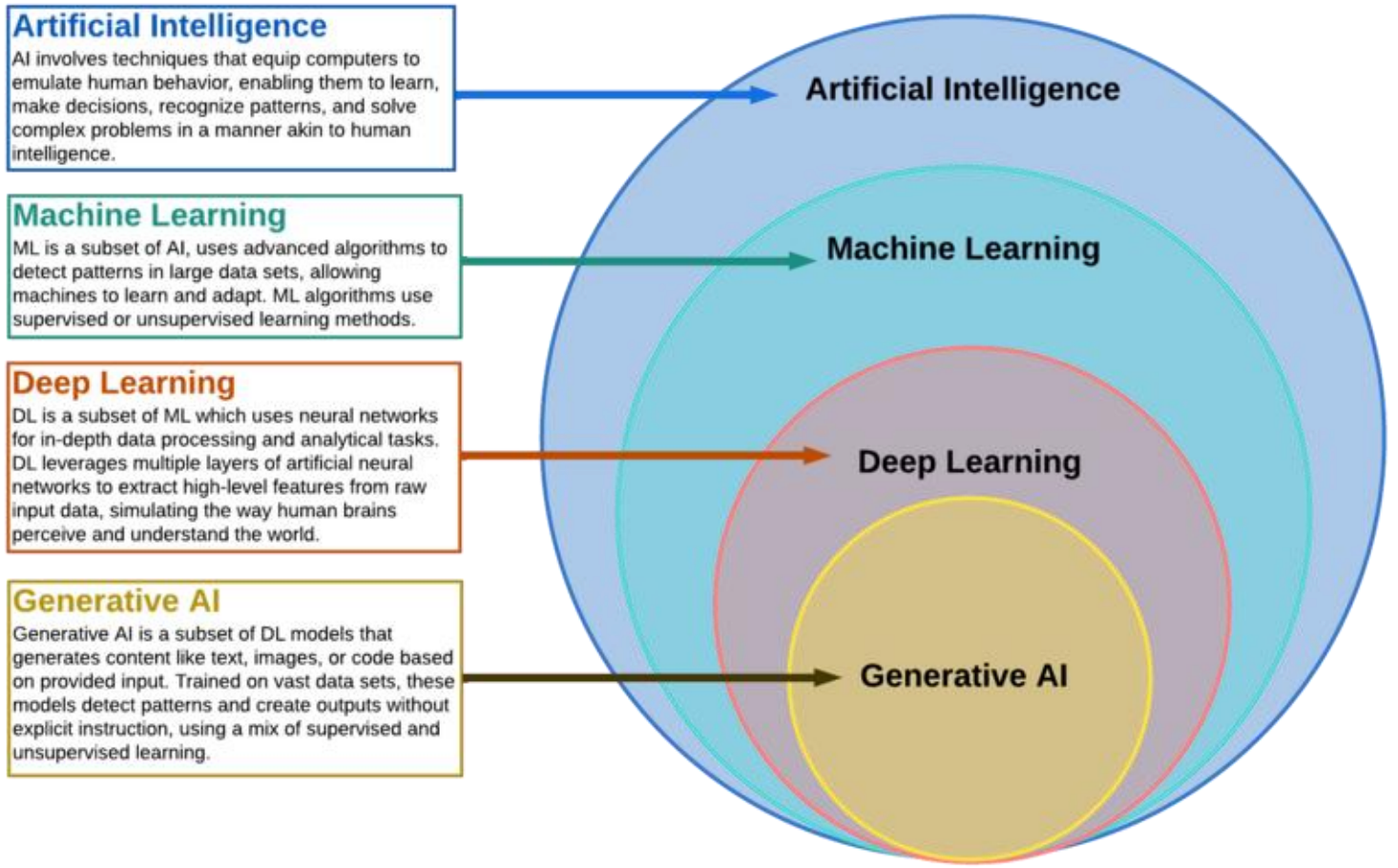
Panorama da IA

A IA envolve técnicas que equipam computadores para emular o comportamento humano, permitindo que aprendam, tomem decisões, reconheçam padrões e resolvam problemas complexos de maneira semelhante à inteligência humana.

O aprendizado de máquina (ML) é um subconjunto da IA que usa algoritmos avançados para detectar padrões em grandes conjuntos de dados, permitindo que as máquinas aprendam e se adaptem. Os algoritmos de ML utilizam métodos supervisionados ou não supervisionados.

O aprendizado profundo (DL) é um subconjunto do aprendizado de máquina que utiliza redes neurais para processamento de dados aprofundado e tarefas analíticas. O DL aproveita múltiplas camadas de redes neurais artificiais para extrair características de alto nível a partir de dados brutos, simulando a forma como o cérebro humano percebe e entende o mundo.

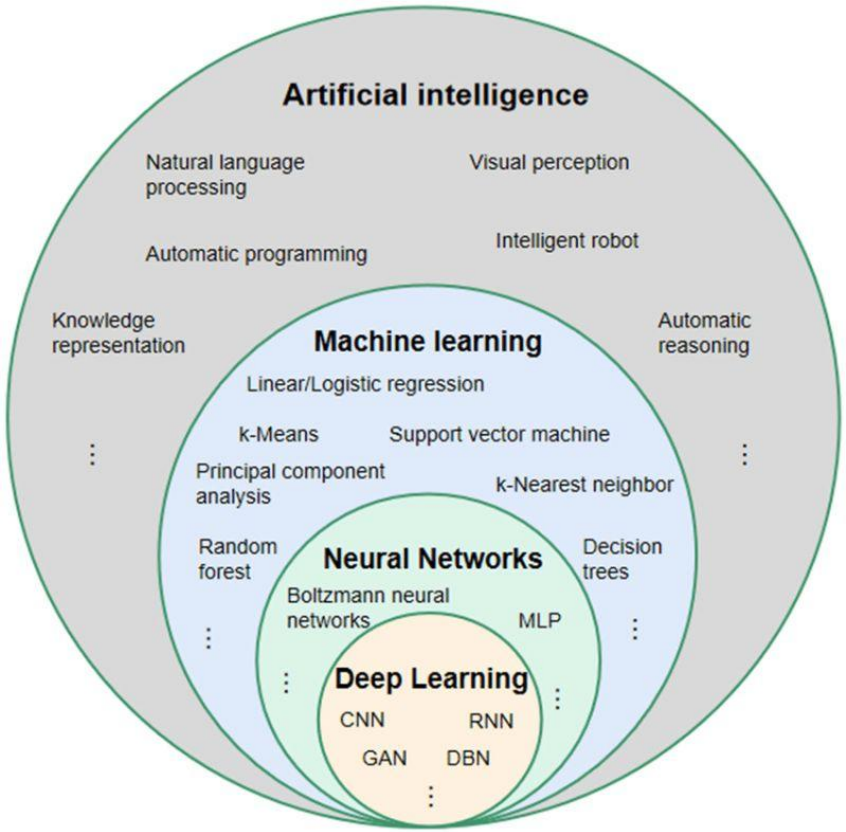
A IA generativa é um subconjunto dos modelos de aprendizado profundo que gera conteúdo como texto, imagens ou código com base na entrada fornecida. Treinados em grandes conjuntos de dados, esses modelos detectam padrões e criam saídas sem instruções explícitas, utilizando uma combinação de aprendizado supervisionado e não supervisionado.

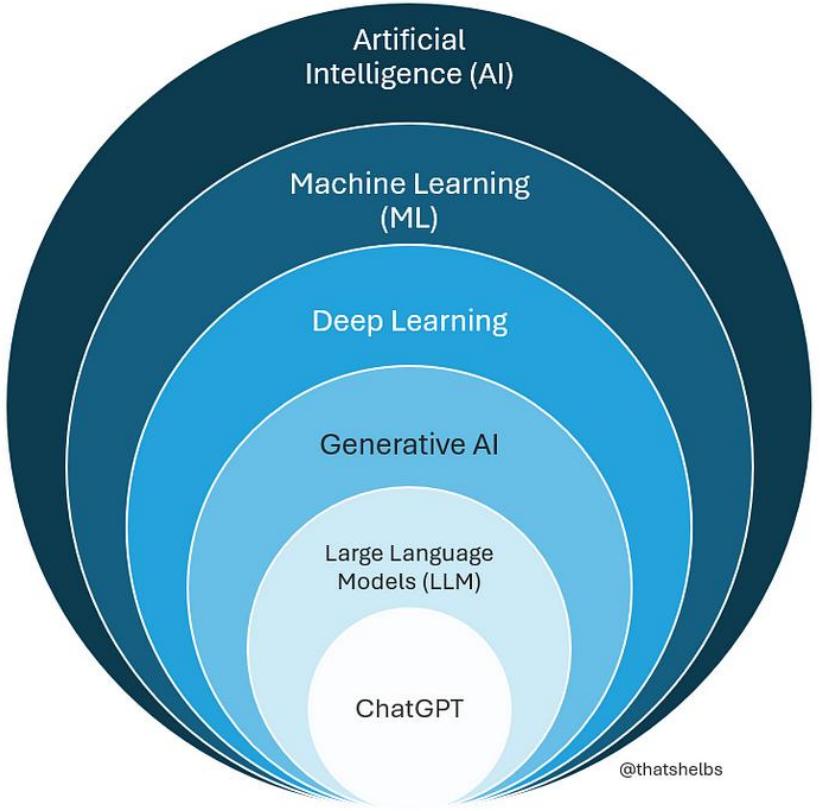


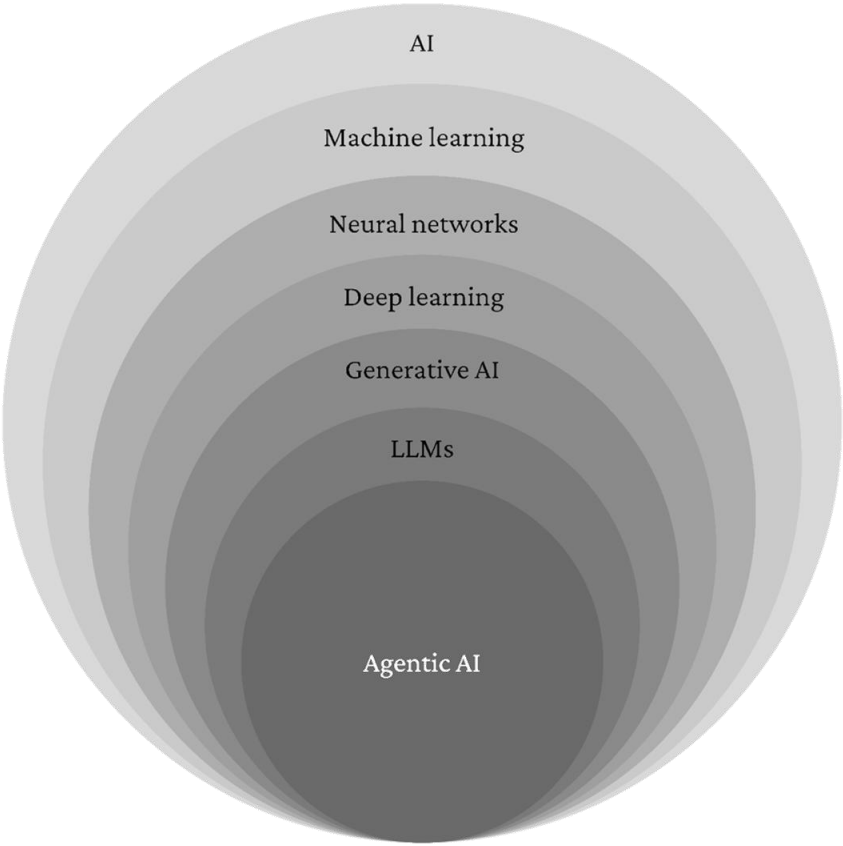
Unraveling AI Complexity - A Comparative View of AI, Machine Learning, Deep Learning, and Generative AI.

