Inteligência Artificial e Álgebra Linear: a importância dos Sistemas Lineares na IA

Aline de L. Z. Lunkes[1] - alinelunkes@unipampa.edu.br
[1] Universidade Federal do Pampa, UNIPAMPA - Alegrete, RS, Brazil

Resumo.*A Inteligência Artificial (IA) está profundamente fundamentada na Álgebra Linear, uma vez que muitos dos algoritmos essenciais dessa área são baseados em conceitos matemáticos, como sistemas lineares do tipo* $A \cdot \mathbf{x} = \mathbf{b}$*. Este artigo explora a interdependência entre IA e Álgebra Linear, analisando como os sistemas lineares são aplicados em modelos de Machine Learning (ML), Redes Neurais Artificiais (RNAs) e processos de otimização. Além disso, discutimos o papel da Álgebra Linear na representação de dados e na atualização de parâmetros, evidenciando como essas técnicas são importantes para o desenvolvimento de modelos de IA eficientes e escaláveis.*

Keywords: *Inteligência Artificial, Álgebra Linear, Sistemas Lineares, Aprendizado de Máquina, Redes Neurais Artificiais*

1 Introdução

A Inteligência Artificial (IA) tem desempenhado uma revolução tecnológica nas últimas décadas, influenciando diversas áreas do conhecimento, desde a medicina até a engenharia. Seu avanço é impulsionado pelo desenvolvimento de algoritmos cada vez mais sofisticados, que dependem de conceitos matemáticos para seu funcionamento. Entre essas ferramentas matemáticas, a Álgebra Linear e o Cálculo Diferencial e Integral se destacam como um dos pilares fundamentais, fornecendo a base para representação de dados, otimização de parâmetros e implementação de modelos de aprendizado de máquina, em inglês *Machine Learning* (ML). Na Figura 1, apresenta-se uma ilustração conceitual das relações entre IA, ML e Redes Neurais Profundas, em inglês *Deep Neural Networks* (DNNs), por Haykin (2000); Goodfellow et al. (2016).

A partir da compreensão dos sistemas lineares do tipo $A \cdot \mathbf{x} = \mathbf{b}$, podemos explorar como as Redes Neurais Artificiais (RNAs) se beneficiam desses conceitos matemáticos. As RNAs são compostas por camadas interconectadas de neurônios, nas quais cada camada realiza operações matriciais, como multiplicações e somas, que podem ser expressas de forma compacta e eficiente. Essa representação matemática permite que os pesos das conexões entre os neurônios sejam ajustados durante o treinamento do modelo, otimizando a capacidade de prever e aprender

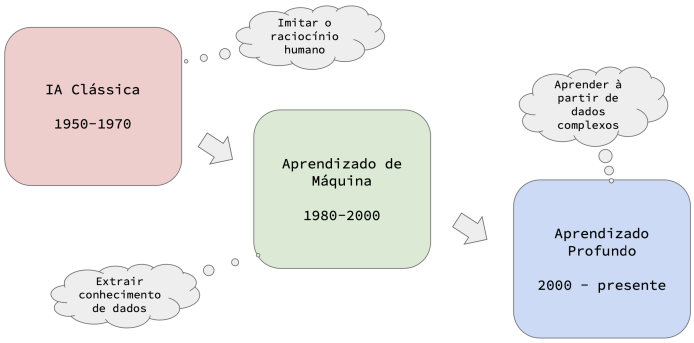


Figura 1: Esboço ilustrativo sobre Inteligência Artificial. Fonte: Autora.

com os dados, consultar Strang (2019). Técnicas como *backpropagation* e *gradient descent* dependem fortemente da Álgebra Linear para calcular os gradientes e atualizar os parâmetros das redes, permitindo que as RNAs aprendam com grandes volumes de dados e se adaptem a novas informações de forma dinâmica.

Além disso, os sistemas lineares são utilizados na IA, especialmente em problemas de Regressão Linear, RNAs e decomposição de matrizes, como a decomposição LU, em inglês *Factorization (Lower-Upper Decomposition*). A redução de dimensionalidade, através de técnicas como *Singular Value Decomposition* (SVD), simplifica os dados, mantendo as informações essenciais. Ao eliminar redundâncias e reduzir a complexidade dos dados de entrada, essa técnica torna o treinamento mais rápido e menos suscetível ao *overfitting*. Assim, a aplicação de operações matriciais e transformações lineares, aliadas ao poder da Álgebra Linear, é fundamental para o desenvolvimento de modelos de IA robustos e escaláveis, capazes de resolver problemas complexos de maneira eficaz e precisa.

