Aula 8

Sarsa Semi-gradiente de n-passos (Semi-gradient n-step Sarsa)

Podemos obter uma versão de n-passos do algoritmo Sarsa semi-gradiente para tarefas episódicas de forma bastante direta: basta utilizar o retorno de n-passos como o alvo da atualização na regra que já conhecemos.

Generalizando o Alvo da Atualização

O retorno de n-passos, que vimos no caso tabular, é generalizado para o caso de aproximação de funções da seguinte forma:

$$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \, \hat{q}(S_{t+n}, A_{t+n}, \mathbf{w}_{t+n-1}), \quad \text{para } t+n < T. \tag{1}$$

Onde definimos $G_{t:t+n} \doteq G_t$ (o retorno convencional) se $t+n \geq T$.

Com este novo alvo, a regra de atualização do Sarsa semi-gradiente de n-passos se torna:

$$\mathbf{w}_{t+n} \doteq \mathbf{w}_{t+n-1} + \alpha \big[G_{t:t+n} - \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1}) \big] \, \nabla \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1}). \tag{2}$$

Algoritmo Completo

O pseudocódigo a seguir detalha a implementação do Sarsa semi-gradiente de n-passos para estimar $\hat{q} \approx q_*$ ou $\hat{q} \approx q_\pi$.

Sarsa semi-gradiente de n-passos episódico para estimar $\hat{q}$

Entrada: uma parametrização de função de valor-ação diferenciável $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$.

Entrada: uma política π (se o objetivo for estimar q_π).

Parâmetros: passo $\alpha > 0$, um pequeno $\epsilon > 0$, inteiro positivo n.

Inicialize os pesos da função de valor $\mathbf{w} \in \mathbb{R}^d$ arbitrariamente (por exemplo, $\mathbf{w} = \mathbf{0}$).
Todas as operações de armazenamento (para S_t, A_t, R_t) podem usar o índice $(t \bmod (n+1))$.
Para cada episódio:

1. Inicialize e armazene $S_0 \neq$ terminal.
2. Selecione e armazene uma ação $A_0 \sim \pi(\cdot \mid S_0)$ ou ϵ-greedy em relação a $\hat{q}(S_0, \cdot, \mathbf{w})$.
3. $T \leftarrow \infty$.
4. **Para** $t = 0, 1, 2, \ldots$:
   (a) Se $t < T$:
      i. Execute a ação A_t.
      ii. Observe e armazene R_{t+1} e S_{t+1}.
      iii. Se S_{t+1} é terminal, então $T \leftarrow t + 1$.
      iv. Senão, selecione e armazene $A_{t+1} \sim \pi(\cdot \mid S_{t+1})$ ou ϵ-greedy em relação a $\hat{q}(S_{t+1}, \cdot, \mathbf{w})$.
   (b) $\tau \leftarrow t - n + 1$ (passo cuja estimativa está sendo atualizada).
   (c) Se $\tau \geq 0$:
      i. $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$.
      ii. Se $\tau + n < T$, então $G \leftarrow G + \gamma^n \hat{q}(S_{\tau+n}, A_{\tau+n}, \mathbf{w})$.
      iii. $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[G - \hat{q}(S_\tau, A_\tau, \mathbf{w})\big] \nabla \hat{q}(S_\tau, A_\tau, \mathbf{w})$.
5. **Até que** $\tau = T - 1$.

Desempenho e a Importância de n

Como já vimos em outros contextos, o desempenho do aprendizado costuma ser melhor quando se utiliza um nível intermediário de *bootstrapping*, o que corresponde a um valor de $n > 1$.

Um valor $n = 1$ equivale ao Sarsa de um passo, enquanto um valor muito grande de n faz com que o algoritmo se aproxime de um método Monte Carlo. A pesquisa empírica mostra que valores intermediários frequentemente levam a convergência mais rápida e melhor desempenho assintótico.

