Clusterização Espectral: Implementação e Análise

Matheus Augusto Vargas da Silva[1]

[1]Instituto de Computação – Universidade Federal do Rio de Janeiro (UFRJ)
Caixa Postal 68.530 – 21.941-590 – Rio de Janeiro – RJ – Brazil

matheusavs@dcc.ufrj.br

Resumo. *Esse relatório descreve a implementação de duas versões de um algoritmo de clusterização espectral e demonstra o seu desempenho quando testado em diferentes conjuntos de dados bidimensionais de difícil agrupamento. Além disso, é fornecida uma intuição sobre a ideia inicial do algoritmo, e uma visualização da parte mais intrigante dele, que é a transposição de pontos através dos autovetores de uma representação matricial específica dos dados, chamada de Laplaciana.*

1. Introdução

Clusterizar um conjunto de pontos é designar a cada um desses pontos um rótulo, de tal forma que pontos parecidos (de acordo com alguma métrica escolhida) tenham o mesmo rótulo, e portanto pertençam ao mesmo grupo.

Entretanto, esse é um problema difícil, pois o que define uma boa clusterização é subjetivo, e um mesmo algoritmo pode resultar em uma clusterização considerada satisfatória para um conjunto de dados, mas, para um conjunto diferente, gerar resultados insatisfatórios, como pode ser visto na Figura 1.

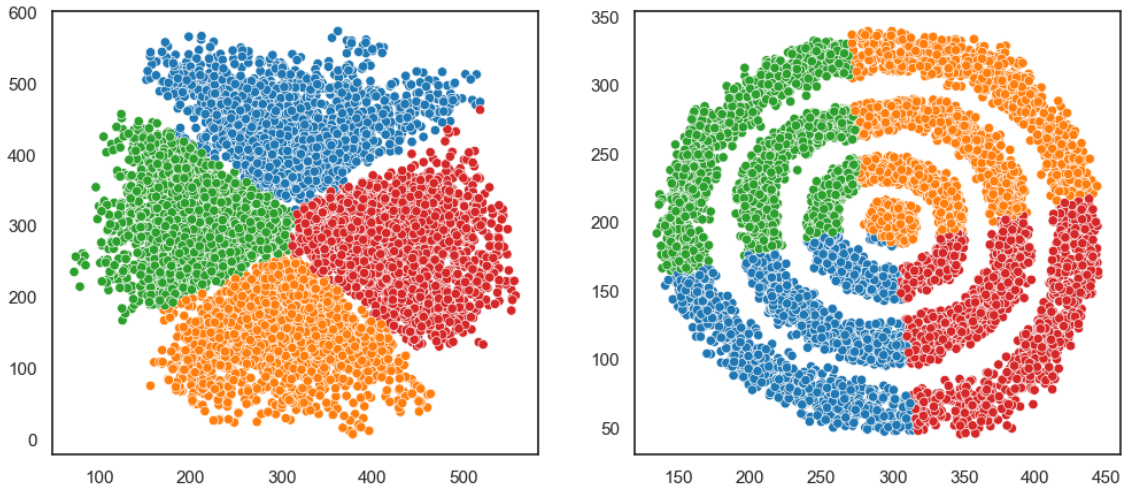

Figura 1. K-Means: clusterização boa (esquerda) e clusterização ruim (direita)

Uma heurística que encontrou muito sucesso é a empregada na clusterização espectral, que utiliza autovetores de uma representação matricial específica dos dados, chamada de Laplaciana, para transpor os pontos de modo a melhor separá-los. O algoritmo de Ng, Jordan e Weiss foi implementado com algumas modificações. Essa implementação faz uso de um parâmetro global de escala σ, que deve ser ajustado manualmente e tem

grande influência sob a qualidade da clusterização. Por isso, também foi implementada uma segunda versão desse algoritmo, com as modificações de Zelnik-manor e Perona, que utiliza escalas locais e automáticas.

Esses algoritmos serão descritos, explicados e por fim serão testados com o dataset do Kaggle [Joonas 2022].

