Comparação de Algoritmos 2-Aproximados e K-Means para o Problema dos k-Centros

Izabela G. T. P. Pérez, Gustavo N. da Cruz

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

Resumo. *Este artigo apresenta uma análise comparativa entre dois algoritmos de 2-aproximação e o algoritmo K-Means para a resolução do problema dos k-centros. O estudo avalia o desempenho desses algoritmos utilizando distâncias Euclidiana e de Manhattan em datasets reais e sintéticos. Métricas como o índice de silhueta, o índice Rand ajustado e o tempo de execução são utilizadas para avaliar a qualidade e a eficiência das soluções apresentadas assim como a comparação com o desempenho da implementação existente na biblioteca Scikit-learn.*

1. Introdução

O problema dos k-centros envolve alocar k centros dentre um conjunto de pontos para minimizar a maior distância entre um ponto e qualquer centro. Sendo NP-difícil, torna-se praticamente impossível obter uma solução ótima para grandes instâncias desse problema em tempo viável.

Por isso, utilizam-se algoritmos de aproximação para obter soluções próximas ao ótimo em tempo polinomial. Os algoritmos 2-aproximados são aqueles que no pior caso o raio é duas vezes maior que o ótimo. Já o algoritmo K-means da biblioteca Scikit-learn resolve o problema com os algoritmos de [Lloyd. 1982] ou [Elkan. 2003], com complexidade média de $O(knT)$, onde n é o número de amostras e T o número de iterações. A sua complexidade no pior caso é $O(n^{(k+2/p)})$, mas na prática, o algoritmo é muito rápido, embora possa cair em mínimos locais.

Este trabalho compara o desempenho de dois algoritmos 2-aproximados e do K-means para o problema dos k-centros usando distâncias Euclidiana e de Manhattan. A análise utiliza bases de dados sintéticos e reais para explorar o comportamento e a aplicabilidade deles em diferentes cenários.

Precisamos equilibrar a precisão e o custo computacional, especialmente quando tempo e qualidade dos agrupamentos são cruciais. Em problemas complexos de clusterização, as métricas de avaliação determinam a eficácia dos algoritmos. Este trabalho investiga inconsistências em métricas, como o índice de silhueta e o índice de Rand ajustado, e sua influência na escolha dos algoritmos.

2. Métodos e métricas

2.1. Distância de Minkowski

Esse algoritmo é uma generalização da distância euclidiana que permite, dado um ponto em R^n decidir sua distância euclidiana ou de Manhattan de qualquer outro ponto nesse

espaço. Isso é feito apartir do somatório abaixo:

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

Nesse caso quando $p = 1$ a distância é a de Manhattan e quando $p = 2$ a distância é a euclidiana.

2.2. Score de Silhueta

Essa é uma métrica utilizada para avaliar a qualidade da clusterização, medindo a coesão dentro dos clusters e a separação entre clusters distintos. Para um ponto i, ela é calculada como:

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

onde $a(i)$ é a distância média entre o ponto i e todos os outros pontos no mesmo cluster, representando a coesão, e $b(i)$ é a distância média entre o ponto i e todos os pontos no cluster mais próximo, representando a separação.

Valores próximos a 1 indicam uma boa qualidade de clustering, valores próximos a 0 sugerem que o ponto está na fronteira entre clusters, e valores negativos indicam que o ponto pode estar mal posicionado. Um valor médio elevado do índice de Silhouette sugere uma melhor qualidade global dos clusters formados.

2.3. Índice de Rand Ajustado (ARI)

Utilizada para avaliar a concordância entre duas partições de dados, ajustando o Rand Index para levar em conta o acaso. O índice é definido como:

$$\text{ARI} = \frac{\text{RI} - \text{E[RI]}}{\text{max(RI) - E[RI]}}$$

onde RI é o Rand Index, e E[RI] é o valor esperado do Rand Index para partições aleatórias. O valor do ARI varia de -1 a 1, onde 1 indica uma perfeita concordância entre as partições, 0 indica que a concordância é igual ao que seria esperado, e valores negativos indicam uma discordância maior do que a esperada.

2.4. Algoritmo de maiores distâncias

2.4.1. Implementação

O algoritmo consiste em, primeiramente, escolher um ponto aleatório para ser o primeiro centro. Depois, os próximos k-1 centros são escolhidos da seguinte forma: dentre todos os pontos, toma-se aquele que maximiza a menor distância para os centros já escolhidos.

