Inteligência Artificial

Ministrado por Raul Steinmetz

Oferecimento de:

https://www.versusai.org/

Contexto Histórico

Para compreendermos a definição de inteligência artificial e sua importância, motivações e aplicações em um contexto contemporâneo, é importante começarmos com uma breve contextualização histórica.

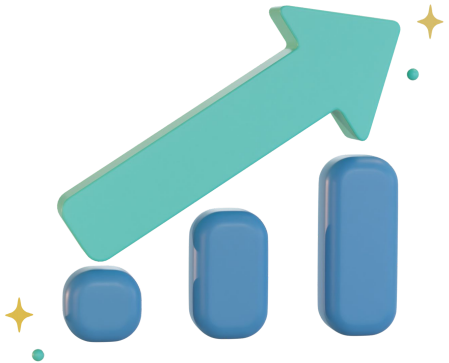
"Google AI defeats human Go champion", BBC News, 25/05/2017, https://www.bbc.com/news/technology-40042581

Antes do Machine Learning

Os seres humanos já tentavam prever eventos com base em dados conhecidos mesmo antes da invenção dos computadores.

Métodos estatísticos tradicionais

- Análise de Regressão
- Análise de Séries Temporais
- Teste de Hipóteses
- Estatística Bayesiana



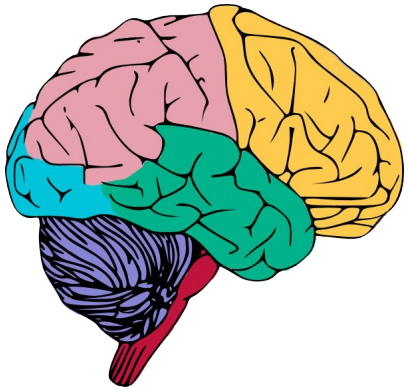
Machine Learning

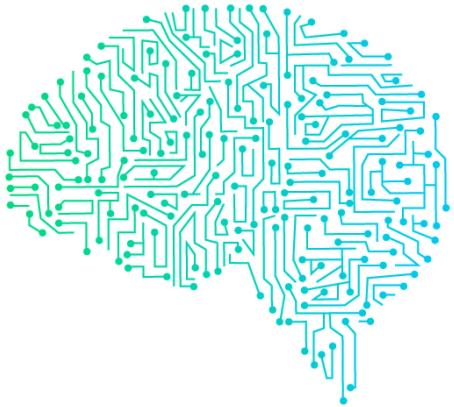
Com o surgimento dos computadores na década de 1950, os métodos estatísticos puderam ser executados em grande escala e com muito mais rapidez, o que levou a uma explosão no campo da análise de dados.

Os primeiros algoritmos de aprendizado de máquina foram baseados em técnicas estatísticas, como a regressão linear e a análise de variância. Esses algoritmos foram desenvolvidos para encontrar padrões em dados, fazer previsões e tomar decisões com base em resultados passados.

Começo do Deep Learning?

Máquinas imitando inteligência humana começaram a ser ideia no início do século 20. Durante a Segunda Guerra, cientistas trabalharam em sistemas de inteligência artificial para decodificação de mensagens. A inteligência artificial se tornou um campo de pesquisa formal nos anos 50 e 60, com algoritmos como Perceptron e Adaline.



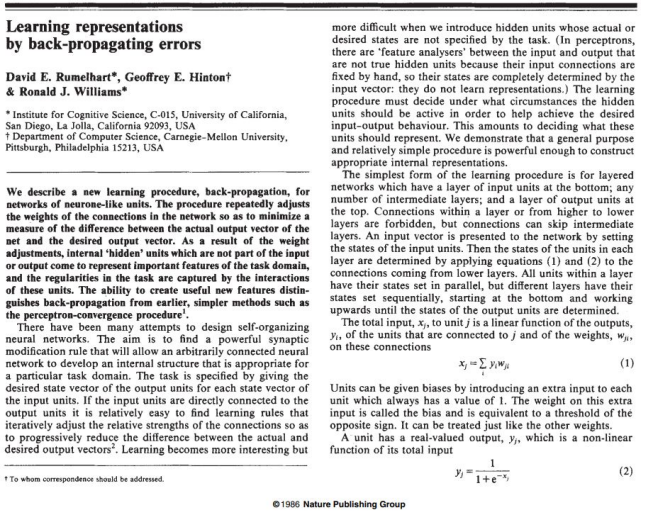


Começo do Deep Learning?

Durante as décadas de 70 e 80, algoritmos mais complexos envolvendo redes neurais foram estudados. No entanto, devido à falta de poder computacional e outros problemas, esse campo foi quase totalmente abandonado na década de 90.

Começo do Deep Learning?

Com o avanço do poder computacional e o desenvolvimento de algoritmos fundamentais como o backpropagation, o objetivo de treinar redes neurais profundas - ou deep learning - se tornou cada vez mais possível. Esse sonho foi alimentado pela possibilidade de se utilizar várias camadas para se obter uma representação mais rica dos dados e solucionar problemas complexos.



Learning representations by back-propagating errors

David E. Rumelhart*, Geoffrey E. Hinton† & Ronald J. Williams*

* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure[1].

There have been many attempts to design self-organizing neural networks. The aim is to find a powerful synaptic modification rule that will allow an arbitrarily connected neural network to develop an internal structure that is appropriate for a particular task domain. The task is specified by giving the desired state vector of the output units for each state vector of the input units. If the input units are directly connected to the output units it is relatively easy to find learning rules that iteratively adjust the relative strengths of the connections so as to progressively reduce the difference between the actual and desired output vectors[2]. Learning becomes more interesting but more difficult when we introduce hidden units whose actual or desired states are not specified by the task. (In perceptrons, there are 'feature analysers' between the input and output that are not true hidden units because their input connections are fixed by hand, so their states are completely determined by the input vector: they do not learn representations.) The learning procedure must decide under what circumstances the hidden units should be active in order to help achieve the desired input–output behaviour. This amounts to deciding what these units should represent. We demonstrate that a general purpose and relatively simple procedure is powerful enough to construct appropriate internal representations.

The simplest form of the learning procedure is for layered networks which have a layer of input units at the bottom; any number of intermediate layers; and a layer of output units at the top. Connections within a layer or from higher to lower layers are forbidden, but connections can skip intermediate layers. An input vector is presented to the network by setting the states of the input units. Then the states of the units in each layer are determined by applying equations (1) and (2) to the connections coming from lower layers. All units within a layer have their states set in parallel, but different layers have their states set sequentially, starting at the bottom and working upwards until the states of the output units are determined.

The total input, x_j, to unit j is a linear function of the outputs, y_i, of the units that are connected to j and of the weights, w_{ji}, on these connections

$$x_j = \sum_i y_i w_{ji} \qquad (1)$$

Units can be given biases by introducing an extra input to each unit which always has a value of 1. The weight on this extra input is called the bias and is equivalent to a threshold of the opposite sign. It can be treated just like the other weights.

