UNIDADE 4 - Algoritmos de clusterização

Marta Noronha

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
CURSO: CIÊNCIA DE DADOS
APRENDIZADO DE MÁQUINA II

SUMÁRIO

- 4.7 Conceitos de métodos baseados em Modelos
- 4.8 Algoritmos Expectation-Maximization e SOM
- 4.9 Noções básicas sobre biclusterização e triclusterização

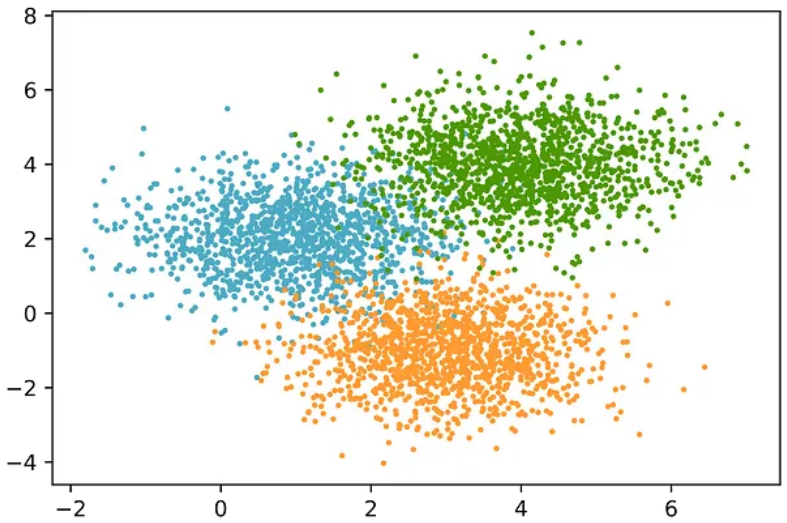
Métodos baseados em Modelos

- Abordagem estatística para agrupamento de dados.
- Permite estimar parâmetros em modelos probabilísticos com dados incompletos (valores ausentes ou corrompidos).
- Dados multivariados observados são considerados criados a partir de uma combinação finita de modelos de componentes.
- Cada componente do modelo é uma distribuição de probabilidade, geralmente uma distribuição paramétrica multivariada.
- O componente responsável por gerar uma determinada observação determina o cluster ao qual a observação pertence.

Fonte: What is model-based clustering?

Métodos baseados em Modelos

- Tentativa de melhorar o ajuste entre os dados fornecidos e algum modelo matemático.
- Baseado na suposição de que os dados são criados por uma combinação de uma distribuição de probabilidade básica.

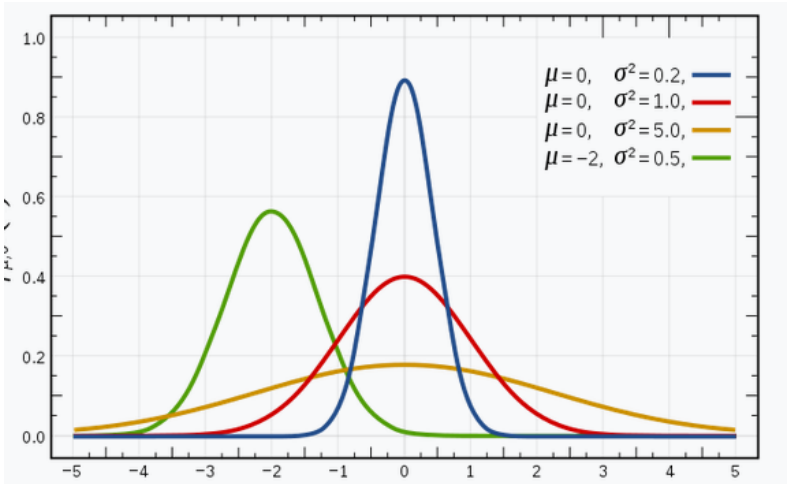


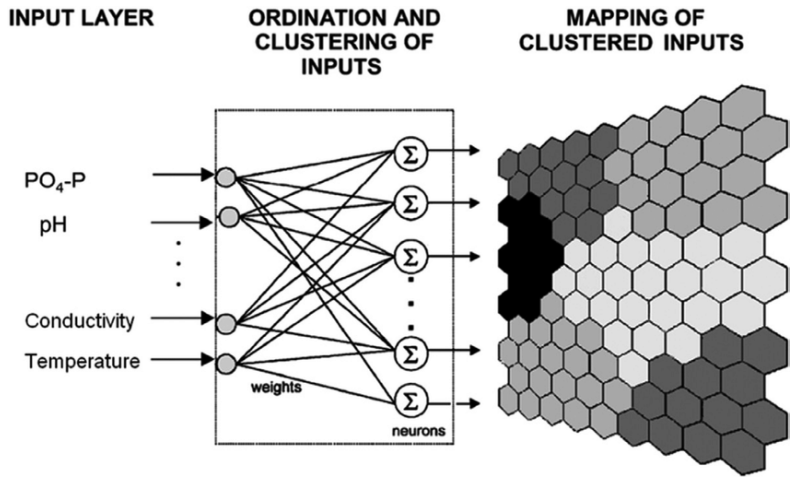
Fonte: What is model-based clustering?
Fonte da imagem: O raciocínio por trás do Algoritmo Expectation-Maximization

Métodos baseados em Modelos

Tipos de clustering baseado em modelo

- Abordagem estatística.
- Redes neurais.