2.4.2. Complexidade

Para cada uma das k-1 iterações, percorremos todos os n pontos, e para cada um, verificamos as distâncias para os $\leq k$ centros já escolhidos. No final, ficam $O(nk^2)$ iterações.

2.4.3. Corretude

Vamos provar que esse algoritmo é de fato 2-aproximativo. Sejam r o raio ótimo e $C = \{c_1, c_2, \ldots, c_k\}$ a escolha ótima de centros. Sejam $\{d_1, d_2, \ldots, d_k\}$ os centros escolhidos pelo nosso algoritmo.

Se existe par i, j com $i \leq j$ tal que dist$(i, j) \leq 2r$, significa que na j-ésima iteração do algorimto o ponto que maximiza a menor distância para os centros tem distância $\leq 2r$, o que significa que todos os pontos estão a uma distância menor ou igual a $2r$ de algum centro, como queríamos demonstrar.

Caso contrário, associe, para todo centro d_i, o centro c_t que minimiza a distância para d_i. Como a solução dos c_t's é ótima, a distância entre d_i e c_t deve ser no máximo r. Além disso, se dois centros d_i e d_j fossem associados a um mesmo c_t, teríamos que

$$\text{dist}(d_i, d_j) \leq \text{dist}(d_i, c_t) + \text{dist}(c_t, d_j) \leq r + r = 2r,$$

contradizendo a nossa hipótese de que não existem i, j com dist$(i, j) \leq 2r$. Logo, cada centro c_t está associado a exatamente um d_i. Já que para todo ponto p do dataset existe c_t com dist$(p, c_t) \leq r$; se d_i é o centro associado a c_t, temos que

$$\text{dist}(p, d_i) \leq \text{dist}(p, c_t) + \text{dist}(c_t, d_i) \leq r + r = 2r,$$

como queríamos demonstrar.

2.5. Algortimo de variação do raio

2.5.1. Implementação

O algoritmo consiste em escolher um raio r, e depois proceder da seguinte maneira: selecionar aleatoriamente um ponto não coberto do dataset, adicionar ele aos centros, e marcar todos os pontos a uma distância menor ou igual a r como cobertos. Se ao final precisarmos de mais do que k centros para cobrir todos os pontos, o r escolhido foi pequeno demais, caso contrário foi grande demais.

Para escolher r, mantemos um intervalo de busca, que inicialmente é $[0, rmax]$, onde rmax é a distância máxima entre pontos do dataset. A cada iteração, r será o ponto médio do intervalo. Se r foi pequeno demais, atualizamos o limite inferior para r, caso contrário atualizamos o limite supeior.

2.5.2. Complexidade

Para uma dada escolha de r, precisamos fazer k iterações de escolhas de centros, e para cada uma dessas iterações, precisamos passar pelos n pontos para saber se foi coberto ou não. Logo, fixado r, a complexidade é $O(nk)$. Em cada iteração da busca binária reduzimos o intervalo de busca pela metade, portanto, se buscamos um intervalo p vezes o tamanho original, precisamos de $\log(\frac{1}{p})$ iterações. Deste modo, a complexidade final é $O(nk \log(\frac{1}{p}))$.

2.5.3. Corretude

Vamos provar que o algoritmo é 2-aproximado. Sendo r_{opt} o raio ótimo, queremos provar que o algoritmo sempre encontra menos que k centros quando $r \geq 2r_{opt}$. Sejam $C_1, C_2, \ldots, C_k$ os clusters de uma solução ótima, e $c_1, \ldots, c_k$ seus respectivos centros. Seja d um centro escolhido pelo algoritmo. Temos que $d \in C_i$ para algum i. Nesse caso, para qualquer outro $p \in C_i$ vale que:

$$\text{dist}(p, d) \leq \text{dist}(p, c_i) + \text{dist}(c_i, p) \leq 2r_{opt} \leq r$$

Logo, d cobre todo C_i quando $r \geq 2r_{opt}$. Deste modo, a cada centro cobrimos uma cluster inteira, o que implica que, no total, utilizamos no máximo k centros.

