

# UNIDADE 4 - Algoritmos de clusterização

Marta Noronha

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



# SUMÁRIO

- 4.1 Conceitos de métodos baseados em Particionamento
- 4.2 Algoritmos k-Médias e k-Medoides
- 4.3 Conceitos de métodos baseados em Hierarquia
- 4.4 Algoritmos BIRCH e ROCK
- 4.5 Conceitos de métodos baseados em Densidade
- 4.6 Algoritmos DBSCAN e DENCLUE
- 4.7 Conceitos de métodos baseados em Modelos
- 4.8 Algoritmos Expectation-Maximization e SOM
- 4.9 Noções básicas sobre biclusterização e triclusterização

# Particionamento

- 4.1 Conceitos de métodos baseados em Particionamento
- 4.2 Algoritmos k-Médias e k-Medoides

# Conceitos de métodos baseados em Particionamento

Considere:

- $D$  é um conjunto de dados contendo  $n$  objetos (instâncias ou pontos de dados).
- $k$  é o número de clusters informado pelo usuário que deve ser descoberto em  $D$ .
- $C_j$  é uma partição (ou cluster) onde  $j = \{1, 2, \dots, k\}$ .
- Os  $n$  objetos de  $D$  devem ser assinalados a um cluster  $C_j$ .
- Em *soft clustering*, cada objeto de  $D$  deve ser assinalado a um único cluster  $C_j$ .
- Em *hard clustering*, cada objeto possui uma probabilidade de pertencer a cada cluster  $C_j$ .
- Um *critério (função) deve ser otimizado* para que objetos do cluster  $C_j$  sejam mais similares aos outros objetos em  $C_j$  e, ao mesmo tempo, qualquer objeto em  $C_j$  deve ser mais dissimilar a qualquer objeto em  $C_i$ , com  $i \neq j$ .

## Conceitos de métodos baseados em Particionamento

- Objetos são iterativamente movidos de um cluster para outro até melhorar o valor do critério (função) de agrupamento, sendo por isso chamados de métodos particionais.
  - Busca por partição de variância mínima.
  - Podem não convergir para o global ótimo, mas para o local ótimo por serem abordagens gulosas.
  - Problema  $NP-hard$ : Minimizar o erro quadrático de um cluster (ou variação intra-cluster) é o mesmo que maximizar a variação inter-clusters. Verificar todas as possibilidades é exaustivo. Por isso que a solução k-médias usa uma heurística gulosa.

## k-médias ( $k - means$ )

- Algoritmo **popular** na indústria e na academia.
- Fácil implementação.
- Computacionalmente eficiente em comparação a outros algoritmos de clusterização.
- Também conhecido como método **baseado em protótipo** porque considerar cada centroide (ou medoide) como um protótipo para guiar a clusterização.
- O **centroide** é considerado como um centro de gravidade do cluster.
- O **medóide** é um objeto considerado como representativo no cluster.
- A moda pode ser usada frente a dados categóricos.

## k-médias

- Clusters identificados pelo k-médias possuem o formato esférico, são compactos e bem separados entre si.
- Necessário informar alguns parâmetros para guiar o processo de busca, sendo obrigatório informar a quantidade de clusters,  $k$ , que devem ser buscados.
- A escolha incorreta de  $k$  resulta em clusters de baixa qualidade, onde há maior variância intra cluster e menor variância inter cluster quando comparados a soluções melhores (produzidas com valor de  $k$  adequado).
- Agrupamento baseado na similaridade dos objetos de um cluster em relação a todos os seus atributos.

## k-médias

- Uso de medidas de distância para verificar a similaridade.
- Distância é o oposto da similaridade.
- Algumas medidas de distância podem não convergir usando este algoritmo.
- Medida mais usada para verificar a distância é a euclidiana, porém essa medida é sensível a *outliers*.

$$dist\_euclidiana = \sqrt{\sum_{j=1}^m (X_j - Y_j)^2} = \|X - Y\|_2^2$$

onde  $m$  é a quantidade de atributos do conjunto de dados  $D$ .

## k-médias

- Método da silhueta ou cotovelo são usados para verificar qual a quantidade ideal de clusters que pode ser descoberto em  $D$ , a qual minimiza a função objetivo.
- Problema de otimização em uma abordagem iterativa para minimizar a **soma intracluster dos erros quadráticos (SSE)**.

$$SSE = \sum_{i=1}^n \sum_{j=1}^k W^{(i,j)} \|X^{(i)} - \mu^{(j)}\|_2^2$$

onde  $W^{(i,j)} = 1$  se  $X^{(i)} \in C_j$ . Caso contrário,  $W^{(i,j)} = 0$ . Na equação,  $\mu^{(j)}$  é o centroide do cluster  $C_j$ .

Particionamento

oooooooooooo●oooooooooooooooooooo

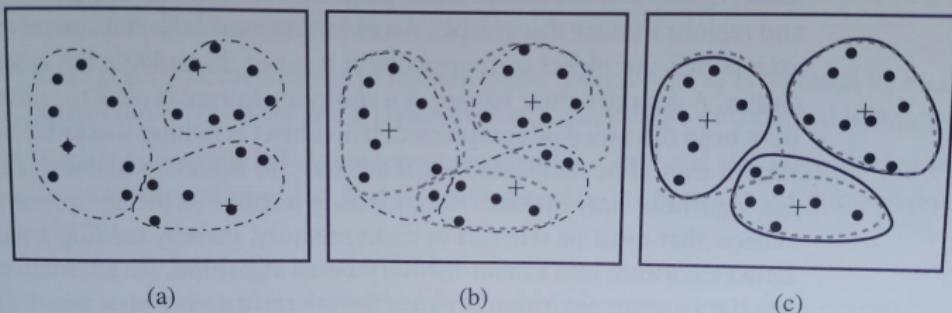
Hierárquico

oooooooooooooooooooooooooooo

Densidade

oooooooooooooooooooooooooooo

## k-médias



**Figure 7.3** Clustering of a set of objects based on the  $k$ -means method. (The mean of each cluster is marked by a “+”.)

Fonte: Faceli et al. 2021

## k-médias

- Variações do k-médias buscam solucionar deficiências do algoritmo original. Como exemplo, k-médias busca somente por clusters esféricos. Uma modificação usa a distância de Mahalanobis para buscar por clusters elipsoidais.
- Pode-se usar k-médias e k-moda, baseado na frequência da ocorrência dos valores possíveis de um atributo, para localizar clusters em dados que possuem atributos numéricos e categóricos.
- Complexidade:  $\mathcal{O}(nkt)$ , onde  $n$  é o total de objetos em  $D$ ,  $k$  é a quantidade de clusters em  $D$  e  $t$  é o total de iterações máxima a serem realizadas.
- Comumente  $k \ll n$  e  $t \ll n$ , portanto em alguns materiais a complexidade é dada como  $\mathcal{O}(n)$ .

# k-médias

## Desvantagens

- O número de clusters,  $k$ , deve ser informado pelo usuário podendo prejudicar o desempenho do algoritmo.
- Não localiza clusters não convexos ou de tamanhos arbitrários.
- Uso somente em dados numéricos devido ao uso da média para cálculo da distância.
- Sensível a ruídos e *outliers*.

# k-médias

## Passos do algoritmo

- ① Escolher aleatoriamente  $k$  centroides dentre os objetos de  $D$  como centros iniciais dos clusters.
- ② Assinalar cada objeto ao centroide,  $\mu^j$  com  $j = \{1, 2, \dots, k\}$ , mais próximo.
- ③ Mover o centroide de cada cluster ao centro dos objetos que foram assinalados respectivamente em cada cluster.
- ④ Repetir os passos 2 e 3 até que o cluster não tenha mais alteração (o centroide não mude) ou até atingir a tolerância definida pelo usuário ou até que o máximo de iterações seja atingido.

OBS: O processo de troca de centroides é chamado de relocação iterativa.

