REINFORCEMENT LEARNING

UNIDADE IV
Q-LEARNING

Elaboração

Natasha Sophie Pereira

Atualização

Bruno Iran Ferreira Maciel

Produção

Equipe Técnica de Avaliação, Revisão Linguística e Editoração

SUMÁRIO

Q-LEARNING UNIDADE IV

Qual é a diferença entre um bebê e um adulto? Mesmo desconsiderando as características físicas, isto pode parecer uma pergunta com uma resposta óbvia. Você pode dizer algo sobre um adulto normalmente ter habilidades de linguagem, melhor memória, melhor coordenação motora ou uma série de outras coisas. Mas por que não nascemos com essas habilidades?

Humanos têm uma longa fase de desenvolvimento antes de atingir a maturidade. A razão pela qual nascemos com alguns instintos específicos e muito espaço para aprendizado é o que nos permite ter a capacidade de nos adaptarmos a uma extensão maior para qualquer ambiente em que nos encontremos. Embora os montantes mais perceptíveis de aprendizado ocorram durante nosso estágio de desenvolvimento, estamos constantemente a afinar nossas habilidades e conhecimentos até morrermos. Porque o meio ambiente, bem como um corpo do próprio organismo, está sempre mudando de forma sutil, um organismo deve ser capaz de reajustar constantemente as suas circunstâncias. Pense sobre isto também: os organismos vivos são tão complexos que seria impossível para o DNA codificar cada possível sequência de ações ou habilidades que o organismo poderia precisar. Em vez disso, ele codifica nossos instintos básicos e, quando aprendemos, ajustamos os instintos em nossos músculos, cérebro, etc., de acordo com nossas necessidades específicas.

Embora existam alguns tipos de aprendizado, o aprendizado por reforço normalmente ajuda a ajustar nossas ações físicas e habilidades motoras. As ações que um organismo executa resultam num *feedback*, que, por sua vez, é traduzido em recompensa positiva ou negativa por essa ação. Quando um bebê aprende a andar, ele pode cair e sentir um pouco de dor. Esse *feedback* negativo ajudará o bebê a aprender o que não fazer. Se o bebê for capaz de ficar em pé por uma questão de tempo, então está fazendo algo certo e a realização do objetivo será um *feedback* positivo. Como o bebê continua a tentar andar, ele desenvolverá coordenação motora de tal forma que a recompensa será maximizada. Dor, fome, sede e prazer são alguns exemplos de reforços naturais. As ações podem resultar em recompensas imediatas ou fazer parte de uma cadeia mais longa de ações que acabam levando à recompensa.

Q-Learning é um tipo específico de aprendizado por reforço que atribui valores a pares de estado-ação. O estado do organismo é uma soma de todos os seus dados sensoriais, incluindo sua posição corporal, sua localização no ambiente, a atividade neural em sua cabeça, etc. Assim, em *Q-Learning*, isso significa que, para cada estado há um número de ações possíveis que poderiam ser tomadas, cada ação dentro de cada estado tem um valor de acordo com quantas ou quão poucas recompensas o organismo obterá por completar aquela ação.

Diante do contexto apresentado, nesta Unidade veremos o aprendizado Q que se originou no trabalho de Watkins (1989) e Watkins e Dayan (1992). Duas formas básicas de aprendizado Q serão apresentadas nesta Unidade: **Aprendizado Q ótimo** e **Aprendizado Q baseado em política**. Esses métodos podem ser vistos como um tipo de aproximação estocástica (BERSEKAS; TSITSIKLIS, 1996) para construir estimativas da função Q para qualquer política ótima ou não ótima.

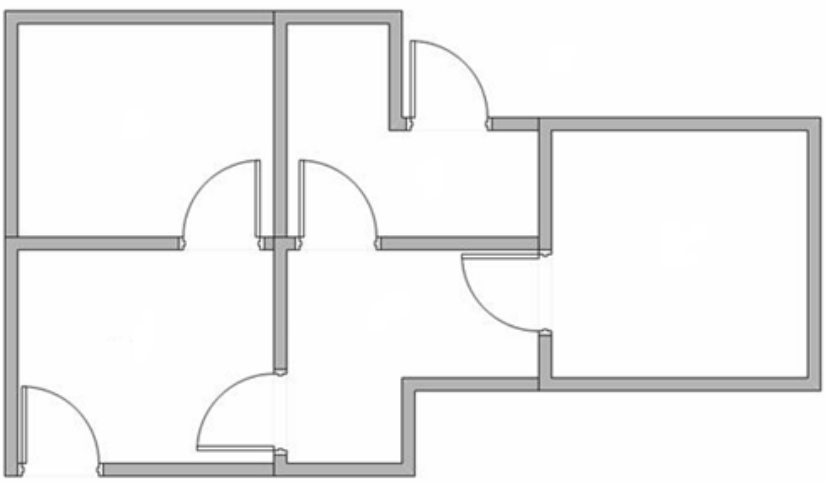
CAPÍTULO 1

APRENDIZADO Q

Vamos introduzir o conceito de *Q-Learning* por meio de um exemplo simples, porém abrangente (adaptado de McCULLOCK, 2012). O exemplo descreve um agente que usa treinamento não supervisionado para aprender sobre um ambiente desconhecido.

Suponha que, num prédio, 5 salas estejam conectadas por portas, conforme mostrado na Figura 34. Vamos numerar cada sala de 0 a 4. A parte externa do prédio pode ser considerada uma grande sala (indicado pelo número 5). Observe que as portas 1 e 4 levam a parte externa do prédio.

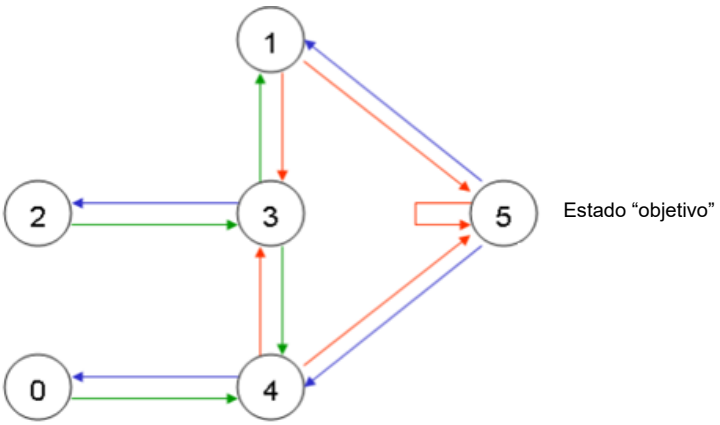
Figura 34. Representação de um prédio e suas salas.



Fonte: McCullock (2012).

Podemos representar as salas como um grafo, a Figura 35 mostra onde cada nó representa uma sala e cada porta é representada por um *link*.

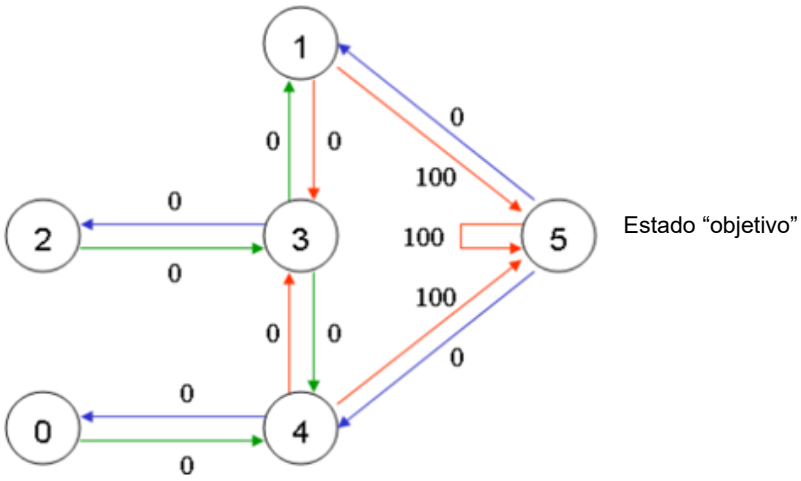
Figura 35. Nós e portas representados por um *link*.



