Análise dos algoritmos 2-Aproximados para o problema dos K-Centros

Bernardo D. Lemos[1], **João Lucas S. Moreira**[1]

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brazil

{bernardolemos, joaomoreira}@dcc.ufmg.br

1. Introdução

Nesse trabalho o problema tratado é conhecido como K-Centros e é super comum no área de aprendizado de máquina e possui diversas aplicações práticas em diversos meios, como na biologia computacional, agrupamento de documentos, deteccção de anomalias, segmentação de clientes, etc.

Esse problema pode ser definido de diversas maneiras, mas a usada nesse trabalho será: dado um conjunto de n pontos $P = \{p_1, p_2, \ldots, p_n\}$ no espaço vetorial (X, d), e um parâmetro k, o objetivo é encontrar k centros $C = \{c_1, c_2, \ldots, c_k\}$ tal que a distância máxima r de qualquer ponto $p_i \in P$ ao centro mais próximo $c_j \in C$ seja minimizada. Ou seja:

$$\text{minimizar } r = \max_{p_i \in P}(\min_{c_j \in C} dist(p_i, c_j))$$

sujeito à restrição de que cada ponto p_i está a uma distância máxima r de um dos centros c_j, isto é:

$$\forall p_i \in P, \exists c_j \in C \text{ tal que } d(p_i, c_j) \leq r$$

Entretanto, esse é um problema que pertence a classe *NP-Completa*, ou seja, não existe, até o momento, um algoritmo capaz de resolvê-lo em uma máquina de Turing determinística em tempo polinomial para todas as instâncias. Em outras palavras, a melhor solução conhecida para resolver o problema de maneira exata pode exigir tempo exponencial em relação ao número de pontos n, tornando-se impraticável para conjuntos de dados muito grandes.

Com essa informação, urge o desenvolvimento de algoritmos aproximativos para contornar em partes esse problema, onde será abdicada a certeza da otimalidade, em troca de uma melhora no tempo de execução. No trabalho em questão, serão tratados dois algoritmos 2-Aproximados, ou seja, possuem um raio mínimo encontrado que é no máximo duas vezes pior que o ótimo. Os métodos estudados serão: a abordagem gulosa e a abordagem por meio do refinamento de intervalos. Ambos serão explicados de forma mais completa nos próximos tópicos.

2. Metodologia

Para o trabalho foram usados 30 bases de dados diferentes, sendo divididas igualmente em 3 categorias: dados reais obtidos no site *UCI Machine Learning Repository*, dados sintéticos obtidos a partir de um exemplo de dados da documentação do $sickit-learn$, que vai ser mais detalhada a frente no relatório, e por último 10 conjuntos de dados que foram gerados com apenas duas dimensões utilizando uma distribuição normal multivariada, em que os pontos foram criados em torno de centros com médias diferentes e controlando o desvio padrão para que houvesse vários niveis de sobreposição entre eles.

Para fins de comparação no nível de qualidade da solução de ambos algoritmos aproximativos, o baseline escolhido foi a implementação de K-means do pacote *Scikit Learn* do Python. Já as métricas que foram usadas são clássicas nesse problema e são chamadas de Índice de Rand e Silhueta. A seguir uma breve explicação sobre cada uma delas e o que representam.

2.1. Índice de Rand

O Índice de Rand é uma métrica que avalia a similaridade entre duas partições de um conjunto de dados, comparando quantas vezes pares de pontos estão no mesmo ou em diferentes clusters em ambas as partições. Valores próximos de 1 indicam alta similaridade, enquanto valores próximos de 0 indicam baixa similaridade. Pode ser calculado pela seguinte formúla:

$$\text{Indíce de Rand} = \frac{a+b}{\binom{n}{2}}$$

onde a é o numero de pares de pontos que estão no mesmo cluster em ambas partições, b é o numero de pares de pontos que estão em clusters diferentes em ambas partições e n escolhe 2 é o número total de pares no conjunto. Para calculá-lo são usados o real e o predito.

