preferência para ela aumenta, e vice-versa. As ações não selecionadas se movem na direção oposta.

A Figura 7 mostra o resultados do algoritmo do gradiente ascendente em uma variante do bandido 10-armado onde as recompensas são escolhidas de uma distribução $\mathbb{N}(4, 1)$. Essa

mudança não faz com que o algoritmo que usa a referência ($\bar{R}_t$) sofra algum efeito, mas se a referência for omitida, ou seja, se $H_{t+1}(A_t) \doteq H_t(A_t) + \alpha R_t(1 - \pi_t(A_t))$, a performance será significativamente pior, como mostra a figura.

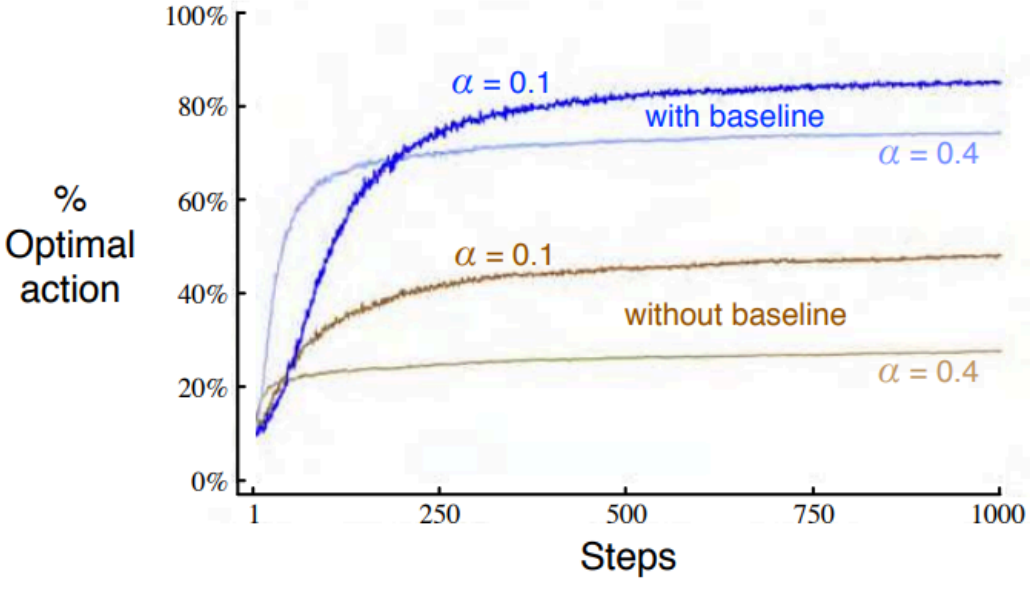

Figura 7: Desempenho médio do algoritmo do bandido 10-armado de gradiente com e sem referência quando q_*(a) está perto de 4 e não perto de 0.

olhar no livro a explicação e explicar dps

2.9 Pesquisa associativa

Em uma tarefa geral de aprendizado por reforço há sempre mais de uma situação, e o objetivo é aprender a melhor política, ou seja, o mapeamento de situações para as ações que são melhores nessas situações. Até agora, vimos apenas tarefas não-associativas, ou seja, tarefas onde não precisamos associar ações diferentes para situações diferentes.

Como exemplo, suponha que haja várias tarefas diferentes de bandidos k-armados, e que a cada passo você escolha uma delas aleatoriamente. Assim, a tarefa do bandido muda aleatoriamente de um passo para o outro. Isso pareceria para o agente como uma única tarefa não estacionária, cujo verdadeiro valore de ação mudam aleatoriamente.

Agora, suponha, no entanto, que quando uma tarefa é selecionada, o agente recebe uma pista sobre sua identidade (mas não sobre os valores de ação. Agora você pode aprender uma política que associa cada tarefa ao sinal recebido - por exemplo, se vermelho, selecionar o braço 1; se verde, selecionar o braço 2. Com a política correta você pode se sair muito melhor do que se não tivesse nenhuma pista distinguindo uma tarefa de outra.

Esse é um tipo de tarefa de pesquisa assoiativa, onde usa tentativa e erro para pesquisar a melhor ação, e associação das ações com as situações em que são melhores. Esse é um problema que intermedia o problema do bandido k-armado e o problema total do aprendizado por reforço.

2.10 Sumário

Foi apresentado várias formas de balancear exploration e exploitation, com o ε-greedy escolhendo uma ação aleatóriamente por uma pequena fração de tempo, enquanto o

método UCB escolhe deterministicamente, mas alcançam a exploração enquanto favoreciam as ações que recebiam menos amostras. O gradiente estimava não valores, mas preferências e definem as melhores ações baseando-se na preferência utilizando a softmax. Até inicializar as estivativas de recompensa otimistamente causa um bom método de exploração inicial.