A unit has a real-valued output, y_j, which is a non-linear function of its total input

$$y_j = \frac{1}{1+e^{-x_j}} \qquad (2)$$

† To whom correspondence should be addressed.

©1986 Nature Publishing Group

Finalmente Deep Learning!

Em 2006, Geoffrey Hinton publicou um artigo que mostrava como treinar uma rede neural de múltiplas camadas capaz de reconhecer dígitos escritos à mão com uma precisão superior a 98%. Esse artigo revolucionou o mundo da aprendizagem de máquina e marcou o surgimento do Deep Learning.

LETTER Communicated by Yann Le Cun

A Fast Learning Algorithm for Deep Belief Nets

Geoffrey E. Hinton
hinton@cs.toronto.edu
Simon Osindero
osindero@cs.toronto.edu
Department of Computer Science, University of Toronto, Toronto, Canada M5S 3G4

Yee-Whye Teh
tehyw@comp.nus.edu.sg
Department of Computer Science, National University of Singapore, Singapore 117543

We show how to use "complementary priors" to eliminate the explaining-away effects that make inference difficult in densely connected belief nets that have many hidden layers. Using complementary priors, we derive a fast, greedy algorithm that can learn deep, directed belief networks one layer at a time, provided the top two layers form an undirected associative memory. The fast, greedy algorithm is used to initialize a slower learning procedure that fine-tunes the weights using a contrastive version of the wake-sleep algorithm. After fine-tuning, a network with three hidden layers forms a very good generative model of the joint distribution of handwritten digit images and their labels. This generative model gives better digit classification than the best discriminative learning algorithms. The low-dimensional manifolds on which the digits lie are modeled by long ravines in the free-energy landscape of the top-level associative memory, and it is easy to explore these ravines by using the directed connections to display what the associative memory has in mind.

1 Introduction

Learning is difficult in densely connected, directed belief nets that have many hidden layers because it is difficult to infer the conditional distribution of the hidden activities when given a data vector. Variational methods use simple approximations to the true conditional distribution, but the approximations may be poor, especially at the deepest hidden layer, where

Introdução a Machine Learning

Antes de mergulharmos no mundo do deep learning, é importante entendermos os fundamentos do machine learning, pois são a base dessa área de conhecimento. O estudo do machine learning nos permite entender como os algoritmos são capazes de aprender a partir dos dados e como podemos utilizar esses algoritmos para resolver problemas complexos.

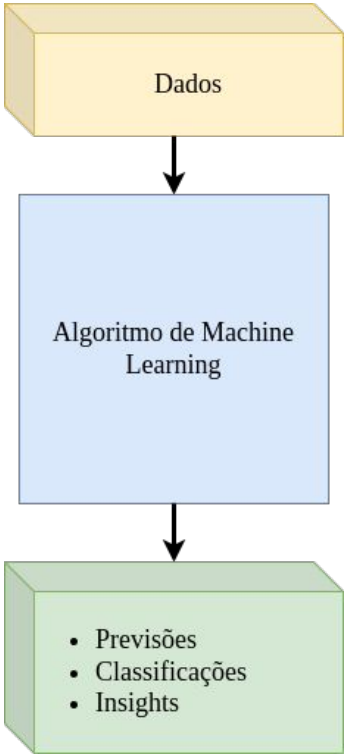
O que é Machine Learning?

Em resumo, Machine Learning é o conjunto de algoritmos que possibilita que computadores aprendam a partir de dados.

"Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed" - Artur Samuel, 1959

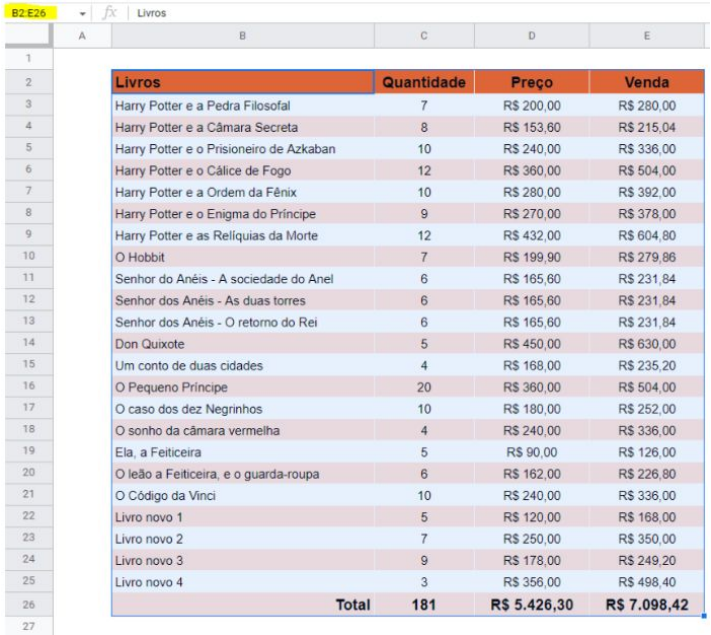
O que é Machine Learning?

Aprendizado de máquina é um ramo da inteligência artificial que usa algoritmos e modelos estatísticos para permitir que sistemas de computador aprendam a partir de dados, identifiquem padrões e façam previsões ou decisões sem serem programados explicitamente para isso. Isso envolve treinar um sistema de computador para reconhecer e responder a determinados tipos de entrada, alimentando-o com grandes quantidades de dados e permitindo que ele aprenda com esses dados, melhorando sua capacidade de fazer previsões ou decisões ao longo do tempo.



O que são dados?

- Dados são informações, fatos ou estatísticas que são coletados, armazenados e processados por computadores ou outros dispositivos eletrônicos. Eles podem ser representados em diferentes formas, como números, texto, imagens ou sons.
- Os dados podem ser estruturados ou não estruturados, dependendo da forma como são organizados e armazenados. Os dados estruturados são geralmente organizados em tabelas, com campos e colunas bem definidos, enquanto os dados não estruturados não possuem uma estrutura definida e podem incluir informações de texto, imagem ou som.