2.2. Silhueta

É a média dos coeficientes de silhueta de todos os pontos. Para calcular o coeficiente de silhueta para um ponto p_i, usa-se a fórmula:

$$s(p_i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

onde $a(i)$ é a distância média entre o ponto p_i e todos os outros pontos que pertencem ao mesmo cluster que p_i, isso mede quão próximo ele está dos outros pontos dentro do seu próprio cluster. Já $b(i)$ é a distância entre o ponto p_i e o cluster mais próximo que não é o seu próprio cluster.

O valor desse coeficiente varia entre -1 e 1, e sua interpretação é que valores perto de zero indicam sobreposição de grupos, valores negativos indicam que o cluster pode ter sido assinalado ao cluster errado e valores próximos de são o ideal.

2.3. Distâncias

Para realizar o cálculo das distâncias entre os pontos, o grupo implementou a distância de Minkowski, que nada mais é do que a generalização da fórmula da distância euclidiana para um espaço n-dimensional, que possui um parâmetro adicional p que determina qual distância será calculada. Segue a fórmula:

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

Os valores de p escolhidos foram $p = 1$, para o cálculo da distância de Manhattan, e $p = 2$, para o cálculo da distância euclidiana.

3. Implementação

Em relação à implementação, a dupla desenvolveu, utilizando o Python, ambas as abordagens 2-aproximadas para o problema em questão. Elas serão detalhadas à seguir.

3.1. Abordagem gulosa

A abordagem gulosa para o problema de K-Centros consiste em pegar um ponto arbitrário s de um conjunto de pontos $S = P$, colocá-lo no conjunto C de centros e, enquanto o tamanho deste não superar k, será acrescentado à C o ponto s' que possua a maior distância em relação aos demais pontos do conjunto de centros.

Essa abordagem dispensa o cálculo de um raio, por selecionar sempre o ponto mais distante possível. No entanto, ela é 2-aproximada em relação ao raio ótimo, já que podemos considerar a maior distância existente entre um ponto e seu centro mais próximo como o raio da solução. Ou seja, mesmo que ele não seja explicitamente calculado, ele ainda é relevante para a demonstração do desempenho deste algoritmo. Abaixo o pseudocódigo referente a essa abordagem.

2-Aproximado-Guloso(S, k)

```
1:  if k ≥ |S| then return S
2:  else
3:      Selecione um ponto arbitrário s
4:      C ← {s}
5:      while |C| < k do
6:          Selecione s que maximize dist(s, C)
7:          C ← C ∪ {s}
8:      end while
9:      return C
10: end if
```

3.2. Abordagem de refinamento de raios

Esta abordagem é referente ao processo de 'busca binária' feita para selecionar um valor de raio que funcione. Para isso, é definido um intervalo cujo valor mínimo r_{min} é 0 e valor máximo r_{max} é a maior distância entre dois pontos existente. O primeiro valor

de r_{max}, que será chamado de $originalr_{max}$ é salvo e utilizado na condição de parada que será especificada ao decorrer desta seção. Em seguida, o raio da iteração atual é definido como $r = \frac{rmin+rmax}{2}$. Com r definido, se executa o algoritmo *2-Aproximado-Refinamento(S,r)*

2-Aproximado-Refinamento(S, r)

1: $S' \leftarrow S$
2: $C \leftarrow \emptyset$
3: **while** $S' \neq \emptyset$ **do**
4: Selecione arbitrariamente um ponto $s \in S'$
5: Coloque s em C
6: Remova de S' todos os pontos que estiverem a uma distância máxima de $2r$ de s
7: **if** $|C| \leq k$ **then**
8: **return** C
9: **else**
10: **não há solução**
11: **end if**
12: **end while**

Se este algoritmo retornar uma solução válida, ou seja, quando $|C| <= k$, o valor de r_{max} é atualizado para o valor de r definido. Caso contrário, o valor de r será atribuído ao valor de r_{min}. O processo ocorre até que o número de centros seja válido e $(r_{max} - r_{min})/originalr_{max} <= p$, onde p é um percentual de refinamento e pode assumir os valores $0.01, 0.04, 0.09, 0.14, 0.25$. Caso essas duas condições sejam atendidas, ocorreu uma clusterização com sucesso.