Fontes: Complete Guide to Expectation-Maximization Algorithm (esquerda) e Rosli e Yahya 2017 (direita)

EM - **E**xpectation-**M**aximization

- Proposto em 1977, demonstrando utilidade em modelagem estatística e estimativa.
- Empregado em problemas de estimativa de densidade para representar os dados em uma forma compacta utilizando uma distribuição estatística.
- EM é um modelo generativo probabilístico, onde a atribuição de um cluster a um ponto é dado por probabilidade ("*soft clustering*").
- Não existe um única distribuição que, sozinha, mapeie milhares de instâncias em k clusters.

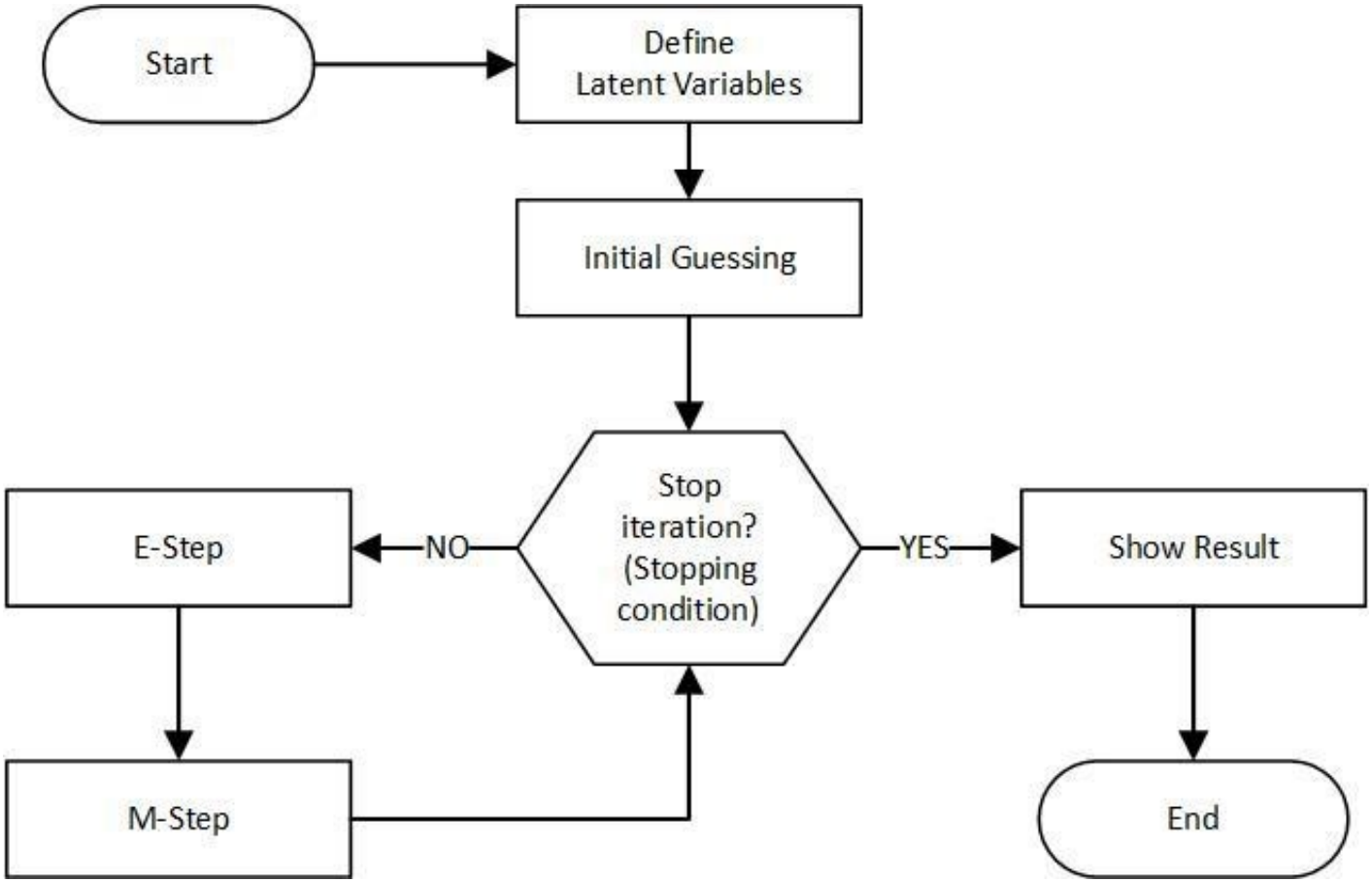
EM - **E**xpectation-**M**aximization

- Usado quando existem variáveis no conjunto de dados que foram parcialmente observáveis.
