REINFORCEMENT LEARNING

UNIDADE I
APRENDIZADO POR REFORÇO

Elaboração

Natasha Sophie Pereira

Atualização

Bruno Iran Ferreira Maciel

Produção

Equipe Técnica de Avaliação, Revisão Linguística e Editoração

SUMÁRIO

APRENDIZADO POR REFORÇO

UNIDADE I

Aprendizado por reforço – *Reinforcement Learning* –, ou simplesmente RL, é um campo estudado em áreas como Estatística, Psicologia, Neurociência e Ciência da Computação. Atrai o interesse de pesquisadores, principalmente ligados ao aprendizado de máquina e grande área da inteligência artificial, e é um método de programação de agentes de *software* que oferece recompensas e punições, sem a necessidade de especificar como uma tarefa deve ser realizada.

Pode ser entendido como o problema encontrado por um agente de *software*, projetado para aprender como se comportar em cenários por meio de interações do tipo tentativa e erro. A abordagem que utilizada neste trabalho usa técnicas estatísticas e métodos de programação dinâmica, buscando estimar qual a vantagem de se tomar determinadas ações em diferentes estados do ambiente.

Esta técnica é indicada quando se deseja obter uma política ótima (define-se política ótima como o comportamento que o agente segue para alcançar seu objetivo) nos casos em que não se conhece, *a priori,* a função que modela essa política. Para isto, o agente deve interagir com seu ambiente diretamente para obter informações que serão processadas por algoritmos apropriados, a fim de executar as ações que levem o agente a atingir seus objetivos.

Suponha que você gostaria de construir um robô/agente que aprendesse a se virar sozinho. Para que o robô interaja com o ambiente, você dará a ele um conjunto de sensores, que o fará ser capaz de ler o estado do ambiente e um conjunto de ações que ele poderá executar de forma a modificar o estado do ambiente. Seu objetivo ou tarefa é aprender uma estratégia de controle ou uma política que o faça escolher as ações que o fizeram alcançar seu objetivo (MITCHELL, 1997).

Essa interação do agente com o meio através das ações tomadas mediante as recompensas e mudanças de estado o fará compreender o efeito que suas ações causaram no meio; dessa forma, ele irá guardar as ações que obtiveram sucesso, fazendo-o aprender o que fazer para atingir seu objetivo.

Em síntese, aprendizado por reforço trabalha com conhecimento de causa e efeito. É aprender o que fazer e como mapear as ações realizadas de modo a maximizar as recompensas. Nesta primeira unidade buscaremos esclarecer os principais pontos sobre RL, bem como quais funções são utilizadas para armazenar informações sobre o estado do ambiente e funções de recompensa.

CAPÍTULO 1

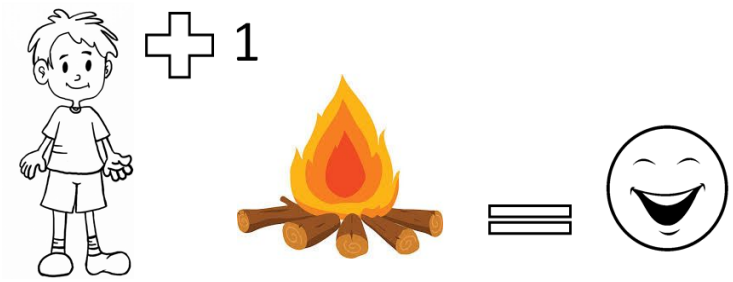
INTRODUÇÃO AO APRENDIZADO POR REFORÇO

Reinforcement learning é um tipo importante de aprendizado de máquina em que um agente aprende a se comportar em um ambiente realizando ações e vendo os resultados. A ideia por trás do *reinforcement learning* é que um agente aprenderá com o ambiente interagindo com ele e recebendo recompensas pela execução de ações.

Aprender através da interação com o ambiente vem de nossas experiências naturais. Imagine que o tempo está frio e você está em uma casa de campo bem aconchegante, e encontra uma fogueira, e decide se aproximar para se aquecer. Em se tratando de aprendizado, você é o agente, e a fogueira é o ambiente no qual você quer interagir. Se você conhece uma fogueira, e sabe que é quente próximo a ela, e você se sente bem se aquecendo nela, obtendo, assim uma recompensa positiva (+ 1) de interação com este ambiente, como podemos ver na Figura 1. Logo, dentro desta situação, você entende que o fogo é uma coisa positiva.

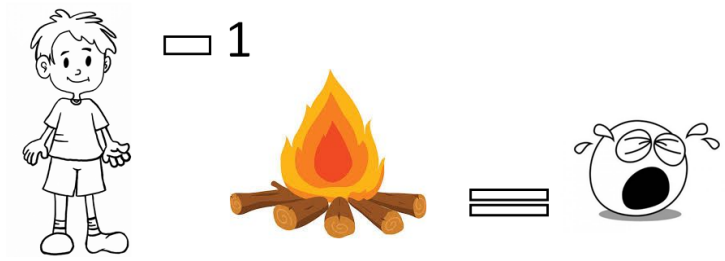
No entanto, você se aproxima demais e percebe que o fogo não é tão perfeito assim, e você se queima, tendo uma recompensa negativa (- 1) da situação, como podemos ver na Figura 2. Nesse caso, você conclui que o fogo é positivo quando você está a uma distância suficiente, porque produz calor. Porém, quando você chega perto demais, você pode se queimar.

Figura 1. Exemplo de proximidade para recompensa positiva (+ 1).



Fonte: Elaborado pelo autor.

Figura 2. Exemplo de proximidade para recompensa negativa (- 1).



Fonte: Elaborado pelo autor.

Podemos visualizar outro exemplo também referente a fogueira. Imagine ainda você perto da fogueira, você já aprendeu que quando está com frio e chega a uma distância pequena mais segura da fogueira, é recompensado por se aquecer, porém, quando chega perto demais você é penalizado por se queimar nela.

Neste novo caso, vamos considerar a questão do tempo. Imagine que é noite de São João e você está numa casa de campo, se aquecendo perto desta fogueira e tem uma montanha de milho verde do seu lado. Você não tem dúvidas, pega um espeto, coloca o milho e o vai assá-lo na fogueira.

Bem, se você não tem muita noção de quanto tempo ele deve ficar na fogueira, sua experiência inicial vai retornar situações como: se você deixar seu milho por uma quantidade de tempo relativamente pequena, acompanhando o modo como ele está assando e tirá-lo neste ponto – como é visto na Figura 3, a sua recompensa é de um milho assado e agradável para consumo (Recompensa positiva + 1).

Mas se você deixar o milho assando, e sair para dar uma volta pelo campo e admirar as estrelas, não acompanhando seu estado, e voltar depois de muito tempo, seu milho vai penalizar você com um gosto de queimado não satisfatório (Recompensa negativa - 1), ilustrado na Figura 4.

Figura 3. Exemplo de duração para recompensa positiva (+ 1).

Fonte: Elaborado pelo autor.

Figura 4. Exemplo de duração para recompensa negativa (- 1).

Fonte: Elaborado pelo autor.

É assim que os seres humanos aprendem, pela interação, por suas experiências. O *reinforcement learning* é apenas uma abordagem computacional para aprender a partir de uma ação e os reflexos que esta ação causa.

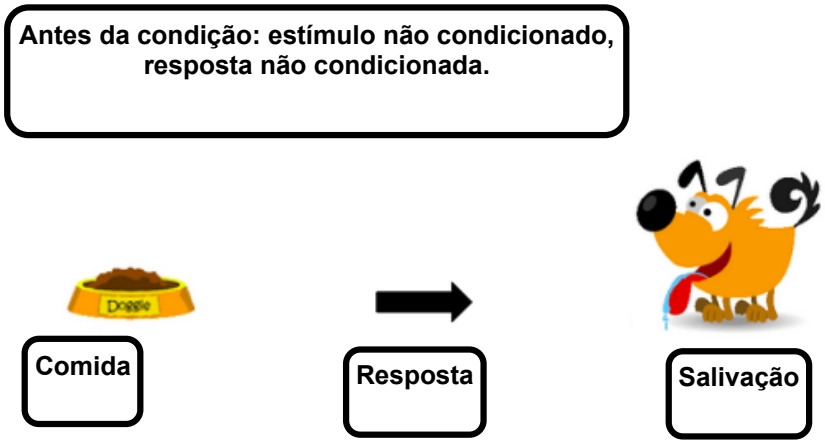
Mudando o contexto da fogueira e do fogo, podemos analisar mais um exemplo. Ele é representado nas figuras a seguir, e mostra o aprendizado de um cachorro em relação a suas recompensas, de acordo com a sua associação. A Figura 5 representa o momento anterior à condição, no qual o estímulo neural associado à comida ainda não foi condicionado a nada, o cachorro simplesmente saliva em resposta à presença da comida.

A Figura 6 representa o momento em que há o processo de estímulo neural, no qual o cachorro passou pela condição dada pelo sino, mas que ainda não representa uma resposta condicionada porque não tinha nada associado ao sino; portanto, não há salivação.

A Figura 7 também representa o condicionamento para o cachorro entre a comida e o sino, então, como a comida está presente, o cachorro saliva, mas ainda não houve o condicionamento da resposta porque ele, neste caso, está aprendendo (antes da condição), e não aplicando o que aprendeu. Diferentemente do que é apresentado na Figura 8, que mostra exatamente a conclusão do treinamento e do aprendizado do cachorro, com o momento em que ele aprendeu que quando o sino for tocado ele terá a recompensa de sua comida, desta maneira, a imagem mostra que depois da condição, o estímulo foi condicionado e a resposta também foi condicionada.

