REINFORCEMENT LEARNING

UNIDADE III
ABORDAGENS DE APRENDIZADO POR REFORÇO

Elaboração

Natasha Sophie Pereira

Atualização

Bruno Iran Ferreira Maciel

Produção

Equipe Técnica de Avaliação, Revisão Linguística e Editoração

SUMÁRIO

ABORDAGENS DE APRENDIZADO POR REFORÇO

UNIDADE III

Nesta unidade estudaremos a abordagem de aprendizado por reforço passiva e ativa. A abordagem passiva é quando o ambiente passa de um estado para outro sem intervenção do agente, onde, a cada transição, o agente apenas percebe o estado e recebe reforço, atualizando sua representação do valor desse estado.

Já na abordagem de aprendizado por reforço ativa, o agente tenta localizar uma boa política de ação, em que, a cada transição, o agente compreende o estado e obtém o reforço, atualizando sua interpretação do valor nesse estado e escolhendo uma ação a ser executada que mudará o estado do ambiente.

A grande diferença destas abordagens é a exploração, em que um agente deve experimentar tanto quanto possível do seu ambiente, a fim de aprender a se comportar nele.

CAPÍTULO 1

ABORDAGEM PASSIVA

A abordagem de aprendizado por reforço passiva faz uso de um conceito baseado em estados, num ambiente completamente observável. No aprendizado passiva a política π do agente é fixa, no estado s ele sempre executa a ação $\pi(s)$.

A abordagem passiva é semelhante à de avaliação de política. A principal diferença é que o agente de aprendizado passivo não conhece o modelo de transição *P(s' | s, a)* descrito no algoritmo de iteração de política (visto na seção 2.1.4 da Unidade II), que especifica a probabilidade de alcançar o estado *s'* a partir do estado *s* e depois realizar a ação *a*; ele também não conhece a função de recompensa *R(s)*, que especifica a recompensa para cada estado (RUSSELL e NORVIG, 2010).

O agente executa um conjunto de experiências no ambiente usando sua política π. Em cada experiência, o agente começa num estado inicial e experimenta uma sequência de transições de estados até alcançar um dos estados terminais. Suas percepções fornecem tanto o estado atual quanto a recompensa recebida neste estado.

A seguinte equação descreve a abordagem de aprendizado por reforço passiva:

$$U^{\pi}(s) = E\sum_{t=0}^{\infty} \gamma^{t} R(S_t)$$

O objetivo é utilizar as informações sobre recompensas para aprender a utilidade esperada U^{π} associada a cada estado não terminal s. A utilidade é definida como a soma esperada de recompensas (descontadas) obtidas se a política π for seguida.

A recompensa R(s) para o estado S_t (uma variável aleatória) é o estado alcançado no tempo t quando é executada a política π e S_0 = s, e a variável γ representa o fator de desconto.

Os métodos que compreendem o aprendizado passivo são: **Estimativa de Utilidade Direta**, **Programação Dinâmica Adaptativa** e **Diferença Temporal**, que serão vistas a seguir.

Utilizamos ao abordagem de aprendizado por reforço passiva quando queremos que um agente aprenda sobre as utilidades de vários estados sob uma política fixa. Como as escolhas para cada estado são predeterminadas, o aprendizado por reforço passivo não é particularmente útil para permitir que um agente aprenda como deve se comportar em um ambiente, mas é útil para nós aprendermos um passo no caminho para o aprendizado de reforço ativo.

1.1. Estimativa de utilidade direta

Ainda estamos operando sob um ambiente estocástico, portanto, uma ação específica executada em um estado específico nem sempre leva ao mesmo estado seguinte. Se quisermos aprender as utilidades desses estados sob uma política fixa, poderemos imaginar uma maneira bastante direta de fazê-lo:

- execute a política várias vezes;
- no final de cada execução, calcule a utilidade para cada estado na sequência (lembre-se, a utilidade de um estado é a soma de recompensas para esse estado e todos os estados subsequentes);
- atualize a utilidade média para cada um dos estados que observamos com nossos novos pontos de dados.

A ideia é que a utilidade de cada estado é a recompensa total que se espera a contar desse estado em diante (conhecido como recompensa a obter), e a cada teste propicia uma amostra dessa quantidade para cada estado visitado (RUSSELL; NORVIG, 2010).

Como os resultados menos prováveis acontecerão com menos frequência, eles afetarão menos nossas estimativas, o que significa que não precisamos conhecer um modelo de transição para que isso funcione. Na verdade, isso (provavelmente) acabará convergindo para as verdadeiras utilidades, mas é lento, porque não tira proveito do processo de decisão de Markov. Lembre-se, uma vez que calculamos a utilidade de um estado como recompensa, a utilidade de cada estado pode ser escrita estritamente em termos das utilidades de seus vizinhos imediatos. Isto é, os valores de utilidade obedecem às equações de Bellman para uma política fixa:

$$U^{\pi}(s) = R(s) + \gamma \sum_{s'} P(s' \mid s, \pi(s)) U^{\pi}(s')$$

> Em termos mais amplos, podemos visualizar a estimativa de utilidade direta como a busca em um espaço de hipóteses *U* muito maior do que precisa ser no sentido de incluir muitas funções que violam as equações de Bellman (RUSSELL; NORVIG, 2010, p. 726).

1.2. Programação dinâmica adaptativa

A ideia da programação dinâmica adaptativa (ou PDA) é aprender o modelo de transição e a função de reforço empiricamente (ao invés das utilidades). Para isso, ela usa a iteração de valor simplificada (sem o *max*) para obter as utilidades.

Um agente PDA obtém proveito das limitações entre as utilidades de estados compreendendo o modelo de transição que os conecta e soluciona o processo de decisão de Markov correspondente, fazendo uso de um método de programação dinâmica. Isso significa, para um agente de aprendizado passivo, inserir o modelo de transição aprendido P(s' | s, π(s)) e as recompensas observadas R(s) nas equações de Bellman para calcular as utilidades dos estados.