Os testes foram realizados utlizando apenas pontos bidimensionais, para facilitar a verificação do funcionamento do algoritmo. Na Figura 2 estão dispostos os conjuntos de pontos escolhidos para a testagem, coloridos com as clusterizações referência (disponibilizadas pelo autor da base de dados).

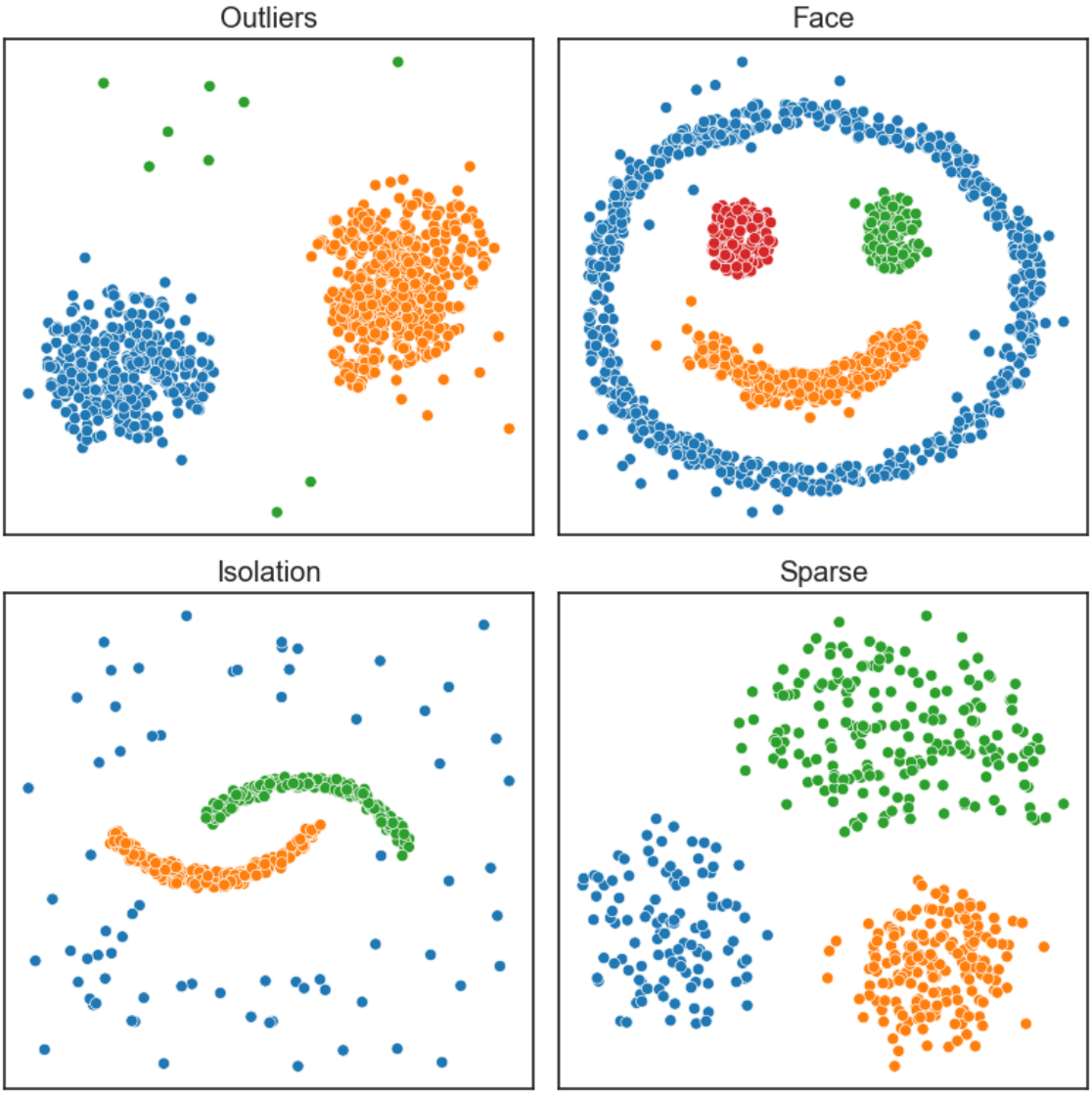

Figura 2. Conjuntos de pontos de teste escolhidos

2. Definições

Antes de explicar o passo-a-passo do algoritmo, é preciso definir duas matrizes utilizadas: a matriz de afinidade e a Laplaciana.

