Comparação de Algoritmos de Agrupamento Não Supervisionado: K-means, DBSCAN e SOM no Dataset Iris

Samuel Vitor Pedro e Araujo
Programa de Engenharia de Sistemas e Computação
Universidade Federal do Rio de Janeiro
Emails: samvpa.20221@poli.ufrj.br

***Abstract*—Este trabalho implementa e compara três algoritmos de agrupamento não supervisionado – K-means, DBSCAN e Self-Organizing Maps (SOM) – aplicados ao dataset Iris. São investigadas diferentes configurações dos parâmetros de cada método, analisando seu impacto nos resultados de agrupamento.**

I. Introdução

O agrupamento, ou *clustering*, é uma técnica de aprendizado não supervisionado que visa organizar dados em subconjuntos (clusters) de forma que elementos do mesmo grupo sejam mais similares entre si do que em relação aos elementos de outros grupos. Esta técnica é amplamente utilizada em problemas onde não há rótulos disponíveis previamente, como segmentação de mercado, análise de comportamento de usuários, agrupamento de imagens, compressão de dados e em bioinformática para agrupamento de genes.

Diferentemente do aprendizado supervisionado, que requer dados rotulados para treinar um modelo, o agrupamento busca revelar estruturas subjacentes nos dados de forma autônoma, explorando padrões e densidades. Por este motivo, o problema de agrupamento é considerado um dos pilares do aprendizado de máquina.

Neste trabalho, implementam-se três algoritmos clássicos de clustering: **K-means**, **DBSCAN** e **Self-Organizing Maps (SOM)**. O objetivo principal é investigar como diferentes configurações de parâmetros impactam os resultados de agrupamento ao aplicar estes métodos ao conhecido dataset Iris.

II. Fundamentação Teórica

A. K-means

O algoritmo K-means é um dos métodos de agrupamento mais populares, proposto por MacQueen em 1967. Seu objetivo é particionar um conjunto de n pontos em k grupos (clusters) de forma que a soma das distâncias quadráticas dos pontos ao centróide do cluster ao qual pertencem seja minimizada. Formalmente, busca-se minimizar o seguinte critério de custo:

$$J = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

onde C_i representa o conjunto de pontos pertencentes ao cluster i, e μ_i é o centróide do cluster i, calculado como a média aritmética dos pontos pertencentes a C_i.

O algoritmo segue um processo iterativo que alterna entre duas etapas principais: (i) a *atribuição* dos pontos aos clusters cujos centróides estão mais próximos, e (ii) a *atualização* dos centróides com base nos pontos atribuídos. O processo se repete até que a mudança nos centróides seja inferior a um limite pré-definido, ou até atingir um número máximo de iterações.

O K-means é eficiente e simples, mas assume implicitamente que os clusters têm formato esférico e distribuições aproximadamente semelhantes, sendo sensível a outliers e à inicialização dos centróides.

B. DBSCAN

O DBSCAN (Density-Based Spatial Clustering of Applications with Noise), introduzido por Ester et al. em 1996, é um algoritmo projetado para identificar agrupamentos com base na densidade dos pontos, permitindo detectar clusters de forma arbitrária e pontos que podem ser considerados ruído.

O algoritmo depende de dois parâmetros principais: ϵ, que define o raio da vizinhança de um ponto, e *minPts*, que indica o número mínimo de pontos necessário dentro da vizinhança para que o ponto seja considerado denso.

Um ponto é classificado como:

- *Core point*, se possui pelo menos *minPts* vizinhos dentro do raio ϵ.
- *Border point*, se possui menos de *minPts* vizinhos mas está na vizinhança de um ponto central.
- *Noise point*, se não atende a nenhum dos critérios acima.

A partir dos pontos centrais, o DBSCAN expande clusters incluindo pontos conectados diretamente e densamente alcançáveis. O algoritmo termina ao processar todos os pontos do conjunto de dados. A principal vantagem do DBSCAN é a capacidade de identificar clusters de forma irregular e ignorar ruídos sem precisar especificar o número de clusters a priori.