No treinamento de RNAs, operações matriciais, como multiplicação de matrizes e álgebra tensorial, são fundamentais para o cálculo eficiente de pesos e gradientes, permitindo a atualização precisa dos parâmetros. Além disso, transformações lineares, como rotação, escalonamento e projeção, são utilizadas no processamento de imagens, construção de RNAs e análise de dados, ajustando as entradas para melhorar o desempenho dos modelos. O trabalho está estruturado da seguinte forma: na Seção 2, discutimos a representação matricial de dados, dividida em Subseção 2.1, que aborda a representação matricial de dados, e Subseção 2.2, que se concentra na representação matricial de dados em RNA. Finalizamos com considerações sobre a aplicação e a importância da Álgebra Linear em IA e ML.

2 Sistemas Lineares e Representação de Dados

A Álgebra Linear é fundamental para a IA e no ML, pois oferece as ferramentas matemáticas necessárias para representar, manipular e processar grandes volumes de dados de maneira eficiente. A utilização de vetores e matrizes é importante para armazenar e organizar informações, permitindo realizar operações matemáticas rápidas e eficazes. Além disso, as transformações lineares, como rotação, escalonamento e projeção, são essenciais para o processamento de imagens, RNAs, em inglês Artificial Neural Networks (ANNs), e análise de dados. A resolução de sistemas lineares, expressos pela Equação $A \cdot \mathbf{x} = \mathbf{b}$, são complexos em problemas de otimização, como Regressão Linear e ajuste de modelos estatísticos.

Outro aspecto importante da Álgebra Linear no ML é a redução de dimensionalidade, com métodos como a Decomposição em Valores Singulares (SVD) e a Análise de Componentes Principais, em inglês *Principal Component Analysis* (PCA), que ajudam a diminuir a complexidade dos dados, preservando informações relevantes. No treinamento de redes neurais, as operações matriciais, como multiplicação de matrizes e álgebra tensorial, são fundamentais para o cálculo de pesos e gradientes, permitindo a atualização eficiente dos parâmetros em modelos complexos. Dessa forma, a Álgebra Linear não apenas facilita a manipulação de grandes volumes de dados, mas também é um alicerce vital para a construção e aprimoramento de modelos de ML.

2.1 Representação Vetorial e Matricial

Um sistema linear é representado da forma Lima (2004); Hoffman and Kunze (1971)

$$A \cdot \mathbf{x} = \mathbf{b}, \tag{1}$$

onde:

- $A \in \mathbb{R}^{m \times n}$ é uma matriz de ordem $m \times n$, representando os coeficientes das variáveis do sistema. Cada linha de A corresponde a uma Equação linear, enquanto cada coluna representa os coeficientes das variáveis em uma Equação específica;
- $\mathbf{x} \in \mathbb{R}^{n \times 1}$ é o vetor coluna das variáveis desconhecidas, ou seja, as quantidades que queremos determinar ao resolver o sistema;
- $\mathbf{b} \in \mathbb{R}^{m \times 1}$ é o vetor coluna de constantes.

A solução do sistema linear consiste em encontrar o vetor $\mathbf{x}$ tal que a Equação (1) seja satisfeita. Dependendo da matriz A e do vetor $\mathbf{b}$, o sistema pode ter uma solução única, nenhuma solução ou infinitas soluções. Métodos de resolução incluem a eliminação de Gauss, a decomposição LU, a SVD, e a inversão de matriz (quando A é invertível). A decomposição LU fatora a matriz A no produto de duas matrizes triangulares, L (inferior) e U (superior), permitindo a resolução eficiente de sistemas lineares e facilitando o cálculo da inversa. Já a SVD expressa a matriz A como o produto de três matrizes: $U, \sum$, e V^T, em que U e V são matrizes ortogonais e $\sum$ é uma matriz diagonal que contém os valores singulares. Para mais detalhes, consulte Lima (2004); Strang (2019). Essa decomposição é utilizada na redução de dimensionalidade, na compressão de dados e na regularização de modelos, desempenhando um papel essencial em redes neurais profundas e algoritmos de reconhecimento de padrões.

