FGA0221 - Inteligência Artificial

Portfólio 06

Fevereiro, 2025

Universidade de Brasília

Faculdade UnB Gama

Tema do portfólio:

Aprendizado de Máquina.

Aluno: João Matheus de Oliveira Schmitz
Matrícula: 200058525
Turma: T01
Semestre: 2024.2

Fevereiro, 2025

Sumário

Fevereiro, 2025

1. Aprendizado de Máquina

Os agentes de inteligência artificial são usados para resolver inúmeros problemas e situações, desde um aspirador de pó automático até diagnósticos médicos. Entretanto, em muitas situações os programadores responsáveis por um agente são incapazes de prever todas as formas que o mundo pode evoluir ou de programar uma solução por conta própria. Nesses casos, a resposta encontrada foi **permitir que o próprio agente seja capaz construir e atualizar os modelos do problema**, melhorando seu desempenho conforme vai obtendo novos dados e experiências - ou seja, **permitir que a máquina aprenda**.

O aprendizado de máquina pode ser utilizado de vários modos, como no reconhecimento de voz, navegação autônoma e personalização de experiências de compra. Ele pode ser utilizado para **melhorar o desempenho de qualquer parte do Programa Agent**e de um agente de IA. As técnicas de aprendizado utilizadas irão depender de quatro fatores:

- Qual componente deve ser aprimorado (mapeamento condição-ação, modelo transitório, modelo sensorial...)?
- Que conhecimento prévio o agente possui?
- Que representação é usada para os dados e o componente?
- Quais feedbacks estão disponíveis para aprendizagem?

Para treinar um agente através do aprendizado de máquina, **é preciso utilizar uma grande quantidade de dados** de treinamento. Quanto mais dados estiverem disponíveis e quanto mais fielmente eles representarem o mundo real, melhor será o desempenho do agente após seu treinamento. Entretanto, **os dados de treinamento diferem dependendo do tipo de aprendizado utilizado**, o qual pode ser supervisionado, não supervisionado ou por reforço.

1.1. Aprendizado supervisionado

O aprendizado supervisionado consiste em treinar um agente através do uso de **conjuntos de dados rotulados** para classificar dados ou prever resultados com precisão. Segundo a Google Cloud [3], "Em termos simples, para treinar o algoritmo a reconhecer fotos de maçãs, alimente-o com fotos rotuladas como maçãs".

1.2. Aprendizado não supervisionado

Em contrapartida ao aprendizado supervisionado, os seus dados de treinamento são **conjuntos de dados não rotulados** - ou seja, o algoritmo não sabe qual a saída correta para um dado de entrada, sendo ele mesmo o responsável por tentar classificar os dados corretamente. Segundo a IBM [4], "Esses algoritmos descobrem padrões ocultos ou agrupamentos de dados sem a necessidade de intervenção humana...", o que os torna ideais para a análise exploratória de dados, por exemplo. Existem dois principais métodos de aprendizado não supervisionado: **transformação do conjunto de dados** e **Clustering**.

1.3. Aprendizado por reforço

O aprendizado por reforço é o que mais tem foco na frase "Aprender com a experiência", isso porque os agentes treinados desse modo **aprendem através de tentativa e erro**. Nesse caso, o agente realiza uma tarefa e, após seu término, recebe um feedback sobre como foi seu desempenho - uma recompensa caso tenha feito corretamente e uma punição (ou falta de recompensa) caso esteja errado.

2. Classificação e regressão

Um problema de aprendizado é categorizado a depender de suas saídas. **Quando a saída for de um conjunto finito de valores** (cachorro , gato ou pássaro, por exemplo), **o problema será de classificação** - ou seja, ele irá classificar os dados de entrada em diferentes categorias. **Quando a saída for um número** (como a temperatura de um local), **o problema será de regressão**.

Para obter o melhor desempenho, é comum que os algoritmos de aprendizado não utilizem os dados de treinamento "puros", ou seja, sem nenhuma mudança. Isso ocorre principalmente em problemas de classificação.