É natural se questionar qual é o melhor método. Por isso, o autor fez uma plotagem de um treinamento completo em um problema de bandido k-armado, testando a média dos vários valores dos parâmetros de cada método após mil passos. Olhe:

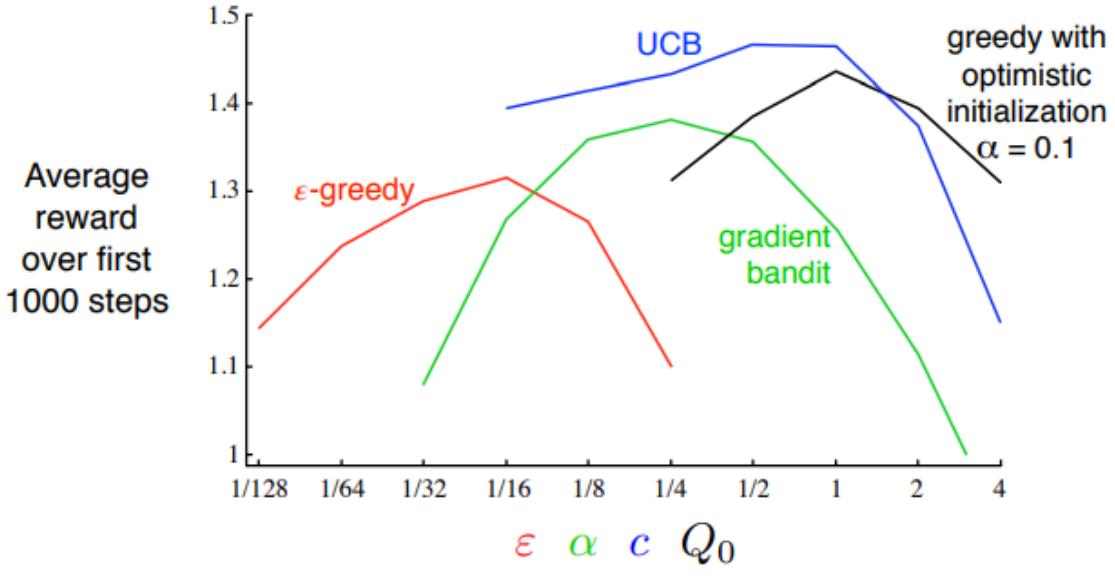

Figura 8: Desempenho médio da recompensa de todos os algoritmos do bandido k-armado após mil passos

No geral, neste problema, o UCB parece apresentar o melhor desempenho.

Todos esses métodos são úteis mas não abrangem a solução de um problema completo de aprendizado por reforço, mas são a base que precisamos aprender. O autor termina a seção falando sobre outra forma de abordar o balanceamento de exploitation e exploration usando um método chamado “Gittins index” que usa distribuições a priori e posteriores, além de priores conjugadas.

3. Processos de Decisão de Markov Finitos

Nesse novo capítulo introduziremos Processos de Decisão de Markov Finitos, ou MDPs, que envolve uma clássica formalização de tomada de decisão, onde ações não influenciam somente recompensas imediatas, mas também situações subsequentes, estados e recompensas futuras.

3.1 A interface do agente-ambiente

MDPs são uma formulação direta do problema de aprender de interações para alcançar um objetivo. O tomador de decisão é chamado de agente. A coisa que interage com ele, compreendendo tudo de fora do agente, é chamado de ambiente. Eles interagem

continuamente, com o agente selecionando ações e o ambiente reagindo a essas ações e retornando outras situações e retornando recompensas.

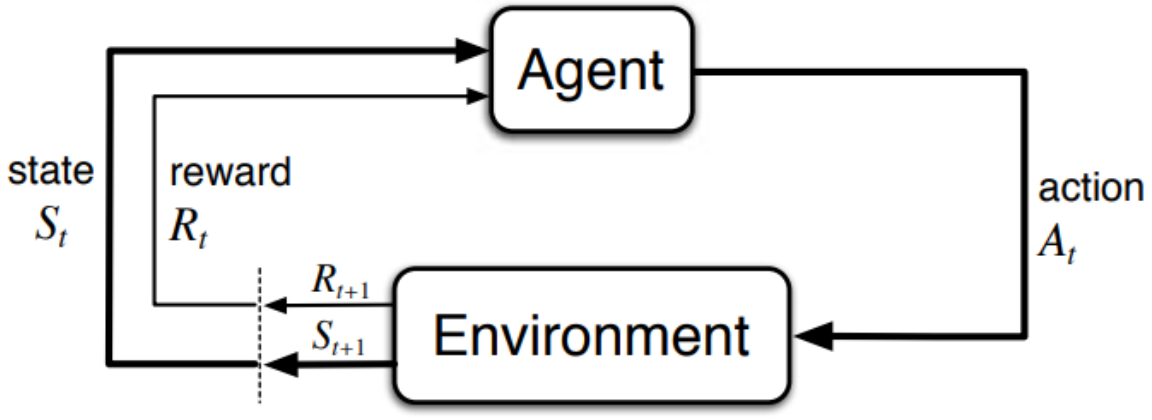

Figura 9: O agente e o ambiente interagindo em um Processo de Decisão de Markov.

