O Algoritmo K-médias Geodésico para Agrupamento de Dados Baseado em Grafos

Antônio C. A. de Azevedo and Alexandre L. M Levada

Departamento de Computação, Universidade Federal de São Carlos, Rod. Washington Luis, km. 235, São Carlos, 13565-905, SP, Brasil.

Contributing authors: azevedoantonio@estudante.ufscar.br; alexandre.levada@ufscar.br;

Abstract

O agrupamento é uma das tarefas mais importantes em aprendizado de máquina e ciência de dados. Vários algoritmos de agrupamento foram propostos para mitigar limitações do aprendizado não supervisionado no reconhecimento de padrões. No entanto, o agrupamento de dados de alta dimensão ainda é um desafio. Neste artigo, propomos k-means topológicos, um método baseado em grafos para agrupamento de dados que usa o algoritmo de Dijkstra para calcular distâncias geodésicas entre pontos amostrais na variedade de dados aproximada e discreta. Além disso, a complexidade computacional do algoritmo proposto é razoavelmente baixa em comparação com técnicas modernas de agrupamento, pois é linear no número de amostras e também no número de arestas do grafo *k-NN*. Experimentos computacionais com conjuntos de dados reais mostram que o método proposto é capaz de melhorar a qualidade dos agrupamentos obtidos em termos de medidas de validade externa em comparação com k-means regulares.

Keywords: Agrupamento, k-means, dados com alta dimensionalidade, distancia geodésica, Dijkstra, Clustering

1 Introdução

O aprendizado de máquina testemunhou um crescimento sem precedentes nos últimos anos e, dentro desse domínio expansivo, o agrupamento de dados se destaca como uma técnica fundamental com implicações profundas para a análise de dados e reconhecimento de padrões [1–3]. Clustering, no domínio do aprendizado de máquina, é uma técnica que envolve agrupar pontos de dados semelhantes, com base em certas características inerentes, sem a necessidade de rótulos predefinidos. A premissa fundamental subjacente ao clustering é a identificação de agrupamentos ou padrões naturais dentro dos dados, o que facilita uma compreensão mais detalhada da estrutura [4]. Esta abordagem de aprendizagem não supervisionada tem aplicações de longo alcance em vários domínios, incluindo processamento de imagens, reconhecimento de padrões e mineração de dados [5].

O método de agrupamento mais popular é o k-means, um algoritmo particional, que visa dividir um conjunto de dados em k subgrupos ou clusters distintos e não sobrepostos, minimizando a soma das distâncias quadradas entre os pontos de dados e o centroide do cluster atribuído [6, 7]. As principais vantagens do k-means são: 1) eficiência computacional, pois é linear no número de amostras, tornando-o adequado para grandes conjuntos de dados; e 2) simplicidade e versatilidade, pois é fácil de implementar e aplicável em vários domínios, pois pode lidar com diferentes tipos de dados. No entanto, o k-means também tem várias limitações, entre as quais podemos citar: 1) como depende da distância euclidiana, sofre de um tipo de cegueira não esférica e alta sensibilidade a outliers em dados; e 2) dificuldade com clusters com densidades e formas variadas.

Algoritmos de clusterização modernos são frequentemente capazes de superar algumas limitações de k-means. *Hierarchical Density-Based Spatial Clustering of Applications with Noise*, ou simplesmente HDBSCAN, é um algoritmo de clusterização avançado que estende as capacidades de métodos tradicionais baseados em densidade [8]. A principal inovação está na sua capacidade de descobrir clusters com formas e densidades variadas em um conjunto de dados, sendo robusto ao ruído. O HDBSCAN opera construindo uma árvore de clustering hierárquica e, em seguida, extraindo clusters estáveis e significativos dessa estrutura [9]. As principais vantagens do HDBSCAN são: 1) tratamento de ruído, pois identifica e rotula explicitamente os pontos de ruído, fornecendo uma distinção clara entre clusters válidos e ruído nos dados; 2) flexibilidade nas formas dos clusters, pois não é restringido por suposições sobre a forma ou tamanho dos clusters, tornando-o adequado para conjuntos de dados com estruturas complexas; e 3) adaptabilidade a diferentes densidades, pois se adapta bem a clusters com densidades variadas, tornando-o particularmente eficaz em cenários onde os clusters exibem diferentes níveis de compactação. Por outro lado, as principais limitações do HDBSCAN são [10]: 1) complexidade computacional, pois é um algoritmo intensivo, especialmente para grandes conjuntos de dados; 2) sensibilidade aos parâmetros, pois o desempenho pode ser significativamente influenciado pelo ajuste de seus parâmetros; e 3) limitado para estrutura global, pois foca em estruturas de densidade local, e embora seja excelente para identificar microclusters. Outro problema com HDBSCAN está relacionado ao desempenho e aproximação. Para grandes conjuntos de dados, uma

amostra aleatória para k-means pode obter uma boa aproximação do resultado geral. Mas para HDBSCAN, não está claro como avaliá-lo em subconjuntos dos dados.

Apesar de muitos avanços, um problema em aberto no agrupamento é como lidar com dados de alta dimensionalidade, ou seja, situações em que o número de características m é uma ordem de grandeza maior do que o número de amostras n [11–13]. Para resolver esse problema, propomos o k-means topológico, ou simplesmente *top k-means*, um algoritmo baseado em grafo que calcula distâncias geodésicas entre amostras na variedade oculta dos dados usando o algoritmo de Dijkstra. Basicamente, a ideia é aproximar a varidade ocultas dos dados por um grafo k-NN e então usar o comprimento dos caminhos mais curtos como uma aproximação para as distâncias geodésicas subjacentes. As principais contribuições deste artigo são duplas: 1) após construir uma representação gráfica e substituir as distâncias euclidianas pelas distâncias geodésicas, o k-means topológico é capaz de detectar clusters com diferentes formas, dependendo da topologia do grafo; e 2) o k-means topológico evita a falta de poder discriminatório da distância euclidiana em espaços de alta dimensão, produzindo clusters significativos mesmo quando o número de características m é muito maior que o número de amostras n (a maldição da dimensionalidade).

O restante do artigo está organizado da seguinte forma: a Seção 2 apresenta uma visão geral do algoritmo de agrupamento k-means. A Seção 3 discute o algoritmo de Dijkstra em detalhes, mostrando que ele sempre retorna as distâncias geodésicas em grafos cujos pesos de aresta são positivos. A Seção 4 descreve o algoritmo topológico k-means proposto, explicando como ele funciona e analisando sua complexidade computacional. A Seção 5 mostra os experimentos computacionais com k-means regulares, k-means topológicos, apresentando os resultados obtidos em termos de 9 metricas de qualidade de *cluster*. Finalmente, a Seção 6 apresenta nossas conclusões e comentários finais.

2 O algoritmo de agrupamento k-means

O algoritmo k-means é um método não supervisionado no sentido de que o aprendizado é completamente autônomo. Sua primeira versão foi proposta originalmente em 1957 por Stuart P. Loyd no contexto da modulação por código de pulsos, mas só publicada décadas depois, em 1982 [14]. A ideia básica do k-means consiste em agrupar pontos de acordo com uma medida de similaridade, sendo os resultados obtidos (clusters) fortemente dependentes da medida de similaridade escolhida. Em sua versão regular, k-means adota a distância euclidiana, o que torna o algoritmo "cego" para clusters não esféricos.