2.1. Matriz de Afinidade

Também chamada de matriz de similaridade, ela é uma representação matricial de um grafo. Se um grafo G tem n vértices, a matriz de afinidade A terá dimensão $n \times n$, com cada entrada A_{ij} representando a similaridade entre os vértices i e j. Essa similaridade pode ser medida de diversas formas, mas a utilizada aqui é o kernel Gaussiano. Isto é, dado um conjunto de pontos $S = \{s_1, \ldots, s_n\}$, a matriz de afinidade é definida como:

$$A_{ij} = \exp(-||s_j - s_i||^2/2\sigma^2)$$

O σ da fórmula acima é o já mencionado fator de escala global. Quanto menor o sigma, mais fracas são as afinidade entre pontos distantes.

2.2. Matriz Laplaciana

A Laplaciana é mais uma representação matricial de um grafo, e é a matriz que terá os autovalores e autovetores calculados. A Laplaciana de um grafo G com n vértices é definida como:

$$L = D - A$$

Onde A é a matriz de afinidade de G e D é a **matriz de grau** de G, que é uma matriz diagonal $n \times n$ onde cada entrada D_{ii} contém o grau do vértice i.

3. Algoritmo: Escala Global

Duas versões do algoritmo foram implementadas. A primeira contém o parâmetro de escala de afinidade σ global, e que deve ser definido pelo usuário. A segunda define σ's locais para cada ponto do conjunto de dados automaticamente. A primeira versão será detalhada nesta seção.

A implementação inicial do algoritmo, baseada no pseudocódigo de [Ng et al. 2001], é descrita a seguir. Ela tem como entrada o conjunto de pontos $S = \{s_1, \ldots, s_n\}$, o número de clusters k e o fator de escala de afinidade σ. A saída é o rótulo de cada um dos pontos, que representa a qual cluster ele pertence.

1. Cria a matriz de afinidade A definida como $A_{ij} = \exp(-||s_j - s_i||^2/2\sigma^2)$ para $i \neq j$, e $A_{ii} = 0$
2. Construa a matriz diagonal D em que o i-ésimo elemento é igual à soma da i-ésima linha de A
3. Construa a matriz Laplaciana $L = D - A$
4. Construa a matriz Laplaciana Normalizada $L = D^{-1/2}LD^{-1/2}$
5. Encontra os k menores autovetores e forma a matriz $V = [v_1, \ldots, v_k]$, onde v_i é um (auto)vetor coluna
6. Normaliza cada linha de V
7. Considera cada linha de V como um ponto em $\mathbb{R}^k$. Agrupa os pontos em k clusters via K-Means
8. Um ponto original s_i pertence ao grupo atribuido ao i-ésimo ponto de V. Retorna os pontos agrupados

Observe que o algoritmo K-Means é utilizado dentro da clusterização espectral. A utilização dos autovetores serve para transpor os pontos originais de modo a obter uma clusterização mais satisfatória via K-Means.

4. Resultados: Escala Global

Para o primeiro teste, foi utilizado $\sigma = 1$ para todos os conjuntos. Na Figura 3 estão os resultados.