(Created by Dr. Lily Popova Zhuhadar, 07, 29, 2023)

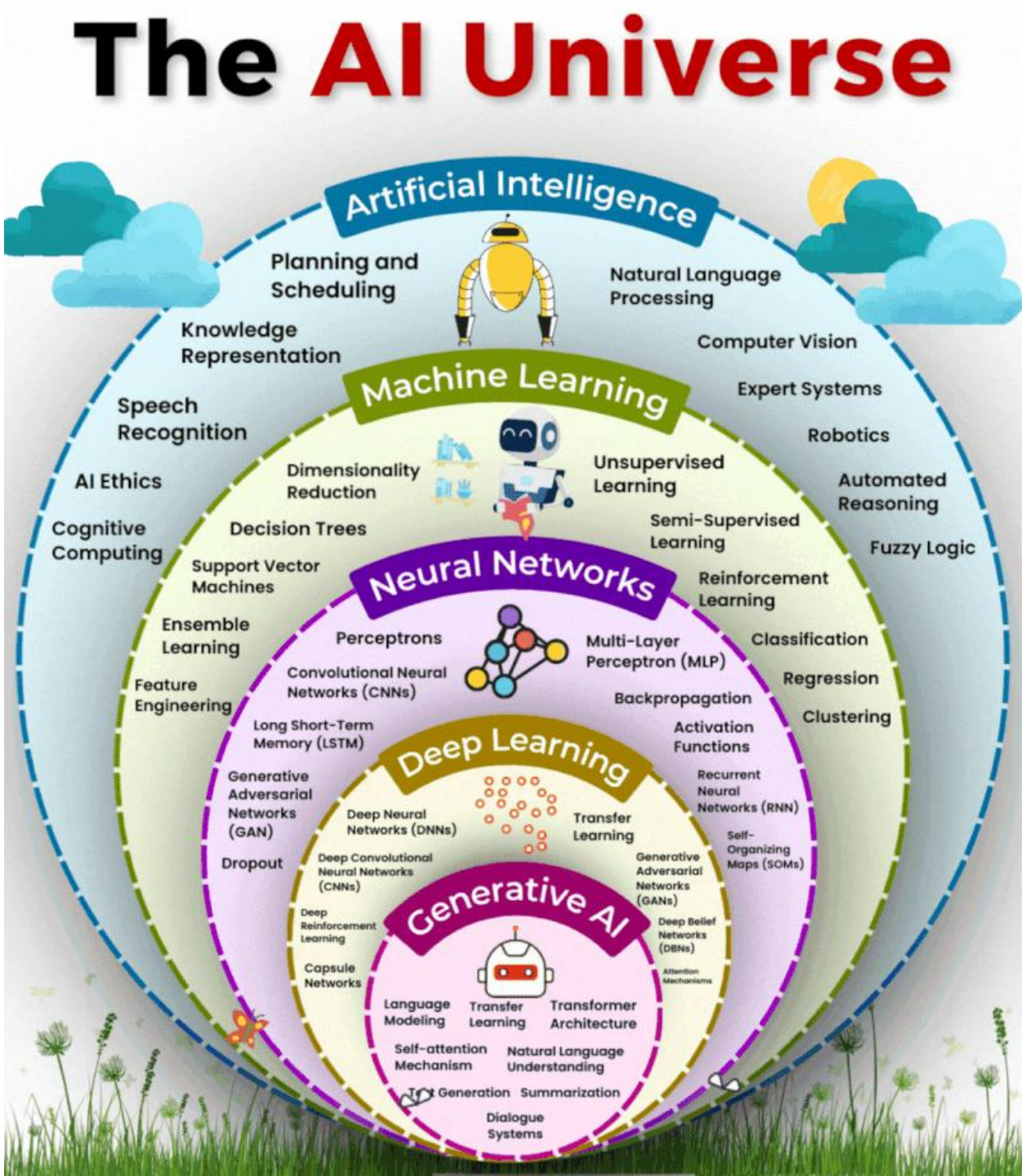
Panorama da IA



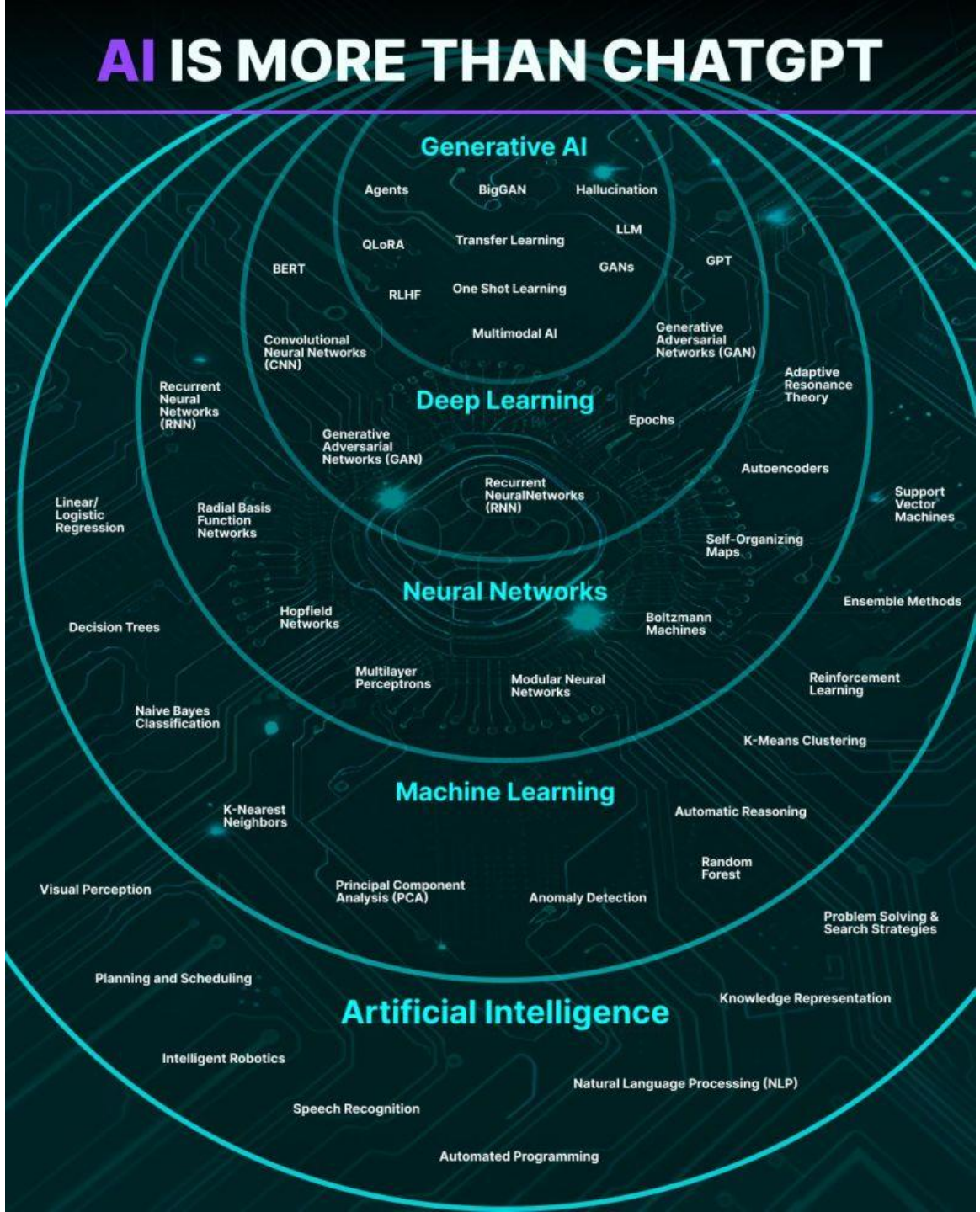


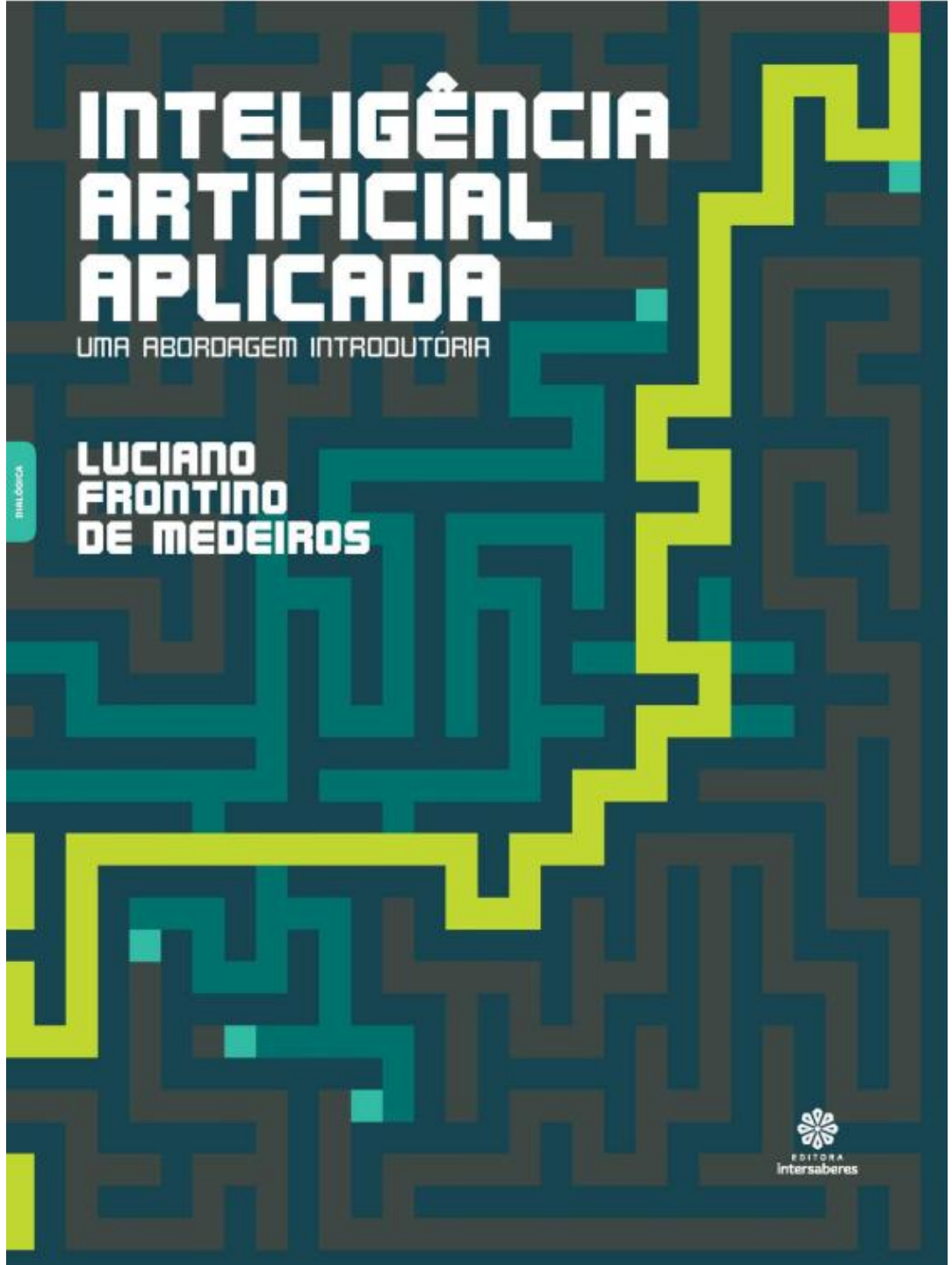


Panorama da IA



Panorama da IA





INTRODUÇÃO A REDES NEURAIS ARTIFICIAIS

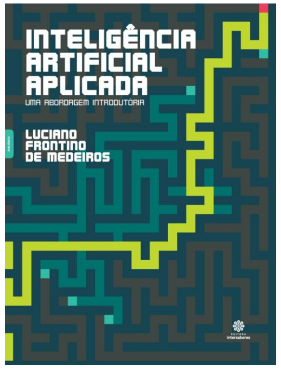
Introdução a Redes Neurais Artificiais

De uma perspectiva diferente da linha simbólica, as redes neurais artificiais (RNA) podem ser consideradas um dos adventos mais significativos da área da inteligência artificial (IA). Seu funcionamento possibilita o aprendizado de padrões que emerge com base na complexidade da interligação de elementos mais simples que simulam o comportamento dos neurônios.