Fonte: McCulloc, 2012.

Para este exemplo, gostaríamos de colocar um agente em qualquer sala e, a partir dela, sair do prédio. Ou seja, a sala de número 5 é o objetivo. Para definir essa sala como um objetivo, associamos um valor de recompensa a cada porta (o *link* entre os nós). As portas que levam imediatamente ao objetivo têm uma recompensa imediata de 100. Outras portas que não estão diretamente conectadas à sala-alvo têm zero como valor de recompensa. Como as portas são bidirecionais (0 leva a 4 e 4 leva de volta a 0), duas setas são atribuídas a cada sala. Cada seta contém um valor imediato de recompensa, conforme o grafo da Figura 36.

Figura 36. Grafo contendo um valor imediato de recompensa.



Fonte: McCullock, 2012.

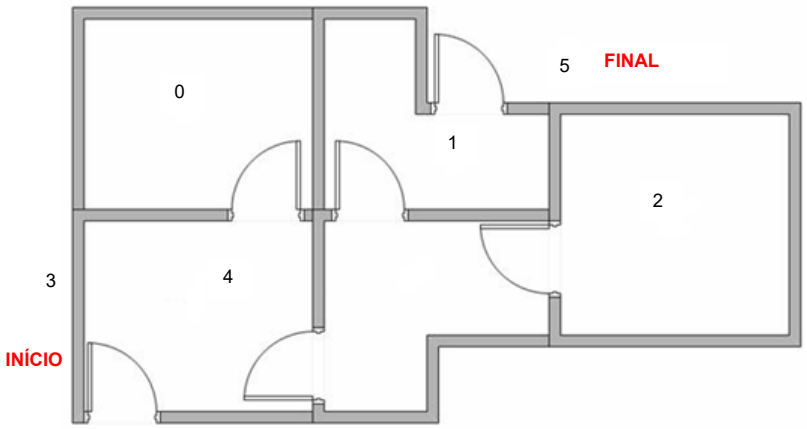
Como podemos ver, a sala 5 volta para si mesma com uma recompensa de 100, e todas as outras conexões diretas com a sala “objetivo” levam uma recompensa de 100. Na *Q-Learning*, o objetivo é atingir o estado com a maior recompensa, de modo que se o agente chega ao objetivo, ele permanecerá lá para sempre. Esse tipo de meta é chamado de *absorbing goal*.

Imagine o nosso agente como um robô virtual “burro” que pode aprender com a experiência. O agente pode passar de uma sala para outra, mas não tem conhecimento do ambiente e não sabe qual sequência de portas leva ao exterior.

Suponha, ainda, que queremos modelar algum tipo de evacuação simples de um agente, a partir de qualquer sala do prédio. Agora suponha que temos um agente na sala 2 e queremos que o agente aprenda a chegar até a parte externa do prédio (5). A Figura 37 apresenta essa situação.

A terminologia *Q-Learning* inclui os termos “estado” e “ação”.

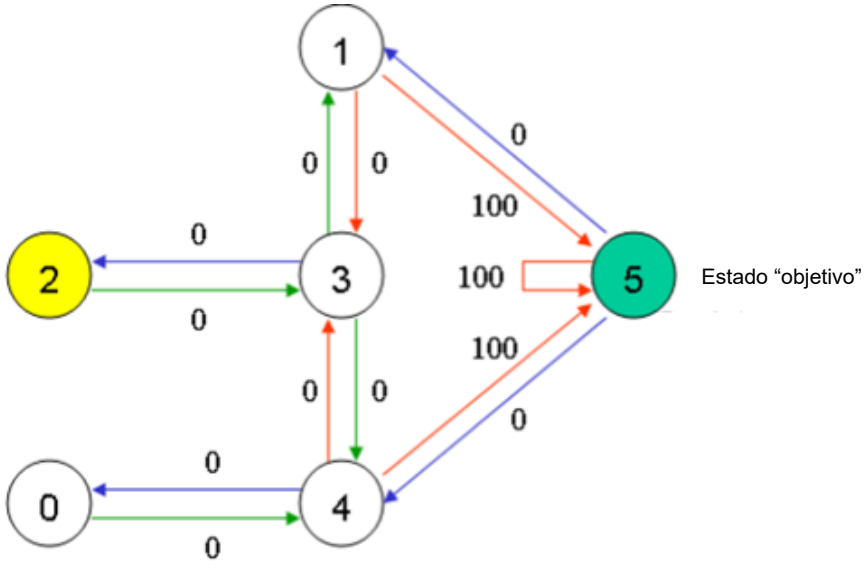
Figura 37. Agente saindo da sala 2 até a parte externa do prédio (5).



Fonte: McCullock, 2012.

Conforme pode ser visto na Figura 38, vamos ligar para cada sala, incluindo a parte externa, um "estado", e o movimento do agente de uma sala para outra será uma "ação". Em nosso grafo, um "estado" é representado como um nó, enquanto uma "ação" é representada pelas setas.

Figura 38. Ligando cada sala a estados "ação".



Fonte: McCullock, 2012.

Suponha que o agente esteja no estado 2. No estado 2, ele pode ir para o estado 3 porque o estado 2 está conectado ao 3. No estado 2, no entanto, o agente não pode ir diretamente para o estado 1 porque não há porta direta conectando a sala 1 e 2. Do estado 3, ele pode ir para o estado 1 ou 4 ou vice-versa (observe todas as setas sobre o estado 3). Se o agente estiver no estado 4, então as três ações possíveis devem ser: ir para o estado 0, 5 ou 3. Se o agente estiver no estado 1, ele pode ir para o estado 5 ou 3. No estado 0, ele só pode ir de volta ao estado 4.

Podemos colocar o diagrama de estados e os valores imediatos de recompensa numa tabela de recompensas, que chamaremos de "matriz R", conforme a Figura 39.

Figura 39. Matriz "R".

$$R = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

Fonte: McCullock, 2012.

Onde, os "-1s" representam os valores nulos (isto é, não há um *link* entre os nós). Por exemplo, o estado 0 não pode ir para o estado 1.

Agora vamos adicionar uma matriz similar Q que representa o cérebro do nosso agente, representando a memória do que o agente aprendeu pela experiência. As linhas da matriz Q representam o estado atual do agente e as colunas representam as possíveis ações que levam ao próximo estado (os *links* entre os nós).

O agente começa não sabendo nada, a matriz Q é inicializada em zero. Neste exemplo, assumiremos que o número de estados é conhecido (seis). Se não soubéssemos quantos estados estavam envolvidos, a matriz Q poderia começar com apenas um elemento. É uma tarefa simples adicionar mais colunas e linhas na matriz Q se um novo estado for encontrado.

A regra de transição do aprendizado Q é uma fórmula muito simples:

*Q(estado, ação) = R(estado, ação) + Gama * Máx [Q(próximo estado, todas as ações)]*

De acordo com essa fórmula, um valor atribuído a um elemento específico na matriz Q é igual à soma do valor correspondente na matriz R e ao parâmetro de aprendizado Gama, multiplicado pelo valor máximo de Q para todas as ações possíveis no próximo estado.

Nossa agente robô irá aprender pela experiência, sem um "professor" (isto é chamado de **aprendizado não supervisionado**). O agente irá explorar estado por estado até atingir o objetivo. Vamos chamar cada exploração de episódio. Cada episódio consiste no movimento do agente do estado inicial para o estado do objetivo. Cada vez que o agente chega ao estado objetivo, o programa vai para o próximo episódio.