O processo de aprender o modelo em si é fácil porque o ambiente é completamente observável, ou seja, temos uma tarefa de aprendizado supervisionada em que a entrada é um par de estado-ação e a saída é o estado resultante. Pensando na forma mais simples de representar esse modelo de transição, poderíamos pensar numa tabela de probabilidades, onde o controle da frequência com que ocorre cada resultado de ação é mantido e a avaliação da probabilidade de transição *P(s', s, a)* a partir da frequência com que *s'* é alcançado é computada quando se executa *a* em *s* (RUSSELL; NORVIG, 2010).

Russell e Norvig (2010) propuseram o algoritmo AGENTE-PDA-PASSIVO, mostrado no Quadro 4, que implementa um programa de agente completo para um agente de PDA passivo.

Com relação à rapidez com que suas suposições de valores se aprimoram, o agente de PDA é delimitado apenas por sua capacidade de compreender o modelo de transição. Assim sendo, ele provê um padrão de semelhança com outros algoritmos de aprendizado por reforço, todavia, ele é um tanto impraticável para grandes espaços de estados.

Quadro 4. Agente de aprendizado por reforço passivo baseado em PDA.

função AGENTE-PDA-PASSIVO(*percepção*) **retorna** uma ação
entradas: *percepção,* um percepção indicando o estado atual *s'* e o sinal de recompensa *r'*
variáveis estáticas: , uma política fixa
mdp, um MDP com modelo *T*, recompensas *R* e desconto
U, uma tabela de utilidades, inicialmente vazia
, uma tabela de frequências referente aos pares estado-ação, inicialmente zero
s, a, estado e ação anteriores, inicialmente nulos
se *s'* é novo **então faça** *U[s']* *r'; R[s']* r'
se *s* é não nulo **então faça**
incrementar [*s, a*] e [*s', s, a*]
para cada *t* tal que [*t, s, a*] é diferente de zero **faça**
P(t, s, a) [*t, s, a*] / [*s, a*]
U AVALIAÇÃO-DE-POLITICA(*p, U, mdp*)
se *s'. TERMINAL?* **então** *s, a* nulo **senão** *s, a s',* [*s'*]
retornar *a*

Fonte: Russell e Norvig (2010, p. 726).

1.3. Aprendizado por diferença temporal

Introduzidos por Sutton (1984 e 1988), os algoritmos de aprendizado por diferença temporal (DT) aprendem novas estimativas do valor com base em outras estimativas. O método DT não exige um modelo exato do sistema. Essa procura estima valores de utilidade para cada estado do ambiente por recompensas oriundas das transições e de valores de estados sucessivos.

O aprendizado ocorre diretamente a partir da experiência, sem a necessidade de um modelo completo do ambiente, com a vantagem de atualizar as estimativas da função valor a partir de outras estimativas já aprendidas em estados sucessivos (bootstrap), sem a necessidade de alcançar o estado final de um episódio antes da sua atualização. Assim, a avaliação de uma política é encarada como um problema de predição, isto é, estima-se a função valor-estado U^{π} sob a política π.

Perante os outros dois métodos apresentados anteriormente, o método DT apresenta algumas vantagens relevantes:

- não exige o modelo MDP do ambiente (não exige conhecimento prévio do modelo de transição do ambiente);
- pode ser implementado de forma totalmente incremental para aplicações *on-line* (a sua atualização considera apenas o estado seguinte);
- tem garantida a convergência assintótica para a resposta correta (embora as atualizações da função valor não sejam obtidas a partir dos dados reais, mas de valores aproximados);
- os métodos DT são mais rápidos na sua convergência para tarefas estocásticas (TSITSIKLIS, 1994).

A DT usa as transições obtidas para ajustar os valores dos estados observados, de forma a esses concordarem com as equações de restrições (que definem o ambiente). Assim, numa transição de estado $s \rightarrow s'$, é aplicada a seguinte atualização à utilidade:

$$U^{\pi}(s) = U^{\pi}(s) + \alpha\left(R(s) + \gamma U^{\pi}(s') - U^{\pi}(s)\right)$$

Sendo α a taxa de aprendizado, assumindo valores no intervalo [0, 1].

A taxa de aprendizado α determina a velocidade com que o agente assimila a informação, apresentando-se menor à medida que $\alpha \rightarrow 0$. Se a não for um parâmetro fixo, mas uma função que diminui à medida que aumenta o número de vezes que o estado é visitado, então $U^{\pi}(s)$ convergirá para o valor correto (RUSSELL; NORVIG, 2010). A atualização

somente envolve o sucessor observado s' (e não todos os estados seguintes possíveis), tendo implícito um cálculo pouco exigente em termos computacionais.

Russell e Norvig (2010) propuseram o algoritmo AGENTE-DT-PASSIVO, mostrado no Quadro 5, que implementa um programa de agente de aprendizado por reforço que aprende estimativas de utilidade com diferenças temporais.