Mountain Car
Steps per episode
log scale
averaged over 100 runs

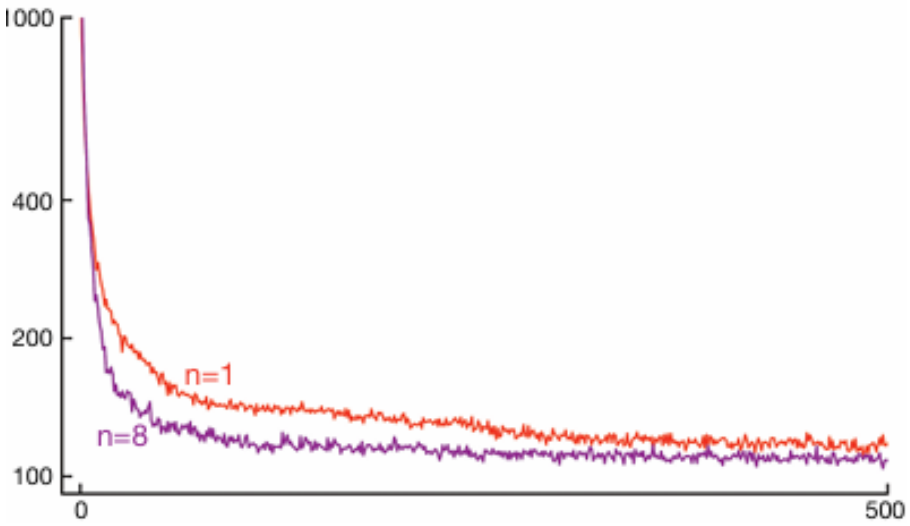

Figura 1: Desempenho do Sarsa semi-gradiente de 1 passo vs. 8 passos na tarefa *Mountain Car*. Foram utilizados tamanhos de passo: $\alpha = 0.5/8$ para $n = 1$ e $\alpha = 0.3/8$ para $n = 8$.

Recompensa Média

Vamos introduzir agora uma nova forma de definir o nosso retorno: a Recompensa Média. Até agora nos introduzimos duas formas de formular o Retorno (G_t):

1. **Forma Episódica**

   Nessa forma o o Retorno é definido simplesmente como a soma das recompensas até o final do episódio, ou seja, $G_t = \sum_{i=0}^{n} R_{t+i}$.

2. **Forma Descontada**

   Já nessa forma o cada recompensa é multiplicado por um fator de desconto γ^i, onde cada recompensa mais distante do momento atual tem sua importância reduzida, de forma que: $G_t = \sum_{i=0}^{n} \gamma^i R_{t+i}$.

O que estamos propondo agora é uma nova forma de formular o G_t, baseado na recompensa média que a nossa política está encontrando. Fazemos isso por alguns motivos:

- Ao observar como está a Recompensa Média de uma política (π), em qualquer estado de tempo, podemos ter uma ideia muito mais clara sobre a performance de π no geral, servindo como métrica para comparar políticas.

- Essa forma pode ser aplicada tanto para casos contínuos quanto para casos episódicos, o que não é o caso da Forma Episódica.
- Veremos na próxima seção que, para métodos de aproximação de funções, o caso descontado perde algumas propriedades teóricas que eram relevantes.
- Não será preciso ficar escolhendo valores de γ, o que facilita na hora de treinar os agentes.
- Não tem o problema da Forma Descontada de gerar políticas que priorizam demais ações de curto prazo, pois todas as recompensas mudam a média da mesma foma.

Dessa forma, definimos o Retorno como:

$$G_t = R_{t+1} - r(\pi) + R_{t+2} - r(\pi) + R_{t+1} - r(\pi) + ... \tag{3}$$

Onde $r(\pi)$ é a Recompensa Média de π até o momento e cada recompensa é subtraída pela média, o que não permite que o Retorno seja infinito.

Dessa forma, para vermos um exemplo, podemos escrever o Sarsa como:

Algorithm 1 Semi-Gradinet Sarsa para estimar q^*

Entrada: Uma função diferenciável $\hat{q}$ para aproximar q_π
Parâmetros de incremento $\alpha, \beta > 0$
Vetor de pesos w, inicializado arbitrariamente
Valor inicial para a aproximação da Recompensa Média de π: $\bar{R}$
Inicializa em um estado S com uma ação A
repeat
  Executa A e observa R e S'
  Escolhe A' de acordo com $\hat{q}(S', \cdot, w)$ (pode ser ϵ-greedy)
  $\delta \leftarrow (R - \bar{R}) - (\hat{q}(S, A, \mathbf{w}) - \hat{q}(S', A', \mathbf{w}))$
  $\bar{R} \leftarrow \bar{R} + \beta\delta$
  $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\nabla\hat{q}(S, A, \mathbf{w})$
  $S \leftarrow S'$
  $A \leftarrow A'$
until convergência =0

Onde δ é o **erro de Diferença Temporal (TD)**, que podemos interpretar como um **"sinal de surpresa"**. Ele mede o quão diferente a realidade foi da nossa expectativa, nos dizendo se a transição $(S, A) \rightarrow (S', R)$ foi melhor ou pior do que o previsto.

Este sinal de surpresa, δ, é a peça central do aprendizado e é usado em dois lugares cruciais:

- **Ajuste da Recompensa Média Estimada ($\bar{R}$)**

O primeiro uso de δ é para refinar nossa estimativa da recompensa média, $\bar{R}$. A lógica é a seguinte:

- Se δ é **positivo**, significa que a recompensa R que recebemos, somada ao valor futuro de S', foi **maior do que esperávamos**. Isso sugere que nossa estimativa atual de $\bar{R}$ talvez esteja muito baixa.

- Se δ é **negativo**, a experiência foi **pior do que o esperado**, sugerindo que nossa estimativa $\bar{R}$ pode estar muito alta.