Mais especificamente, o agente e o ambiente interagem a cada sequência de passos discreta, $t = 1, 2, ...$(restringindo a uma quantidade discreta para entendermos melhor). A cada passo t, o agente recebe alguma representação sobre o estado do ambiente, $S_t \in \mathcal{S}$, e com base nisso seleciona uma ação, $A_t \in \mathcal{A}(s)$. Após a ação selecionada, o agente recebe uma recompensa numérica $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$ e acha para si mesmo um novo estado, S_{t+1}. O agente e o ambiente numa MDP faz uma trajetória desse tipo:

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, ... \tag{13}$$

Em um MDP finito, o conjunto de ações e recompensas têm um número finito de elementos. Nesse caso, conseguimos definir variáveis aleatórias discretas R_t e S_t dependendo apenas do estado e ação anterior. Ou seja, para $s' \in \mathcal{S} \text{ and } r \in \mathcal{R}$, existe uma probabilidade desses valores ocorrerem no tempo t, dado valores do estado e ação:

$$p(s', r \mid s, a) \doteq \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\} \tag{14}$$

para todo $s', s \in \mathcal{S}, r \in \mathcal{R}$ e $a \in \mathcal{A}(s)$. Essa probabilidade significa basicamente: depois de estar no estado s e tomar a ação a, o ambiente responda com a recompensa r e transite para o estado s'. Lembre-se que p especifica uma distribuição de probabilidade para cada escolha de s e a, ou seja

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) = 1, \quad \text{para todo } s \in \mathcal{S}, a \in \mathcal{A}(s) \tag{15}$$

Em MDPs, a probabilidade caracteriza completamente a dinâmica do ambiente. Ou seja, a probabilidade de cada valor possível para R_t e S_t depende apenas no estado e ação imediatamente anterior S_{t-1} e R_{t-1}. Note então que o estado deve incluir todas as informações relevantes sobre as interações passadas entre o agente e o ambiente que possam influenciar no futuro. Dizemos que, se isso acontece, o estado tem a propriedade de Markov, e assumiremos essa propriedade durante o resumo.

Podemos calcular outras quantidades válidas baseado nessa probabilidade geral, como a probabilidade de transição de estado(como uma função de 3 arguentos $p : \mathcal{S}$ x $\mathcal{S}$ x $\mathcal{A} \to [0, 1]$):

$$p(s'|s,a) \doteq \Pr\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in \mathcal{R}} p(s', r \mid s, a) \tag{16}$$

Nós também podemos podemos calcular as recompensas esperadas para pares estado-ação, definido como $r : \mathcal{S}$ x $\mathcal{A} \to \mathbb{R}$:

$$r(s,a) \doteq \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r \mid s, a) \tag{17}$$

E as recompensas esperadas condicionadas também ao próximo estado, como uma função de três argumentos $r : \mathcal{S}$ x $\mathcal{A}$ x $\mathcal{S} \to \mathbb{R}$:

$$r(s,a,s') \doteq \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s'] = \sum_{r \in \mathcal{R}} r \frac{p(s', r \mid s, a)}{p(s' \mid s, a)} \tag{18}$$

O MDP pode ser um pouco abstrato e flexível às vezes, e pode ser aplicado a muitos tipos de maneiras diferentes. Os passos de tempo($t, t+1$, etc.) não precisam corresponder a intervalos fixos de tempo real, eles podem se referir a estágios arbitrários e sucessivos de tomada de decisão e ação.

Por exemplo, as ações podem ser de baixo nível, como controles de voltagem de um braço robótico, ou alto nível, como decidir ou não se deve almoçar. A mesma coisa acontece para os estados, que podem ser de baixo nível, como leituras de sensores, ou podem ser mais abstratos e de alto nível, como descrições simbólicas de elementos numa sala. Ou seja, cada “time step” representa apenas uma unidade lógica de decisão e consequência.

Em geral, seguimos uma regra que diz o seguinte: tudo que não pode ser alterado arbitrariamente pelo agente é considerado parte do ambiente. Não assumimos que tudo no ambiente é desconhecido pelo agente, mas sempre consideramos o cálculo das recompensas como algo externo do agente. Por fim, em alguns casos o agente pode saber tudo sobre como o ambiente funciona e ainda assim enfrentar uma tarefa de aprendizado por reforço difícil, assim como podemos conhecer exatamente as regras de um cubo mágico e ainda assim não conseguir resolvê-lo.

A fronteira entre o agente e o ambiente pode ser localizado de diferentes lugares, para diferentes propósitos. Por exemplo, um agente pode tomar decisões de alto nível que formam parte dos estados enfrentados por um agente de nível mais baixo, que implementa as decisões de alto nível.

Um quadro de MDP é uma abstração considerável do problema de aprender com o objetivo de atingir metas a partir da interação. Esse modelo propõe que, qualquer problema de aprendizado de comportamento orientado a objetivos pode ser reduzido a três sinais trocados entre um agente e seu ambiente:

- Um sinal para representar as escolhas feitas pelo agente (as ações),
- Um sinal para representar a base sobre a qual as escolhas são feitas (os estados), e
- Um sinal para definir o objetivo do agente (as recompensas).

Exemplo: **Biorreator**

Suponha que o aprendizado por reforço esteja sendo aplicado para determinar, momento a momento, as temperaturas e taxas de agitação de um biorreator (um grande tanque de nutrientes e bactérias usado para produzir substâncias químicas úteis).

As ações, nesse tipo de aplicação, podem ser temperaturas-alvo e taxas de agitação-alvo que são passadas para sistemas de controle de baixo nível que, por sua vez, ativam diretamente elementos de aquecimento e motores para atingir esses valores.

Os estados provavelmente consistem em leituras de sensores (como termopares), possivelmente filtradas e atrasadas, além de entradas simbólicas representando os ingredientes no tanque e o produto químico desejado.

As recompensas podem ser medidas momento a momento da taxa de produção da substância útil pelo biorreator.