C. Self-Organizing Maps (SOM)

Os Mapas Auto-Organizáveis (SOM), introduzidos por Teuvo Kohonen em 1982, são um tipo de rede neural não supervisionada que projeta dados de alta dimensão em uma

grade discreta de menor dimensão (tipicamente 2D), preservando propriedades topológicas — isto é, pontos próximos no espaço original tendem a ser mapeados para neurônios vizinhos.

Cada neurônio da grade possui um vetor de pesos w_i com a mesma dimensão dos dados de entrada. Para cada amostra x, o algoritmo identifica o neurônio vencedor (Best Matching Unit — BMU), cujo vetor de pesos é mais próximo de x em termos de distância euclidiana. Em seguida, atualiza os pesos do BMU e também de seus vizinhos na grade de acordo com:

$$w_i(t+1) = w_i(t) + \eta(t)\, h_{ci}(t)\, (x - w_i(t))$$

onde $\eta(t)$ é a taxa de aprendizagem, e $h_{ci}(t)$ é a função de vizinhança que decresce com a distância entre o neurônio i e o BMU na grade, assim como ao longo do tempo.

Ao longo das iterações, o SOM forma um mapeamento ordenado onde regiões distintas da grade especializam-se em representar diferentes subconjuntos dos dados. Esta propriedade torna o SOM útil não apenas para clustering, mas também para visualização e redução de dimensionalidade.

D. Parâmetros equivalentes nos métodos

Embora cada algoritmo adote mecanismos distintos para formar agrupamentos, é possível estabelecer paralelos entre os papéis dos principais parâmetros. Estes parâmetros determinam, de forma equivalente, o número de grupos formados, o critério de proximidade, o tamanho mínimo necessário para se constituir um cluster, o critério de parada do treinamento e, no caso do SOM, a taxa de aprendizagem.

- **Número de grupos esperados:** No K-means, o número de clusters k é definido explicitamente. No SOM, é controlado pelo tamanho da grade (ex: 2×2, 3×3). No DBSCAN, o número de clusters emerge da escolha de ϵ e *minPts*.
- **Distância mínima entre pontos:** No K-means, está implícita pela busca do centróide que minimiza a distância intra-cluster. O DBSCAN utiliza diretamente o parâmetro ϵ, que representa o raio de vizinhança ao redor de cada ponto para considerar conexões. No SOM, a vizinhança é regulada pelo raio σ, que define a influência do neurônio vencedor sobre seus vizinhos durante a atualização.
- **Tamanho mínimo do grupo:** O DBSCAN possui o parâmetro *minPts*, que indica o número mínimo de pontos dentro do raio ϵ para formar um cluster. No K-means e SOM não existe um parâmetro equivalente direto, mas o critério de formação de clusters acaba sendo indiretamente influenciado pelos demais parâmetros e pela distribuição dos dados.
- **Critério de parada:** No K-means, geralmente utiliza-se a mudança mínima dos centróides entre iterações como critério de parada. O DBSCAN para ao visitar todos os pontos. O SOM é governado pelo número de épocas (epochs), determinando quantas vezes o dataset será percorrido para ajustar os pesos.
- **Taxa de aprendizagem:** Embora o K-means e o DBSCAN não possuam uma taxa de aprendizagem, o SOM é atualizado por um parâmetro *learning_rate* que controla a velocidade de ajuste dos pesos durante o treinamento.

III. Dataset Iris

O dataset Iris é um dos conjuntos de dados mais famosos e amplamente utilizados no campo da aprendizagem de máquina e estatística, servindo frequentemente como um *benchmark* para algoritmos de classificação e agrupamento. Coletado por Edgar Anderson em 1936 e popularizado por Ronald Fisher em 1936, ele contém 150 amostras de flores de íris, categorizadas em três espécies distintas: *Iris setosa*, *Iris versicolor* e *Iris virginica*. Cada espécie contribui com 50 amostras, totalizando um dataset balanceado em relação às classes.