2.1. Pré-processamento

Como o nome já diz, pré-processamento consiste em **processar os dados de treinamento antes de utilizá-los** na aprendizagem do agente. Seu objetivo é retirar partes irrelevantes dos dados de entrada - ou seja, que não afetam a saída - para que as operações seguintes fiquem computacionalmente mais simples de serem realizadas.

Um exemplo pode ser feito no problema de classificação de imagens, nesse caso diferenciando um jacaré de um crocodilo. O pré-processamento irá então retirar da imagem tudo que não é o animal em questão, ou seja, irá retirar o "fundo" da imagem. Outra possibilidade é alterar o contraste das imagens, caso alguma característica essencial fosse mais identificável em certos níveis de contraste.

2.2. Extração de características

A extração de características é uma das formas de pré-processar dados de entrada com a intenção de **obter características específicas** desse dado, para assim poder posteriormente compará-las com as características de outros dados, tornando mais fácil a classificação.

Seguindo o exemplo da classificação de uma imagem entre imagens de jacaré ou crocodilo, o algoritmo de extração de características será responsável por medir e obter atributos como: comprimento, formato do focinho, presença

dos dentes de baixo se a boca estiver fechada, entre outros. A classificação será então realizada com base na comparação entre tais atributos.

2.3. Overfitting e underfitting

O overfitting e o underfitting são dois problemas que precisam ser levados em conta durante o treinamento de um agente. Eles são caracterizados por:

- **Overfitting**: Ocorre quando o agente é capaz de classificar com perfeição os dados de treinamento, porém, **ele se torna tão específico** para estes que, ao testá-lo com dados diferentes, seu desempenho provavelmente será bem abaixo do esperado.
- **Underfitting**: Ocorre quando o agente é incapaz de encontrar padrões nos dados de treinamento - ou seja, é incapaz de classificar os dados corretamente com suas operações atuais.

Para que o agente seja capaz de alcançar um bom desempenho em situações reais, é importante que nenhum desses problemas ocorra. Para isso é necessário que o treinamento encontre um meio-termo entre eles, tornando o agente capaz de encontrar padrões para classificação que sejam gerais o suficiente para funcionar com dados diferentes dos de treinamento.

3. Algoritmos de aprendizado supervisionado

3.1. K-nearest Neighbors

O K-nearest Neighbors é um algoritmo classificador, capaz de resolver problemas de classificação e de regressão, embora seu uso mais comum seja na classificação. Ele é, possivelmente, o mais simples entre os algoritmos de aprendizado. Isso ocorre pois seu modelo consiste somente no armazenamento dos dados de treinamento, devido que, ao verificar um novo dado, ele irá **atribuir a categoria mais comum entre 1 ou mais dos seus vizinhos mais próximos**, característica que nomeia este algoritmo.

Figura 01 - Novo dado sendo classificado pelo algoritmo K-nearest Neighbors

Como exemplo, podemos utilizar a figura acima (IBM)[5], a qual demonstra um novo dado sendo classificado com base nos vizinhos encontrados até determinada distância dele. Isso nos leva a dois parâmetros importantes para este algoritmo: a **distância máxima** e o **número de vizinhos relevantes**. O algoritmo leva em conta estes dois parâmetros para determinar quais são os vizinhos relevantes na categorização de um novo dado.

3.2. Modelos Lineares

Os modelos lineares realizam uma previsão usando uma função linear das características de entrada para separar diferentes categorias. Se pensarmos em uma única característica, teremos uma regressão linear. Estes modelos seguem a seguinte forma:

$$\hat{y} = w[0] * x[0] + w[1] * x[1] + \ ... \ + w[n] * x[n] + b$$

- $x[k]$ representa as **características** de uma amostra;
- $w[k]$ e b são **parâmetros dos modelos** aprendidos, específicos de cada problema;
- $\widehat{y}$ é a **predição** que o modelo faz.