Note que aqui cada estado é uma lista (ou vetor) de leituras de sensores e entradas simbólicas, e cada ação é um vetor consistindo de uma temperatura-alvo e uma taxa de agitação. É típico em tarefas de aprendizado por reforço que estados e ações tenham representações estruturadas desse tipo. As recompensas, por outro lado, são sempre números únicos (escalares).

Exemplo: **Robô de Pegar e Colocar (Pick-and-Place)**

Considere o uso de aprendizado por reforço para controlar o movimento do braço de um robô em uma tarefa repetitiva de pegar e colocar objetos. Se quisermos aprender movimentos rápidos e suaves, o agente de aprendizado precisará controlar diretamente os motores e ter informações de baixa latência sobre as posições e velocidades atuais das articulações mecânicas.

As ações, nesse caso, podem ser as tensões elétricas aplicadas a cada motor em cada articulação, e os estados podem ser as leituras mais recentes dos ângulos e velocidades das juntas.

A recompensa pode ser +1 para cada objeto que o robô pegar e colocar com sucesso. Para encorajar movimentos suaves, a cada passo de tempo pode ser dada uma pequena recompensa negativa, em função da “tremedeira” (ou irregularidade) do movimento no momento.

3.2 Metas e recompensas

Em teoria, o objetivo do agente é maximizar o total de recompensas que recebe. Como mostrado em seções anteriores, isso não significa maximizar apenas a recompensa imediata, mas a recompensa acumulada ao longo do tempo.

O autor diz: “Tudo o que entendemos por metas e propósitos pode ser considerado como a maximização do valor esperado da soma cumulativa de um sinal escalar recebido (chamado recompensa.“

Em particular, o sinal de recompensa não deve ser usado para transmitir ao agente conhecimento prévio sobre como alcançar um objetivo. Por exemplo, um agente que joga xadrez deve ser recompensado apenas por vencer o jogo, e não por sub-objetivos intermediários

como capturar peças ou controlar o centro do tabuleiro. Se esses sub-objetivos fossem recompensados separadamente, o agente poderia encontrar uma maneira de atingi-los sem alcançar o verdadeiro objetivo — por exemplo, capturar várias peças, mas ainda assim perder a partida, o que não traria o resultado esperado.

3.3 Retornos e episódios

Como explicado até agora, a meta do agente é maximizar a recompensa cumulativa do agente ao longo da execução. Mas como definir isso formalmente? O retorno esperado, denominado G_t, após o tempo t, pode ser definido no caso simples como:

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + ... + R_T \tag{19}$$

Onde T é o passo final. Essa formulação faz sentido em aplicações nas quais existe uma noção natural de término, ou seja, quando a interação entre o agente e o ambiente se divide em subsequências, que chamamos de episódios — como partidas de um jogo, travessias de um labirinto ou qualquer tipo de interação repetida. Cada episódio termina em um estado especial chamado estado terminal, seguido de um reinício para um estado inicial padrão.

Mesmo que os episódios terminem de formas diferentes (por exemplo, vitória ou derrota em um jogo), o próximo episódio começa independentemente de como o anterior terminou. Assim, podemos considerar que todos os episódios terminam em um mesmo estado terminal, apenas com recompensas diferentes para os diferentes resultados.

Tarefas com episódios desse tipo são chamadas de tarefas episódicas. Em tarefas episódicas, às vezes precisamos distinguir o conjunto de todos os estados não terminais, denotado por $\mathcal{S}$, do conjunto de todos os estados incluindo o terminal, denotado por $\mathcal{S}^+$.O tempo de término T é uma variável aleatória, que normalmente varia de episódio para episódio.

Por outro lado, em muitos casos, a interação entre o agente e o ambiente não se divide naturalmente em episódios identificáveis, mas continua indefinidamente, sem limite de tempo. Por exemplo, isso seria o modo natural de formular uma tarefa de controle de processo contínuo, ou uma aplicação em um robô de longa duração. Chamamos essas de tarefas contínuas (continuing tasks).

Note que no caso de tarefas contínuas teríamos uma soma infinita de termos de valores de recompensas, o que poderia causar $G_t = \infty$. Mas não é exatamente isso que queremos. Assim, adicionamos um novo conceito que precisamos, chamado de desconto, da forma:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{20}$$

onde $0 \leq \gamma \leq 1$, chamado de taxa de desconto. Esse parâmetro faz com que recompensas futuras valham menos, mantendo G_{t+1} finito mesmo em tarefas infinitas e expressa a ideia intuitiva que recompensas imediatas valem mais do que futuras.

Ainda, note que, se $\gamma < 1$, a soma infinita tem um valor finito, desde que a sequência de recompensas seja limitada. Se $\gamma = 0$, o agente é chamado de míope, pois se preocupa

apenas em maximizar recompensas imediatas, seu objetivo então se torna apenas escolher A_t de modo a maximizar R_{t+1}.

Por fim, note que

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + ... \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma_{t+3} + \gamma^2 R_{t+4} + ...) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned} \tag{21}$$

e isso funciona para todos os instantes de tempo $t < T$, mesmo que a terminação ocorra em $t + 1$, se definirmos $G_T = 0$.

Exemplo: Equilíbrio de um Pêndulo (Pole-Balancing)

O objetivo nesta tarefa é aplicar forças a um carrinho que se move ao longo de um trilho, de modo a manter uma haste presa ao carrinho sem cair. Considera-se que ocorre falha quando a haste ultrapassa um certo ângulo limite a partir da vertical ou quando o carrinho sai dos trilhos. Após cada falha, a haste é reposta na posição vertical.