4. Experimentos

Para os experimentos, foram utilizados 3 conjuntos de dados, sendo dois destes gerados por meio de algoritmos (sintéticos) e um coletado de dados reais. Eles serão detalhados a seguir. É válido ressaltar que todos eles foram normalizados utilizando a normalização min-max, que é regida pela seguinte fórmula:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

onde X_{max}, X_{min} são o maior e o menor valor presente no conjunto X, respectivamente. O objetivo de realizar essa normalização foi conseguir ter todos as métricas na mesma faixa de valores e assim poder compara-las entre si.

4.1. Distribuição normal multivariada

Para esse conjunto de dados, utilizou-se a função da distribuição normal multivariada da biblioteca Numpy. Foi necessário passar um parâmetro de média e de covariância, além do número de pontos que seriam gerados. Para cada média gerada, os pontos relacionados à ela teriam uma label especifica, que serviria para categorizar o 'agrupamento ótimo' esperado. Um exemplo pode ser visualizado na Figura 1. Vale ressaltar que os dados foram gerados de forma aleátoria, com 8 a 15 médias, sendo o valor delas entre -15 e 15.

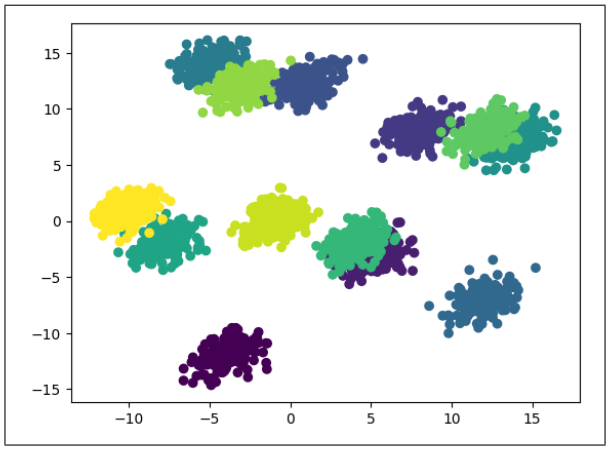

Figura 1. Exemplo de pontos gerados de forma artificial.

4.2. Dados do Scikit Learn

Este conjunto foi extraído da documentação do Scikit-Learn, onde foram criados dados sintéticos utilizando várias abordagens de geração de clusters. Os exemplos incluem distribuições como blobs isotrópicos, círculos concêntricos e luas em forma de meia-lua, permitindo a comparação de diferentes algoritmos de clustering em cenários com diversas formas e complexidades. Essa abordagem tem como objetivo testar a eficácia e a robustez de técnicas de agrupamento. É possível visualizar um exemplo na Figura 2.

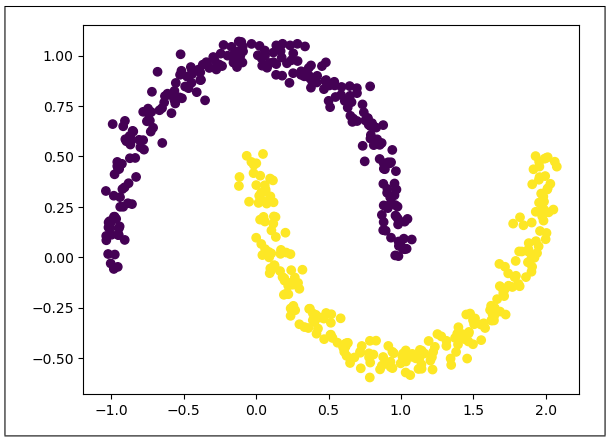

Figura 2. Exemplo de pontos do Scikit Learn.

4.3. Dados reais

O conjunto de dados reais foi extraído do acervo de repositórios da Universidade da Califórnia Irvine que podem ser utilizados para tarefas de aprendizado de máquina, classificação e clusterização. Foram escolhidos conjuntos que possuíssem mais de 1000 instâncias e capazes de serem utilizados nas tarefas de classificação, uma vez que pode-se relacionar a classe de um determinado ponto a um cluster que ele assumiria no caso ótimo. Para estes dados, extraímos os seguinte conjuntos: índices de qualidade de vinhos tintos e brancos, índices de qualidade de grãos secos, dados sobre obesidade, dados sobre

flores, entre outros. Foram coletadas as 5 primeiras colunas iniciais para a execução do algoritmo de clusterização, por uma decisão de projeto.