Quadro 5. Agente de aprendizado por reforço passivo baseado em DT.

função AGENTE-DT-PASSIVO(*percepção*) **retorna** uma ação
entradas: *percepção,* um percepção indicando o estado atual *s'* e o sinal de recompensa *r'*
variáveis estáticas: , uma política fixa
U, uma tabela de utilidades, inicialmente vazia
, uma tabela de frequências para estados, inicialmente zero
s, a, r estado, ação e recompensas anteriores, inicialmente nulos
se *s'* é novo **então faça** *U[s']* *r'*
se *s* é não nulo **então faça**
incrementar [*s*]
U[s] *U[s]* + *a*([*s*]) (*r* + *U[s']* – *U[s])*
se *TERMINAL? [s']* **então** *s, a, r* nulo **senão** *s, a, r s',* [*s'*], *r'*
retornar *a*

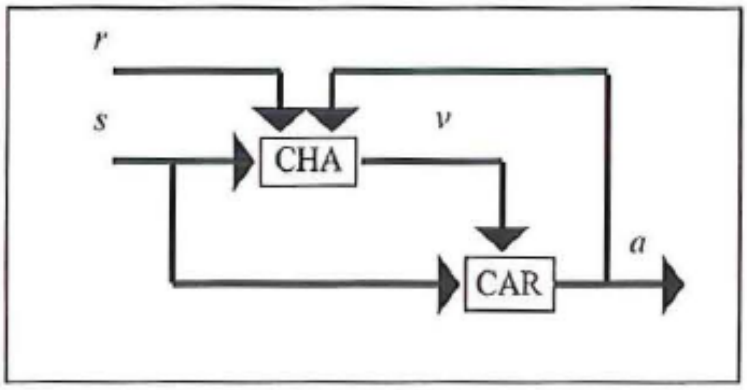
Fonte: Russell e Norvig (2010, p. 729).

As duas principais estratégias encontradas na literatura para aprendizado por reforço utilizando DT são as seguintes (KAELBLING; LITTMAN; MOORE, 1996):

» **Crítico Heurístico Adaptativo (CHA) (**BARTO; SUTTON; ANDERSON, 1983): esse algoritmo consiste de dois componentes, um crítico adaptativo (CHA) e um de aprendizado por reforço (CAR). A Figura 31 apresenta a arquitetura para o crítico heurístico adaptativo.

O componente de aprendizado por reforço busca agir de forma a maximizar o valor heurístico *v*, que é calculado pelo crítico. O crítico usa o sinal real externo de reforço *r* para aprender a avaliar os estados *s*. Na maioria das implementações, os componentes CHA e CAR operam simultaneamente e apenas para essas implementações há garantia, sob condições adequadas, de convergência para uma política ótima.

Figura 31. Arquitetura para o crítico heurístico adaptativo.



Fonte: Kaelbling, Littman e Moore (1996).

» ***Q-Learning*:** esse é o algoritmo de aprendizado por reforço mais estudado para os propósitos de controle, possuindo provas de convergência. *Q-Learning* estima a avaliação do par estado-ação diretamente a partir de sinais externos de reforço utilizando uma política gulosa. Veremos esse algoritmo em detalhes na Unidade IV.

CAPÍTULO 2

ABORDAGEM ATIVA

Na abordagem de aprendizado de reforço ativa, um agente deve escolher quais ações deve executar, de modo diferente do aprendizado passivo, onde é a política fixa que estabelece sua conduta. Isso torna um agente de reforço ativo mais poderoso, mas, também, adiciona algumas complicações quando tentamos formalizar seu processo em um algoritmo. Não só é preciso escolher a ação em cada iteração, mas, também, é necessário garantir que nossas escolhas garantam que possamos encontrar uma política ótima.

No aprendizado ativo, o agente precisa aprender um modelo completo com probabilidades de resultados para todas as ações, em vez de aprender apenas o modelo para a política fixa. O mecanismo de aprendizado do algoritmo AGENTE-PDA-PASSIVO funciona bem para isso, desde que se leve em conta o fato de que o agente tem uma escolha de ações. As utilidades que ele precisa aprender são aquelas definidas pela política ótima, elas obedecem à equação de Bellman (RUSSELL; NORVIG, 2010).

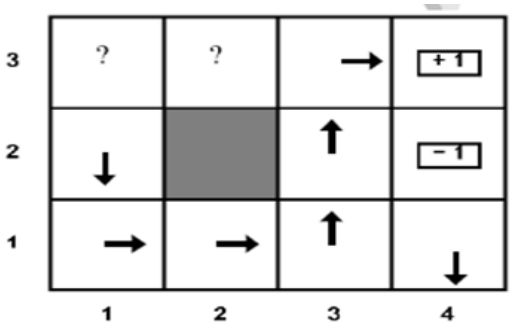
Se você pensar sobre esta equação no contexto de tentar aprender uma política ótima, pode fazer sentido escolher a melhor ação que conhecemos cada vez, e usar o resultado dessa ação para atualizar a utilidade do estado *s*. Considere o seguinte cenário: se sua refeição favorita desde criança é frango e macarrão com queijo, então comer esta refeição quando você está com fome é a escolha que maximiza sua recompensa (ela é sua comida favorita!). Porém, se você sempre escolhe esta ação (que maximiza a sua recompensa), não tem a chance de descobrir melhores opções de comida, caso elas existam. Então, para encontrar uma política ótima, temos que arriscar alguns resultados ruins pela possibilidade de que algumas ações possam levar a melhores resultados do que já conhecemos. Isso nos dá o problema de como equilibrar a **exploitação** (fazendo escolhas que sabemos que vai nos ajudar) contra a **exploração** (descobrir quão valiosas são as escolhas que ainda não fizemos).

2.1. Exploração

Suponha um agente de PDA e considere o modo como ele deve ser modificado para decidir quais ações executar. A Figura 32 apresenta um agente PDA que segue a recomendação da política ótima para o modelo aprendido em cada etapa. Podemos verificar na figura que o agente não aprende as utilidades verdadeiras ou a política ótima verdadeira, em vez disso, o que acontece é que ele encontra uma política que alcança a recompensa +1 ao longo da rota inferior.

Após experiências com pequenas variações, o agente se fixa nessa política. Dessa forma, ele nunca aprende as utilidades dos outros estados e, assim, nunca encontra a rota ótima via (1, 2), (1, 3) e (2, 3). A esse agente é dado o nome de **agente guloso**. Esse agente raramente converge para a política ótima correspondente a esse ambiente, e, às vezes, converge para políticas realmente ruins.