De acordo com Haykin (2001, p. 27),

> O trabalho em RNA tem sido motivado desde o começo pelo reconhecimento de que o cérebro humano processa informações de uma forma inteiramente diferente de um computador convencional. O cérebro é um computador altamente complexo, não linear e paralelo. Possui a capacidade de organizar seus constituintes estruturais, os neurônios, de forma a realizar certos processamentos tais como reconhecimento de padrões, percepção e controle motor, de forma mais rápida que o mais rápido computador existente.

Com base na configuração de uma rede neural presente em um cérebro vivo, busca-se emular tal força de representação por meio de uma RNA. Apesar de os neurocientistas terem conseguido grandes avanços na pesquisa a respeito



Medeiros (2016, p. 127)

Introdução a Redes Neurais Artificiais

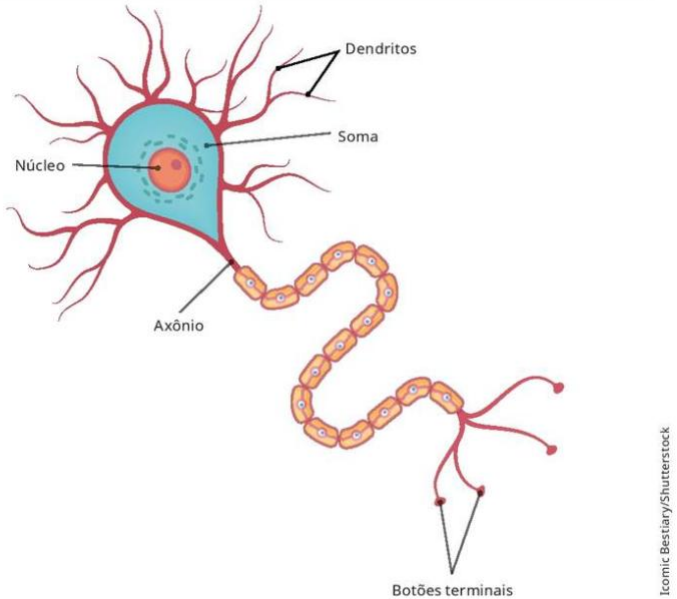
Com base na configuração de uma rede neural presente em um cérebro vivo, busca-se emular tal força de representação por meio de uma RNA. Apesar de os neurocientistas terem conseguido grandes avanços na pesquisa a respeito do funcionamento fisiológico de uma coleção de neurônios, essa ainda é uma tarefa desafiadora.

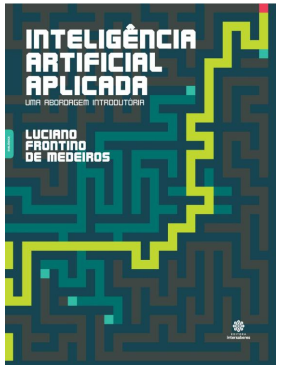
Os neurônios são as unidades fundamentais dos tecidos do sistema nervoso, incluindo o cérebro. Cada neurônio consiste de um corpo celular, também designado *soma*, que contém um núcleo. Do corpo da célula partem filamentos denominados *dendritos*. O filamento mais longo chama-se *axônio* (Figura 5.1). ao longo das conexões pelo axônio, chegando eventualmente a outras sinapses e liberando neurotransmissores no corpo de outras células (Gazzaniga; Ivry; Mangun, 2006).

Sinapses que incrementam o potencial de outras células são as **excitatórias**; as que decrementam esse potencial são as **inibitórias**. Os neurônios, por sua vez, podem formar novas conexões com outros neurônios, e tais mecanismos formam a base para o aprendizado do cérebro.

Uma rede neural biológica pode, dessa forma, ser abstraída para a simulação de seu comportamento. Assim, é possível conceituar uma *rede neural artificial* "como um processador maciçamente [e] paralelamente distribuído constituído de unidades de processamento simples, que têm a propensão natural para armazenar conhecimento experimental e torná-lo disponível para o uso" (Haykin, 2001, p. 28).

Figura 5.1 – Representação de um neurônio biológico

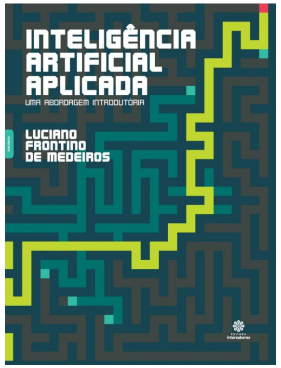




Medeiros (2016, p. 128)

Haykin (2001) reconhece seis propriedades que caracterizam uma rede neural artificial:

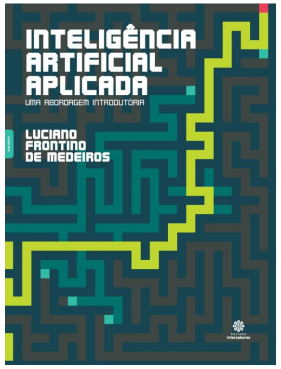
1. **Não linearidade** – Os neurônios podem ser lineares ou não lineares, permitindo aproximações robustas de funções de mapeamento que tenham característica não linear.
2. **Mapeamento de entrada-saída** – A rede aprende a partir de exemplos, estabelecendo mapeamento entre os padrões apresentados na entrada com as saídas dadas pelos exemplos.
3. **Adaptabilidade** – Redes neurais podem ser treinadas e armazenar o conhecimento nos pesos sinápticos, podendo adaptar-se caso o conjunto de amostras utilizado para o treinamento se modifique ao longo do tempo.
4. **Resposta a evidências** – Uma rede neural pode perfazer uma tarefa de seleção de um padrão, mas também informar sobre o grau de confiança ou crença no padrão escolhido.
5. **Informação contextual** – O conhecimento é armazenado na própria estrutura e pela ativação da rede neural.
6. **Tolerância a falhas** – Redes implementadas em *hardware* são tolerantes a falhas, em caso de neurônios ou conexões que possam ser danificados, ou mesmo em *softwares* que utilizem técnicas de poda de redes que reduzem a quantidade de neurônios ou sinapses, mantendo a mesma condição de *performance*.



Medeiros (2016, p. 129)

A construção de uma rede neural artificial, por seu turno, depende de quatro elementos:

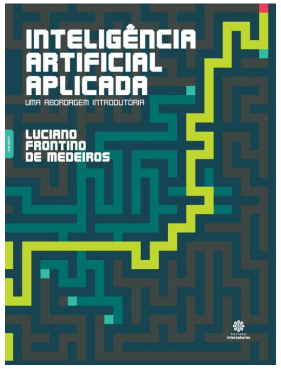
1. **Número de camadas** – As RNAs têm pelo menos uma camada de entrada, de onde recebem os sinais ou as características das amostras; e uma camada de saída, que apresenta os padrões ou as classes mapeados para os conjuntos de treinamento. Também podem apresentar uma ou mais camadas ocultas, como no caso do perceptron multicamada e das redes de base radial.
2. **Quantidade de neurônios em cada camada** – A quantidade de neurônios depende da natureza do problema abordado. A camada de entrada terá tantos neurônios quantas forem as características das amostras do conjunto de treinamento. A camada de saída terá os neurônios referentes às classes a que pertencem as amostras do conjunto de treinamento. A camada oculta ou escondida, por seu turno, pode ter uma quantidade variável de neurônios, conforme a característica do mapeamento que se deseja.
3. **Tipo de função de transferência** – A função de transferência define a forma como o neurônio é ativado. Para isso, podem ser utilizadas funções *discretas* (como a função *degrau*, empregada no perceptron, sobre o qual falaremos na Seção 5.1) ou funções *contínuas* (como a função *sigmoide* para o perceptron multicamada).
4. **Método de treinamento** – Ao longo dos anos, têm-se desenvolvido diversos métodos ou algoritmos de treinamento, entre quais o mais utilizado é o algoritmo de retropropagação (*backpropagation*), que aplica a informação do erro na atualização dos pesos.



Medeiros (2016, p. 130)

As RNAs podem ser aprendidas de diversas formas. Haykin (2001) enumera cinco modos de aprendizagem particularmente importantes:

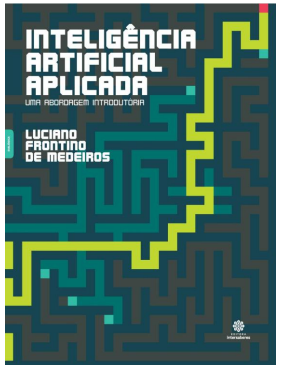
1. **Aprendizagem mediante correção de erros** – A informação do erro é utilizada para modificar os pesos sinápticos. Encontramos esse tipo de aprendizagem, por exemplo, no perceptron de Rosenblatt (1958) e no Adaline de Widrow e Hoff (1960).
2. **Aprendizagem baseada em memória** – Armazena-se um grande número de exemplos de entrada e saída, e uma amostra é comparada com a sua vizinhança, a fim de que se identifique a que classe pertence.
3. **Aprendizagem hebbiana** – Com base nos estudos de Hebb, emprega uma regra associativa, que aumenta a força dos pesos positivamente correlacionados ou diminui a daqueles negativamente correlacionados.
4. **Aprendizagem competitiva** – Os neurônios competem entre si para tornarem-se ativos. O vencedor estará ativo durante certo instante (aprendizagem também denominada *winner-takes-all*).
5. **Aprendizagem de Boltzmann** – Apresenta características estocásticas, derivando-se das ideias da mecânica estatística. Nesse caso, "os neurônios constituem uma estrutura recorrente e operam de uma maneira binária" (Haykin, 2001, p. 86), controlados por uma função de energia. A atualização dos pesos ocorre por correlação, operando em duas condições: uma presa, em que "os neurônios visíveis estão todos presos a estados específicos determinados pelo ambiente" (Haykin, 2001, p. 86); e outra livre, em que "todos os neurônios [...] podem operar livremente" (Haykin, 2001, p. 86).



Medeiros (2016, p. 130)

A aprendizagem pode ainda ser caracterizada como **supervisionada** (em que há o *feedback* que retorna à rede para orientar o treinamento) e **não supervisionada** (a rede aprende de forma auto-organizada). Entre as tarefas básicas que podem ser executadas por uma RNA estão:

1. **Associação de padrões** – Quando determinado conjunto de dados deve ser associado a outro conjunto. Exemplo: compressão de dados.
2. **Reconhecimento de padrões** – Quando um conjunto de padrões é associado a um identificador. Exemplo: reconhecimento de caracteres, reconhecimento de fala.
3. **Aproximação de funções** – Quando uma rede neural pode ser utilizada para regressão de dados. Exemplo: previsão de comportamento de ativos da bolsa de valores.
4. **Controle** – Quando uma rede neural executa uma função de controle de um sistema de forma automatizada, monitorando o *feedback* proveniente da sua saída. Exemplo: sistema de frenagem.
5. **Filtragem** – quando uma rede neural é utilizada para extração de ruído em um determinado sinal. Exemplo: filtragem adaptativa de sons.