3. Datasets

3.1. Dados reais

Para os datasets com dados reais, escolhemos 10 conjuntos de dados da plataforma *UC Irvine Machine Learning Repository* [UCI.] para testar os algoritmos em dados diversos. Os conjuntos escolhidos foram:

1. **Absenteeism at Work:** Dados sobre a ausência de funcionários em uma empresa, com 740 instâncias e 21 features.
2. **Statlog (Vehicle Silhouettes):** Dados de silhuetas de veículos para classificação, com 946 instâncias e 18 features.
3. **Rice (Cammeo and Osmancik):** Dados de imagens de grãos de arroz de duas variedades, com 3810 instâncias e 7 features.
4. **Raisin:** Dados de imagens de uvas com 900 instâncias e 7 features.
5. **Wine Quality:** Avaliação de qualidade de vinhos (tinto e branco), com 6497 instâncias e 12 features.
6. **Blood Transfusion Service Center:** Dados sobre doações de sangue, com 748 instâncias e 4 features.
7. **Mammographic Mass:** Dados de massas mamárias para diagnóstico, com 961 instâncias e 5 features.
8. **Breast Cancer Wisconsin (Original):** Dados de diagnóstico de câncer de mama, com 700 instâncias e 10 features.
9. **Maternal Health Risk:** Avaliação de riscos de saúde materna, com 1014 instâncias e 7 features.
10. **Yeast:** Dados sobre localização de proteínas em células de levedura, com 1484 instâncias e 8 features.

3.2. Dados sintéticos

3.2.1. Dados retirados da Scikit-learn

Esse conjunto de dados foi obtido apartir da referência fornecida na descrição do trabalho [Scikit]. Com código obtido foram gerados seis tipos diferentes de conjuntos de dados, cada um com 500 amostras:

1. **Noisy Circles:** Um conjunto de círculos concêntricos com ruído adicionado.

2. **Noisy Moons:** Dois semicírculos com ruído adicionado.
3. **Varied:** Conjunto de dados com ”blobs”de variâncias variadas, para testar a robustez dos algoritmos com clusters de diferentes densidades.
4. **Aniso:** Dados gerados com distribuição anisotrópica, obtidos pela transformação linear de ”blobs”
5. **Blobs:** Conjunto de dados com três grupos bem definidos, usados como base para outros conjuntos.
6. **No Structure:** Dados gerados aleatoriamente sem uma estrutura aparente..

Os datasets podem ser visualizados na figura 1 abaixo com a mesma numeração:

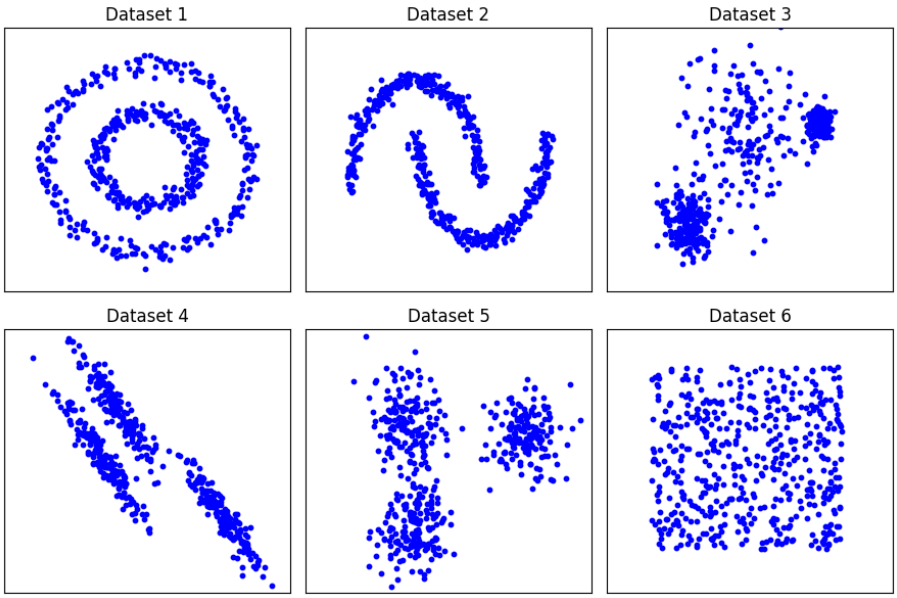

Figura 1. Visualização dataset sintético 1

3.2.2. Distribuição normal multivariada

Esse dataset foi gerado usando o método de distribuição normal multivariada, onde cada conjunto contém três centros de clusters, posicionados aleatoriamente em um espaço bidimensional. Para cada centro, 100 pontos foram amostrados, resultando em clusters de tamanhos iguais. Os conjuntos de dados foram gerados com três diferentes desvios padrão (0.1, 0.5, 1.0), representando baixa, média e alta dispersão dos pontos em torno dos centros dos clusters. Ao todo, foram criados 30 datasets, variando tanto a localização dos centros quanto a dispersão dos pontos.