Para cada amostra, são fornecidas quatro características quantitativas, medidas em centímetros:

- Comprimento da sépala (*sepal length*)
- Largura da sépala (*sepal width*)
- Comprimento da pétala (*petal length*)
- Largura da pétala (*petal width*)

A espécie *Iris setosa* é geralmente considerada linearmente separável das outras duas, enquanto as espécies *Iris versicolor* e *Iris virginica* apresentam um grau maior de sobreposição em suas características, tornando a distinção entre elas um desafio interessante para algoritmos de agrupamento não supervisionado. Essa característica do dataset Iris, com uma classe bem separada e duas com alguma intersecção, o torna ideal para demonstrar como diferentes algoritmos lidam com variados níveis de separabilidade entre os grupos.

IV. Implementação

Todos os algoritmos foram implementados do zero em Python, utilizando apenas as bibliotecas *numpy* para operações numéricas e *matplotlib* para visualização. Abaixo, detalhamos a estrutura de cada implementação, destacando as classes e seus métodos principais. O código-fonte completo está disponível no seguinte repositório GitHub:

https://github.com/samzax/
Trabalho-Final-Gestao-do-Conhecimento

A. K-means

A implementação do K-means está encapsulada na classe `KMeans`, que simula o comportamento iterativo do algoritmo, buscando a convergência dos centróides.

- `__init__(self, k, max_iter=100, tol=1e-4)`: Inicializa o algoritmo com o número de clusters k, um número máximo de iterações max_iter e uma tolerância tol para o critério de convergência dos centróides.
- `fit(self, X)`: Executa o processo de agrupamento. Inicializa os centróides aleatoriamente e, em um laço iterativo, atribui cada ponto ao cluster do centróide mais próximo e recalcula os centróides. O processo se repete até que os centróides mudem menos que tol ou que max_iter seja atingido.

- **`_assign_clusters(self, X)`**: Método auxiliar que calcula as distâncias dos pontos aos centróides atuais, atribuindo cada ponto ao cluster mais próximo.
- **`_update_centroids(self, X, labels)`**: Recalcula a posição dos centróides como a média dos pontos atribuídos a cada cluster.
- **`predict(self, X)`**: Retorna os rótulos dos clusters para um novo conjunto de dados, usando os centróides finais obtidos no ajuste.

B. DBSCAN

A implementação do DBSCAN está contida na classe **`DBSCAN`**, que identifica regiões densas e separa pontos considerados ruído.

- **`__init__(self, eps, min_pts)`**: Define os parâmetros principais: o raio ϵ da vizinhança e min_pts, o número mínimo de pontos para formar uma região densa.
- **`fit(self, X)`**: Inicia o processo de agrupamento iterando sobre cada ponto do dataset. Verifica se o ponto é um *core point* e, caso seja, expande o cluster a partir dele, incluindo todos os pontos densamente alcançáveis. Pontos que não se enquadram como centrais ou fronteira são marcados como ruído.
- **`_get_neighbors(self, X, point_idx)`**: Método auxiliar que retorna os índices de todos os pontos dentro do raio ϵ de um dado ponto.
- **`_expand_cluster(self, X, point_idx, neighbors, cluster_id)`**: Método recursivo que, a partir de um *core point*, expande o cluster incorporando vizinhos que também satisfaçam a densidade mínima.

C. Self-Organizing Maps (SOM)

A implementação do SOM está na classe **`SOM`**, responsável por treinar uma rede que projeta dados de alta dimensão em uma grade de neurônios.