Esta tarefa pode ser tratada como episódica, em que os episódios naturais são as tentativas repetidas de equilibrar a haste. A recompensa neste caso poderia ser +1 a cada passo de tempo em que não ocorre falha, de modo que o retorno em cada instante seria igual ao número de passos até a falha. Nesse caso, conseguir equilibrar a haste para sempre implicaria em um retorno infinito.

Alternativamente, poderíamos tratar o equilíbrio da haste como uma tarefa contínua, usando desconto. Nesse caso, a recompensa seria -1 a cada falha e zero nos outros momentos. O retorno em cada instante seria então relacionado a $-\gamma^K$, onde K é o número de passos de tempo antes da falha.

Em qualquer dos casos, o retorno é maximizado mantendo a haste equilibrada pelo maior tempo possível.

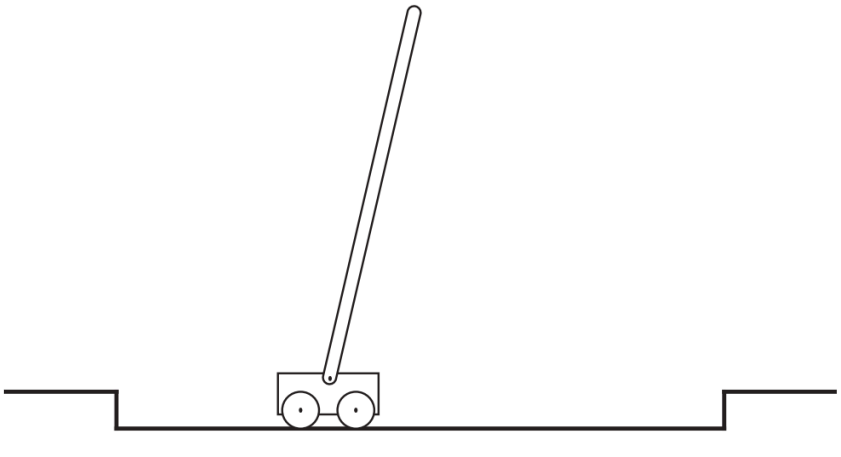

Figura 10: Imagem representativa do exemplo

3.4 Notação Unificada para Tarefas Contínuas e episódicas

Para conseguirmos nos referir não só a Tarefas Contínuas mas também Episódicas sem perda de generalidade e na mesma notação, precisamos nos referir não apenas a S_t, a representação no tempo t, mas sim a $S_{t,i}$, a representação do estado t no episódio i (Análogo para $A_{t,i}$, $R_{t,i}$, etc.).

Agora, precisamos apenas de mais uma convenção de notação única que cubra tanto as tarefas episódicas quanto as contínuas. Os dois casos podem ser unificados se considerarmos que a terminação de um episódio corresponde à entrada de um estado especial absorvente, que faz transição apenas para si mesmo e gera recompensas iguais a zero.

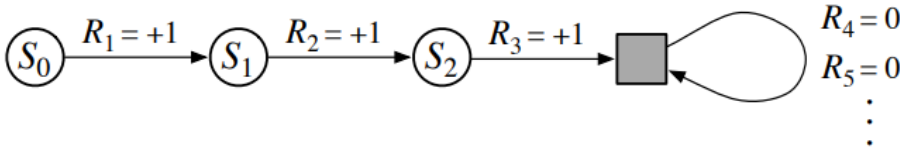

Figura 11: Exemplo de unificação dos casos

No exemplo acima, vemos que o quadrado é exatamente o estado que descrevemos, e que ao somar as recompensas, obtemos o mesmo retorno se somarmos até $T = 3$ ou se somarmos a sequência infinita completa.

Assim, podemos definir o retorno em geral da forma:

$$G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k \tag{22}$$

Incluindo a probabilidade de $T = \infty$ ou $\gamma = 1$.

3.5 Políticas e Funções de valor

Quase todos os algoritmos de Aprendizado por Reforço envolvem a estimação de funções de valor - funções de estados (ou de pares estado-ação) - que estimam o quão bom é para o agente estar em um determinado estado. Essa noção de “quão bom” é definida em termos de recmpensas futuras que podem ser esperadas, ou em termos de retorno esperado.

Assim, as funções de valor são definidas com respeito a modos particulares de agir, chamados políticas. Uma política é um mapeamento de estados para probabilidades de selecionar cada ação possível. Se o agente está seguindo a política π no tempo t, então $\pi(a|s)$ é a probabilidade de que $A_t = a$ e $S_t = s$.

A função de valor de um estado s em uma política π, denotado $v_\pi(s)$ é o valor esperado quando começamos em s e seguindo π depois disso. Para MDPs, conseguimos definir v_π formalmente como:

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right], \text{ para todo } s \in \mathcal{S} \tag{23}$$

onde $\mathbb{E}_\pi[.]$ denota a esperança de uma variável aleatória dado que o agente segue uma política π, e t é qualquer passo. Nós chamamos a função v_π de função de valor de estado para a política π.

Similarmente, podemos definir o valor de tomar a ação a no estado s sobre a política π, denotada $q_\pi(s,a)$, como a esperança partindo de s, tomando a ação a, e depois disso seguir a política π como:

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right] \tag{24}$$

e chamamos q_π de função de valor da ação para a política π.