5. Análise dos resultados

Nesta seção, serão descritos os resultados alcançados por ambos os algoritmos desenvolvidos. Como grupo de controle, foi utilizado o algoritmo de Kmeans da biblioteca Sklearn. Serão descritas subseções para cada tipo de dado, e em cada uma delas existirão mais seções discutindo o desempenho de cada um dos 3 algoritmos.

5.1. Dados de distribuição multivariada

5.1.1. Refinameto de raios

Primeiro será feita a análise de como o percentual de refinamento citado antes altera a qualidade da solução(raio encontrado e as outras métricas), a média dos resultados obtidos foram apresentados na Tabela 1 e na Tabela 2, utilizando a distância Euclidiana e de Manhattan, repectivamente.

Com base nos valores fornecidos, podemos concluir que o método utilizando $p_1 = 0.01$ produziu os melhores resultados em termos de minimizar o raio e o Índice de Rand, sendo o mais eficiente. Apesar disso, para a métrica de silhueta, houve uma diferença significativa entre os diferentes valores de p, sugerindo que a qualidade da separação dos clusters tende a ser melhor quando os valores de p são maiores que 0.09.

p	Raio	Silhueta	Rand
0.01	**0.154 ± 0.01**	0.561 ± 0.03	**0.956 ± 0.01**
0.04	0.160 ± 0.01	0.570 ± 0.04	0.953 ± 0.01
0.09	0.173 ± 0.02	**0.590 ± 0.04**	0.947 ± 0.01
0.16	0.175 ± 0.02	0.589 ± 0.05	0.946 ± 0.01
0.25	0.175 ± 0.02	0.587 ± 0.04	0.946 ± 0.01

Tabela 1. Métricas para o algoritmo de refinamentos com a distância euclidiana no conjunto de dados gerados pela distribuição normal multivariada

p	Raio	Silhueta	Rand
0.01	**0.205 ± 0.01**	0.558 ± 0.04	**0.954 ± 0.01**
0.04	0.214 ± 0.02	0.558 ± 0.04	0.949 ± 0.01
0.09	0.229 ± 0.02	0.572 ± 0.05	0.944 ± 0.01
0.16	0.236 ± 0.02	**0.580 ± 0.05**	0.941 ± 0.01
0.25	0.240 ± 0.02	0.585 ± 0.05	0.941 ± 0.01

Tabela 2. Métricas para o algoritmo de refinamentos com a distância de Manhattan no conjunto de dados gerados pela distribuição normal multivariada

Além disso, foi gerado um plot que permitisse a visualização de uma clusterização, na imagem 3.

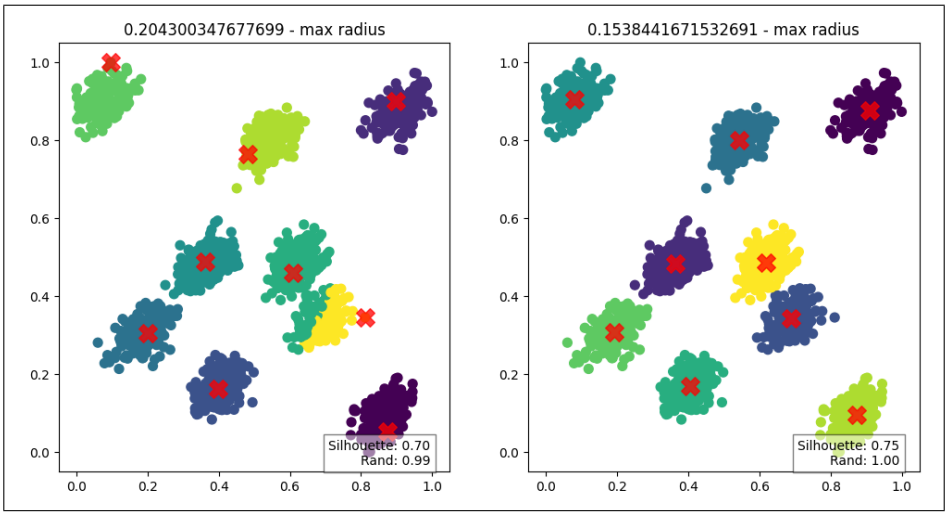

Figura 3. Comparação da clusterização feita pelo algoritmo de refinamento com p = 0.01 e o algoritmo do Kmeans para dados gerados pela distribuição normal multivariada.

