Comparação de Algoritmos de Agrupamento

Beatriz Farias do Nascimento
DRE 122053127

Natan Azevedo Gomes
DRE 122047435

Thomas Cardoso de Miranda
DRE 122050797

***Abstract*—Implementação e comparação de três algoritmos de agrupamento não supervisionado: Kohonen (SOM), K-means e DBSCAN no conjunto de dados Iris Dataset.**

1. Introdução ao problema de Agrupamento

Na era da informação, a capacidade de extrair conhecimento e identificar padrões a partir de grandes volumes de dados brutos é um diferencial competitivo e científico. Grande parte desses dados é gerada sem rótulos ou categorias predefinidas, tornando impossível o uso de técnicas de aprendizagem supervisionada. É nesse cenário que o problema de agrupamento (ou clusterização) se estabelece como uma das tarefas mais fundamentais e desafiadoras da Aprendizagem de Máquina Não Supervisionada.

O objetivo central do agrupamento é organizar um conjunto de objetos de dados em subgrupos, ou clusters, de forma que os itens dentro de um mesmo cluster sejam mais semelhantes entre si do que com os itens de outros clusters. Contudo, a aparente simplicidade dessa definição esconde uma grande complexidade: não há uma única ”resposta certa”. A validade de um agrupamento depende do contexto, da estrutura dos dados e da própria definição de ”semelhança”.

Para lidar com essa complexidade, diversas famílias de algoritmos foram desenvolvidas, cada uma partindo de uma premissa diferente sobre o que constitui um ”grupo”. A escolha do algoritmo correto é, portanto, crucial para o sucesso da análise. Este relatório se propõe a implementar, na linguagem Python, e comparar criticamente três algoritmos icônicos que representam abordagens distintas para o problema:

- K-means: A clássica abordagem de particionamento baseada em centroides, que busca por clusters esféricos e compactos, sendo um pilar pela sua eficiência e simplicidade.

- DBSCAN: Um método exemplar da família baseada em densidade, que redefine um cluster como uma região contínua de alta concentração de pontos, permitindo-lhe encontrar grupos de formatos arbitrários e identificar ruído.

- Kohonen (SOM): Uma sofisticada técnica de aprendizagem competitiva inspirada em redes neurais, que não apenas agrupa os dados, mas também os mapeia em um espaço de baixa dimensão (geralmente 2D), preservando as relações topológicas entre eles e servindo como uma poderosa ferramenta de visualização.

Ao analisar e contrastar o funcionamento e os resultados desses três métodos em conjuntos de dados com estruturas variadas, este trabalho visa demonstrar como a filosofia de cada algoritmo influencia sua capacidade de descobrir padrões. O objetivo final é fornecer um panorama claro sobre as vantagens, desvantagens e os cenários de aplicação mais adequados para cada uma dessas importantes ferramentas da ciência de dados.

2. Descrição conceitual e matemática dos três algoritmos

2.1. DBSCAN

2.1.1. Descrição Conceitual.

- **Abordagem:** Baseada em densidade.
- **Objetivo:** Encontrar áreas de alta densidade no espaço de dados, que são separadas por áreas de baixa densidade.
- **Conceitos-chave:**
  - ϵ **(eps):** O raio de vizinhança em torno de um ponto.
  - **min_samples:** O número mínimo de pontos necessários dentro do raio ϵ para que uma região seja considerada densa.
  - **Ponto Central (Core Point):** Um ponto que possui pelo menos ‘min_samples‘ vizinhos (incluindo ele mesmo) dentro de um raio ϵ.
  - **Ponto de Borda (Border Point):** Um ponto que está na vizinhança de um ponto central, mas não é um ponto central.
  - **Ruído (Noise):** Um ponto que não é central nem de borda.
- **Resultado:** Consegue encontrar clusters de formatos arbitrários e é robusto a ruído.

2.1.2. Descrição Matemática.

Dado um conjunto de dados D, os parâmetros $\epsilon > 0$ e min_samples $\in N$..