2.2 Redes Neurais e Multiplicação Matricial

As RNA são modelos computacionais inspirados no funcionamento dos neurônios biológicos, capazes de aprender padrões e realizar previsões a partir de um conjunto de dados. Matematicamente, podem ser representadas como sequências de transformações lineares seguidas por operações não lineares. Considerando um sistema linear conforme a Equação (1), a matriz A pode ser interpretada como os pesos sinápticos do modelo, ou seja, $A = W$ determinando a contribuição de cada entrada para a saída, enquanto o vetor $\mathbf{b} = \mathbf{y}$ representa a saída do sistema. No entanto, para tornar o modelo mais expressivo, é necessário incluir um termo adicional: o viés (b), responsável por deslocar a transformação linear e permitir que a rede aprenda relações mais flexíveis, levando à Equação (2)

$$W \cdot \mathbf{x} + \mathbf{b} = \mathbf{y}, \tag{2}$$

onde:

- $W \in \mathbb{R}^{m \times n}$ representa a matriz de pesos sinápticos;
- $\mathbf{x} \in \mathbb{R}^{n \times 1}$ contém as características do problema;
- $\mathbf{b} \in \mathbb{R}^{m \times 1}$ é o vetor de viés, em inglês baias;
- $\mathbf{y} \in \mathbb{R}^{m \times 1}$ representa a saída da camada.

A matriz de pesos W pode ser escrita explicitamente como:

$$W = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \dots & w_{mn} \end{bmatrix}.$$

Cada linha de W representa os coeficientes que conectam um neurônio da camada atual a todas as unidades da camada anterior. A multiplicação matricial $W\mathbf{x}$ realiza a transformação linear das entradas $\mathbf{x}$ nas saídas $\mathbf{y}$:

$$\begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \dots & w_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}.$$

A saída y_i do i-ésimo neurônio pode ser expressa pela Equação (3):

$$y_i = \sum_{j=1}^{n} w_{ij} x_j + b_i. \tag{3}$$

Ou seja, a multiplicação de W (matriz de pesos) por $\mathbf{x}$ (entradas) gera $\mathbf{y}$ (vetor de saída). Cada peso w_{ij} controla o quanto cada entrada x_j contribui para cada neurônio i da camada de saída. Esses pesos são ajustados durante o treinamento da rede para minimizar o erro nas previsões e otimizar sua performance.

Para aumentar a capacidade da RNA, introduzimos funções de ativação σ, que aplicam não linearidades às transformações lineares, tornando a rede capaz de aprender relações não lineares, ver Goodfellow et al. (2016). Nos métodos de ML, os dados de entrada são frequentemente organizados em uma matriz $\mathbf{X} \in \mathbb{R}^{m \times n}$, em que cada linha $\mathbf{x}_i$ representa uma amostra (ou observação/instância) e cada coluna $\mathbf{x}_j$, corresponde a uma característica (ou atributo/variável). A transformação de dados na RNA segue a equação descrita em Eq. (4):

$$\hat{\mathbf{y}} = \sigma(W\mathbf{X} + \mathbf{b}), \tag{4}$$

onde:

- $\hat{\mathbf{y}}$ representa a previsão da saída $\mathbf{y}$;
- σ é uma função de ativação, responsável por introduzir não linearidades essenciais para a capacidade de aprendizado do modelo.
- $W\mathbf{X} + \mathbf{b}$ corresponde à combinação linear das entradas, dos pesos e do vetor de viés, frequentemente denotada por

  $$z = W\mathbf{X} + \mathbf{b},$$

  sendo z a entrada da função de ativação;
- $\sigma(\cdot)$ representa a função de ativação, responsável por introduzir não linearidades no modelo, permitindo que a rede neural aprenda relações complexas entre as variáveis de entrada e de saída. Entre as funções de ativação mais utilizadas destacam-se:
  - ReLU (Rectified Linear Unit): $\sigma(z) = \max(0, z)$;
  - Função sigmoide: $\sigma(z) = \frac{1}{1+e^{-z}}$;
  - Tangente hiperbólica: $\sigma(z) = \tanh(z)$.

Assim, DNNs podem modelar relações complexas e resolver problemas de alta dimensionalidade com eficiência. Na Figura 2, as entradas são representadas por $\mathbf{X} = \mathbf{x_j}$, correspondendo às características do problema, enquanto os pesos $\mathbf{W} = \mathbf{w_{ij}}$ são os parâmetros ajustáveis do modelo, responsáveis por determinar a influência de cada entrada na saída. O operador $\sum$ representa a combinação linear das entradas ponderadas pelos pesos, e a função de ativação σ introduz a não linearidade necessária para que a rede aprenda padrões complexos. Por fim, o neurônio de saída, representado por $\hat{\mathbf{y}}$, corresponde à previsão final do modelo.