- Usando instâncias com variáveis observáveis, podemos usá-las para aprender e estimar seus valores, prevendo-se seus valores quando estas não são observáveis.
- EM é aplicável a variáveis latentes, ou seja, variáveis que não são diretamente observáveis, mas têm o seu valor inferido a partir de valores de outras variáveis observadas.
- EM encontra os parâmetros locais de máxima verossimilhança de um modelo estatístico e infere variáveis latentes nos casos em que os dados estão ausentes ou incompletos.
- Garantia de convergência dado que o valor da probabilidade aumenta após cada iteração.

EM - **E**xpectation-**M**aximization

Expectation: Atualiza as variáveis.

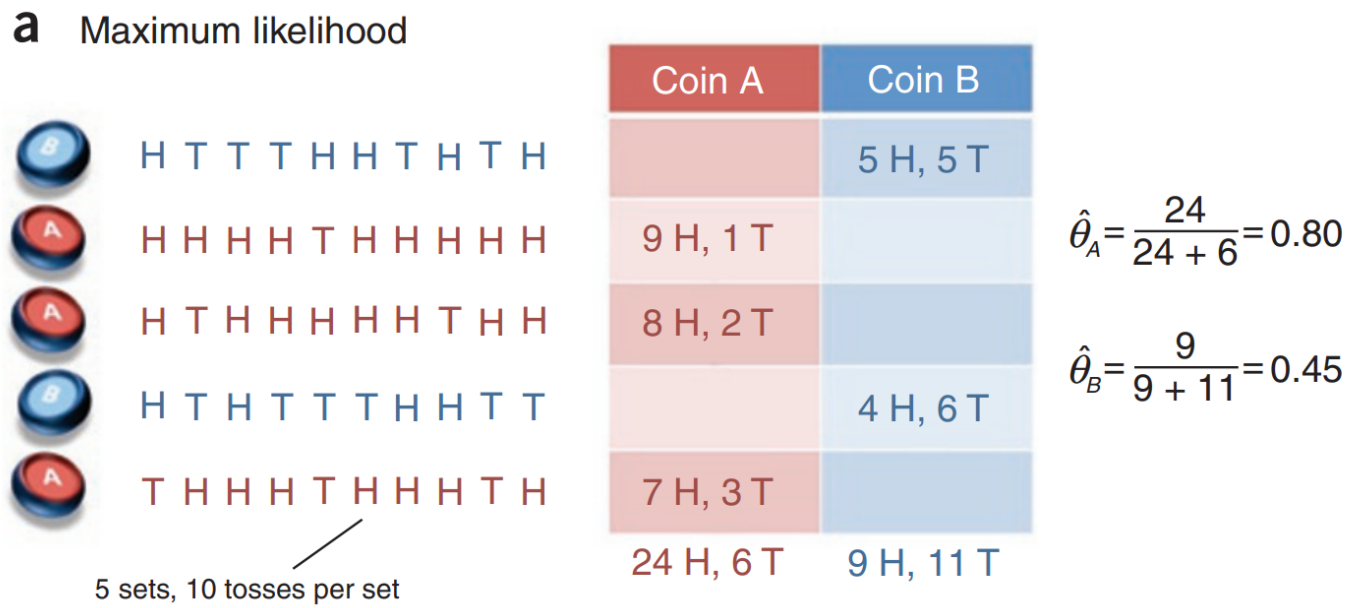
Maximization: Atualiza as hipóteses.