B2:E26 | Livros

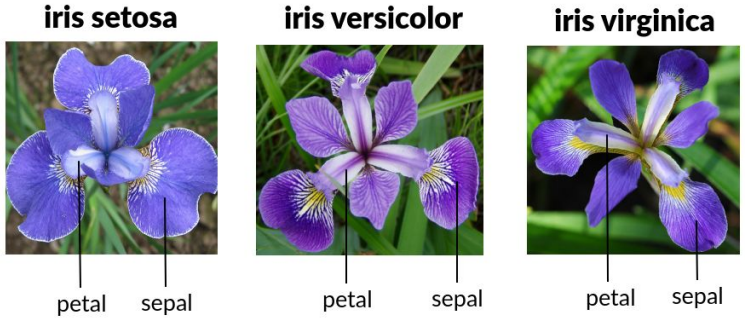
Livros	Quantidade	Preço	Venda
Harry Potter e a Pedra Filosofal	7	R$ 200,00	R$ 280,00
Harry Potter e a Câmara Secreta	8	R$ 153,60	R$ 215,04
Harry Potter e o Prisioneiro de Azkaban	10	R$ 240,00	R$ 336,00
Harry Potter e o Cálice de Fogo	12	R$ 360,00	R$ 504,00
Harry Potter e a Ordem da Fênix	10	R$ 280,00	R$ 392,00
Harry Potter e o Enigma do Príncipe	9	R$ 270,00	R$ 378,00
Harry Potter e as Relíquias da Morte	12	R$ 432,00	R$ 604,80
O Hobbit	7	R$ 199,90	R$ 279,86
Senhor do Anéis - A sociedade do Anel	6	R$ 165,60	R$ 231,84
Senhor dos Anéis - As duas torres	6	R$ 165,60	R$ 231,84
Senhor dos Anéis - O retorno do Rei	6	R$ 165,60	R$ 231,84
Don Quixote	5	R$ 450,00	R$ 630,00
Um conto de duas cidades	4	R$ 168,00	R$ 235,20
O Pequeno Príncipe	20	R$ 360,00	R$ 504,00
O caso dos dez Negrinhos	10	R$ 180,00	R$ 252,00
O sonho da câmara vermelha	4	R$ 240,00	R$ 336,00
Ela, a Feiticeira	5	R$ 90,00	R$ 126,00
O leão a Feiticeira, e o guarda-roupa	6	R$ 162,00	R$ 226,80
O Código da Vinci	10	R$ 240,00	R$ 336,00
Livro novo 1	5	R$ 120,00	R$ 168,00
Livro novo 2	7	R$ 250,00	R$ 350,00
Livro novo 3	9	R$ 178,00	R$ 249,20
Livro novo 4	3	R$ 356,00	R$ 498,40
Total	**181**	**R$ 5.426,30**	**R$ 7.098,42**

Training and Testing Data

- Em aprendizado de máquina, é comum dividir o conjunto de dados em duas partes: o conjunto de treinamento (train) e o conjunto de teste (test).
- A ideia por trás da divisão do conjunto de dados em treinamento e teste é evitar o sobreajuste (overfitting) do modelo, que ocorre quando o modelo se ajusta muito bem aos dados de treinamento, mas não consegue generalizar para novos dados. Ao usar um conjunto de teste separado, é possível avaliar a capacidade de generalização do modelo e detectar o sobreajuste.
- A proporção de dados usados para treinamento e teste depende do tamanho do conjunto de dados e da complexidade do modelo. Geralmente, cerca de 70% a 80% dos dados são usados para treinamento e o restante é usado para teste. É importante garantir que os dados usados para teste sejam representativos dos dados do mundo real que o modelo encontrará.

Vamos para o código!
Machine Learning

Prevendo espécies de flor!



Tipos de Machine Learning

1. Aprendizado supervisionado: um algoritmo é treinado com um conjunto de dados rotulados para prever resultados em novos dados. É útil para problemas de classificação e regressão.
2. Aprendizado não supervisionado: o algoritmo é treinado com um conjunto de dados não rotulados para encontrar padrões e relações ocultas nos dados. É útil para análise exploratória de dados e agrupamento.
3. Aprendizado por reforço: o algoritmo aprende através da interação com um ambiente dinâmico, recebendo feedback positivo ou negativo em relação às suas ações. É útil para problemas de controle e tomada de decisão em tempo real.

Aprendizado supervisionado

- É um gato ou um cão?
- Esse investimento é de risco ou seguro?
- Que algarismo está desenhado nessa imagem?
- Quanto vale esse apartamento?

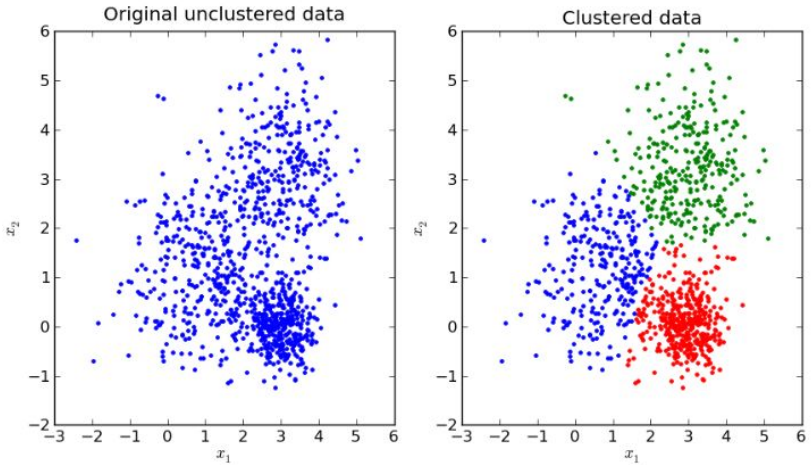
O algoritmo feito a pouco (Classificação Iris) é supervisionado!

Vamos para o código!
Machine Learning Supervisionado

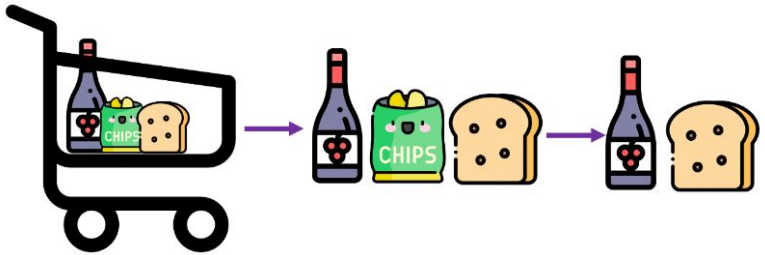
Prevendo índice de evolução de diabetes!