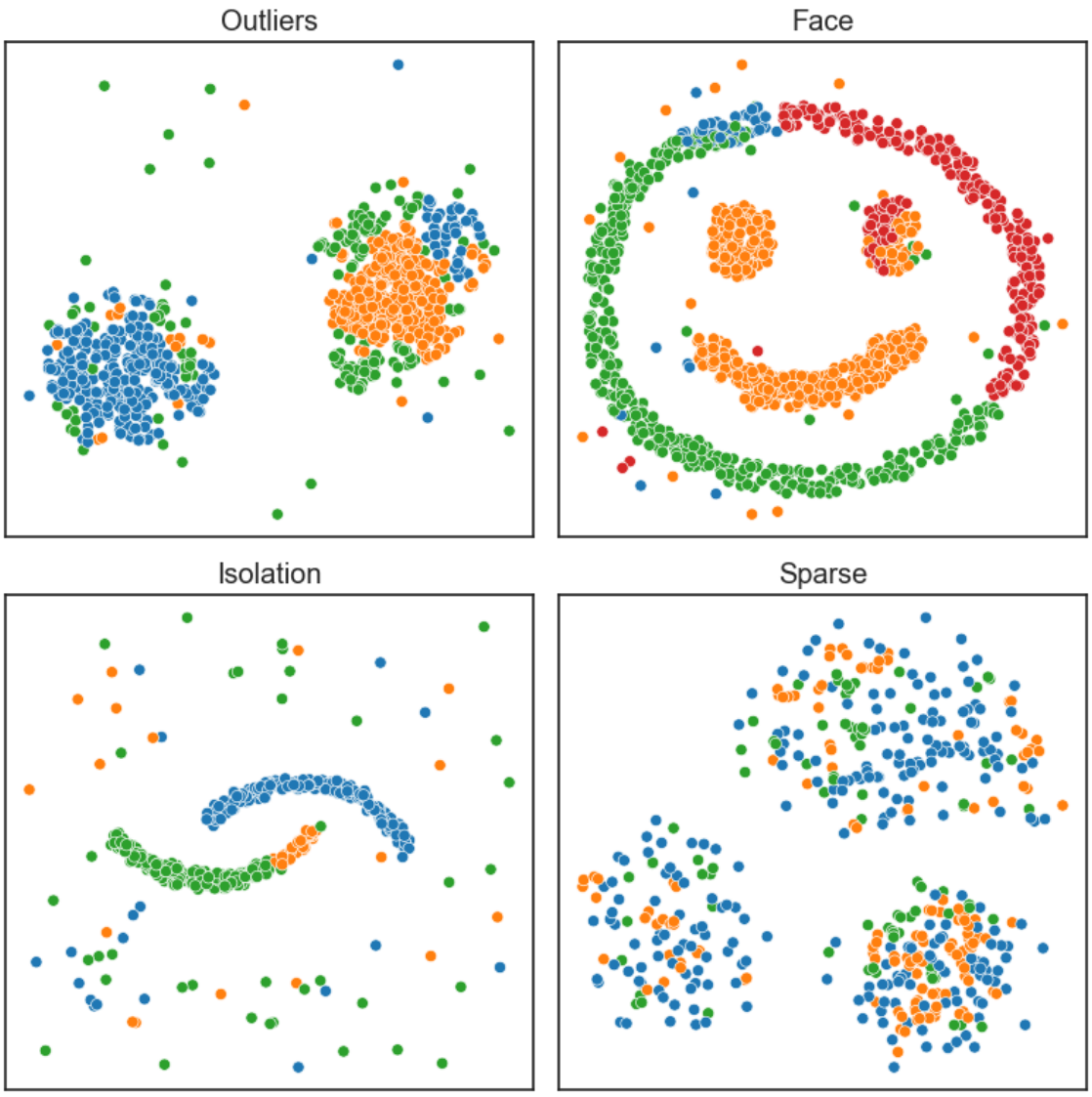

Figura 3. Resultados para $\sigma = 1$

Para o segundo teste, o σ foi ajustado manualmente para cada conjunto de pontos. Os melhores valores encontrados foram de 3.7, 7, 1.7 e 8, respectivamente. Os resultados estão na Figura 4.