5.1.2. Comparativo entre métodos

Com o melhor valor de p extraído ($p = 0.01$), serão construídas tabelas, tanto para a distância euclidiana quanto da Manhattan, de comparação entre o algoritmo de refinamento de raios para este p, o algoritmo guloso e o algoritmo Kmeans da biblioteca Sklearn. Elas podem ser visualizadas nas Tabelas 3 e 4

p	Raio	Silhueta	Rand	Tempo Médio
0.01	0.154 ± 0.01	0.561 ± 0.03	0.954 ± 0.01	0.38
Guloso	0.185 ± 0.01	0.498 ± 0.04	0.944 ± 0.01	0.72
Kmeans	0.143	0.591	0.976	0.05

Tabela 3. Comparativo entre algoritmos com a distância euclidiana no conjunto de dados gerados pela distribuição normal multivariada

p	Raio	Silhueta	Rand	Tempo Médio
0.01	0.205 ± 0.01	0.558 ± 0.04	0.954 ± 0.01	0.39
Guloso	0.239 ± 0.01	0.483 ± 0.03	0.941 ± 0.01	0.79
Kmeans	0.190	0.591	0.976	0.05

Tabela 4. Comparativo entre algoritmos com a distância Manhattan no conjunto de dados gerados pela distribuição normal multivariada

Apesar de possuir uma perfomance melhor que o do outro algoritmo 2-aproximado, o algoritmo de refinamento de raios não foi capaz de produzir nenhum valor que superasse os resultados apresentados pelo Kmeans da Sklearn.

5.2. Dados do Scikit Learn

5.2.1. Refinamento de raios

Assim como na seção anterior, será analisado como a mudança dos percentuais de refinamento afetam a qualidade da solução. Os resultados obtidos podem ser visualizados na Tabela 5 para a distância euclidiana e na 6 para a distância de Manhattan.

É possível perceber que, para a distância euclidiana, os melhores valores foram produzidos pelo $p = 0.01$. O mesmo não ocorre para a distância da Manhattan, que possui diferentes valores de p que são ótimos, mas em ambos, é notável que porcentagens menores produzem resultados melhores.

p	Raio	Silhueta	Rand
0.01	**0.463 ± 0.03**	**0.492 ± 0.03**	**0.755 ± 0.03**
0.04	0.463 ± 0.04	0.488 ± 0.03	0.747 ± 0.04
0.09	0.472 ± 0.04	0.485 ± 0.04	0.744 ± 0.04
0.16	0.475 ± 0.04	0.487 ± 0.04	0.747 ± 0.06
0.25	0.479 ± 0.04	0.488 ± 0.04	0.752 ± 0.04

Tabela 5. Métricas para o algoritmo de refinamentos com a distância euclidiana no conjunto de dados do Scikit Learn

p	Raio	Silhueta	Rand
0.01	0.604 ± 0.05	0.482 ± 0.04	**0.747 ± 0.04**
0.04	0.608 ± 0.06	**0.483 ± 0.04**	0.744 ± 0.04
0.09	**0.603 ± 0.06**	0.479 ± 0.05	0.742 ± 0.05
0.16	0.614 ± 0.06	0.481 ± 0.05	0.745 ± 0.05
0.25	0.622 ± 0.06	0.479 ± 0.05	0.742 ± 0.04

Tabela 6. Métricas para o algoritmo de refinamentos com a distância de Manhattan no conjunto de dados do Scikit Learn

5.2.2. Comparativo entre métodos

Assim como na distribuição normal, o algoritmo de refinamento performou melhor quando $p = 0.01$. Apesar de isso não ser 100% verdade para o caso da distância de Manhattan, é possível perceber que os valores para esta porcentagem que não são ótimos são muito próximos aos valores do K-means do sklearn. Os compartivos podem ser visualizados nas tabelas 7 e 8.