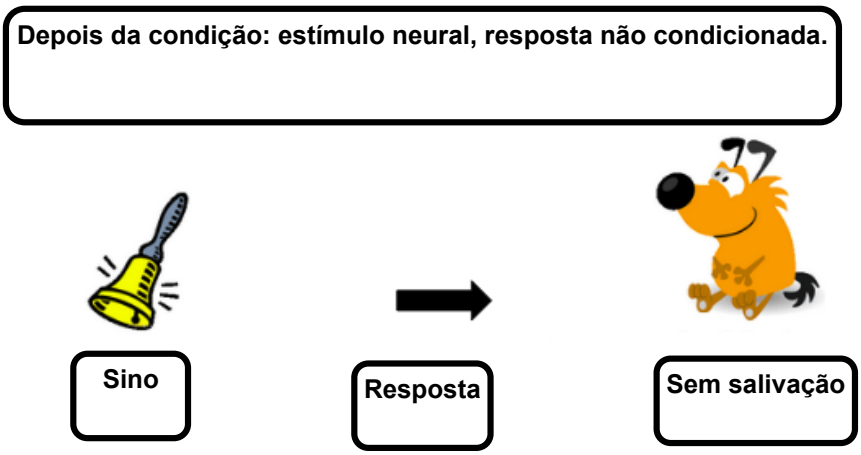
Assim, todas as vezes que este cachorro ouvir o som do sino, saberá que será recompensado com comida.

Figura 5. Aprendizagem de um cachorro com estímulo não condicionado.



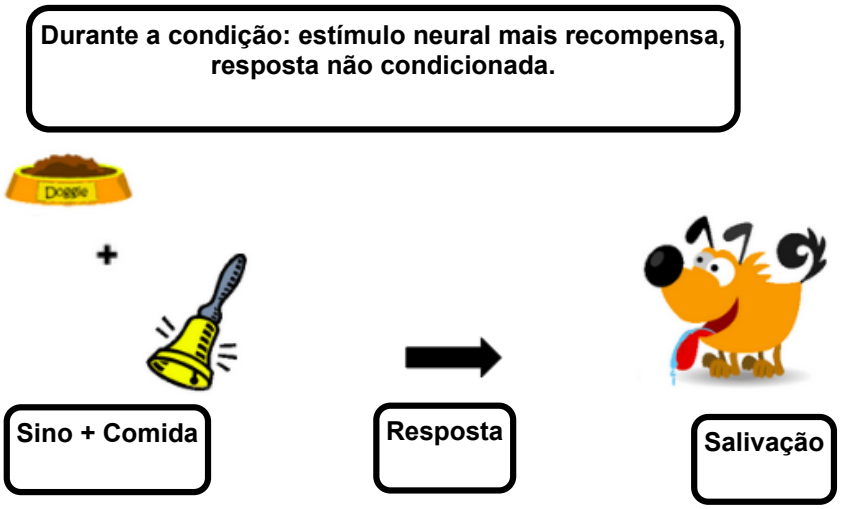
Fonte: Elaborado pelo autor.

Figura 6. Aprendizagem de um cachorro com estímulo neural.



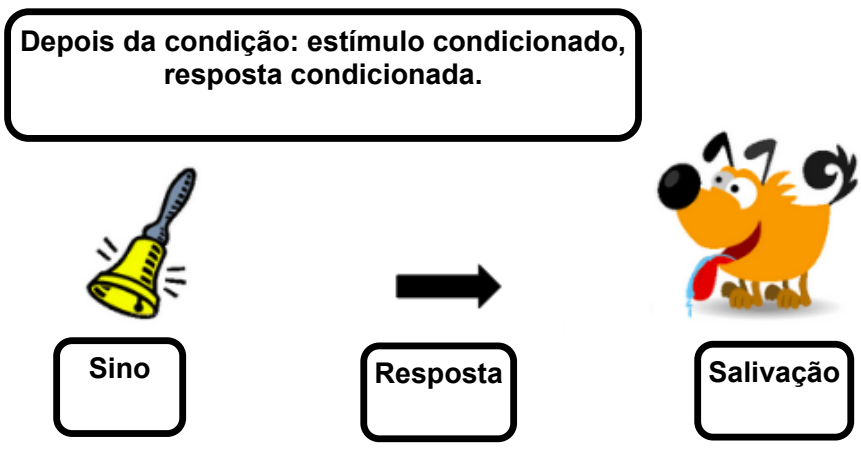
Fonte: Elaborado pelo autor.

Figura 7. Aprendizagem de um cachorro com estímulo associado.



Fonte: Elaborado pelo autor.

Figura 8. Aprendizagem de um cachorro com estímulo neural.



Fonte: Elaborado pelo autor.

Portanto, podemos observar que os algoritmos de RL funcionam com base no princípio das reações do agente ao meio ambiente. O agente interage com o ambiente fornecido e, com base no comportamento que mostra, é recompensado ou adicionado às observações. Aqui, observação é a lista de eventos em que o agente não é recompensado e, portanto, as instâncias específicas são evitadas na próxima vez que o agente interagir com o ambiente. As Figuras 5 a 8 apresentam os exemplos mais comuns de aprendizado por reforço, o treinamento de cachorros. Por meio do aprendizado por reforço, o agente aprende as maneiras certas de interagir com o ambiente e fornece os resultados desejados.

1.1. Definições de aprendizado por reforço

O aprendizado por reforço pode ser entendido usando os conceitos de **agentes**, **ambientes**, **estados**, **ações** e **recompensas**. Letras maiúsculas tendem a denotar conjuntos de coisas, e letras minúsculas denotam uma instância específica daquela coisa, por exemplo: A são todas as ações possíveis, enquanto a é uma ação específica contida no conjunto.

Vejamos detalhadamente os conceitos do aprendizado por reforço:

- **Agente (a)**: um agente executa ações, por exemplo: o Super Mário percorrendo caminhos em um *videogame*. O algoritmo utilizado pelo Super Mário para percorrer esses caminhos é o agente. Na vida, o agente é você.
- **Meio ambiente (*T*)**: é o mundo através do qual o agente se move. O ambiente toma o estado e a ação atuais do agente como entrada e retorna como saída à recompensa do agente e seu próximo estado. Se você é o agente, o ambiente pode ser as leis da Física e as regras da sociedade que processam suas ações e determinam as consequências delas.
- **Ação (A)**: conjunto de todas as jogadas possíveis que o agente pode fazer. Uma ação é quase autoexplicativa, mas deve-se notar que os agentes escolhem entre uma lista de possíveis ações. Nos *videogames*, a lista pode incluir correr para a direita ou para a esquerda, pular alto ou baixo, agachar-se ou ficar parado.
- **Estado (S)**: um estado é uma situação concreta e imediata em que o agente se encontra, ou seja, um lugar e momento específicos, uma configuração instantânea que coloca o agente em relação a outras coisas significativas, como ferramentas, obstáculos, inimigos ou prêmios. Pode ser a situação atual retornada pelo ambiente ou por qualquer situação futura. Você já esteve no lugar errado na hora errada? Isso é um estado.

» **Recompensa (R)**: uma recompensa é o *feedback* pelo qual medimos o sucesso ou fracasso das ações de um agente. Por exemplo: em um *videogame*, quando Super Mario toca em uma moeda, ele ganha pontos. De qualquer estado, um agente envia a saída na forma de ações para o ambiente, e o ambiente retorna o novo estado do agente (que resultou da ação no estado anterior), bem como recompensas, se houver alguma. Recompensas podem ser imediatas ou atrasadas. Elas efetivamente avaliam a ação do agente.

  Fator de desconto: o fator de desconto é multiplicado por recompensas futuras, conforme descobertas pelo agente, a fim de amortecer o efeito dessas recompensas na escolha da ação do agente. Por quê? Ele é projetado para fazer recompensas futuras que valham menos do que recompensas imediatas, isto é, impõem uma espécie de hedonismo em curto prazo no agente. Muitas vezes expressa com a letra grega Gama γ: se γ é 0.8 e há uma recompensa de 10 pontos após três etapas de tempo, o valor atual dessa recompensa é 0.8^3 x 10. Um fator de desconto de 1 faria as recompensas futuras valerem tanto quanto as recompensas imediatas.

» **Política (π)**: a política é a estratégia que o agente emprega para determinar a próxima ação com base no estado atual. Mapeia os estados para ações, para as ações que prometem a maior recompensa.