Com um σ padrão igual à 1, o algoritmo teve muita dificuldade em agrupar os pontos. Nenhum dos quatro conjuntos foi bem agrupado, inclusive com um resultado surpreendentemente ruim no conjunto Sparse, onde um simples K-Means provavelmente

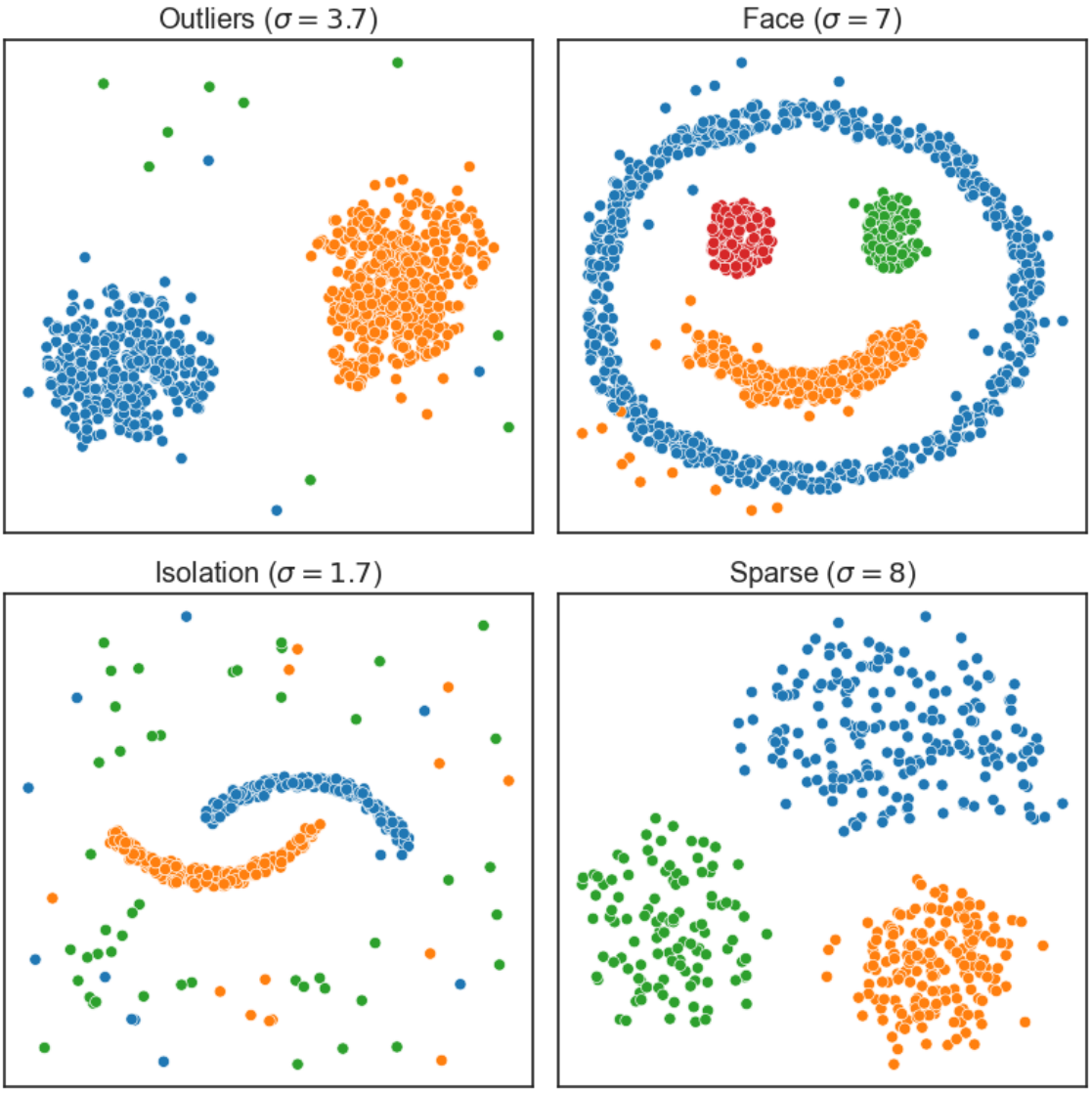

Figura 4. Resultados com σ's ajustados

teria bons resultados. Ao ajustar o σ, é possível ver a real capacidade do algoritmo, que encotrou agrupamentos bem melhores.