1.1. Função Q

A função Q, ou função ação-valor, é definida por $Q^{\pi}: S \times A \rightarrow R$,

$$Q^{\pi}(s_t, a_t) = E_{s_{t+1}} \sim f(s_t, a_t)\left[r(s_t, a_t, s_{t+1}) + \gamma U^{\pi}(s_{t+1})\right]$$

A partir dessa definição e a da equação de Bellman, observa-se que $Q^{\pi}(s_t, \pi(s_t)) = U^{\pi}(s_t)$, e, portanto, a função Q^{π} pode ser expressa na forma de Bellman como:

$$Q^{\pi}(s_t, a_t) = E_{s_{t+1}} \sim f(s_t, a_t)\left[r(s_t, a_t, s_{t+1}) + \gamma Q^{\pi}(s_{t+1}, \pi(s_{t+1}))\right]$$

Assim, a função Q_L ótima deve satisfazer:

$$\begin{aligned} Q(s_t, a_t) = \quad & E_{s_{t+1}} \sim f(s_t, a_t)\left[r(s_t, a_t, s_{t+1}) + \gamma U(s_{t+1})\right] \\ & E_{s_{t+1}} \sim f(s_t, a_t)\left[r(s_t, a_t, s_{t+1}) + \gamma Q(s_{t+1}, \pi(s_{t+1}))\right] \end{aligned}$$

Desta forma, uma equação da otimalidade de Bellman para funções *Q* é dada por:

$$Q(s_t, a_t) = E_{s_{t+1}} \sim f(s_t, a_t)\left[r(s_t, a_t, s_{t+1}) + \gamma \max_{\alpha \in U} Q(s_{t+1}, a)\right]$$

Essa equação caracteriza Q, e declara que o valor ótimo de uma ação a_t aplicada no estado s_t é o valor esperado da soma da recompensa imediata com o valor ótimo descontado obtido pela melhor ação no estado seguinte. Consequentemente, uma política ótima πpode ser determinada a partir de Q por:

$$\pi(s_t) = \operatorname{argmax}_a Q(s_t, a)$$

Uma vez que a função Q depende também da ação, ela já inclui informação sobre a qualidade das transições. Por outro lado, a função valor de estado U somente descreve a qualidade dos estados. Nesse caso, para inferir a qualidade das transições, essas devem ser explicitamente levadas em consideração. Consequentemente, um modelo do MDP é necessário na forma da dinâmica f e da função de utilidade r, enquanto que na formulação usando função Q, o problema de decisão markoviano é tratado sem referência a tais modelos.

1.1.1. Aprendizado Q ótimo

O aprendizado Q é um método de política-off, o que significa que a função valor ótima Q é estimada, independentemente da política atual (exploração) que está sendo utilizada para gerar as trajetórias amostradas. A regra de aprendizado Q ótimo pode ser expressa como:

$$Q_{t+1}(s_t, a_t) + a_t\left[r(s_t, a_t, s_{t+1}) + \gamma \max_{a \in U} Q_t(s_{t+1}, a) - Q_t(s_t, a_t)\right]$$

sendo Q_{t+1} a t-ésima estimativa para Q e α_t os tamanhos de passos (stepsizes).

Admitindo-se uma representação tabular da função Q, aprendizado Q converge com probabilidade 1 para o valor ótimo Q quando $t \to \infty$, sob as seguintes suposições, (WATKINS; DAYAN, 1992):

1. uma sequência decrescente apropriada de tamanhos dos passos α_t que satisfaz as condições $\sum_{t=0}^{\infty} \alpha_t = \infty$ e $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$, e
2. todos os pares estado-ação são visitados, assintoticamente, infinitas vezes.

Por exemplo, uma sequência padrão de tamanhos de passos que garante a convergência do aprendizado Q é dada por:

$$\alpha_t = \frac{\tau}{\eta + t}, t = 1, 2, \ldots,$$

sendo τ e η números positivos.

A condição 2 pode ser satisfeita se o agente de aprendizado selecionar com probabilidade não nula uma ação aleatória em cada estado visitado (exploração) e, além disso, exploitar seu conhecimento atual para obter um bom desempenho, por exemplo, selecionando ações gulosas com respeito à sua função Q atual (exploitação). Uma abordagem característica que se relaciona com exploração-exploitação em algoritmos de aprendizado por reforço é chamada exploração gulosa ε (SUTTON; BARTO, 1998), a qual seleciona ações de acordo com a seguinte regra:

$\alpha_t = a\ \varepsilon\ \text{argmax}_a Q_t(s_t, a)$, com probabilidade $1 - \varepsilon_t$ uma ação uniformemente aleatória em U (s_t), com probabilidade ε_t sendo $\varepsilon \in (0, 1)$ a probabilidade de exploração no passo de tempo t.

1.1.2. Aprendizado baseado em política

O aprendizado Q baseado em política tenta aprender Q^{π}, a função Q para alguma política projetada π. A política π pode ser ou não a política que de fato está sendo seguida durante o treinamento. A regra de aprendizado baseada em política é dada por:

$$Q_{\text{uuuuuuuu}}(s, a) + a\left[r(s, a, s_+) + \gamma Q\left(s_+, \pi(s_+)\right) - Q(s, a)\right]$$

Sendo Q_t a t-ésima estimativa para Q^{π}.

Pouco depois dos trabalhos de Watkins e Dayan (1992) e Watkins (1989), Peng e Williams (1996) apresentaram um método de aprendizado Q multipassos incremental Q(λ), uma combinação de aprendizado Q a um passo e TD(λ). Sutton e Barto (1998) fizeram uma

comparação entre aprendizado Q multipassos, Q(λ) e o método sugerido por Watkins (1989). Similarmente, Kuzmin (2002) sugeriu uma modificação ao aprendizado Q tabular tradicional. Ele propõe um perceptron multicamadas para atuar como um aproximador de avaliação de aprendizado Q. Ele denominou essa combinação de "conexionismo". Antes do trabalho dele, um esquema similar para aproximação de função valor de estado foi proposto por Pipe (1998), onde o método tabular tradicional foi substituído por uma aproximação de função via uma função de base radical. No esquema proposto, ele denominou a função valor de estado como um campo potencial. Deng e Er (2004) propuseram o aprendizado Q Fuzzy. Neste esquema, o aprendizado Q é usado para gerar e ajustar automaticamente regras fuzzy para uma tarefa de navegação de robô móvel. Bertsekas e Yu (2010) propuseram aprendizado Q e melhoraram a iteração de política em programação dinâmica com desconto.

No *Q-Learning*, os valores Q são armazenados e atualizados para cada par de ação de estado. Nos casos em que o número de pares de ação do estado é muito grande, a implementação pode ser impraticável. Os sistemas de inferência *fuzzy* têm a vantagem de alcançar boas aproximações na função Q e, simultaneamente, possibilitam o uso do Q-*Learning* em problemas contínuos de espaço de estados (*Fuzzy Q-Learning*).

Em *fuzzy Q-Learning*, x é o conjunto nítido das entradas que definem o estado do agente. Esses são convertidos em valores *fuzzy* e cada regra *fuzzy* corresponde a um estado. Em outras palavras, a força de acionamento de cada regra define o grau em que o agente está em um estado. Além disso, as regras não têm consequências fixas; ou seja, não há pares de ação de estado fixos (predefinidos), mas, através do algoritmo de exploração/exploitação, surgem os consequentes de cada regra (par de ação de estado).

Texto retirado do artigo de Kofinas e Dounis (2018).

CAPÍTULO 2

ALGORITMO *Q-LEARNING*

Os passos do algoritmo Q-*Learning* podem ser assim:

1. defina o parâmetro gama e as recompensas do ambiente na matriz R;
2. inicialize a matriz Q com zero;
3. para cada episódio:
    a. selecione um estado inicial aleatório;
    b. enquanto o estado objetivo não for alcançado:
        I. selecione uma entre todas as ações possíveis para o estado atual;
        II. usando essa ação possível, considere ir para o próximo estado;
        III. obtenha o valor Q máximo para este próximo estado com base em todas as ações possíveis;
        IV. calcule: *Q(estado, ação) = R(estado, ação) + Gama * Máx [Q(próximo estado, todas as ações)];*
        V. defina o próximo estado com o estado atual.