A formulação do problema é a seguinte: dada uma matriz de dados $X^T = \{\vec{x}_1, \vec{x}_2, ..., \vec{x}_n\}$, onde $\vec{x}_i \in R^d$, o objetivo é particionar X em $k < n$ grupos ou clusters para minimizar o espalhamento intra-cluster, ou seja, encontrar a partição ótima $S^* = \{s_1^*, s_2^*, ..., s_k^*\}$ que satisfaz o seguinte critério, também conhecido como soma dos quadrados dentro do cluster (WCSS) [15]:

$$S^* = \arg\min_S \sum_{i=1}^{k} \sum_{\vec{x} \in s_i} \|\vec{x} - \vec{\mu}_i\|^2 \tag{1}$$

onde k denota o número de partições e $\vec{\mu}_i$ é o centróide do conjunto de partições s_i. Observe que, intuitivamente, a função objetivo determina que devemos fazer com que o centróide de cada partição esteja o mais próximo possível dos pontos de dados que definem essa partição.

Foi demonstrado que encontrar a solução ótima para este problema é NP-Difícil para qualquer número de dimensões d (até mesmo duas dimensões). A heurística adotada por k-means para simplificar o problema e consequentemente levar a um algoritmo de tempo polinomial, consiste em fixar o valor de k. Portanto, o parâmetro k (número de clusters) deve ser escolhido antes da execução do algoritmo, o que não se sabe são algumas situações. A seguir, apresentamos dois resultados principais que nos ajudam a entender melhor o algoritmo k-means.

Theorem 1. *Dado um cluster não vazio s_k, seu centróide ou média é a única escolha de centro que minimiza o espalhamento interno*

$$WS(\vec{c}^{(k)}) = \sum_{\vec{x} \in s_k} \|\vec{x} - \vec{c}^{(k)}\|^2 \tag{2}$$

Primeiro, observe que:

$$WS(\vec{c}^{(k)}) = \sum_{\vec{x} \in s_k} \|\vec{x} - \vec{c}^{(k)}\|^2 = \sum_{\vec{x} \in s_k} (\vec{x} - \vec{c}^{(k)})^T (\vec{x} - \vec{c}^{(k)}) \tag{3}$$

Aplicando a distributiva, temos:

$$WS(\vec{c}^{(k)}) = \sum_{\vec{x} \in s_k} \left(\vec{x}^T \vec{x} - 2\vec{x}^T \vec{c}^{(k)} + {\vec{c}^{(k)}}^T \vec{c}^{(k)} \right) \tag{4}$$

A condição necessária para a minimização é:

$$\frac{\partial}{\partial \vec{c}^{(k)}} WS(\vec{c}^{(k)}) = 0 \tag{5}$$

o que leva a:

$$2 \sum_{\vec{x} \in s_k} \vec{x} - 2 \sum_{\vec{x} \in s_k} \vec{c}^{(k)} = 0 \tag{6}$$

cuja solução é:

$$\vec{c}^{(k)} = \frac{1}{n_k} \sum_{\vec{x} \in s_k} \vec{x} \tag{7}$$

onde n_k é o número de amostras no k-ésimo cluster. O seguinte resultado mostra que a regra do centróide mais próximo, ou seja, atribuir cada ponto de dados $\vec{x}$ ao cluster cujo centro é o mais próximo, produz uma solução ótima para o problema de agrupamento k-means.

Theorem 2. *Dado um conjunto X de pontos de dados e uma sequência de k centróides $\vec{c}^{(1)}, \vec{c}^{(2)}, ..., \vec{c}^{(k)}$ uma partição em clusters minimiza a função objetivo.*

$$\sum_{i=1}^{k} \sum_{\vec{x} \in s_i} \|\vec{x} - \vec{\mu}_i\|^2 \tag{8}$$

se o algoritmo atribui cada amostra $\vec{x}$ ao cluster s_i com o centróide mais próximo.

A prova é trivial, pois cada amostra $\vec{x}$ contribui exatamente uma única vez para a soma definida pela função objetivo anterior e ao optar por atribuir $\vec{x}$ ao cluster cujo centróide é o mais próximo, minimizamos claramente a contribuição para a função objetivo. O algoritmo 1 ilustra o pseudocódigo do método de agrupamento k-means.

Algorithm 1 algoritmo de agrupamento k-means

```
// Parâmetros:
// X: a matriz de dados n × d (cada linha é uma amostra)
// n: o número de amostras
// k: o número de clusters
function K-MEANS(X, n, k)
    s⃗_1, s⃗_2, .., s⃗_k ← select_random_seeds(X, k)   ▷ Selecione k centroides aleatórios
    for i ← 1; i ≤ k; i++ do
        μ⃗_i ← s⃗_i
    end for
    while não converge do
        for i ← 1; i ≤ k; i++ do
            ω_i ← {}                                  ▷ Comece com partições k vazias
        end for
        for i ← 1; i ≤ n; i++ do
            j ← arg min_l ‖μ⃗_l − x⃗_i‖                ▷ Encontre o centroide mais proximo
            ω_j ← ω_j ∪ {x⃗_i}                         ▷ Atribuir amostra ao cluster
        end for
        for i ← 1; i ≤ k; i++ do
            μ⃗_i ← (1/|ω_i|) Σ_{x⃗∈ω_i} x⃗              ▷ Recalcule o valor dos centroides
        end for
    end while
    return ω_1, ω_2, ..., ω_k
end function
```

A convergência do algoritmo k-means é geralmente medida de três maneiras diferentes: 1) os centróides dos clusters recém-formados não se movem ou o seu movimento é inferior a um limite; 2) os pontos dos dados permanecem nos mesmos clusters; e 3) um número máximo de iterações é atingido.

2.1 Análise de complexidade

Basicamente, a complexidade computacional do algoritmo k-means depende essencialmente dos seguintes fatores: o número de amostras n, o número de centros k, o número de características d e o número de iterações para convergência t (que muitas vezes é desconhecido).

Primeiro, observe que a função select_random_seeds(X, k) e o *loop for* inicial (fora do *while*) são realizados em $O(k)$. A parte crítica é executar o *loop while*, que analisaremos a seguir. Para obter o centróide mais próximo, observe que devemos compará-lo com k centróides. Sabendo que para isso serão necessárias k distâncias euclidianas e que, para calcular cada distância, temos um custo de $O(d)$. Assim, para decidir qual centróide está mais próximo de uma amostra, o custo computacional é $O(kd)$. Como o processo é repetido para todos os pontos de dados n, temos um custo de $O(nkd)$. Finalmente, como este processo é repetido para um número desconhecido t de iterações, o custo total do algoritmo k-means é $O(tnkd)$, que é, no entanto, linear no número de amostras, levando a um algoritmo bastante eficiente.

3 Distâncias geodésicas em grafos

O problema de encontrar distâncias geodésicas em grafos é resolvido pelo cálculo dos caminhos mais curtos em grafos ponderados.