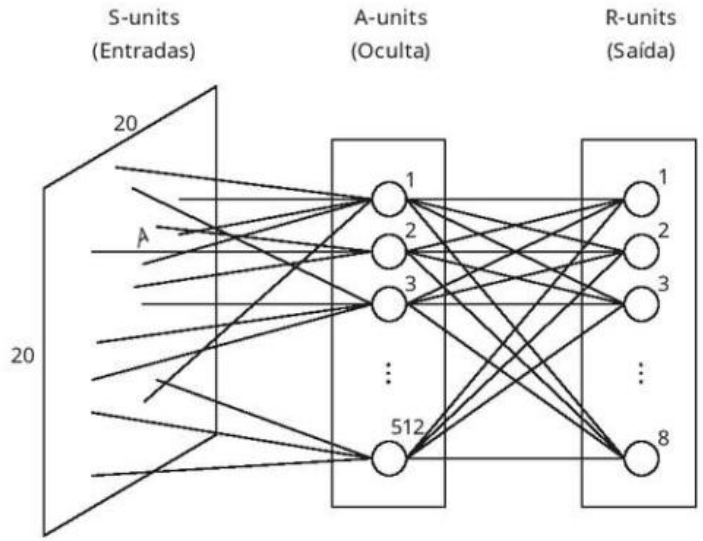
Medeiros (2016, p. 131)

5.1 Perceptron

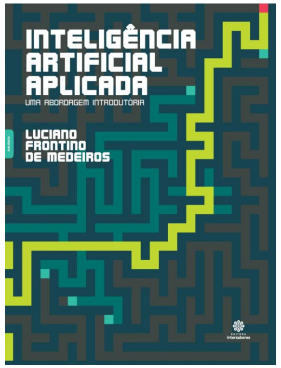
O perceptron é um dispositivo eletrônico inventado em 1957 por Frank Rosenblatt (1928-1971), psicólogo estadunidense, considerado uma espécie de "homem da Renascença, devido à sua excelência em várias áreas, incluindo [...] computação, matemática, neurofisiologia, astronomia e música" (Rosenblatt's Contributions, 2018, tradução nossa). O dispositivo foi construído de acordo com princípios biológicos e mostrava capacidade de aprendizado (Rosenblatt, 1958). Rosenblatt (1958) organizou-o em três camadas de unidades (ou neurônios): **sensoriais** (S), **associativas** (A) e **geradoras de respostas** (R). A camada de entrada com unidades sensoriais recebe os padrões visuais, a camada com as unidades associativas relaciona as unidades de entrada com as unidades geradoras de saída, funcionando como uma camada oculta, e a camada com unidades geradoras de respostas fornecem um resultado de acordo com os padrões apresentados à camada de entrada.

Rosenblatt (1958) desenvolveu também um perceptron implementado em *hardware* com 400 unidades fotossensoras, uma camada de associação de 512 unidades e outra de saída, de oito unidades. Trata-se do protótipo denominado *Mark I* (Figura 5.2).

Figura 5.2 – Esboço do perceptron Mark I



Fonte: Elaborado com base em Rosenblatt, 1958.

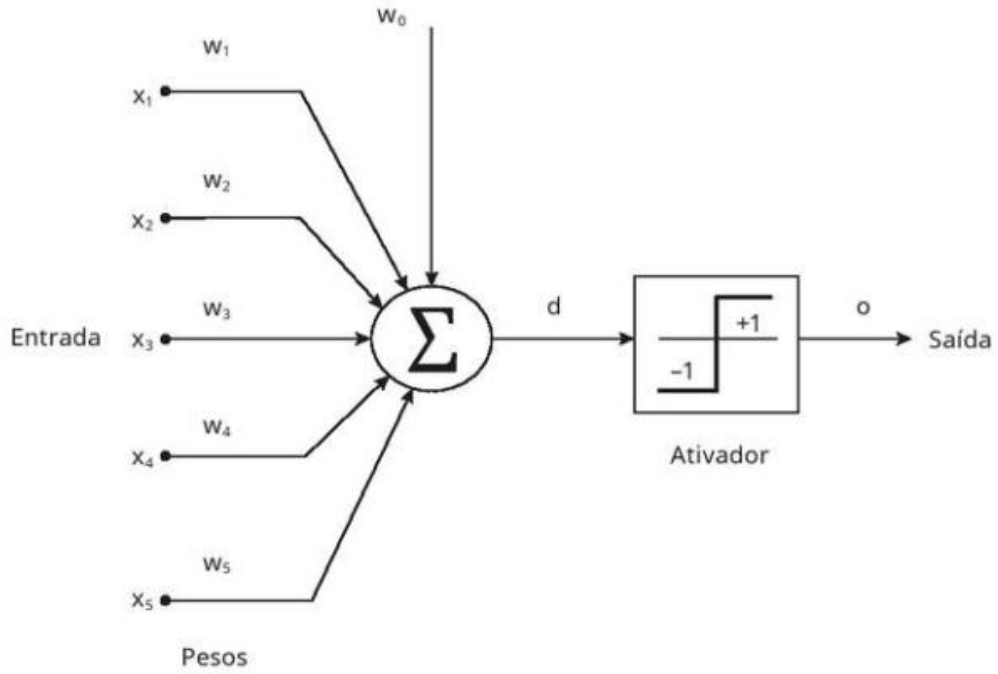


Medeiros (2016, p. 131)

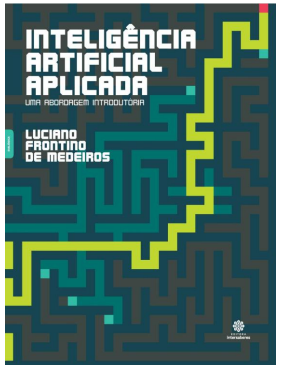
5.1 Perceptron

Assim, para poder efetuar uma classificação, o perceptron obtém os sinais do ambiente por meio das **entradas**, nas quais são apresentados valores correspondentes aos padrões que pretendemos classificar. Um exemplo seria os valores de *pixels* de imagem. Na Figura 5.3, podemos observar um perceptron de cinco entradas, cujo funcionamento será detalhado a seguir.

Figura 5.3 – Arquitetura de um perceptron



Nota: x = valores de entrada; w = valores dos pesos; d = saída intermediária; o = saída ativada.
Fonte: Medeiros, 2007.



Medeiros (2016, p. 131)

5.1 Perceptron

Quanto à estrutura interna do perceptron, ela é composta de **pesos** ou **sinapses**. Os pesos assumem valores tais que, quando aplicamos um padrão na entrada, obtemos uma saída intermediária[1] *d*. O **aprendizado** da rede fica armazenado nos pesos, e seus valores são obtidos mediante um processo de treinamento. O alor w_0, chamado de *bias*, é fixo e pode ser entendido como uma espécie de ajuste fino, que não se multiplica com entrada nenhuma. Esse modelo de neurônio é denominado *Neurônio de McCulloch-Pitts*[2].

A saída intermediária *d* é calculada por meio do somatório da multiplicação entre cada entrada e seu peso. Assim:

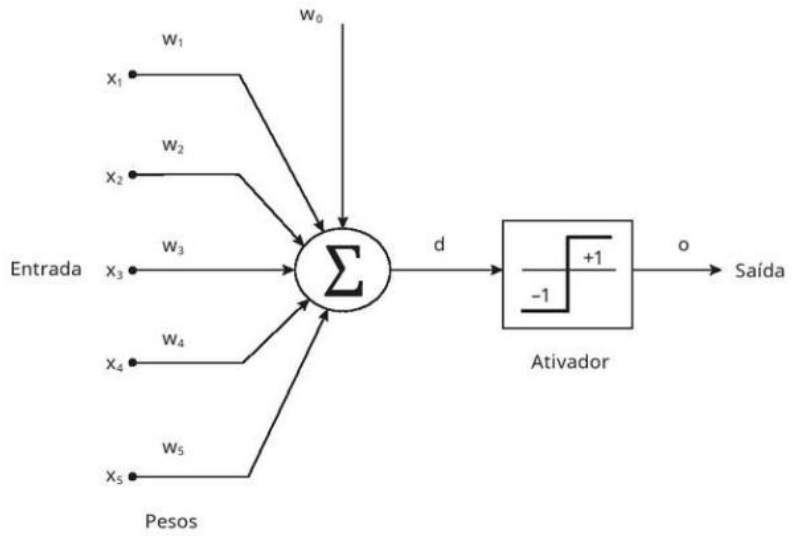
$$d = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_0$$

Ou, de forma genérica (com n = 5):

$$d = w_0 + \sum_{i=1}^{n} x_i w_i \qquad \textbf{1}$$

Após o cálculo da saída intermediária *d*, precisamos calcular a saída ativada *o* (do inglês *output*). Como visto na Figura 5.3, a saída ativada *o* depende do resultado da saída intermediária *d*. Caso o resultado obtido em *d* seja maior ou igual a zero, *o* será igual a *+1*; se *d* for menor que zero, *o* será igual a *–1*. Ou, em forma de pseudo-código:

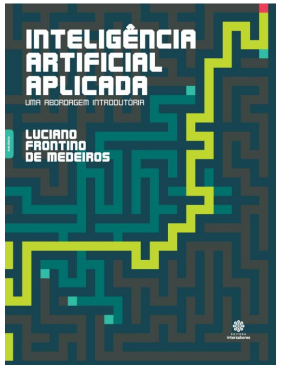
Figura 5.3 – Arquitetura de um perceptron



Nota: x = valores de entrada; w = valores dos pesos; d = saída intermediária; o = saída ativada.
Fonte: Medeiros, 2007.

1 Também chamada de *campo local induzido*.

2 O nome se deve ao trabalho pioneiro de McCulloch e Pitts, de 1943. Em tal modelo, a saída assume o valor 1, caso o campo local induzido seja positivo; e 0 (zero), em caso contrário (Haykin, 2001).



Medeiros (2016, p. 131)

Exercício: Calcule o resultado do Perceptron para as entradas abaixo

Caso 1

w0	w1	w2	w3	w4	w5	x1	x2	x3	x4	x5	d = w0 + ∑(wi * xi)	o (ativação)
0.5	-0.2	0.3	0.8	-0.5	0.1	0.4	-0.7	0.2	0.6	-0.9	?	?

Caso 2

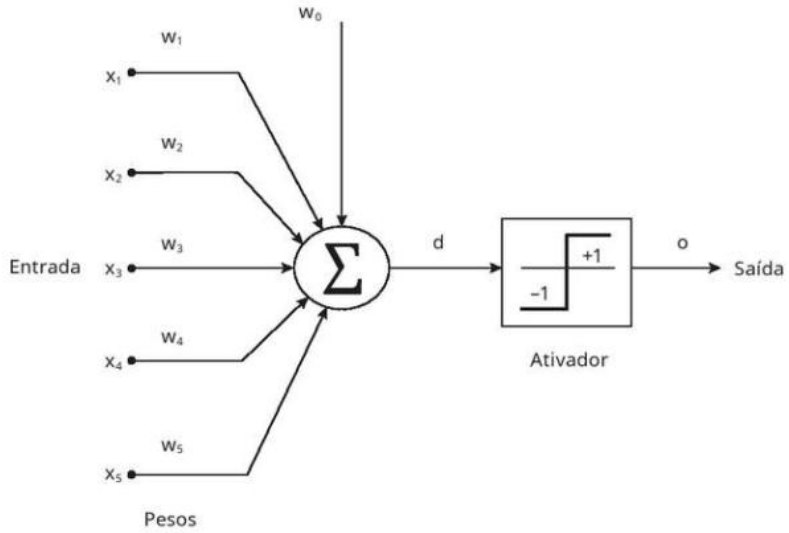
w0	w1	w2	w3	w4	w5	x1	x2	x3	x4	x5	d = w0 + ∑(wi * xi)	o (ativação)
-0.3	0.7	-0.1	0.5	-0.6	0.2	-0.5	0.3	0.8	-0.2	0.4	?	?