# k-médias

Iteração →

Centróides →

Elementos {

I1	x	y
C1	0	5
C2	10	7
E	x	y
1	0	5
2	16	18
3	12	27
4	20	30
5	10	7
6	13	1
7	2	18
8	25	9
9	10	3
10	1	2

## k-médias



## k-médias

Iteração →

Centróides →

Elementos →

I1	x	y		
C1	0	5	D.C1	D.C2
C2	10	7		
E	x	y		
1	0	5	0	10
2	16	18	21	13
3	12	27	25	20
4	20	30	32	25
5	10	7	10	0
6	13	1	14	7
7	2	18	13	14
8	25	9	25	15
9	10	3	10	4
10	1	2	3	10

Distância do Centróide 1

Distância do Centróide 2

Elementos mais próximos do Centróide 2

Soma das distâncias C1 = 16

Soma das distâncias C2 = 84

Total: 100

# k-médias

Baseado na média dos elementos de cada cluster, calcula-se os novos centróides

I1	x	y		
C1	0	5		
C2	10	7		
E	x	y	D-C1	D-C2
1	0	5	0	10
2	16	18	21	13
3	12	27	25	20
4	20	30	32	25
5	10	7	10	0
6	13	1	14	7
7	2	18	13	14
8	25	9	25	15
9	10	3	10	4
10	1	2	3	10

I2	x	y		
C1	1	8		
C2	15	14		
E	x	y	D-C1	D-C2
1	0	5	3	17
2	16	18	18	5
3	12	27	22	14
4	20	30	29	17
5	10	7	9	8
6	13	1	14	13
7	2	18	10	14
8	25	9	24	11
9	10	3	10	12
10	1	2	6	18

Com os centróides novos, o elemento 9 ficou mais perto do Centróide 1, trocando de grupo

# k-médias

Baseado na média dos elementos de cada cluster, calcula-se os novos centróides

I1	x	y		
C1	0	5		
C2	10	7		
E	x	y	D-C1	D-C2
1	0	5	0	10
2	16	18	21	13
3	12	27	25	20
4	20	30	32	25
5	10	7	10	0
6	13	1	14	7
7	2	18	13	14
8	25	9	25	15
9	10	3	10	4
10	1	2	3	10

I2	x	y		
C1	1	8		
C2	15	14		
E	x	y	D-C1	D-C2
1	0	5	3	17
2	16	18	18	5
3	12	27	22	14
4	20	30	29	17
5	10	7	9	8
6	13	1	14	13
7	2	18	10	14
8	25	9	24	11
9	10	3	10	12
10	1	2	6	18

Com os centróides novos, o elemento 9 ficou mais perto do Centróide 1, trocando de grupo

Soma das distâncias  
C1 = 29

Soma das distâncias  
C2 = 68

Total: 97

Particionamento

oooooooooooooooooooo●oooooooooooo

Hierárquico

oooooooooooooooooooooooooooooooooooo

Densidade

oooooooooooooooooooooooooooooooooooo

# k-médias

I2	x	y		
C1	1	8		
C2	15	14		
E	x	y	D-C1	D-C2
1	0	5	3	17
2	16	18	18	5
3	12	27	22	14
4	20	30	29	17
5	10	7	9	8
6	13	1	14	13
7	2	18	10	14
8	25	9	24	11
9	10	3	10	12
10	1	2	6	18

I3	x	y		
C1	3	7		
C2	16	15		
E	x	y	D-C1	D-C2
1	0	5	4	19
2	16	18	17	3
3	12	27	22	12
4	20	30	28	15
5	10	7	7	10
6	13	1	11	15
7	2	18	11	14
8	25	9	22	11
9	10	3	8	14
10	1	2	5	20

Soma das distâncias  
C1 = 46

Soma das distâncias  
C2 = 41

Total: 87

Particionamento

oooooooooooooooooooo●oooooooooooo

Hierárquico

oooooooooooooooooooooooooooooooooooo

Densidade

oooooooooooooooooooooooooooooooooooo

# k-médias

Nenhum elemento trocou de posição,  
então o algoritmo termina.

I3	x	y		
C1	3	7		
C2	16	15		
E	x	y	D-C1	D-C2
1	0	5	4	19
2	16	18	17	3
3	12	27	22	12
4	20	30	28	15
5	10	7	7	10
6	13	1	11	15
7	2	18	11	14
8	25	9	22	11
9	10	3	8	14
10	1	2	5	20

I4	x	y		
C1	6	6		
C2	18	21		
E	x	y	D-C1	D-C2
1	0	5	6	24
2	16	18	16	4
3	12	27	22	9
4	20	30	28	9
5	10	7	4	16
6	13	1	9	21
7	2	18	13	17
8	25	9	19	14
9	10	3	5	20
10	1	2	6	26

Soma das distâncias  
C1 = 43

Soma das distâncias  
C2 = 36

Total: 79

# k-médias - python

```
from sklearn.cluster import KMeans
import numpy as np
X = np.array([[0,5],[16,18],[12,27],[20,30],[10,7],[13,1],[2,18],[25,9],[10,3],[1,2]])
```

```
kmeans = KMeans(n_clusters=2, random_state=0, max_iter = 4, n_init=3).fit(X)
```

```
kmeans.labels_
```

```
array([0, 1, 1, 1, 0, 0, 0, 1, 0, 0])
```

```
kmeans.predict([[0, 0], [16, 17]])
```

```
array([0, 1])
```

```
kmeans.cluster_centers_
```

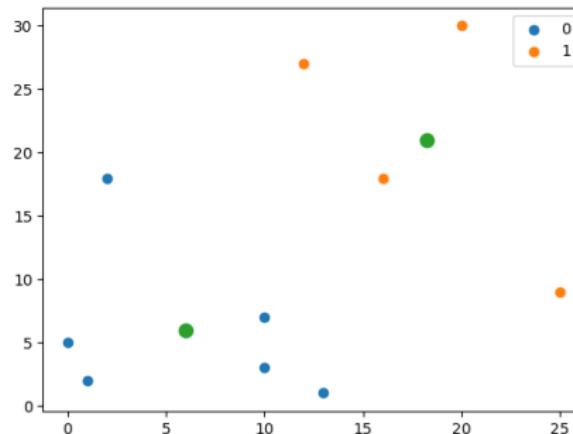
```
array([[ 6. ,  6. ],
       [18.25, 21. ]])
```

# k-médias - python

```
import matplotlib.pyplot as plt
#Getting the Centroids
centroids = kmeans.cluster_centers_
label = kmeans.labels_
u_labels = np.unique(label)

#plotting the results:

for i in u_labels:
    plt.scatter(X[label == i , 0] , X[label == i , 1] , label = i)
plt.scatter(centroids[:,0] , centroids[:,1] , s = 80)
plt.legend()
plt.show()
```



## k-médias - Exercício

Utilize o *k-means* para fazer a clusterização dos dados. Use  $k = 2$  e considere os centróides iniciais representados pelo objeto 4 (4,5) e 6 (13,15).

	x	y
1	12	25
2	9	18
3	15	27
4	4	5
5	8	7
6	13	15
7	12	18
8	25	19
9	7	3
10	8	12

## k-médias - python

### Observações sobre os parâmetros

- *n\_init*: número de execuções com diferentes centróides, onde o melhor modelo terá o melhor SSE e será selecionado.
- *max\_iter*: número de iterações máximo em cada execução do algoritmo. O algoritmo para quando se atinge esse número ainda que a função não tenha convergido.
- Algumas execuções podem não convergir quando se usa grandes valores de *max\_iter*. Logo o valor do parâmetro *tol* (tolerância) deve ser alto dado que o parâmetro controla a tolerância em relação as variações do SSE para atingir a convergência do algoritmo.

## K-medoides

- Baseado na representatividade dos objetos no cluster.
- Não usa a média para calcular o centroide do cluster, mas um objeto como referência para representar os medoides dos clusters.
- Objetos que não foram selecionados como representativos (medoides) são iterativamente assinalados ao medoide mais similar a cada objeto.
- O algoritmo minimiza a soma de dissimilaridades entre os objetos e seus respectivos clusters de referência.
- Uso da função de erro absoluto (E).

$$E = \sum_{j=1}^k \sum_{p \in C^{(j)}} |p - O_j|$$

onde E é a soma do erro absoluto de todos os objetos em  $D$ ,  $p$  é um objeto não representativo em um cluster  $C^{(j)}$  e  $O_j$  é o objeto representativo do cluster  $C^{(j)}$  (medoide).