Por isso, usamos a regra de atualização $\bar{R} \leftarrow \bar{R} + \beta\delta$. Ela corrige nossa estimativa na direção da surpresa: se a surpresa foi positiva, aumentamos $\bar{R}$; se foi negativa, diminuímos. O parâmetro β controla o tamanho desse passo de correção.

É aí que percebemos que não calculamos $\bar{R}$ por uma **média simples**, pois essa abordagem incremental é muito mais poderosa e adequada para o aprendizado contínuo. As vantagens são claras:

1. **Adaptabilidade à Política em Evolução:** O agente está constantemente aprendendo e melhorando sua política. Uma média simples daria o mesmo peso para recompensas recebidas no início (com uma política ruim) e para as recebidas agora (com uma política melhor). A atualização incremental, que funciona como uma **média móvel ponderada**, dá mais importância às experiências recentes, permitindo que $\bar{R}$ **se adapte e convirja para a verdadeira recompensa média da política *atual***.

2. **Eficiência Computacional e de Memória:** Para calcular uma média simples, como $\frac{1}{t}\sum_{i=1}^{t} R_i$, precisaríamos armazenar todas as recompensas passadas ou, no mínimo, a soma total e a contagem de passos, o que se torna inviável em tarefas que rodam para sempre. A abordagem incremental só precisa do valor atual de $\bar{R}$ e do último erro δ, sendo extremamente leve e eficiente.

3. **Capacidade de Lidar com Ambientes Não-Estacionários:** Se as regras do ambiente mudarem com o tempo, a atualização incremental permite que $\bar{R}$ se adapte a essa nova realidade, enquanto uma média simples ficaria "presa" ao passado, contaminada por dados de uma dinâmica que não existe mais.

- **O Segundo Papel de δ: Atualização dos Pesos da Função de Valor (w)**

O segundo, e principal, papel do erro δ é guiar o aprendizado da própria função de valor, ou seja, **ajustar os pesos** $\mathbf{w}$ para que as predições futuras sejam mais precisas, o que já foi analisado anteriormente no curso.

Deprecar o cenário *descontado* em problemas contínuos

Em tarefas *contínuas* (sem inícios/fins naturais), a formulação tradicional de controle com desconto (fator $\gamma \in [0, 1)$) é útil no caso tabular, mas torna-se questionável quando usamos *aproximação de função*. A intuição é que, sem episódios bem definidos e com estados pouco distinguíveis (p. ex.,

apenas vetores de atributos semelhantes), o desempenho da política precisa ser avaliado diretamente a partir da sequência de recompensas no longo prazo, o que nos leva naturalmente ao critério de **recompensa média** (average reward).

Recompensa média vs. retorno descontado (em média) Considere a recompensa média de uma política π:

$$r(\pi) \;=\; \lim_{T\to\infty} \frac{1}{T}\,\mathbb{E}_\pi\left[\sum_{t=0}^{T-1} R_{t+1}\right].$$

Agora, se calcularmos o retorno descontado $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$ em *cada* passo de tempo t e depois tomarmos a média de G_t ao longo do tempo, em problemas contínuos essa média é simplesmente proporcional a $r(\pi)$:

$$\overline{G} \;=\; \frac{r(\pi)}{1-\gamma}.$$

Consequência importante: **o ordenamento de políticas induzido pela média do retorno descontado coincide com o ordenamento pela recompensa média**. Logo, o valor de γ *não altera o problema de controle* (apenas reescala o objetivo).

Argumento de simetria (intuição) Em um processo estacionário e contínuo, cada recompensa R_t aparece uma vez "sem desconto" em algum retorno, uma vez com desconto γ em outro, e assim por diante. Somando esses pesos ao longo de todos os retornos, cada recompensa recebe o fator $1+\gamma+\gamma^2+\cdots = 1/(1-\gamma)$. Como todas as recompensas são ponderadas do mesmo modo, a média dos retornos descontados sobre o tempo torna-se $r(\pi)/(1-\gamma)$, o que preserva o mesmo ordenamento de políticas que o critério de recompensa média.

Formulação por distribuição estacionária on-policy Outra forma de ver o mesmo resultado é definir o objetivo descontado como a soma, sobre a distribuição estacionária μ^π, dos valores descontados v_γ^π:

$$J(\pi) \;=\; \sum_s \mu^\pi(s)\, v_\gamma^\pi(s).$$

Usando a equação de Bellman e identidades padrão para processos ergódicos, obtém-se

$$J(\pi) \;=\; \frac{r(\pi)}{1-\gamma},$$

o que mostra formalmente que **o objetivo descontado agregado sobre μ^π ordena políticas exatamente como o objetivo de recompensa média**.

Implicações para controle com aproximação de função Se, ao otimizarmos controle, de fato maximizássemos o valor descontado agregado sobre a distribuição on-policy, estaríamos (na prática) maximizando recompensa média. Na realidade, porém, *algoritmos descontados com aproximação de função* não otimizam exatamente esse objetivo agregado e, portanto, **não há garantia** de que otimizem recompensa média.