Vizinhança-ϵ:. A vizinhança de um ponto $\mathbf{p}$ é o conjunto de pontos a uma distância menor ou igual a ϵ.

$$N_\epsilon(\mathbf{p}) = \{\mathbf{q} \in D \mid \text{dist}(\mathbf{p}, \mathbf{q}) \leq \epsilon\}$$

Ponto Central:. Um ponto $\mathbf{p}$ é um ponto central se sua vizinhança é densa.

$$|\mathrm{N}_\epsilon(\mathbf{p})| \geq \text{min\_samples}$$

Alcançabilidade por Densidade (Directly Density-Reachable):. Um ponto $\mathbf{q}$ é diretamente alcançável por densidade a partir de $\mathbf{p}$ se $\mathbf{q} \in N_\epsilon(\mathbf{p})$ e $\mathbf{p}$ é um ponto central.

Cluster:. Um cluster C é um subconjunto não vazio de D que satisfaz duas condições:

1) **Maximalidade:** Se $\mathbf{p} \in C$ e $\mathbf{q}$ é alcançável por densidade a partir de $\mathbf{p}$, então $\mathbf{q} \in C$.
2) **Conectividade:** Para todos os pontos $\mathbf{p}, \mathbf{q} \in C$, $\mathbf{p}$ e $\mathbf{q}$ são mutuamente alcançáveis por densidade.

2.2. K-Means

2.2.1. Descrição Conceitual.

- **Abordagem:** Particionamento baseado em centroides.
- **Objetivo:** Particionar um conjunto de N pontos de dados em K clusters pré-definidos.
- **Funcionamento:** É um processo iterativo que alterna entre duas fases:
    1) **Fase de Atribuição (Assignment):** Cada ponto de dado é atribuído ao cluster cujo centroide (média) é o mais próximo.
    2) **Fase de Atualização (Update):** O centroide de cada cluster é recalculado como a média de todos os pontos que foram atribuídos a ele.
- **Resultado:** Tende a encontrar clusters de formato esférico (globular) e de tamanhos semelhantes.

2.2.2. Descrição Matemática. O objetivo do K-means é minimizar a soma dos quadrados dentro dos clusters (WCSS), também conhecida como inércia (J). Dado um conjunto de dados $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ e um número de clusters K:

$$J = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in C_k} ||\mathbf{x}_i - \boldsymbol{\mu}_k||^2$$

Onde C_k é o conjunto de pontos no cluster k e $\boldsymbol{\mu}_k$ é o seu centroide.

Fase de Atribuição (iteração t):. Cada ponto $\mathbf{x}_i$ é atribuído ao cluster C_k que minimiza a distância ao seu centroide $\boldsymbol{\mu}_k^{(t-1)}$ da iteração anterior.

$$C_k^{(t)} = \{\mathbf{x}_i : ||\mathbf{x}_i - \boldsymbol{\mu}_k^{(t-1)}||^2 \leq ||\mathbf{x}_i - \boldsymbol{\mu}_j^{(t-1)}||^2 \quad \forall j, 1 \leq j \leq K\}$$

Fase de Atualização (iteração t):. O novo centroide $\boldsymbol{\mu}_k^{(t)}$ é a média dos pontos em $C_k^{(t)}$.

$$\boldsymbol{\mu}_k^{(t)} = \frac{1}{|C_k^{(t)}|} \sum_{\mathbf{x}_i \in C_k^{(t)}} \mathbf{x}_i$$

2.3. Kohonen (SOM - Self-Organizing Map)

2.3.1. Descrição Conceitual.

- **Abordagem:** Rede neural de aprendizagem competitiva não supervisionada.
- **Objetivo:** Produzir uma representação de baixa dimensão (geralmente 2D), chamada de mapa, a partir de dados de alta dimensão, preservando as relações topológicas dos dados originais.
- **Funcionamento:** Consiste em uma grade de neurônios. O treinamento ocorre em um processo de três etapas:
    1) **Competição:** Para cada ponto de dado de entrada, todos os neurônios competem para ver qual é o mais semelhante. O neurônio vencedor é chamado de Best Matching Unit (BMU).
    2) **Cooperação:** O BMU e seus neurônios vizinhos na grade são ativados.
    3) **Adaptação:** Os vetores de peso do BMU e de seus vizinhos são ajustados para se tornarem mais parecidos com o ponto de dado de entrada.
- **Resultado:** Um mapa treinado onde neurônios vizinhos representam clusters de dados semelhantes.

