Explicando Deep Reinforcement Learning com Super Mario ao invés de matemática

Uma introdução simples para quem não tem paciência para fórmulas

Paulo Vasconcellos

10 de outubro de 2018 · 9 min de leitura

Deep Reinforcement Learning é uma técnica que tem como objetivo treinar um **Agente** (programa, código, algoritmo, etc.) a interagir em um **Ambiente** por meio de **Ações** para atingir um **Objetivo**. Por meio de **Recompensas ou punições** dadas a esse agente, ele irá **aprender quais ações deve executar para aumentar a recompensa e atingir o objetivo**.

Só de escrever essa explicação já fiquei com sono. Vamos tentar algo mais simples. **Esse tutorial tem como objetivo explicar os conceitos básicos do Aprendizado Profundo de Reforço**, além de deixar algumas dicas de conteúdos para você seguir após a leitura. **Não utilizarei códigos nem fórmulas durante esse tutorial** e, para criarmos uma fácil analogia, vamos utilizar como exemplo a criação de um algoritmo que conseguiria passar de uma fase do Super Mario.

Processo de decisão de Markov

Você conhece o Mario — sem trocadilhos — ? Ele é um encanador bigodudo, personagem de videogames famoso e que alguém aqui gosta muito. Na maioria de seus jogos, Mario tem um objetivo bem claro: **resgatar a princesa**. Para isso, ele precisa passar por diversas fases enfrentando inimigos e coletando itens. Agora, imagine que a gente queira treinar um algoritmo para passar de uma fase do jogo do Super Mario, sabendo quando e quais botões apertar para cada tipo de situação. Como ele funcionaria?

Todo algoritmo de Deep Reinforcement Learning segue um processo chamado **Processo de Decisão de Markov — Markov Decision Process**. Mas, antes de falar disso, vamos entender o que é uma simples Cadeia de Markov.

O que é Cadeia de Markov?

Um Processo Markoviano ou Cadeia de Markov é um **processo estocástico**, ou seja, é totalmente aleatório, onde o **Estado futuro** depende apenas de seu **Estado atual**. Ficou complicado? Vamos deixar as coisas mais claras.

Vamos supor que nosso herói bigodudo possa fazer apenas três coisas: **andar**, **ficar parado/esperando**, ou **pular**. Cada uma dessas três coisas é um **Estado**.

Cadeia de Markov com Super Mario

Segundo é previsto pela Cadeia de Markov, **o próximo estado em que Mario vai estar apenas depende do estado atual dele**. Ou seja, não importa se Mario ficou pulando nos últimos 5 minutos, se nesse momento ele está parado, é tudo que precisamos saber.

Por que isso é importante? Simples, já que os Estados anteriores não são interessantes para nós, podemos saber qual o próximo Estado de Mario ao calcularmos a probabilidade de transição dele. **Em outras palavras, podemos saber a chance que Mario tem de ir do estado parado para o estado andando**, por exemplo. Podemos fazer isso com um amiguinho chamado Matriz de Transição.

A Matriz de Transição

Não, não essa Matrix

Você pode imaginar uma Matriz de Transição como uma tabela onde cada linha é um possível Estado do Mario. Em nosso caso, teremos uma tabela com três linhas. Assim como temos um estado em cada linha, também teremos uma coluna para cada estado. No fim das contas, você acaba com uma matriz cruzada. Tipo assim:

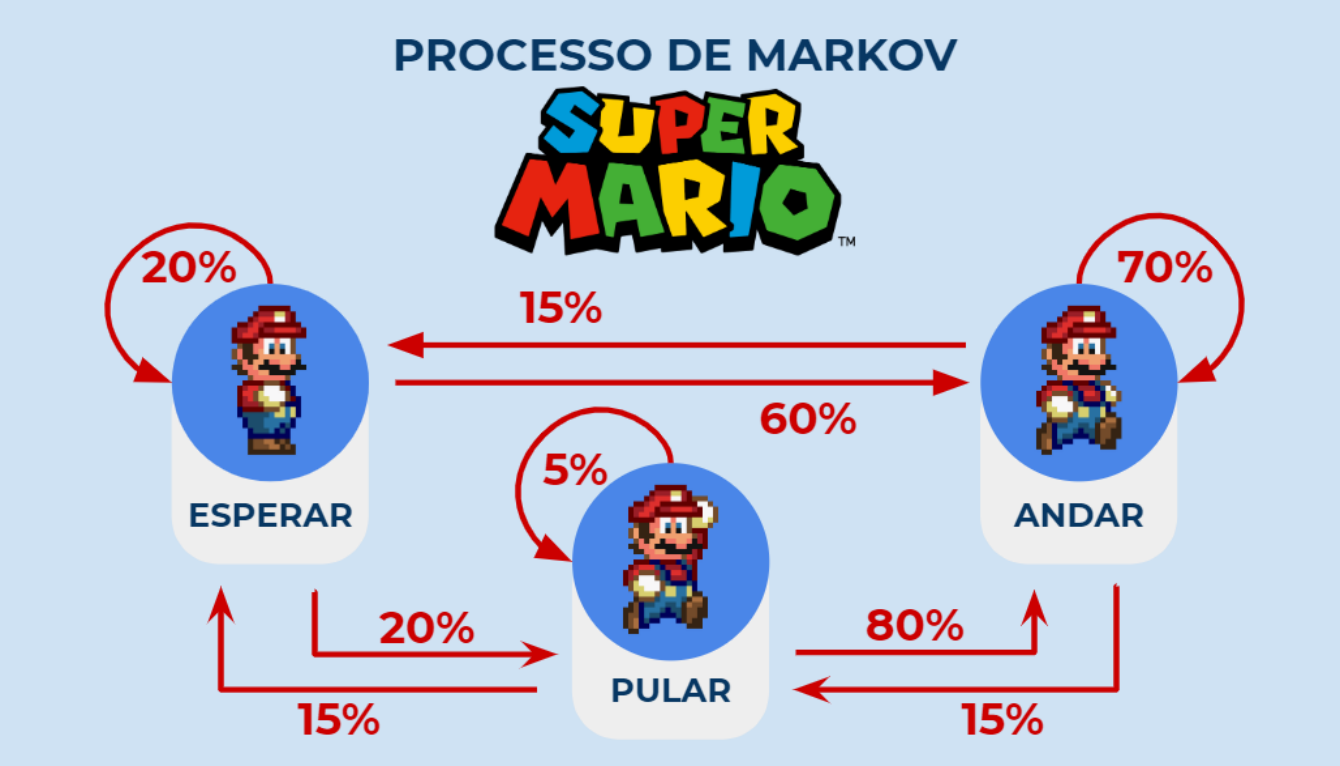
	Matriz de transição Mario		
	Andar	Esperar	Pular
Andar	70%	15%	15%
Esperar	60%	20%	20%
Pular	80%	15%	5%

Matriz de transição do Super Mario

O valor de cada célula da tabela é a probabilidade que Mario tem de passar daquele estado (linha) para o futuro (coluna). Um desses estados, inclusive, é ficar no mesmo estado em que está.

Supondo alguns valores na planilha acima, **se Mario estiver no estado Andar, ele tem 70% de probabilidade de continuar no Estado Andar**; da mesma forma, ele tem 15% de chance de ir para o estado Esperar ou Pular. Caso nosso amigo bigodudo esteja no estado Pular, ele tem apenas 5% de probabilidade de continuar nesse estado.

Podemos ilustrar a tabela acima da seguinte forma também:



Matriz de transição ilustrada

Beleza, e onde Deep Learning entra nisso?

Excelente pergunta, por sinal

Deep Reinforcement Learning utiliza Processo de Decisão de Markov, que é uma Cadeia de Markov igual explicamos acima, mas que muda um pouco suas propriedades. Resumidamente, todo Processo de Decisão de Markov — que, daqui pra frente, irei chamar apenas de MDP, *Markov Decision Process* — é uma tupla com cinco valores:

1. Conjunto finito de Estados (denotado pela letra S)

Esse conjunto de estados podem ser os que falamos acima: **andar**, **esperar** e **pular**. Contudo, cada projeto terá seu próprio conjunto de Estados, como por exemplo, a informação se uma porta está aberta ou fechada, ou a posição geográfica de algo.

2. Conjunto finito de ações (A)

Ações são justamente o que parece: são ações que, em nosso caso, Mario pode realizar em cada Estado. Ação aqui pode estar no ato de apertar um botão para Mario passar de um Estado para outro, **como apertar o botão para frente ou para trás para fazer o bigodudo sair do Estado parado para andando**.