Fonte: The Simple Concept of Expectation – maximization (EM) Algorithm

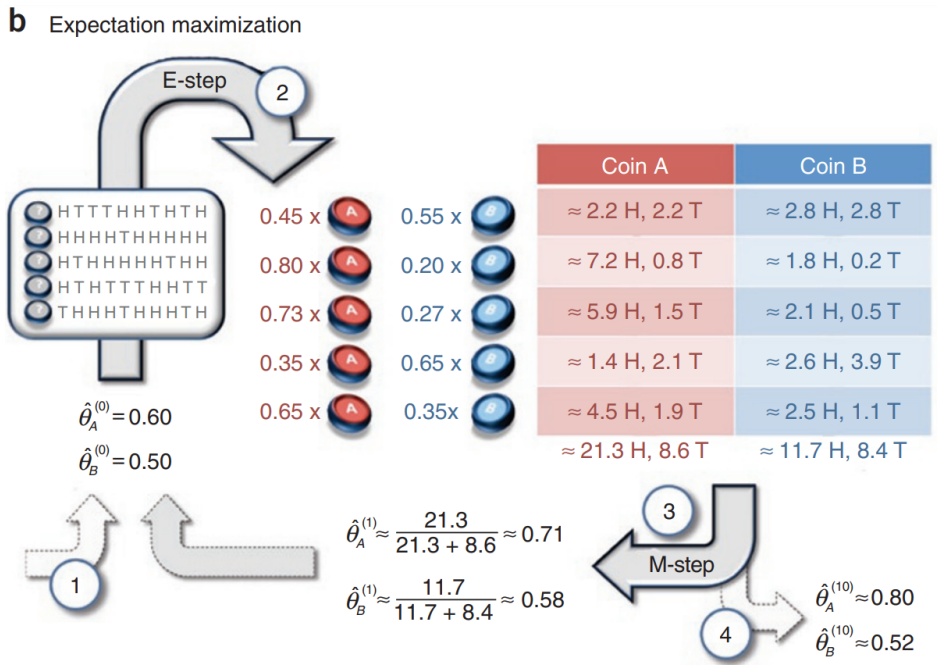
Exemplo do lançamento de moeda

Estimativa de máxima verossimilhança: Para cada conjunto de dez lançamentos, o procedimento de máxima verossimilhança acumula as contagens de caras e coroas para as moedas A e B separadamente. Essas contagens são usadas para estimar os vieses da moeda. (Traduzido de Do e Batzoglou 2008)



Exemplo do lançamento de moeda

Maximização da expectativa: 1. EM começa com uma estimativa inicial dos parâmetros. **2.** No passo E, uma distribuição de probabilidade sobre possíveis conclusões é calculada usando os parâmetros atuais. As contagens mostradas na tabela são os números esperados de caras e coroas de acordo com esta distribuição. **3.** Na etapa M, novos parâmetros são determinados usando as conclusões atuais. **4.** Após várias repetições do passo E e do passo M, o algoritmo converge.(Traduzido de Do e Batzoglou 2008)



Exemplo do lançamento de moeda

Dado um experimento com lançamento de duas moedas A e B. a) Escolhe-se uma moeda e a lança por dez vezes. b) Anota-se a quantidade de caras obtidos. Repete a partir do passo a) por 5 vezes. Têm-se a tabela:

Coin	Flips	coin A heads	coin B heads
B	HTTTHHTHTH	0	5
A	HHHHTHHHHH	9	0
A	HTHHHHHTHH	8	0
B	HTHTTTHHTT	0	4
A	THHHTHHHTH	7	0

$$\boldsymbol{\theta_A} = \frac{\#\text{ of heads using coin A}}{\text{total }\#\text{ of flips using coin A}} = \frac{24}{30} = 0.8$$

$$\boldsymbol{\theta_B} = \frac{\#\text{ of heads using coin B}}{\text{total }\#\text{ of flips using coin B}} = \frac{9}{20} = 0.45$$

Fonte: Intuitive Explanation of the Expectation-Maximization (EM) Technique

Exemplo do lançamento de moeda

Na Figura, n é o número de lançamentos em uma série e x é o número de caras. a) Escolher um valor inicial aleatório de probabilidade para θ_A e θ_B. b) Supor uma saída aleatória para este valor e, com a saída, determinar o valor inicial de caras.

$\boldsymbol{\theta_A} = 0.6$ and $\boldsymbol{\theta_B} = 0.5$

Suponha que obtivemos a saída HHHHHTTTTT.

$$P(E \mid Z_A) = P(HHHHHHHHHT \mid \text{A chosen}) = \binom{n}{x} \theta_A^x (1-\theta_A)^{n-x}$$

$$P(E \mid Z_B) = P(HHHHHHHHHT \mid \text{B chosen}) = \binom{n}{x} \theta_B^x (1-\theta_B)^{n-x}$$

Fonte: Intuitive Explanation of the Expectation-Maximization (EM) Technique

Exemplo do lançamento de moeda

Com o Teorema de Bayes e da lei da probabilidade total, pode-se estimar a probabilidade de uma determinada moeda ter sido usada para gerar os lançamentos:

$$P(Z_A|E) = \frac{P(E|Z_A)P(Z_A)}{P(E|Z_A)P(Z_A) + P(E|Z_B)P(Z_B)}$$

$$P(Z_B|E) = \frac{P(E|Z_B)P(Z_B)}{P(E|Z_A)P(Z_A) + P(E|Z_B)P(Z_B)}$$

Fonte: Intuitive Explanation of the Expectation-Maximization (EM) Technique

Exemplo do lançamento de moeda

$$\boldsymbol{\theta_A} = 0.6 \qquad \boldsymbol{\theta_B} = 0.5$$

N heads	$P(Z_A\|E)$	$P(Z_B\|E)$	$Heads_A$	$Tails_A$	$Heads_B$	$Tails_B$
5	0.45	0.55	2.2	2.2	2.8	2.8
9	0.8	0.2	7.2	0.8	1.8	0.2
8	0.73	0.27	5.9	1.5	2.1	0.5
4	0.35	0.65	1.4	2.1	2.6	3.9
7	0.65	0.35	4.5	1.9	2.5	1.1
		$Total$	21.3	8.6	11.7	8.4