2.3.2. Descrição Matemática.

Competição:. Para um vetor de entrada $\mathbf{x}(t)$, o Best Matching Unit (BMU) é o neurônio c cujo vetor de pesos $\mathbf{w}_c$ é o mais próximo de $\mathbf{x}(t)$.

$$c(\mathbf{x}) = \arg\min_j \{||\mathbf{x}(t) - \mathbf{w}_j(t)||\}$$

Adaptação:. O vetor de pesos de todos os neurônios j é atualizado para se aproximar do vetor de entrada, com uma intensidade que depende da distância ao BMU no mapa.

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \eta(t) \cdot h_{j,c}(t) \cdot (\mathbf{x}(t) - \mathbf{w}_j(t))$$

Onde:

- t é a iteração (tempo).
- $\eta(t)$ é a **taxa de aprendizado**, uma função monotonicamente decrescente (ex: $\eta_0 \cdot \exp(-t/\tau_1)$).
- $h_{j,c}(t)$ é a **função de vizinhança** que define a influência do BMU (c) sobre um neurônio vizinho j. Geralmente é uma função Gaussiana:

$$h_{j,c}(t) = \exp\left(-\frac{\text{dist\_grid}(j,c)^2}{2\sigma(t)^2}\right)$$

- $\sigma(t)$ é o **raio da vizinhança**, que também decresce com o tempo, diminuindo a área de influência.

3. Explicação do dataset Iris

O conjunto de dados Iris é um dos benchmarks mais famosos e clássicos na área de Aprendizagem de Máquina e reconhecimento de padrões. Introduzido pelo estatístico e biólogo britânico Ronald Fisher em seu artigo de 1936, este dataset é amplamente utilizado para testar e demonstrar o funcionamento de algoritmos de classificação e agrupamento devido às suas características bem definidas.

O dataset é composto por 150 amostras de flores de Íris, divididas igualmente em três espécies distintas. Para cada amostra, foram medidos quatro atributos quantitativos.

Estrutura do Dataset:

- **Número total de Amostras:** 150
- **Número de Classes (Espécies):** 3
  - *Iris Setosa*
  - *Iris Versicolour*
  - *Iris Virginica*
- **Amostras por Classe:** 50 por espécie, o que o torna um conjunto de dados perfeitamente **balanceado**.
- **Número de Atributos (Features):** 4, todos medidos em centímetros:
  1) Comprimento da Sépala (Sepal Length)
  2) Largura da Sépala (Sepal Width)
  3) Comprimento da Pétala (Petal Length)
  4) Largura da Pétala (Petal Width)
- **Valores Ausentes:** Nenhuma das amostras possui valores ausentes.

Características para Análise de Agrupamento:

O dataset Iris é popular em estudos comparativos devido às suas características de separabilidade. Uma das classes, a Iris Setosa, é linearmente separável das outras duas. No entanto, as outras duas classes, Iris Versicolour e Iris Virginica, não são linearmente separáveis entre si, apresentando uma sobreposição no espaço de atributos.

Essa estrutura o torna um excelente caso de teste: ele permite avaliar a capacidade de um algoritmo de agrupamento em identificar tanto um cluster bem definido e isolado (Setosa) quanto em lidar com clusters que se sobrepõem e são mais difíceis de distinguir (Versicolour e Virginica).

Neste trabalho, o conjunto de dados Iris será utilizado para avaliar e comparar a performance dos algoritmos K-means, DBSCAN e SOM, observando a capacidade de cada um em reconstruir os três agrupamentos naturais correspondentes às espécies de flores.

4. Implementações dos algoritmos

4.1. DBSCAN

.

O algoritmo *DBSCAN* (*Density-Based Spatial Clustering of Applications with Noise*) foi implementado manualmente com base na definição proposta por Ester et al. A abordagem consiste em agrupar pontos com base na densidade de vizinhos ao redor de cada ponto, sem a necessidade de definir previamente o número de clusters.