Aprendizado não supervisionado

- Clustering

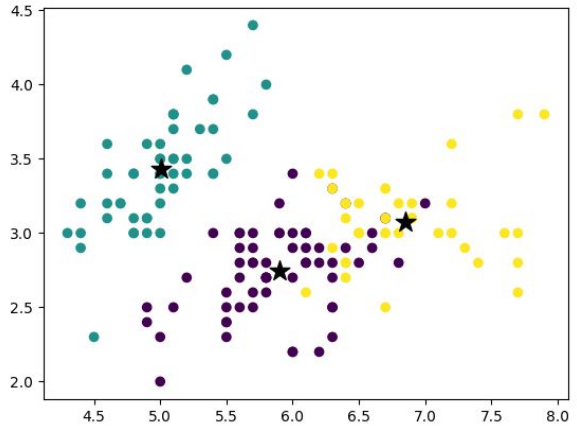


- Apriori

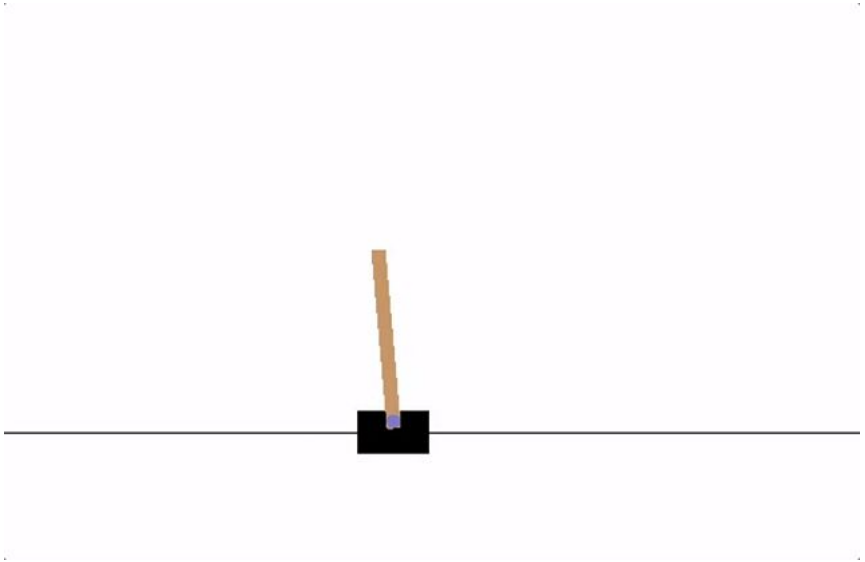


Vamos para o código!
Machine Learning Não Supervisionado

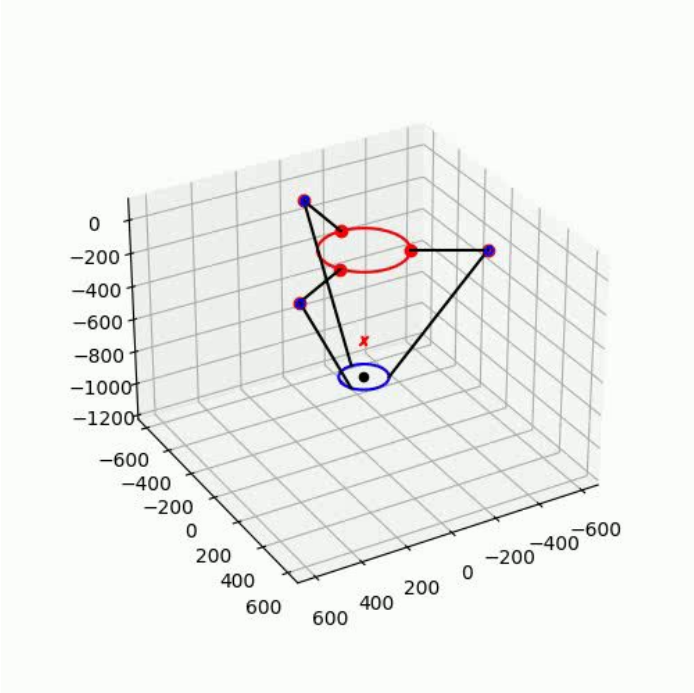
Separando flores em grupos! Sem saber a que espécie pertencem!



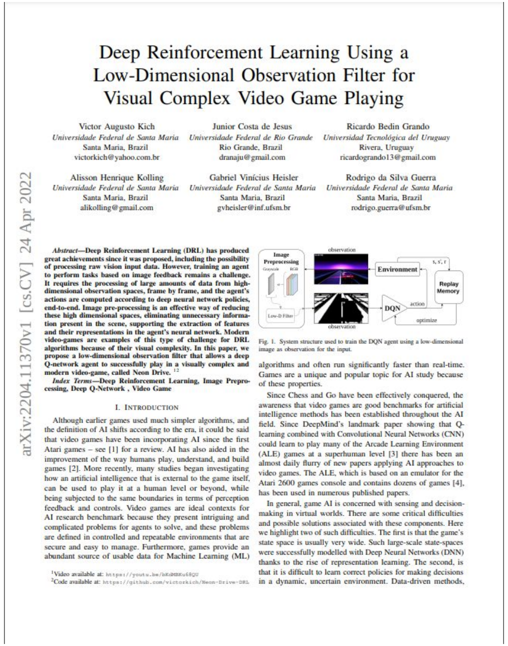
Aprendizado por reforço



Robô manipulador que aprendeu sozinho



Aumento da complexidade dos problemas



Deep Reinforcement Learning Using a Low-Dimensional Observation Filter for Visual Complex Video Game Playing

Victor Augusto Kich
Universidade Federal de Santa Maria
Santa Maria, Brazil
victorkich@yahoo.com.br

Junior Costa de Jesus
Universidade Federal de Rio Grande
Rio Grande, Brazil
dranaju@gmail.com

Ricardo Bedin Grando
Universidad Tecnológica del Uruguay
Rivera, Uruguay
ricardogrando13@gmail.com

Alisson Henrique Kolling
Universidade Federal de Santa Maria
Santa Maria, Brazil
alikolling@gmail.com

Gabriel Vinícius Heisler
Universidade Federal de Santa Maria
Santa Maria, Brazil
gvheisler@inf.ufsm.br

Rodrigo da Silva Guerra
Universidade Federal de Santa Maria
Santa Maria, Brazil
rodrigo.guerra@ufsm.br

arXiv:2204.11370v1 [cs.CV] 24 Apr 2022

***Abstract*—Deep Reinforcement Learning (DRL) has produced great achievements since it was proposed, including the possibility of processing raw vision input data. However, training an agent to perform tasks based on image feedback remains a challenge. It requires the processing of large amounts of data from high-dimensional observation spaces, frame by frame, and the agent's actions are computed according to deep neural network policies, end-to-end. Image pre-processing is an effective way of reducing these high dimensional spaces, eliminating unnecessary information present in the scene, supporting the extraction of features and their representations in the agent's neural network. Modern video-games are examples of this type of challenge for DRL algorithms because of their visual complexity. In this paper, we propose a low-dimensional observation filter that allows a deep Q-network agent to successfully play in a visually complex and modern video-game, called Neon Drive.** [1,2]

***Index Terms*—Deep Reinforcement Learning, Image Preprocessing, Deep Q-Network , Video Game**

I. INTRODUCTION

Although earlier games used much simpler algorithms, and the definition of AI shifts according to the era, it could be said that video games have been incorporating AI since the first Atari games – see [1] for a review. AI has also aided in the improvement of the way humans play, understand, and build games [2]. More recently, many studies began investigating how an artificial intelligence that is external to the game itself, can be used to play it at a human level or beyond, while being subjected to the same boundaries in terms of perception feedback and controls. Video games are ideal contexts for AI research benchmark because they present intriguing and complicated problems for agents to solve, and these problems are defined in controlled and repeatable environments that are secure and easy to manage. Furthermore, games provide an abundant source of usable data for Machine Learning (ML) algorithms and often run significantly faster than real-time. Games are a unique and popular topic for AI study because of these properties.