## K-medoides

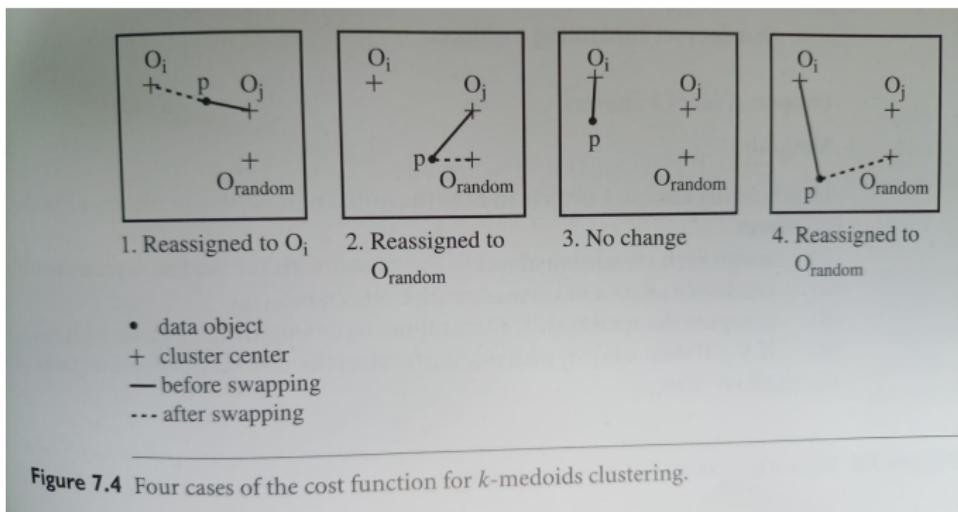
- O algoritmo contorna os problemas de sensibilidade a ruídos e *outliers* do k-médias.
- O algoritmo k-medóide escolhe arbitrariamente os medoides iniciais.
- O algoritmo itera até que a troca de medoides não melhore a função de erro absoluto.
- Um algoritmo que implementa o k-medoides é o algoritmo de particionamento em torno de medoides (PAM - Partitioning Around Medoids).
- Complexidade  $\mathcal{O}(k(n - k)^2)$ , onde o custo aumenta quando k e n aumentam.

# K-medoides

4 casos são analisados:

- ①  $p \in O_j$ . Se trocar  $O_j$  com um objeto representativo  $O_{random}$  e  $p$  ficar mais próximo de um objeto representativo  $O_i$ ,  $i \neq j$ , então  $p$  é assinalado a  $O_i$ .
  - ②  $p \in O_j$ . Se trocar  $O_j$  com um objeto representativo  $O_{random}$  e  $p$  ficar mais próximo de  $O_{random}$ , então  $p$  é assinalado a  $O_{random}$ .
  - ③  $p \in O_i$ ,  $i \neq j$ . Se trocar  $O_j$  com um objeto representativo  $O_{random}$  e  $p$  continuar próximo ao  $O_i$ , então  $p$  continua assinalado a  $O_i$ .
  - ④  $p \in O_i$ ,  $i \neq j$ . Se trocar  $O_j$  com um objeto representativo  $O_{random}$  e  $p$  ficar mais próximo de  $O_{random}$ , então  $p$  é assinalado a  $O_{random}$ .
- 
- A cada iteração é recalculada a função de erro absoluto. Se a diferença entre o erro atual e o anterior é negativa então  $O_j$  é trocado pelo  $O_{random}$  porque houve redução de custo. Caso contrário,  $O_j$  é aceitável e não muda nada na iteração.

# K-medoides



## k-médias e k-medoides

- Quanto à robustez: k-medoides é melhor do que k-médias por ser menos sensível a *outliers*.
- Quanto ao custo computacional: k-médias possui menor custo do que o k-medoide.
- Alternativa: Utilizar o algoritmo k-médias, porém substituindo a média pela mediana, já que essa não é influenciada pelo *outlier*.

Fontes para algoritmos de particionamento: Faceli et al. 2021, Raschka 2015, Han e Kamber 2006.

# Hierárquico

- 4.3 Conceitos de métodos baseados em Hierarquia
- 4.4 Algoritmos BIRCH e ROCK

# Conceitos de métodos baseados em Hierarquia

Considere:

- Algoritmos baseados em hierarquia usam divisão/aglomeramento iterativo até que o critério de parada seja atingido.
- Algoritmos usam métricas de integração (*linkage metrics*) para medir a distância (ou similaridade) entre os clusters.
- Possuem complexidade quadrática ( $\mathcal{O}(n^2)$ ) ou cúbica ( $\mathcal{O}(n^3)$ ) dependendo do algoritmo e da estratégia de implementação.
- Não é possível melhorar um cluster ruim durante o processo por não ter um *traceback*, ou seja, objetos agrupados em um cluster permanecem no cluster até o final das iterações.
- Sensibilidade a ruídos e outliers.
- Dificuldade em lidar com aglomerados de diferentes tamanhos e formas não globulares.
- Ponto para dividir grandes clusters: Dependendo da estrutura dos dados, pode ser difícil identificar subclusters relevantes dentro de um grande cluster.

# Conceitos de métodos baseados em Hierarquia

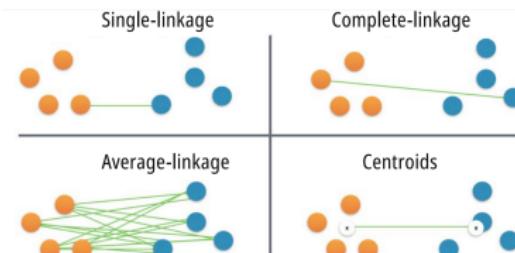
Considere:

- A entrada do algoritmo é a matriz de proximidade (ou distância), produzindo uma sequência de partições aninhadas.
- Algoritmos não tem função global. Todas as decisões são baseadas na melhor opção local.
- Critério de parada é vago: A escolha do critério de parada pode ser ambígua. Definir o número de clusters ou a distância mínima entre clusters pode ser subjetivo e depende do contexto do problema.
- Objetos que pertencem a um nível também estão juntos em níveis superiores, construindo uma hierarquia.
- Uso de dendograma para visualização dos agrupamentos.

# Conceitos de métodos baseados em Hierarquia

Qual distância usar para gerar os clusters?

- A distância entre os pontos mais próximos nos dois clusters.
- A distância entre os pontos mais distantes nos dois clusters.
- A distância média entre todos os pontos de dados nos dois clusters.
- A distância entre os centróides de dois clusters.



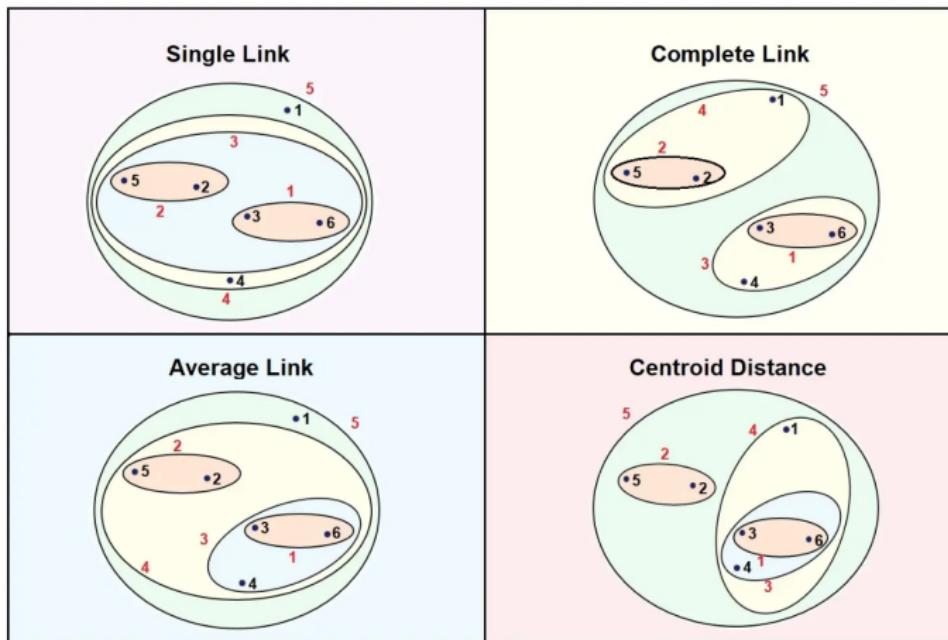
Fonte da imagem: Piedrahita 2020

# Distâncias

- Distância mínima:  $d_{min}(C_i, C_j) = \min_{p \in C_i, p' \in C_j} |p - p'|$
- Distância máxima:  $d_{max}(C_i, C_j) = \max_{p \in C_i, p' \in C_j} |p - p'|$
- Distância média entre os pontos:  
$$d_{medP}(C_i, C_j) = \frac{1}{n_i \times n_j} \sum_{p \in C_i} \sum_{p' \in C_j} |p - p'|$$
- Distância média entre os centróides:  $d_{medC}(C_i, C_j) = |\mu_i, \mu_j|$