A implementação segue os seguintes passos principais:

1) **Identificação de vizinhos:** Para cada ponto do conjunto de dados, calcula-se a distância euclidiana em relação a todos os demais pontos. Aqueles cuja distância é menor ou igual ao parâmetro ε (*eps*) são considerados vizinhos. Essa etapa corresponde à chamada *region query*.
2) **Classificação dos pontos:** Com base na contagem de vizinhos:
   - Um ponto é classificado como **núcleo** (*core point*) se possui ao menos $minPts$ vizinhos dentro do raio ε.
   - Um ponto é considerado **de borda** (*border point*) se está dentro do raio ε de um núcleo, mas não possui vizinhos suficientes para ser núcleo.
   - Pontos que não se enquadram nas categorias anteriores são rotulados como **ruído** (*noise points*).
3) **Expansão dos clusters:** A partir de cada núcleo identificado, o algoritmo realiza uma expansão recursiva do cluster, incluindo os vizinhos densamente conectados (outros núcleos) e os pontos de borda. Para garantir que um ponto seja processado apenas uma vez, é utilizado um vetor booleano `visited`.
4) **Organização dos resultados:** Ao final da execução, a implementação retorna:
   - `clusters`: um array que identifica a qual cluster um ponto pertence.
   - `core_points_indice`: o conjunto de pontos classificados como núcleos.
   - `border_points_indice`: o conjunto de pontos de borda, vizinhos de núcleos que não atendem ao critério de densidade.
   - `noise_points_indice`: o conjunto de pontos classificados como ruído, que não pertencem a nenhum cluster.
   - `num_clusters`: número de clusters total.

4.2. K-Means

O algoritmo *K-Means* foi implementado manualmente com base na definição clássica de clustering centrado em partições. Ao contrário do DBSCAN, o K-Means exige que o número de clusters (k) seja definido previamente. O objetivo é particionar os dados em k grupos, minimizando a soma das distâncias quadradas entre os pontos e seus respectivos centróides.

A implementação segue os seguintes passos principais:

1) **Inicialização dos centróides:** Seleciona-se aleatoriamente k pontos do conjunto de dados como

centróides iniciais. Esses centróides servirão como referência para a formação inicial dos clusters.
css Copiar Editar

2) **Atribuição dos pontos aos clusters:** Para cada ponto no conjunto de dados, calcula-se a distância euclidiana em relação a cada centróide. O ponto é então atribuído ao cluster cujo centróide estiver mais próximo.
3) **Atualização dos centróides:** Para cada cluster formado, calcula-se a média de todos os pontos atribuídos a ele, e essa média se torna o novo centróide. Caso um cluster fique vazio (sem pontos atribuídos), seu centróide é re-inicializado aleatoriamente com base no conjunto de dados.
4) **Critério de parada:** O algoritmo itera entre os passos de atribuição e atualização até que os centróides deixem de variar significativamente (convergência) ou até atingir um número máximo de iterações definido pelo parâmetro `max_iters`.
5) **Organização dos resultados:** Ao final da execução, a implementação retorna:
   - `clusters`: um array indicando o índice do cluster atribuído a cada ponto.
   - `centroids`: coordenadas finais dos k centróides.
   - `metric`: valor da **inércia**, que representa a soma das distâncias quadradas entre cada ponto e o centróide do seu cluster. Esse valor é comumente usado como métrica de avaliação da compactação dos clusters.

4.3. SOM (Self-Organizing Map)

O algoritmo *SOM* (Mapa Auto-Organizável) foi implementado manualmente com base na proposta de Kohonen. Trata-se de uma técnica de aprendizado não supervisionado que projeta dados de alta dimensionalidade em uma grade de neurônios bidimensional, preservando relações topológicas. Ao contrário de algoritmos como K-Means, o SOM considera a vizinhança entre neurônios, o que favorece uma organização mais estruturada do espaço de saída.