Caso 3

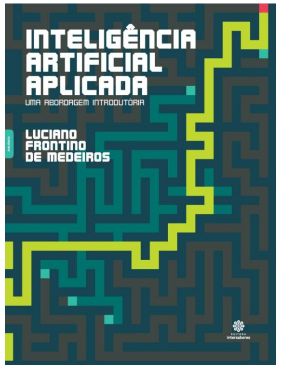
w0	w1	w2	w3	w4	w5	x1	x2	x3	x4	x5	d = w0 + ∑(wi * xi)	o (ativação)
0.2	-0.4	0.6	-0.3	0.5	-0.7	0.1	0.9	-0.6	0.2	-0.8	?	?

```python
def soma(w, x):
    d = w[0]  # Inicializa com o viés (bias)
    for i in range(1, 6):  # Itera de 1 a 5
        d += w[i] * x[i]  # Soma ponderada dos pesos e entradas
    return d

def ativacao(d):
    return 1 if d >= 0 else -1  # Função degrau (threshold)

# Exemplo de uso:
w = [0.5, -0.2, 0.3, 0.8, -0.5, 0.1]  # Pesos (incluindo o bias em w[0])
x = [1, 0.4, -0.7, 0.2, 0.6, -0.9]  # Entradas (x[0] normalmente é 1 para o bias)

d = soma(w, x)
o = ativacao(d)

print(f"Saída intermediária (d): {d}")
print(f"Saída ativada (o): {o}")
```

Figura 5.3 – Arquitetura de um perceptron



Nota: x = valores de entrada; w = valores dos pesos; d = saída intermediária; o = saída ativada.
Fonte: Medeiros, 2007.



Adaptado de Medeiros (2016, p. 133)

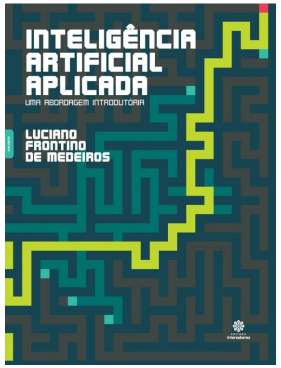
5.1.1 Exemplo de perceptron simples: classificação de parafusos

Para mostrar o uso do perceptron em uma tarefa simples de classificação, vejamos o exemplo da classificação de parafusos. Imaginemos que são fabricados parafusos de duas classes: A e B. Os critérios para classificá-los são o **comprimento** e o **diâmetro**. Suponhamos que, em média, os parafusos de comprimento próximo a 5 cm sejam os maiores (pertencentes à classe A) e aqueles que meçam em torno de 1 cm, os menores (pertencentes à classe B). Consideremos que parafusos produzidos são de diversos comprimentos dentro desses limites. O mesmo vale para a dimensão *diâmetro*: o maior diâmetro é 5 mm, e o menor, 1 mm, havendo diâmetros intermediários. O que o perceptron terá de informar é se um parafuso com certo comprimento e diâmetro é pertencente à classe A ou à classe B.

Recorreremos, para tal, ao treinamento por amostragem. A Tabela 5.1 mostra um conjunto de medidas de parafusos que serão utilizados para treinar o perceptron, ou seja, para calcular os pesos.

Parafuso	x_1	x_2
1	4 cm	3 mm
2	1 cm	1 mm
3	3 cm	2 mm
4	2 cm	3 mm
5	5 cm	3 mm
6	3 cm	2 mm

Tabela 5.1 – Dimensões das amostras de parafusos que serão utilizadas para o treinamento do perceptron

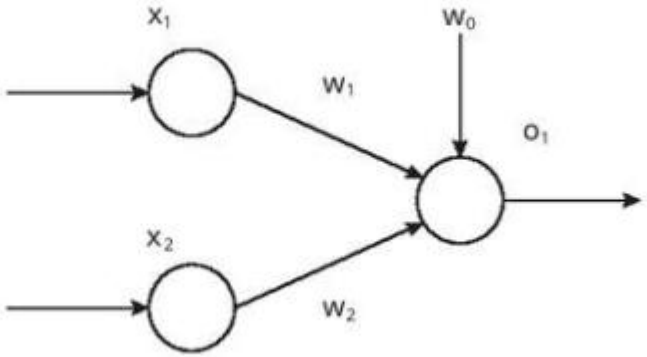


Medeiros (2016, p. 134)

5.1.1 Exemplo de perceptron simples: classificação de parafusos

Na Figura 5.4, temos uma representação simplificada da **arquitetura** (também chamada de *topologia*) desse perceptron, contendo duas unidades de entrada (x_1 e x_2), dois pesos (w_1 e w_2, com um peso fixo w_0) e uma unidade de saída (o_1) para inferir sobre a classe a que pertence o parafuso.

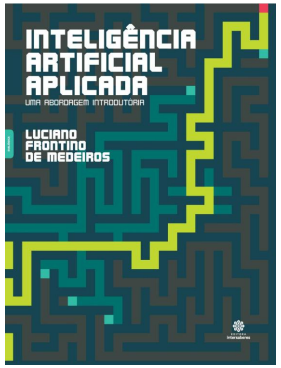
Figura 5.4 – Topologia do perceptron para a tarefa de classificação de parafusos



Suponhamos que o treinamento do perceptron já tenha sido feito e que os valores de pesos encontrados (exceto w_0, que foi prefixado em –1) correspondem aos descritos na Tabela 5.2.

Tabela 5.2 – Valores dos pesos após um processo de treinamento do exemplo de classificação dos parafusos

w_1	w_2	w_3
0,21	0,22	–1



Medeiros (2016, p. 134)

5.1.1 Exemplo de perceptron simples: classificação de parafusos

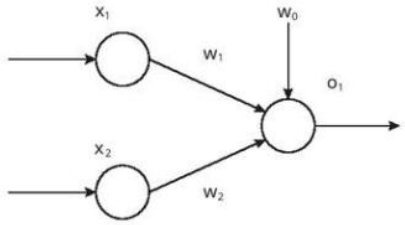
Assumamos que o valor da saída +1 seja atribuído à classe A, e o valor de saída –1, à classe B. Para o cálculo da variável d_1, utilizamos a fórmula genérica (1) vista anteriormente. Pela função de transferência, se o valor d_1 é positivo, a saída será +1; senão, será –1 (Tabela 5.3).

Tabela 5.3 – Classificação após o treinamento

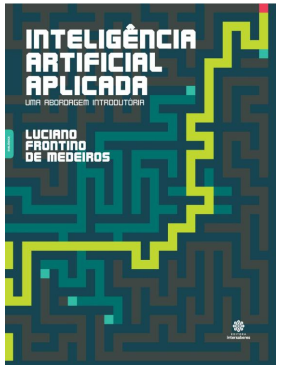
Parafuso	x_1	x_2	d_1	o_1	Classe
1	4 cm	3 mm	0,50	+1	A
2	1 cm	1 mm	–0,57	–1	B
3	2 cm	2 mm	–0,14	–1	B
4	2 cm	3 mm	0,08	+1	A
5	5 cm	3 mm	0,71	+1	A
6	3 cm	2 mm	0,07	+1	A

w_1	w_2	w_3
0,21	0,22	–1

Figura 5.4 – Topologia do perceptron para a tarefa de classificação de parafusos



Dessa forma, os valores a serem buscados para o aprendizado da rede devem ser semelhantes aos da Tabela 5.3. Depois, o processo de treinamento deverá se encarregar de definir esses valores, para que a rede classifique corretamente.



Medeiros (2016, p. 134)

Exercício. Cálculo do Perceptron.

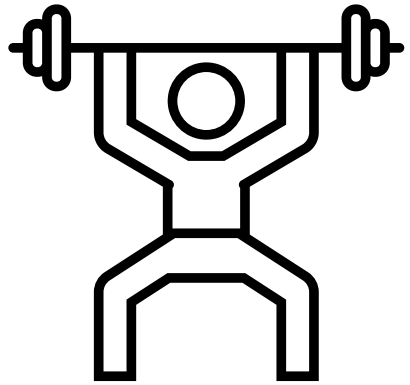
Suponhamos que, em média, os parafusos de comprimento próximo a 5 cm sejam os maiores (pertencentes à classe A) e aqueles que meçam em torno de 1 cm, os menores (pertencentes à classe B). Consideremos"

Para o cálculo da variável d_1, utilizamos a fórmula genérica (1) vista anteriormente. Pela função de transferência, se o valor d_1 é positivo, a saída será +1; senão, será -1

w_1	w_2	w_3
0,21	0,22	–1

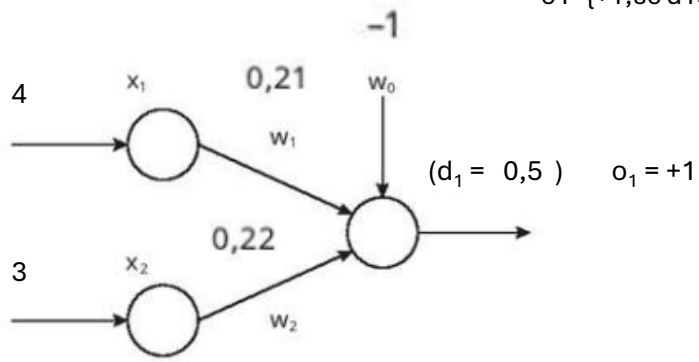
Tabela 5.3 – Classificação após o treinamento

Parafuso	x_1	x_2	d_1	o_1	Classe
1	4 cm	3 mm	0,50	+1	A
2	1 cm	1 mm			
3	2 cm	2 mm			
4	2 cm	3 mm			
5	5 cm	3 mm			
6	3 cm	2 mm			



$d1=(4\times0.21)+(3\times0.22)-1 = 0,5$

$o1=\{+1, \text{se } d1\geq 0 \quad -1, \text{se } d1<0$



Calcular a saída para todos os outros parafusos!

Suponhamos que, em média, os parafusos de comprimento próximo a 5 cm sejam os maiores (pertencentes à classe A) e aqueles que meçam em torno de 1 cm, os menores (pertencentes à classe B). Consideremos"

Para o cálculo da variável d_1, utilizamos a fórmula genérica (1) vista anteriormente. Pela função de transferência, se o valor d_1 é positivo, a saída será +1; senão, será -1

w_1	w_2	w_3
0,21	0,22	–1

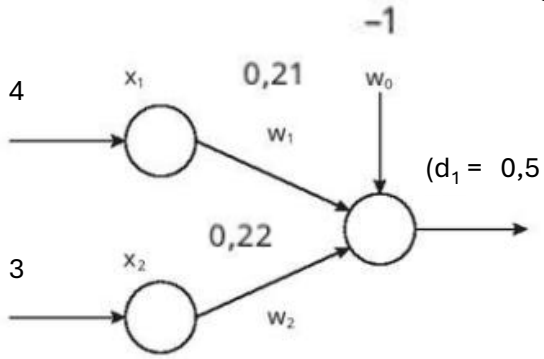
Tabela 5.3 – Classificação após o treinamento

Parafuso	x_1	x_2	d_1	o_1	Classe
1	4 cm	3 mm	0,50	+1	A
2	1 cm	1 mm	–0,57	–1	B
3	2 cm	2 mm	–0,14	–1	B
4	2 cm	3 mm	0,08	+1	A
5	5 cm	3 mm	0,71	+1	A
6	3 cm	2 mm	0,07	+1	A

Exercício. Cálculo do Perceptron.