[1]Video available at: https://youtu.be/bKdMBKu68QU
[2]Code available at: https://github.com/victorkich/Neon-Drive-DRL

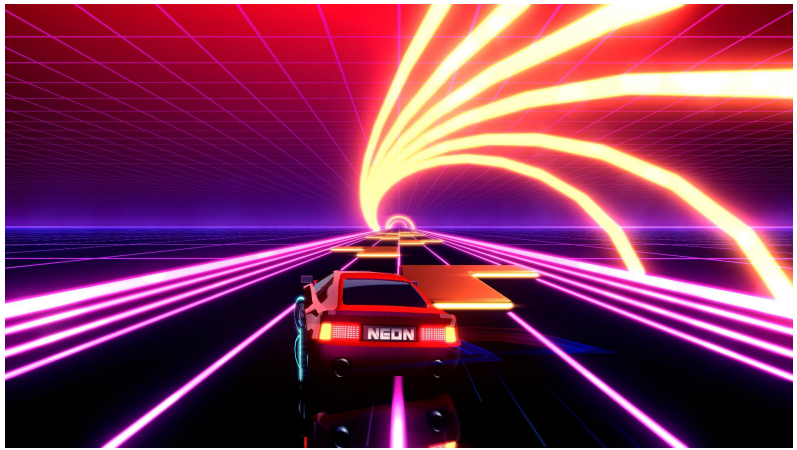
Image Preprocessing
Grayscale
RGB
Low-D Filter
observation
observation
Environment
s, s', r
Replay Memory
DQN
action
optimize

Fig. 1. System structure used to train the DQN agent using a low-dimensional image as observation for the input.

Since Chess and Go have been effectively conquered, the awareness that video games are good benchmarks for artificial intelligence methods has been established throughout the AI field. Since DeepMind's landmark paper showing that Q-learning combined with Convolutional Neural Networks (CNN) could learn to play many of the Arcade Learning Environment (ALE) games at a superhuman level [3] there has been an almost daily flurry of new papers applying AI approaches to video games. The ALE, which is based on an emulator for the Atari 2600 games console and contains dozens of games [4], has been used in numerous published papers.

In general, game AI is concerned with sensing and decision-making in virtual worlds. There are some critical difficulties and possible solutions associated with these components. Here we highlight two of such difficulties. The first is that the game's state space is usually very wide. Such large-scale state-spaces were successfully modelled with Deep Neural Networks (DNN) thanks to the rise of representation learning. The second, is that it is difficult to learn correct policies for making decisions in a dynamic, uncertain environment. Data-driven methods,

Machine Learning Clássico não resolve



É possível encaixar 1920 X 1080 (2.073.600) pixels, com 3 canais de cor cada (RGB), ou seja, 6.220.800 variáveis em uma equação linear? Ou quadrática? E fazer ela decidir entre 3 ações baseado no output daquela equação? Não!

Deep Learning

"Birds inspired us to fly, burdock plants inspired Velcro, and nature has inspired countless more inventions. It seems only logical, then, to look to the brain's architecture for inspiration on how to build an intelligent machine." (Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow).

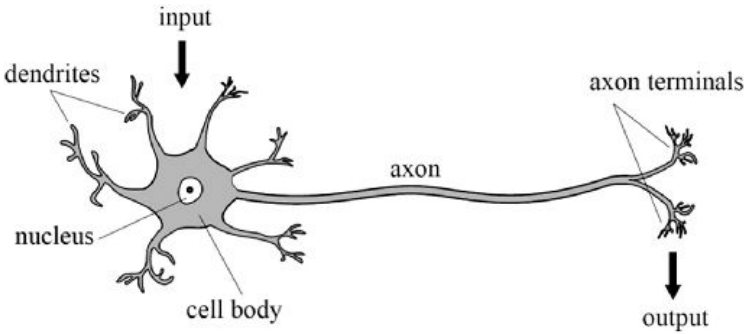
Deep Learning

Antes de discutirmos a parte mais essencial de uma rede neural artificial, um neurônio artificial, vamos estudar um neurônio biológico, seguindo os mesmos passos de Warren McCulloch e Walter Pitts, que idealizaram as redes neurais artificiais em 1943

Neurônio Biológico

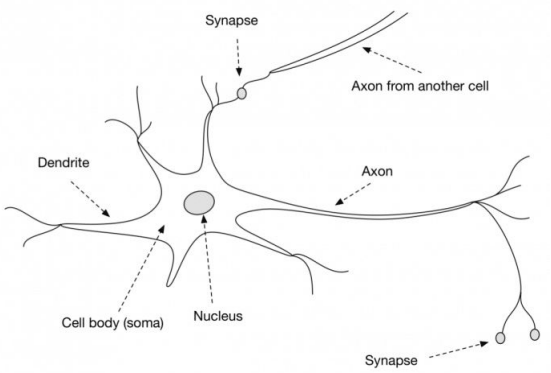
Composto por:

- Um corpo que contém um núcleo e a maioria dos componentes complexos da célula
- Diversas extensões chamadas de dendritos
- Uma grande extensão chamada de axônio, que possui em sua extremidades, vários segmentos chamados de telodendros, nas extremidades destes, por sua vez, existem os terminais sinápticos, que são conectados a dendritos de outros neurônios



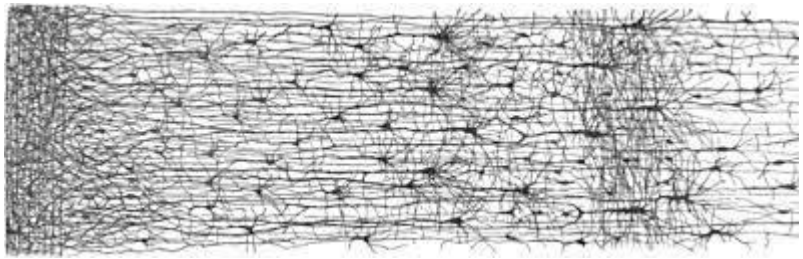
Neurônio Biológico

Neurônios biológicos produzem impulsos elétricos curtos chamados potenciais de ação, que viajam pelos axônios e fazem as sinapses liberarem sinais químicos chamados neurotransmissores. Quando um neurônio recebe uma quantidade suficiente desses neurotransmissores em poucos milissegundos, ele dispara seus próprios impulsos elétricos.



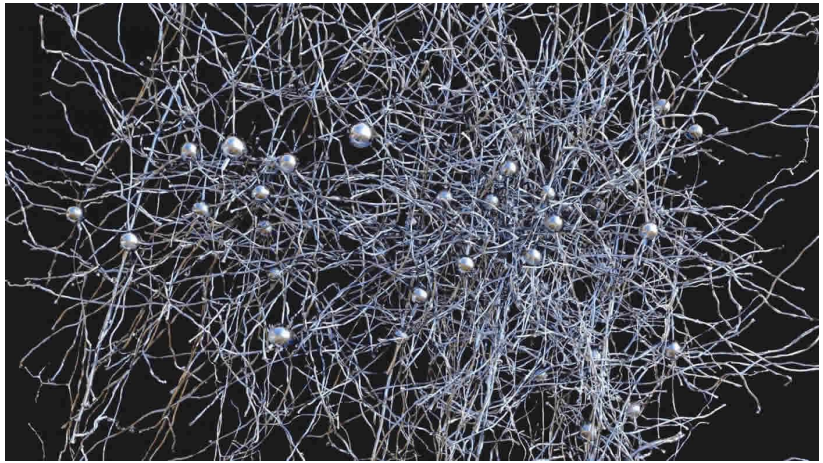
Neurônio Biológico

Os neurônios biológicos parecem se comportar de maneira simples, mas como são organizados em uma rede de bilhões de unidades, com cada neurônio sendo conectado com milhares de outros neurônios, impulsos extremamente complexos podem ser realizados.