A figura 2 mostra os datasets com seus clusters verdadeiros exibidos em azul, verde e amarelo.

4. Experimentos

4.1. Algoritmo de maiores distâncias

Inicialmente, os dados foram padronizados usando StandardScaler, garantindo que todas as características tivessem média zero e variância unitária. Para cada dataset, foram geradas as matrizes de distância utilizando as métricas de distância de Manhattan ($p = 1$) e

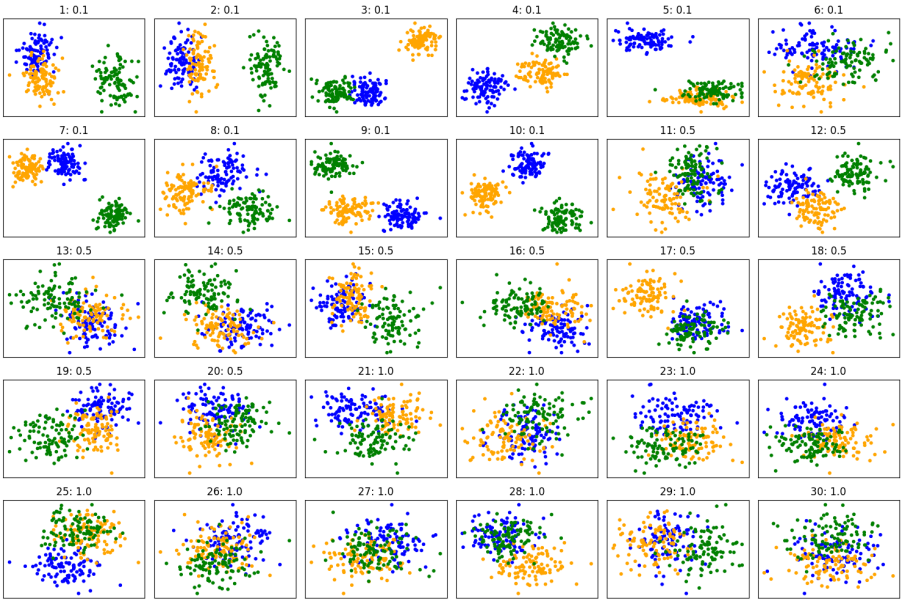

Figura 2. Visualização dataset sintético 2

Euclidiana ($p = 2$). Em seguida, o algoritmo foi executado 30 vezes em cada dataset para garantir robustez nos resultados já que a inicialização do primeiro centro é aleatória.

Durante cada execução, o algoritmo de maiores distâncias selecionou iterativamente k centros que maximizavam a menor distância entre qualquer par de centros. O tempo de execução foi registrado, e os rótulos dos clusters foram determinados pelo centro mais próximo em termos da métrica escolhida. Após a atribuição dos rótulos, o desempenho do agrupamento foi avaliado por meio do score de silhueta da Scikit-learn (personalizado, para casos de métricas não-euclidianas), e pelo índice de Rand ajustado da mesma biblioteca, que compara os agrupamentos encontrados com os rótulos verdadeiros.

Os resultados obtidos, incluindo o raio dos clusters, score de silhueta, índice de Rand ajustado e tempo de execução, foram salvos para análise posterior. E também foram feitos e armazenados os mesmos experimentos utilizando o algoritmo padrão K-means da biblioteca Scikit-learn para que possamos comparar os resultados.

4.2. Algoritmo de variação do raio

Primeiramente, os dadasets foram padronizados da mesma forma que nos experimentos com algoritmo das maiores distâncias, e as distâncias calculadas de mesma maneira. Em seguida, duas estratégias foram testadas separadamente para lidar com a aleatoriedade do algoritmo:

1. Executar o algoritmo 30 vezes;
2. Garantir que a busca binária só desista de um determinado raio após 30 tentativas fracassadas.

Foram testados também valores diferentes para a quantidade de refinamentos no intervalo de busca. A cada iteração, o tamanho do intervalo se reduz pela metade, o que significa que após 2 iterações o intervalo fica em 25% do tamanho original, e que após 6 iterações fica em 1.6% do tamanho original.