Como as funções de valor são esperanças, entçao elas podem ser estimadas a partir de experiência, ou seja, se um agente segua a política π e mantém uma média, então essa média convergirá para o valor do estado, à medida que o número de vezes em que o estado é encontrado tende ao infinito (claro uso da Lei dos Grandes Números).

O único problema é que, se houverem muitos estados, pode não ser muito prático manter médias separadas para cada estado individualmente. Nesse caso, o agente deveria representar $v_\pi \text{ e } q_\pi$ como funções parametrizadas(menos parâmetros do que estados) e ajustar os parâmetros de modo a corresponder melhor aos retornos observados.

Uma propriedade legal das funções de valor são que elas mantém relações recursivas, o que sempre traz benefícios computacionalmente. Especificamente, para qualquer política π e qualquer estado s, vale que:

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r \mid s, a)[r + \gamma \mathbb{E}[G_{t+1} \mid S_{t+1} = s']] \\ &= \sum_a \pi(a|s) \sum_{r,s'} p(s', r \mid s, a)[r + \gamma v_\pi(s')], \text{ para todo } s \in \mathcal{S} \end{aligned} \tag{25}$$

de onde vem isso kkkk

Note que na última equação não juntamos duas somas, de todos os valores de s' e de todos os valores r. Note ainda que a expressão final pode ser facilmente lida como um valor esperado. Ela é, na verdade, uma soma sobre todos os valores das três variáveis, $a, s' \text{ e } r$.

A equação (25) é chamada de Equação de Bellman para v_π. Pense em olhar para frente a partir de um estado até seus possíveis estados sucessores (olhe a imagem).

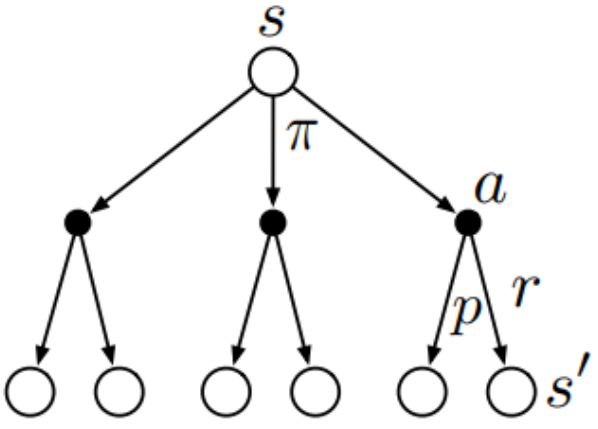

Figura 12: Diagrama intuitivo da Equação de Bellman.

Cada círculo aberto representa um estado, e cada círculo preenchido representa um par estado-ação. Começando do estado s, o agente pode tomar qualquer uma dentre várias ações (de acordo com a política π). A partir dessas ações, o ambiente pode responder com um de vários possíveis estados s', junto com uma recompensa r, dependendo de sua dinâmica, que é dada pela função p.

A equação de Bellman faz a média sobre todas as probabilidades, ponderando cada uma pela chance de ocorrer, e ela afirma que o valor do estado inicial deve ser igual ao valo esperado (descontado) do próximo estado esperado, mais a recompensa esperada ao longo do caminho. Existe apenas uma solução v_π que satisfaz a equação.

Exemplo: GridWorld - Mundo ganancioso

A figura abaixo (esquerda) mostra um GridWorld de um MDP simples e finito. O objetivo do desafio é partir do espaço A' ou B' e chegar em suas respectivas recompensas A e B. Logo, podemos definir que cada estado é cada quadrado (posição da matriz), e que a cada estado temos 4 escolhas de ação: Esquerda, direita, cima, baixo.

Ainda, se o agente chegar a A, ele recebe +10 de recompensa e é teletransportado a A'. O mesmo acontece para B e B' com recompensa +5. Qualquer outra ação que o leve no quadriculado mas não no objetivo recompensa 0. Caso a ação o faça sair do quadriculado, ele permanece no quadriculado anterior, e recebe uma recompensa –1.

Na figura à direita, temos a tabela de valores de estado $v_\pi(s)$ para uma política equiprovável - o agente escolhe qualquer ação com probabilidade $\frac{1}{4}$ (isso significa que $\pi(a|s) = \frac{1}{4}$). A figura mostra o retorno médio esperado se o estado começar naquele estado e seguir uma política aleatória.

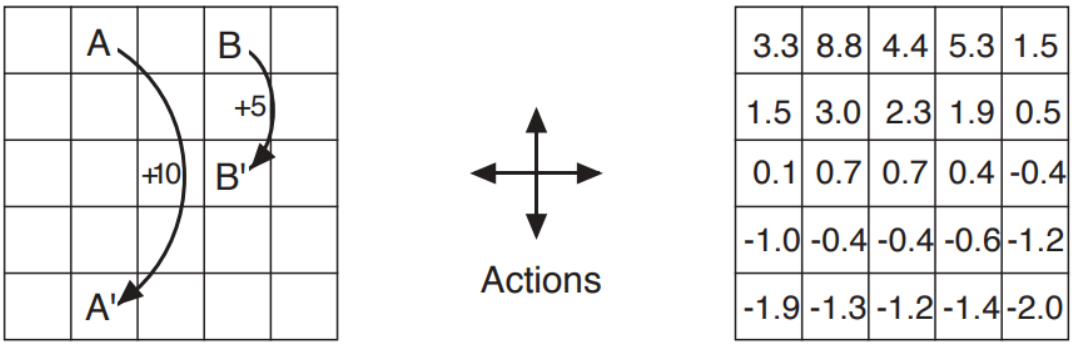

Figura 13: Exemplo do algoritmo GridWorld.