Figura 32. Política não ótima para a qual o agente guloso converge nessa sequência.



Fonte: Russell e Norvig (2010, p. 732).

As ações fornecem recompensas de acordo com o modelo atual aprendido, no entanto, elas fazem bem mais que isso, elas também contribuem para o aprendizado do modelo verdadeiro afetando as percepções que são recebidas, isso faz com que haja um melhoramento do modelo, fazendo com que o agente receba recompensas melhores no futuro. Mas, para que isso ocorra, um agente deve assumir um compromisso entre **exploitação** (aproveitamento) para maximizar a sua recompensa (que reflete em suas estimativas de utilidades atuais) e **exploração**, a fim de maximizar seu bem-estar em longo prazo.

Usando apenas a exploitação, corre-se o risco de ficar paralisado num único lugar. Já a exploração sozinha para melhorar o conhecimento é inútil, se não colocar em prática o conhecimento. Os agentes devem, então, explorar no começo e aproveitar no final.

O problema de decisão de um agente pode ser visto como o problema de selecionar determinada ação quando ele está em determinado estado. Um exemplo bem conhecido de um problema de decisão envolvendo um único agente é uma variação do problema do "**bandido de *k* braços**" ou *k-armed bandit* (SUTTON e BARTO, 1998), nomeado em analogia a uma hipotética máquina de caça-níqueis com *k* alavancas. Esse problema consiste em um agente que pode escolher dentre *k* opções de ação, cada uma com uma probabilidade diferente e imutável de ser a escolha correta. Cada vez que o agente escolhe a opção correta, esta recebe um sinal de recompensa positivo, nos demais casos recebe um sinal de recompensa negativo. Conforme o agente explora estas opções ao longo de múltiplas iterações, tende a adequar sua política de tomada de decisões para favorecer a opção com maior probabilidade de resultar em recompensa positiva. Problemas como

esse são extremamente difíceis de resolver com exatidão para se obter um método de exploração ótimo. No entanto, eles apresentam um esquema razoável que eventualmente leva a um comportamento ótimo do agente.

Se mantivermos estimativas dos valores das ações, a qualquer momento haverá pelo menos uma ação cujo valor estimado seja maior. Chamamos essas ações de **ações gulosas**. Quando selecionamos uma dessas ações, dizemos que estamos explorando seu conhecimento atual dos valores das ações. Se, ao invés disso, selecionarmos uma ação não gulosa dizemos que estamos explorando, porque isso permite o aumento da estimativa de valor da ação não gulosa. Exploitação é o certo a se fazer para maximizar a recompensa esperada em um único passo, contudo, a exploração pode produzir a maior recompensa total em longo prazo. Por exemplo, suponha que o valor de uma ação gulosa seja conhecido com certeza, enquanto várias outras ações são estimadas como quase tão boas, mas com uma incerteza substancial. A incerteza é tal que pelo menos uma dessas ações provavelmente é melhor do que a ação gulosa, mas não se sabe qual. Se tivermos muitos passos à frente para fazer seleções de ações, talvez seja melhor explorar as ações não gulosas e descobrir quais delas são melhores do que a ação gulosa. A recompensa é menor em curto prazo, durante a exploração, porém, maior em longo prazo, porque depois de ter descoberto as melhores ações, podemos explorá-las mais vezes. Como não é possível realizar exploração e exploitação com uma única seleção de ação, geralmente nos referimos ao conflito entre exploração e exploitação (SUTTON e BARTO, 2018).

De qualquer forma, escolher se é melhor explorar ou exploitar depende, de maneira complexa, dos valores precisos das estimativas, das incertezas e do número de etapas restantes. Existem muitos métodos sofisticados para equilibrar exploração e exploitação. A estratégia mais fácil para atingir esse equilíbrio é simplesmente tomar uma ação aleatória algumas vezes (por razões técnicas, precisamos tomar a ação aleatória de forma cada vez menor, à medida que nosso agente aprende). Uma maneira um pouco sofisticada é definir uma **função de exploração** *f(u, n)*, dessa forma, influenciará no valor percebido das ações que determinam sua entrada. Consequentemente, podemos apenas escolher a melhor ação; mas, agora, os estados menos visitados parecerão melhores do que realmente são.

A função de exploração *f(u, n)* toma como entradas a utilidade atual conhecida de um estado (*u)* e o número de vezes que esse estado foi visitado (*n*). A função de exploração precisa estar aumentando em *u* e diminuindo em *n*. Como no exemplo:

$$f(u, n) = u, sen \geq N_e$$

R^+ de outra forma

Onde N_e é o número de vezes que se deseja explorar um estado antes que esteja razoavelmente certo que este é o seu valor, e R^+ é uma estimativa otimista do valor dos estados explorados.

Em Las Vegas, um bandido com um braço é uma máquina caça-níqueis. Um jogador pode inserir uma moeda, puxar a alavanca e recolher os ganhos (se houver). Um bandido com *n*-braços tem *n* alavancas. O jogador deve escolher qual alavanca acionar em cada moeda sucessiva – aquela que pagou a melhor recompensa ou, quem sabe, a que ainda não foi experimentada.

O problema do bandido com *n*-braços é um modelo formal para problemas reais em muitas áreas de importância vital, como a decisão sobre o orçamento anual para pesquisa e desenvolvimento em IA. Cada braço corresponde a uma ação (como a de alocar $20 milhões para o desenvolvimento de novos livros didáticos sobre IA) e a recompensa de puxar a alavanca correspondente aos benefícios (imensos) obtidos a partir da execução da ação. A exploração, quer seja de um novo campo de pesquisa ou de um novo centro comercial, é arriscada, dispendiosa e tem recompensas incertas; por outro lado, deixar de explorar significa que nunca se descobrirá alguma ação que valha a pena.