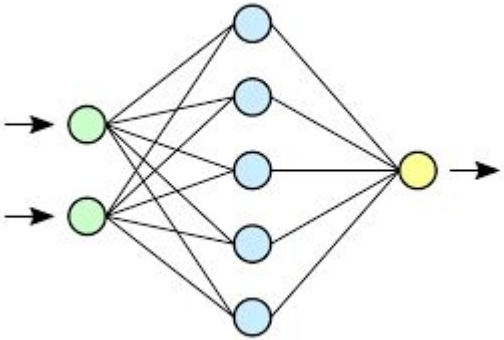
Rede neural Biológica

A arquitetura das redes neurais biológicas ainda é objeto de pesquisa ativa, mas algumas partes do cérebro já foram mapeadas, e parece que os neurônios muitas vezes estão organizados em camadas consecutivas.



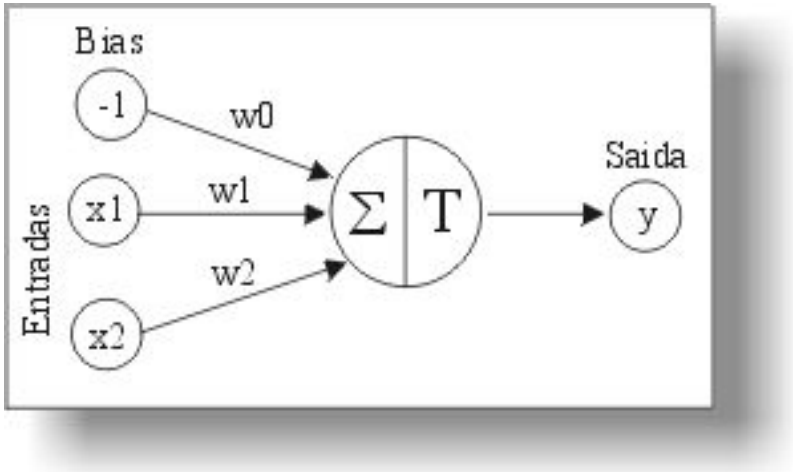
Neurônio Artificial

Warren McCulloch e Walter Pitts propuseram um modelo muito simples do neurônio biológico, que mais tarde se tornou conhecido como um neurônio artificial: ele tem uma ou mais entradas binárias (ligado/desligado) e uma saída binária. O neurônio artificial simplesmente ativa sua saída quando mais do que um certo número de suas entradas estão ativas. McCulloch e Pitts mostraram que mesmo com um modelo simplificado como esse, é possível construir uma rede de neurônios artificiais que calcula qualquer proposição lógica desejada



LTU (Limiar Threshold Unit)

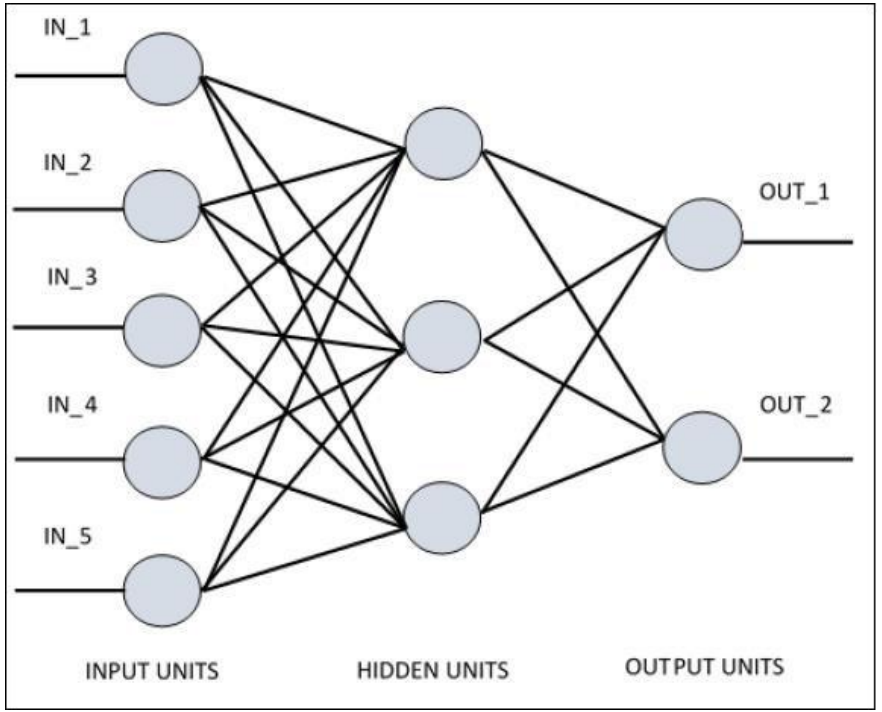
O LTU calcula a soma ponderada de suas entradas, aplica uma função de passo e produz uma saída. A fórmula matemática usada é z = w1 x1 + w2 x2 + ... + wn xn = wT · x, onde z é a soma ponderada, w é o peso de cada entrada, x é a entrada e wT · x é o produto escalar entre os vetores de pesos e entradas.



$$\text{heaviside}\,(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases} \qquad \text{sgn}\,(z) = \begin{cases} -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \\ +1 & \text{if } z > 0 \end{cases}$$

Perceptron

O Perceptron é composto simplesmente por uma única camada de LTUs, com cada neurônio conectado a todas as entradas. Essas conexões são frequentemente representadas usando neurônios de passagem especiais chamados neurônios de entrada: eles simplesmente produzem como saída qualquer entrada que receberem. Além disso, é adicionado um recurso adicional de viés (x0 = 1). Esse recurso de viés é geralmente representado usando um tipo especial de neurônio chamado neurônio de viés, que produz 1 o tempo todo.



Como é treinado?

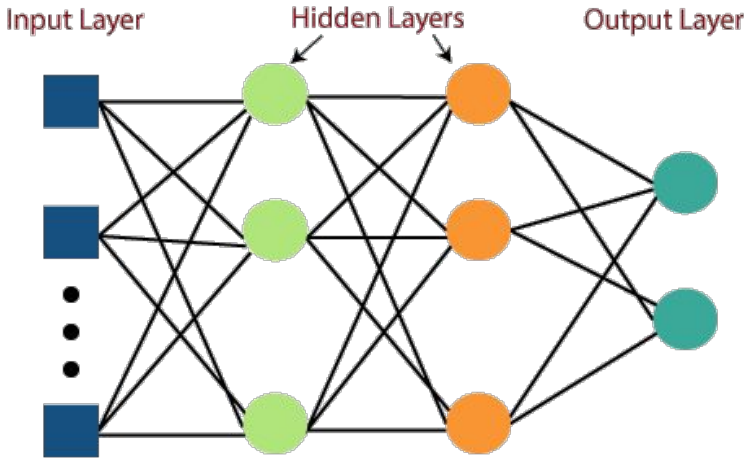
Damos uma entrada para o Perceptron, e verificamos a saída dele. Comparamos essa saída com o valor que esperávamos. Se a resposta estiver incorreta, para cada neurônio (LTU) de saída que produziu uma previsão incorreta, reforçamos os pesos das conexões a partir das entradas que teriam contribuído para a previsão correta.