Para realizar uma classificação com modelos lineares, utilizamos o valor de $\widehat{y}$, por exemplo: caso $\widehat{y} \geq 0$ então a categoria é cachorro, caso $\widehat{y} < 0$ então a categoria é gato. Caso existam mais de duas categorias, é comum utilizar a **abordagem um contra todos**, onde criamos uma função linear que separa de forma única uma categoria de todas as outras, criando uma função para categoria. A figura a seguir demonstra essa abordagem:

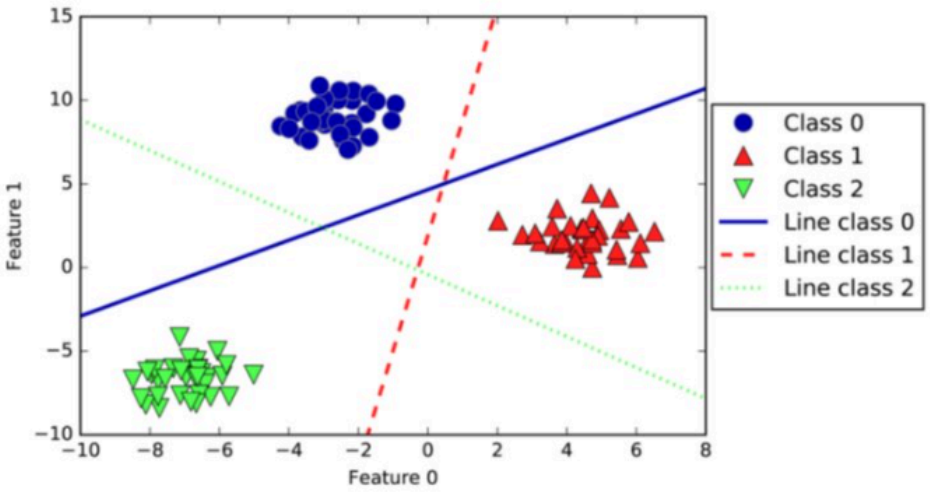

Figura 02 - Abordagem um contra todos

3.3. Classificadores Bayesianos

Os Classificadores Bayesianos Ingênuos são parecidos com os classificadores de modelos lineares vistos acima. Entretanto, eles têm uma vantagem e uma desvantagem em comparação a estes: eles geralmente são **mais rápidos** no treinamento, mas fornecem **um desempenho de generalização inferior**. Esses classificadores podem ser divididos em três tipos mais comuns:

- **Gaussiano**: é mais utilizado em dados contínuos e/ou com um grande número de dimensões;
- **Bernoulli**: assume dados binários e é mais utilizado em dados esparsos;
- **Multinormal**: assume dados contáveis, como a frequência que uma palavra aparece em uma frase, e também é utilizado em dados esparsos.

4. Redes neurais e aprendizado profundo

As redes neurais são modelos de aprendizado de máquina construídos para tomar decisões de forma parecida com o cérebro humano. Elas são **organizadas em camadas com vários "neurônios artificiais"** em cada, estes neurônios, por sua vez, são responsáveis por passar informações para neurônios conectados com ele na próxima camada. Entretanto, nem todos os neurônios são ativados ao mesmo tempo: dependendo da entrada, certos neurônios serão acionados, estes irão realizar uma computação dos dados de entrada e, se o valor de saída passar de um limiar definido de antemão, passarão a informação para os neurônios relevantes na próxima camada. (IBM)[6]

O aprendizado profundo (deep learning) é um subconjunto de modelos de aprendizado baseados em rede neural. Sua principal diferença para modelos de redes neurais comuns é simples: **a quantidade de camadas**. Enquanto algoritmos de redes neurais "comuns" são geralmente compostos de uma ou duas camadas, algoritmos de deep learning possuem três ou mais camadas (geralmente centenas ou milhares). (IBM, 2024)[7]

Uma importante distinção quando falamos de camadas em redes neurais, seja ela profunda ou não, é a de **camadas visíveis** e **camadas ocultas**. As visíveis são as camadas de input (recebe os dados de treinamento) e output (realiza a previsão ou classificação final dos dados). Já as ocultas, são aquelas que estão entre as visíveis, realizando a maioria do processamento dos dados.

Nos dias de hoje, agentes de inteligência artificial treinados com deep learning são muito comuns e, para os maiores modelos, **é necessário uma quantidade gigantesca de poder computacional**. Devido a isso, o uso de GPUs (Graphics Processing Unit, ou Unidade de Processamento Gráfico) para treinar agentes de IA cresceu explosivamente, tornando empresas como a Nvidia essenciais para o desenvolvimento de inteligência artificial de ponta e, consequentemente, uma das empresas mais valiosas do mundo.