Voltando ao exemplo dado no capítulo 1 desta Unidade, podemos constatar que o algoritmo definido anteriormente é empregado pelo agente para aprender com a experiência. Cada episódio é equivalente a uma sessão de treinamento. Em cada sessão de treinamento, o agente explora o ambiente (representado pela matriz R), recebe a recompensa (se houver) até atingir o estado objetivo. O objetivo do treinamento é melhorar o "cérebro" do nosso agente, representado pela matriz Q. Mais treinamento resulta em uma matriz mais otimizada Q. Neste caso, se a matriz Q for aprimorada, em vez de explorar ao redor, e indo para trás e para as mesmas salas, o agente encontrará a rota mais rápida para o estado objetivo.

O parâmetro gama tem um intervalo de 0 a 1. Se o gama estiver mais próximo de zero, o agente tenderá a considerar apenas recompensas imediatas. Se o gama estiver mais próximo de um, o agente considerará recompensas futuras com maior peso, disposto a atrasar a recompensa.

Para usar a matriz Q, o agente simplesmente rastreia a sequência de estados, do estado inicial ao estado objetivo. O algoritmo localiza as ações com os maiores valores de recompensa registrados na matriz Q para o estado atual.

Os passos do algoritmo para utilizar a matriz Q são:

1. defina o estado atual = estado inicial;
2. a partir do estado atual, localize a ação com maior valor Q;
3. defina o estado atual = próximo estado;
4. repita as etapas 2 e 3 até o próximo estado atual = estado objetivo.

O algoritmo acima retornará a sequência de estados do estado inicial para o estado objetivo.

Para entender como o algoritmo Q-*Learning* funciona, passaremos por alguns episódios passo a passo. Começaremos definindo o valor do parâmetro de aprendizado gama = 0.8 e o estado inicial como Sala 1.

Inicialize a matriz Q como uma matriz zero, conforme a Figura 40:

Figura 40. Inicialização da Matriz Q.

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Fonte: McCullock, 2012.

Observe a segunda linha (estado 1) da matriz R, conforme a Figura 41. Existem duas ações possíveis para o estado atual 1: vá para o estado 3 ou vá para o estado 5. Por seleção aleatória, selecionamos para ir para o estado 5 como nossa ação.

Figura 41. Matriz R.

$$R = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

Fonte: McCullock, 2012.

Agora vamos imaginar o que aconteceria se nosso agente estivesse no estado 5. Olhe para a sexta linha da matriz de recompensa R (estado 5). Temos três ações possíveis: vá para o estado 1, 4 ou 5.

*Q(estado, ação) = R(estado, ação) + Gama * Máx [Q(próximo estado, todas as ações)]*

$$Q(1, 5) = R(1, 5) + 0.8 * \text{Máx}\,[Q(5, 1), Q(5, 4), Q(5, 5)] = 100 + 0.8 * 0 = 100$$

Como a matriz Q ainda é inicializada com zeros, Q(5, 1), Q(5, 4), Q(5, 5) são todos zero. O resultado deste cálculo para Q(1, 5) é 100 por causa da recompensa imediata de R(1, 5).

O próximo estado, 5, agora se torna o estado atual. Como 5 é o estado objetivo, terminamos um episódio. O cérebro do nosso agente agora contém uma matriz atualizada Q, conforme a Figura 42.

Figura 42. Matriz atualizada Q.

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Fonte: McCullock, 2012.

Para o próximo episódio, começamos com um estado inicial escolhido aleatoriamente. Desta vez, temos o estado 3 como nosso estado inicial.

Olhe para a quarta linha da matriz R; o agente tem 3 ações possíveis: vá para o estado 1, 2 ou 4. Por seleção aleatória, escolhemos ir para o estado 1 como nossa ação.

Agora imaginamos que estamos no estado 1. Veja a segunda linha da matriz de recompensa R (o estado 1). Ela tem duas ações possíveis: vá para o estado 3 ou para o estado 5. Então, calculamos o valor Q:

*Q(estado, ação) = R(estado, ação) + Gama * Máx [Q(próximo estado, todas as ações)]*

$$Q(1, 5) = R(1, 5) + 0.8 * \text{Máx}\,[Q(1, 2), Q(1, 5)] = 0 + 0.8 * \text{Máx}(0, 100) = 80$$

Usamos a matriz atualizada Q do último episódio. Q (1, 3) = 0 e Q (1, 5) = 100. O resultado do cálculo é Q (3, 1) = 80 porque a recompensa é zero. Como pode ser visto na Figura 43, a matriz Q se torna:

Figura 43. Matriz Q atualizada.

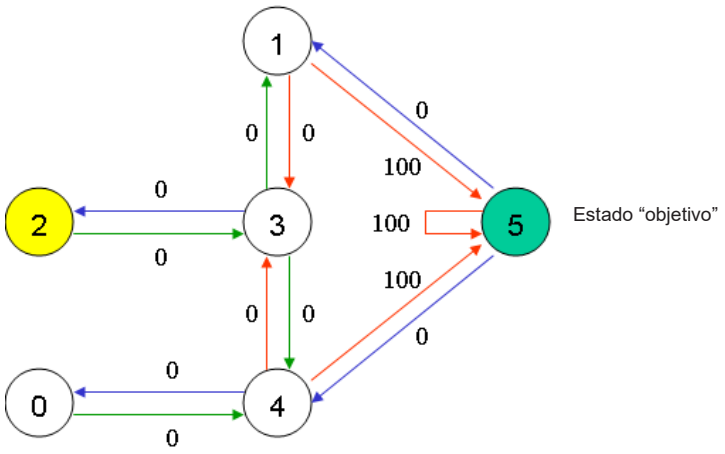
$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Fonte: McCullock, 2012.

O próximo estado, 1, agora se torna o estado atual. Repetimos o *loop* interno do algoritmo de aprendizado Q porque o estado 1 não é o estado do objetivo.

Assim, iniciando o novo *loop* com o estado atual 1, há duas ações possíveis: vá para o estado 3 ou vá para o estado 5. Por seleção aleatória, nossa ação selecionada é 5, como pode ser visto na Figura 44.

Figura 44. Novo *loop* com estado atual 1.



Fonte: McCullock, 2012.

Agora que estamos no estado 5, há três ações possíveis: vá para o estado 1, 4 ou 5. Calculamos o valor Q usando o valor máximo dessas ações possíveis.

*Q(estado, ação) = R(estado, ação) + Gama * Máx [Q(próximo estado, todas as ações)]*

$$Q(1, 5) = R(1, 5) + 0.8 * \text{Máx}\,[Q(5, 1), Q(5, 4), Q(5, 5)] = 100 + 0.8 * 0 = 100$$

As entradas atualizadas da matriz Q, Q (5, 1), Q (5, 4), Q (5, 5), são todas zero. O resultado desse cálculo para Q (1, 5) é 100 por causa da recompensa instantânea de R (5, 1). Este resultado não altera a matriz Q.

Como 5 é o estado objetivo, terminamos este episódio. O cérebro do nosso agente agora contém a matriz Q atualizada conforme a Figura 45.

Figura 45. Matriz Q atualizada.

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Fonte: McCullock (2012).

Se o nosso agente aprender mais com mais episódios, ele finalmente alcançará os valores de convergência na matriz Q, como mostra a Figura 46.

Figura 46. Convergência na matriz Q.

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 500 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{bmatrix} \end{array}$$

Fonte: McCullock, 2012.