- **`__init__(self, grid_shape, input_dim, sigma, learning_rate, epochs)`**: Define a forma da grade $grid\_shape$, a dimensão dos dados de entrada $input\_dim$, o raio inicial de vizinhança σ, a taxa de aprendizagem inicial $learning\_rate$ e o número de épocas de treinamento $epochs$. Os pesos dos neurônios são inicializados aleatoriamente.
- **`fit(self, X)`**: Método principal de treinamento. Para cada época e para cada ponto de dados, encontra o neurônio vencedor (BMU) e atualiza seus pesos e de seus vizinhos. Tanto $learning\_rate$ quanto σ decaem ao longo do tempo.
- **`_find_bmu(self, x)`**: Identifica o neurônio cujo vetor de pesos é mais próximo do ponto x, isto é, o Best Matching Unit (BMU).
- **`_update_weights(self, x, bmu_idx, epoch)`**: Atualiza os pesos do BMU e de seus vizinhos com base na distância na grade e nos parâmetros atuais. O decaimento exponencial garante uma convergência gradual.
- **`predict(self, X)`**: Para um novo conjunto de dados, retorna as coordenadas do BMU responsável por cada ponto, efetivamente agrupando-os aos neurônios da grade.

V. Experimentos e Resultados

A. Parâmetros testados

Algoritmo	Parâmetro	Valores Testados
K-means	k	2, 3, 4, 5
DBSCAN	ϵ, minPts	(0.3, 0.5, 0.7), (3, 5, 7)
SOM	Grade, σ, LR	(2x2, 3x3, 4x4), (0.5, 1.0), (0.3, 0.5)

Tabela I: Configurações testadas para cada algoritmo

B. Discussão dos resultados

A análise dos resultados obtidos nos experimentos com o dataset Iris revela como cada algoritmo de agrupamento reage de forma distinta às variações de seus parâmetros e à estrutura intrínseca dos dados.

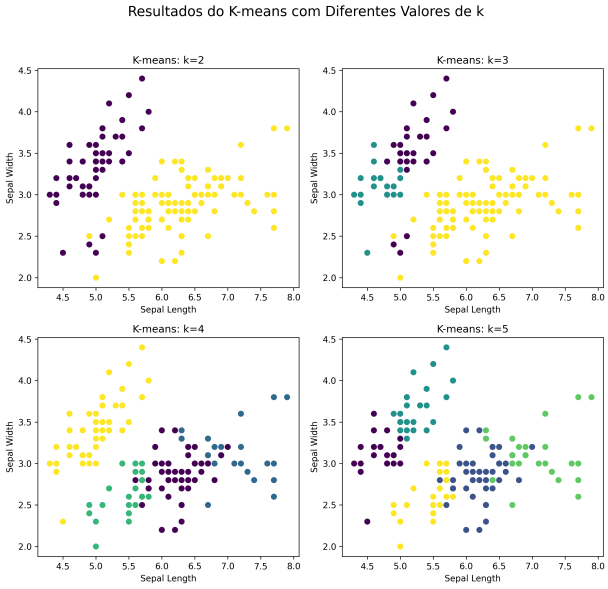

Fig. 1: Clusters obtidos pelo K-means para diferentes valores de k.

A aplicação do **K-means** demonstrou a **influência direta do parâmetro k no número e na formação dos clusters**. Como pode ser observado na Figura 1, quando $k = 2$, o algoritmo agrupou as 150 amostras em dois grandes grupos (97 e 53 pontos, respectivamente), o que não reflete a estrutura de três espécies inerente ao dataset Iris. No entanto, ao definir $k = 3$, observamos a formação de três clusters com tamanhos de 50, 61 e 39 pontos. Essa distribuição está notavelmente alinhada com as 50 amostras de cada uma das três espécies de íris, indicando que $k = 3$ é o valor mais apropriado para este dataset. Valores de k superiores, como $k = 4$ ou $k = 5$, levaram a uma **fragmentação excessiva**, onde clusters

naturais foram subdivididos, resultando em grupos menores e menos coesos (por exemplo, com $k = 5$, os tamanhos dos clusters foram 47, 23, 28, 22 e 30). Isso sublinha a necessidade crítica de um conhecimento prévio sobre o número esperado de grupos ou a utilização de métodos adicionais (como o método do cotovelo) para determinar o k ótimo no K-means, uma vez que o algoritmo sempre forçará a divisão dos dados no número de grupos especificado.