$$\theta_A^1 = \frac{21.3}{21.3 + 8.6} = 0.71$$

$$\theta_B^1 = \frac{11.7}{11.7 + 8.4} = 0.58$$

Fonte: Intuitive Explanation of the Expectation-Maximization (EM) Technique

Exemplo do lançamento de moeda

Critérios de parada:

- Iteração máxima: o algoritmo EM irá parar se um certo número de iterações for atingido.
- Limiar de convergência: significa que o passo M não oferece nenhuma melhoria significativa nos parâmetros; em comparação com a melhoria na iteração anterior. As mudanças são muito pequenas abaixo do nosso limite.

Fonte: The Simple Concept of Expectation – maximization (EM) Algorithm

Exemplo do lançamento de moeda

Iteration	θ_A	θ_B	Differences
0	0.6	0.5	0.7810
1	0.713	0.581	0.1390
2	0.745	0.569	0.0342
3	0.768	0.550	0.0298
4	0.783	0.535	0.0212
5	0.791	0.526	0.0120
6	0.795	0.522	0.0057
7	0.796	0.521	0.0014
8	0.796	0.520	0.0010
9	0.796	0.520	0.0000

Fonte: The Simple Concept of Expectation – maximization (EM) Algorithm

Exemplo do lançamento de moeda

Distância euclidiana para calcular as diferenças entre cada iteração:

$$\mathrm{d(q,p)} = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \ + (q_n - p_n)^2}$$

$$d(\theta, \theta') = \sqrt{(\theta_A - \theta'_A)^2 + (\theta_B - \theta'_B)^2}$$

$$d(iter1, iter2) = \sqrt{(0.713 - 0.745)^2 + (0.581 - 0.569}$$

$$= 0.0342$$

Fonte: The Simple Concept of Expectation – maximization (EM) Algorithm

EM - **E**xpectation-**M**aximization

Fraquezas

- Cada iteração sempre melhora o parâmetro mais próximo da máxima verossimilhança local. Portanto EM garante a convergência, mas não garante uma probabilidade máxima global.
- Não há garantia de que obterá a estimativa de máxima verossimilhança (MLE).
- Escolha do valor estimado inicial pode afetar o resultado e o algoritmo pode convergir para um resultado inesperado.
- Convergência lenta.

Fonte: The Simple Concept of Expectation – maximization (EM) Algorithm

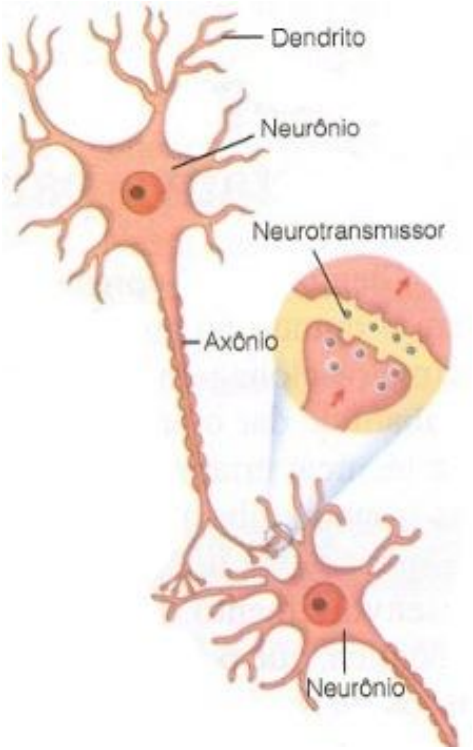
Aplicações para o EM - **E**xpectation-**M**aximization

- Cálculo estimado para encontrar a densidade gaussiana de uma função.
- Preenchimento de dados ausentes em uma amostra.
- Usado em diferentes domínios, como Processamento de Linguagem Natural (PNL), Visão Computacional, etc.
- Reconstrução de imagens na área de Medicina e Engenharia Estrutural.
- Estimação de parâmetros do Modelo Oculto de Markov (HMM) e também para alguns outros modelos mistos, como Modelos de Mistura Gaussiana, etc.
- Descoberta de valores de variáveis latentes.