p	Raio	Silhueta	Rand	Tempo Médio
0.01	0.463 ± 0.03	0.492 ± 0.03	0.755 ± 0.03	0.03
Guloso	0.530 ± 0.04	0.448 ± 0.06	0.710 ± 0.06	0.04
Kmeans	0.435	0.532	0.805	0.03

Tabela 7. Comparativo entre algoritmos com a distância euclidiano no conjunto de dados gerados pela distribuição normal multivariada

p	Raio	Silhueta	Rand	Tempo Médio
0.01	0.603 ± 0.05	0.482 ± 0.04	0.747 ± 0.04	0.04
Guloso	0.688 ± 0.07	0.432 ± 0.06	0.705 ± 0.05	0.05
Kmeans	0.585	0.532	0.976	0.04

Tabela 8. Comparativo entre algoritmos com a distância Manhattan no conjunto de dados gerados pela distribuição normal multivariada

Assim como no conjunto de dados anterior, o Kmeans apresentou números melhores do que os algoritmos 2-aproximados desenvolvidos.

5.3. Dados reais

5.3.1. Refinamento de raios

De forma análoga aos anteriores, serão construídas duas tabelas (10 e 9) para a análise dos algoritmos executados com dados reais.

p	Raio	Silhueta	Rand
0.01	$\mathbf{0.810 \pm 0.03}$	$\mathbf{0.227 \pm 0.05}$	$\mathbf{0.592 \pm 0.04}$
0.04	0.813 ± 0.03	0.227 ± 0.05	0.586 ± 0.04
0.09	0.820 ± 0.03	0.220 ± 0.05	0.581 ± 0.05
0.16	0.821 ± 0.04	0.224 ± 0.04	0.589 ± 0.04
0.25	0.830 ± 0.03	0.225 ± 0.05	0.589 ± 0.04

Tabela 9. Métricas para o algoritmo de refinamentos com a distância euclidiana no conjunto de dados do Scikit Learn

p	Raio	Silhueta	Rand
0.01	$\mathbf{1.604 \pm 0.05}$	$\mathbf{0.209 \pm 0.05}$	0.594 ± 0.04
0.04	1.621 ± 0.06	0.207 ± 0.04	$\mathbf{0.603 \pm 0.04}$
0.09	1.623 ± 0.06	0.207 ± 0.04	0.596 ± 0.04
0.16	1.621 ± 0.06	0.204 ± 0.05	0.589 ± 0.03
0.25	1.628 ± 0.06	0.208 ± 0.04	0.589 ± 0.05

Tabela 10. Métricas para o algoritmo de refinamentos com a distância de Manhattan no conjunto de dados do Scikit Learn

Assim como nos dados anteriores, houve um desempenho melhor do algoritmo quando o valor de p é baixo.

5.3.2. Comparativo entre métodos

Apesar de utilizarmos mais de duas dimensões para os dados reais, foram feitas algumas visualizações de clusterizações em duas dimensões para a avaliação visual do desempenho do algoritmo. Na figura 4, é possível perceber uma das fraquezas do método guloso, apresentado à esquerda: apesar de produzir o mesmo número de clusters, eles não são distribuidos de forma homogênea, como no modelo Kmeans. No modelo guloso

muito dos centros sao atribuidos a $outliers$, o que diminui o raio máximo, mas aumenta o raio médio, o que por sua vez piora as métricas.