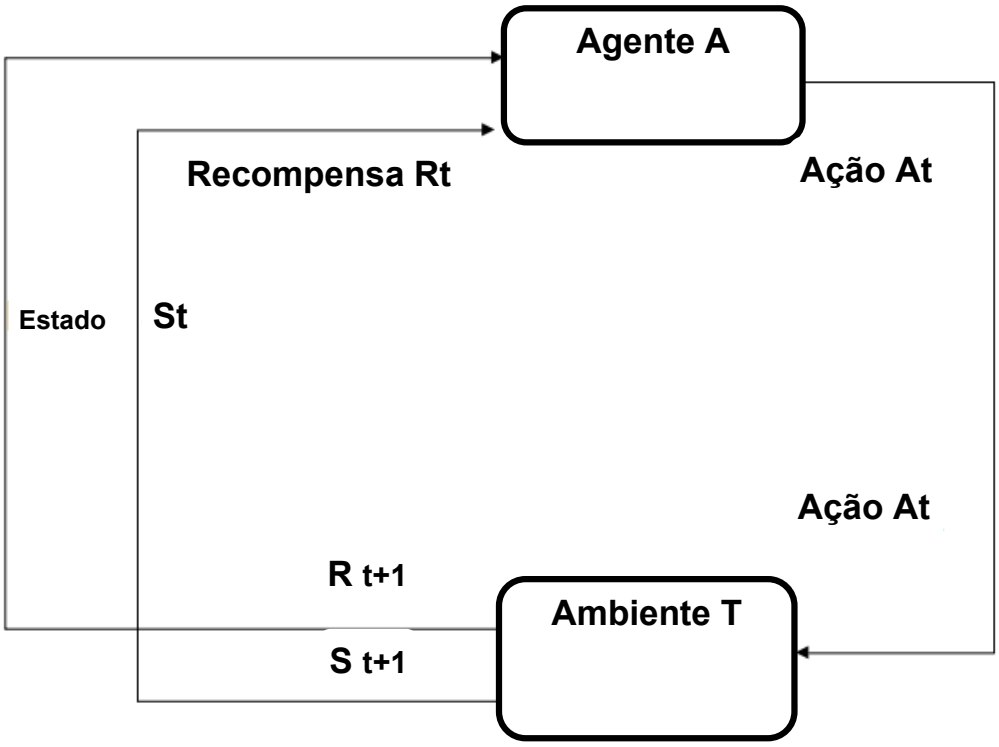
» **Valor (V)**: é o retorno esperado de longo prazo com desconto, em oposição à recompensa de curto prazo **R**. **$V_{\pi}(S)$** é definido como o retorno esperado em longo prazo do estado atual sob a política π. Nós descontamos recompensas, ou diminuímos seu valor estimado, quanto mais longe no futuro elas forem ocorrer. Veja o fator de desconto.

Então, podemos resumir ambientes como funções que transformam uma ação tomada no estado atual no próximo estado e numa recompensa, e agentes como funções que transformam o novo estado e recompensa na próxima ação. Podemos conhecer a função do agente, mas não podemos conhecer a função do ambiente. Ela funciona como uma espécie de caixa-preta, onde só vemos as entradas e saídas, mas não o seu conteúdo. Fazendo uma analogia, podemos dizer que é como o relacionamento da maioria das pessoas com a tecnologia: sabe-se o que a tecnologia faz, mas não se sabe como funciona internamente. Ou, então, por exemplo, no ramo da culinária, pode-se pedir para comer em um restaurante bem requintado um prato que tem nome exótico, você comê-lo e nem descobrir o que tinha ali, se era algo realmente diferente, novo, ou só o comum e "maquiado".

Aprendizado por reforço representa uma tentativa de um agente de aproximar a função do ambiente, de modo que podemos enviar ações para o ambiente da "caixa-preta" que

maximizem as recompensas que ele gera. A Figura 9 mostra graficamente como funciona essa interação entre agente e ambiente.

Figura 9. Interação entre agente e ambiente.



Fonte: Elaborado pelo autor.

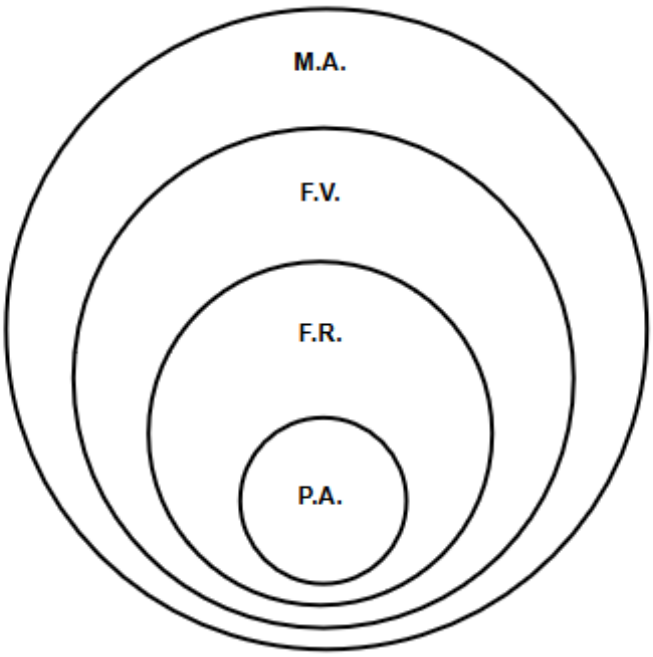
Um agente, a cada instante de tempo *t*, está em um estado *S* do ambiente. Ele executa uma ação *A*, recebe um estado *S* e uma recompensa *R*. Os intervalos de tempo *t* e *t+1* referem-se a estados diferentes: o estado no momento *t* e o estado no momento *t+1*. Ao contrário de outras formas de aprendizado de máquina – como o aprendizado supervisionado e não supervisionado – o aprendizado por reforço só pode ser pensado sequencialmente em termos de pares de ações que ocorrem uma após a outra (SUTTON; BARTO, 2018).

O aprendizado por reforço julga as ações pelos resultados que elas produzem. Seu objetivo é aprender sequências de ações que levem o agente a atingir seu objetivo. O agente, por meio dessa dinâmica, "aprende" ao interagir com o ambiente buscando atingir o objetivo proposto. Sendo assim, a principal tarefa do agente é encontrar, no sistema, uma política que mapeia as ações, os estados, e intensifica a medida de reforço em longo prazo. Por considerar que as mesmas ações tomadas em mesmos estados podem levar a estados diferentes e com diferentes valores de retorno, esses sistemas são geralmente não determinísticos.

1.2. Elementos para um aprendizado por reforço

Vimos anteriormente que o ambiente e o agente são essenciais em sistemas de aprendizado por reforço, no entanto, há ainda mais quatro elementos importantes para que ocorra a interação dinâmica do sistema. A Figura 10 apresenta graficamente esses elementos.

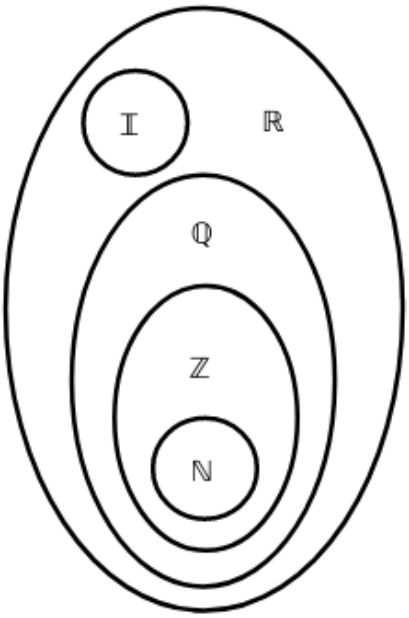
Figura 10. Representação dos elementos para um aprendizado por reforço, no qual os níveis e as correlações de continência dos conjuntos são: M.A. como modelo do ambiente; F.V. como função de valor; F.R. como função de reforço; e P.A. como política de ação.



Fonte: Elaborado pelo autor.

Caso você tenha dificuldade para entender o que esta imagem representa, você pode voltar para os conceitos dos conjuntos de números, que vai ajudar bastante, pois eles têm uma similaridade abstrata (é representado pela mesma ideia da associatividade entre si). Observe a regra de conjuntos na Figura 11:

Figura 11. Conjunto dos números.



Fonte: Elaborado pelo autor.

Na imagem temos cinco conjuntos de números. Vamos ignorar o comentário sobre os números irracionais, porque eles não nos ajudam no nosso exemplo associativo da Figura 10. Mas se analisarmos: o conjunto dos números naturais, que é representado por todos os números positivos, onde $\mathbb{N}$ = {0, 1, 2, 3, 4, 5 , 6 , 7, ...}; o conjunto dos números inteiros que é formado pelos elementos do conjunto dos números naturais e os números inteiros negativos, onde $\mathbb{Z}$ = {... ; -4, -3, -2, -1, 0, 1, 2, 3, 4, ...}; o conjunto dos números racionais, que é formado pelos números naturais, inteiros, decimais, fracionários e dízima periódica, onde $\mathbb{Q}$ = {... ; -2; -1,333; -1; 0; 1; 2; 3; 3,5; ...}; e o conjunto dos números reais que é composto por todos os conjuntos descritos anteriormente, onde $\mathbb{R}$= {... ; -4; -3; -2; -1,23; 0; 1; 2,3; 4,3333... ; 5; 6; ...}.

Então, com os números fica fácil de visualizar, e podemos compreender a ideia de continência dos elementos existentes no aprendizado por reforço. A política de ação é representada no conjunto dos números naturais, seria a situação-base e que está contida em todas as demais. A função de reforço tem alguns pontos a mais que a política de ação, mas que incorpora os seus conceitos. A função de valor agrega algo mais que a política de ação e a função de reforço. E o modelo do ambiente contém todos os conceitos anteriores a ele.

Agora vamos conceituar cada um desses pontos para entendermos um pouco mais sobre o que eles representam no *Reinforcement Learning*:

» **Política de ação (π)**: mapeia o estado percebido do ambiente pelo agente para a ação a ser executada nesse estado, maximizando a satisfação dos seus objetivos. Em

alguns casos, as políticas podem ser informações passadas por tabelas de consulta ou podem ser funções simples que podem ou não conter algoritmos complexos de pesquisa, por exemplo. A base de um agente de aprendizado por reforço são as políticas, que, sozinhas, podem determinar o comportamento.

A seleção da ação do agente pode ser modelada como um mapa de políticas:

$$\pi : S * A \rightarrow [0,1]$$

$$\pi(a \mid s) = P(a_t = a \mid s_t = s)$$

O mapa de políticas fornece a probabilidade de escolher uma ação *a* quando estiver no estado *s*.