A implementação segue os seguintes passos principais:

1) **Inicialização da grade de neurônios:** Os pesos dos neurônios são inicializados aleatoriamente. Cada neurônio da grade possui um vetor de pesos com a mesma dimensionalidade dos dados de entrada. Parâmetros como taxa de aprendizado inicial, raio de vizinhança (σ) e número de épocas também são definidos nesta etapa.
kotlin Copiar Editar
2) **Identificação do neurônio BMU:** Para cada vetor de entrada, calcula-se a distância euclidiana até todos os neurônios da grade. O neurônio com a menor distância é chamado de **BMU** (*Best Matching Unit*), sendo aquele que melhor representa a entrada atual.
3) **Atualização dos pesos dos neurônios vizinhos:** Uma vez identificado o BMU, os pesos de todos os neurônios dentro de um raio de vizinhança são atualizados proporcionalmente à sua proximidade com o BMU. A função de influência utilizada é uma Gaussiana, que depende da distância topológica na grade e do parâmetro σ.
4) **Ajuste dinâmico dos parâmetros:** Ao longo das épocas de treinamento, a taxa de aprendizado e o raio de vizinhança são reduzidos linearmente, promovendo uma estabilização gradual da grade.
5) **Atribuição dos clusters:** Após o treinamento, cada ponto do conjunto de dados é atribuído ao seu BMU correspondente. O identificador de cluster pode ser definido como o índice linear do neurônio BMU na grade 2D.
6) **Organização dos resultados:** A implementação retorna:
   - `clusters`: um array que associa cada ponto de dado ao seu neurônio BMU (ou seja, ao seu cluster).
   - `grade`: a grade final de neurônios com os pesos treinados, representando o mapa auto-organizado.

5. Análise dos resultados

5.1. *Ground Truth*

Para efeitos de comparação, foi plotado um gráfico para cada combinação de *features* do dataset a partir da classificação já fornecida.

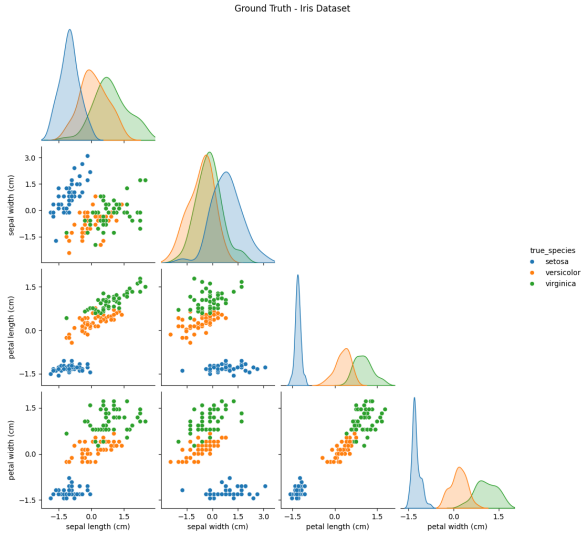

Figure 1. Gráficos do *Ground Truth*

5.2. DBSCAN

Foram testadas todas as combinações dos dois parâmetros do DBSCAN: ϵ foi variado de 0.4 a 1.2, com

intervalos de 0.1, e minPts foi variado de 2 até 10, com intervalos de 2. Foi possível observar que o número de *clusters* diminuía conforme o valor dos parâmetros aumentava, com parâmetros de valores menores também gerando *clusters* com menos pontos e mais *noise*.