Outro fato surpreendente é a diferença de tempo de execução com sigmas diferentes. Com $\sigma = 1$, o algoritmo demorou 21 segundos para encontrar os quatro agrupamentos, enquanto que com os sigmas ajustados o tempo foi de 11 segundos. Isso me leva a acreditar que um σ melhor ajustado ao conjunto de pontos tem um impacto positivo sob o tempo de execução do algoritmo. Não foram investigadas as causas desse fenômeno, mas uma hipótese é a de que um sigma melhor facilitaria o trabalho do K-Means, que necessitaria de menos iterações para convergir.

Com esses resultados, fica claro que a escolha do σ é de alta influência na qualidade dos agrupamentos. Também fica evidente que uma forma de encontrar um σ bom automaticamente é essencial, ainda mais considerando dimensões maiores onde não é possível visualizar a clusterização obtida.

5. Algoritmo: Escala Local

Para produzir os resultados da Figura 4, foram necessárias muitas execuções do algoritmo, em busca de um valor de σ que fosse bom o suficiente. Além disso, um σ global ideal nem sempre existe, já que a escala dos clusters nem sempre é igual (por exemplo, um cluster pode ser denso e outro, esparso) [Zelnik-manor and Perona 2004].

Por conta dessas desvantagens, Zelnik-manor e Perona introduziram modificações no algoritmo. O algoritmo permanece basicamente igual ao descrito na seção 3, a única mudança é na computação do σ.

No lugar do parâmetro σ definido pelo usuário, um parâmetro de escala local σ_i para cada ponto i do grafo é computado. Ele é definido como:

$$\sigma_i = d(s_i, s_K)$$

Ou seja, é a distância para o K-ésimo vizinho mais próximo. Os autores identificaram que um valor de $K = 7$ sempre resultava em bons agrupamentos, mesmo para altas dimensões. Esse, portanto, será o valor utilizado. Dessa forma, a nova matriz de afinidade seria definida como:

$$A_{ij} = \exp\left(\frac{-d^2(s_i, s_j)}{\sigma_i \sigma_j}\right)$$

Entretanto, empiracamente encontrei melhores resultados com uma expressão similar, que será utilizada nos testes:

$$A_{ij} = \exp^2\left(\frac{-d^2(s_i, s_j)}{\sigma_i \sigma_j}\right)$$

Assim, os parâmetros de entrada foram reduzidos para apenas o conjunto de pontos que se deseja agrupar e o número de grupos. A saída ainda é um vetor que indica o grupo ao qual cada um dos pontos pertence.

6. Resultados: Escala Local

Como pode ser visto na Figura 5, essa segunda versão apresenta os melhores resultados vistos neste relatório, no sentido de que é o mais parecido com a clusterização referência da Figura 2.

Apesar de utilizar fatores de escala locais que precisam ser computados, o que consequentemente necessita de mais passos e provavelmente levaria mais tempo, essa versão do algoritmo foi a que teve o menor tempo de execução, com 8 segundos registrados. Isso corrobora com a observação feita na seção 4, de que um ajuste melhor dos parâmetros de escala tem impacto positivo sob o tempo de execução.

Também foi feito um teste adicional, num conjunto com 12800 pontos chamado Waves, para avaliar o tempo de execução com um *dataset* massivo. A clusterização, que ficou perfeita, é apresentada na Figura 6, e foi encontrada em incríveis 51 minutos. Isso demonstra que, apesar de encontrar bons agrupamentos, o algoritmo é custoso.