» **Função de reforço**: em cada espaço de tempo o ambiente envia para o agente de aprendizado por reforço um valor, um número correspondente a uma recompensa. Se pensarmos em um sistema biológico, o agente humano obtém as recompensas que podem ser as experiências vividas e aprendidas para resolver seus problemas. Em um sistema de aprendizado por reforço o agente tem um único objetivo, maximizar essas recompensas a cada interação. O sinal recebido de cada recompensa define quais ações são boas ou ruins para o agente. A recompensa enviada depende da ação do agente e do estado do ambiente em que o agente se encontra.

» **Função de valor**: mapeia o estado do ambiente para um número indicando a **satisfação futura atingível** dos objetivos do agente a partir desse estado. As funções de valores, ao contrário das funções de reforço, indicam o que é bom para o sistema em longo prazo. O que a função faz é garantir totalmente a recompensa que um agente espera acumular a partir daquele estado.

A função de valor Vπ(S) é definida como um retorno esperado que começa no estado s, ou seja, s_o = s, e segue sucessivamente até a política π. Assim sendo, $V\pi(s) = E[R] = E\left[\sum_{t=0}^{\infty} y^t r_t \vee S_0 = S\right]$, onde a variável randômica R, detona o retorno e é definida como o somatório que será descontado no futuro;

Então, $R = \sum_{t=0}^{\infty} y^t \, r_t$ onde r_t é a recompensa no passo t, y ∈ [0, 1] é o percentual de desconto.

O algoritmo deverá encontrar a política que demonstra o maior retorno esperado. A teoria de MDP (do inglês, *Markov Decision Processes*) diz que sem perda de generalidade a busca pode se restringir ao conjunto das chamadas de políticas consideradas estacionárias (SUTTON; BARTO, 2018).

Uma política é dita **estacionária**, se a distribuição da ação retornada por ela depender do último estado em que ela esteve. A busca pode ser ainda mais restringida por políticas estacionárias determinísticas. Essa política busca ações com base no estado atual. Ela pode ser classificada com o mapeamento do conjunto de estados para o conjunto de ações e sem perder a sua generalidade.

» **Modelo do ambiente**: cada sistema de aprendizado por reforço aprende um mapeamento de ações por meio de tentativa e erro com um ambiente dinâmico. O ambiente deve ser em partes, observável por esse sistema, que pode utilizar para observação, equipamentos com sensores que fazem a leitura do ambiente, descrição de símbolos ou processos mentais, como exemplo, a sensação de estar perdido em um lugar desconhecido e pensar em como sair daquele lugar, ou seja, mapeiam as percepções internas do estado do ambiente. Em outros casos, ele pode mapear a ação a executar para representação interna do ambiente. No modelo de sistema de RL proposto por (SUTTON; BARTO, 2018), o ambiente é representado por um MDP, onde são desconhecidos os parâmetros iniciais.

No aprendizado por reforço, um agente produz ações em cada etapa, como "mover para a esquerda", "mover para frente", etc. A cada etapa, ele recebe observações (como os quadros de um *videogame*) e recompensas (por exemplo, r = + 1 se executar uma ação correta, r = 0 caso contrário).

Formalmente, o modelo é constituído por:

» Um conjunto discreto de estados que o ambiente pode assumir.

» Um conjunto discreto de ações que o agente pode tomar sobre o ambiente.

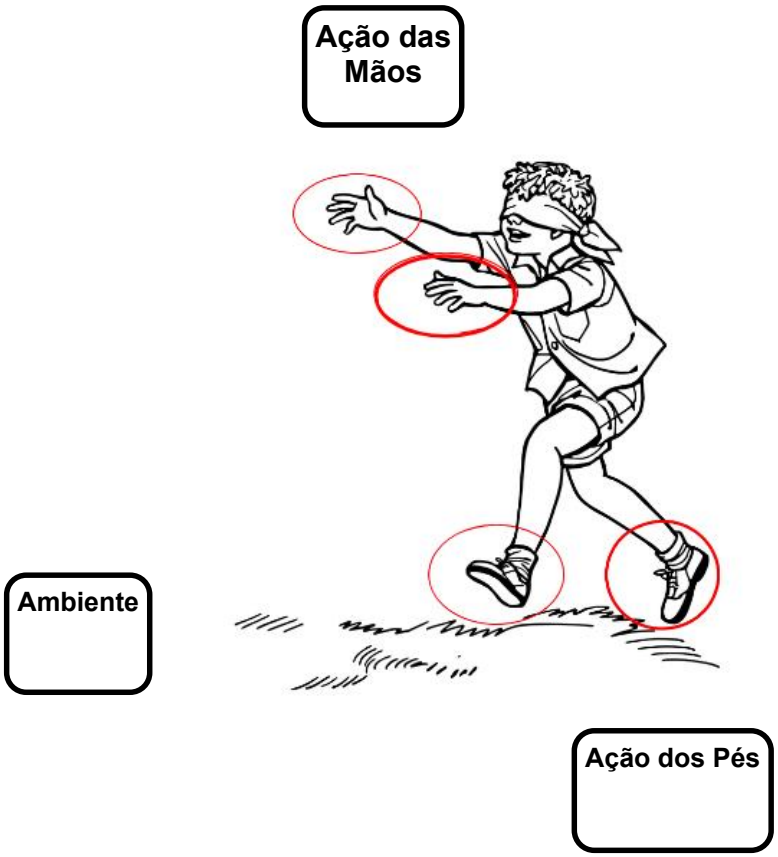
» Um conjunto de valores escalares de reforço; geralmente {0,1}, ou os números reais.

Na Figura 12 existe uma função de entrada *e*, que é representado pelos movimentos das mãos e dos pés do garoto, que sugere a maneira como o agente (o garoto) "lê" o estado atual do ambiente. Ambiente este que seria esse campo por onde ele está vagando (sugestivo por causa da grama). Todo agente que age, ele age dentro de um ambiente.

Na abordagem de aprendizado por reforço, um agente está inserido em um ambiente T (o campo) e interage com ele (o agente, o garoto) através de percepções (mãos e pés) e ações (movimentos do agente).

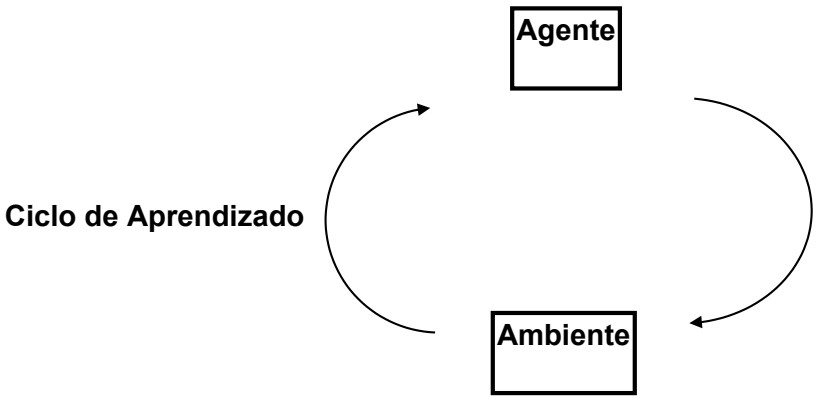
Portanto, podemos concluir que há um ciclo de aprendizado entre o agente e o ambiente em que ele está inserido, para que ações sejam tomadas e possam gerar recompensas. Toda ação vai gerar um reforço no aprendizado, como pode ser observado na Figura 13.

Figura 12. Representação de modelo-padrão de aprendizado por reforço.



Fonte: Elaborado pelo autor.

Figura 13. Representação de modelo-padrão dada, a interação entre o ambiente e o agente, gerando o ciclo de aprendizado e a recompensa.



Fonte: Elaborado pelo autor.

A cada passo, o agente recebe como entrada *"e"*, uma indicação do estado atual do ambiente *"s"*. O agente escolhe, então, uma ação *"a"* a tomar, e gera sua saída. A ação altera, então, o estado do ambiente, e uma medida dessa mudança de estado é informada ao agente através de um valor de sinal de reforço, "R". A função que mapeia os estados do ambiente nas ações que o agente deve tomar é definida como a política do agente.

A política de comportamento do agente, “P”, deve escolher tomar ações que maximizem o valor final da soma dos reforços recebidos em um intervalo de tempo. Tal política de comportamento pode ser aprendida através de um processo de tentativa e erro, guiado por diferentes algoritmos.

O objetivo da tarefa do agente é encontrar uma política π, que mapeia estados em ações, de modo a maximizar a medida de reforço em longo prazo. Geralmente, o sistema é não determinístico. Logo, uma mesma ação tomada em um mesmo estado pode levar a diferentes estados, com diferentes valores de retorno percebidos.

Ao contrário dos métodos conhecidos de aprendizado supervisionado, no aprendizado por reforço não existem pares “entrada/saída”, para serem utilizados no treinamento. Após tomar uma ação, o agente imediatamente recebe uma recompensa, mas não fica sabendo qual deveria ser a melhor ação para atingir o melhor desempenho na tarefa (maximizar o retorno a longo prazo). Logo, é preciso obter, por meio de experiência, os possíveis estados, ações, transições e recompensas do sistema para atingir o amadurecimento do modelo, ou seja, um mecanismo eficiente em termos de mapeamento das ações nas tarefas a desempenhar.

- As políticas mapeiam o ambiente para as ações que serão tomadas no mesmo estado, ou seja, percebem o estado do ambiente e escolhem a melhor política para as ações que serão tomadas.
- As transições entre estados do ambiente são mapeadas pelas funções de reforço, que são responsáveis por retornar ao agente o resultado equivalente à recompensa em detrimento da mudança de estado, ou seja, ela retorna o nível de satisfação percebido para aquela transição entre os estados.
- As funções de estado mapeiam as respostas acumuladas para cada estado, ou seja, um estado pode retornar valores ótimos – sendo considerado um estado bom – ou não, retornando valores ruins, sendo considerado um estado ruim.