3. Modelo de probabilidade (P)

É a probabilidade de uma Ação levar Mario do seu Estado atual para um Estado futuro.

4. Recompensa (R)

A Recompensa é um valor número que Mario irá receber após realizar aquela Ação naquele Estado. Quando esse número é positivo, quer dizer que a Ação que Mario fez naquele Estado foi benéfica; caso o número seja negativo, Mario será punido por aquela Ação. O objetivo principal de Mario é realizar o conjunto de Ações que maximizam a Recompensa até seu objetivo final: completar a fase.

5. Fator de desconto (Y)

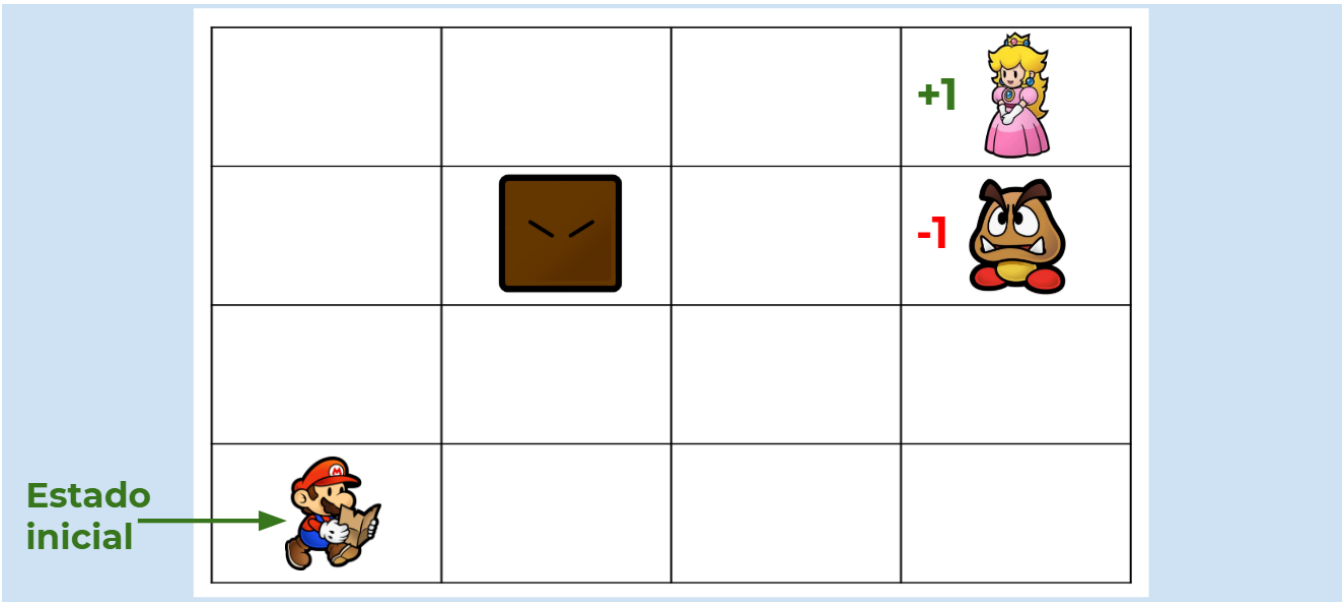
O fator de desconto é um número escalar, geralmente entre 0 e 1. Ele influencia no total de recompensa futura que o seu Agente irá receber. Por exemplo, um fator de desconto de 0,9 indica que quanto mais avançado no futuro está seu Agente, maior será a recompensa dele.

Em outras palavras, Mario irá preferir percorrer caminhos na fase que irão garantir uma recompensa maior no futuro, ao invés de uma Recompensa imediata.

Juntando as peças

Com base nessas cinco propriedades base, um algoritmo de RL é capaz de aprender, através de tentativa e erro, quais Ações (A) ele deve fazer em cada Estado (S) a fim de que sua Recompensa (R) seja a maior possível para atingir um Objetivo. **Tenha em mente que, durante o treinamento, o Agente irá tentar todas as possibilidades e selecionar a que garantir um melhor resultado no final**.