Vamos para outro exemplo um pouco mais difícil:

Exemplo: Golfe

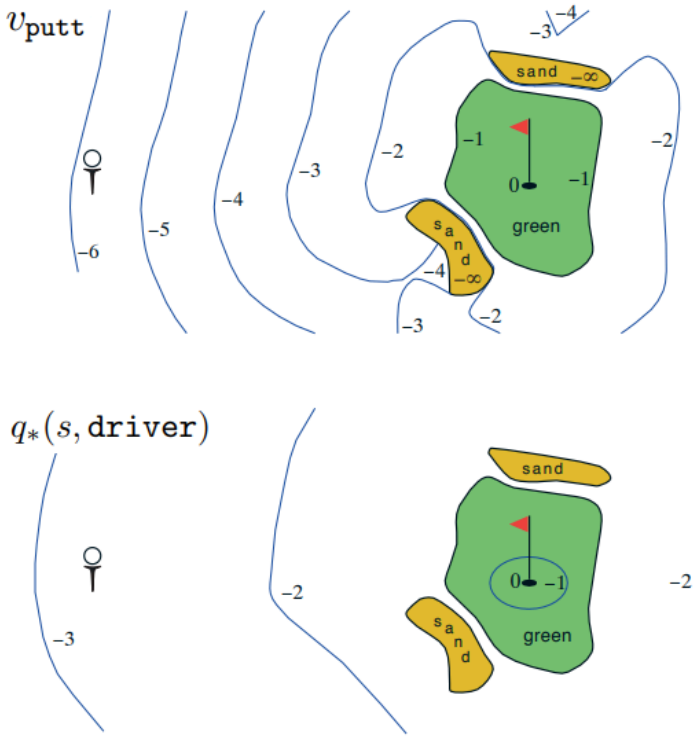

Figura 14: Exemplo do algoritmo para simular um jogo de golfe, imagem superior usa apenas a tacada putter, enquanto a imagem inferior usa $q_*(s, a)$, com a sendo a tacada driver.

Para formular o ato de jogar um buraco de golfe como uma tarefa de RL, contamos uma penalidade de –1 para cada tacada até que acertemos a bola no buraco, recebendo a recompensa 0. O estado é a localização da bola, e as ações são a forma como miramos e escolhemos o taco.

Vamos supor que a mira já está determinada, e consideremos apenas a escolha do taco, que pode ser putter (curta distância) ou driver (longa distância). A imagem superior ao lado mostra uma possível função de valor de estado $v_{\text{putt}}(s)$, para política que sempre usa o putter.

De qualquer ponto do green, supomos que podemos fazer um putt (acertar o buraco); esses estados têm valor –1. Fora do green, não conseguimos chegar ao buraco apenas no putter, então o valor é mais negativo.

Como mostrado na primeira imagem do exemplo, o contorno da imagem mostra a distância do objetivo e, usando o taco putter, notamos que partindo do início, precisariamos de 6 tacadas até chegar ao buraco. Ainda, no segundo exemplo, notamos que usando a tacada driver, precisaríamos apenas de 3 tacadas para chegar ao buraco. Porém, note que temos a areia, que traz a recompensa negativa de $-\infty$.

Como a areia está apenas perto do buraco, e, sabendo que a tacada driver consegue jogar a bola mais longe mas com menos precisão, é esperado que o agente aprenda

a melhor política π_* que combinaria as duas, usando o driver no começo e o putter no fim

3.6 Políticas Ótimas e Funções de Valor Ótimas

Uma política π é ótima se ela for melhor ou igual a qualquer outra política π'. Enunciando melhor, $\pi \geq \pi' \Leftrightarrow v_\pi(s) \geq v_{\pi'}(s)$ para todo $s \in \mathcal{S}$. Sabendo que podem existir mais do que uma, as políticas ótimas são denotadas como π_*. Elas compartilham a melhor função de estado-valor, chamada de função de estado-valor ótima e denotada como:

$$v_*(s) \doteq \max_\pi v_\pi(s), \text{ para todo } s \in \mathcal{S} \tag{26}$$

Ainda, políticas ótimas compartilham as mesmas funções de ação-valor ótimos, definidas como:

$$q_*(s,a) \doteq \max_\pi q_\pi(s,a), \text{ para todo } s \in \mathcal{S} \text{ e } a \in \mathcal{A} \tag{27}$$

Exemplo: Continuação do exemplo de golfe

A imagem de baixo mostra uma possível função de valor-ótima, q_*. Como dito, o driver nos permite bater na bola mais longe, mas com menos precisão. Podemos alcançar o buraco em uma única tacada usando o driver apenas se estivermos bem perto, por isso, o contorno de valor -1 de $q_*(s, \text{driver})$ cobre apenas uma pequena região ao redor da área verde.

Se tivermos duas tacadas, entretanto, podemos chegar ao buraco a partir de locais mais distantes, conforme mostrado pelo contorno de -2. Nesse caso, não precisamos atingir diretamente o buraco com o driver, basta chegar até a área verde, onde então podemos usar o putter.