Os algoritmos de deep learning podem ser usados em várias situações distintas, devido a isso, foram desenvolvidos **diferentes algoritmos para conjuntos de dados específicos**, como Redes neurais convolucionais (CNNs) e Redes neurais recorrentes (RNNs). Esses algoritmos podem ser utilizados para treinar modelos de IAs generativas, visão computacional, processamento de linguagem natural, entre outros.

5. Explainable Artificial Intelligence (XAI)

Os algoritmos de IA podem se provar complexos demais em seus raciocínios, dificultando o entendimento de usuários e fazendo-os questionar: **como o agente chegou a tal resposta?** Para remediar esse problema, foi desenvolvida a ideia de *Explainable AI*, que nada mais é do que uma série de métodos e processos que visam permitir que humanos sejam capazes de entender melhor e, consequentemente, confiar mais nos resultados alcançados pelo agente.

Existem muitas vantagens em um melhor entendimento do processo de raciocínio de uma IA. Algumas delas são:

- A capacidade de desenvolvedores confirmarem se o agente está trabalhando de forma correta ou não;
- Provar que o agente está atingindo possíveis padrões regulatórios;
- Possibilitar que o usuário ou desenvolvedor conteste a decisão/resposta do agente, caso necessário;
- Ter certeza que o resultado entregue pelo agente é confiável;
- A capacidade de desenvolvedores de monitorar o desempenho do agente com o tempo, necessidade criada devido a dados de teste serem diferentes de dados de produção;
- Além de reduzir riscos de segurança, complacência, entre outros.

Devido a essas vantagens, o uso de *Explainable AI* em certas áreas se prova essencial, principalmente por trazer uma **maior confiança no resultado do agente**. Algumas das áreas que se beneficiam são a assistência médica, serviços financeiros e a justiça criminal. (IBM)[8]

6. Algoritmos de aprendizado não supervisionado

6.1. K-means Clustering

K-means Clustering é um algoritmo relativamente simples de Clustering, ou seja, é utilizado para particionar dados em categorias distintas. Sua principal característica é o modo como ele separa os dados em categorias: ele busca o centro de cada cluster (categoria) de dados. Isso é feito através de duas etapas, repetidas *N* vezes:

- Atribuição de cada amostra de dados ao centro de cluster mais próximo;
- Definição do centro de cada cluster como a média das amostras atribuídas a eles.

O algoritmo inicia com uma atribuição (aleatória ou de amostragem inicial) de *K* centros de cluster (onde *K* representa a quantidade de categorias) e é finalizado quando os centros de clusters não apresentam mais mudanças significativas em sua posição. Segundo a IBM [9], "Esse tipo de análise de cluster é comumente utilizado em ciência de dados para segmentação de mercado, agrupamento de documentos, segmentação de imagens e compactação de imagens".