Esta matriz Q pode, então, ser normalizada (isto é, convertida em porcentagem) dividindo todas as entradas diferentes de zero pelo número mais alto (500 neste caso), conforme a Figura 47.

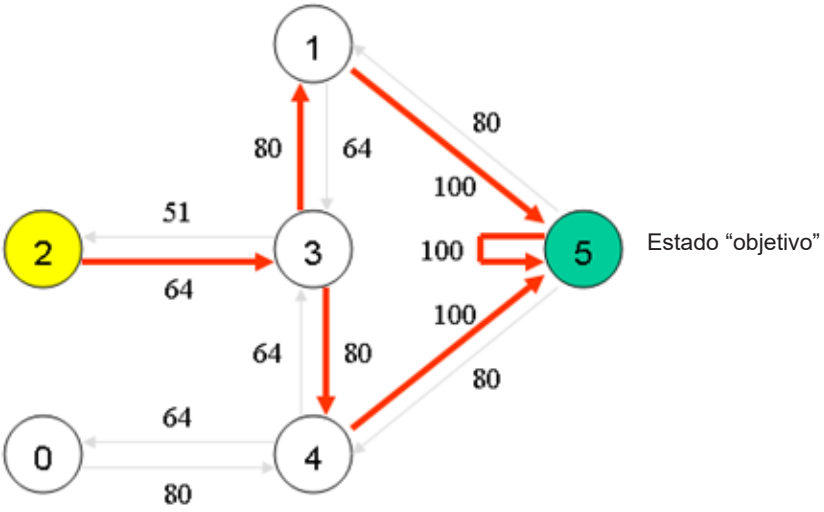
Figura 47. Matriz Q normalizada.

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 64 & 0 & 100 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 80 & 51 & 0 & 80 & 0 \\ 64 & 0 & 0 & 64 & 0 & 100 \\ 0 & 80 & 0 & 0 & 80 & 100 \end{bmatrix} \end{array}$$

Fonte: McCullock, 2012.

Quando a matriz Q se aproximar o suficiente de um estado de convergência, sabemos que nosso agente aprendeu os melhores caminhos para o estado objetivo. Rastrear as melhores sequências de estados é tão simples quanto seguir os *links* com os valores mais altos em cada estado, conforme a Figura 48.

Figura 48. Rastreando as melhores sequências de estados.



Fonte: McCullock, 2012.

Por exemplo, do estado inicial 2, o agente pode usar a matriz Q como um guia:

- do estado 2, os valores máximos de Q sugerem a ação para ir para o estado 3;
- do estado 3, os valores máximos de Q sugerem duas alternativas: vá para o estado 1 ou 4. Suponha que escolhemos arbitrariamente ir para 1;
- do estado 1, os valores máximos de Q sugerem a ação para ir para o estado 5.

Assim, a sequência é 2 - 3 - 1 - 5.

2.1. Detalhes do algoritmo *Q-Learning*

Watkins (1989) definiu o algoritmo Q-*Learning*. O Quadro 6 apresenta uma versão adaptada do seu algoritmo original.

Quadro 6. Algoritmo *Q-Learning*.

```
Algoritmo Q-Learning

procedure QLearning(r, , , )
Inicialize Q(s, a)
        repeat
                Inicialize s
                repeat
                        Selecione  de acordo com a política -gulosa
                        Observe os valores de r e s'
                        Q(s, a) (1 - ) Q(s, a) + (r + (Q(s', a)))
                        s s'
                until Encontrar um estado final
        until Atingir N  p episódios
        return Q(s, a)                          {Matriz dos Q-valores}
end procedure
```

Fonte: Elaborado pelo autor.

Apesar de o critério de convergência do algoritmo Q-Learning exigir a execução de um número infinito de episódios, na prática executa-se um número suficientemente grande de episódios (configura-se o número de episódios NE_{ps}, de acordo com o tamanho ou contexto da tarefa a ser aprendida).

2.2. Políticas de seleção de ações para o algoritmo *Q-Learning*

No processo de resolução de um problema de aprendizado por reforço, uma política de seleção de ações tem como objetivo estabelecer o comportamento do agente aprendiz para que ele alterne adequadamente entre o uso do conhecimento já adquirido e a aquisição de novo conhecimento, de forma a otimizar o processo de exploração/exploitação do espaço de busca.

A ideia de experimentar mais de uma política de seleção de ações para o Q-*Learning* tem como meta verificar qual dessas políticas é mais adequada para ser utilizada na implementação dos métodos híbridos propostos.

2.3. Política ε- gulosa

A política ε-gulosa escolhe a ação que tem o maior valor esperado, com probabilidade definida por (1 - ε), e de ação aleatória, com probabilidade ε. Matematicamente, dada a matriz Q-valores Q obtém-se a ação gulosa a para um estado s fazendo:

$$a = \max_{a \in A(s)} Q(s, a)$$

$$\pi(s, a) = 1 - \varepsilon + \frac{\varepsilon}{A(s) \vee}$$

$$\pi(s, a) = \frac{\varepsilon}{A(s) \vee \forall_{a \in A(s)}} - \{a\}$$

Onde |A(s)| corresponde ao número de ações possíveis de serem executadas a partir de s, e ε é o parâmetro de controle entre gula e aleatoriedade. A restrição presente permite que o Q-Learning explore o espaço de estados do problema, e é uma das condições necessárias para que ele encontre uma política de controle ótima.

2.3.1. 2.3.1 Política ε- gulosa adaptativa

A política ε -gulosa adaptativa é semelhante à política ε-gulosa descrita anteriormente, ou seja, ela permite escolher a ação que tem o maior valor esperado, com probabilidade definida por (1 - ε), e ação aleatória, com probabilidade ε. O que diferencia, e também justifica o termo "adaptativa" é que o valor de ε sofre um decaimento exponencial calculado por:

$$\varepsilon = \max\left\{v_i, v_f \,.\, b^k\right\}$$

Onde k é o contador de episódios do Q-Learning, b é um valor próximo de 1 e $v_1 < v_f$ ε [0, 1]. Dessa forma, inicialmente o algoritmo utilizará valores grandes ε (próximos a v_f) e à medida que o valor de k cresce a escolha de ε é direcionada para valores menores (mais próximos a v_i). A ideia é permitir que inicialmente sejam feitas escolhas mais aleatórias e, à medida que o número de episódios aumente, o aspecto guloso seja mais explorado.

2.4. Política baseada na contagem de visitas

Nesta política a escolha de ações é feita baseada em uma técnica denominada Comparação de Reforço (*Reinforcement Comparison*) (SUTTON; BARTO, 1998). Nesta técnica, a escolha das ações é feita com base no princípio de que ações seguidas de grandes recompensas devem ser preferidas em detrimento de ações seguidas de pequenas recompensas. Para

definir o que significa uma "grande recompensa" é feita uma comparação com um nível de recompensa padrão denominado **recompensa referencial**.

A ideia da política aqui proposta é semelhante à técnica de comparação de reforço; entretanto, a escolha das ações preferidas é feita com base na contagem de visitas aos estados atingidos por tais ações, ou seja, os estados mais visitados indicarão as ações preferidas, em detrimento de ações que levam a estados com menor número de visitas. De posse dessa medida de preferência das ações, pode-se determinar a probabilidade de seleção das ações de acordo com a seguinte relação:

$$\pi_t(a) = P_r\{a_t = a\} = \frac{e^{p_{t-1}(a)}}{\sum_b e^{p_{t-1}(b)}}$$

Onde $\pi_t(a)$ denota a probabilidade de se escolher a ação a no passo t e $p_t(a)$ denota a preferência da ação a no tempo t, que é calculada por:

$$p_{t+1}(a_t) = p_t(a_t) + \beta\left(N_v(s, a_t)/NE_p\right)$$

Onde s é o estado atingido em consequência da escolha da ação a no passo t, $N_v(s, a_t)$ é o número de visitas ao estado s, NE_p é o número total de episódios e $\beta \in [0, 1]$ é um parâmetro de controle que pondera o nível de influência das ações preferenciais.