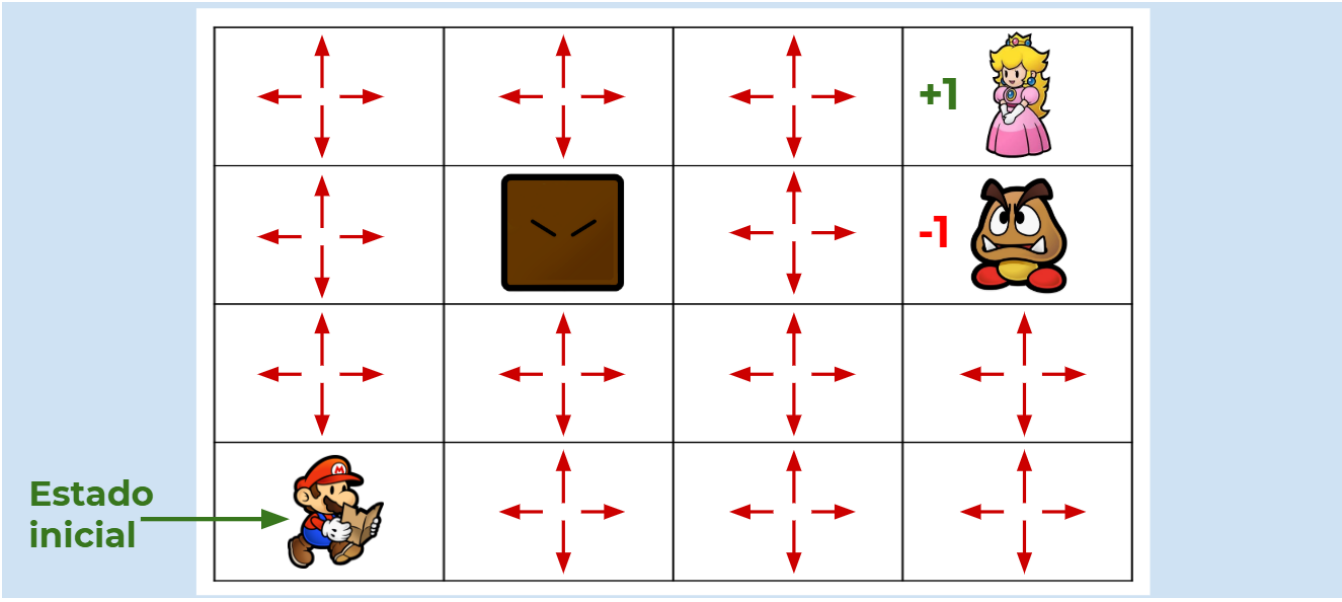
Vamos aprimorar nosso exemplo com o clássico mundo em grade. Mario iria interagir com ambiente da seguinte forma.



Grid World Super Mario

Imagine que esse mundo em grade é a fase de nosso jogo. No canto inferior esquerdo está Mario em seu estado inicial — onde ele começará a fase. No canto superior direito está a princesa Peach, seu objetivo; logo abaixo dela está um inimigo de Mario, que ele precisa evitar; e, por fim, um obstáculo é encontrado na segunda linha, segunda coluna, bloqueando a passagem de Mario por aquele quadrado.

Cada quadrado da nossa fase gradeada é um Estado em que Mario poderá estar. Agora, para nosso exemplo, imagine que Mario pode fazer uma de quatro Ações: ir para cima, para baixo, para esquerda e para direita. Em outras palavras, nosso conjunto de Ações (A) é quatro.