$d1=(4\times0.21)+(3\times0.22)-1 = 0,5$

$o1=\{+1, \text{se } d1\geq0 \; -1, \text{se } d1<0$

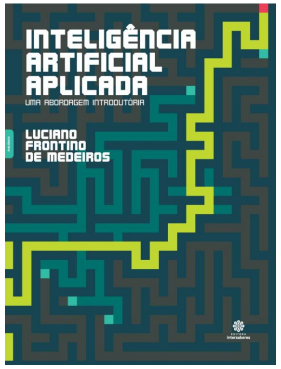


Calcular a saída para todos os outros parafusos!

5.1.2 Treinamento do perceptron

Um dos algoritmos mais estudados para o treinamento de um perceptron é o algoritmo do **mínimo quadrado médio** (em inglês, *Least-Mean-Square* – LMS), que foi criado por Widrow e Hoff (1960). Sua implementação não é complexa e permite alcançar bons resultados no treinamento de um perceptron (Haykin, 2001). O algoritmo LMS consiste em calcular o erro de classificação para, na sequência, utilizar uma fração desse erro para o ajuste dos pesos.

Costuma-se começar o processo de treinamento adotando valores aleatórios para os pesos. Ao proceder à primeira alimentação de valores à entrada, acontece o erro de classificação após as amostras terem sido apresentadas ao perceptron. Durante o treinamento e de forma gradativa, um pequeno percentual desse erro é propagado de volta ao perceptron. Esse percentual é denominado *taxa de aprendizagem*. No que se refere aos erros, examinaremos dois conceitos: o individual e o global.



Medeiros (2016, p. 134)

5.1.2 Treinamento do perceptron

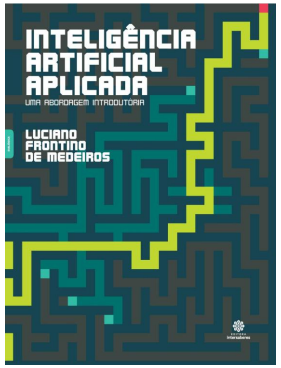
O **erro individual** refere-se a cada classificação de amostra e tem a seguinte fórmula:

$e_i = (o_i - f_i)$	2

Nessa fórmula, o erro *ei* corresponde à diferença entre a saída desejada *oi* e a saída calculada após a ativação *fi*.

O **erro global**, por sua vez, consiste em uma medida de desempenho global do perceptron, que pode ser calculada pela média quadrática dos erros individuais:

$E = \frac{1}{2}\sum_{i=1}^{n} e_i^2$	3



Medeiros (2016, p. 137)

5.1.2 Treinamento do perceptron

Quanto ao aprendizado, ele será gradativo e atualizará os pesos a cada execução da alimentação à frente.

Para calcular a parcela do erro que será realimentado no perceptron, utilizamos a fórmula de cálculo dos deltas[3], qual seja:

$$\Delta_j = \frac{1}{n}\sum_{i=1}^{n}\eta(o_i - f_i)x_j = \frac{1}{n}\sum_{i=1}^{n}\eta e_i x_j \qquad 4$$

Em que:

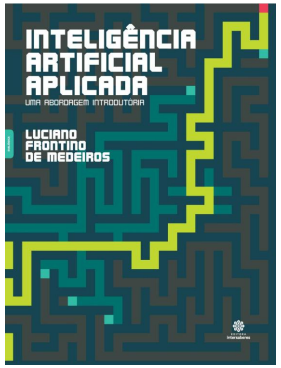
η = taxa de aprendizado;
o_i = saída desejada;
f_i = saída calculada;
e_i = erro ou diferença;
x_i = neurônio de entrada;
n = número de amostras.

Para calcular os deltas, multiplicamos o erro de cada amostra pela entrada referente ao delta. Logo após, extraímos a média dos valores calculados para os deltas. Em seguida, os pesos são atualizados com os valores dos deltas, da seguinte maneira:

$$w_j(n+1) = w_j(n) + \Delta_j(n) \qquad 5$$

Por essa característica, tal regra de aprendizagem costuma ser chamada de ***regra delta***. A seguir, demonstraremos como acontece o treinamento do perceptron e constataremos, em termos práticos, como ele aprende a classificar os parafusos.

3 Usamos uma variação do algoritmo LMS por meio do cálculo do valor médio dos erros, em vez de aplicar o erro instantâneo previsto para o algoritmo original.



Medeiros (2016, p. 137)

5.1.3 Demonstrando o treinamento

O treinamento do perceptron é ilustrado de forma didática nas Tabelas 5.4 a 5.8. O treinamento começa no passo 1, com os valores iniciais dos pesos w_1 e w_2 iguais a 1, e o peso *bias* w_0 fixado em –1.

Tabela 5.4 – Dados do treinamento do perceptron para classificação de parafusos

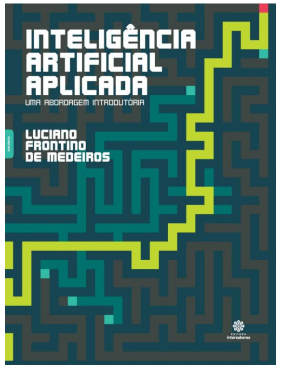
Taxa de aprendizado (η)	0,01
Passo	1

w_1	w_2	w_0
1	1	–1

Amostra	x_1 (cm)	x_2 (mm)	d_1	f_1	o_1	Atual	Alvo	e_1	Δ_1	Δ_2
1	4	3	6,00	1	1	A	A	–	–	–
2	1	1	1,00	1	–1	A	B	(2,00)	(0,02)	(0,02)
3	2	2	3,00	1	–1	A	B	(2,00)	(0,04)	(0,04)
4	2	3	4,00	1	1	A	A	–	–	–
5	5	3	7,00	1	1	A	A	–	–	–
6	3	2	4,00	1	–1	A	B	(2,00)	(0,06)	(0,04)
						Soma	∑e	(6,00)	(0,02)	(0,02)
						Soma Quad	$\sum e^2$	12,00		

Nota: Na parte superior, estão os valores da taxa de treinamento ou aprendizado (igual a 1%); os valores iniciais dos pesos (w_1 e w_2 iguais a 1,0 e w_0 igual a –1); e no quadro maior, as amostras com os valores das entradas de comprimento e largura dos parafusos produzem os resultados de acordo com as fórmulas 1 a 4.

4 Na Tabela 5.4, o número negativo aparece entre parênteses.



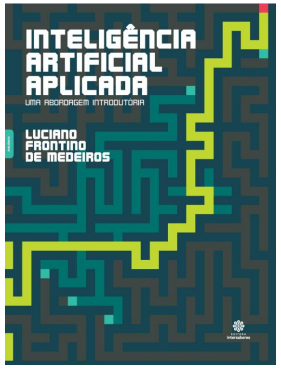
Medeiros (2016, p. 138)

5.1.3 Demonstrando o treinamento

Na Tabela 5.4, as amostras consideradas são colocadas em linhas. A coluna d_1 mostra o cálculo referente à fórmula 1. Por exemplo, na amostra 1 (comprimento de 4 cm e largura de 3 mm), temos $d_1 = x_1 \times w_1 + x_2 \times w_2 + w_0 = 4 \times 1 + 3 \times 1 - 1 = 6$. Na coluna f_1, representamos a função de ativação: caso d_1 seja maior que zero, ele assume o valor 1 (positivo); se for menor que zero, assume o valor –1. A coluna o_1 corresponde à classe desejada: caso a amostra faça parte da classe A, o valor de o_1 deve ser 1; se for da classe B, o valor é –1. As colunas intituladas *Atual* e *Alvo* explicitam as classes dos parafusos (atual para f_1 e alvo para o_1). A coluna e_1, por sua vez, refere-se ao erro de classificação. Seus dados são calculados mediante a fórmula 2. Por exemplo, a amostra 2 deveria ser da classe *B*, mas está sendo classificada pelo perceptron como classe *A*. Então, o erro e_1 é igual à diferença entre a saída desejada o_1 pela calculada f_1. Assim, $e_1 = o_1 - f_1 = -1 - 1 = -2$.[4]

4 Na Tabela 5.4, o número negativo aparece entre parênteses.

Na parte inferior da Tabela 5.4, verificamos o erro global, que é calculado por meio da fórmula 3. Cada erro na planilha é elevado ao quadrado, e todos são somados entre si. O resultado corresponde ao erro global *E*. O erro global *E* representa uma medida de desempenho do sistema. O fato de que o erro global *E* tende a zero indica que o perceptron está aprendendo.



Medeiros (2016, p. 138)

5.1.3 Demonstrando o treinamento

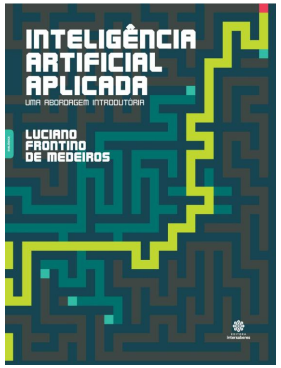
O valor do erro permite calcular os deltas parciais (Δ'_1 e Δ'_2). Para tal, usamos a fórmula 4. No caso da amostra 2, os valores dos deltas parciais serão: $\Delta'_1 = \eta e_1 x_1 = 0{,}01 \times -2{,}0 \times 1 = -0{,}02$; e $\Delta'_2 = \eta e_1 x_2 = 0{,}01 \times -2{,}0 \times 1 = -0{,}02$.

Note que nas amostras cuja classificação está correta não há erro, e os deltas são iguais a zero.

Na parte inferior da Tabela 5.1, representamos o valor correspondente à média de todos os deltas calculados, chegando a um valor de delta para cada entrada: $\Delta_1 = -0{,}02$ e $\Delta_2 = -0{,}02$.

Por fim, depois de termos calculado os deltas, aplicamos a fórmula 5 para atualizar os valores dos pesos. Dessa forma, o peso *w1*, que utilizaremos no passo 2, é reajustado conforme o valor de Δ_1: $w_1(2) = w_1(1) + \Delta_1 = 1 - 0{,}02 = 0{,}98$; e $w_2(2) = w_2(1) + \Delta_2 = 1 - 0{,}02 = 0{,}98$.

4 Na Tabela 5.4, o número negativo aparece entre parênteses.



Medeiros (2016, p. 138)

Exercício – Praticando o treinamento

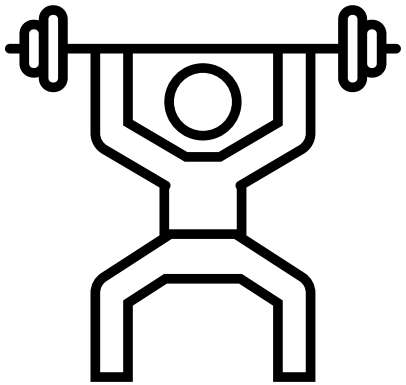


Tabela 5.4 – Dados do treinamento do perceptron para classificação de parafusos

Taxa de aprendizado (η)	0,01
Passo	1

w_1	w_2	w_0
1	1	–1

o_1 é a classe desejada.

4 Na Tabela 5.4, o número negativo aparece entre parênteses.

Amostra	x_1 (cm)	x_2 (mm)	d_1	f_1	o_1	Atual	Alvo	e_1	Δ_1	Δ_2
1	4	3								
2	1	1								
3	2	2								
4	2	3								
5	5	3								
6	3	2								
						Soma	$\sum e$			
						Soma Quad	$\sum e^2$			

Calcular ao final os novos pesos w1 e w2

Novo_w1 = w1 + Δ1

Novo_w2 = w2 + Δ2

Medeiros (2016, p. 138)

Resposta Exercício – Praticando o treinamento

Tabela 5.4 – Dados do treinamento do perceptron para classificação de parafusos

Taxa de aprendizado (η)	0,01
Passo	1

w_1	w_2	w_0
1	1	–1

o_1 é a classe desejada.