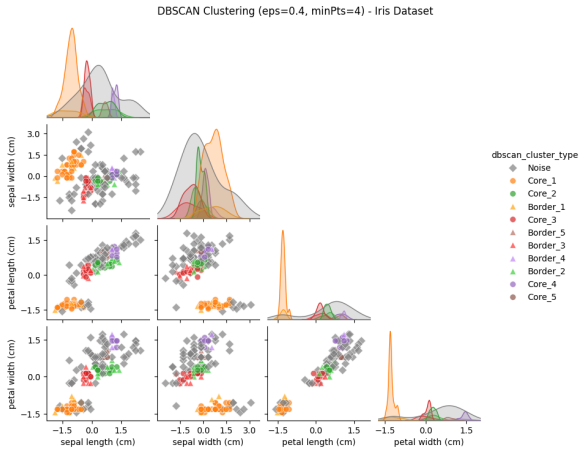

Figure 2. DBSCAN com ϵ de 0.4 e minPts de 4

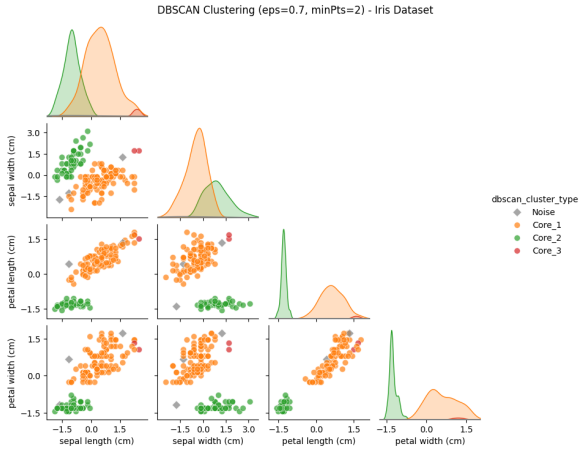

Figure 3. DBSCAN com ϵ de 0.7 e minPts de 2

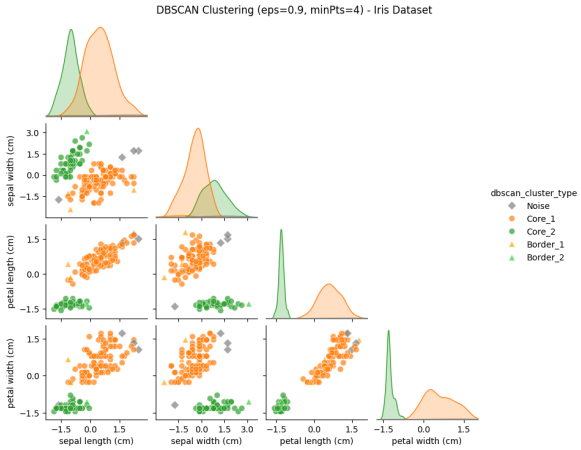

Figure 4. DBSCAN com ϵ de 0.9 e minPts de 4

Como é possível observar nos gráficos, o DBSCAN identifica corretamente o *cluster* correspondente à classificação *setosa*, porém não consegue dividir *versicolor* e *virginica* de forma satisfatória. Os parâmetros que geraram 3 *clusters* resultaram em casos semelhantes ao da 3: um terceiro cluster formado por apenas alguns pontos mais isolados. Elevar os parâmetros para além dos da 4 resultava em gráficos muito similares nos casos testados, e elevar para muito além disso levaria todos os pontos a estarem em um mesmo *cluster*. Isso condiz com o embasamento teórico do DBSCAN, já que *versicolor* e *virginica* não estão divididos por uma área de baixa densidade, ao contrário dos dois e *setosa*.

5.3. K-Means

Foram testados valores para k de 2, 3 e 4. Para k igual a 2, os gráficos resultantes foram muito similares ao melhor resultado encontrado para o DBSCAN, o que condiz com os dois agrupamentos isolados que podem ser observados.

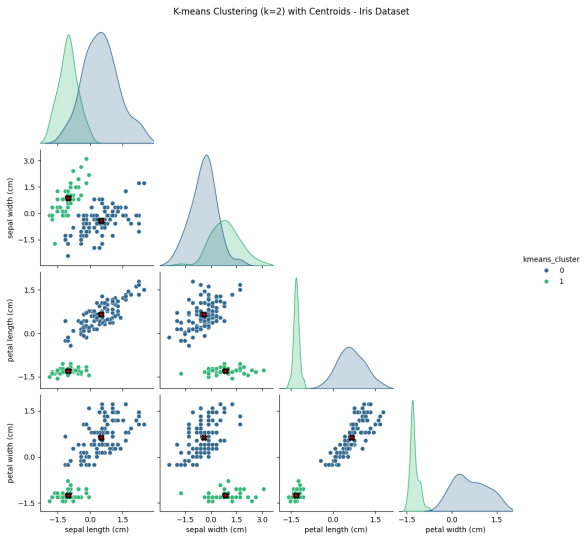

Figure 5. K-Means com k valendo 2

Para k igual a 3, o agrupamento que representa *versicolor* e *virginica* é dividido em dois *clusters* de forma semelhante à divisão real, algo que o DBSCAN não foi capaz de fazer.