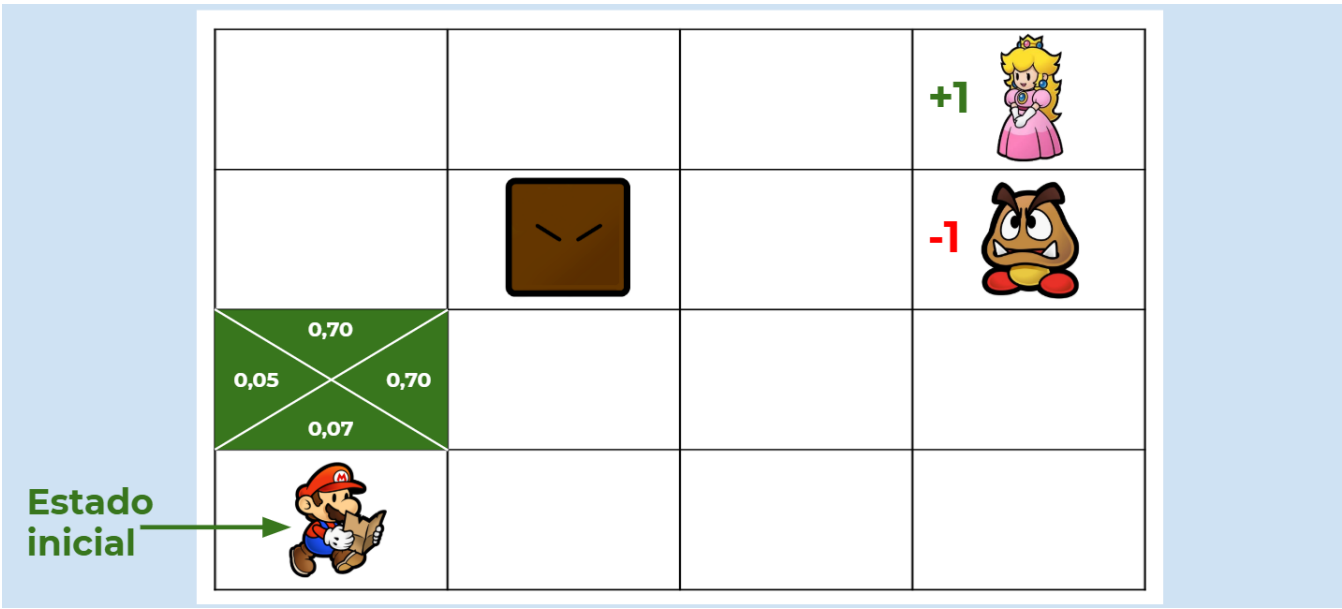


Para cada estado (quadradinho), Mario pode fazer um de quatro ações

O principal objetivo de nosso Agente (Mario) é seguir a **melhor política**, ou seja, **escolher as melhores Ações em cada Estado para coletar o máximo de Recompensas no caminho**. Mas, como o Marinho sabe o que é uma Ação boa?

Recompensas e punições em MDP

Mario saberá quais as melhores Ações para se realizar em cada Estado ao receber uma Recompensa por ela. Recompensas nada mais são que números inteiros ou flutuantes, e o Agente sempre irá tentar coletar o máximo possível de Recompensa, o que faz o processo de definição do valor da Recompensa uma verdadeira arte.



O quadrado verde mostra qual será a Recompensa final esperada de Mario se ele escolher cada uma das Ações possíveis

Dá uma olhada na imagem acima. Esse quadrado verde indica qual será a Recompensa esperada no final se ele escolher uma das Ações. Por exemplo: **se Mario escolher ir para cima, a Recompensa que ele receberá no final do Episódio (fase) será 0,70**. Caso ele escolha ir para a esquerda, ele vai receber uma Recompensa menor de 0,05, afinal, ele está batendo na parede.

Eu preenchi apenas um quadrado para facilitar o entendimento, mas, na prática, **todos os quadrados estarão preenchidos da mesma forma**, onde cada Ação em um Estado resultará em uma Recompensa final diferente. O **Agente sempre escolherá as Ações que maximizam a Recompensa final ao chegar na Princesa**.

Onde aprendo mais sobre Reinforcement Learning?

O conteúdo que trouxe aqui apenas riscou a superfície desse iceberg chamado Deep Reinforcement Learning. Seria impossível ilustrar todos os conceitos desse campo em um artigo, mas espero ter ao menos ilustrado um pouco sobre o processo para vocês. Abaixo, deixo algumas dicas para caso você queira se aprofundar mais no assunto:

Curso gratuito da School of AI: esse curso foi criado pelo Siraj Raval para ensinar gratuitamente sobre Deep Reinforcement Learning. O conteúdo do curso é bem direto e vai lhe oferecer um bom *overview* sobre o assunto.

Curso gratuito — e antigo pá car@!#0 — do Sebastian Thrun: esse curso foi criado por Sebastian Thrun — fundados da Udacity — há mais de 5 anos, mas ainda se mantém atualizado. Para você ter uma idéia, o sucesso estrondoso desse curso foi o fator que motivou Thrun a fundar a Udacity, cujo o curso teve mais de 50 mil alunos inscritos em menos de duas semanas.