4 Na Tabela 5.4, o número negativo aparece entre parênteses.

Amostra	x_1 (cm)	x_2 (mm)	d_1	f_1	o_1	Atual	Alvo	e_1	Δ_1	Δ_2
1	4	3	6,00	1	1	A	A	–	–	–
2	1	1	1,00	1	–1	A	B	(2,00)	(0,02)	(0,02)
3	2	2	3,00	1	–1	A	B	(2,00)	(0,04)	(0,04)
4	2	3	4,00	1	1	A	A	–	–	–
5	5	3	7,00	1	1	A	A	–	–	–
6	3	2	4,00	1	–1	A	B	(2,00)	(0,06)	(0,04)
						Soma	$\sum e$	(6,00)	(0,02)	(0,02)
						Soma Quad	$\sum e^2$	12,00		

Calcular ao final os novos pesos w1 e w2

Novo_w1 = w1 + Δ1
Novo_w1 = 1 – 0,02 = 0,98

Novo_w2 = w2 + Δ2
Novo_w2 = 1 – 0,02 = 0,98

Medeiros (2016, p. 138)

5.1.3 Demonstrando o treinamento

A Tabela 5.5 mostra uma representação do passo 2, com os valores dos pesos atualizados.

Tabela 5.5 – Dados do treinamento do perceptron para classificação de parafusos no passo 2

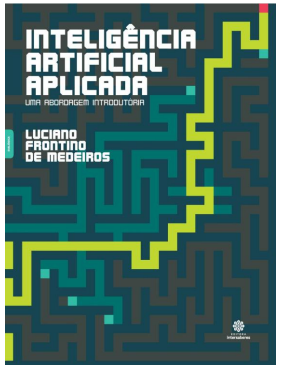
Passo	2

w_1	w_2	w_0
0,98	0,98	–1

o_1 é a classe desejada.

Amostra	x_1 (cm)	x_2 (mm)	d_1	f_1	o_1	Atual	Alvo	e_1	Δ_1	Δ_2
1	4	3	5,87	1	1	A	A	–	–	–
2	1	1	0,96	1	–1	A	B	(2,00)	(0,02)	(0,02)
3	2	2	2,93	1	–1	A	B	(2,00)	(0,04)	(0,04)
4	2	3	3,91	1	1	A	A	–	–	–
5	5	3	6,85	1	1	A	A	–	–	–
6	3	2	3,91	1	–1	A	B	(2,00)	(0,06)	(0,04)
						Soma	$\sum e$	(6,00)	(0,02)	(0,02)
						Soma Quad	$\sum e^2$	12,00		

Os valores atualizados dos pesos w_1 e w_2 têm impacto em cascata sobre os valores da Tabela 5.5. Observe que os valores de d_1 são diferentes daqueles do passo 1.



Medeiros (2016, p. 138)

Exercício – Praticando o treinamento

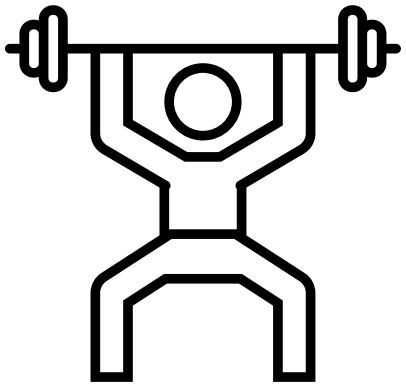


Tabela 5.5 – Dados do treinamento do perceptron para classificação de parafusos no passo 2

Passo	2

w_1	w_2	w_0
0,98	0,98	–1

o_1 é a classe desejada.

Amostra	x_1 (cm)	x_2 (mm)	d_1	f_1	o_1	Atual	Alvo	e_1	Δ_1	Δ_2
1	4	3						-	-	-
2	1	1								
3	2	2								
4	2	3						-	-	-
5	5	3						-	-	-
6	3	2								
						Soma	$\sum e$			
						Soma Quad	$\sum e^2$			

Calcular ao final os novos pesos w1 e w2

Novo_w1 = w1 + Δ1

Novo_w2 = w2 + Δ2

Exercício – Praticando o treinamento

Tabela 5.5 – Dados do treinamento do perceptron para classificação de parafusos no passo 2

Passo	2

w_1	w_2	w_0
0,98	0,98	–1

o_1 é a classe desejada.

Amostra	x_1 (cm)	x_2 (mm)	d_1	f_1	o_1	Atual	Alvo	e_1	Δ_1	Δ_2
1	4	3	5,87	1	1	A	A	–	–	–
2	1	1	0,96	1	–1	A	B	(2,00)	(0,02)	(0,02)
3	2	2	2,93	1	–1	A	B	(2,00)	(0,04)	(0,04)
4	2	3	3,91	1	1	A	A	–	–	–
5	5	3	6,85	1	1	A	A	–	–	–
6	3	2	3,91	1	–1	A	B	(2,00)	(0,06)	(0,04)
						Soma	$\sum e$	(6,00)	(0,02)	(0,02)
						Soma Quad	$\sum e^2$	12,00		

Calcular ao final os novos pesos w1 e w2

Novo_w1 = w1 + Δ1
Novo_w1 = 0,98 – 0,02 = 0,96

Novo_w2 = w2 + Δ2
Novo_w2 = 0,98 – 0,02 = 0,96

5.1.3 Demonstrando o treinamento

Quando o treinamento chega no passo 34 (Tabela 5.6), verificamos que os pesos foram reduzidos para $w_1 = 0{,}36$ e $w_2 = 0{,}47$. Nesse passo, a amostra 2 é classificada corretamente na classe *B*. No entanto, ainda podemos observar que as amostras 3 e 6 estão classificadas de maneira incorreta.

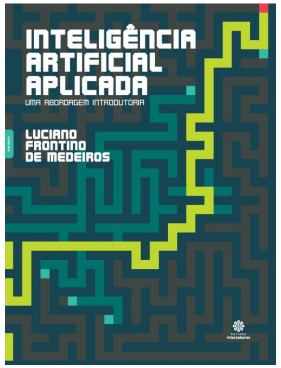
Tabela 5.6 – Dados do treinamento do perceptron para classificação de parafusos no passo 34

Passo	34

w_1	w_2	w_0
0,36	0,47	–1

Amostra	x_1 (cm)	x_2 (mm)	d_1	f_1	o_1	Atual	Alvo	e_1	Δ_1	Δ_2
1	4	3	1,83	1	1	A	A	–	–	–
2	1	1	(0,18)	–1	–1	B	B	–	–	–
3	2	2	0,65	1	–1	A	B	(2,00)	(0,04)	(0,04)
4	2	3	1,11	1	1	A	A	–	–	–
5	5	3	2,18	1	1	A	A	–	–	–
6	3	2	1,00	1	–1	A	B	(2,00)	(0,06)	(0,04)
						Soma	$\sum e$	(4,00)	(0,02)	(0,01)
						Soma Quad	$\sum e_2$	8,00		

Observe que a amostra 2 foi classificada corretamente na classe *B*. O erro global *E* diminuiu de 12,0 para 8,0.



Medeiros (2016, p. 138)

5.1.3 Demonstrando o treinamento

Quando o treinamento chega ao passo 48 (Tabela 5.7), com os pesos w_1 = 0,14 e w_2 = 0,30, a amostra 6 ainda é classificada de modo incorreto. O erro global é E = 4.0, demonstrando assim que, ao longo de todo o treinamento, o perceptron foi capaz de aprender gradativamente a classificar de maneira correta. Quando se chega ao passo (Tabela 5.8), o valor de d_1 da amostra 6 torna-se negativo, com pesos w_1 = 0,13 e w_2 = 0,29. Assim, o perceptron passa a classificá-la corretamente na classe B. Como não há mais valores de erro, o erro global *E* é 0 (zero), e podemos afirmar que o perceptron classifica o conjunto de parafusos nas classes corretas. Na teoria de redes neurais, esse processo está relacionado ao chamado ***teorema de convergência do perceptron***.

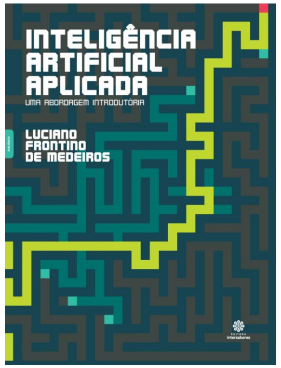
Tabela 5.7 – Dados do treinamento do perceptron para classificação de parafusos no passo 48

Passo	48

w_1	w_2	w_0
0,14	0,30	–1

Amostra	x_1 (cm)	x_2 (mm)	d_1	f_1	o_1	Atual	Alvo	e_1	Δ_1	Δ_2
1	4	3	0,47	1	1	A	A	–	–	–
2	1	1	(0,56)	–1	–1	B	B	–	–	–
3	2	2	(0,11)	–1	–1	B	B	–	–	–
4	2	3	0,19	1	1	A	A	–	–	–
5	5	3	0,62	1	1	A	A	–	–	–
6	3	2	0,03	1	–1	A	B	(2,00)	(0,06)	(0,04)
						Soma	$\sum e$	(2,00)	(0,01)	(0,01)
						Soma Quad	$\sum e^2$	4,00		

Nessa altura do treinamento, a amostra 6 ainda é classificada de maneira incorreta na classe *A*. O erro global *E*, porém, diminuiu para 4.0.



Medeiros (2016, p. 138)

5.1.3 Demonstrando o treinamento

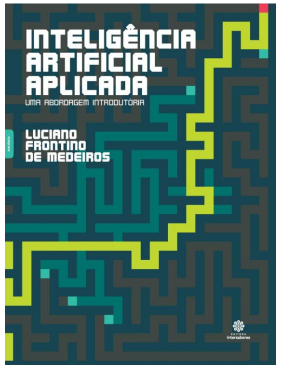
Tabela 5.8 – Dados do treinamento do perceptron para classificação de parafusos no passo 49

Passo	49

w_1	w_2	w_0
0,13	0,29	–1

Amostra	x_1 (cm)	x_2 (mm)	d_1	f_1	o_1	Atual	Alvo	e_1	Δ_1	Δ_2
1	4	3	0,41	1	1	A	A	–	–	–
2	1	1	(0,57)	–1	–1	B	B	–	–	–
3	2	2	(0,15)	–1	–1	B	B	–	–	–
4	2	3	0,15	1	1	A	A	–	–	–
5	5	3	0,55	1	1	A	A	–	–	–
6	3	2	(0,01)	–1	–1	B	B	–	–	–
						Soma	$\sum e$	–	–	–
						Soma Quad	$\sum e^2$	–		

Com os valores dos pesos indicados, o perceptron classifica corretamente todas as amostras. O erro global é igual a zero.