Mais profundamente, com aproximação de função **perdemos o teorema de melhoria de política**: aumentar o valor descontado em um estado não garante melhoria global da política. Esse "buraco" teórico aparece tanto no cenário descontado quanto nos objetivos episódico-total e de recompensa média. Uma classe alternativa com garantias é a de *métodos de gradiente de política*, que contam com o *policy-gradient theorem* como análogo de garantia local de melhoria.

Prática recomendada (resumo)

- Em problemas **contínuos**, prefira formular o objetivo em termos de **recompensa média**; encare γ como *parâmetro do método* (horizonte de bootstrapping/estabilidade), não como parâmetro do problema.
- Em problemas **episódicos** (com términos naturais), o desconto continua sendo uma modelagem conveniente.
- Ao avaliar políticas em contínuo, reporte métricas de longo prazo coerentes com média (p. ex., diferencial de valor) e seja cauteloso ao interpretar melhorias locais no valor descontado.

Métodos de Gradiente de Política

Até agora, a maioria dos métodos que vimos tem sido baseada em **valores de ação** (action-value methods); eles aprendiam os valores das ações e, em seguida, selecionavam as ações com base nesses valores estimados. As políticas desses métodos não existiriam sem as estimativas de valores de ação.

Nesta seção, exploramos métodos que aprendem uma **política parametrizada** que pode selecionar ações sem consultar uma função de valor. Uma função de valor ainda pode ser usada para aprender o parâmetro da política, mas não é estritamente necessária para a seleção de ações. Usamos a notação $\theta \in \mathbb{R}^{d'}$ para o vetor de parâmetros da política.

Dessa forma, escrevemos:

$$\pi(a|s,\theta) = \Pr\{A_t = a \mid S_t = s, \theta_t = \theta\}$$

para a probabilidade de a ação a ser tomada no tempo t, dado que o ambiente está no estado s no tempo t com o parâmetro θ. Se um método também utiliza uma função de valor aprendida, o vetor de pesos da função de valor é denotado por $w \in \mathbb{R}^d$, como de costume, em $\hat{v}(s,w)$.

Os métodos de gradiente de política baseiam-se no **gradiente de alguma medida de desempenho escalar** $J(\theta)$ em relação ao parâmetro da política. Esses métodos procuram maximizar o desempenho, então suas atualizações aproximam o **stochastic gradient ascent** em J:

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t) \quad (1)$$

onde $\nabla J(\theta_t) \in \mathbb{R}^{d'}$ é uma estimativa estocástica cuja expectativa se aproxima do gradiente da medida de desempenho em relação ao seu argumento θ_t.

Todos os métodos que seguem este esquema geral são chamados de **métodos de gradiente de política** (policy gradient methods), independentemente de também aprenderem ou não uma função de valor aproximada. Métodos que aprendem aproximações tanto para a política quanto para as funções de valor são frequentemente chamados de **métodos ator–crítico** (actor–critic methods), onde *'ator'* é uma referência à política aprendida e *'crítico'* refere-se à função de valor aprendida, geralmente uma função de valor de estado.

Inicialmente, tratamos o caso **episódico**, no qual o desempenho é definido como o valor do estado inicial sob a política parametrizada, antes de considerar o caso **contínuo**, no qual o desempenho é definido como a taxa de recompensa média.

Aproximação de Política e Suas Vantagens

Nos métodos de gradiente de política, a política pode ser parametrizada de qualquer forma, desde que $\pi(a|s,\theta)$ seja **diferenciável** em relação aos seus parâmetros. Na prática, para garantir a exploração, geralmente exigimos que a política nunca se torne determinística (ou seja, $\pi(a|s,\theta) \in (0,1)$, para todo s, a, θ).

Se o espaço de ações for discreto e não muito grande, um tipo de parametrização comum é formar **preferências numéricas parametrizadas** $h(s,a,\theta) \in \mathbb{R}$ para cada par estado–ação. As ações com as maiores preferências em cada estado recebem as maiores probabilidades de seleção, por exemplo, de acordo com uma distribuição **soft-max exponencial**:

$$\pi(a|s,\theta) \doteq \frac{e^{h(s,a,\theta)}}{\sum_b e^{h(s,b,\theta)}} \quad (2)$$

onde $e \approx 2.71828$ é a base do logaritmo natural. Chamamos este tipo de parametrização de política de **soft-max em preferências de ação**.

As preferências de ação podem ser parametrizadas arbitrariamente. Por exemplo, elas podem ser calculadas por uma rede neural artificial profunda, onde θ é o vetor de todos os pesos de conexão. Alternativamente, as preferências podem ser simplesmente **lineares em *features***:

$$h(s,a,\theta) = \theta^\top x(s,a), \quad (3)$$

onde $x(s,a) \in \mathbb{R}^{d'}$ são vetores de *features*.