Definition 1. *Dado um grafo ponderado $G = (V, E, w)$, onde V é o conjunto de vértices, E é o conjunto de arestas e $w : E \to R^+$ é uma função de ponderação para as arestas, o peso de um caminho $P = v_0v_1v_2...v_n$ é dado por:*

$$w(P) = \sum_{i=1}^{n} w(v_{i-1}, v_i) \tag{9}$$

O caminho ótimo P^* de v_0 para v_n é um minimizador para $w(P)$, ou seja:

$$P^* = \arg\min_{P} w(P) \tag{10}$$

desde que exista um caminho entre v_0 e v_n, caso contrário o custo do caminho é infinito. A seguir apresentamos uma boa propriedade de caminhos ótimos.

Theorem 3. *Dado $G = (V, E, w)$ e $P = v_0v_1v_2...v_n$ o caminho ideal de v_0 para v_n. Para $0 \leq i < j \leq n$, seja $P' = v_iv_{i+1}...v_j$ um subcaminho de P. Então, P' é o caminho ótimo de v_i para v_j.*

Em outras palavras, qualquer subcaminho de um caminho ótimo P também é ótimo. Este resultado mostra que a programação dinâmica pode ser aplicada na solução de problemas de caminho mínimo. O algoritmo de Dijkstra combina programação dinâmica e uma estratégia gananciosa para resolver problemas de caminho mais curto de forma computacionalmente eficiente. Primeiramente, definimos as principais variáveis utilizadas pelo algoritmo de Dijkstra.

- $\lambda(v)$: comprimento do caminho mais curto da raiz s ao vértice v.
- $\pi(v)$: predecessor do vértice v na árvore de caminhos mais curtos.
- Q: fila de prioridade para armazenar os vértices (quanto menor $\lambda(v)$, maior a prioridade).

A fila de prioridade Q possui três primitivas básicas:

- Insert(Q, v): insere um vértice v no final de Q.
- ExtractMin(Q): remove de Q o vértice que possui o menor $\lambda(v)$.
- DecreaseKey(Q, v, $\lambda(v)$): atualiza a prioridade do vértice v em Q.

Diante do exposto, o método de Dijkstra para construção de caminhos mais curtos em grafos ponderados é apresentado no Algoritmo 2 [16].

Algorithm 2 Algoritmo de Dijkstra

```
// Parâmetros:
// G: o grafo de entrada com n vértices e m arestas
// w: o peso das arestas
// s: a raiz da árvore de caminho ideal
function DIJKSTRA(G, w, s)
    for each v ∈ V do                                   ▷ Inicializa λ(v) e π(v)
        λ(v) ← ∞
        π(v) ← nil
    end for
    λ(s) ← 0                                            ▷ Raiz começa com λ(s) = 0
    S ← ∅
    Q ← ∅
    for each v ∈ V do                                   ▷ Insere todas as vertices em Q
        Insert(Q, v)
    end for
    while Q ≠ ∅ do                                      ▷ Loop principal
        u ← ExtractMin(Q)                               ▷ Extrai o vértice de maior prioridade
        S ← S ∪ {u}
        for each v ∈ N(u) do                            ▷ Processe cada vizinho de u
            λ(v) ← min{λ(v), λ(v) + w(u, v)}            ▷ Vale a pena chegar a v a partir de u?
            if λ(v) was updated then                    ▷ Encontre uma entrada melhor de s para u
                π(v) ← u                                ▷ Atualiza o antecessor de v
                Decrease_Key(Q, v, λ(v))                ▷ Atualize as prioridades
            end if
        end for
    end while
end function
```

Em resumo, o algoritmo funciona da seguinte forma: inicialmente, todos os vértices são inicializados com $\lambda(v) = \infty$, pois neste ponto não sabemos se haverá um caminho de s para todos os outros vértice ou não. Depois disso, todos os vértices são inseridos

na fila de prioridade e a raiz s é a única que tem sua prioridade definida como zero (isso faz com que a raiz seja o primeiro vértice a sair de Q). A cada iteração, o vértice u de prioridade mais alta (menor $\lambda(v)$) u é removido de Q e inserido em S. Uma propriedade importante é que, quando um vértice v é inserido em S, seu $\lambda(v)$ não muda mais, pois neste ponto $\lambda(v) = d(s, v)$. Depois de remover u de Q, olhamos para cada vizinho v de u e verificamos se é uma boa ideia passar por u para chegar a v. Esta operação é conhecida como relaxação da aresta (u, v). Se $\lambda(v)$ for reduzido (u é uma boa rota para v), atualizamos o predepredecessor de v e a prioridade v em Q. Este processo é repetido até que não haja mais vértices em Q.

3.1 Análise de complexidade

A implementação do algoritmo de Djikstra pode usar estruturas de dados estáticas (matriz de adjacência para G e um array simples para Q) ou dinâmicas (lista de adjacências para G e um heap binário para Q). A seguir, discutimos a complexidade computacional do algoritmo ao utilizar estruturas dinâmicas devido ao melhor gerenciamento de memória.

Primeiramente observe que de acordo com o algoritmo, temos os seguintes fatos:

- A inicialização de $\lambda(v)$ e $\pi(v)$ para cada vértice em G é $O(n)$.
- A inserção dos vértices em Q pode ser realizada em $O(n)$ na pilha binária, pois inicialmente todos os vértices possuem a mesma prioridade.
- O loop WHILE principal é executado exatamente n vezes, uma para cada vértice.
- A primitiva ExtractMin(Q) é $O(log\ n)$, pois é proporcional a uma busca em uma árvore binária.
- O número de iterações do loop FOR depende essencialmente do grau do vértice u, ou seja, $d(u)$. Sem perda de generalidade, suponha que os vértices sejam removidos de Q na seguinte ordem: $v_1, v_2, ..., v_n$. Portanto, o número total de iterações do loop FOR é $d(v_1) + d(v_2) + ... + d(v_n)$. Mas, de acordo com o Handshaking Lema da teoria dos grafos, a soma dos graus dos n vértices de qualquer grafo é igual a duas vezes o número de arestas m, ou seja:

$$\sum_{i=1}^{n} d(v_i) = 2m \tag{11}$$

- A atualização de $\lambda(v)$ é $O(1)$ (tempo constante).
- A primitiva DescreaseKey é $O(log\ n)$, pois Q é um heap binário e temos que procurar por v

Diante do exposto, a função $T(n)$ que mede a complexidade computacional do algoritmo de Djikstra é dada por:

$$\begin{aligned} T(n) &= O(n) + O(n) + (O(1) + O(log\ n)) * \sum_{i=1}^{n} d(v_i) \\ &= O(n) + O(1) * O(m) + O(m) * O(log\ n) \end{aligned} \tag{12}$$

$$= O(m\ log\ n)$$

Portanto, o algoritmo de Dijkstra é linear no número de arestas e logarítmico no número de vértices, o que é bastante eficiente para um grande número de vértices n. No algoritmo topológico k-means proposto, o algoritmo de Dijkstra é executado em grafos k-NN.

3.2 Correção do algoritmo de Dijkstra

Antes de prosseguirmos, é crucial mostrar que sempre que aplicamos o algoritmo de Dijkstra em um grafo ponderado $G = (V, E, w)$ com pesos positivos, partindo de um nó raiz s, ele sempre constrói um caminho ótimo árvore com a raiz em s. Além disso, os comprimentos dos caminhos ótimos são a distância geodésica de s a todos os vértices restantes.