Curso de DRL para Python no Udemy: particulamente, não cheguei a fazer este curso, mas, um profissional de referência para mim indicou e, por analisar as qualificações, acho que vale a pena conferir. O curso foca em ensinar os conceitos teóricos e práticos de RL utilizando Python.

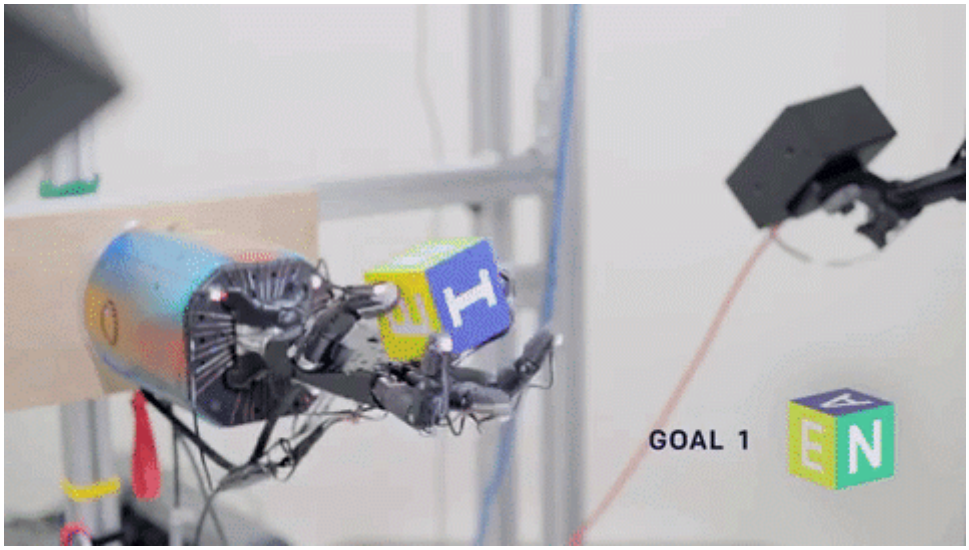
Livro gratuito de Introdução a Reinforcement Learning (Sutton and Barto): esse livro é a verdadeira Bíblia sobre Aprendizado de Reforço. Nele você vai entender tudo que tem para se saber sobre RL. Inclusive, o exemplo que você conheceu nesse post foi inspirado no capítulo 3 desse livro.

Por fim, uma menção honrosa: a OpenAI

A OpenAI é uma das maiores empresas que tem contribuído para RL no mundo. Trata-se de uma organização que tem criado coisas incríveis através de pesquisas e desenvolvimento. Uma de suas aplicações mais destacadas é um time de Dota 2 — jogo eletrônico — feito interamente por bots: o OpenAI Five.

Bot da OpenAI descendo o cacete nas inimiga

Sua atuação é tão impressionante que o pessoal da OpenAI realiza testes — *benchmarks* — **contra jogadores humanos**. Isso mesmo: para saber se o algoritmo tem funcionado bem, jogadores de alto nível jogam partidas contra o OpenAI Five. O resultado: o OpenAI Five vem destruindo times humanos durante as partidas. O algoritmo está longe de ser perfeito, mas o que mais me deixa fascinado pela OpenAI é que **todos os seus resultados são divulgados para a comunidade**, seja na forma de papers ou reports.



Um dos últmos esforços do OpenAI: ensinar destreza a um robô

Mas, por que eu estou aqui babando o ovo da OpenAI? Primeiro, porque eu me tornei fã dos pesquisadores e acompanho muito de perto os avanços que tem feito, principalmente através de seu blog. Segundo, como disse anteriormente, eles são os maiores contribuidores para a comunidade de Deep Reinforcement Learning que conheço — junto com a fast.ai, é claro —, oferecendo ferramentas de pesquisa como o Gym e o improved GANs. Não deixe de conferir o trabalho deles e ajudar na divulgação desse trabalho primoroso.

Valeu, é nois

E aí, curtiu o post de hoje? Já tinha ouvido falar sobre Deep Reinforcement Learning? O que cachorro? Siga o blog para ficar por dentro de mais posts sobre Machine Learning e Inteligência Artificial raiz. Ah, vem ser meu amiguinho no Linkedin e no Twitter também, vai ter bolo.

Aprendizado de Máquina Aprendizagem Profunda AI Aprendizagem por Reforço