2.5. Determinação da função de recompensa

No aprendizado por reforço um agente aprendiz tem como objetivo maximizar o total de recompensa recebida ao longo do processo. Para atingir este objetivo, o agente necessita mensurar quão bom é escolher determinada ação a partir do estado corrente por meio de um valor numérico, isso é feito pela função de recompensa. Assim, para cada problema específico é necessário definir a função de recompensa. Neste item serão discutidos os detalhes da função de recompensa utilizado no algoritmo *Q-Learning* para os métodos propostos.

A função de recompensa dada pela equação:

$$P^a_{ss'} = P_r\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t\}$$

Estabelece de forma trivial a recompensa imediata como sendo inversamente proporcional à distância de um estado de partida c_i à cidade de destino c_j. Esta forma de estabelecer a recompensa imediata apresenta a inconveniência de, em casos de valores de distâncias muito próximos, ocorrer casos de empate na escolha dos Q-valores Q(s,a). Uma forma

de resolver este inconveniente é ponderar o inverso da distância por algum valor que permita distinguir os casos de empate.

O cálculo da função de recompensa aqui proposto faz a ponderação do inverso da distância entre dois estados distintos utilizando a contagem de visitas aos estados do ambiente. Dessa forma, as ações que levam a estados mais frequentemente visitados serão premiadas com recompensa imediata maior. Isso pode ser feito utilizando a seguinte equação:

$$R(s,a) = \frac{1}{d_{ij}} * N_v(s,a)$$

Onde $\frac{1}{d_{ij}}$ é o inverso da distância entre os estados c_i e c_j (sendo o estado c_i representado pelo estado s, e o estado c_j representado pelo estado a ser atingido em consequência da escolha da ação a) e $N_v(s, a)$ o número de visitas ao estado corrente.

2.6. Q-Learning λ (eligibility traces)

A base teórica do conceito de *eligibility traces* (ou traços de elegibilidade) foi inicialmente apresentada por Klopf (1972) no âmbito de um trabalho sobre sistemas adaptativos para a USAF (*United States Air Force*). No entanto, a integração desse conceito na problemática de aprendizado por reforço somente foi realizada em 1989 por Watkins (1989) ao incorporar este no mecanismo de aprendizado *Q-Learning*.

O conceito de *elegibility traces* consiste no registro temporário da ocorrência de determinado evento, quer seja a passagem por um estado ou a seleção de uma determinada ação. Este registro permite a sinalização de quaisquer parâmetros associados ao evento como elegíveis para posteriormente poderem sofrer alterações na sua definição. Assim, quando da ocorrência de um erro, este poderá ter a sua origem relacionada com os estados e/ou ações sinalizadas.

Este conceito, ao permitir encurtar a separação entre eventos e a informação apreendida, faz emergir a possibilidade da existência de métodos intermediários ou intercalares (SUTTON; BARTO, 1998).

A definição para o mecanismo de aprendizado Q-Learning λ (elegibility traces) definida por Watkins (1989) é a apresentada no Quadro 7:

Quadro 7. Algoritmo Q-Learning λ.

1.Inicialize ***Q(s, a)*** arbitrariamente e ***e(s, a) = 0***; para todo ***s, a***
2. Repita (para cada episódio)
   a) Inicialize ***s***
   b) Escolha ***a*** a partir de ***s*** usando a política derivada de Q
   c) Repita (para cada episódio)
   d)
      I. Tome uma ação ***a***, e observe ***r***, ***s'***
      II. Escolha ***a'*** a partir de ***s'*** usando a política derivada de Q
      III. $\delta \leftarrow r + \gamma Q(s', a\ ) - Q(s, a)$
      IV. $e(s, a) \leftarrow e(s, a) + 1$
      V. Para todo ***s***, ***a:***
         1. $Q(s, a) \leftarrow Q(s, a) + \alpha\delta e(s, a)$
         2. Se $a' = a$ então
            a. $e(s, a) \leftarrow \gamma\lambda e(s, a)$
         3. else
            a. $e(s, a) \leftarrow 0$
      VI. $s \leftarrow s'; a \leftarrow a'$
   e) até o terminal ***s***

Fonte: Watkins, 1989.

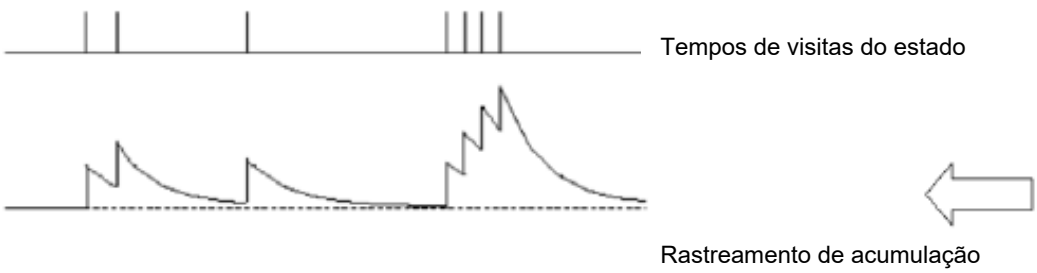
Este algoritmo possui associado um parâmetro de fator de rastreamento $\lambda \in [0, 1]$ que tem uma relação inversa com a taxa de decaimento de sinalização do evento. Assim, quanto maior o seu valor, menor será a velocidade com que a sinalização do evento decai ao longo dos diversos passos de evolução do algoritmo.

Embora os eventos sejam sinalizados por uma função de ação-valor auxiliar *e(s, a)*, essa sinalização é automaticamente incorporada na estrutura única de definição do ambiente (a função Q), permitindo, assim, ser exercida uma influência transparente sobre todo o mecanismo base de aprendizado.

Sendo este algoritmo a aplicação de um conceito sobre o mecanismo de aprendizado *Q-Learning*, é natural que a política associada a ele coincida com a do mecanismo em foco. Assim, em cada estado *s* existente, somente são sinalizados os eventos referentes à ação *a* que nesse estado se encontre mais valorizada, sendo as possíveis ações restantes mantidas sem qualquer sinalização.

A sinalização de um evento (seleção de determinada ação *a* num determinado estado *s*) é realizada de forma incremental e unitária, o que significa que eventos da mesma natureza que ocorram próximos no tempo irão incrementar cumulativamente a sua sinalização, tal como ilustrado na Figura 49:

Figura 49. Sinalização de eventos de forma acumulada.



Fonte: Watkins, 1989.

2.7. Q-Learning λ (replacing)

Sutton e Satinder (1996) apresentaram uma alternativa ao conceito de *eligibility traces*. Devido à sinalização de um evento no conceito original ser realizada de forma incremental (ou seja, acumulativa), constatou-se que para eventos da mesma natureza que ocorressem próximos no tempo, seria possível que estes ficassem sinalizados de forma excessiva (valores superiores a 1), e, por consequência, avaliados pelo processo de aprendizado.

Dessa forma, sugere-se como alternativa a substituição do processo de sinalização de eventos pela atribuição de um valor unitário único, não acumulável, demonstrando ainda que essa alteração se manifesta mais rápido e origina informação de maior qualidade durante o processo de aprendizado.

Assim, a definição desse mecanismo de aprendizado é idêntica à do mecanismo *eligibility traces*, apresentando uma sutil, porém relevante, diferença, ao afetar com um valor unitário não acumulável a sinalização de um evento.

A definição para o mecanismo de aprendizado Q-Learning λ(replacing) definida por Sutton e Satinder (1996) pode ser vista no Quadro 8.

Quadro 8. Algoritmo Q-Learning λ (replacing).