CAPÍTULO 2

EXEMPLOS DE APRENDIZADO POR REFORÇO

Um agente deve aprender, por meio de suas interações com o ambiente, qual a política que melhor o leva a atingir seu objetivo. Porém, como o agente não possui conhecimento sobre todas as variáveis envolvidas no processo, é necessário que ele explore (tenha experiência), isso é, encontre políticas alternativas àquelas que já utiliza e compare seus resultados com os já aprendidos até o momento. Pode-se dizer que, com o conhecimento adquirido com a experiência até determinado momento, o agente julgue que, para um dado estado "s", a melhor ação a ser tomada seja "a_1". Em caso de existir uma ação alternativa "a_2" que leve a um melhor resultado. Vale ressaltar que apenas será possível ao agente aprender essa nova informação se ele possuir um mecanismo de exploração que tome ações aleatórias com determinada variabilidade e mecanismo de monitoramento da eficiência das ações.

Na exploração, é preciso levar em consideração a taxa de exploração. Caso de valores muito altos pode comprometer o alcance do objetivo, que é maximizar as recompensas. Um agente que decida explorar muito seu ambiente pode deixar de aproveitar o conhecimento já adquirido.

Geralmente, o problema da taxa de exploração pode ser resolvido com soluções híbridas. No início do processo, quando o agente ainda possui poucas informações sobre o ambiente, a taxa de exploração mantém-se alta; com o tempo, essa taxa cai, levando a um maior aproveitamento do conhecimento adquirido e, consequentemente, maior probabilidade de obtenção de maiores retornos.

2.1. Política ótima (π)

Em problemas de aprendizado por reforço, deseja-se que o agente aprenda uma política que seja eficaz, no sentido de obter, ao final da execução de uma sequência (limitada ou não) de ações, o melhor valor possível como recompensa acumulada.

Quando o sistema se encontra em determinado estado, o valor recebido $V_{\pi}(s)$ como recompensa, após a execução das ações seguindo uma política π, é definido como a soma das recompensas r_t recebidas a cada ação tomada. Ações tomadas mais tarde podem ter peso menor, e isso é compensado com um fator de desconto temporal, γ.

$$V\pi(s) = E\left(\sum_{t=0}^{\infty} \gamma^t r_t\right)$$

Para cada estado que o sistema assume, o valor máximo para a recompensa acumulada recebida após seguir-se a política ótima π* é definido como:

$$V(s) = \max \pi E(V_{\pi}(s))$$

Seja R(s,a) a recompensa percebida ao se tomar a ação "*a*" no estado "*s*", *T*(s,a,s') a distribuição de probabilidade para, ao se tomar a ação "*a*" no estado "*s*", o ambiente migrar para o estado "s'", V*(s'), o valor esperado para a soma das recompensas das ações tomadas seguindo a política ótima π*, a partir do estado "*s'*", γ um fator de desconto no tempo, e "*S*" o conjunto de estados que o ambiente pode assumir. Quando as recompensas recebidas em instantes diferentes de tempo apresentam a mesma importância para o sistema como um todo, o valor de γ fica fixado como 1, ou seja, a probabilidade total é igual a 1.

2.2. Aplicações do aprendizado por reforço

Uma variedade de problemas diferentes pode ser resolvida usando o aprendizado por reforço. Como seus agentes podem aprender sem supervisão especializada, os tipos de problemas mais adequados são aqueles complexos, nos quais parece não haver uma solução óbvia ou facilmente programável. Vejamos alguns exemplos a seguir.

2.2.1. Jogo da Velha

Usando aprendizado por reforço, vamos implementar o jogo da velha. A Figura 14 apresenta a árvore de possibilidades durante a execução do jogo.

A ideia de implementar o jogo da velha usando a técnica de aprendizado por reforço consiste basicamente em construir um agente que identifica as imperfeições do oponente e aprende a maximizar as chances de sua vitória. A técnica Minimax, baseada na teoria dos jogos, assume um comportamento particular do oponente. O comportamento do oponente não pode ser conhecido antecipadamente no aprendizado por reforço, mas é viável a sua identificação por meio de interações. A performance do algoritmo passaria a depender da taxa de aprendizado e da suposta habilidade do adversário.

Técnica Minimax

John Von Neumann foi o primeiro a fornecer a descrição matemática completa de um jogo e provou seu resultado fundamental utilizando o Teorema Minimax. Esse teorema assegura que para todos os jogos de duas pessoas e soma zero existia uma estratégia mista ótima para cada jogador e se eles a utilizassem teriam o mesmo resultado médio esperado, que seria o melhor ganho que cada jogador poderia esperar se o adversário jogasse racionalmente.

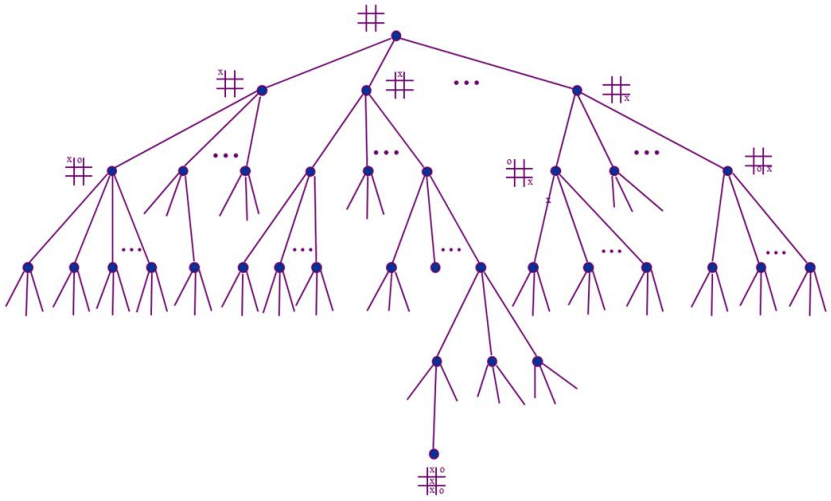
Pode-se aplicar aprendizado por reforço ao jogo da velha da seguinte maneira:

- cria-se uma tabela de números, *V(s)*, com uma entrada para cada estado *s* do jogo, conforme pode ser visto na Tabela 1;
- *V(s)* é o valor do estado *s* e representa a estimativa do agente vencer o jogo a partir do estado *s*;
- um estado s_1 é considerado melhor do que um estado s_2 se $V(s_1) > V(s_2)$;
- como definir o valor de um estado?
- uma linha com três "x" tem probabilidade 1 de vitória – **o agente já ganhou o jogo**;
- uma linha com três "o" tem probabilidade 0 de vitória – **o agente perdeu o jogo**;
- para os demais estados define-se a probabilidade de vitória de ½;
- estas probabilidades podem assumir os valores na tabela *V*;
- *Exploration x Exploitation;*
- na maioria das vezes o agente escolhe o movimento que o leva ao estado com maior valor, isto é, escolhe os estados com maior probabilidade de vitória (*Exploitation*);
- às vezes, o agente seleciona aleatoriamente alguns dos demais movimentos possíveis (*Exploration*);
- durante as jogadas, a tabela *V* é atualizada. Com isto, é possível obter estimativas mais precisas das probabilidades de vencer;
- para o aprendizado, pode-se utilizar o método de aprendizado por diferença temporal. Com este método, os valores contidos na tabela *V* são atualizados de acordo com a seguinte regra:

$$V(s) \leftarrow V(s) + \alpha\left[V(s') - V(s)\right]$$

Onde α é a taxa de aprendizado e $V(s')$ é o valor da jogada seguinte feita pelo adversário após a jogada no estado s feita pelo agente.

Figura 14. Árvore de possibilidades durante a execução do jogo da velha.



Fonte: Robin, 2002.

Tabela 1. Tabela com uma entrada para cada estado do jogo.

Estado	*V(s)* – Probabilidade estimada de ganhar	
	.5	?
	.5	?
• • •	. .	
	1	Vitória
• • •	. .	
	.	Derrota
	0	
• • •	. .	Empate
	0	

Fonte: Robin, 2002.

Mas você entendeu como funciona essa árvore da Figura 14? É simples! Vamos começar pelo primeiro nível da árvore, o nível zero, no qual só existe um nó. Neste nível só se tem a estrutura do jogo da velha e a permissão para o primeiro jogador (o jogador “x”) começar e decidir entre as nove possibilidades ele quer colocar o seu “x”. Então, o nível um, por mais que não esteja explícito na imagem, contém nove nós que representam as nove possibilidades de se encaixar o “x”, e assim que essa decisão é tomada, o nível seguinte, o nível dois, ele automaticamente desconsidera essa posição escolhida pelo jogador “x”, dado que no nível dois, a permissão de jogar é do jogador “o”, em oito posições possíveis (menos a posição que o primeiro “x” já ocupa).

Daí em diante, o jogo segue nesta mesma sequência de passos, no qual o nível três volta a ser responsável pela escolha do jogador “x”, com as (n -1) posições disponíveis a cada passo das posições que já estão ocupadas. O jogo é concluído em nove jogadas, que são as nove casas disponíveis.