Para datasets em que a distância máxima entre pontos é muito grande, refinar o intervalo de busca até 1% do original pode ainda não ser satifatório. Sendo assim, foi testada também a busca binária com 20 iterações, o que significa um intervalo final de tamanho $2^{-20} \approx 0.0001\%$ do inicial.

Foram testados, então, 2, 3, 4, 5, 6 e 20 iterações, correspondendo, respectivamente, a 25%, 12.5% 6.2%, 3.1%, 1.6% e 0.0001% do tamanho inicial. Para cada um desses valores, e para cada uma das estratégias apresentadas acima, foram armazenados os raios das soluções, os índices de rand ajustado e os scores de silhueta, para uma comparação posterior com os outros algoritmos.

5. Análise dos resultados

5.1. Dados sintéticos

5.1.1. Algoritmo de maiores distâncias

Nos conjuntos de dados com estrutura clara, como os BLOBS, VARIED_BLOBS e desvios padrão de 0.1, o algoritmo de maiores distâncias obteve índices de Silhueta e Rand ajustado relativamente elevados. No caso do conjunto BLOBS com distância Euclidiana, o índice de Silhueta foi de aproximadamente 0.51, sugerindo que houve uma identificação razoável dos grupos. Isso acontece de forma semelhante para o conjunto VARIED_BLOBS com distância de Manhattan, onde o algoritmo alcançou uma Silhueta de 0.54. E no caso do desvio padrão de 0.1 obtivemos índices de Silhueta médios de até 0.69.

Essa performance pode ser atribuída ao fato de que, em datasets onde as classes são bem separadas ou possuem variação controlada nas distâncias entre os pontos, o critério de maximização da distância mínima para a escolha dos centros tende a capturar a estrutura global dos dados de forma satisfatória. Assim, mesmo que o algoritmo não seja otimizado para maximizar a coesão interna dos clusters, ele ainda é capaz de criar agrupamentos que refletem bem as separações naturais do espaço de dados.

Em contrapartida, o desempenho do algoritmo de maiores distâncias em datasets mais complexos e com pontos distribuídos ao redor de um centro, como NOISY_CIRCLES, NOISY_MOONS e o desvio padrão de 1.0, foi inferior, com índices de Silhueta na faixa de 0.32 a 0.40 e Rand ajustado em torno de 0.01 a 0.38. Esses resultados indicam que o algoritmo encontrou dificuldade em capturar a estrutura desses dados, provavelmente devido à dificuldade de posicionar os centros dada a disposição dos dados não ser favorável ao algoritmo ou à sobreposição dos clusters naturais.

Esses resultados sugerem que o algoritmo de maiores distâncias, ao se basear na maximização da distância mínima entre centros, pode falhar em contextos onde os clusters não são bem definidos ou onde há significativa sobreposição entre eles. O método prioriza a escolha de centros distantes, sem considerar adequadamente a distribuição interna dos dados dentro de cada cluster, o que leva a uma clusterização menos coesa.

Porém, o algoritmo de maiores distâncias, embora limitado em alguns cenários, é uma alternativa 2-aproximada que, em alguns casos, é capaz de retornar resultados aceitáveis em menor tempo de execução. Por exemplo, o algoritmo mostrou-se competitivo, alcançando tempos de 1.2 ms, especialmente para distâncias Euclidianas em datasets como BLOBS e desvio padrão de 0.1, onde o tempo médio de execução foi cerca de me-

tade do tempo utilizado pelo K-means. Essa efetividade se manteve em todos os datasets sintéticos.

5.1.2. Algoritmo de variação do raio

Nos conjuntos de dados gerados a partir da distribuição normal multivariada, o algoritmo mostrou ótimos resultados quando os grupos são gerados com baixo desvio padrão. Nesses, o índice de silhueta médio ficou acima de 0.6, chegando a um máximo de 0.75, maiores que aqueles alcançados pelos outros algoritmos. Já nos datasets com maior desvio padrão, a performance foi equivalente ao do de maiores distâncias, e inferior ao k-means.

Observa-se que um aumento número de refinamentos no intervalo da busca binária não altera expressivamente os resultdos em relação ao índice se Silhueta e ao índice de Rand, porém trás uma leve melhora no raio da solução.