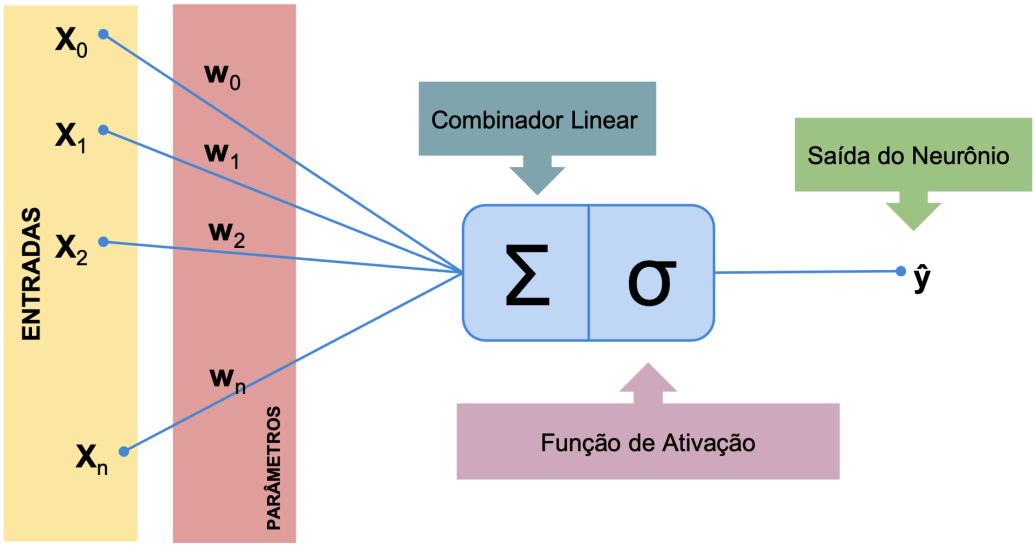


Figura 2: Exemplo de uma RNA com uma única camada de saída. Fonte: Autora.

As operações das equações (2) e (3) são essenciais para a propagação da informação e o ajuste dos parâmetros do modelo durante o treinamento. Devemos nos perguntar: como a

RNA aprende? O aprendizado em RNA ocorre por meio de um processo iterativo de ajuste de parâmetros. Inicialmente, os valores de entrada $\mathbf{X}$ são fornecidos ao modelo, e para cada entrada existe um valor esperado de saída $\mathbf{y}$. Os pesos $\mathbf{W}$ representam os parâmetros do modelo, que são ajustados ao longo do treinamento, ver Goodfellow et al. (2016). A IA gera uma previsão $\hat{\mathbf{y}}$ com base em $\mathbf{X}$ e $\mathbf{W}$, e o treinamento consiste em minimizar a diferença entre os valores reais $\mathbf{y}$ e os valores previstos $\hat{\mathbf{y}}$, processo conhecido como otimização. Durante essa etapa, os dados de treinamento alimentam a rede, que calcula o erro $E = \mathbf{y} - \hat{\mathbf{y}}$ e ajusta os parâmetros por meio de um processo iterativo de atualização

$$\mathbf{W}' = \mathbf{W} - \eta \frac{\partial E}{\partial \mathbf{W}},$$

onde η representa a taxa de aprendizado. Esse ajuste contínuo permite que a IA aprenda padrões e melhore suas previsões ao longo do tempo.

3 Conclusão

Neste trabalho, propomos destacar a importância fundamental da Álgebra Linear para o avanço da IA e do ML, fornecendo a base matemática essencial para lidar com grandes volumes de dados e trabalhar com alta dimensionalidade. Operações algébricas, como multiplicação matricial e resolução de sistemas lineares, são importantes para o treinamento e a manipulação de redes neurais complexas. A integração de transformações lineares com funções de ativação não lineares possibilita que as RNA aprendam funções complexas e resolvam uma vasta gama de problemas em aprendizado supervisionado. Além disso, a Álgebra Linear não só permite a criação de modelos mais precisos, mas também aprimora sua performance, contribuindo para avanços significativos na capacidade de generalização e na eficácia das soluções em IA. Em última instância, a Álgebra Linear permanece como o alicerce dessas tecnologias, sendo uma ferramenta indispensável para o desenvolvimento e o futuro dessas áreas.

Acknowledgements

Agradeço à agência de fomento CAPES, FAPEMIG pelo apoio financeiro e pelo incentivo à pesquisa.

Referências

Haykin, S. (2000), *Redes Neurais: Princípios e Prática*, São Paulo: Bookman.
Hoffman, K. and Kunze, R. (1971), *Linear Algebra*, Upper Saddle River, NJ: Pearson.
Lima, E. L. (2004), *Álgebra Linear*, Rio de Janeiro: IMPA.
Goodfellow, I., Bengio, Y. and Courville, A. (2016), *Deep Learning*, MIT Press. Disponível em: `https://www.deeplearningbook.org/`. Acesso em: 27 fev. 2025.
Strang, G. (2019), *Linear Algebra and Learning from Data*, Wellesley-Cambridge Press.