A conclusão do jogo também é bem simples, como mostra a Tabela 1. Com seus dois jogadores (jogador “x” e jogador “o”) só se pode chegar a três possíveis resultados. A vitória do jogador “x” se dá quando ele consegue estabelecer uma linha, uma coluna ou uma diagonal de “x”s, e assim, consequentemente, o jogador “o” é derrotado. O inverso é igualmente verdadeiro. Porém, o jogo ainda tem a situação onde nenhum dos dois jogadores consegue garantir a meta das linhas ou colunas ou diagonais, e o jogo é considerado empatado.

Quando se analisa as probabilidades de vitória ou derrota (neste caso, estamos investigando as possibilidades de vitórias), podemos afirmar que como há somente dois jogadores, um jogo incompleto, no início, dá aos dois jogadores as mesmas probabilidades de vitória – os 50%.

2.2.2. Outros exemplos

- Em 1995, Gerard Tesauro publicou um artigo que demonstrava uma modelagem do jogo de gamão como um problema de aprendizado por reforço. A cada vitória na interação entre agente e ambiente, o sistema recebia uma recompensa de +100 pontos ou -100 pontos e zero (0), para os demais estados do jogo, após milhares de partidas contra ele mesmo, o sistema conseguiu jogar tão bem quanto o melhor jogador humano.
- **Problemas de controle**: como agendamento de elevador. Mais uma vez, não é óbvio quais estratégias forneceriam o melhor serviço de elevador. Para problemas de controle como esse, os agentes de aprendizado por reforço podem ser deixados para aprender em um ambiente simulado e, eventualmente, obterão boas políticas de controle. Algumas vantagens de usar aprendizado por reforço para problemas de controle é que um agente pode ser treinado com facilidade para se adaptar às mudanças do ambiente e treinado continuamente enquanto o sistema está *on-line*, melhorando o desempenho o tempo todo.
- Jogos com agente treinado jogando contra ele mesmo, ou várias encarnações ligeiramente diferentes dele.
- Navegação e manipulação robótica.
- Filtragem de *e-mails* sem aquisição manual e regras de filtragem.

Segundo Clemen (1996), o processo de tomada de decisão, seja complexo ou não, passa por etapas específicas, tais como, identificar situações, entender os objetivos, identificar e escolher melhores prováveis opções e implementar e analisar cada decisão tomada. O processo de tomada de decisão, bem como sua análise, demanda não somente uma simples utilização de informações do ambiente, e provoca em muitos casos, a mudança da própria concepção do ambiente, e isso pode inquietar a realidade, o que, em geral, não ocorre de forma natural, e deve ser aprendido ao longo do tempo (SANTOS; DACORSO, 2016).

Using Reinforcement Learning in Chess Engines

Mastering the game of Go without human Knowledge

Vídeo: Robô humanoide jogando air hockey

Vídeo: Robô aprendendo a imitar um humano.

CAPÍTULO 3

TIPOS DE APRENDIZADO POR REFORÇO

Vamos verificar neste capítulo que o aprendizado por reforço, que é se conceituando de forma geral, a realização pelo aprendizado de máquina do treinamento de modelos para que seja tomada uma sequência de decisões. Neste tipo de aprendizado o agente inteligente está contido e tem que atuar em um ambiente incerto e na maior parte das vezes complexo, porém, mesmo assim, ele tem que aprender a concluir suas metas.

Ainda nesta abordagem de aprendizado, o sistema inteligente tem que lidar com problemas, casos, nos quais a máquina aplica a técnica de tentativa e erro para achar a solução para o problema em questão. É claro, como já sabemos, que na computação, o computador somente faz o que o desenvolvedor programou para ele fazer, e a tarefa da inteligência artificial é aprender, ganhando informações ou percebendo que algumas delas não são úteis de acordo com as ações que o computador toma a cada ciclo de execução. Pode-se afirmar que é possível maximizar a eficiência de execução e obter maiores quantidades de resultados favoráveis, garantindo, assim, o objetivo desta abordagem de aprendizado.

Neste capítulo são vistos dois tipos de aprendizado por reforço: o aprendizado por reforço seguro a seguir; e após o aprendizado por reforço inverso, que é apresentado na sequência.

3.1. Aprendizado por reforço seguro

O objetivo do aprendizado por reforço seguro é criar um algoritmo de aprendizado que seja seguro durante o teste e o treinamento. Apesar de a ideia parecer óbvia, na teoria é um conceito jovem, ainda em definição em aprendizado por reforço. Além disso, existem poucos algoritmos de aprendizado por reforço seguro, e seu desempenho precisa ser melhorado para garantir a viabilidade em problemas mais complexos.

Segundo Garcia, Vezhnevets e Ferrari (2015), aprendizado por reforço seguro pode ser definido como o processo de aprender políticas que maximizam a expectativa de retorno em problemas nos quais é importante garantir o desempenho razoável do sistema e/ou respeitar as restrições de segurança durante os processos de exploração e aprendizado. As restrições podem ser, por exemplo, o caso de resfriamento do centro de dados, onde temperaturas e pressões devem ser mantidas abaixo dos respectivos limites em todos os momentos; ou um robô que não deve exceder os limites de velocidade, ângulos e torques; ou mesmo um veículo autônomo que deve respeitar suas restrições cinemáticas. Esse problema pode ser abordado de duas principais maneiras: alterando os critérios

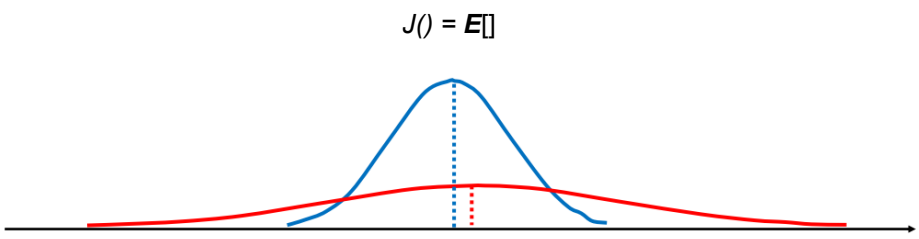
de otimização ou alterando o processo de exploração. Vejamos esses dois critérios mais detalhadamente a seguir.

O aprendizado por reforço pode treinar modelos sem grandes quantidades de dados de treinamento e é muito útil quando a função de recompensa de um resultado desejado é conhecida, mas o caminho para alcançá-la é desconhecido e exige muitas iterações para ser descoberto. Por exemplo, otimização de cadeias de suprimento e resolução de desafios de jogos são algumas das áreas que o aprendizado por reforço pode ajudar a abordar. No entanto, o aprendizado por reforço tem uma curva de aprendizado íngreme e muitas partes móveis, o que, na prática, impede o seu uso por organizações sem muitos recursos financeiros e técnicos.

3.1.1. Critérios de otimização

A Figura 15 apresenta o desempenho de uma política.

Figura 15. Retorno esperado do desempenho de uma política.



Fonte: Sikchi, 2018.

Seja J(π) o desempenho de uma política (e suponha que ela não represente o quão boa a política é). Suponha que você esteja em cassino, qual a política que você deseja seguir? A política vermelha ou a azul? A política vermelha supõe uma recompensa mais esperada. Enquanto a política azul funciona melhor porque tem menor variação, ou seja, menos chance de perder porque as baixas recompensas são as menos prováveis. Portanto, esse exemplo sugere que podemos considerar a variância como uma das medidas de risco.

Existem alguns métodos para incorporar o risco ao objetivo de otimização, são eles:

3.1.2. Critérios de pior caso

Uma política é considerada ótima se tiver o retorno máximo do pior caso, ou seja, a pior recompensa obtida pela política é maximizada. Toda a tarefa simplifica a resolução do objetivo *min-max* abaixo:

$$\max_{\pi \in \Pi} \min_{\omega \in \Omega^{\pi}} E_{\pi,\omega}(R) = \max_{\pi \in \Pi} \min_{\omega \in \Omega^{\pi}} E_{\pi,\omega}\left(\sum_{t=0}^{\infty} \gamma^{t} r_{t}\right)$$

Onde Ω é um conjunto de trajetórias da forma (s_0, a_0, s_1, a_1, ...) que ocorre sob a política π.

3.1.3. Critérios sensíveis ao risco

Inclui a notação de "risco" no objetivo de maximização da recompensa ao longo do prazo. Algumas literaturas definem como a **variância do retorno**. Considerando um exemplo de sensibilidade ao risco com base na função exponencial, uma função objetiva típica pode se parecer com:

$$\max_{\pi\in\Pi} \beta^{-1} \log E_{\pi}\left(\exp^{\beta R}\right) = \max_{\pi\in\Pi} \beta^{-1} \log E_{\pi}\left(\exp^{\beta\sum_{t=0}^{\infty}\gamma^{t}rt}\right)$$

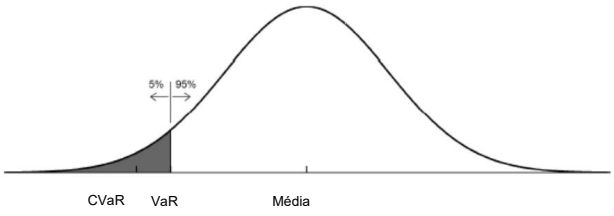
Uma expansão de Taylor do termo *exp* e *log* nos dá:

$$\max_{\pi\in\Pi} \beta^{-1} \log E_{\pi}\left(\exp^{\beta R}\right) = \max_{\pi\in\Pi} E_{\pi}(R) + \frac{\beta}{2}\mathrm{Var}(R) + \vartheta(\beta^2)$$

Onde β denota o parâmetro sensível ao risco, com o efeito de que β é negativo tem como o objetivo reduzir a variação nas recompensas e, consequentemente o risco.