Fonte: Complete Guide to Expectation-Maximization Algorithm

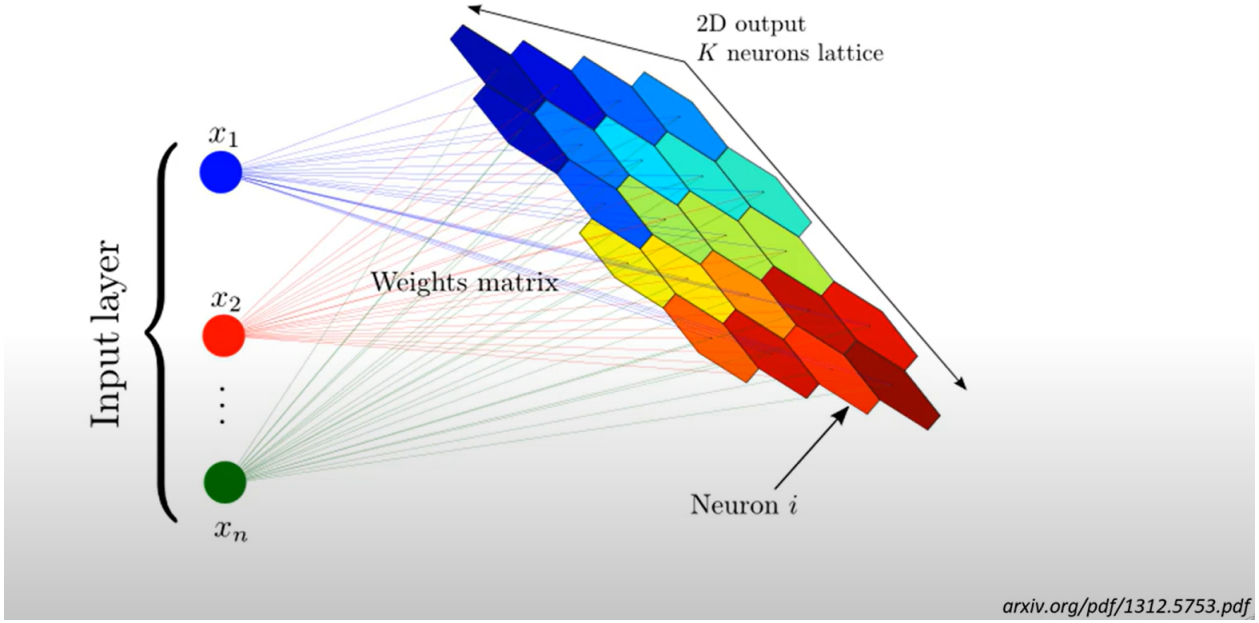
SOM - **S**elf **O**rganizing **M**aps

- Tipo de rede neural artificial (RNA) e não supervisionada.
- Uma RNA é inspirado na forma em que o cérebro adquire e armazena o conhecimento.



SOM - **S**elf **O**rganizing **M**aps

- Uso de um algoritmo de aprendizagem competitivo para seu treinamento.
- SOM mapeia dados de dimensões superiores em dados de dimensões inferiores
- O modelo consiste da camada de entrada e da camada de saída (Kohonen), sem usar Bias para ajuste de resposta.



SOM - **S**elf **O**rganizing **M**aps

Auto-organização

- Auto-amplificação de pesos sinápticos.
- Competição entre as sinapses (pesos) de um único neurônio ou um conjunto de neurônios, sendo que o que responde de forma mais forte é considerado o vencedor.
- O neurônio com maior similaridade, ou seja, menor distância euclidiana, vence.
- Pode ser implementado tendo conexões com inibições laterais, onde o neurônio vencedor pode reduzir a atividade dos seus vizinhos dando um *feedback* negativo a eles.
- Cooperação entre as sinapses, no nível dos neurônios, e dos neurônios, no nível da rede, que se modificam.
- Utiliza conceito de topologia, onde se um neurônio ficar excitado, os seus vizinhos ficarão mais excitados do que os vizinhos distantes.
- A informação estrutural, ordenamento e estrutura dos dados de entrada, é adquirida por um sistema auto-organizável.

SOM - **S**elf **O**rganizing **M**aps

Ajuste dos pesos dos neurônios

$$w_j(n+1) = w_j(n) + \alpha h_{ij}\Big(d\left(x, w_j(n)\right)\Big)$$

- x é o vetor de entrada
- $w_j(n)$ é a representação vetorial do neurônio j na iteração n (w_i(n) é o neurônio vencedor)
- α é a taxa de aprendizagem
- d é uma função de distância

Pode-se usar a função Gaussiana para verificar a influência do vencedor sobre os neurônios próximos à ele.

$$h(d(i,j)) = e^{-\frac{d^2(i,j)}{2\sigma^2}}$$

Fonte: How Do Self-Organizing Maps Work?