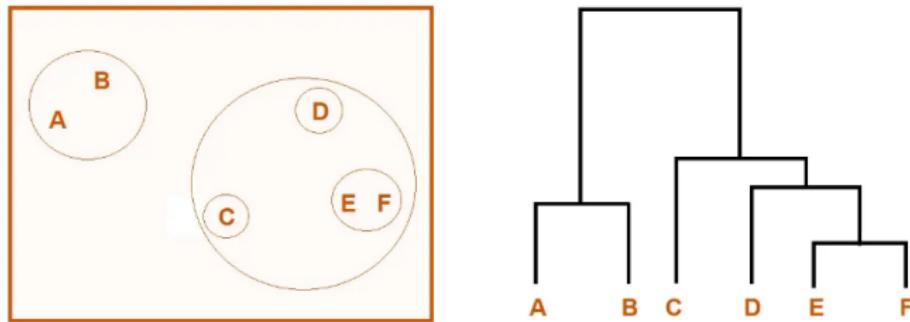
OBS:  $d_{medP}(C_i, C_j)$  e  $d_{medC}(C_i, C_j)$  tendem a superar os efeitos de sensibilidade aos ruídos e *outliers* por ser uma "combinação" entre  $d_{min}(C_i, C_j)$  e  $d_{max}(C_i, C_j)$ .

# Conceitos de métodos baseados em Hierarquia



Fonte da imagem: Hierarchical Clustering in Machine Learning

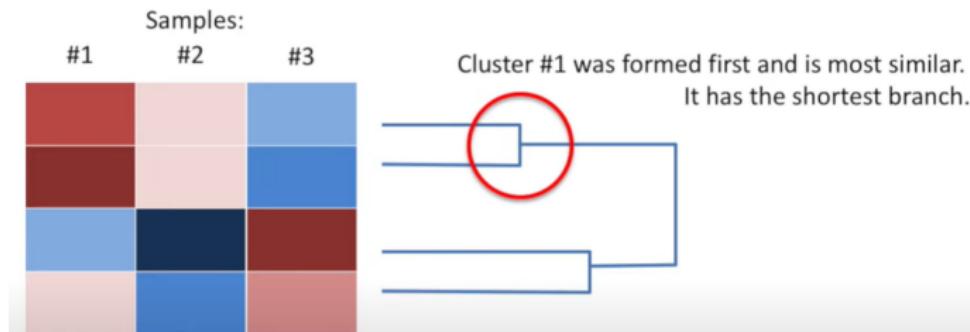
# Conceitos de métodos baseados em Hierarquia



Fonte da imagem: Hierarchical Clustering in Machine Learning

# Conceitos de métodos baseados em Hierarquia

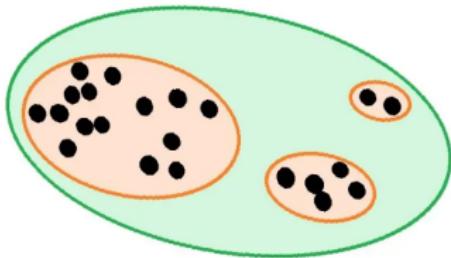
Dendograma e mapa de calor (*heatmap*) para visualização dos dados



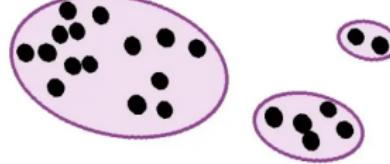
Fonte da imagem: StatQuest: Hierarchical Clustering (YouTube)

# Hierarquia vs Particionamento

Hierarchical Clustering



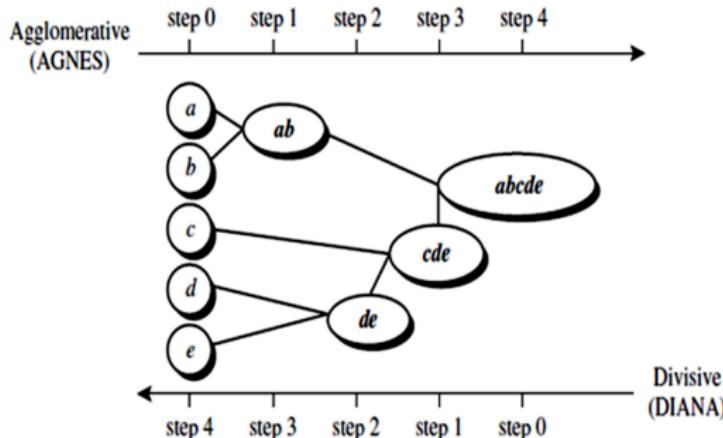
Partitional Clustering



Fonte da imagem: Hierarchical Clustering in Machine Learning

## AGNES e DIANA

- AGNES: AGglomerative NESting. Critério de particionamento pode ser, como exemplo, a menor distância euclidiana.
- DIANA: DIvisive ANALysis. Critério de particionamento pode ser, como exemplo, a maior distância euclidiana.



Fonte da imagem: Han e Kamber 2006

## AGNES - AGglomerative NESting

- Inicialmente cada objeto é um cluster individual.
- Os clusters mais próximos são fundidos em um único cluster.
- O processo de fusão é repetido até que o número de clusters seja alcançado.

## DIANA - DIvisive ANAlysis

- Inicialmente todos os objetos estão no mesmo cluster.
- O maior cluster é repartido entre dois clusters similares menores.
- O processo é repetido até que o número desejado de clusters é atingido.

## BIRCH - Balanced Iterative Reducing and Clustering using Hierarchies [Han e Kamber 2006]

- Particionamento hierárquico iterativo de objetos usando estrutura de árvore.
- Agrupamento de dados numéricos por integrar clusterização hierárquica (etapa de microclustering) com outros métodos de particionamento (etapa de macroclustering).
- Mitiga problemas de escalabilidade e de desfazer operações que ocasionaram em clusters de baixa qualidade.
- Baseados nos conceitos de cluster de características (objeto) e árvore de clusters de características para sintetizar as representações dos clusters.
- Complexidade  $\mathcal{O}(n)$  em relação ao número de objetos.
- Ideal para clusters esféricicos devido ao uso da noção de raio e diâmetro para controlar os limites de um cluster.

# BIRCH - Balanced Iterative Reducing and Clustering using Hierarchies [Han e Kamber 2006]

Cluster de características (*CF* - Clustering Feature): Vetor tri-dimensional que sintetiza as informações do cluster de objetos.

Considere um cluster  $x_i$  contendo  $n$  objetos de  $d$  dimensões (características), então o *CF* do cluster é definido como:

$$CF = \langle n, LS, SS \rangle$$

*LS* é a soma linear dos pontos (características dos objetos) e *SS* é a soma quadrada dos pontos.

$$LS = \sum_i^n X_i \quad SS = \sum_i^n X_i^2$$

# BIRCH - Balanced Iterative Reducing and Clustering using Hierarchies [Han e Kamber 2006]

Centroide do cluster:  $x_0 = \frac{\sum_{i=1}^n x_i}{n}$

Raio do cluster:  $R = \sqrt{\frac{\sum_{i=1}^n (x_i - x_0)^2}{n}}$

Diâmetro do cluster:  $D = \sqrt{\frac{\sum_{i=1}^n \sum_{j=1}^n (x_i - x_j)^2}{n(n-1)}} = \sqrt{\frac{2n(SS) - 2(LS)^2}{n(n-1)}}$

## Exemplo BIRCH

Considere 4 pontos, (1,9), (3,6),(2,4) e (4,5), em um cluster  $C_1$ . O cluster de características é:

$$\langle 4, (1+3+2+4, 9+6+4+5), (1^2 + 3^2 + 2^2 + 4^2, 9^2 + 6^2 + 4^2 + 5^2) \rangle$$
$$\langle 4, (10, 24), (30, 158) \rangle$$