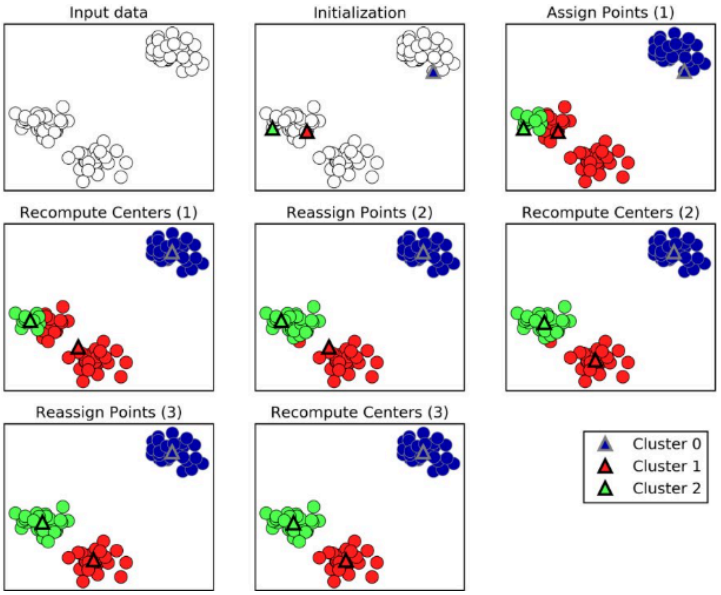

Figura 03 - Exemplo de funcionamento do algoritmo K-means Clustering

6.2. Self-organized Maps

Self-organized Maps (SOM) - também chamados de **Mapa de Kohonen**, em homenagem ao seu criador - são muito utilizados quando se tem datasets com muitos atributos e possivelmente uma saída com número muito menor de dimensões, geralmente duas (embora mais dimensões também sejam aceitas). **A saída é uma representação discreta dos dados de entrada** em um grid, seja ele quadrangular, hexagonal ou de outro formato. (ALVES, 2018)[10]

Os Mapas de Kohonen funcionam da seguinte forma:

- **Inicializar os pesos do SOM**: Ao iniciar, cada neurônio recebe um peso representando um função de alguns exemplos de treinamento que caíram ao seu redor, devido a isso, neurônios próximos têm pesos semelhantes;

A etapa acima é realizada somente no início, já as descritas abaixo são repetidas até que haja uma convergência ou o algoritmo realize uma quantidade máxima de iterações definida previamente:

- **Embaralhar os exemplos de treinamento**;
- **Para cada instância de treinamento X, encontrar o BMU correspondente**: Quando um exemplo de treinamento é inserido no grid, a Unidade de Melhor Correspondência (BMU, do inglês *Best Matching Unit*) é determinada através de competição - ou seja, os neurônios competem para ver quem será ativado por um dado de entrada - e o BMU é aquele com o peso mais próximo do valor do exemplo de treinamento;
- **Atualizar o vetor de peso do BMU e dos neurônios vizinhos**: É possível implementar um Mapa de Kohonen com inibições laterais, ou seja, após um BMU ser definido, ele tem a capacidade de reduzir a atividade dos seus vizinhos dando um feedback negativo a eles.

As figuras a seguir, retiradas de (MORIWAKI, 2022)[11] demonstram o grid inicial e final de um SOM (após 75 mil iterações), onde cada quadrado é um neurônio e as categorias dos dados de treinamento são mostradas através de cores:

- Laranja para dados com valor entre ~1,35 e 2;
- Vermelho para dados com valor entre ~0,65 e ~1,35;
- Verde para dados com valor entre 0 e ~0,65.