Equation 1-2. Perceptron learning rule (weight update)

$$w_{i,j}^{(\text{next step})} = w_{i,j} + \eta(y_j - \hat{y}_j)x_i$$

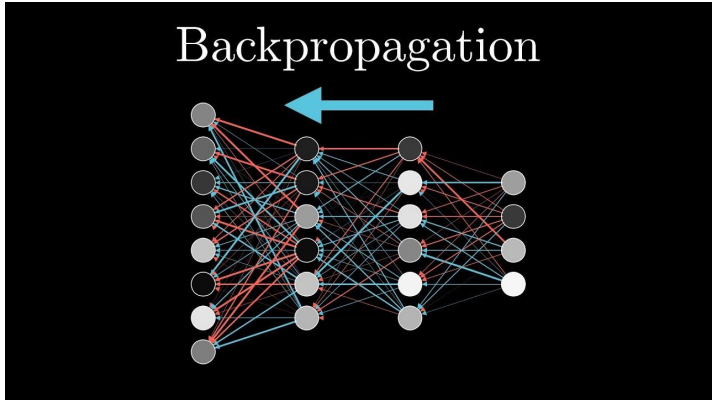
Redes Neurais Artificiais

Uma arquitetura de rede neural muito famosa é o Multi Layer Perceptron (MLP), é composta por uma camada de entrada (de passagem), uma ou mais camadas de LTUs chamadas camadas ocultas e uma camada final de LTUs chamada camada de saída (veja a figura 1-7). Cada camada, exceto a camada de saída, inclui um neurônio de viés e está totalmente conectada à próxima camada. Quando uma ANN possui duas ou mais camadas ocultas, é chamada de rede neural profunda (DNN). O MLP usa a técnica de retropropagação (backpropagation) para ajustar os pesos da conexão durante o treinamento.



Como um MLP é treinado?

Por muitos anos, os pesquisadores lutaram para encontrar uma maneira de treinar MLPs, sem sucesso. Mas em 1986, D. E. Rumelhart et al. publicaram um artigo revolucionário introduzindo o algoritmo de treinamento backpropagation. Hoje, descrevemos esse algoritmo como Descida de Gradiente usando autodiferenciação de modo reverso.

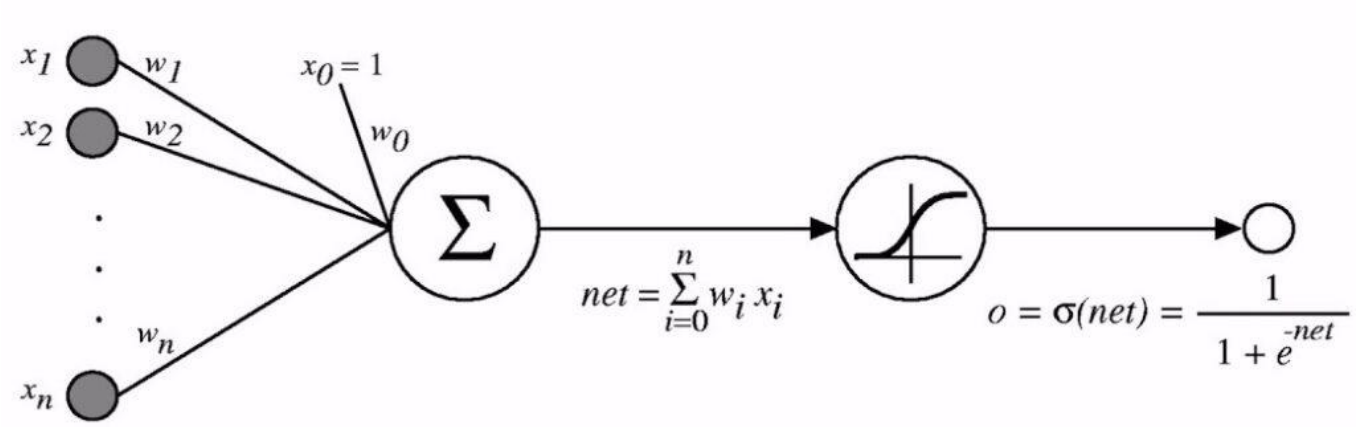


Como um MLP é treinado?

Resumindo, para cada instância de treinamento, o algoritmo de backpropagation primeiro faz uma previsão (passe para frente), mede o erro, depois passa por cada camada em ordem inversa para medir a contribuição de erro de cada conexão (passe reverso) e, finalmente, ajusta ligeiramente os pesos de conexão para reduzir o erro (passo de Descida de Gradiente).

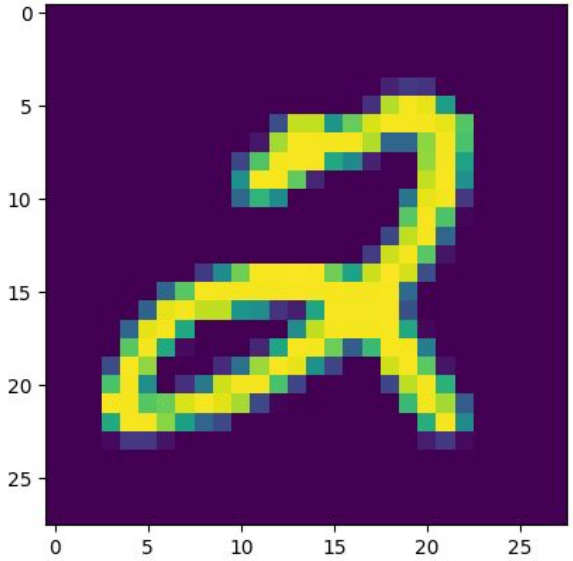
Curiosidade para os matemáticos de plantão

Para que este algoritmo funcione corretamente, os autores fizeram uma mudança fundamental na arquitetura do MLP: eles substituíram a função de etapa pela função logística, σ (z) = 1 / (1 + exp (-z)). Isso foi essencial porque a função de etapa contém apenas segmentos planos, portanto, não há gradiente para trabalhar (Descida de Gradiente não pode se mover em uma superfície plana), enquanto a função logística tem uma derivada bem definida e não nula em todos os lugares, permitindo que a Descida de Gradiente faça algum progresso a cada passo. O algoritmo backpropagation pode ser usado com outras funções de ativação, em vez da função logística.