Theorem 4. *Given $G = (V, E, w)$, with $w : E \to R^+$, O algoritmo de Dijkstra termina com$\lambda(v) = d(s, v), \forall v \in V$, quando $d(s, v)$ é a distância geodésica de s a v.*

As etapas a seguir mostram uma prova por contradição.

1. Observe que ao longo de toda a execução do algoritmo de Dijkstra $\lambda(v) \geq d(s, v), \forall v \in V$.
2. Suponha que u esteja saindo de Q e seja o primeiro vértice a ter $\lambda(u) \neq d(s, u)$ quando isso acontece (quando entra em S).
3. Então, $u \neq s$, porque caso contrário teríamos $\lambda(s) = 0 = d(s, s)$ (u não pode ser a raiz da árvore).
4. Portanto, existe um caminho P_{su}, porque caso contrário $\lambda(u) = \infty = d(s, u)$. Logo, existe um caminho mais curto P^*_{su}.
5. Logo antes do vértice u ser removido de Q, o caminho tem $s \in S$ e $u \in V - S$.
6. Seja y o primeiro vértice em P^*_{su} tal que $y \in V - S$ e seja $x \in S$ seu predecessor.
7. Como $x \in S$, $\lambda(x) = d(s, x)$. Além disso, no momento em que x foi adicionado em S, a aresta (x, y) estava relaxada, ou seja:

$$\lambda(y) = \lambda(x) + w(x, y) = d(s, x) + w(x, y) = d(s, y) \tag{13}$$

8. Mas y precede u em P^*_{su} e como os pesos das arestas são positivos, tem $d(s, y) \leq d(s, u)$. Então, combinando as afirmações (7), (8) e (1), temos:

$$\lambda(y) = d(s, y) \leq d(s, u) \leq \lambda(u) \tag{14}$$

9. Mas, como u e y pertencem a $V - S$, quando u é escolhido para deixar Q, devemos ter $\lambda(u) \leq \lambda(y)$.
10. Como $\lambda(u) \leq \lambda(y)$ e $\lambda(y) \leq \lambda(u)$, segue que $\lambda(u) = \lambda(y)$, o que implica:

$$\lambda(y) = d(s, y) = d(s, u) = \lambda(u) \tag{15}$$

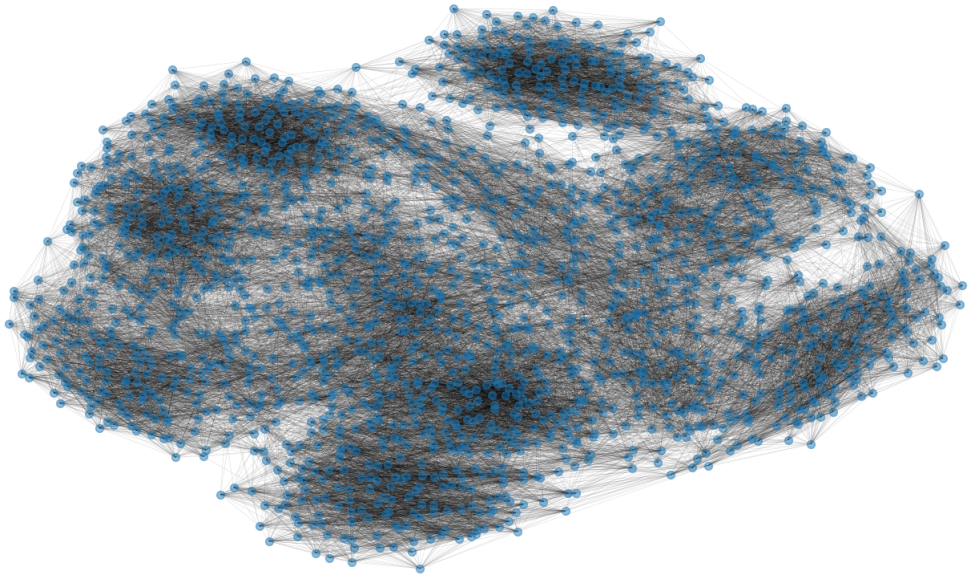

Fig. 1 grafo k-NN do conjunto de dados de dígitos para $nn = 42$ vizinhos.

produzindo uma contradição à nossa hipótese inicial. Portanto, $u \in V$ tal que $\lambda(u) \neq d(s,u)$ quando sai de Q.

4 O algoritmo k-means topológico

O método proposto de k-means topológico é um algoritmo baseado em grafos para agrupamento que utiliza um grafo k-NN para aproximar a variedade oculta dos dados no espaço de entrada. A hipótese múltipla afirma que, muitas vezes, muitos conjuntos de dados de alta dimensão observados em problemas do mundo real residem em uma estrutura geométrica não linear latente de baixa dimensão dentro do espaço ambiente [17, 18]. A figura 1 mostra uma ilustração do grafo k-NN para o conjunto de dados de dígitos, com $n = 1797$ amostras, $d = 64$ características e composto por $c = 10$ classes para $nn = \sqrt{n} = 42$ vizinhos.

A primeira questão que surge no k-médias topológicas é: como construir um grafo a partir de um conjunto de dados multivariado para criar uma aproximação discreta da variedade oculta dos dados? No aprendizado de máquina baseado em grafos, dois métodos populares são particularmente interessantes devido à eficiência computacional [19, 20]:

1. Grafo k-NN: para cada ponto de dados $\vec{x}_i, i = 1, 2, ..., n$, devemos calcular a distância euclidiana para todas as outras amostras $\vec{x}_j, j \neq i, j = 1, 2, ..., n$. Então, temos que selecionar apenas as k amostras com menor distância (vizinhos mais próximos) e criar uma aresta entre elas.
2. Grapho ϵ-vizinhos: devemos definir um raio ϵ e para cada amostra $\vec{x}_i, i = 1, 2, ..., n$ e todas as outras amostras $\vec{x}_j, j \neq i, j = 1, 2, ..., n$, calcule a distância euclidiana

entre eles. Para cada ponto dentro da bola de raio ϵ centrado em $\vec{x}_i$, uma aresta deve ser criada.

Uma segunda questão que surge é: quão bem o comprimento dos caminhos mais curtos em grafos k-NN pode aproximar as verdadeiras distâncias geodésicas subjacentes na variedade oculta dos dados? O Teorema da Convergência Assintótica mostra que, sob certas condições de regularidade, o comprimento dos caminhos mais curtos em grafos k-NN $d_G(\vec{x}_i, \vec{x}_j)$ converge para as distâncias geodésicas $d_M(\vec{x}_i, \vec{x}_j)$ [21]. Em resumo, os autores mostram que as duas métricas de distância, $d_G(\vec{x}_i, \vec{x}_j)$ e $d_M(\vec{x}_i, \vec{x}_j)$ aproximam uns aos outros arbitrariamente próximos, já que a densidade dos pontos de dados tende ao infinito. Em resumo, os autores mostram que as duas métricas de distância, $d_G(\vec{x}_i, \vec{x}_j)$ e $d_M(\vec{x}_i, \vec{x}_j)$ aproximam arbitrariamente uma da outra à medida que a densidade dos pontos de dados tende ao infinito.