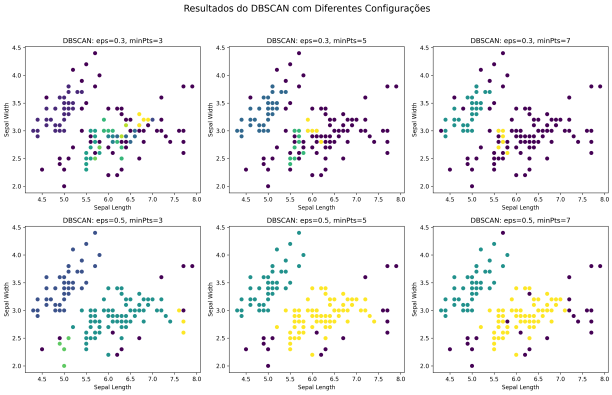

Fig. 2: Resultados do DBSCAN variando ϵ e minPts.

A análise do **DBSCAN** revelou sua **sensibilidade crítica aos parâmetros ϵ e minPts**, especialmente no que diz respeito à identificação de ruído e ao número de clusters emergentes. Conforme ilustrado na Figura 2, com um raio de vizinhança pequeno ($\epsilon = 0.3$), independentemente do valor de minPts, o algoritmo identificou um número excessivamente alto de pontos como ruído (variando de 68 a 108 pontos, quase metade do dataset), e os clusters formados eram numerosos e pequenos, fragmentando a estrutura dos dados. Por exemplo, com ($\epsilon = 0.3, minPts = 3$), foram encontrados 9 clusters com 68 pontos de ruído. Isso indica que $\epsilon = 0.3$ é muito restritivo para capturar a densidade natural do dataset Iris. Ao aumentar ϵ para 0.5, observamos uma **melhora drástica nos resultados**: o número de pontos de ruído diminuiu significativamente (para 10 a 24 pontos), e a formação de clusters se tornou mais robusta. A configuração com ($\epsilon = 0.5, minPts = 5$) resultou em apenas 2 clusters principais (com 49 e 84 pontos, respectivamente) e apenas 17 pontos de ruído, sugerindo que o DBSCAN conseguiu identificar duas regiões densas principais, provavelmente unindo as duas espécies de Iris que possuem maior sobreposição, enquanto a espécie mais distinta formou seu próprio cluster. Este comportamento ressalta a capacidade do DBSCAN de identificar clusters de formas arbitrárias e lidar explicitamente com ruído, mas também a sua forte dependência da calibração de seus parâmetros para refletir a estrutura real dos dados.

Os resultados do **Self-Organizing Map (SOM)** demonstraram sua capacidade de mapear dados de alta dimensão em uma grade 2D, preservando a topologia. O **tamanho da grade** atua como um análogo ao número de clusters, com uma grade 2×2 (4 neurônios) resultando em uma distribuição de pontos que, embora mostrasse alguma especialização (e.g.,

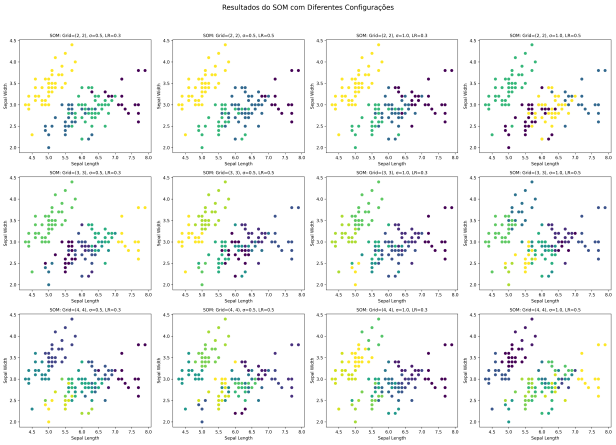

Fig. 3: Agrupamentos projetados pela grade SOM.