Em relação aos tempos de execução, os resultados foram muito variados. As execuções do algoritmo de variação do raio com poucas refinações no intervalo foram extremamente rápidas para a distância de Manhattan, sendo de 3 a 10 vezes mais rapidas que os outros algoritmos. Por exemplo: o de variação do raio com 3 iterações demorou uma média de 0.0007 segundos no NOISY CIRCLES com distância de Manhattan, enquanto o de maiores distâncias teve 0.006 em média. Porém, com as distâncias euclideanas, o de variação do raio foi mais lento: 0.01 versus 0.003. Essa discrepância também foi observada nos outros datasets.

5.2. Dados Reais

5.2.1. Algoritmo de maiores distâncias

Em alguns datasets como o **Blood Transfusion Service Center**, **Mammographic Mass** e o **Raisin** o algoritmo de Maiores Distâncias obteve uma silhueta média acima do K-means em ambas as métricas de distância. Já em datasets como o **Wine Quality** e o **Statlog(Vehicle Silhouettes)** isso aconteceu apenas com a métrica euclidiana, apesar dos valores serem bem próximos. Isso mostra que, provavelmente devida à melhor distinção das classes desses dados, o nosso algoritmo consegue selecionar os clusters de forma satisfatória.

Porém, em todos os casos o desvio padrão das métricas de qualidade para o K-means se manteve em 0, diferentemente do nosso algoritmo. Isso indica que o K-means tem uma performance mais estável nas execuções. O índice de Rand ajustado do K-means também foi superior ao Maiores Distâncias em todos os datasets, apesar dessa métrica ainda ser positiva em todos os experimentos para o nosso algoritmo.

De forma semelhante aos experimentos com o dataset sintético, o tempo de execução médio do Maiores Distâncias foi significante menor que o do algoritmo de K-means, chegando em metade deste na maioria dos casos.

5.2.2. Algoritmo de variação do raio

Em relação ao índice de silhueta, o algoritmo de variação do raio obteve resultados muito similares ao algoritmo das maiores distâncias. Entretanto, em relação ao índice de Rand ajustado, o de variação do raio ficou levemente pior, alcançando índices negativos em alguns casos.

Da mesma forma que os experimentos nos datasets sintéticos, o tempo de execução médio do algoritmo variou de acordo com o tipo de distância: usando a distância euclideana ele foi mais lento, e usando a de Manhattan ele foi mais rápido.

6. Imagens

Na figura 3 podemos ver o resultado de uma execução de cada um dos algoritmos no dataset sintético 1 e na figura 4 o resultado de uma execução no dataset sintético 2.

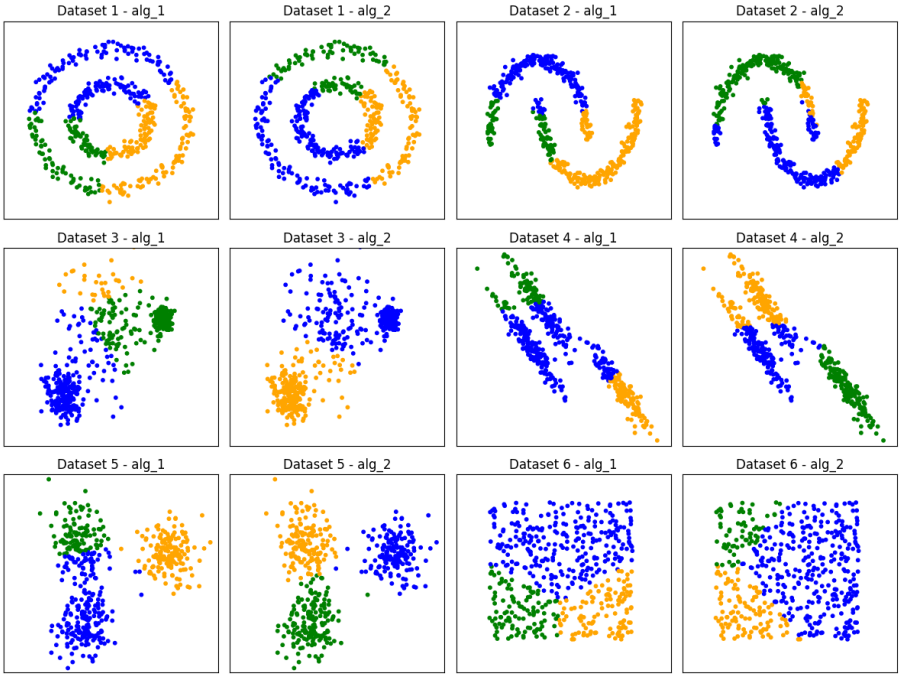

Figura 3. Resultado de uma execução dataset sintético 1

Referências

Elkan., C. (2003). Using the triangle inequality to accelerate k-means. *In Proc. ICML.*

Lloyd., S. (1982). Least square quantization in pcm. *IEEE Trans. on Information Theory, 28(2).*

Scikit. Comparison of clustering algorithms on toy datasets. Accessed: 2024-08-15.