Alguns outros métodos incluem maximizar o Valor em Risco (VaR), Valor Condicional em Risco (CVaR) ou outro objetivo robusto. No VaR, especificamos um valor com pelo menos, 95% das vezes que obteríamos um desempenho melhor do que esse valor, conforme é apresentado na Figura 16. CVaR é a média dos retornos do pior caso abaixo do VaR (região sombreada).

Figura 16. Valor em Risco e Valor Condicional em Risco.



Fonte: Sikchi, 2018.

3.1.4. Critérios restritos

A expectativa de retorno está sujeita a uma ou mais restrições. A forma geral dessas restrições é mostrada a seguir:

$$\max_{\pi\in\Pi} E_{\pi}(R) \text{ sujeito a } c_i \in C, c_i = \{h_i \leq \alpha_i$$

Onde c_i são as restrições pertencentes ao conjunto C. Uma política é atualizada se for segura com certa confiança, dadas as restrições.

3.1.5. Processo de exploração

Comportamentos exploratórios clássicos no aprendizado por reforço assumem que o agente tem que explorar e aprender a pesar ações diferentes e a agir de forma otimizada. O agente ignora o risco de ações, potencialmente terminando em estados perigosos. Explorações como a ε-greedy podem resultar em situações desastrosas. Além disso, as políticas de exploração aleatória desperdiçam uma quantidade significativa de tempo explorando as regiões do estado e do espaço de ação onde a política ideal nunca será encontrada.

É impossível evitar completamente situações indesejáveis em ambientes de risco sem conhecimento externo, porque o agente precisa visitar o estado perigoso pelo menos uma vez antes de rotulá-lo como "perigoso". Pode haver duas maneiras de modificar o processo de exploração:

3.1.6. Incorporar conhecimento externo

Podemos incorporar conhecimento externo para fornecer conhecimento inicial (pode ser considerado como um tipo de procedimento de inicialização) ou derivar uma política usando um conjunto finito de exemplos. Por exemplo, podemos registrar um conjunto finito de demonstrações de um professor humano e fornecer a ele um algoritmo de regressão, para construir uma função Q parcial que pode ser usada para guiar ainda mais a exploração. Esses tipos de métodos também foram usados em abordagens de neuroevolução, como em Siebel e Sommer (2007), cujos pesos de inicialização são obtidos usando um professor. No entanto, essas abordagens de inicialização não são suficientes para evitar situações perigosas que ocorrem na exploração.

Para derivar uma política de um conjunto de demonstrações, um professor, por exemplo, demonstra uma tarefa e as trajetórias de ações do estado são registradas. Todas essas tarefas são usadas para derivar um modelo da dinâmica do sistema, e um algoritmo de aprendizado por reforço encontra a política ótima nesse modelo. Essas técnicas se enquadram na categoria "aprendendo com demonstrações". Nessa metodologia, o desempenho dos alunos é limitado pela qualidade das demonstrações dos professores.

3.1.7. Exploração dirigida ao risco

Nessa tendência, uma das abordagens define uma métrica de risco sobre a noção de "controlabilidade". Intuitivamente, se um estado particular (ou par de ação de estado) produzir muita variabilidade no sinal de erro de diferença temporal, ele será menos controlável. A controlabilidade do par de ação do estado é definida como:

$$C^{\pi}(s,a) = -E_{\pi}\left[\left|\delta_t\right| s_t = s, a_t = a\right]$$

$$C(s_t,a_t) \leftarrow C(s_t,a_t) - \alpha'\left(\left|\delta_t\right| + C(s_t,a_t)\right)$$

Onde δ é o sinal de erro de diferença temporal. O algoritmo de exploração procura usar a controlabilidade como uma heurística de exploração em vez de uma exploração geral de Boltzmann. O agente é encorajado a escolher regiões controláveis do ambiente.

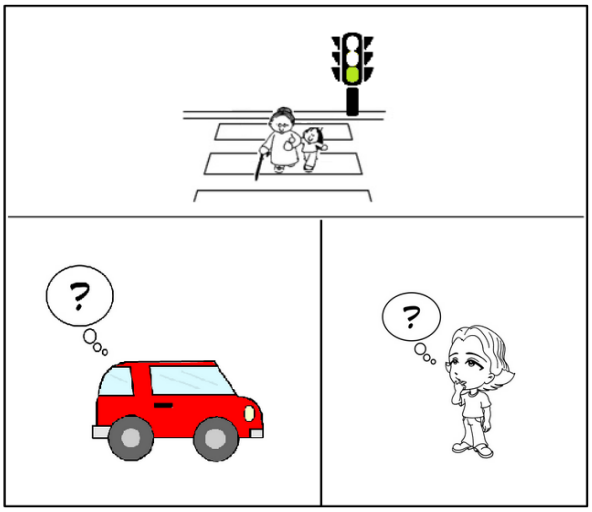
3.2. Aprendizado por reforço inverso

Em um cenário tradicional de aprendizado por reforço, o objetivo é aprender um processo de decisão para produzir um comportamento que maximize alguma função de recompensa predefinida.

Aprendizado por Reforço Inverso (IRL), como descrito por Russell e Norvig (2010), aborda o problema e tenta extrair a função de recompensa do comportamento observado de um agente.

Por exemplo, considere a tarefa de dirigir um carro autônomo. Uma abordagem simples seria criar uma função de recompensa que capte o comportamento desejado de um motorista, como parar nos semáforos, ficar fora da calçada, evitar pedestres e, assim, por diante. Infelizmente, isso exigiria uma lista exaustiva de todos os comportamentos que gostaríamos de considerar em um veículo autômato, já que a tarefa de deixá-lo em contato com o ambiente, principalmente dos pedestres (claro que considerando também os demais motoristas nos carros a sua volta), é extremamente delicada se não há uma completa certeza de que seus atos estejam bem treinados e que evitem quaisquer tipos de acidentes como uma pessoa faria. Para que isso seja possível, sua programação precisaria ter de uma lista excessiva de pesos bem detalhados descrevendo a importância de cada comportamento.

Figura 17. Exemplo do carro autômato.



Fonte: Elaborado pelo autor.

Imagine o computador do carro ter que decidir numa situação de acidente, a escolha entre atravessar um sinal vermelho e bater em nenhum, um, dois, vinte carros – não tem como se prever, mas todas essas condições seriam possíveis. Ou livrar um acidente maior atropelando somente um pedestre que está passando na faixa. Esses tipos de questionamentos são extremamente filosóficos, não só são difíceis de "explicar", programar um carro, como, também, é uma decisão difícil para nós como pessoas decidir quem vai ou quantos vão se machucar ou não de acordo com as nossas decisões.

A Figura 17 mostra um exemplo mais simples, mas que também pode se tornar contraditório se não for bem especificado a maneira como se deve agir. O exemplo mostra pedestres passando com o sinal verde para o carro, a reação tem que ser rápida e não pode causar dúvidas a quem/o que estiver guiando o carro.

Em vez disso, na estrutura da IRL, a tarefa é pegar um conjunto de dados de condução gerados pelo homem e extrair uma aproximação da função de recompensa desse humano para a tarefa. É claro que essa aproximação necessariamente lida com um modelo simplificado de direção. Ainda assim, grande parte da informação necessária para resolver um problema é capturada dentro da aproximação da função de recompensa verdadeira. Segundo Russell e Norvig (2010), a função de recompensa, e não a política, é a definição mais sucinta, robusta e transferível da tarefa, já que quantifica quão boas ou ruins certas ações são. Uma vez que temos a função de recompensa correta, o problema é encontrar a política correta, podendo ser resolvido com métodos de aprendizado por reforço padrão. É sabido que para cada tipo de problema que precisa ser resolvido por meio da tecnologia, este precisa ser previamente observado e adaptado para a situação em questão. Caso a situação seja muito complicada, no sentido de prover o aprendizado, neste caso, por reforço, seria melhor reavaliar e indicar uma outra alternativa.

Para nosso exemplo de carro autônomo, estaríamos usando dados de condução humana para aprender automaticamente os pesos de recursos certos para a recompensa. Como a tarefa é descrita completamente pela função de recompensa, nem precisamos saber os detalhes da política humana, desde que tenhamos a função de recompensa correta para otimizar. No caso geral, os algoritmos que resolvem o problema da IRL podem ser vistos como um método para alavancar o conhecimento especializado para converter uma descrição de tarefa em função de recompensa compacta.

Eis que nos deparamos com alguns problemas, sendo o principal deles converter uma tarefa complexa em uma função de recompensa simples, pois determinada política pode ser ideal para muitas funções de recompensas diferentes. Ou seja, mesmo que tenhamos as ações de um especialista, existem muitas funções diferentes de recompensas que o especialista pode estar tentando minimizar. Algumas dessas funções são bem simples, por exemplo, todas as políticas são ideais para a função de recompensa que é zero em todos os lugares, portanto, essa função de recompensa é sempre uma possível solução para o problema da IRL. No entanto, é necessário que a função de recompensa capture informações significativas sobre a tarefa e seja capaz de diferenciar claramente entre políticas desejadas e não desejadas.

Diante desse problema, Russell e Norvig (2010) formularam o aprendizado de reforço inverso como um problema de otimização, onde se quer escolher uma função de recompensa para a qual a política especializada especificada é a ideal. Mas, dada essa restrição, também se quer uma função de recompensa que maximize adicionalmente certas propriedades importantes.

3.2.1. Aprendizado por reforço para espaços finitos

Considere o conjunto de políticas ótimas para um Processo de Decisão de Markov (MDP) (que será visto na Unidade II) como um espaço de estado finito S, um conjunto de ações A e matrizes de probabilidade de transição P_a para cada ação.