Considere também 3 pontos, (15, 9), (12, 12), (14, 11), em um cluster  $C_2$ , o qual é disjunto de  $C_1$ . O cluster de características é:

$$\langle 3, (15 + 12 + 14, 9 + 12 + 11), (15^2 + 12^2 + 14^2, 9^2 + 12^2 + 11^2) \rangle$$
$$\langle 3, (41, 32), (565, 346) \rangle$$

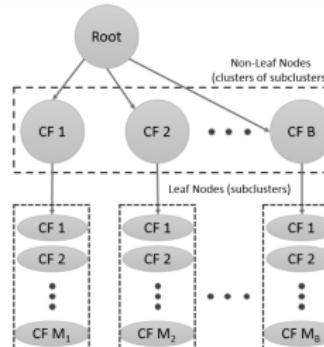
Ao unir os clusters  $C_1$  e  $C_2$  em um cluster  $C_3$ , teremos:

$$\langle 4 + 3, (10 + 41, 24 + 32), (30 + 565, 158 + 346) \rangle$$
$$\langle 7, (51, 56), (595, 504) \rangle$$

## BIRCH: CF Tree

A árvore de características possui dois parâmetros que influenciam no tamanho da árvore:

- Fator de ramificação (*branching*)  $B$ : Número máximo de filhos por **nó não folha**, o qual depende do tamanho da página na memória para que **um nó da árvore CF caiba em uma página**, minimizando a E/S.
- *Threshold T*: Diâmetro máximo dos subclusters armazenados nos **nós folhas** da árvores, sendo o valor de  $T$  selecionado de forma que **toda a árvore CF caiba na memória**, evitando múltiplas iterações para construir a árvore.



Fonte da imagem: Fontanini e Abreu 2018

## BIRCH: Passos iniciais

- Fase 1 — O algoritmo verifica os objetos e constrói uma árvore CF inicial na memória, que pode ser vista como uma compactação multinível dos dados que tenta preservar a estrutura de cluster inerente dos dados.
- Fase 2 — O algoritmo usa um método de agrupamento qualquer para agrupar os nós folha da árvore CF, para obter os clusters finais.

Passos traduzidos de BIRCH Algorithm with working example

## BIRCH: Passos iniciais

- Durante a Fase 1, os objetos são inseridos dinamicamente para construir a árvore CF.
- Um objeto é inserido na entrada folha em que o centroide está mais próximo ao seu valor.
- Se o diâmetro do subcluster armazenado no nó folha, após a inserção, for maior que o limite especificado  $T$ , então o nó folha será dividido.
- Após a inserção bem-sucedida do objeto atual, as informações sobre o objeto inserido são passadas para a raiz da árvore.
- Se a memória necessária para armazenar a árvore CF for maior que a memória disponível então o limite de diâmetro  $T$  é atualizado (com valor maior) e a árvore é reconstruída.

Passos traduzidos de BIRCH Algorithm with working example

## BIRCH: Exercício

Criar um CF Tree e inserir os valores na seguinte ordem:

{22, 9, 12, 15, 18, 27, 11, 36, 10, 3, 14, 32}.

Considere o fator de ramificação  $B = 2$  e o *threshold* do diâmetro  $T = 5$ .

Passo a passo da resolução (parcial) disponível em [BIRCH Algorithm with working example](#)

## ROCK - RObust Clustering using linKs

- Considera o número de vizinhos comuns (**número de links**) entre dois objetos com atributos categóricos.
- Mais robusto que outros algoritmos de clusterização padrões que focam na similaridade entre os pontos.
- Decisões de fusão de clusters baseados em medidas de similaridade (decisão local) para dados binários podem não produzir clusters de boa qualidade (Clusters distintos podem possuir *outliers* ou poucos objetos próximos).
- Abordagem mais global que analisa a vizinhança de pares individuais de objetos, agrupando estes objetos em um cluster se possuírem uma vizinhança similar a ambos (grande número de *links* comuns dado o *threshold*  $\theta$ ).
- Dois pontos  $p$  e  $p'$  são vizinhos se  $sim(p,p') \geq \theta$ . Neste,  $sim(p,p')$  pode ser uma métrica de distância ou não métrica fornecida pelo especialista do domínio, normalizada entre 0 e 1. Mais próximo de 1, mais similar.

# ROCK - RObust Clustering using linKs

Passos:

- Construção de um grafo esparso a partir de uma matriz de similaridade usando o *threshold* de similaridade e o conceito de vizinhança.
- Execução de clusterização hierárquica no grafo esparso.

Complexidade no pior caso:  $\mathcal{O}(n^2 + nm_m m_a + n^2 \log n)$ , com  $n$ ,  $m_m$  e  $m_a$  relacionados respectivamente ao número de objetos, ao número máximo de vizinhos e número médio de vizinhos.

# ROCK - Métrica de similaridade de Jaccard

Considere os clusters contendo transações (valores categóricos)  $C1 = \langle a, b, c, d, e \rangle$  e  $C2 = \langle a, b, f, g \rangle$ .

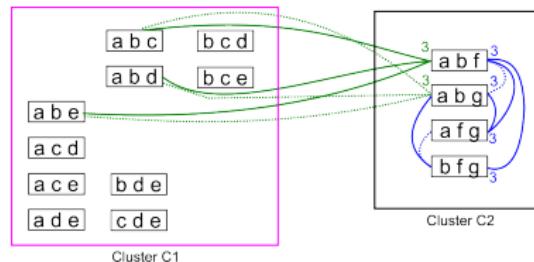
Seja a similaridade de Jaccard dada por  $sim(T1, T2) = \frac{|T1 \cap T2|}{|T1 \cup T2|}$ .

Em  $C1$ , qualquer similaridade entre transações varia entre 0,2 e 0,5.

$sim(\{a, b, c\}, \{b, d, e\}) = \frac{1}{5} = 0,2$  e  $sim(\{a, b, c\}, \{a, b, d\}) = \frac{2}{4} = 0,5$ .

Em  $C2$ , como todos as transações são conectadas entre si, então  $sim(T1, T2) = 0,5$  para qualquer par de transação.

Portanto, Jaccard não é ideal para separar clusters de objetos com atributos categóricos.



# ROCK - Métrica de similaridade baseada em *links*

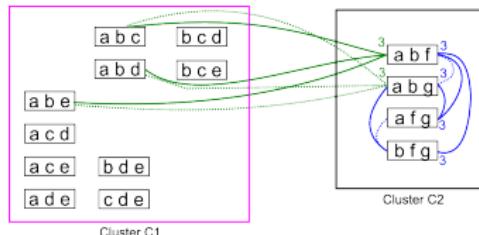
Considere novamente os clusters  $C1 = \langle a, b, c, d, e \rangle$  e  $C2 = \langle a, b, f, g \rangle$ .

A métrica baseada em *links* considera a quantidade de *links* em comum entre duas transações para definir o agrupamento.

Na Figura, observa-se que as transações  $\{a, b, f\}$  e  $\{a, b, g\}$  possuem conexão com as mesmas transações em  $C1$  ( $\{a, b, c\}, \{a, b, d\}$  e  $\{a, b, e\}$ ) e em  $C2$  ( $\{a, f, g\}$  e  $\{b, f, g\}$ ). Logo possuem 5 *links* em comum.

Já  $\{a, f, g\}$  possui 2 *links* com qualquer transação em  $C2$  mas nenhum *link* com  $C1$ .

Portanto a métrica baseada em *links* é ideal para medir a similaridade entre objetos com atributos que assumem valores categóricos.



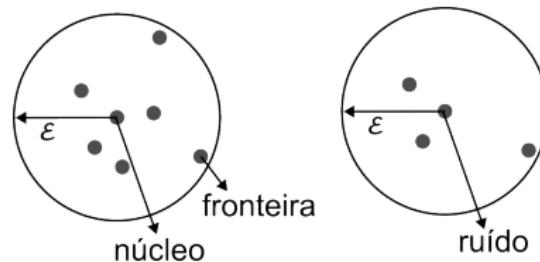
Fontes para algoritmos baseados em hierarquia: Faceli et al. 2021, Raschka 2015, Han e Kamber 2006, Guha, Rastogi e Shim 2000.