Para formular um problema de bandido corretamente, devemos definir com exatidão o que queremos dizer com comportamento ótimo. A maior parte das definições na literatura supõe que o objetivo é maximizar a recompensa total esperada obtida ao longo do tempo de duração do agente. Essas definições exigem que a expectativa seja considerada sobre os mundos possíveis em que o agente poderia estar, bem como sobre os resultados possíveis de cada sequência de ações em qualquer mundo dado. Aqui, um "mundo" é definido pelo modelo de transição $P(s', s, a)$. Desse modo, para agir de forma ótima, o agente precisa de uma distribuição *a priori* sobre os modelos possíveis. Os problemas de otimização resultantes em geral são terrivelmente intratáveis.

Em alguns casos – por exemplo, quando o retorno de cada máquina é independente e são usadas recompensas descontadas –, é possível calcular um índice *Gittins* para cada máquina caça-níqueis. O índice é uma função apenas do número de vezes que a máquina caça-níqueis foi usada em jogos e do quanto ela retornou de recompensa. O índice para cada máquina indica o quanto vale a pena investir mais; falando de modo geral, quanto maior o retorno esperado e a incerteza na utilidade de uma escolha, melhor. A escolha da máquina com o valor de índice mais alto fornece uma política de exploração ótima. Infelizmente, não foi encontrado nenhum modo de estender os índices de *Gittins* a problemas de decisão sequencial.

Se você considerar cada braço em um problema de bandido com n-braços uma sequência possível de genes, pode ser provado que os algoritmos genéticos vão alocar moedas de forma ótima, dado um conjunto apropriado de suposições de independência.

[Texto extraído do livro de Russell e Norvig (2010)].

2.2. Aprendizado de uma função de ação-valor

Vimos na Unidade II que o conceito de função valor associa o valor diretamente a estados do ambiente, sendo o valor de uma ação indiretamente definido como valor do estado – consequência da aplicação da ação –, e que era aceitável apenas para agente com modelo efetivo do ambiente (prevendo o estado resultante da execução de cada ação).

Nesta seção veremos a função de ação-valor que associa o valor diretamente a pares (estado, ação). Ela é aplicável a agente sem modelo efetivo do ambiente, quando não há nenhum modelo de ambiente acessível e com o modelo apenas perceptivo de ambiente inacessível. A Figura 33 apresenta uma comparação entre o aprendizado por reforço de uma função valor e de uma função ação-valor.

Na seção anterior, de exploração, construímos um agente de PDA ativo, agora vamos considerar como construir um agente de aprendizado ativo de diferença temporal (DT). A principal diferença em relação ao caso passivo é que o agente não está mais equipado com uma política fixa π. Dessa forma, se ele for aprender uma função utilidade U, necessitará compreender um modelo apto a escolher uma ação que se baseia em U por meio da observação prévia de um passo (RUSSELL e NORVIG, 2010).

O problema de aquisição de modelo para o agente DT é idêntico ao do agente de PDA, sendo a mesma equação de atualização do DT utilizada para este novo agente.

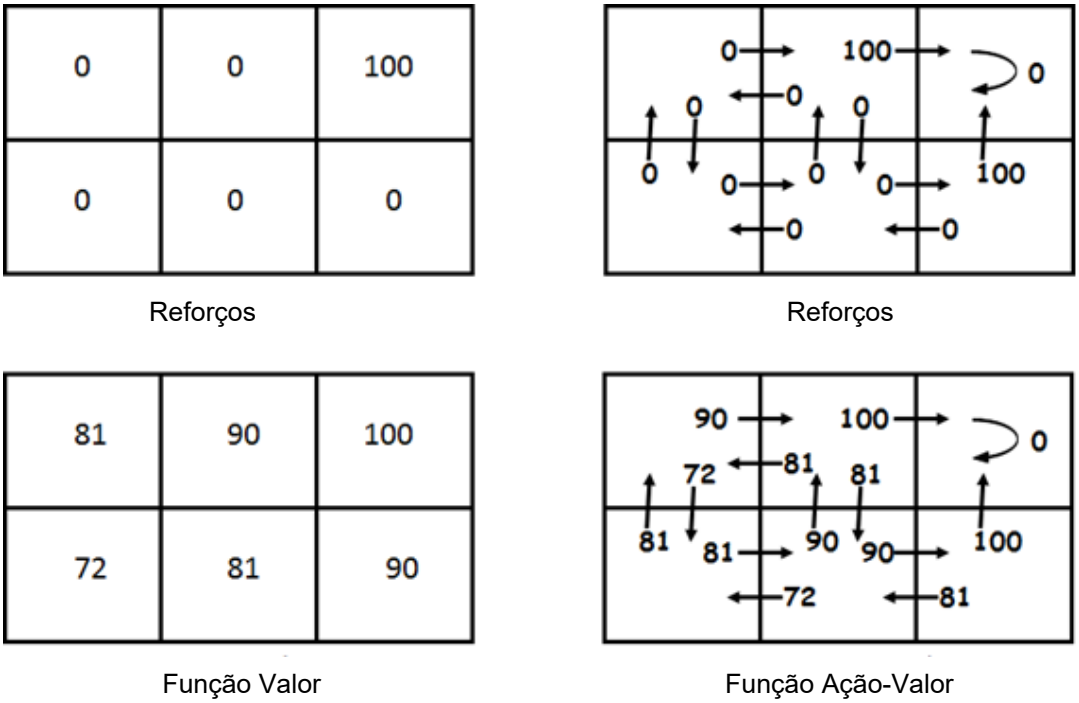
Relembre a equação de atualização DT:

$$U^{\pi}(s) = U^{\pi}(s) + \alpha\left(R(s) + \gamma U^{\pi}(s') - U^{\pi}(s)\right)$$

Isto é possível pela seguinte razão: considere que o agente realize um passo que comumente leve a um bom destino, mas devido ao não determinismo do ambiente, o agente acaba em um estado catastrófico. A regra de atualização DT levará a circunstância tão a sério de forma que seu resultado se tornasse o resultado normal da ação, apesar de poder supor que, devido ao fato de o resultado ter sido obtido por casualidade, o agente não deve se preocupar muito com ele. É claro que o resultado improvável ocorrerá com pouca frequência em um grande conjunto de sequências de treinamento. No entanto, em longo prazo seus efeitos serão ponderados de forma proporcional a sua probabilidade. Isso nos mostra que o algoritmo de DT convergirá para os mesmos valores que a PDA, à medida que o número de sequências de treinamento tender a infinito (RUSSELL e NORVIG, 2010).