Vantagens sobre Métodos Baseados em Valor

- **Políticas Quase Determinísticas:** A política aproximada pode se aproximar de uma política determinística, o que não é possível com a seleção de ações ε-gananciosa (epsilon-greedy) sobre valores de ação, que sempre mantém uma probabilidade ε de selecionar uma ação aleatória.
- **Políticas Estocásticas Otimais:** Permite a seleção de ações com probabilidades arbitrárias. Em problemas com aproximação de função significativa ou informações imperfeitas (como jogos

de cartas), a melhor política aproximada pode ser **estocástica**, algo que os métodos baseados em valor não encontram de forma natural.

- **Função de Política Mais Simples:** Para alguns problemas, a política pode ser uma função mais simples de aproximar do que a função de valor de ação, levando a um aprendizado mais rápido e uma política assintótica superior.
- **Injeção de Conhecimento Prévio:** A escolha da parametrização da política é uma excelente maneira de **injetar conhecimento prévio** sobre a forma desejada da política no sistema de aprendizado por reforço.

Policy Gradient Theorem

Agora que já estabelecemos as vantagens de estimar uma política $\pi_\theta(a|s,\theta)$ ao invés de encontrar as funções-valor $v_\pi(s)$/ ação-valor $q_\pi(s,a)$, é necessário discutir como podemos de fato aprimorar nossa política em nosso modelo. De forma geral, o formato que iremos seguir será o de Gradiente-Ascendente, ou seja, mudaremos nossa política na direção do gradiente de uma função optimizadora (oposto da loss). Para casos contínuos, definimos ela como:

$$J(\theta) = \sum_{s\in S} d^\pi(s) v^\pi(s) = \sum_{s\in S} d^\pi \sum_{a\in A} \pi_\theta(a|s) Q^\pi(s,a) \tag{4}$$

Vamos destrinchar essa equação:

- distribuição estacionária da cadeia de Markov ($d^\pi(s)$): diz-nos quão provável é estar em um determinado estado na cadeia, seguindo π; matematicamente: $d^\pi(s) = \lim_{t\to\infty} P(s_t = s|s_0, \pi_\theta)$
- função valor ($v^\pi(s)$): lembrem da 2° aula; é o valor esperado para nosso retorno dado um estado

Basicamente, o que queremos otimizar aqui é o nosso Retorno médio esperado (lembre da definição de $V(s)$) para todos os estados, com mais ênfase nos estados mais frequentes. É uma otimização bastante razoável.

Há um problema bem evidente no cálculo de $\nabla J(\theta)$. Utilizando a definição acima, calcular esse gradiente nos leva à fórmula:

$$\nabla J(\theta) = \nabla \big[\sum_{s\in\mathcal{S}} d^\pi(s) \sum_{a\in\mathcal{A}} Q^\pi(s,a)\, \pi_\theta(a|s) \big] \tag{5}$$

Trazendo o gradiente para dentro do 1° somatório, teríamos que calcular $\nabla d^\pi(s)$; o problema é que não há maneira de fazê-lo, já que essa mudança depende de forma complexa do ambiente do agente. Porém, o que o Policy-Gradient-Theorem afirma (não provaremos aqui) é que isso não é necessário: basta usar como update o seguinte termo proporcional a esse gradiente ($\propto$ = proporcionalidade):

$$\nabla J(\theta) \propto \sum_{s\in\mathcal{S}} d^\pi(s) \sum_{a\in\mathcal{A}} Q^\pi(s,a)\, \nabla \pi_\theta(a|s) \tag{6}$$

Sabemos ao menos estimar com base empírica cada termo que aparece na equação acima, com tudo que vimos até agora no curso. Os algoritmos que seguem o PGT diferem principalmente (apesar de não somente) na forma de calcular $\nabla\pi_\theta(a|s)$, mas todos seguem essa norma geral acima. Veremos a seguir detalhadamente um deles.

REINFORCE: Monte Carlo Policy Gradient

Vamos partir da proporcionalidade que tinhamos antes. Repare que o primeiro somatório, para todos os estados com pesos maiores para estados mais frequentes, é justamente a definição do valor esperado. Portanto:

$$\nabla J(\theta) \propto \sum_s d^\pi(s) \sum_a q_\pi(s,a)\, \nabla\pi(a|s,\theta) \tag{7}$$

$$= \mathbb{E}_\pi \left[\sum_a q_\pi(S_t,a)\, \nabla\pi(a|S_t,\theta) \right] \tag{8}$$

Agora, seria bom poder modificar a equação acima de modo que pudéssemos estimar as funções ação-valor com base em nossa experiência, não? Observe a seguinte formulação:

$$\nabla J(\theta) = \mathbb{E}_\pi \left[\sum_a \pi(a|S_t,\theta)\, q_\pi(S_t,a)\, \frac{\nabla_\theta \pi(a|S_t,\theta)}{\pi(a|S_t,\theta)} \right] \tag{9}$$

$$= \mathbb{E}_\pi \left[q_\pi(S_t,A_t)\, \frac{\nabla_\theta \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)} \right] \tag{10}$$

$$= \mathbb{E}_\pi \left[G_t\, \frac{\nabla \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)} \right]. \tag{11}$$