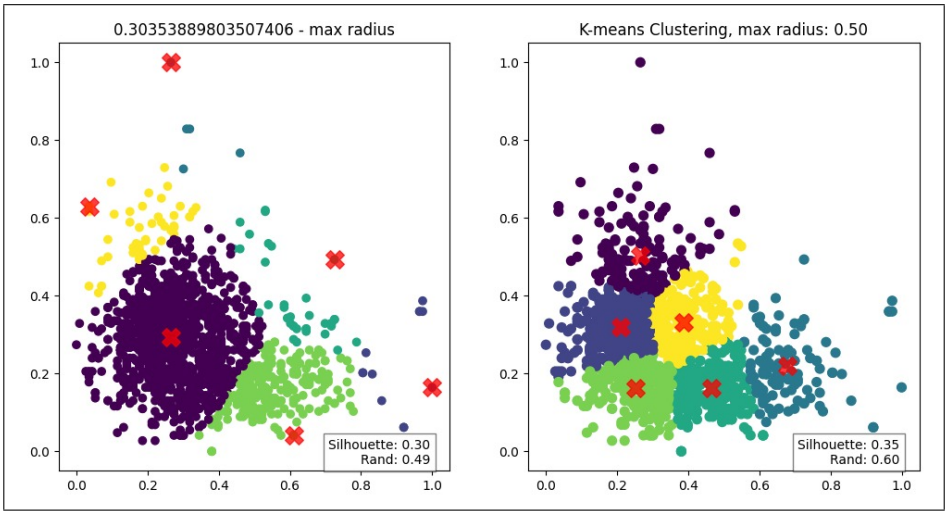

Figura 4. Comparação da clusterização feita pelo algoritmo guloso e o algoritmo do Kmeans para dados reais.

Foram produzidas tabelas para a comparação entre os três métodos.

p	Raio	Silhueta	Rand	Tempo Médio
0.01	0.810 ± 0.03	0.227 ± 0.05	0.592 ± 0.04	0.64
Guloso	0.626 ± 0.04	0.318 ± 0.06	0.610 ± 0.06	0.37
Kmeans	0.746	0.344	0.771	0.08

Tabela 11. Comparativo entre algoritmos com a distância euclidiana no conjunto de dados reais

p	Raio	Silhueta	Rand	Tempo Médio
0.01	1.604 ± 0.05	0.209 ± 0.05	0.593 ± 0.04	0.63
Guloso	1.082 ± 0.06	0.302 ± 0.06	0.635 ± 0.04	0.38
Kmeans	1.330	0.344	0.772	0.09

Tabela 12. Comparativo entre algoritmos com a distância de Manhattan no conjunto de dados reais

Pode ser visto que, apesar de produzir um valor de raio melhor que o do Kmeans, o algoritmo guloso não supera o algoritmo do SkLearn em valores de métrica. Ao contrário do que se mostrou para os dados reais, o algoritmo de refinamento de raios com $p = 0.01$ não houve uma perfomance melhor que o outro 2-aproximativo no conjunto dos dados reais. Isso pode ser justificado pelo aumento da dimensionalidade dos pontos.

6. Conclusão

No que se trata de qualidade de grupos encontrados, a implementação do $sckitlearn$ é bem superior aos metódos aproximativos. Entranto, no que se diz sobre o raio encontrado, a diferença geralmente não é tão grande em nenhuma das três categorias de dados

testados e no caso de conjuntos de dados reais, o raio encontrado pelos aproximativos é ainda menor que o do pacote python.

A abordagem gulosa se mostrou pior que a por refinamento de intervalos, em praticamente todos os quesitos. Mas é importante frisar, que mesmo se considerarmos os melhores resultados de cada um dos testes, nenhum dos outros resultados foi muito pior, sempre se mantendo em um intervalo de 1.2 à 1.5 vezes pior.

Por fim, ajustar o limite do intervalo de busca do algoritmo de refinamentos se mostrou uma estratégia excelente para minimizar o raio, já que os melhores resultados foram com intervalos menores que o $originalr_{max} * 0.01$.

Referências

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.rand_score.html

https://archive.ics.uci.edu/datasets

https://archive.ics.uci.edu/dataset/602/dry+bean+dataset

https://archive.ics.uci.edu/dataset/186/wine+quality

https://archive.ics.uci.edu/dataset/544/estimation+of+obesity+levels+based+on+eating+habits+and+physical+condition

https://archive.ics.uci.edu/dataset/53/iris

https://archive.ics.uci.edu/dataset/545/rice+cammeo+and+osmancik

https://archive.ics.uci.edu/dataset/102/thyroid+disease

https://archive.ics.uci.edu/dataset/697/predict+students+dropout+and+academic+success

https://archive.ics.uci.edu/dataset/76/nursery