para $\sigma = 0.5$ e LR=0.3, {0: 50, 1: 34, 2: 49, 3: 17}), poderia ser insuficiente para as três classes distintas do Iris. À medida que o tamanho da grade aumentou para 3×3 (9 neurônios), como pode ser visto na Figura 3, observamos uma **melhor diferenciação espacial dos dados**, com alguns neurônios atraindo um número significativamente maior de pontos (e.g., para $\sigma = 0.5$ e LR=0.3, neurônio 2 com 50 pontos), indicando a formação de regiões bem definidas para cada tipo de íris. A grade 4×4 (16 neurônios), por sua vez, ofereceu uma granularidade ainda maior, permitindo capturar subestruturas, mas também resultando em muitos neurônios com poucos ou nenhum ponto mapeado. Os parâmetros σ **(raio de vizinhança) e** $learning\_rate$ **(taxa de aprendizagem)** influenciaram a suavidade e a velocidade com que o mapa se organizou; um σ maior promoveu uma organização mais generalizada, enquanto um σ menor resultou em especializações mais localizadas dos neurônios, impactando a forma como os pontos de dados se agruparam nos diferentes nós da grade. A visualização dos SOMs é crucial para entender como os pontos de dados se distribuem e formam padrões topológicos na grade.

VI. Conclusão

Este trabalho implementou e comparou os algoritmos de agrupamento não supervisionado K-means, DBSCAN e Self-Organizing Maps (SOM) no dataset Iris, investigando a influência de seus parâmetros nos resultados. A experimentação demonstrou que o **K-means** requer um k predefinido, com $k = 3$ sendo o mais adequado ao dataset Iris, alinhando-se às classes reais. O **DBSCAN** revelou sua capacidade de identificar clusters baseados em densidade e ruído, mas sua eficácia foi altamente dependente da calibração de ϵ e minPts. Por sua vez, o **SOM** provou ser uma ferramenta robusta para mapeamento e visualização de dados de alta dimensão, preservando relações topológicas e permitindo a emergência de agrupamentos coerentes na grade.

Em síntese, a escolha do algoritmo de agrupamento ideal é ditada pela natureza dos dados e pelas premissas sobre a estrutura dos clusters. K-means é eficaz para grupos esféricos

e bem definidos; DBSCAN sobressai em detecção de formas irregulares e ruído; e SOM é valioso para visualização e organização topológica. A análise reforça a importância crítica da seleção e ajuste de parâmetros para que os métodos de agrupamento revelem insights significativos dos dados. Trabalhos futuros podem explorar a aplicação desses métodos em datasets de maior complexidade ou dimensões, bem como a incorporação de métricas de validação de clusters intrínsecas para uma avaliação mais objetiva do desempenho.

VII. Referências

[1] [1] T. A. S. Costa, "Machine Learning from Scratch: Implementando K-Means Clustering em Python," *iAcomCafé*, 22 de setembro de 2023. [Online]. Disponível em: https://iacomcafe.com.br/machine-learning-kmeans-python-scratch/

[2] [2] A. Rozanov, "DBSCAN, Explained in 5 Minutes. Fastest implementation in python," *Medium (Towards Data Science Archive)*, 20 de agosto de 2021. [Online]. Disponível em: https://medium.com/data-science/dbscan-explained-in-5-minutes-133f6a9766e4

[3] [3] Tech with Tushar, "K-means Clustering in Python with Evaluation, Silhouette on Iris dataset," *YouTube*, 11 de julho de 2020. [Vídeo]. Disponível em: https://www.youtube.com/watch?v=56ZFz6hcSJs