# Densidade

- 4.5 Conceitos de métodos baseados em Densidade
- 4.6 Algoritmos DBSCAN e DENCLUE

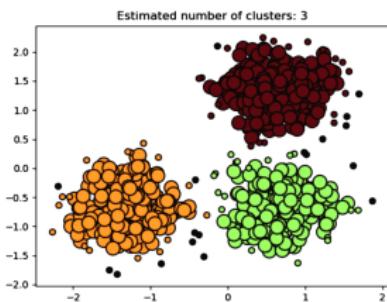
## Métodos baseados em Densidade

- Identificação de grupos densos de pontos para descoberta de clusters de formato arbitrário e identificação de valores discrepantes nos dados.
- Não leva em consideração as distâncias, mas cada região contígua de alta densidade de pontos, separada de outros clusters por regiões esparsas ou vazias.
- Para cada dado dentro de um dado grupo, um número mínimo de pontos deve estar contido na vizinhança de cada dado, considerando uma distância determinada por um raio.

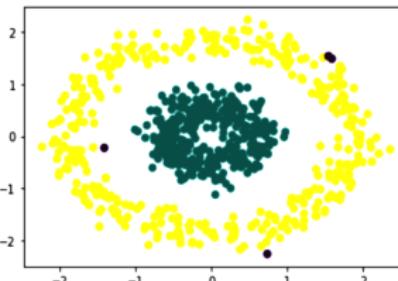


# Métodos baseados em Densidade

- Pontos distantes da região densa são rotulados como ruídos.
- O cluster cresce em qualquer direção dada pela densidade.
- Descoberta de clusters:
  - com formas arbitrárias;
  - de tamanhos diferentes;
  - de máxima homogeneidade, e;
  - com os mesmos níveis de densidade.



DBSCAN to cluster spherical data



DBSCAN to cluster non-spherical data

FONTE: <https://www.kdnuggets.com/2020/04/dbscan-clustering-algorithm-machine-learning.html>

# DBSCAN - *Density-Based Spatial Clustering of Applications with Noise*

- Localiza pontos baseado na densidade dentro de um determinado raio de um ponto núcleo.
- Um dos algoritmos de clusterização mais rápidos e que localiza clusters de diferentes formas.
- Complexidade  $\mathcal{O}(n * \log n)$  no melhor caso, mas pode chegar a  $\mathcal{O}(n^2)$  no pior caso.
- É afetado pela maldição da dimensionalidade quando se usam medidas como a distância euclidiana.
- Otimização de parâmetros para determinar o menor número de pontos (*MinPts*) e o parâmetro de raio *epsilon* para adquirir os melhores clusters.

Particionamento

oooooooooooooooooooooooooooo

Hierárquico

oooooooooooooooooooooooooooo

Densidade

ooo●oooooooooooooooooooo

# DBSCAN - *Density-Based Spatial Clustering of Applications with Noise*

Efeito dos parâmetros:

$\epsilon$	$MinPts$	Saída
Alto	Alto	Poucos clusters, grandes e densos.
Baixo	Alto	Mais clusters, pequenos e densos.
Alto	Baixo	Menos clusters, grandes e pouco densos.
Baixo	Baixo	Muitos clusters, pequenos e pouco densos.

# DBSCAN - *Density-Based Spatial Clustering of Applications with Noise*

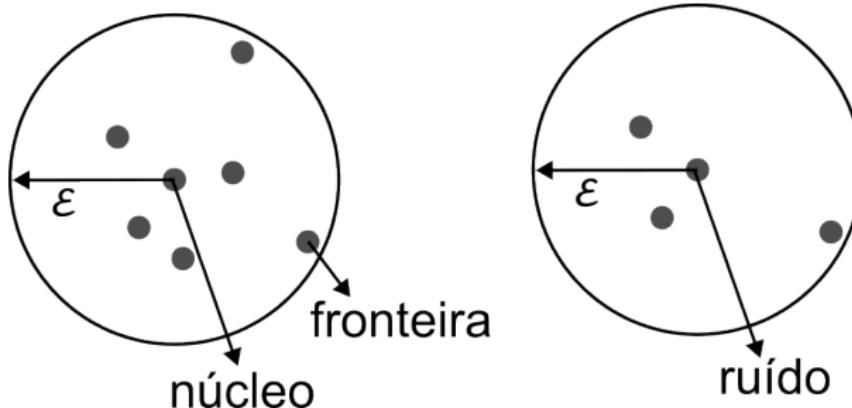
Como estimar corretamente os parâmetros?

- A seleção desses parâmetros geralmente é feita empiricamente, usando uma análise preliminar do conjunto de dados.
- Uma abordagem comum para escolher  $\epsilon$  é a visualização do gráfico de distância dos pontos k-ésimos (*k-distance graph*), onde um cotovelo no gráfico sugere um valor adequado para  $\epsilon$ .
- Quanto ao *MinPts*, recomenda-se escolher um valor próximo ao número de dimensões dos dados mais um (por exemplo, para dados bidimensionais,  $MinPts = 4$ ), mas ajustes adicionais podem ser necessários dependendo do ruído e da densidade dos dados.

Veja mais em **DBSCAN Parameter Estimation Using Python** e **DBSCAN Implementation and Parameter Tuning**

# DBSCAN - *Density-Based Spatial Clustering of Applications with Noise*

- Ponto núcleo: Um ponto é considerado núcleo se houver um número mínimo de pontos (*MinPts*) na vizinhança determinada por um raio  $\epsilon$ .
- Ponto de fronteira: Reside na distância determinado do núcleo até  $\epsilon$  (raio) e possui menor quantidade de vizinhos que *MinPts* considerando a distância  $\epsilon$ .
- Pontos que não estão na fronteira nem são núcleos são chamados de ruídos.



# DBSCAN - *Density-Based Spatial Clustering of Applications with Noise*

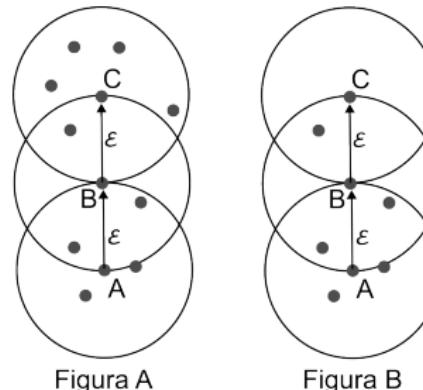
- A vizinhança dentro de um raio  $\epsilon$  de um dado objeto é chamada de vizinhança- $\epsilon$  do objeto.
- Se a vizinhança- $\epsilon$  de um objeto contém pelo menos um número mínimo de objetos ( $MinPts$ ), então o objeto é chamado de um objeto núcleo.
- Dado um conjunto de objetos  $D$ , dizemos que um objeto  $p$  é de densidade diretamente alcançável do objeto  $q$  se  $p$  estiver dentro da vizinhança- $\epsilon$  de  $q$ , e  $q$  for um objeto núcleo.
- Um objeto  $p$  é de densidade alcançável do objeto  $q$  em relação a  $\epsilon$  e  $MinPts$  em um conjunto de objetos  $D$  se houver uma corrente de objetos  $p(1), \dots, p(n)$ , onde  $p(1) = q$  e  $p(n) = p$  tal que  $p(i+1)$  é de densidade diretamente alcançável de  $p(i)$  com relação a  $\epsilon$  e  $MinPts$  para  $1 \leq i \leq n$ ,  $p(i) \in D$ .
- Um objeto  $p$  é de densidade conectada ao objeto  $q$  com relação a  $\epsilon$  e  $MinPts$  em um conjunto de objetos  $D$  se houver um objeto  $o \in D$  tal que  $p$  e  $q$  sejam de densidade alcançável de  $o$  com relação a  $\epsilon$  e  $MinPts$ .

Traduzido diretamente de Han e Kamber 2006

# DBSCAN - *Density-Based Spatial Clustering of Applications with Noise*

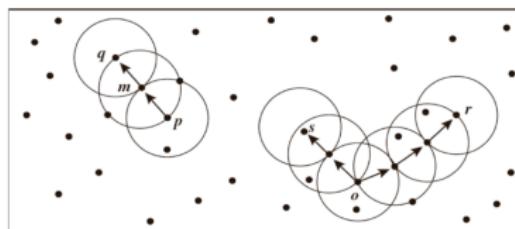
- Alcançabilidade por densidade é um fechamento transitivo da alcançabilidade por densidade direta e o relacionamento deve ser assimétrico.
- Somente objetos núcleos são mutualmente diretamente alcançáveis, logo é uma relação simétrica (Figura A).
- $A$  não é de densidade alcançável de  $C$  porque  $C$  não é um objeto núcleo (número mínimo de pontos na vizinhança inferior a 4 na Figura B).