Perceba que incorporamos à fórmula a probabilidade de uma ação ser tomada dado um estado e uma parametrização específica ($\nabla_\theta\pi(a|S_t,\theta)$); desse modo, podemos tirar o somatório e incorporá-lo ao valor esperado. Note também a capitalização das nossas variáveis de estado e ação: isso indica que agora estamos considerando amostragens, obtidas através de trajetórias experimentais. A substituição de $q_\pi(S_t,A_t)$ por G_t vem da aplicação do valor esperado.

Logo, o que temos em mãos é um update que depende do nosso retorno; logo, desenvolvemos um algoritmo Monte Carlo. O algoritmo está descrito abaixo, com uma pequena abreviação ($\nabla ln(x) = \frac{\nabla x}{x}$):

Algorithm 2 REINFORCE, um método de Gradiente de Política Monte Carlo (episódico)

Entrada: uma parametrização de política diferenciável $\pi(a|s, \boldsymbol{\theta})$
Inicialize o parâmetro da política $\boldsymbol{\theta} \in \mathbb{R}^{d'}$
repeat
  Gere um episódio $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, seguindo $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  for cada passo do episódio $t = 0, \ldots, T-1$ **do**
    $G \leftarrow$ retorno a partir do passo t
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\gamma^t G \nabla_{\boldsymbol{\theta}} \ln \pi(A_t|S_t, \boldsymbol{\theta})$
  end for
until convergência =0

Métodos Actor-Critic

Como um método de Monte Carlo, o REINFORCE utiliza o retorno completo do episódio ($G_t = R_{t+1} + \gamma R_{t+2} + ...$) para atualizar a política. O problema é que esse retorno acumulado pode variar drasticamente de um episódio para outro, mesmo com pequenas mudanças na política. Essa alta variância torna o processo de aprendizagem instável e lento, exigindo muitas amostras para convergir. Além disso, a necessidade de aguardar o final de um episódio para calcular o retorno G_t impede que o algoritmo aprenda de forma online e incremental, ou seja, a cada passo. Isso o torna inadequado para tarefas contínuas ou que exigem aprendizado em tempo real.

Nós já vimos esses problemas antes, e, assim como os Métodos TD foram introduzidos para resolver isso na primeira partedo curso, apresentamos agora os Métodos Actor-Critic, que combinam as vantagens dos métodos de gradiente de política com a eficiência e a menor variância dos Métodos TD, por meio do *bootstrapping*.

Os métodos Actor-Critic são uma arquitetura de aprendizado por reforço que divide o agente em dois componentes distintos, que aprendem simultaneamente:

- **O Ator (Actor):** É responsável por controlar como o agente se comporta. Ele corresponde à política parametrizada $\pi(a|s, \theta)$, que decide qual ação tomar em um determinado estado. O objetivo do Ator é aprender a melhor política possível.
- **O Crítico (Critic):** É responsável por avaliar a ação tomada pelo Ator. Ele aprende uma função de valor $\hat{v}(s, w)$ que estima o quão boa é uma ação a partir de um estado. Em vez de esperar até o final do episódio, o Crítico fornece um feedback de aprendizado a cada passo.

A interação funciona da seguinte forma: o Ator executa uma ação A_t no estado S_t. O Crítico observa a recompensa R_{t+1} e o próximo estado S_{t+1} e calcula o TD error:

$$\delta_t = R_{t+1} - \bar{R}_{t+1} + \hat{v}(S_{t+1}, w_t) - \hat{v}(S_t, w_t) \tag{12}$$

Esse sinal δ_t "critica"a ação do Ator. Se δ_t for positivo, significa que a ação levou a um resultado melhor do que o esperado, e o Ator ajusta seus parâmetros θ para aumentar a probabilidade de tomar

essa ação no futuro. Se for negativo, a ação foi pior que o esperado, e o Ator a torna menos provável. Assim, o Crítico guia o aprendizado do Ator de forma muito mais eficiente do que o retorno completo G_t.

A combinação desses dois componentes oferece vantagens importantes:

- **Redução de Variância**: A principal vantagem é a substituição do retorno de Monte Carlo (G_t) pelo erro TD (δ_t) para atualizar a política. O erro TD tem uma variância muito menor, pois se baseia em uma recompensa real (R_{t+1}) e uma estimativa ($\hat{v}(S_{t+1}, \mathbf{w})$), resultando em uma convergência mais estável e rápida.
- **Aprendizagem Online e Incremental**: Como as atualizações são baseadas em transições de um único passo, o agente pode aprender a cada interação com o ambiente. Ele não precisa esperar o fim de um episódio, o que o torna ideal para tarefas contínuas e aplicações em tempo real.
- **Eficiência**: Ao utilizar o *bootstrapping* (atualizar estimativas com base em outras estimativas), os métodos Actor-Critic são mais eficientes em termos de dados do que os métodos de Monte Carlo puros, acelerando o aprendizado.