Theorem 5 (Teorema da Convergência Assíptica)**.** *Dado $\lambda_1, \lambda_2, \mu > 0$, mas tão pequenos quanto desejado, então, para uma densidade suficientemente grande de pontos, a seguinte desigualdade é válida:*

$$1 - \lambda_1 \leq \frac{d_G(\vec{x}_i, \vec{x}_j)}{d_M(\vec{x}_i, \vec{x}_j)} \leq 1 + \lambda_2 \tag{16}$$

com probabilidade $1 - \mu$, onde $d_G(\vec{x}_i, \vec{x}_j)$ é a distância recuperada (comprimento do caminho mais curto) e $d_M(\vec{x}_i, \vec{x}_j)$ é a verdadeira distância geodésica na variedade oculta dos dados.

Os detalhes matemáticos da prova podem ser encontrados no trabalho de Bernstein e colegas [22]. A seguir, apresentamos o pseudocódigo para as k-médias topológicas propostas no Algoritmo 3.

Algorithm 3 Algoritmo k-means Topologico

// Parâmetros:
// X: a matriz de dados $n \times d$ (cada linha é uma amostra)
// n: o número de amostras
// k: o número de clusters
// nn: o número de vizinhos no grafo k-NN
function TOP-K-MEANS(X, n, k, nn)
  $\vec{s}_1, \vec{s}_2, .., \vec{s}_k \leftarrow select\_random\_seeds(X, k)$ ▷ Seleciona k centros aleatório
  $distances \leftarrow zeros(k, n)$ ▷ Vetor para armazenar as distancias geodésicas
  for $i \leftarrow 1; i \leq k; i++$ **do**
    $\vec{\mu}_i \leftarrow \vec{s}_i$
  end for
  while não converge **do**
    $G \leftarrow kNN\_Graph(X, nn)$ ▷ Construir o grafo k-NN
    for $i \leftarrow 1; i \leq k; i++$ **do**
      $\omega_i \leftarrow \{\}$ ▷ Começa com k partições vazias
    end for
    for $j \leftarrow 1; i \leq k; j++$ **do**
      $distances[j] \leftarrow Dijkstra(G, \vec{\mu}_j)$ ▷ Distâncias geodésicas
    end for
    for $i \leftarrow 1; i \leq n; i++$ **do**
      $j \leftarrow distances[:, i].argmin()$ ▷ Seleciona o centroide mais próximo
      $\omega_j \leftarrow \omega_j \cup \{\vec{x}_i\}$ ▷ Atribui a amostra ao cluster
    end for
    for $i \leftarrow 1; i \leq k; i++$ **do**
      $\vec{\mu}_i \leftarrow \frac{1}{|\omega_i|} \sum_{\vec{x} \in \omega_i} \vec{x}$ ▷ Recalcula os centroides
    end for
  end while
  return $\omega_1, \omega_2, ..., \omega_k$
end function

Neste ponto, alguns comentários sobre o algoritmo devem ser feitos. Primeiro, observe que, em comparação com k-médias regulares, as k-médias topológicas requerem um parâmetro adicional nn, que é o número de vizinhos no grafo k-NN. Alternativamente, pode-se optar por parametrizar o algoritmo em termos de ϵ, que é o raio que define o tamanho da vizinhança (bola centrada em $\vec{x}_i$). Segundo, para construir corretamente o grafo k-NN, ao final de cada iteração, os novos centros devem ser incluídos na matriz de dados X, pois muitas vezes eles não pertencem à matriz de dados original. Finalmente, nesta versão do algoritmo, o centro do cluster ω_j é atualizado calculando a média amostral das amostras atribuídas a ele, mas é possível calcular uma média sobre as distâncias geodésicas, mas isso exigiria outro execução do algoritmo de Dijkstra dentro do loop WHILE. Portanto, para alcançar um equilíbrio entre desempenho e custo computacional, optamos por manter a média amostral.

As principais vantagens das k-médias topológicas propostas sobre as k-médias regulares podem ser resumidas em dois pontos principais:

- O K-means normal usa a distância euclidiana, o que significa que é 'cego' para clusters não esféricos e bastante sensível à presença de ruído e valores discrepantes nos dados. Com a incorporação de uma distância geodésica baseada em grafos, pretendemos aliviar essas limitações.
- Em espaços de alta dimensão, o poder de discriminação da distância euclidiana é pobre devido à geometria complexa dos hiperespaços. Consequentemente, o desempenho de k-means e outros algoritmos de agrupamento pode ser severamente degradado em conjuntos de dados para os quais o número de recursos n é muito maior que o número de amostras m.

4.1 Análise de complexidade

A complexidade computacional do algoritmo k-means topológico proposto depende das seguintes variáveis: o número de amostras n, o número de características d, o número de clusters k, o número de vizinhos no grafo k-NN nn, o número de arestas no grafo k-NN m e o número de iterações t, que muitas vezes é desconhecido.

Observe que a definição dos centros iniciais pode ser feita em $O(k)$. A definição da matriz de distâncias é realizada em $O(kn)$, e o loop FOR inicial também é $O(k)$. A análise do loop WHILE principal revela que:

- A construção do grafo k-NN pode ser feita em $O(nd\ log\ n)$ usando uma KD-Tree.
- O algoritmo de Dijkstra é executado k vezes por iteração, resultando em um custo total de $O(km\ log\ n)$.
- O loop FOR de atribuição de rótulo custou $O(kn)$.
- O cálculo dos novos centros custou $O(nd)$, como na partição $\omega_1 + \omega_2 + ... + \omega_k = n$.

Como o loop WHILE principal é executado um número desconhecido de iterações t, o custo total do k-médias topológico é:

$$\begin{aligned} T(n) &= O(kn) + O(tnd\ log\ n) + O(tkm\ log\ n) + O(tkn) + O(tnd) \\ &= O(tnd\ log\ n) + O(tkm\ log\ n) = O(t(nd + km) log\ n) \end{aligned} \tag{17}$$

Portanto, o K-means topológico depende linear e logaritmicamente ao número de amostras n e linearmente ao número de arestas do grafo k-NN m, que em termos de complexidade computacional é considerado um algoritmo bastante eficiente.

5 Experimentos e resultados

A fim de testar e avaliar o desempenho do método proposto em relação a k-means regular. Foram selecionados algumas métricas de avaliação de cluster (índices externos) para avaliar o desempenho dos algoritmos: Rand index [23, 24], mutual information score [25], (V-measure, completeness, homogeneity) [26], Fowlkes-Mallows [27], Silhouettes [28], Calinski-Harabasz [29] e Davies-Bouldin [30].

O primeiro conjunto de experimentos foi projetado para comparar o desempenho dos métodos em conjuntos de dados reais de alta dimensionalidade. Todos os conjuntos de dados selecionados estão disponíveis gratuitamente no repositório www.openml.org. A Tabela 1 descreve cada conjunto de dados com nomes, número de amostras n, número de características m e número de classes k. Note que a maioria dos conjuntos de dados contém dados de microarranjo do projeto GEMLeR. Este projeto fornece uma coleção de conjuntos de dados de expressão gênica que podem ser usados para avaliar algoritmos de aprendizado de máquina orientados à expressão gênica. Eles podem ser usados para estimar diferentes métricas de qualidade (por exemplo, acurácia, precisão, área sob a curva ROC, etc.) para algoritmos de classificação, seleção de características ou clustering. Cada amostra de expressão gênica no repositório GEMLeR vem de um grande repositório público expO (Expression Project For Oncology) do International Genomics Consortium [31].