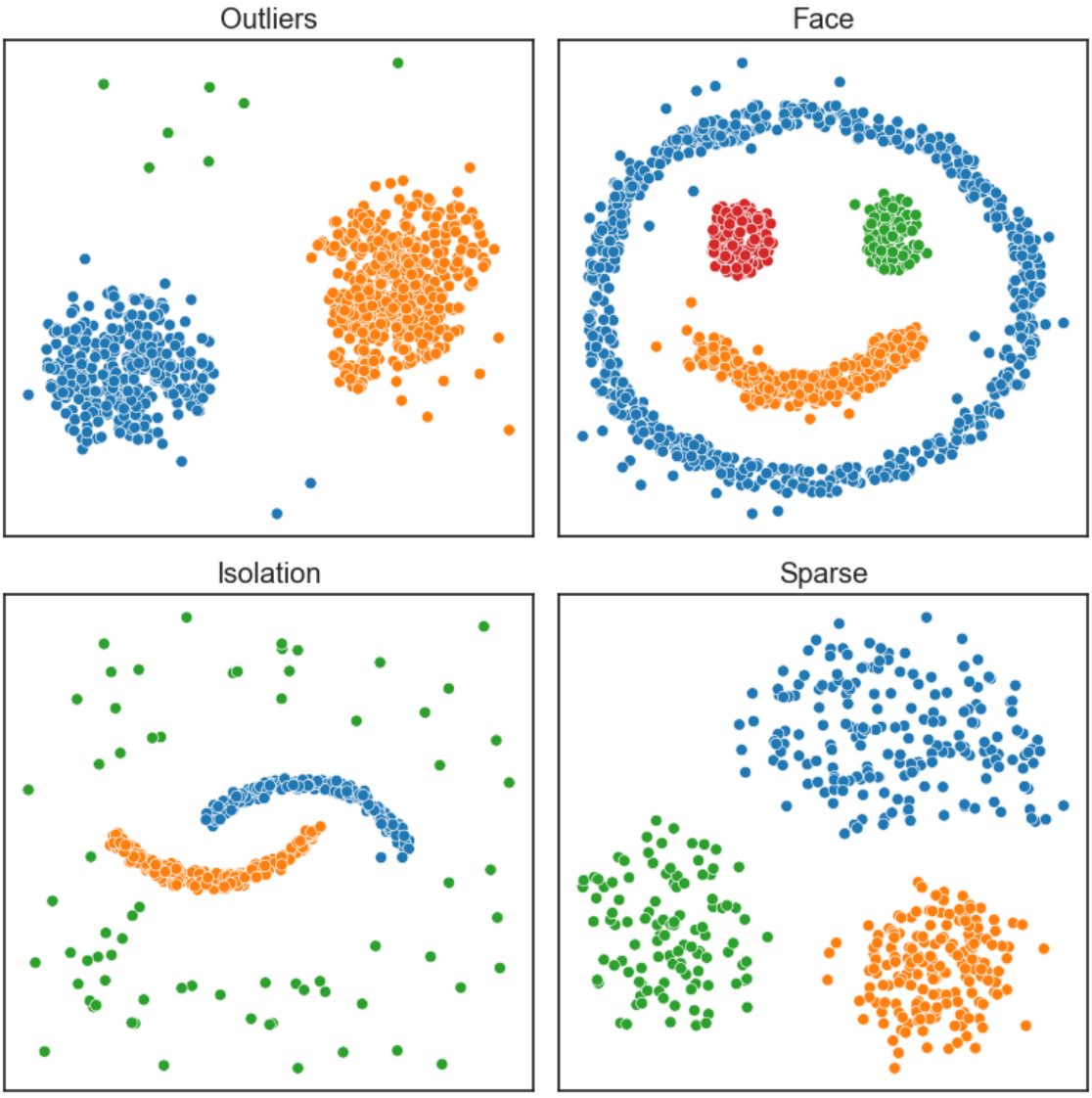

Figura 5. Resultados com σ's locais e automáticos

7. Por Que a Clusterização Espectral Funciona?

Os resultados que podem ser obtidos pela clusterização espectral são impressionantes, mas ao analisar o algoritmo (que é relativamente simples) é difícil determinar o porquê dele obter sucesso. O passo mais "misterioso" do algoritmo é transposição dos pontos originais para as coordenadas obtidas dos autovetores da matriz Laplaciana.

7.1. Derivando a Clusterização Espectral

Uma ideia razoável para formar grupos de pontos é agrupá-los de acordo com sua similaridade, de modo que pontos de um mesmo grupo sejam parecidos e pontos de grupos distintos sejam muito diferentes.