Da posição inicial (tee), o melhor conjunto de ações é duas tacadas com o driver e uma com o putter, totalizando três tacadas até o buraco.

Note que o valor de um estado sob a política ótima, deve ser igual ao retorno esperado de tomar a melhor ação possível a partir desse estado. Formalmente:

$$\begin{aligned}
v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\
&= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a] \\
&= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \\
&= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_*(s')]
\end{aligned} \tag{28}$$

Essa é a chamada equação de Bellman de otimalidade. O mesmo raciocínio segue para a equação de Bellman de otimalidade para q_*:

$$
\begin{aligned}
q_*(s,a) &= \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \mid S_t = s, A_t = a\right] \\
&= \sum_{s',r} p(s', r \mid s, a)\left[r + \max_{a'} q_*(s', a')\right]
\end{aligned} \tag{29}
$$

Os diagramas de backup são os mesmos usados anteriormente, exceto que os arcos nos pontos de escolha do agente tem um max.

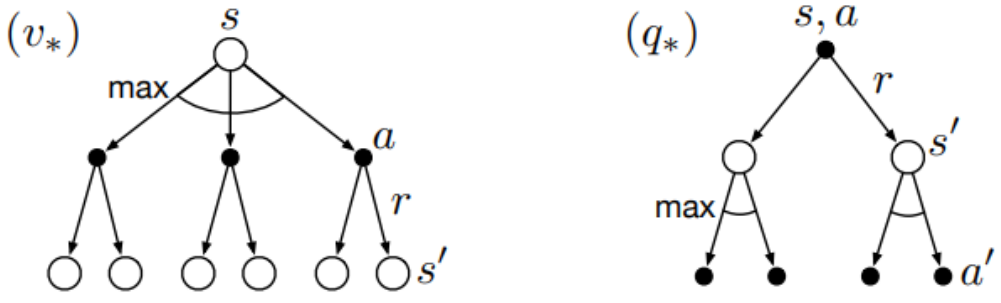

Figura 15: Diagramas anteriores, representando a equação de Bellman de otimalidade para v_π e q_π, respectivamente.

Para MDPs finitos, a equação de otimalidade de Bellman tem uma única solução. Além disso, uma vez que se tenha v_*, é relativamente fácil determinar uma política ótima.

Para cada estado s, haverá uma ou mais ações nas quais o máximo é atingido na equação de otimalidade de Bellman. Qualquer política que atribua probabilidade diferente de zero a essas ações é uma política ótima.

Podemos pensar nisso como uma busca de um passo. Se temos a função de valor ótima v_*, então as ações que parecem melhores após uma busca de um passo já são ações ótimas.

Ter q_* também torna a escolha das ações ótimas ainda mais fácil, pois, para qualquer estado s, o agente pode simplesmente encontrar a ação a que maximiza $q_*(s, a)$. A função de valor-ação q_* efetivamente armazena (faz cache) os resultados de todas as buscas de um passo à frente.

Exemplo: Continuação do exemplo de GridWorld

Suponha que resolvemos a equação de Bellman para v_* para o grid simples introduzido no exemplo anterior. A figura do meio mostra a função ótima do valor, enquanto a função da direita mostra as políticas ótimas correspondentes.

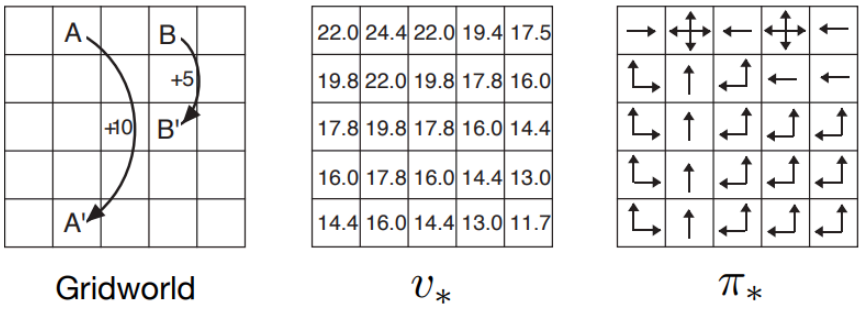

Figura 16: Solução ótima para o problema GridWorld

faltou o exemplo 3.9

3.7 Otimização e aproximação

Um agente que aprende uma política ótima teve um excelente desempenho. Mas, na prática, isso só acontece com alto custo computacional. Mesmo que tenhamos um modelo completo e preciso das dinâmicas do ambiente, geralmente não é possível simplesmente calcular uma política ótima resolvendo a equação de otimalidade de Bellman.

Por exemplo, jogos de tabuleiro como o xadrez representam apenas uma fração minúscula da experiência humana, e mesmo assim grandes computadores especialmente projetados ainda não conseguem calcular as jogadas ótimas. Os principais aspectos que limitam são o poder computacional de tempo e memória.

Em casos tabulares(pequenos e finitos), é possível resolver e armazenas em arrays ou tabelas. Porém, em casos práticos, existem muitos mais estados do que os possíveis de armazenar. Nesses casos, as funções precisam ser aproximadas, usando alguma forma de representação funcional mais compacta e parametrizada.

A natureza online(significa com atualização constante dos dados) do aprendizado por reforço torna possível aproximar políticas ótimas de forma que se dedique mais esforço a aprender boas decisões para estados frequentemente encontrados, à custa de menos esforço para estados raramente encontrados.

3.8 Sumário