Problemas não episódicos

Até agora, a maioria dos algoritmos de gradiente de política que vimos foi desenvolvida para o **caso episódico**, onde as tarefas têm um início e um fim bem definidos. Nestes casos, a medida de performance, $J(\theta)$, é naturalmente definida como o valor do estado inicial, $v_{\pi_\theta}(s_0)$.

No entanto, muitos problemas do mundo real não se encaixam neste modelo. Por exemplo, o controle de um robô numa linha de montagem ou a gestão contínua de uma rede elétrica. Estas são **tarefas contínuas**, que não têm um estado terminal definido.

O Desafio dos Problemas Contínuos

O principal desafio é que o conceito de "retorno total"($G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$) pode divergir para o infinito, especialmente se não houver desconto ($\gamma = 1$). Portanto, a nossa medida de performance e as nossas definições de valor precisam de ser adaptadas para o caso contínuo.

Solução: Novas Definições de Performance e Retorno

Para generalizar os métodos de gradiente de política para o caso contínuo, precisamos redefinir o nosso objetivo.

A Taxa Média de Recompensa como Medida de Performance

Em vez de maximizar um retorno total, o nosso novo objetivo é maximizar a **taxa média de recompensa**, denotada por $r(\pi)$. Esta é a quantidade de recompensa que esperamos receber, em média, a cada passo de tempo, assumindo que seguimos a política π para sempre.

A nossa nova medida de performance, $J(\theta)$, torna-se a própria taxa média de recompensa:

$$J(\theta) \doteq r(\pi_\theta) = \lim_{t\to\infty} \mathbb{E}[R_t | A_{0:t-1} \sim \pi_\theta] = \sum_s \mu(s) \sum_a \pi(a|s,\theta) \sum_{s',r} p(s',r|s,a) r \tag{13}$$

onde $\mu(s)$ é a distribuição de estados estacionária sob a política π_θ. *Fonte: Equação (13.15), página 333.*

O Retorno Diferencial

Se o nosso objetivo agora é a recompensa média, então as ações devem ser avaliadas com base em se elas nos levam a recompensas **melhores ou piores do que a média**. Isto leva a uma nova definição de retorno, o **retorno diferencial** (G_t):

$$G_t \doteq (R_{t+1} - r(\pi)) + (R_{t+2} - r(\pi)) + (R_{t+3} - r(\pi)) + \dots \tag{14}$$

Fonte: Equação (13.17), página 334.

Intuição: O retorno G_t agora mede o "quão melhor"(ou pior) o futuro será em comparação com a média. Uma ação que leva a um G_t positivo é considerada uma boa ação. As funções de valor $v_\pi(s)$ e $q_\pi(s,a)$ são, portanto, definidas em relação a este novo retorno diferencial.

O Teorema do Gradiente de Política e o Algoritmo Actor-Critic

A Validade do Teorema

A grande vantagem teórica é que, com estas novas definições, o **Teorema do Gradiente de Política continua a ser válido** para o caso contínuo. A expressão para o gradiente da performance, $\nabla J(\theta)$, mantém a sua forma fundamental. A prova, embora ligeiramente diferente, chega à mesma conclusão.

O Algoritmo Actor-Critic para Tarefas Contínuas

A adaptação do algoritmo Actor-Critic para o caso contínuo é direta. A principal mudança ocorre no cálculo do **erro TD (δ_t)**, que agora deve usar a recompensa diferencial.

Como não conhecemos o verdadeiro $r(\pi)$, usamos uma estimativa $\bar{R}$ que é aprendida ao mesmo tempo:

$$\delta_t \leftarrow (R_{t+1} - \bar{R}) + \hat{v}(S_{t+1}, w) - \hat{v}(S_t, w) \tag{15}$$

Onde $\bar{R}$ é atualizado a cada passo, por exemplo, com $\bar{R} \leftarrow \bar{R} + \alpha^{\bar{R}}\delta_t$.

Analogia e Conclusão

Para solidificar a ideia, podemos usar a analogia da gestão financeira.

- **Caso Episódico (Planenjamento de Férias):** O seu objetivo é maximizar a "felicidade total"durante uma viagem de 10 dias. Você tem um orçamento e um tempo finitos.
- **Caso Contínuo (Gestão da Vida Financeira):** O seu objetivo é maximizar o seu "saldo médio"ou "rendimento líquido mensal", para sempre. Uma despesa ou um ganho inesperado (R_{t+1}) não é avaliado de forma isolada, mas sim em comparação com o seu orçamento médio esperado ($\bar{R}$). Uma ação (uma compra, um investimento) é "boa"se o seu resultado o deixar, a longo prazo, acima da sua média financeira esperada.