Table 1 Número de amostras, recursos e classes dos conjuntos de dados openML selecionados para o primeiro conjunto de experimentos.

Numero	Conjunto de Dados	# Amostras	# Atributos	# Classes
1	micro-mass	360	1300	10
2	monks-problems-1	556	6	2
3	breast-tissue	106	9	6
4	GCM	190	16063	14
5	balance-scale	625	4	3
6	servo	167	4	2
7	AP_Prostate_Uterus	193	10935	2
8	AP_Colon_Kidney	546	10935	2
9	AP_Breast_Kidney	604	10935	2
10	AP_Breast_Ovary	542	10935	2
11	AP_Breast_Colon	630	10935	2
12	leukemia	72	7129	2
13	hepatitisC	283	54621	3
14	Ovarian	253	15154	2
15	SRBCT	83	2308	4
17	Colon	62	2000	2
17	climate-model-simulation-crashes	540	20	2
18	pasture	36	22	3
19	glass	214	9	6
20	kc1-binary	145	94	2
21	dermatology	366	34	6
22	backache	180	31	2
23	lsvt	126	310	2
24	thoracic-surgery	470	16	2
25	planning-relax	182	12	2
26	tr31.wc	927	10128	7
27	tr45.wc	690	8261	10

É possível notar que uma característica comum à maioria dos conjuntos de dados é que o número de atributos m é muito maior que o número de amostras n. Por exemplo, no conjunto de dados número 7, AP_Prostate_Uterus, o número de características é

mais de 56 vezes o número de amostras, uma situação que representa um difícil desafio para algoritmos de agrupamento.

Os resultados quantitativos para os algoritmos de agrupamento são apresentados na Tabela 2 referentes ao k-means topológico e na Tabela 3 referentes ao k-means regular.

Table 2 Metricas usando o algoritmo de kmeans topológico.

#	Nome	Complet.	F. Mallows	Homog.	V Meas.	Adj. Rand	NMI	Silh.	C. Harab.	D. Bouldin	Tempo
1	micro-mass	0.6	0.44	0.55	0.58	0.35	0.6	0.2	86.58	1.34	125.87
2	monks-problems-1	0.08	0.56	0.08	0.08	0.1	0.08	0.19	154.83	1.8	6.33
3	breast-tissue	0.18	0.24	0.17	0.18	0.08	0.18	0.22	23.45	1.37	0.72
4	GCM	0.447	0.3	0.41	0.43	0.23	0.43	0.19	54.24	1.28	594.42
5	balance-scale	0.12	0.48	0.13	0.12	0.14	0.12	0.12	102.36	1.95	1.12
6	servo	0.04	0.59	0.04	0.04	0.03	0.04	0.2	44.9	1.7	1.73
7	AP_Colon_Kidney	0.11	0.58	0.1	0.1	0.1	0.1	0.15	85.36	2.18	89.54
8	AP_Breast_Kidney	0.11	0.59	0.1	0.1	0.11	0.1	0.13	88.50	2.25	97.11
9	AP_Breast_Ovary	0.01	0.58	0.004	0.004	0.01	0.004	0.25	80.75	1.77	91.47
10	AP_Breast_Colon	0.01	0.54	0.01	0.01	0.012	0.01	0.18	134.51	1.9	94.03
11	AP_Prostate_Uterus	0.17	0.61	0.16	0.17	0.15	0.17	0.16	38.05	1.92	50.27
12	leukemia	0.05	0.58	0.03	0.04	0.03	0.04	0.17	12.26	1.77	2.32
13	hepatitisC	0.15	0.65	0.22	0.17	0.09	0.17	-0.003	13.53	2.94	166.31
14	Ovarian	0.007	0.54	0.007	0.007	0.004	0.007	0.17	56.41	1.88	56.41
15	SRBCT	0.14	0.34	0.11	0.12	0.04	0.12	0.10	10.49	1.69	3.46
16	Colon	0.025	0.56	0.02	0.02	0.001	0.02	0.151	9.33	1.92	2.24
17	climate...	0.06	0.65	0.15	0.09	0.01	0.09	0.54	1126.12	0.62	4.03
18	pasture	0.23	0.45	0.2	0.21	0.15	0.21	0.49	231.73	1.36	0.83
19	glass	0.22	0.33	0.25	0.24	0.13	0.23	0.22	95.99	1.14	2.63
20	kc1-binary	0.15	0.61	0.15	0.15	0.2	0.15	0.46	36.51	0.79	1.55
21	dermatology	0.15	0.24	0.15	0.15	0.06	0.15	0.29	516.92	1.11	18.14
22	backache	0.01	0.63	0.01	0.01	-0.006	0.01	0.41	184.8	0.82	2.36
23	lsvt	0.02	0.54	0.02	0.02	-0.01	0.02	0.63	227.9	0.53	2.91
24	thoracic-surgery	0.003	0.69	0.002	0.002	-0.007	0.002	0.57	430.68	0.71	4.03
25	planning-relax	0.002	0.58	0.002	0.002	0.003	0.002	0.15	32.72	1.93	0.88
26	tr31.wc	0.2	0.38	0.18	0.19	0.12	0.19	-0.01	565.44	2.45	1825.38
27	tr45.wc	0.26	0.25	0.21	0.23	0.08	0.23	-0.02	140.28	2.35	836.54
28	**Media**	0.13	0.5	0.13	0.13	0.08	0.13	0.23	169.81	1.61	151.213
29	**Desvio padrão**	0.14	0.14	0.13	0.13	0.09	0.13	0.17	242.22	0.6	384.91
30	**Máximo**	0.61	0.69	0.55	0.58	0.35	0.58	0.63	1126.12	2.94	1825.38

Table 3 Metricas usando o algoritmo de kmeans padrão.