A aplicação do método da diferença temporal na função-valor da ação permite que o ambiente seja desconhecido, e leva a um dos métodos mais difundidos do aprendizado por reforço, o *Q-Learning* (que será visto em detalhes na Unidade IV).

Figura 33. Comparação função-valor e função ação-valor.



Fonte: Robin, 2002.

REFERÊNCIAS

ABBEEL, P.; ANDREW, Y. Apprenticeship learning via inverse reinforcement learning. **Proceedings of the twenty-first international conference on Machine learning**. ACM, 2004.

BARTO, A. G; SUTTON, R. S.; ANDERSON, C. W. Neuronlike adaptative elements that can solve difficult learning control problems. **IEEE Transactions on Systems, Man and Cybernetics**, v. 3, n. 5, p. 834-846, 1983.

BATISTA, A. F. de M. **Processos de decisão de Markov**. 2008. Disponível em: http://alepho.info/cursos/ia/trab/andre/8/ExercicioMDP.pdf.

BENGIO, Y.; COURVILLE, A.; VINCENT, P. Representation learning: a review and new perspectives. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 35, n. 8, p. 1.798-1828, Aug 2013.

BERTSEKAS, D. P.; TSITSIKLIS, J. N. Neuro-Dynamic Programming. **Athena Scientific**. Belmont, M.A., 1996.

BERTSEKAS, D. P.; YU. H. **Q-learning and enhanced policy iteration in discounted dynamic programming.** In: Decision and Control (CDC), 2010, 49th, IEEE Conference on. p. 1.409-1.416.

CABRAL, D. **Aprendizado por reforço – um conto de Natal**. (2018). Disponível em: https://www.deviante.com.br/noticias/aprendizado-por-reforco-um-conto-de-natal/ Acesso em: 30 abr. 2019.

CARVALHO, A. C. P. L. F; BRAGA, A. P.; LUDEMIR, T. B. Fundamentos de redes neurais artificiais. **XI Escola Brasileira de Computação**, 1998.

CASSANDRA, A. R. **A survey of POMDP applications**. Presented at the AAAI Fall Symposium, 1998.

CLEMEN, R. T. **Making hard decisions**: an introduction to decision analysis. 2. ed. Belmont: Duxbury, 1996.

COPELAND, M. **What is the difference between artificial intelligence, machine learning, and deep learning**? (2016). Disponível em: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/. Acesso em: 4 abr. 2019.

DAHL, G. E; SAINATH, T; HINTON, G. E. **Improving deep neural networks for LVCSR using rectified linear units and dropout**. 2013. Disponível em: https://ieeexplore.ieee.org/document/6639346. Acesso em: 27 maio 2019.

DEAN, J. *et al.* Large scale distributed deep networks. *In*: **Proceedings of the 25th International Conference on Neural Information Processing Systems**, NIPS'12, p. 1.223-1.231, 2012.

DENG, C.; ER, M. J. Real-time dynamic fuzzy Q-learning and control of mobile robots. *In*: **Control Conference**, 2004. 5th Asian. Vol. 3. p. 1.568-1.576.

DEVARAKONDA, M.; TSOU, C.-H. Automated problem list generation from electronic medical records in IBM Watson. *In* **Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence**, AAAI'15, p. 3.942-3.947. AAAI Press, 2015.

DEEPMIND. **AlphaGo**. Disponível em: https://deepmind.com/research/case-studies/alphago-the-story-so-far. Acesso em: 30 maio 2020.

ELMAN, J. L. Finding Structure in Time. **Cognitive Science**, n. 14, p. 179-211, 1990.

GARCIA, A. B; VEZHNEVETS, A; FERRARI, V. An active search strategy for efficient object class detection. In **2015 IEEE Conference on Computer Vision and Pattern Recognition** (CVPR), p. 3.022-3.031. IEEE, 2015.

GERA, D. L. **Ancient greek ideas on speech, language and civilization**. [S.l.]: Oxford University Press, 2003.

GOLMAKANI, A.; SILVA, A. A.; FREIRE, E. M. S.; BARBOSA, M. K.; CARVALHO, P. H. G; ALVES, V. L. **Cadeias de Markov**, 2014. Disponível em: http://www.im.ufal.br/evento/bsbm/download/minicurso/cadeias.pdf. Acesso em: 27 maio 2019.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, **A. Deep Learning**. MIT Press, 2016.

HAUSKRECHT, M. Dynamic decision making in stochastic partially observable medical domains: Ischemic heart disease example. *In*: KERAVNOU, E. *et al.* (Ed.). 6th **Conference on Artificial Intelligence in Medicine**. [S.l.]: Springer, 1997. (Lecture Notes in Artificial Intelligence, v. 1211), p. 296-299.

HAUSKRECHT, M. **Planning and control in stochastic domains with imperfect information**. Tese (Doutorado). EECS, Massachusetts Institute of Technology, 1997.

HAYKIN, S. **Neural networks**: a comprehensive foundation. [S.l.]: Prentice Hall PTR, 1994.