UCI. Uci machine learning repository.

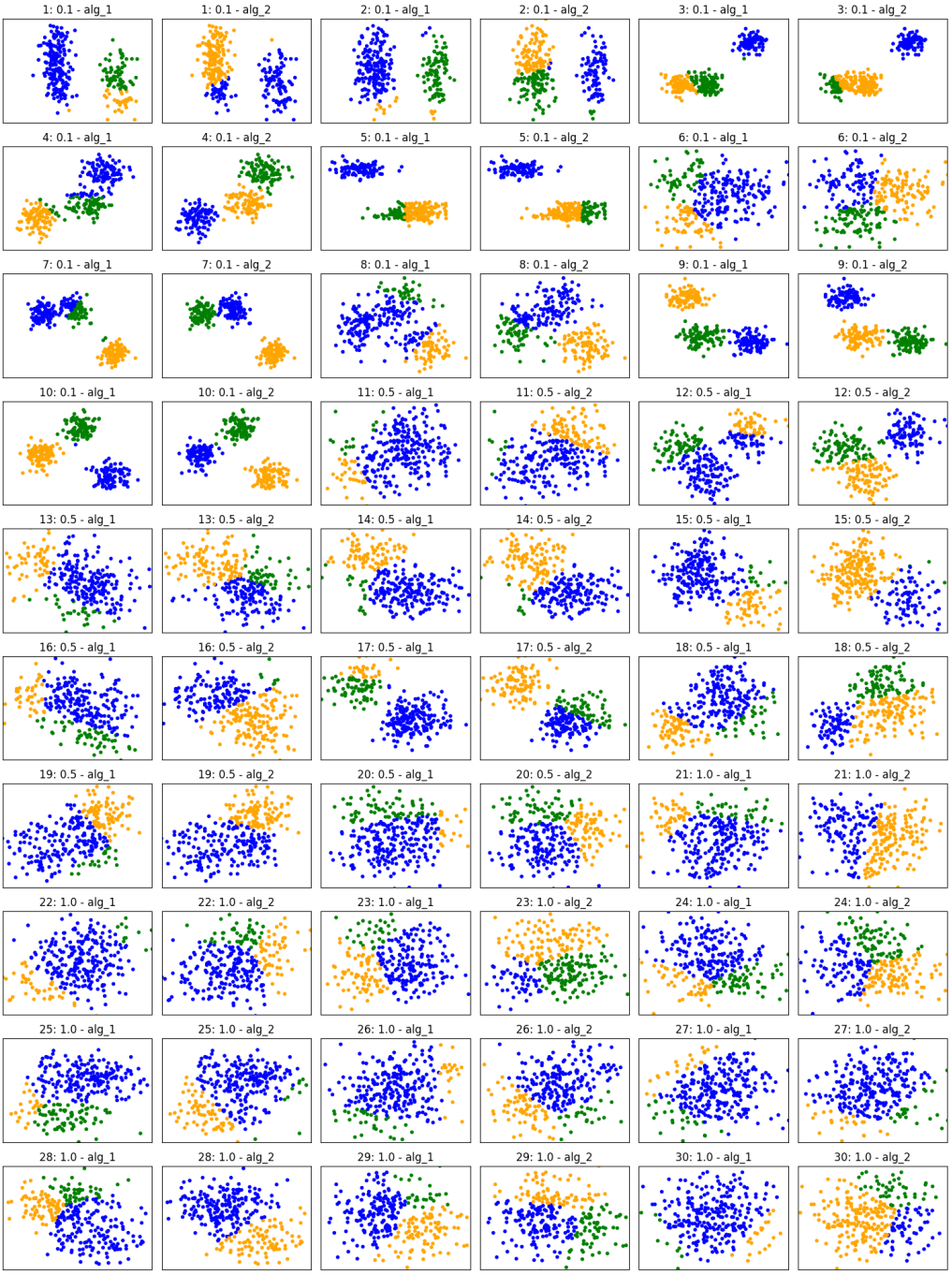

Figura 4. Resultado de uma execução dataset sintético 2