Traduzido diretamente de Han e Kamber 2006



# DBSCAN - *Density-Based Spatial Clustering of Applications with Noise*

- Dos pontos rotulados,  $m$ ,  $p$ ,  $o$  e  $r$  são objetos núcleo porque cada um está em uma vizinhança- $\epsilon$  contendo pelo menos três pontos.
- $q$  é de densidade diretamente alcançável de  $m$ .  $m$  é de densidade diretamente alcançável de  $p$  e vice-versa.
- $q$  é de densidade alcançável (indiretamente) de  $p$  porque  $q$  é de densidade diretamente alcançável de  $m$  e  $m$  é de densidade diretamente alcançável de  $p$ . No entanto,  $p$  não é de densidade alcançável de  $q$  porque  $q$  não é um objeto núcleo. Similarmente,  $r$  e  $s$  são de densidade alcançável de  $o$ , e  $o$  de densidade alcançável de  $r$ .
- $o$ ,  $r$ , e  $s$  são todos conectados por densidade.



Traduzido diretamente de Han e Kamber 2006 e Figura de Han e Kamber 2006

# DBSCAN - *Density-Based Spatial Clustering of Applications with Noise*

---

**Algorithm 1:** DBSCAN Algorithm

---

```
1 Start with CurrentClusterLabel = 1;  
2 for all core points do  
3   if core point has no cluster label then  
4     CurrentClusterLabel = CurrentClusterLabel + 1;  
5     Label the point with cluster label CurrentClusterLabel;  
6   end  
7   for all points in the  $\epsilon$ -neighbourhood except the  $i^{th}$  point do  
8     if point has no cluster label then  
9       CurrentClusterLabel = CurrentClusterLabel + 1;  
10      Label the point with cluster label CurrentClusterLabel  
11    end  
12  end  
13 end
```

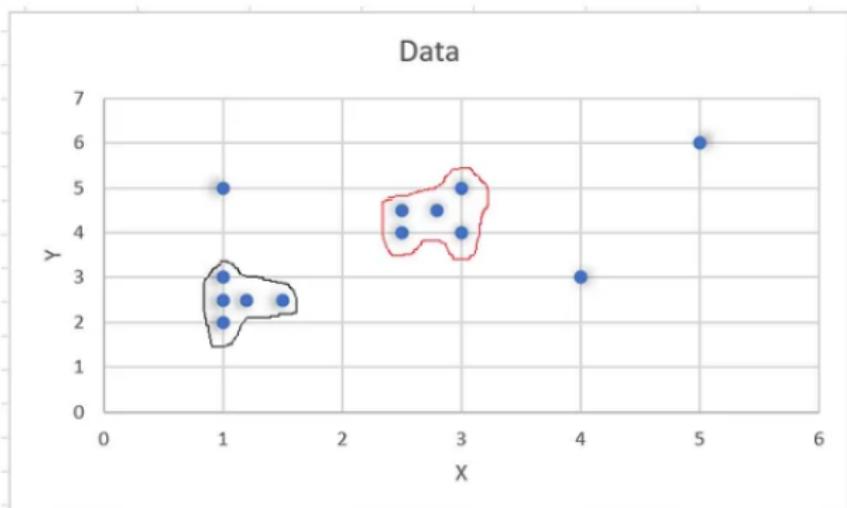
---

**Result:** Optimal density clusters

---

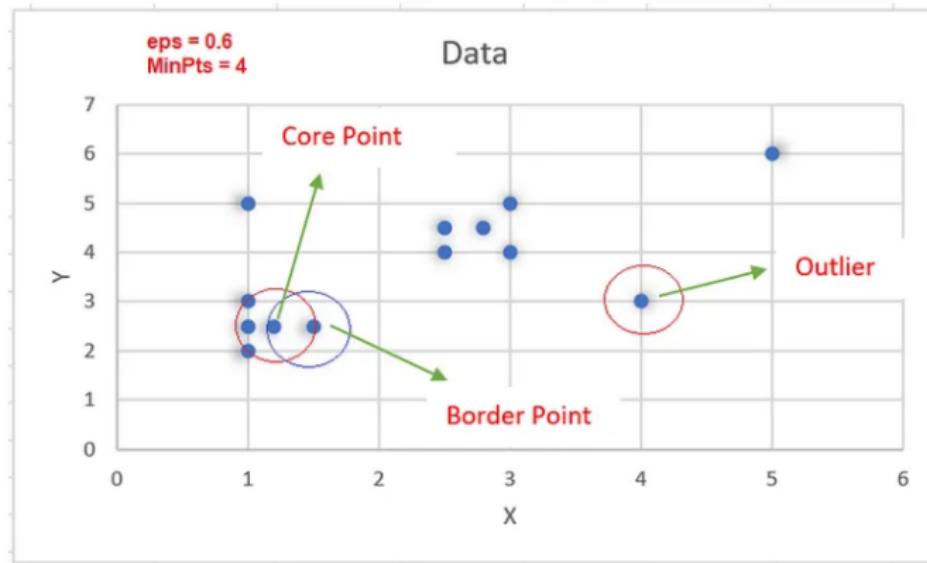
# DBSCAN - *Density-Based Spatial Clustering of Applications with Noise*

X	Y
1	2
3	4
2.5	4
1.5	2.5
3	5
2.8	4.5
2.5	4.5
1.2	2.5
1	3
1	5
1	2.5
5	6
4	3



Fonte: DBSCAN — Make density-based clusters by hand

# DBSCAN - *Density-Based Spatial Clustering of Applications with Noise*



Fonte: DBSCAN — Make density-based clusters by hand

# DBSCAN - *Density-Based Spatial Clustering of Applications with Noise*

X	Y	Distance from (1,2)
1	2	0
3	4	2.8
2.5	4	2.5
1.5	2.5	0.7
3	5	3.6
2.8	4.5	3.08
2.5	4.5	2.9
1.2	2.5	0.53
1	3	1
1	5	3
1	2.5	0.5
5	6	5.6
4	3	3.1

(1, 2.5) e (1.2, 2.5) são os únicos pontos que estão no raio determinado por  $\epsilon$ .

Logo a vizinhança do ponto (1, 2) não possui o número mínimo de pontos necessário para que o ponto (1, 2) seja considerado um ponto núcleo.

Fonte: DBSCAN — Make density-based clusters by hand

# DBSCAN - *Density-Based Spatial Clustering of Applications with Noise*

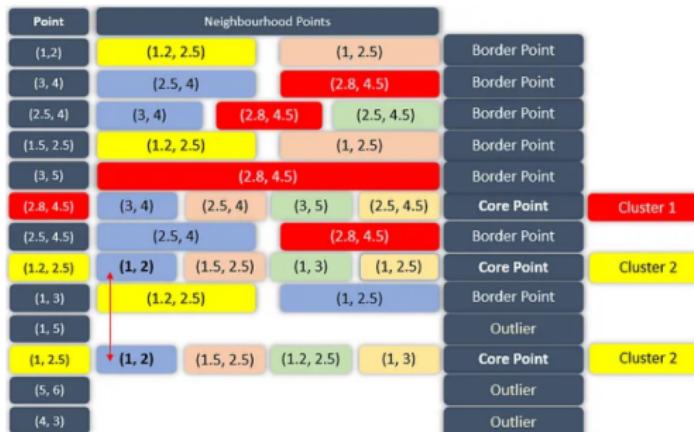
Medir a distância de cada ponto em relação à todos os contidos no conjunto de dados, conforme a imagem mostrada no último slide, e selecionar somente os pontos que estão dentro do raio definido por  $\epsilon$ . Pontos que contém 4 ou mais vizinhos dentro do raio definido por  $\epsilon$  são considerados núcleos. Pontos que não possuem vizinhos são outliers. Os pontos restantes são pontos de fronteira.