HINTON, G.; DENG, L.; YU, D.; DAHL, G. E.; MOHAMED, A.-r.; JAILTLY, N.; SENIOR, A.; VANHOUCKE, V.; NGUYEN, P.; SAINATH, T. N.; and KINGSBURY, B. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. **IEEE Signal Processing Magazine**, 29(6):82-97, 2012.

HINTON, G; *et al.* **Improving neural networks by preventing co-adaptation of feature detectors**. 2012. Disponível em: https://arxiv.org/pdf/1207.0580.pdf. Acesso em: 27 maio 2019.

HOCHREITER, S.; BENGIO, Y.; FRANSCONI, P. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. *In*: KOLEN, J.; KREMER, S., editors, **Field Guide to Dynamical Recurrent Networks**. IEEE Press, 2001.

JAIN, S. **An Overview of Regularization Techniques in Deep Learning (with Python code)**. 2018. Disponível em: https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/. Acesso em: 27 maio 2019.

JONES, M. T. **Um mergulho profundo nas redes neurais recorrentes**, 2017. Disponível em: https://imasters.com.br/data/um-mergulho-profundo-nas-redes-neurais-recorrentes. Acesso em: 27 maio 2019.

KAELBLING, L. P; LITTMAN, M. L.; MOORE, A. W. Reinforcement learning: a survey. **Journal of Artificial Intelligence Research**, V. 4, p. 237-285, 1996.

KLOPF, A. H. **Brain function and adaptive systems.** A heterostatic theory. Bedford, Massachusetts: Air Force Cambridge Research Laboratories, 1972.

KOFINAS, P.; DOUNIS, A. I. Fuzzy Q-Learning agent for online tuning of PID controller for DC motor speed control**. Algorithms**, v. 11, 2018.

KRIZHEVSKY, A; SUTSKEVER, I; HINTON, G. E. **ImageNet classification with deep convolutional neural networks. Advances in Neural information processing systems**, 2012.

KUZMIN, V. **Connectionist Q-learning in robot control task**. 2002.

LANDELIUS, T. **Reinforcement Learning and Distributed Local Model Synthesis**. PhD thesis. Linkoping University, Sweden. SE-581 83 Linkoping, Sweden. Dissertation n. 469, 1997.

LANGE, S.; RIEDMILLER, M. Deep auto-encoder neural networks in reinforcement learning. *In* **Proceedings of the 2010 International Joint Conference on Neural Networks** (IJCNN), p. 1-8, 2010.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient based learning applied to document recognition. **Proceedings of the IEEE**, 86(11): 2.278-2.324, 1998.

LIU, C.; CAO, Y.; LUO, Y.; CHEN, G.; VOKKARANE, V.; MA, Y.; CHEN, S.; HOU, P. A new deep learning-based food recognition system for dietary assessment on an edge computing service infrastructure. **IEEE Transactions on Services Computing**, PP(99):1-13, 2017.

LOPES, R. A. S.; BRAGA, V. G. de M.; LAMAR, M. V. **Sistema baseado em redes neurais e q-learning para jogar jogos eletrônicos**. 2017.

MATARIC, M. J. **Introdução à robótica**. [S.l.]: UNESP, 2014.

MAYER, Z. D. **Advanced Deep Learning with Keras in Phyton**. 2019. Disponível em: https://www.datacamp.com/courses/advanced-deep-learning-with-keras-in-python. Acesso em: 27 maio 2019.

McCULLOCK, J. **Q-Learning.*;.** 2012. Disponível em: http://mnemstudio.org/path-finding-q-learning-tutorial.htm. Acesso em: 4 abr. 2019.

MEDEIROS, F. G. Neto; PINTO, R. F. Júnior; ROCHA, M. G. O; SÁ, J. J. M. Júnior; PAULA, I. C. Júnior. Aprendizado profundo: conceitos, técnicas e estudo de caso de análise de imagens com Java. **III Escola Regional de Informática do Piauí**. Livro Anais – Artigos e Minicursos, v. 1, n. 1, p. 465-486, jun., 2017.

MNIH, V. *et al.* Human-level control through deep reinforcement learning. **Nature**, Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved., v. 518, n. 7540, p. 529-533, fev. 2015.

MOHAMED, A.; DAHL, G.; HINTON, G. Deep belief networks for phone recognition. **NIPS Workshop on deep learning for speech recognition and related applications,** 2009.

MOREIRA, S. **Rede Neural Perceptron Multicamadas**, 2018. Disponível em: https://medium.com/ensina-ai/rede-neural-perceptron-multicamadas-f9de8471f1a9. Acesso em: 27 maio 2019.

MOTA, I. C., (2018). **Aprendizado por reforço utilizando Q-Learning e redes neurais artificiais em jogos eletrônicos**. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-n4, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 50p.

MOUJAHID, A. **A practical introduction to deep learning with Caffe and Python**, 2016. Disponível em: http://adilmoujahid.com/posts/ 2016/06/introduction-deep-learning-python-caffe/.

NAIR, V.; HINTON, G. E. Rectified Linear Units improve Restricted Boltzmann Machines. In **Proceedings of the 27th International Conference on Machine Learning** (ICML-10), pages 807-814, 2010.

NIELSEN, M. A. **Neural networks and deep learning**. Determination Press, 2015.

PELLEGRINI, J.; WAINER, J. On the use of POMDPs to model diagnosis and treatment of diseases. *In*: E**ncontro Nacional de Inteligência Artificial (ENIA)**. Campinas, SP, Brazil: Sociedade Brasileira de Computação, 2003.

PELLEGRINI, J.; WAINER, J. **Processos de Decisão de Markov**: um tutorial. Instituto de Computação, Unicamp, Campinas-SP, Brazil. **RITA**. Volume XIV. Número 2. 2007.