#	Nome	Complet.	F. Mallows	Homog.	V Meas.	Adj. Rand	NMI	Silh.	C. Harab.	D. Bouldin	Tempo
1	micro-mass	0.7	0.48	0.60	0.65	0.38	0.65	0.24	88.81	1.27	4.2
2	monks-problems-1	0.08	0.55	0.08	0.08	0.10	0.08	0.23	202.32	1.63	3.17
3	breast-tissue	0.20	0.25	0.2	0.20	0.1	0.2	0.14	13.31	1.75	0.28
4	GCM	0.49	0.29	0.45	0.47	0.22	0.47	0.11	26.75	1.82	20.29
5	balance-scale	0.1	0.46	0.13	0.11	0.13	0.11	0.17	136.76	1.69	3.23
6	servo	0.14	0.59	0.16	0.15	-0.06	0.15	0.30	69.35	1.33	0.26
7	AP_Colon_Kidney	0.006	0.50	0.006	0.006	0.005	0.006	0.16	105.26	2.19	18.71
8	AP_Breast_Kidney	0.005	0.51	0.005	0.005	0.009	0.005	0.15	108.48	2.25	20.9
9	AP_Breast_Ovary	0.11	0.69	0.003	0.005	0.005	0.005	0.79	577.22	0.43	16.61
10	AP_Breast_Colon	0.01	0.51	0.01	0.01	0.01	0.01	0.18	151.04	1.96	18.53
11	AP_Prostate_Uterus	0.03	0.53	0.03	0.03	0.03	0.03	0.16	45.03	1.98	7.99
12	leukemia	0.008	0.54	0.007	0.007	0.01	0.007	0.12	9.32	2.56	2.96
13	hepatitisC	0.15	0.62	0.31	0.2	0.14	0.2	0.06	16.46	3.37	80.17
14	Ovarian	0.007	0.52	0.007	0.007	0.009	0.007	0.19	69.71	1.82	23.51
15	SRBCT	0.21	0.35	0.19	0.2	0.08	0.2	0.09	7.05	2.4	2.03
16	Colon	0.01	0.52	0.01	0.01	-0.007	0.01	0.09	7.68	2.7	1.39
17	climate...	0.06	0.65	0.16	0.09	0.02	0.09	0.54	1118.47	0.64	3.18
18	pasture	0.42	0.57	0.42	0.42	0.37	0.42	0.5	104.87	0.65	0.24
19	glass	0.45	0.50	0.38	0.41	0.26	0.41	0.44	122.29	0.94	0.31
20	kc1-binary	0.14	0.69	0.04	0.06	0.04	0.06	0.86	181.8	0.59	0.88
21	dermatology	0.09	0.21	0.09	0.09	0.02	0.09	0.27	509.18	1.1	2.92
22	backache	0.01	0.63	0.01	0.01	-0.03	0.01	0.42	185.1	0.86	0.29
23	lsvt	0.02	0.61	0.017	0.02	-0.03	0.02	0.71	281.44	0.47	1.43
24	thoracic-surgery	0.005	0.78	0.002	0.002	-0.017	0.002	0.74	641.0	0.42	3.09
25	planning-relax	0.0	0.54	0.0	0.0	-0.004	0.0	0.16	38.16	2.04	0.29
26	tr31.wc	0.23	0.48	0.05	0.09	0.04	0.09	0.66	1740.9	1.20	39.31
27	tr45.wc	0.28	0.31	0.09	0.14	0.007	0.14	0.26	314.31	1.57	32
28	**Media**	0.15	0.51	0.13	0.13	0.07	0.13	0.32	254.52	1.54	11.41
29	**Desvio padrão**	0.18	0.13	0.16	0.17	0.11	0.17	0.24	389.02	0.77	17.53
30	**Máximo**	0.7	0.78	0.6	0.65	0.38	0.65	0.86	1,740.9	3.37	80.17

Observando os resultados, vemos que, para esses conjuntos de dados, algumas metricas do k-médias topológicas propostas superam as k-médias padrão. Em todos os experimentos, o parâmetro k (número de clusters) é igual ao número de classes e o número de vizinhos nn é definido como $\lfloor\sqrt{n}\rfloor$. Para gerar os resultados, calculamos o desempenho médio após 30 inicializações aleatórias de k-médias e k-médias topológicas. Para verificar se as diferenças são significativas, foi realizado um teste não paramétrico de Wilcoxon. De acordo com o teste, para um nível de significância $\alpha = 0,05$, há fortes evidências de que as k-médias topológicas propostas tiveram melhor desempenho do que as k-médias regulares em termos de índice Rand ($p < 0,001$), informação mútua normalizada ($p < 0,001$) e medida V ($p < 0,001$).

Alguns comentários sobre as k-médias topológicas devem ser abordados neste ponto. Um aspecto importante do método proposto é que devemos garantir que o o k-NN esteja conectado, de forma a permitir que os centros se movam livremente em torno de todo o conjunto de vértices. Caso contrário, se a inicialização aleatória escolher todos os centróides para estarem em um único componente conectado, o algoritmo nunca rotulará as amostras que pertencem a outros componentes conectados de G. Existem duas estratégias para garantir que G esteja conectado: 1) defina o número de

vizinhos $nn = \lfloor \sqrt{n} \rfloor$ e enquanto G não estiver conectado, incremente nn em um e construa um novo k-NN; e 2) começar com um completo, extrair a árvore geradora mínima (MST) e adicionar as arestas do MST ao grafo k-NN construído usando $nn = \lfloor \sqrt{n} \rfloor$. Alguns autores argumentam que em vez da raiz quadrada do número de amostras, uma boa estimativa para o número de vizinhos é o logaritmo de base dois do número de amostras [32, 33]. No entanto, em nossos experimentos, $nn = log_2\ n$ produziu grafos desconectados para vários conjuntos de dados de alta dimensão, então optamos por considerar a configuração de parâmetro padrão como $nn = \lfloor \sqrt{n} \rfloor$.

As principais desvantagens das k-médias topológicas propostas ainda estão relacionadas às limitações do k-means normal. Primeiro, o esquema de inicialização aleatória: o desempenho do algoritmo está diretamente relacionado à seleção aleatória dos centróides iniciais, algo que não está sob controle. Melhores estratégias de inicialização para k-means podem ser adaptadas para k-means topológicos, como k-means++ por exemplo [34]. Métodos baseados em propriedades topológicas do grafo k-NN também podem ser empregados para esse fim. Outra questão está relacionada à definição do número de clusters k. Em k-médias topológicas é possível evitar a dependência do parâmetro k estimando seu valor através de um algoritmo de detecção de comunidade no grafo k-NN [35].

6 Conclusões e observações finais

Algoritmos de clustering são ferramentas versáteis em aprendizado de máquina, contribuindo para vários aspectos de análise, exploração e compreensão de dados. Suas aplicações abrangem diferentes domínios, tornando-as indispensáveis para pesquisadores, cientistas de dados e analistas que buscam descobrir padrões e estruturas em dados complexos. Esses algoritmos desempenham um papel crucial no aprendizado de máquina, especialmente no aprendizado não supervisionado. Alguns exemplos de problemas onde o agrupamento desempenha um papel importante são: descoberta e reconhecimento de padrões, exploração e compreensão de dados, detecção de anomalias, processamento de imagens e sinais, redução de dimensionalidade, sistemas de recomendação, análise de documentos, bioinformática e na redução da carga de rotulagem.

No entanto, agrupar dados de alta dimensão ainda é uma tarefa desafiadora, especialmente em problemas de tamanho amostral pequeno. Parte desta complexidade vem das propriedades geométricas dos hiperespaços, ou seja, espaços com milhares de dimensões. Dados de alta dimensão representam desafios devido à maldição da dimensionalidade porque as métricas de distância tradicionais, como a distância euclidiana, podem tornar-se menos significativas, afetando negativamente o desempenho dos algoritmos de agrupamento.

Neste artigo, propusemos um algoritmo topológico k-means que utiliza caminho ótimo em grafos para aproximar as verdadeiras distâncias geodésicas na variedade de dados. Em resumo, a motivação para k-means topológicos foi melhorar a capacidade do k-means de lidar com dados de alta dimensão. No geral, as principais contribuições do método proposto estão diretamente relacionadas com a mitigação da cegueira do agrupamento não esférico de k-médias regulares. Nossos resultados mostraram que

k-means topológicos são capazes de produzir clusters melhores que k-means nomal. Apesar de ser um algoritmo viável e promissor, o k-means topológico não é perfeito e possui limitações. O mais destacado é que o desempenho depende de uma boa escolha inicial de centróides, que na maioria das vezes não está sob controle, além de um leve aumento no tempo computacional.