Vamos para o código!
Deep Learning

Multi layer Perceptron dizendo que número está escrito



Aplicações do Deep Learning

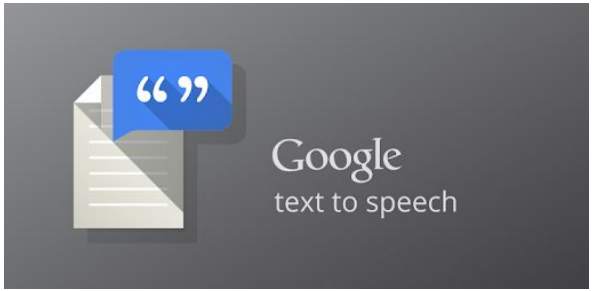
Visão Computacional: A visão computacional envolve ensinar máquinas a interpretar e compreender dados visuais, como imagens e vídeos. Técnicas de aprendizado profundo, como redes neurais convolucionais (CNNs), são amplamente utilizadas em tarefas de visão computacional, como classificação de imagens, detecção de objetos e segmentação de imagens.

Aplicações do Deep Learning

Processamento de Linguagem Natural (PLN): O processamento de linguagem natural envolve ensinar máquinas a entender e gerar linguagem humana. Técnicas de aprendizado profundo, como redes neurais recorrentes (RNNs) e transformers, são amplamente utilizadas em tarefas de PLN, como tradução de idiomas, modelagem de linguagem e análise de sentimento.

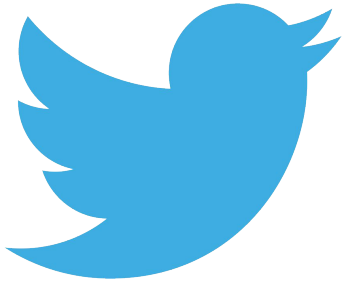
Aplicações do Deep Learning

Reconhecimento de Fala: O reconhecimento de fala envolve ensinar máquinas a reconhecer e entender a fala humana. Técnicas de aprendizado profundo, como redes neurais profundas (DNNs) e redes neurais recorrentes (RNNs), são amplamente utilizadas em tarefas de reconhecimento de fala, como transcrição de fala para texto e reconhecimento de locutor.



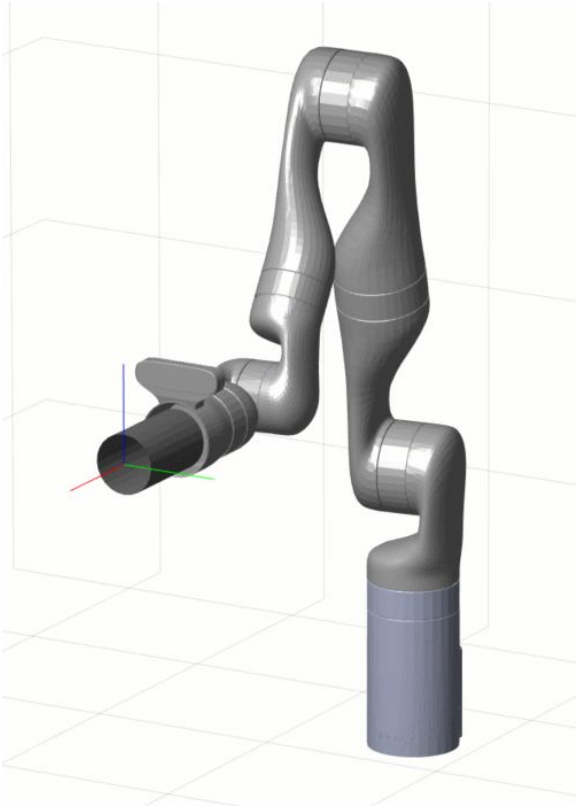
Aplicações do Deep Learning

Sistemas de Recomendação: Os sistemas de recomendação envolvem a predição das preferências do usuário e a recomendação de itens com base nessas preferências. Técnicas de aprendizado profundo, como filtragem colaborativa e autoencoders profundos, são amplamente utilizadas em sistemas de recomendação.



Aplicações do Deep Learning

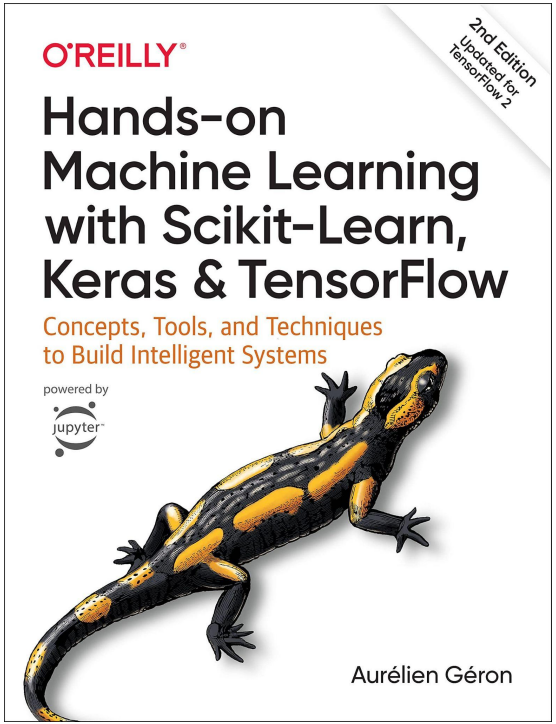
Robótica: A robótica envolve ensinar máquinas a perceber e interagir com o ambiente ao seu redor. Técnicas de aprendizado profundo, como aprendizado por reforço e redes neurais profundas, são amplamente utilizadas em tarefas de robótica, como navegação e manipulação de robôs.

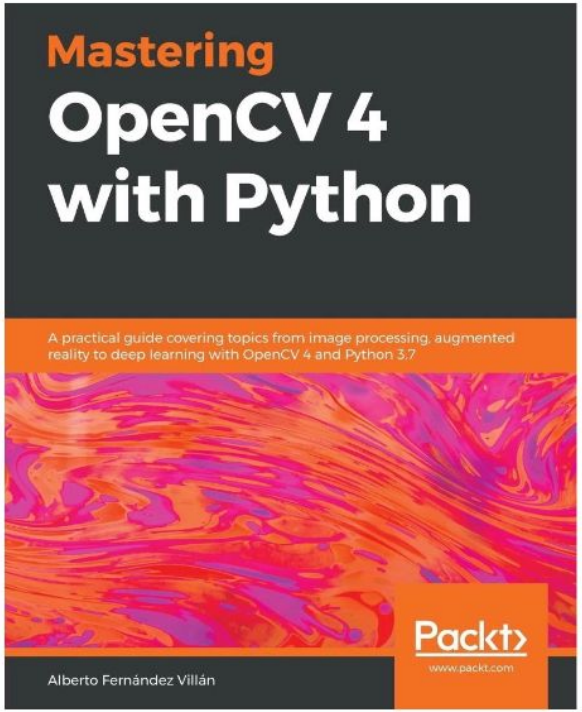


Aplicações do Deep Learning

Entre muitas outras!

Por onde começar a estudar IA?





Referências

- Géron, A. (2017). Hands-On Machine Learning with Scikit-Learn and TensorFlow. O'Reilly Media.
- ChatGPT. (2021), https://openai.com/
- https://intellipaat.com/blog/data-science-apriori-algorithm/
- https://www.tensorflow.org/

Por hoje é isso!

Obrigado pela presença!