Conclusão: Com uma simples mudança na definição de performance e de retorno, os mesmos princípios e algoritmos de gradiente de política podem ser aplicados a uma classe muito mais ampla de problemas contínuos e sem fim, que são comuns em aplicações no mundo real.

O Limite das Ações Discretas

Até agora, a maioria dos algoritmos que estudamos, especialmente os métodos de valor da ação (como Q-learning), funcionam bem para um número finito e discreto de ações. A sua lógica central muitas vezes depende de encontrar a "melhor"ação através de uma operação de maximização, como $\max_a Q(s, a)$.

O Desafio do Mundo Real: Ações Contínuas

Muitos problemas práticos, no entanto, envolvem ações que não são discretas, mas sim contínuas. Por exemplo:

- O ângulo exato para virar o volante de um carro.
- A quantidade de força a ser aplicada por um braço robótico.

Nestes casos, existe um número **infinito** de ações possíveis.

Porque os Métodos de Valor da Ação Falham?

É computacionalmente impossível calcular e armazenar um valor $Q(s, a)$ para cada uma das infinitas ações. Além disso, o passo de maximização ($\max_a Q(s, a)$) torna-se um problema de otimização complexo e intratável que teria de ser resolvido a cada passo de tempo. Os métodos de gradiente de política oferecem uma solução prática e elegante para este desafio.

Aprender uma Distribuição de Probabilidade

A mudança de paradigma proposta pelos métodos de gradiente de política é: em vez de aprender o valor de cada ação, o agente aprende os **parâmetros de uma distribuição de probabilidade** sobre o espaço de ações. A ação a ser tomada é então amostrada a partir desta distribuição.

Exemplo: A Distribuição Normal (Gaussiana)

A abordagem mais comum para ações contínuas é modelar a política como uma distribuição normal (ou Gaussiana). Esta distribuição é definida por apenas dois parâmetros:

- **Média (μ):** Representa a "melhor aposta"ou a ação mais provável a ser tomada.
- **Desvio Padrão (σ):** Representa o nível de **exploração**. Um σ grande significa que o agente irá experimentar ações muito diferentes da média, enquanto um σ pequeno significa que as suas ações serão muito consistentes e próximas da média.

A política, $\pi(a|s, \theta)$, é então a função de densidade de probabilidade da distribuição normal. A ação a ser tomada é **amostrada** a partir desta distribuição.

Parametrizando a Política

A chave para a aprendizagem é fazer com que a média e o desvio padrão dependam do estado atual, s. Para isso, usamos aproximação de funções. O vetor de parâmetros da política, θ, é dividido em duas partes: θ_μ para a média e θ_σ para o desvio padrão.

1. **Parametrização da Média:** A média pode ser aproximada como uma função linear das features do estado:
$$\mu(s, \theta_\mu) \doteq \theta_\mu^T x(s) \tag{16}$$
2. **Parametrização do Desvio Padrão:** O desvio padrão deve ser sempre positivo. Uma forma robusta de garantir isso é aproximá-lo como a exponencial de uma função linear:
$$\sigma(s, \theta_\sigma) \doteq \exp(\theta_\sigma^T x(s)) \tag{17}$$

Fonte: Equações (13.20), página 336.

O Ciclo de Ação e Aprendizagem

A cada passo de tempo, o agente segue este ciclo:

1. **Observar o Estado:** O agente observa o estado atual, S_t.
2. **Calcular a Distribuição:** Usa os seus pesos atuais ($\theta_\mu, \theta_\sigma$) para calcular a média $\mu(S_t)$ e o desvio padrão $\sigma(S_t)$.
3. **Amostrar uma Ação:** O agente **amostra** uma ação A_t da distribuição Normal $\mathcal{N}(\mu(S_t), \sigma(S_t)^2)$ que acabou de calcular.
4. **Executar e Observar:** A ação A_t é executada no ambiente, e o agente recebe um feedback (o retorno G_t ou o erro TD δ_t).
5. **Atualizar os Pesos:** O algoritmo de gradiente de política (REINFORCE, Actor-Critic) usa este feedback para **ajustar os pesos** θ_μ **e** θ_σ na direção do gradiente ascendente, $\nabla \ln \pi(A_t|S_t, \theta)$, de forma a melhorar a performance.

Vantagem Principal e Conclusão

A abordagem de parametrizar a política para ações contínuas é extremamente poderosa porque resolve um problema aparentemente intratável.

Os métodos de gradiente de política transformam um problema de decisão num espaço de ações ***infinito*** *num problema de otimização num espaço de parâmetros* ***finito***.

Esta é a chave que torna tratáveis problemas do mundo real com espaços de ação contínuos, como a robótica (controlo de motores e articulações), finanças (alocação de portfólio) e muitos outros domínios de controlo.