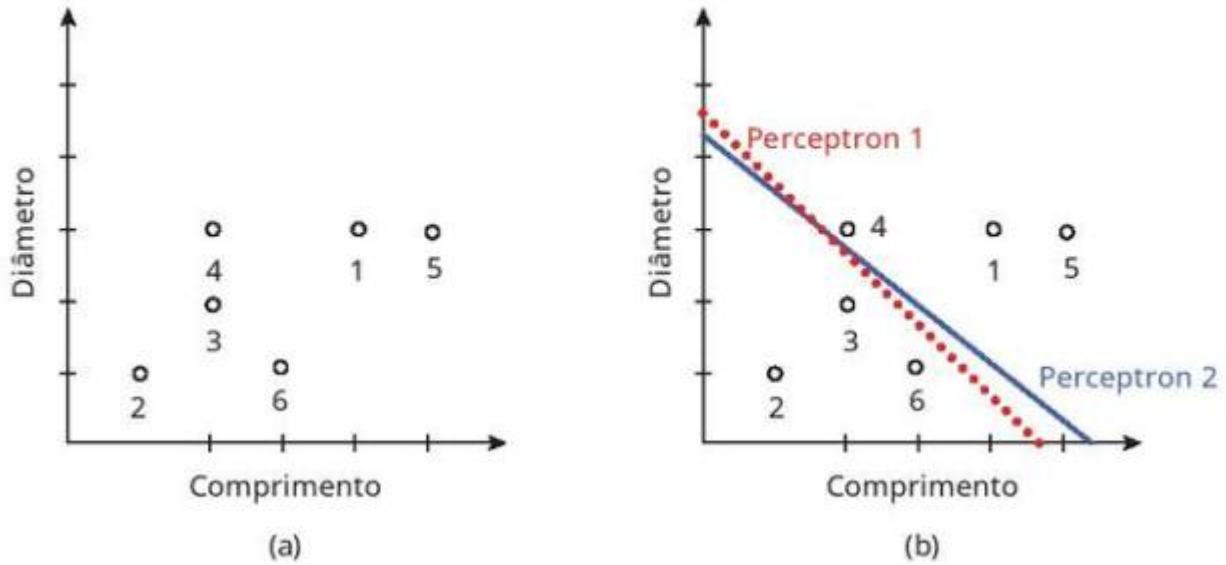


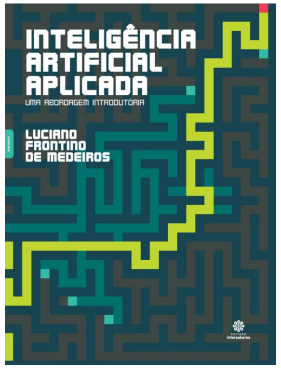
Medeiros (2016, p. 138)

5.1.3 Demonstrando o treinamento

É possível treinar um perceptron para classificar de maneira correta qualquer conjunto de amostras? Nem sempre. Para funcionar corretamente, o perceptron deve trabalhar com classes que sejam **separáveis linearmente**. Isso significa que os padrões ou as amostras pertencentes a classes distintas devem estar suficientemente separados para que uma correta classificação seja possível. Essa característica fica evidente quando plotamos as amostras como pontos em um gráfico de eixos cartesianos. No Gráfico 5.1, em (a), temos a representação das amostras como pontos sobre os eixos que representam o comprimento e o diâmetro de cada parafuso. Os parafusos 2, 3 e 6, da classe *B*, estão situados em posição inferior em relação à dos parafusos 1, 4 e 5, da classe *A*. Em (b), vemos a representação da separabilidade linear.

Gráfico 5.1 – Distribuição das amostras de acordo com as entradas do perceptron





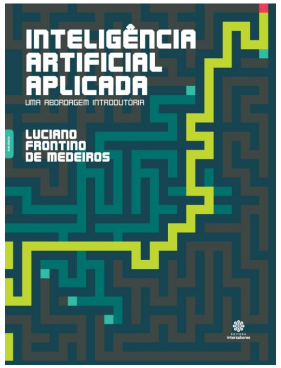
Medeiros (2016, p. 138)

5.1.3 Demonstrando o treinamento

Durante o treinamento, o perceptron "procura" uma forma de separar as duas classes com uma reta. Observe que no Gráfico 5.1, em (b), dois perceptrons podem separar as duas classes. Esse exemplo permite afirmar que é possível haver infinitas retas separando as duas classes. Assim, esse problema de classificação de parafusos pode ser considerado como **separável linearmente**. Caso a amostra 2, por exemplo, pertencesse à classe *B*, o perceptron teria dificuldades em fazer a classificação, pois não seria possível separar as duas classes por meio de uma reta. Em situações como essa, o perceptron consegue convergir para um erro global mínimo, porém este não pode ser reduzido a zero.

A classificação dos parafusos em duas entradas permitiu uma verificação direta da **separabilidade linear**. Entretanto, em problemas com mais de três entradas, torna-se difícil tal visualização. Já em problemas com muitas entradas, o erro global passa a ser a medida de desempenho empregada para identificar o estado de treinamento de um perceptron.

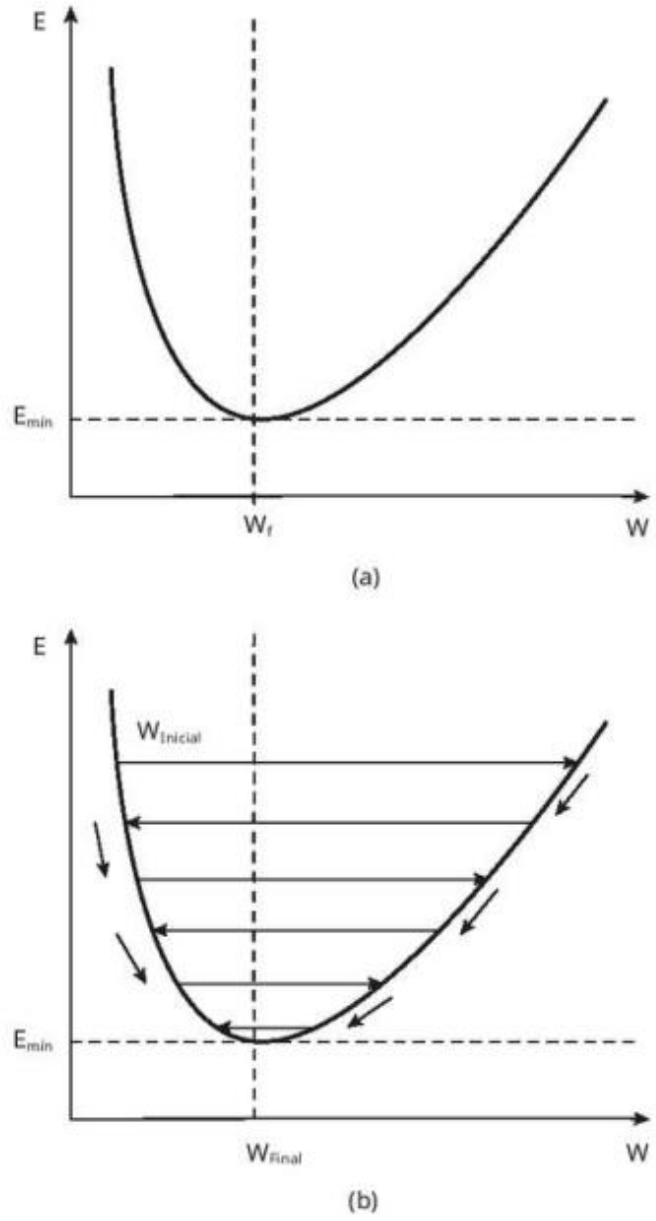
A maneira como o conjunto de treinamento é tratada no algoritmo de aprendizagem também pode variar. A atualização dos pesos pode acontecer por **amostra**, quando, a cada amostra calculada, os pesos são atualizados, ou por **lote**, quando se calcula a variação média de todo o conjunto e depois se aplica a atualização dos neurônios (como no caso do perceptron apresentado anteriormente).



Medeiros (2016, p. 138)

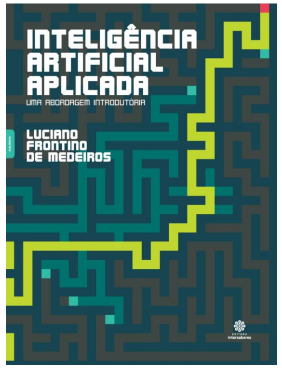
5.1.3 Demonstrando o treinamento

Gráfico 5.2 – Relação dos pesos com o erro global



Nota: (a) relação do peso com o erro global; (b) processo de atualização dos pesos como convergência de seus valores a um valor final ótimo.

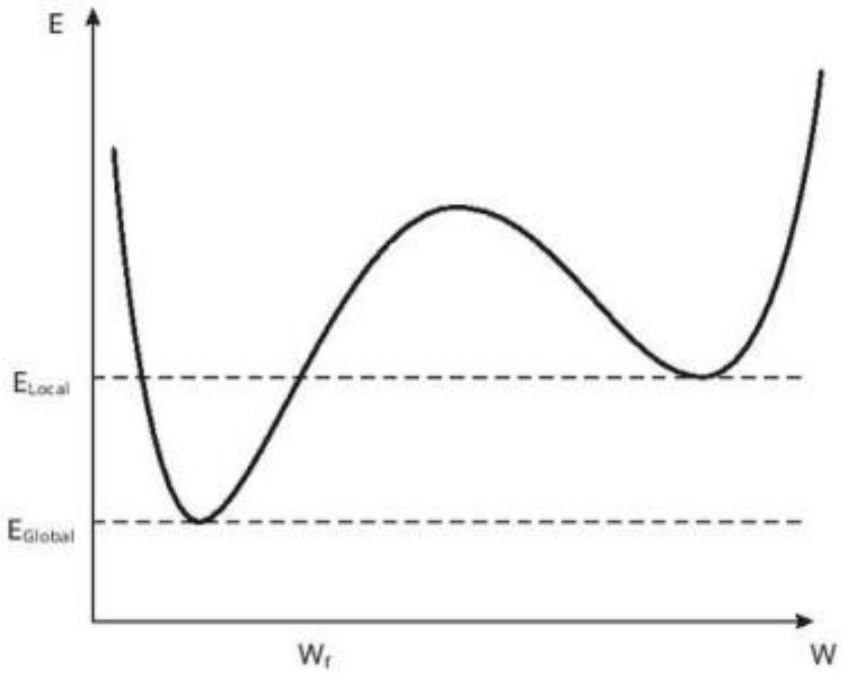
No Gráfico 5.2, a seguir, vemos a relação dos pesos com o erro global, no que se denomina ***espaço de busca de pesos***. Ao longo do treinamento, os pesos são atualizados gradativamente até se obter um valor ótimo. O algoritmo de treinamento empregado nesse caso é designado também como um algoritmo de **descida de gradiente**. O valor ótimo obtido é aquele em que a rede apresentará o menor erro global. Diz-se também que o aprendizado se refere a um processo em que os valores dos pesos convergem para um valor ótimo. O valor ótimo dos pesos é aquele que minimiza o erro global.



Medeiros (2016, p. 138)

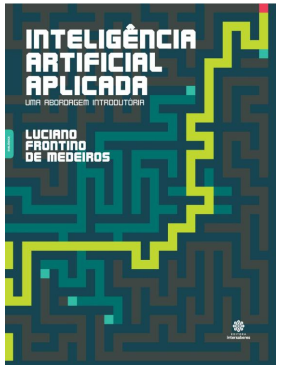
5.1.3 Demonstrando o treinamento

Gráfico 5.3 – Mínimo local e mínimo global de um espaço de busca



Entretanto, nem todos os espaços de busca apresentam as características presentes no Gráfico 5.2. Certos problemas podem apresentar espaços de busca que contenham mais de um mínimo. Assim, quando um valor de erro global converge para um valor mínimo que não é o mais baixo do espaço de busca, trata-se de um **mínimo local**. O valor mínimo mais baixo de um espaço de busca, por sua vez, é considerado um **mínimo global** (Gráfico 5.3).

Eventualmente, uma RNA pode ficar restrita a um mínimo local. No espaço de busca representado no Gráfico 5.3, isso é evidenciado pela descida gradativa da curva do algoritmo de treinamento. No entanto, é possível fazer modificações nos algoritmos de modo a reduzir a possibilidade de o treinamento restringir-se ao mínimo local.



Medeiros (2016, p. 138)