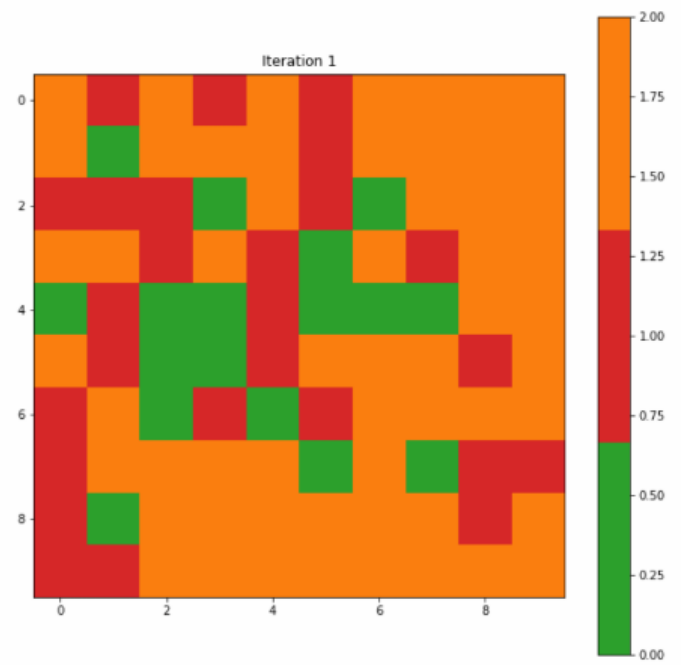

Figura 04 - Grid inicial de um SOM

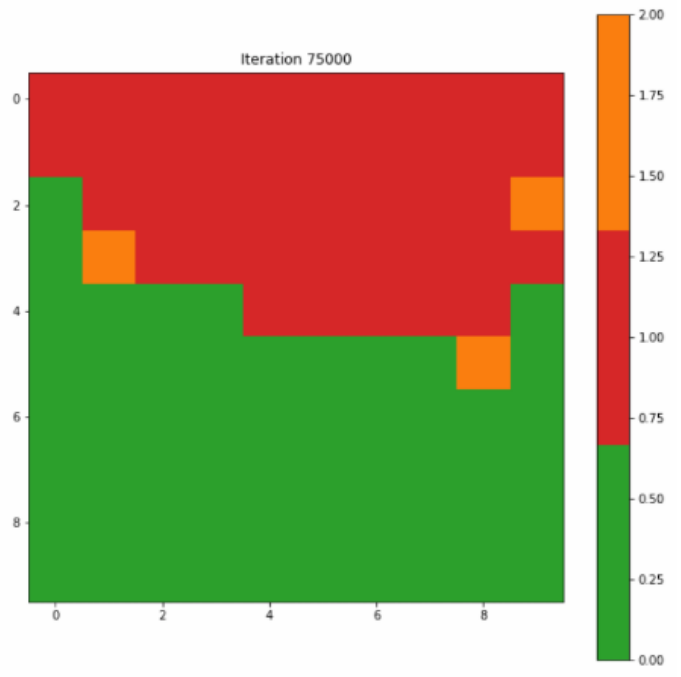

Figura 05 - Grid final de um SOM

7. Algoritmos de aprendizado por reforço

7.1. Q-Learning

No algoritmo Q-Learning, o objetivo é alcançar o estado com a maior pontuação possível, baseado na característica de recompensa e punição do aprendizado por reforço. Para isso, o algoritmo utiliza-se de uma matriz *R* com os valores de recompensa para cada ação tomada em um estado.

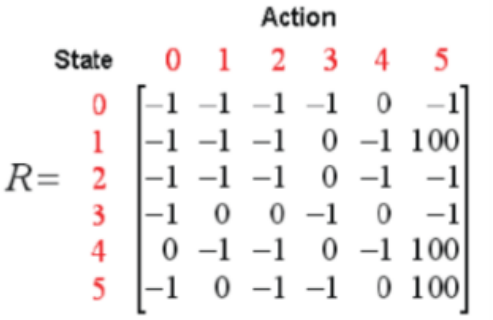

Figura 06 - Matriz de recompensa exemplo do Q-Learning

Na figura acima, a matriz *R* tem seis estados, onde cada um pode ou não ter uma transição para outro e o objetivo é ir de um estado qualquer para o estado 5. Para demarcar onde não existem transições, utilizamos o valor -1, desencorajando o agente de escolher tal ação. Além disso, para esse exemplo temos o valor 0 para transições comuns e o valor 100 para transições que levam ao estado desejado, mas esses valores são arbitrários.

Outra matriz utilizada pelo algoritmo é a matriz *Q*, que representa o que o agente aprendeu. Nela, as linhas representam o estado atual do agente e as colunas as ações que levam ao próximo estado, parecida com a matriz *R*. Como o agente inicia sem nenhuma informação, ela é uma matriz de zeros. Além disso, se o agente não souber a quantidade de estados possíveis, então a matriz é iniciada com tamanho 1x1 e vai aumentando a cada estado descoberto. A regra de transição para essa matriz é:

$$Q(estado,\ ação) = R(estado,\ ação) + Gamma * Max(proxEstado,\ todasAsAções)$$

"Ou seja, o valor de um elemento da matriz *Q* é igual a soma do valor correspondente na matriz *R* e o fator de aprendizado Gamma multiplicado pelo

valor máximo de *Q* para todas as possíveis ações no próximo estado". (SOARES)[1]

É importante comentar que o fator *Gamma* na equação tem valor entre 0 e 1. Se ele for próximo de 0, o agente tenderá a preferir somente recompensas imediatas e, se for próximo de 1, o agente irá considerar mais as recompensas futuras.