Point	Neighbourhood Points				
(1,2)	(1.2, 2.5)		(1, 2.5)		
(3, 4)	(2.5, 4)		(2.8, 4.5)		
(2.5, 4)	(3, 4)	(2.8, 4.5)	(2.5, 4.5)		
(1.5, 2.5)	(1.2, 2.5)		(1, 2.5)		
(3, 5)	(2.8, 4.5)				
(2.8, 4.5)	(3, 4)	(2.5, 4)	(3, 5)	(2.5, 4.5)	Cluster 1
(2.5, 4.5)	(2.5, 4)		(2.8, 4.5)		
(1.2, 2.5)	(1, 2)	(1.5, 2.5)	(1, 3)	(1, 2.5)	Cluster 2
(1, 3)	(1.2, 2.5)		(1, 2.5)		
(1, 5)					
(1, 2.5)		(1.5, 2.5)	(1.2, 2.5)	(1, 3)	Cluster 2
(5, 6)					
(4, 3)					

Fonte: DBSCAN — Make density-based clusters by hand

# DBSCAN - *Density-Based Spatial Clustering of Applications with Noise*

Clusters que possuem pontos em comum são unidos.



Fonte: DBSCAN — Make density-based clusters by hand

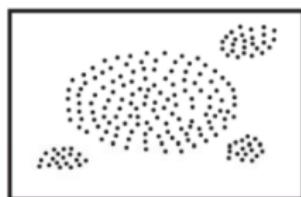
# DBSCAN - *Density-Based Spatial Clustering of Applications with Noise*

Cluster 1	Cluster 2	Outliers
(3,4)	(1, 2)	(1, 5)
(2.5, 4)	(1.5, 2.5)	(5, 6)
(3,5)	(1.2, 2.5)	(4, 3)
(2.8, 4.5)	(1, 3)	
(2.5, 4.5)	(1, 2.5)	

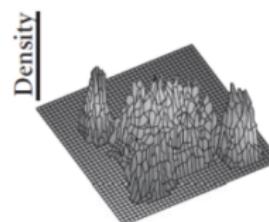
Fonte: DBSCAN — Make density-based clusters by hand

## DENCLUE - *DENsity-based CLUstEring*

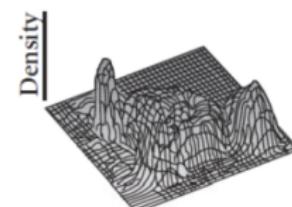
- Densidade global de um conjunto de pontos modelada como soma de funções de "influência" associadas a cada cluster.
- A função de densidade global apresenta picos locais que podem ser usados para definir os clusters.
- A maioria dos pontos não contribui para a função de densidade local, portanto DENCLUE usa uma função que considera somente os pontos que contribuem para tal.
- Se a densidade em um pico local é muito baixa, descarta-se os pontos associados a esse cluster (ruídos). Senão, se dois picos são conectados por um caminho de pontos, onde os pontos têm densidade superior ao *threshold* de densidade mínimo  $\sigma$ , os clusters associados aos picos são unidos.



(a) Data Set



(b) Square Wave



(c) Gaussian

## DENCLUE - *DENsity-based CLUstEring*

- Base matemática sólida.
- Permite uma descrição matemática compacta de clusters de formato arbitrário em conjuntos de dados de alta dimensão.
- Uso de função de estimativa de densidade por *kernel* para descrever a distribuição dos dados pela função de densidade global.
- A função global é a soma das funções associadas, em cada ponto que contribui com uma função de influência ou *kernel*.

$$f_{kernel}^D(x) = \sum_{i=1}^N f_{Square}(x, x_i) = \sum_{i=1}^N 1_{d(x, x_i) \leq \sigma}$$

$$f_{Square}(x, x_i) = \begin{cases} 0 & d(x, x_i) > \sigma \\ 1 & \text{otw} \end{cases}$$

$$f_{kernel}^D(x) = \sum_{i=1}^N f_{Gauss}(x, x_i) = \sum_{i=1}^N e^{-\frac{d^2(x, x_i)}{2\sigma^2}}$$

## DENCLUE - *DENSity-based CLUstEring*

- A medida de distância escolhida deve ser reflexiva e simétrica.
- Função de distância comumente usada é a euclidiana.
- Bom para conjuntos de dados com grandes quantidades de ruído.
- Não é bom para dados com alta dimensionalidade.
- Significativamente mais rápido que outros algoritmos existentes (por exemplo, aprox. 45 vezes mais rápido que DBSCAN).
- Necessita de um grande número de parâmetros portanto é sensível a estes parâmetros, podendo se comportar como DBSCAN, k-médias ou hierárquico.
- Parâmetros: parâmetro de suavização ou largura de banda ( $h$ ), controle de ruído e *outlier* ( $\xi$ ) e taxa de aprendizado para o *hill climbing* encontrar o atrator de densidade ( $\delta$ ).

## DENCLUE - *DENsity-based CLUstEring*

Etapa de pré-processamento:

- Construção do mapa que contém a porção relevante do espaço para acelerar o cálculo da função de densidade.
- Identificação dos atratores de densidade e pontos atraídos correspondentes.
- Geração de hipercubos de dimensão  $d$  com aresta de tamanho  $2\sigma$ .
- Uso de árvore de busca contendo chaves unidimensionais que mapeiam os hipercubos, os quais são numerados de acordo com a sua posição em relação à origem dos dados.

## DENCLUE - *DENsity-based CLUstEring*

Etapa de agrupamento:

- Seleção de cubos mais povoados e cubos conectados a estes.
- Cálculo da função de densidade local para cada ponto  $x_i$  considerando os pontos de clusters conectados ao cluster que contém  $x_i$  e com centróide de distância  $k\sigma$  de  $x_i$ .
- Cada ponto  $x_j$  no caminho de  $x_i$  ao seu atrator de densidade é associado ao cluster que contém  $x_i$  se a  $d(x_i, x_j) \leq \sigma/2$ .
- Clusters associados a atratores de densidade que possui sua densidade menor que  $\xi$  são descartados. Atratores de densidade são unidos se o caminho de pontos entre eles for maior do que  $\xi$ .

# DENCLUE - *DENsity-based CLUstEring*

---

**Algorithm 1:** DENCLUE Algorithm

---

- 1 Partition data space into  $d$ -dimensional grid:  $\epsilon = \sigma$ ;
- 2 Identify highly populated cells and nonempty cells;
- 3 Identify density attractor points,  $C^*$ , using hill climbing:;
- 4 **for** all nonempty cells **do**

  - 5 Randomly pick a point,  $x_i$ ;
  - 6 Compute local density  $f_{kernel}^D(x, x_i)$  with  $\epsilon = 4\sigma$ ;
  - 7 Chose new  $x_{i+1}$  guided by the gradient and compute local density;
  - 8 **if**  $f_{kernel}^D(x, x_i) < f_{kernel}^D(x, x_{i+1})$  **then**

    - 9 Assign all points within distance  $\sigma/2$  of path  $x_i, x_{i+1}, \dots$  to a
    - density attractor  $C^*$

  - 10 **end**
  - 11 Connect the density attractor clusters, using a threshold,  $\xi$ , on the
  - local densities of the attractors;

- 12 **end**

---

**Result:** Optimal density clusters

---

# REFERÊNCIAS

-  FACELI, K. et al. *Inteligência artificial: uma abordagem de aprendizado de máquina*. 2. ed. [S.I.: s.n.], 2021.
-  FONTANINI, A. D.; ABREU, J. A data-driven birch clustering method for extracting typical load profiles for big data. In: IEEE. 2018 IEEE Power & energy society general meeting (PESGM). [S.I.], 2018. p. 1–5.
-  GUHA, S.; RASTOGI, R.; SHIM, K. Rock: A robust clustering algorithm for categorical attributes. *Information systems*, Elsevier, v. 25, n. 5, p. 345–366, 2000.
-  HAN, J.; KAMBER, M. *Data Mining: Concepts and Techniques*. 2rd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2006. ISBN 1-55860-901-6.
-  PIEDRAHITA, E. A. L. Hierarchical clustering for detecting anomalous traffic conditions in power substations. *Ciencia e Ingeniería Neogranadina*, 2020. ISSN 0124-8170. Disponível em: <<https://www.redalyc.org/articulo.oa?id=91164537006>>.
-  RASCHKA, S. *Python machine learning*. [S.I.]: Packt publishing ltd, 2015.