SOM - **S**elf **O**rganizing **M**aps

Algorithm 1: Self-Organizing Maps

Data: training data X; lattice dimensions k and l; the maximal number of iterations M; the neighborhood function h

Result: SOM vectors: V

Function SOM(X, k, l, h, M)

```
Randomly initialize SOM vectors V
for n ← 1 to M do
    for sample in X do
        // competition
        Find the neuron closest to the sample
        // cooperation and adaptation
        Update V with respect to the winning neuron
    end
end
Return V
```

end

SOM - **S**elf **O**rganizing **M**aps

```python
import numpy as np

class SOM:
    def winner(self, weights, sample): #find winning vector
        P0,P1,i = 0,0,0 # i is iterator counter

        while i < (len(sample)):
            P0 +=  (sample[i] - weights[0][i])**2
            P1 +=  (sample[i] - weights[1][i])**2
            print(np.around(P0,decimals=2),np.around(P1,decimals=2))

            if P1 < P0: return 0
            elif P1 > P0: return 1
            i += 1

    def update(self, weights, sample, K, alpha): # update winning vector
        i = 0
        while i < (len(weights[K])):
            weights[K][i] += (sample[i] - weights[K][i]) * alpha
            i += 1
        return weights
```

Fonte: Modificado de What are self-organizing maps (SOMs)?

SOM - **S**elf **O**rganizing **M**aps

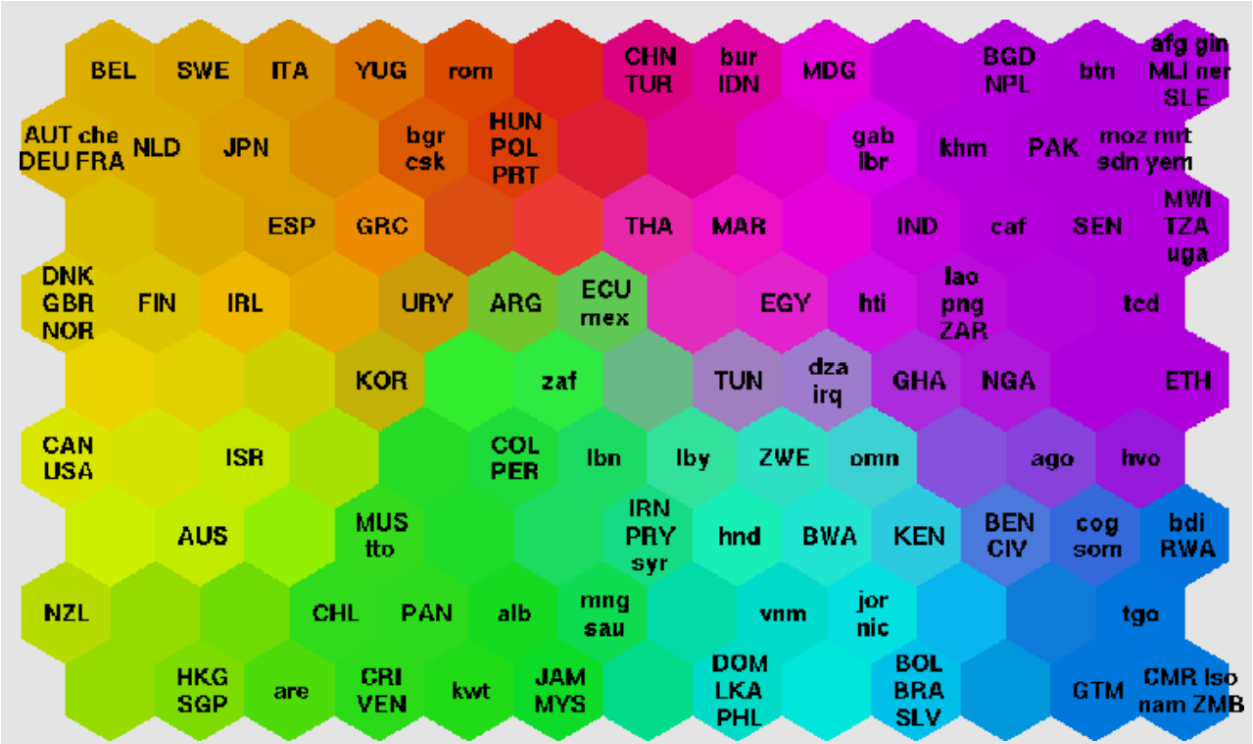
```
def main(): # main function
    # Initialization of weights
    weights = [[0.2, 0.6, 0.3, 0.9], [0.5, 0.2, 0.7, 0.3]]

    # Training Examples
    T = [[1, 1, 1, 0], [0, 1, 0, 1], [0, 0, 0, 1], [0, 1, 0, 1]]

    ob = SOM() # training
    epochs, alpha = 3, 0.4 # epochs and alpha values initialised
    i = 0

    while i < epochs:
        i += 1 # increment counter
        j = 0 # initialise e j as 0
        while j < len(T):
            sample = T[j] # training sample
            print("sample: ", sample, "\n")
            K = ob.winner(weights, sample) # Computing the winner vector
            print("winner: weight ", K, "\n")
            weights = ob.update(weights, sample, K, alpha) # Updating winning vector
            print(np.around(weights,decimals=2))

            j += 1 # increment counter
            print("--------------------------\n")

    # classifying the test sample
    K = ob.winner(weights, [1, 0, 1, 0])

    print(f"Test samples belongs to cluster, {K} ","\n")
    print("The trained weights are  ", *(np.around(weights,decimals=2)), sep = "\n")

main()
```

Fonte: Modificado de What are self-organizing maps (SOMs)?