PENG, J.; WILLIAMS, R. J. Incremental multi-step Q-learning. *In*: **Machine Learning**. Morgan Kaufmann, p. 226-232, 1996.

PINEAU, J. **Tractable Planning under uncertainty**: exploiting structure. Tese (Doutorado). Robotics Institute, Carnegie-Mellon University, 2004.

PIPE, A. G. An architecture for building "potential field" cognitive maps in mobile robot navigation. *In*: **Systems, man and cybernetics**, 1998. IEEE International Conference on. Vol. 3. p. 2.413-2.417.

POUPART, P. **Exploiting structure to efficiently solve large scale partially observable Markov decision processes**. Tese (Doutorado). University of Toronto, 2005.

PUTERMAN, M. L. **Markov decision processes**: discrete stochastic dynamic programming. New York, NY: Wiley-Interscience, 1994.

ROBIN, J. **Aprendizado por reforço**, 2002. Disponível em: www.cin.ufpe.br/~compint/aulas-IAS/ias/ias-021/rl.ppt. Acesso em: 27 maio 2019.

RUSSELL, S.; NORVIG, P. **Artificial intelligence**: a modern approach. 3. ed. New Jersey: Pearson Education, 2010.

SANTOS, M. R; DACORSO, A. L. R. (2016). Intuição e racionalidade: um estudo sobre tomada de decisão estratégica em empresas de pequeno porte. **Rev. Adm. UFSM**, Santa Maria, v. 9, número 3, p. 448-463, jul.-set. 2016.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. **Neural networks**, 61:85-117, 2015.

SIEBEL, N. T; SOMMER, G. Evolutionary Reinforcement Learning of Artificial Neural Networks. **International Jorunal of Hybrid Intelligent Systems**. p. 171-183, 2007.

SIKCHI, H. **Towards Safe Reinforcement Learning**. 2018. Disponível em: https://medium.com/@harshitsikchi/towards-safe-reinforcement-learning-88b7caa5702e. Acesso em: 27 maio 2019.

SILVA, T. L; GOUVÊA, M. M. Aprendizado por reforço clássica e conexionista: análise de aplicações. **Anais do EATI – Encontro Anual de Tecnologia da Informação**. Instituto Politécnico – Pontifícia Universidade Católica de Minas Gerais, p. 299-302, 2015.

SILVER, D., *et al.* Mastering the game of Go with deep neural networks and tree search. **Nature**, 529:484-489, 2016.

SIMON, P. **Too big to ignore**: The Business Case for Big Data. [S.l.]: Wiley, 2013.

SINGH, S.; JAAKKOLA, T.; JORDAN, M. I. Learning without state-estimation in partially observable markovian decision processes. *In*: **International Conference on Machine Leadning** (ICML). New Brunswick, NJ, USA: Morgan Kaufmann, 1994.

SKYMIND. **A Beginner's Guide to LSTMs and Recurrent Neural Networks.** 2017**.** Disponível em: https://skymind.ai/wiki/lstm#feedforward. Acesso em: 27 maio 2019.

SRIVASTAVA, N. *et al.* Dropout: a simple way to prevent neural networks from overfitting. **Journal of Machine Learning Research**. 2014, vol. 15, p. 1.929-1.958.

SUTTON, R. S. **Temporal credit assignment in reinforcement learning**. Tese de doutorado, University of Massachusetts Amherst, 1984.

SUTTON, R. S. **Learning to predict by the method of temporal differences**. Machine Learning, vol. 3, p. 9-44, 1988.

SUTTON, R. S.; BARTO, A. G**. Reinforcement learning: an introduction**, 2. ed. MIT Press, Cambridge, Massachusetts, EUA, 2018.

SUTTON, R. S.; SATINDER, S. P. **Reinforcement learning with replacing elegibility traces.** Cambridge: Dept. of Computer Science, MIT, 1996.

SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. Going deeper with convolutions. *In*: **Proceedings of 2015 IEEE Conference on Computer Vision and Pattern Recognition** (CVPR), p. 1-9, 2015.

TCH, A. **The mostly complete chart of Neural Networks, explained**, 2017. Disponível em: https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464. Acesso em: 27 maio 2019.

TSITSIKLIS, J. N. **Asynchronous stochastic approximation and Q-learning**. Boston: Kluwer Academic Publishers, 1994.

VARGAS, A. C. G.; PAES, A.; VASCONCELOS, C. N. **Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres**. *In*: Proceedings of the XXIX Conference on Graphics, Patterns and Images, p. 1-4. Sociedade Brasileira de Computação, 2016.

VASCONCELLOS, P. **Explicando Deep Reinforcement Learning com Super Mario ao invés de matemática**. 2018. https://paulovasconcellos.com.br/explicando-deep-reinforcement-learning-com-super-mario-ao-inv%C3%A9s-de-matem%C3%A1tica-4c77392cc733.

WAIBEL, A. *et al.* Phoneme Recognition Using Time-Delay Neural Networks. **IEEE Transactions on Acoustics, Speech and Signal Processing**, Volume 37, n. 3, p. 328-339, 1989.

WATKINS, C. J. C. H. **Learning from delayed rewards**. PhD Thesis. University of Cambridge. Cambridge, England, 1989.

WATKINS, C. J. C. H; DAYAN, P. Q-learning. *In*: **Machine Learning**. Vol. 8, p. 279-292, 1992.

WIENER, N. **The human use of human beings: cybernetics and society**. [S.l.]: Free Association Books, 1989.

Watkins, C. J. Models of delayed reinfor-cement learning, Master's thesis, PhD thesis,Psychology Department, Cambridge Univer-sity, Cambridge, United Kingdom. 1989

Mitchell, T. M. Does Machine Learning Really Work?. *AI Magazine*, *18*(3), 11. 1997.