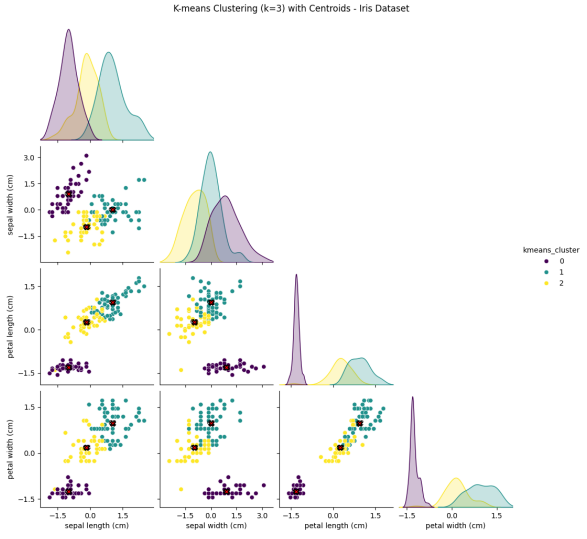

Figure 6. K-Means com k valendo 3

Para k igual a 4, o conjunto das *setosas* acaba sendo dividido em dois, o que está mais distante da verdade e permite inferir que aumentar ainda mais o valor de k geraria resultados piores. Com isso, conclui-se que o valor ideal para k neste *dataset* é 3.

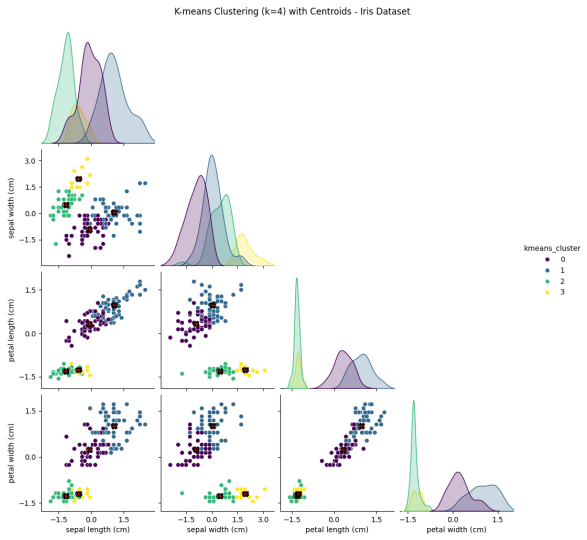

Figure 7. K-Means com k valendo 4

5.4. *Self-Organizing Map* de Kohonen

Para esse algoritmo, foram testadas todas as combinações dos seguintes parâmetros:

- **grade**: 2x2, 4x4, 5x5, 2x5, 5x2
- ***learning rate***: 0.1, 0.2, 0.3
- σ: 1.0, 2.0, 3.0
- ***epochs***: 100, 500

Os melhores resultados foram encontrados para a grade de tamanho 2x2, na qual o conjunto das *setosas* é claramente identificado e os outros dois são divididos em três clusters. Variar os outros parâmetros não parece surtir muito efeito.

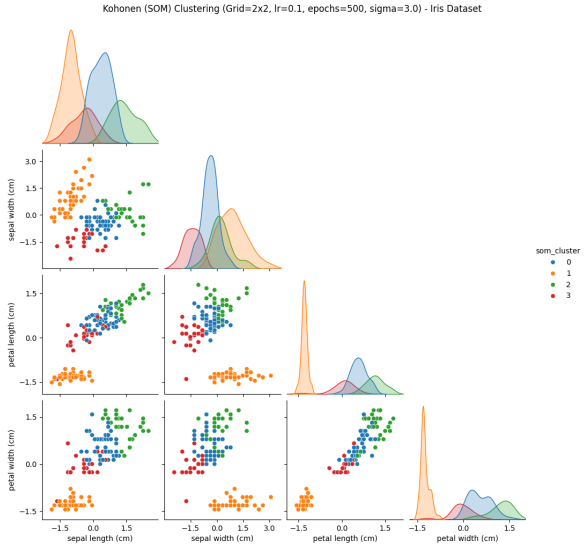

Figure 8. Kohonen com grade tamanho 2x2, *learning rate* de 0.1, 500 *epochs* e σ igual a 3.0