Aplicação do algoritmo SOM

World Poverty Map usando Self-Organizing Maps

- Estatísticas do Banco Mundial relativas aos países em 1992.
- 39 indicadores descrevem vários fatores de qualidade de vida, tais como estado de saúde, nutrição, serviços educativos, etc.

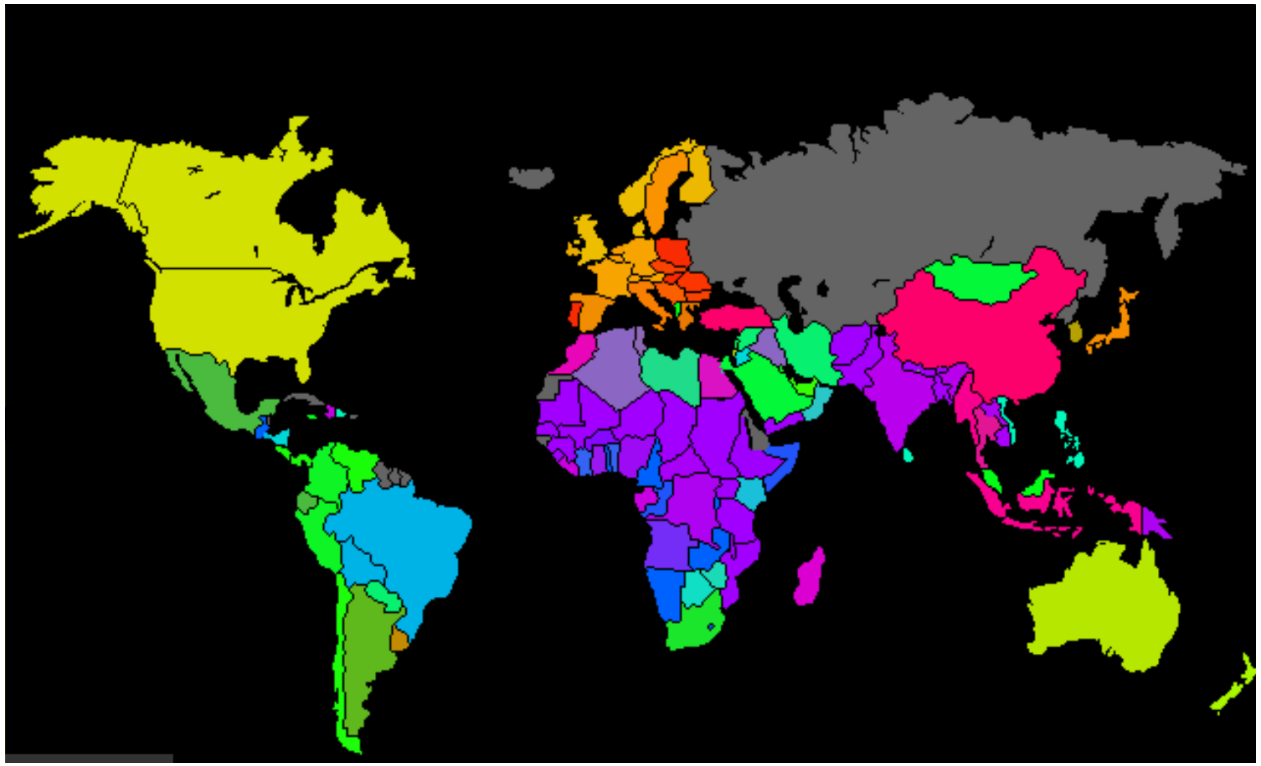


Classificação dos países usando um Self-Organizing Map

Fonte: Neural Networks Research Centre

Aplicação do algoritmo SOM

World Poverty Map usando Self-Organizing Maps



Fonte: Neural Networks Research Centre

REFERÊNCIAS


DO, C. B.; BATZOGLOU, S. What is the expectation maximization algorithm? *Nature biotechnology*, Nature Publishing Group US New York, v. 26, n. 8, p. 897–899, 2008.


ROSLI, N. R. M.; YAHYA, K. Using non-supervised artificial neural network for determination of anthropogenic disturbance in a river system. *Tropical life sciences research*, School of Medical Sciences, Universiti Sains Malaysia, v. 28, n. 2, p. 189, 2017.