A fim de superar as limitações topológicas de k-means, trabalhos futuros podem incluir o desenvolvimento de melhores estratégias de inicialização, como a heurística kmeans++ e um método baseado em curvatura para amostragem de dados baseado no operador de forma. Outra melhoria futura diz respeito à determinação automática do número de clusters por meio da detecção de comunidades no grafo k-NN. Outras estratégias para construção de grafos podem ser adotadas como forma de tornar mais evidente a estrutura subjacente do cluster nos dados, facilitando o trabalho das k-médias topológicas. Finalmente, o uso de diferentes métricas para ponderar as arestas do grafo k-NN pode ser interessante para melhorar a capacidade topológica de k-médias de aprender diferentes formas de clusters.

References

[1] Ezugwu, A.E., Ikotun, A.M., Oyelade, O.O., Abualigah, L., Agushaka, J.O., Eke, C.I., Akinyelu, A.A.: A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. Engineering Applications of Artificial Intelligence **110**, 104743 (2022)

[2] Soheil, S.B., Müller, N., Plant, C., Böhm, C.: Clustering of mixed-type data considering concept hierarchies: problem specification and algorithm. International Journal of Data Science and Analytics **10**, 233–248 (2020)

[3] Benabdellah, A.C., Benghabrit, A., Bouhaddou, I.: A survey of clustering algorithms for an industrial context. Procedia Computer Science **148**, 291–302 (2019)

[4] Li, X., Liang, W., Zhang, X., Qing, S., Chang, P.-C.: A cluster validity evaluation method for dynamically determining the near-optimal number of clusters. Soft Comput. **24**(12), 9227–9241 (2020)

[5] Saxena, A., Prasad, M., Gupta, A., Bharill, N., Patel, O.P., Tiwari, A., Er, M.J., Ding, W., Lin, C.-T.: A review of clustering techniques and developments. Neurocomputing **267**, 664–681 (2017)

[6] Sinaga, K.P., Yang, M.-S.: Unsupervised k-means clustering algorithm. IEEE Access **8**, 80716–80727 (2020)

[7] Chong, B.: K-means clustering algorithm: a brief review. Academic Journal of Computing & Information Science **4**(5), 37–40 (2021)

[8] McInnes, L., Healy, J., Astels, S.: hdbscan: Hierarchical density based clustering. Journal of Open Source Software **2**(11), 205 (2017)

[9] Stewart , G., Al-Khassaweneh, M.: An implementation of the hdbscan* clustering algorithm. Applied Sciences **12**(5) (2022)

[10] Bushra, A.A., Yi, G.: Comparative analysis review of pioneering dbscan and successive density-based clustering algorithms. IEEE Access **9**, 87918–87935 (2021)

[11] Kriegel, H.-P., Kröger, P., Zimek, A.: Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. ACM Trans. Knowl. Discov. Data **3**(1) (2009)

[12] Houle, M.E., Kriegel, H., Kröger, P., Schubert, E., Zimek, A.: Can shared-neighbor distances defeat the curse of dimensionality? In: Proceedings of the 22nd International Conference on Scientific and Statistical Database Management (SSDBM), Heidelberg, Germany, pp. 482–500 (2010)

[13] Thrun, M.C., Ultsch, A.: Using Projection-Based Clustering to Find Distance- and Density-Based Clusters in High-Dimensional Data. Journal of Classification **38**(2), 280–312 (2021)

[14] Loyd, S.P.: Least squares quantization in pcm. IEEE Transactions on Information Theory **28**(2), 129–137 (1982)

[15] Ikotun, A.M., Ezugwu, A.E., Abualigah, L., Abuhaija, B., Heming, J.: K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. Information Sciences **622**, 178–210 (2023)

[16] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 4th edn. The MIT Press, New York (2022)

[17] Gorban, A.N., Tyukin, I.Y.: Blessing of dimensionality: mathematical foundations of the statistical physics of data. Philosophical Transactions of the Royal Society of London Series A **376**(2118), 20170237 (2018)

[18] Fefferman, C., Mitter, S., Narayanan, H.: Testing the manifold hypothesis. Journal of the American Mathematical Society **29**(4), 983–1049 (2016)

[19] Dong, W., Moses, C., Li, K.: Efficient k-nearest neighbor graph construction for generic similarity measures. In: Proceedings of the 20th International Conference on World Wide Web. WWW '11, pp. 577–586. Association for Computing Machinery, New York, NY, USA (2011)

[20] Abu-Aisheh, Z., Raveaux, R., Ramel, J.-Y.: Efficient k-nearest neighbors search in graph space. Pattern Recognition Letters **134**, 77–86 (2020)

[21] Silva, V., Tenenbaum, J.: Global versus local methods in nonlinear dimensionality reduction. In: Becker, S., Thrun, S., Obermayer, K. (eds.) Advances in Neural

Information Processing Systems, vol. 15. MIT Press, Vancouver, Canada (2002)

[22] Bernstein, M., Silva, V., Langford, J.C., Tenenbaum, J.B.: Graph approximations to geodesics on embedded manifolds. Preprint at: https://users.math.msu.edu/users/iwenmark/Teaching/MTH995/Papers/MMod_BSLT00.pdf (2000)

[23] Rand, W.M.: Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association **66**(336), 846–850 (1971)

[24] Hubert, L., Arabie, P.: Comparing partitions. Journal of Classification **2**(1), 193–218 (1985)

[25] Vinh, N.X., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: Is a correction for chance necessary? In: Proceedings of the 26th Annual International Conference on Machine Learning. ICML '09, pp. 1073–1080. Association for Computing Machinery, New York, NY, USA (2009)

[26] Rosenberg, A., Hirschberg, J.: V-measure: A conditional entropy-based external cluster evaluation measure. In: Eisner, J. (ed.) Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pp. 410–420. Association for Computational Linguistics, Prague, Czech Republic (2007)

[27] Fowlkes, E.B., Mallows, C.L.: A method for comparing two hierarchical clusterings. Journal of the American Statistical Association **78**(383), 553–569 (1983). Accessed 2024-12-05

[28] Rousseeuw, P.J.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics **20**, 53–65 (1987) https://doi.org/10.1016/0377-0427(87)90125-7

[29] Caliński, T., Harabasz, J.: A dendrite method for cluster analysis. Communications in Statistics **3**(1), 1–27 (1974) https://doi.org/10.1080/03610927408827101

[30] Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence **PAMI-1**(2), 224–227 (1979) https://doi.org/10.1109/TPAMI.1979.4766909

[31] Stiglic, G., Kokol, P.: Stability of ranked gene lists in large microarray analysis studies. Journal of Biomedicine and Biotechnology, 616358 (2010)

[32] Marchette, D.J.: Random Graphs for Statistical Pattern Recognition, 1st edn. Wiley, New York (2004)

[33] Maier, M., Hein, M., Luxburg, U.V.: Optimal construction of k-nearest-neighbor graphs for identifying noisy clusters. Theoretical Computer Science **410**(19), 1749–1764 (2009)

[34] Arthur, D., Vassilvitskii, S.: K-means++: The advantages of careful seeding. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms. SODA '07, pp. 1027–1035. Society for Industrial and Applied Mathematics, USA (2007)

[35] Fortunato, S.: Community detection in graphs. Physics Reports **486**(3), 75–174 (2010)