1. Inicialize ***Q(s, a)*** arbitrariamente e ***e(s, a)* = 0**; para todo ***s, a***
2. Repita (para cada episódio)

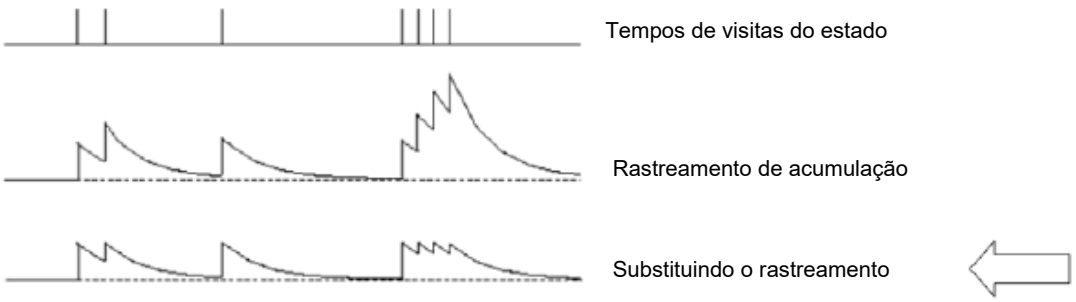
a) Inicialize ***s***
b) Escolha ***a*** a partir de ***s*** usando a política derivada de Q
c) Repita (para cada episódio)

I. Tome uma ação ***a***, e observe ***r***, ***s'***
II. Escolha ***a'*** a partir de ***s'*** usando a política derivada de Q
III. $\delta \leftarrow r + \gamma Q(s', a) - Q(s, a)$
IV. $e(s, a) \leftarrow 1$
V. Para todo ***s***, ***a:***
(1) $Q(s, a) \leftarrow Q(s, a) + \alpha\delta e(s, a)$
(2) Se $a' = a$ então
(a) $e(s, a) \leftarrow \gamma\lambda e(s, a)$
(3) else
(a) $e(s, a) \leftarrow 0$
VI. $s \leftarrow s'; a \leftarrow a'$
d) até o terminal ***s***

Fonte: Sutton e Satinder, 1996.

Podemos notar que, sendo a sinalização de um evento (seleção de determinada ação *a* num determinado estado *s*) realizada de forma unitária e não incremental, eventos da mesma natureza que ocorram próximos no tempo serão sempre sinalizados com um valor máximo e não acumulativo com sinalizações anteriores, tal como ilustrado na Figura 50:

Figura 50. Sinalização de eventos de forma unitária.



Fonte: Sutton e Satinder, 1996.

CAPÍTULO 3

APLICAÇÕES *Q-LEARNING*

3.1. Robô humanoide

Exemplo retirado do artigo de Silva e Gouvêa (2015).

Este exemplo apresenta um robô humanoide que aprende a andar. Os motores do humanoide podem assumir três velocidades: 0, 30 e 50 unidades de potência. Os motores são acionados de forma intercalada, por 1 segundo. As ações possíveis são:

» não alterar a potência do motor;

» diminuir ou aumentar a potência.

Quando o robô humanoide identifica um obstáculo, simulando uma presa, o robô começa a andar. O crítico (no aprendizado por reforço, o agente executa uma ação que é avaliada por um crítico, que dá ao agente um sinal de reforço positivo ou negativo que é utilizado para se autoajustar e, assim, aperfeiçoar suas ações) avalia a ação em função do deslocamento do robô em relação ao alvo por um período de tempo.

» se o robô se desloca até um limite x_1, a ação é punida;

» se o deslocamento é de x_1 até x_2, a ação é punida com menor severidade;

» finalmente, se o robô se desloca a uma distância maior que x_2, a ação é recompensada.

A ideia é que se a ação não for boa o suficiente, ela não produzirá um deslocamento do robô em relação ao alvo e, assim, será punida.

A tabela Q-Learning tem dimensão N_e x N_a, sendo N_e o número de estados e N_a o número de ações. Na descrição do exemplo foi dado que há dois motores assumindo três estados e cada motor pode executar três ações. Assim, $N_e = 3^2 = 9$ e $N_a = 3^2 = 9$. A Tabela 4 mostra a tabela Q-Learning antes e depois do treinamento on-line, isto é, durante o tempo em que o humanoide aprende a andar. Sabe-se, previamente, que o Q(s, a) é ótimo nas velocidades (50, 50), pois são máximas e não desestabilizam o humanoide e ação (0, 0), pois não alteram a velocidade máxima. Trata-se, portanto, da posição Q(9, 5). A Tabela 4, que tem seus valores iniciais aleatórios no intervalo [0, 1], tem sua posição Q(9, 5) maximizada em relação aos demais valores depois de aproximadamente 20 minutos de treinamento por reforço, o que demonstra o sucesso da abordagem

Tabela 4. Tabela *Q-Learning* antes e depois do treinamento.

Matriz antes de aprender								
0,95	0,01	0,48	0,36	0,49	0,79	0,4	0,18	0,79
0,07	0,71	0,11	0,56	0,7	0,66	0,64	0,88	0,17
0,62	0,37	0,88	0,69	0,95	0,03	0,08	0,14	0,45
0,35	0,14	0,39	0,89	0,75	0,57	0,44	0,78	0,4
0,79	0,12	0,92	0,91	0,43	0,44	0,14	0,01	0,02
0,25	0,91	0,75	0,39	0,08	0,01	0,85	0,65	0,69
0,3	0,34	0,73	0,77	0,61	0,98	0,44	0,46	0,62
0,19	0,02	0,28	0,3	0,9	0,63	0,65	0,17	1
0,17	0,59	0,03	0,01	0,84	0,23	0,75	0,16	0,21
Matriz depois de aprender								
0,48	-0,7	0,42	-0,65	-0,57	0,023	-0,183	0,03	0,32
-0,01	0,46	-0,92	-0,66	0,37	0,6	0,55	0,07	-0,26
-0,69	0,3	0,83	0,22	0,42	-0,21	0,07	-0,2	-0,47
-0,67	-0,535	-0,48	0,37	0,01	0,14	-0,03	0,56	0,19
0,03	-0,78	0,52	0,8	-0,88	-0,78	-0,59	-0,14	0,1
-0,528	0,3	0,11	-0,88	-0,53	-0,03	0,52	0,07	0,23
-0,84	0,13	0,67	0,76	0,523	0,07	-0,52	-0,69	0,012
0,1	-0,33	-0,304	0,28	0,56	0,26	0,64	-0,34	0,1
-0,36	0,21	-0,13	-0,2	8,424	-0,46	0,24	-0,391	-0,368

Fonte: Silva e Gouvêa, 2015.

3.2. Um Conto de Natal

Exemplo retirado do artigo de Cabral (2018).

Nosso personagem, Jonas, é motorista de táxi e gosta de trabalhar de madrugada (sim, ele mora em uma cidade pouco violenta). Em muitos momentos, quando está sem passageiros, ele estuda sobre inteligência artificial e coisas relacionadas. Seu objetivo é conseguir um emprego nessa área. Nos últimos dias ele se interessou por uma técnica chamada Q-*Learning*, um aprendizado por reforço, e escreveu alguns códigos no tempo livre sobre como ensinar um robô virtual a encontrar a saída de um labirinto com armadilhas.

Em determinado dia, por acaso a madrugada do dia 24 para 25 de dezembro, ele estava em seu carro estudando e esperando passageiros (sim, ele não tem família). Como era quase manhã, Jonas já pensava em ir para sua casa dormir no momento que viu um senhor correndo em direção ao seu carro. O homem era barrigudo, com uma grande barba branca, vestia roupas vermelhas e carregava um embrulho em suas mãos. Este abriu a porta do táxi, entrou e, quase sem fôlego, gritou: Preciso de sua ajuda jovem, estamos quase sem tempo. Me leve a este local o mais rápido possível, VAMOS!