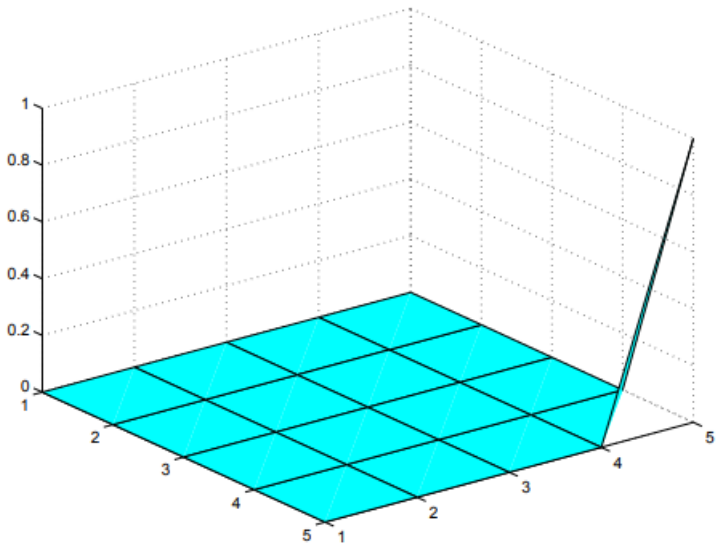
Russell e Norvig (2010) provaram que uma política π dada por π (s) = a_1 é ideal se, e somente se, para todas as outras ações a_o, o vetor de recompensa R (que lista as recompensas para cada estado possível) satisfaz a condição:

$$\left(P_{a_1} - P_a\right)\left(I - P_{a_1}\right)^{-1} R \geq 0$$

A importância desse teorema é que ele mostra que podemos selecionar de forma eficiente a "melhor" função de recompensa para a qual *a* é política ótima, usando algoritmos de programação linear, conforme pode ser visto na Figura 18. Observe que apenas um

estado é realmente recompensado. Isso permitiu que diversas literaturas tratem a IRL como um problema de otimização tratável, em que se tenta otimizar as heurísticas a seguir para que a função de recompensa se “encaixe” bem nos dados do especialista.

Figura 18. Função de recompensa.

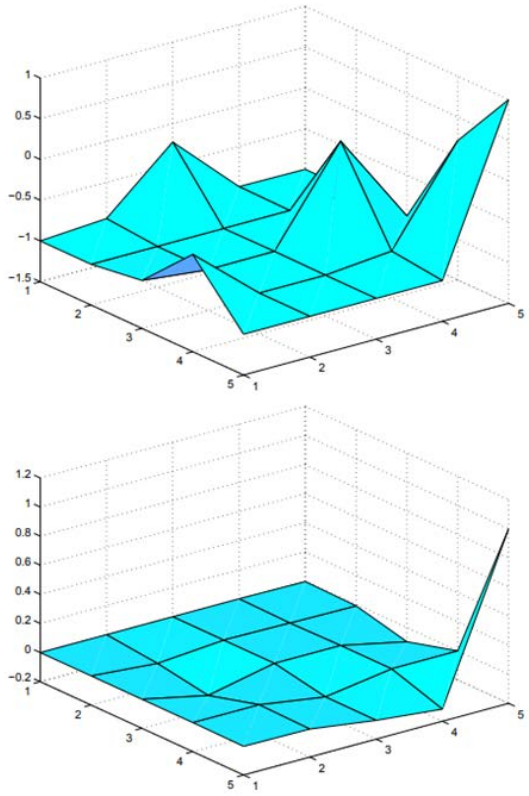


Fonte: Russel e Norvig (2010).

1. Maximize a diferença entre a qualidade da ação ótima e a qualidade da próxima melhor ação (sujeita a um limite na magnitude da recompensa, para evitar diferenças arbitrariamente grandes). Logo, se quer encontrar uma política ótima que se distingue claramente de outras políticas possíveis.

2. Minimize o tamanho das recompensas na função/vetor de recompensa. Grosso modo, a intuição é que o uso de pequenas recompensas incentiva a função de recompensa a ser mais simples, semelhante à regularização no aprendizado supervisionado. Russell e Norvig (2010) escolhem a norma L1 com um coeficiente de penalidade ajustável, que incentiva o vetor de recompensa a ser diferente de zero em alguns estados, conforme podemos ver na Figura 19.

Figura 19. Quando aumentamos o coeficiente de penalidade λ, a função de recompensa estimada torna-se mais simples.



Fonte: Russell e Norvig, 2010.

3.2.2. Aprendizado por reforço inverso de trajetórias amostradas

Russell e Norvig (2010) também descrevem algoritmos de IRL para casos em que, em vez de uma política ótima total, podemos apenas amostrar trajetórias, a partir de uma política ótima. Ou seja, são conhecidos os estados, as ações e recompensas geradas por uma política para um número finito de episódios, mas não a política em si. Esta situação é mais comum em casos aplicados, especialmente aqueles que lidam com dados de humanos especialistas.

Nesta formulação do problema, substitui-se o vetor de recompensa que é usado para espaços de estados finitos com uma aproximação linear da função de recompensa, que usa um conjunto de funções Φ_i para obter vetores de recursos com valor real $\Phi_i(s)$. Esses recursos capturam informações importantes de um espaço de estados de alta dimensão (por exemplo, ao invés de armazenar a localização de um carro durante cada etapa de tempo, podemos armazenar sua velocidade média como um recurso). Para cada recurso $\Phi_i(s)$ e peso α_1 temos:

$$R(s) = \alpha_1\phi_1(s) + \acute{a}_2\phi_2(s) + \ldots + \acute{a}_d\phi_d(s)$$

Onde o objetivo é encontrar os valores mais adequados para cada peso característico α_i.

A ideia por trás do IRL com trajetórias amostradas é melhorar de forma iterativa uma função de recompensa comparando o valor da política especializada aproximadamente ótima com um conjunto de políticas *k* geradas. O algoritmo é inicializado gerando pesos aleatoriamente para a função de recompensa estimada e inicializando o conjunto de políticas candidatas com uma política gerada aleatoriamente. Dessa forma, os principais passos para o algoritmo são:

1. estimar o valor da política ótima para o estado inicial $V^{\pi}(s_0)$, bem como o valor de cada política gerada $V^{\pi_i}(s_0)$ tomando a recompensa acumulada média de muitos ensaios aleatoriamente amostrados;
2. gerar uma estimativa da função de recompensa R resolvendo um problema de programação linear. Especificamente, define-se α_i para maximizar a diferença entre a política ótima e cada uma das outras k políticas geradas;
3. após um grande número de iterações, finalizar o algoritmo nessa etapa;
4. caso contrário, usar um algoritmo de aprendizado por reforço-padrão para encontrar a política ideal para *R*. Essa política pode ser diferente da política ótima dada, já que nossa função de recompensa estimada não é necessariamente idêntica à função de recompensa estimada que estamos procurando;
5. adicionar a política recém-gerada ao conjunto de políticas *k* candidatas e repetir o procedimento.

3.2.3. Aprendizado do aprendizado: aprender com um especialista

Além de aprender uma função de recompensa de um especialista, podemos também aprender diretamente uma política para ter um desempenho comparável ao do especialista. Isso é particularmente útil se tivermos uma política especializada que seja apenas aproximadamente ideal.

A seguir, vamos apresentar um algoritmo de aprendizado formulado por Abbeel e Andrew (2004). Esse algoritmo usa um MDP e uma política de um "especialista" aproximadamente ideal, e, em seguida, aprende uma política com desempenho comparável ou melhor do que a política do especialista, usando exploração mínima.

Essa propriedade mínima de exploração acaba sendo muito útil em tarefas frágeis como um voo autônomo de helicóptero. Um algoritmo tradicional de aprendizado por reforço poderia começar a explorar aleatoriamente, o que quase certamente levaria a

um acidente de helicóptero no primeiro teste. Idealmente, poderíamos usar dados de especialistas para começar uma política de linha de base que pode ser melhorada com segurança ao longo do tempo. Essa política de linha de base deve ser significativamente melhor do que uma política inicializada aleatoriamente, o que acelera a convergência.

3.2.4. Algoritmo de aprendizado

A principal ideia por trás do algoritmo é usar ensaios da política de especialistas para obter informações sobre o MDP subjacente e, em seguida, executar iterativamente uma melhor estimativa da política ótima para o MDP real.

A execução de uma política também nos fornece dados sobre as transições do ambiente, que pode ser usada para melhorar a precisão do MDP estimado. Vejamos como o algoritmo funciona em um ambiente discreto:

Primeiro, usamos uma política de especialistas para aprender sobre o MDP:

- execute uma quantidade fixa de testes usando a política de especialistas, registrando cada trajetória de ação do estado;
- estime as probabilidades de transição para cada par de ação do estado usando os dados registrados por meio da estimativa de máxima verossimilhança;
- estime o valor da política especializada, calculando a média da recompensa total em cada tentativa.

Então, aprendemos uma nova política:

- aprenda uma política ótima para o MDP estimado usando qualquer algoritmo de aprendizado por reforço-padrão;
- teste a política de aprendizado no ambiente real;
- se o desempenho não for suficientemente próximo do valor da política especializada, adicione as trajetórias de ação do estado dessa avaliação ao conjunto de treinamento e repita o procedimento para aprender uma nova política.

O benefício desse método é que, em cada estágio, a política que está sendo testada é a melhor estimativa para a política ótima do sistema. Há uma diminuição na exploração, mas a ideia central do aprendizado é que podemos supor que a política especializada já está próxima do ideal.

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*: **Encontro 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.