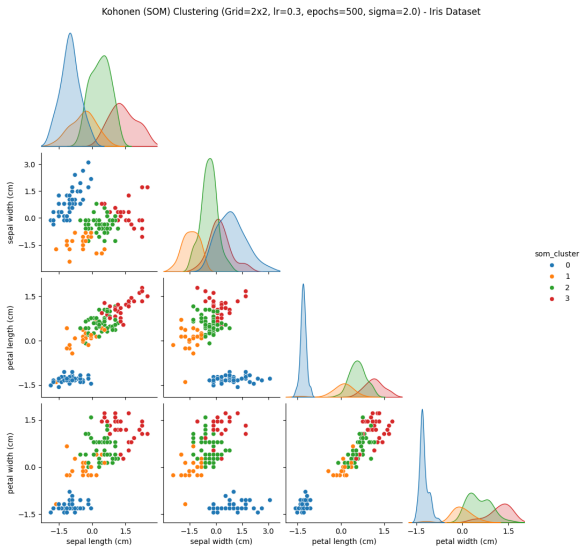

Figure 9. Kohonen com grade tamanho 2x2, *learning rate* de 0.3, 500 *epochs* e σ igual a 2.0

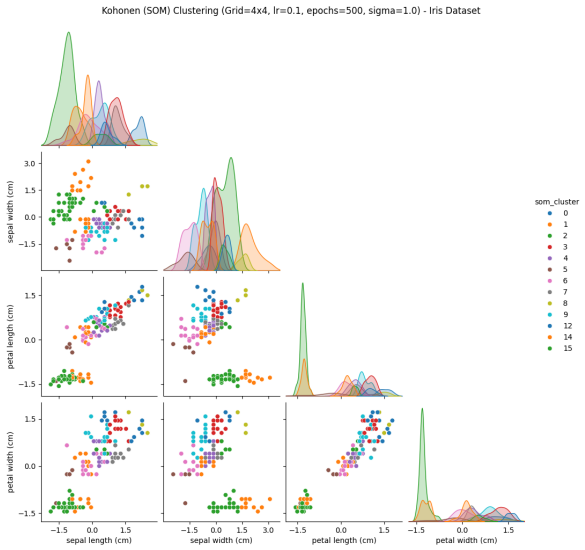

Figure 10. Kohonen com grade tamanho 4x4, *learning rate* de 0.1, 500 *epochs* e σ igual a 1.0

6. Conclusão

Este trabalho explorou a aplicação e o desempenho de três importantes algoritmos de agrupamento não supervisionado (*K-means*, DBSCAN e *Self-Organizing Map* (SOM) de Kohonen) no clássico dataset Iris. Através das implementações detalhadas, foi possível observar as diferentes filosofias por trás de cada método: o *K-means*, com sua abordagem baseada em centroides, revelou-se eficaz para identificar clusters esféricos; o DBSCAN, por sua vez, demonstrou sua capacidade de descobrir agrupamentos com formas arbitrárias e de lidar com ruído, operando por densidade; e o SOM de Kohonen evidenciou seu poder de mapeamento topológico e visualização, organizando os dados em uma representação de baixa dimensão.

A análise no dataset Iris, com suas classes de separabilidade variada, permitiu uma comparação prática da robustez e adequação de cada algoritmo a diferentes estruturas de dados. Os resultados obtidos reforçam a premissa de que a escolha do algoritmo de agrupamento ideal depende intrinsecamente das características do problema e da natureza dos dados. Compreender as vantagens e limitações de cada técnica é, portanto, fundamental para a extração de padrões significativos em cenários de dados não rotulados.”

7. Código

O código usado no trabalho está disponível no GitHub: https://github.com/beatriz-farias/algoritmos-agrupamento.

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.