Jonas não queria acreditar que o Papai Noel havia aparecido diante dele e pedido sua ajuda, mas, de qualquer maneira, era um cliente que precisava de seus serviços. Ficou claro que deveria chegar ao destino antes que o sol nascesse. O problema é que ele conhecia vários caminhos possíveis até o destino final e sua pouca experiência não permitia a ele saber qual o mais rápido (neste universo não existe *google maps*). Como, então, ele iria descobrir qual o menor caminho até lá (neste caso, o menor caminho é também o mais rápido)? Eis que então surgiu uma expressão de esperança no rosto de nosso herói quando ele se lembrou do assunto que vinha estudando. Rapidamente pegou seu *notebook* no banco de trás do carro, abriu e começou a digitar algumas coisas.

Alguns segundos foram suficientes para que Jonas rodasse um de seus códigos e, com um largo sorriso no rosto, arrancasse com seu carro – Não se preocupe, Papai Noel, não deixarei que isso aconteça.

Experimento virtual: O que Jonas fez em seu computador foi estabelecer uma espécie de modelo para a cidade indicando o destino. Representamos a cidade de maneira simplificada como mostra a Figura 51. Cada um dos números representa um dos lugares em que está o táxi, chamaremos de células. Neste caso, a movimentação permitida é apenas para cima, baixo, esquerda ou direita. As células com um "T" marcado são lugares em que há ruas em obras, e Jonas ficaria preso se entrasse ali, já o "X" marca o destino, onde o presente final será entregue.

Figura 51. Representação de uma cidade.

1	5	9	13 T
2	6	10	14
3	7	11	15
4	8 T	12	16 X

Fonte: Cabral, 2018.

Com este "ambiente" pronto, a técnica utilizada faz uma espécie de experimento virtual que funciona **3.2 Um Conto de Natal** da seguinte maneira: um robô taxista, chamado de agente, é colocado aleatoriamente em qualquer uma das células e ele se move também aleatoriamente para uma das células vizinhas; a posição dele é chamada de estado. O movimento vai se repetindo e termina em três ocasiões: quando chega às células 8, 13 ou 16. Se chegar em 8 ou 13 recebe um reforço negativo, e chegando em 16 recebe um reforço positivo. É como se fosse concedida uma gorjeta extra por chegar em 16 e uma multa por chegar em 8 ou 13. Quando uma das condições é satisfeita, a tentativa acaba, chamamos isso de episódio. O agente então guarda em sua memória informações sobre o caminho que fez, por exemplo, se ele começou em 7 e foi para 8, ele sabe agora que o caminho 7 → 8 fornece uma recompensa negativa e tentará evitar esta ação (ir de 7 para 8). Todo esse processo é repetido diversas vezes (vários episódios) e o agente vai melhorando sua memória, ou seja, vai aprendendo sobre os melhores caminhos da cidade para chegar à célula 16.

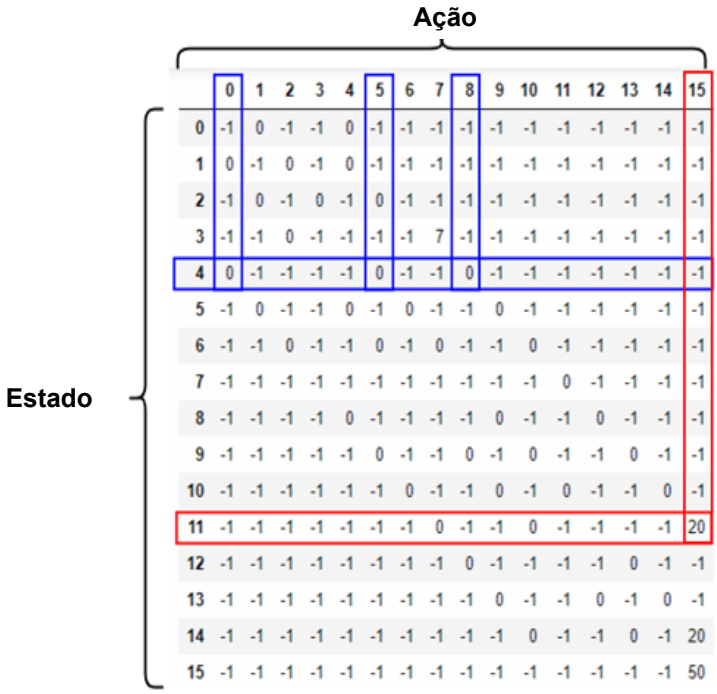
O procedimento é o seguinte:

» Sorteamos uma posição de início aleatória para o taxista: se ele cair em um bloqueio, "perde uma vida" e a tentativa acaba; caso contrário, ele escolhe aleatoriamente uma das células vizinhas e muda de posição (estado); isso é repetido até encontrar a saída.

Quando muda de estado está aprendendo sobre a cidade e atualiza sua memória (Q) com a equação de aprendizado Q. Exemplo: se ele foi para a célula 4, "olha" para as células vizinhas (3 e 8), procura nas suas experiências passadas qual das ações "ir de 4 para 3" e "ir de 4 para 8" retornou uma recompensa maior. Isso é representado pelo termo *max*(Q(s', a')). Se ele nunca executou as ações antes, esta quantidade é zero. A recompensa que o ambiente gera para ele está armazenada em R; são valores que nós definimos. Então o agente faz as contas, soma tudo à memória anterior e avança aleatoriamente para a célula 3 ou 8.

Em cada episódio de nosso experimento virtual, o agente descobre o mundo ao seu redor. Para definir a recompensa a ganhar em cada ação, façamos o seguinte: queremos que o taxista entenda que a célula 16 contém o destino final, então estar nesta e ali permanecer deve fornecer a maior recompensa, e ir para ela através das ações "12 → 16" e "15 → 16" deve fornecer recompensas um pouco menores. Representamos isso, na matriz R, na Figura 52.

Figura 52. Matriz R.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	-1	0	-1	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	0	-1	0	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
2	-1	0	-1	0	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
3	-1	-1	0	-1	-1	-1	-1	7	-1	-1	-1	-1	-1	-1	-1	-1
4	0	-1	-1	-1	-1	0	-1	-1	0	-1	-1	-1	-1	-1	-1	-1
5	-1	0	-1	-1	0	-1	0	-1	-1	0	-1	-1	-1	-1	-1	-1
6	-1	-1	0	-1	-1	0	-1	0	-1	-1	0	-1	-1	-1	-1	-1
7	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1
8	-1	-1	-1	-1	0	-1	-1	-1	-1	0	-1	-1	0	-1	-1	-1
9	-1	-1	-1	-1	-1	0	-1	-1	0	-1	0	-1	-1	0	-1	-1
10	-1	-1	-1	-1	-1	-1	0	-1	-1	0	-1	0	-1	-1	0	-1
11	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	0	-1	-1	-1	-1	20
12	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	0	-1	-1
13	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	0	-1	0	-1
14	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	0	-1	20
15	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	50

Fonte: Cabral, 2018.

Na matriz R as colunas (zero = célula 1, ..., 15 = célula 16) representam o estado e as linhas representam a ação. Exemplo: se o agente estiver na célula 12 (S = 12), e for para a célula 16 (a = 12 → 16), olhamos no destaque em vermelho que receberá uma recompensa igual a 20. Os valores -1 representam ações proibidas. Exemplo: no destaque em azul, a partir da célula 5, as únicas ações possíveis são a = {5 → 1, 5 → 6, 5 → 9}. Neste caso com valor zero, ou seja, nenhuma recompensa.

A memória do agente é parecida com essa matriz R. Ela precisa associar um valor alto para as ações boas e baixo para ações ruins. No primeiro episódio, a cidade é completamente nova para o taxista, então não há memórias (a matriz contém apenas zeros). Depois de muito treino podemos ver como ele se sai tomando suas próprias decisões.

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.