O funcionamento desse algoritmo ocorre através das seguintes etapas:

- **1.** Definir o parâmetro *Gamma* e a matriz *R* de recompensas;
- **2.** Iniciar a matriz *Q* com zeros;
- **3.** Para cada episódio (iteração do ponto de partida até o ponto de chegada):
  - **3.1.** Selecionar o ponto de partida aleatoriamente;
  - **3.2.** Repetir até chegar ao estado desejado:
    - **3.2.1.** Selecione uma das ações possíveis;
    - **3.2.2.** Considerando a ação escolhida, considere o que aconteceria no novo estado;
    - **3.2.3.** Calcule o valor Q máximo para as ações desse novo estado baseado em todas as ações possíveis;
    - **3.2.4.** Calcule $Q(estado,\ ação)$;
    - **3.2.5.** Atualize o estado novo como estado atual.

8. Impressões sobre o conteúdo

Ao introduzir aprendizado de máquina aos agentes vistos em outros portfólios, adicionamos a peça final e, provavelmente, a mais importante do quebra-cabeça que forma a chamada “inteligência” artificial, afinal, assim como dito no portfólio 01, a inteligência pode ser vista como o ato de “aprender com experiências passadas” e analisar o impacto de suas ações no futuro. Os agentes já eram capazes de aplicar a segunda parte desse conceito, mas ele só se torna completo quando o aprendizado entra em jogo.

Ao estudar sobre a evolução da inteligência artificial ao longo dos anos, fica claro que os agentes foram se tornando capazes de lidar com situações cada vez mais complexas, se tornando melhores até mesmo que humanos em algumas delas.

É incrível pensar que essa evolução ocorreu dentro de um intervalo de, aproximadamente, 80 anos - existem pessoas vivas atualmente que nasceram em uma época onde a própria inteligência artificial como área de estudo não existia. Essa consideração me leva a pensar: do que serão capazes os agentes de IA daqui a 40, 50 ou até 80 anos?

9. Referências

[1] SOARES, Fabiano Araujo. Slides da aula 21 à 25. Apresentação do PowerPoint.

[2] Russell, S. & Norvig, P. **Artificial Intelligence - A Modern Approach**. 3ª ed. Pearson Education, Inc. 2009.

[3] O que é machine learning (ML)?. **Google Cloud**, [s.d.]. Disponível em: https://cloud.google.com/learn/what-is-machine-learning?hl=pt-BR. Acesso em: 17 fev. 2025.

[4] O que é aprendizado de máquina (ML)?. **IBM**, [s.d.]. Disponível em: https://www.ibm.com/br-pt/topics/machine-learning. Acesso em: 17 fev. 2025.

[5] What is the k-nearest neighbors (KNN) algorithm?. **IBM**, [s.d.]. Disponível em: https://www.ibm.com/think/topics/knn. Acesso em: 18 fev. 2025.

[6] O que é uma rede neural?. **IBM**, [s.d.]. Disponível em: https://www.ibm.com/br-pt/topics/neural-networks. Acesso em: 18 fev. 2025.

[7] HOLDSWORTH, J. & SCAPICCHIO, M. O que é deep learning?. **IBM**, 2024. Disponível em: https://www.ibm.com/br-pt/topics/deep-learning. Acesso em: 18 fev. 2025.

[8] What is explainable AI?. IBM, [s.d.]. Disponível em: https://www.ibm.com/think/topics/explainable-ai. Acesso em: 18 fev. 2025.

[9] KAVLAKOGLU, Eda & WINLAND, Vanna. O que é agrupamento k-means?. **IBM**, 2024. Disponível em: https://www.ibm.com/br-pt/topics/k-means-clustering. Acesso em: 19 fev. 2025.

[10] ALVES, Gisely. Descobrindo SOM, uma Rede Neural com aprendizado não supervisionado. **Medium**, 2018. Disponível em: https://medium.com/neuronio-br/descobrindo-som-uma-rede-neural-com-aprendizado-n%C3%A3o-supervisionado-f22bc1e55eca. Acesso em: 19 fev. 2025.

[11] MORIWAKI, Ken. Understanding Self-Organising Map Neural Network with Python Code. **towards data science**, 2022. Disponível em: https://towardsdatascience.com/understanding-self-organising-map-neural-network-with-python-code-7a77f501e985/. Acesso em: 19 fev. 2025.