No contexto da matriz Laplaciana e do grafo correspondente à ela, clusterizar seria encontrar uma partição desse grafo tal que arestas entre vértice do mesmo grupo tenham peso baixo e arestas entre grupos diferente tenham peso alto.

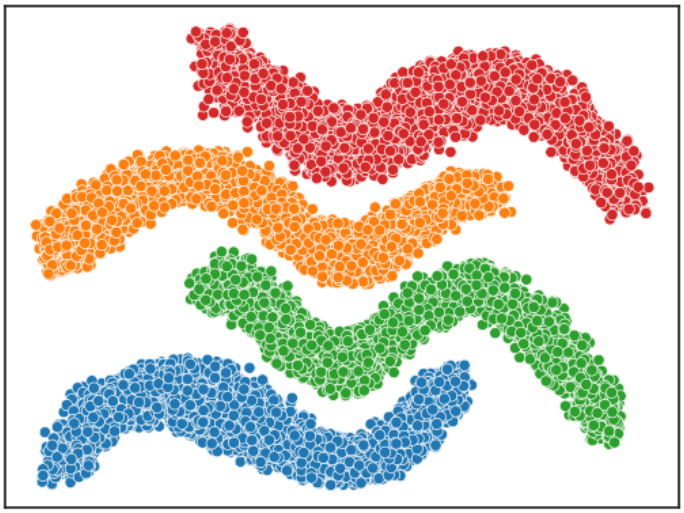

Figura 6. Clusterização espectral em um conjunto de 12800 pontos

Logo, para encontrar uma clusterização, é preciso encontrar uma partição $A_1, \ldots, A_k$ que minimiza

$$cut(A_1, \ldots, A_k) := \frac{1}{2} \sum_{i=1}^{k} \sum_{i \in A, j \in \overline{A}} w_{ij}$$

Onde w_{ij} é o peso da aresta entre os vértices i e j.

O problema é que o *mincut* na prática é insatisfatório, pois em muitos casos ele separa apenas um vértice do resto do grafo [Von Luxburg 2007]. É preciso então de uma função que force as partições a serem razoavelmente grandes. Uma das funções que resolve esse problema é a *Ncut*, que utiliza o tamanho das partições na forma de volume de um subgrafo, em que o volume é o somatório dos pesos internos do subgrafo.

$$Ncut := \sum_{i=1}^{k} \frac{cut(A_i, \overline{A}_i)}{vol(A_i)}$$

Porém, o problema de minimização da função *cut*, que é eficientemente resolvível, deu lugar à minimização da função *Ncut*, que é NP-difícil [Wagner and Wagner 1993].

A clusterização espectral é uma forma de resolver uma versão relaxada desse problema. Para isso, ele precisa ser reescrito. No caso em que $k = 2$, é definido o vetor indicador de cluster

$$f_i = \begin{cases} \sqrt{\frac{\mathrm{vol}(\overline{\mathrm{A}}_\mathrm{i})}{\mathrm{vol}(\mathrm{A}_\mathrm{i})}} & \text{se } v_i \in A \\ -\sqrt{\frac{\mathrm{vol}(\mathrm{A}_\mathrm{i})}{\mathrm{vol}(\overline{\mathrm{A}}_\mathrm{i})}} & \text{se } v_i \in \overline{A} \end{cases}$$

Com isso, é possível demonstrar que $f^T L f = vol(V) Ncut(A, \overline{A})$ utilizando propriedades da Laplaciana [Von Luxburg 2007]. O problema de minimização então se torna

$$\min_{A} f^T L f, \quad \text{sujeito à } f$$

Todas as entradas do vetor f só podem ter dois valores diferentes. O problema pode ser relaxado para permitir que $f_i \in \mathbb{R}^n$. Com isso, pelo teorema de Rayleigh-Ritz, a solusção será dada pelo vetor f que é o autovetor de L correspondente ao segundo menor autovalor de L. Nesse caso de apenas dois grupos, bastaria olhar o sinal de cada entrada do autovetor para definir os grupos (positivos em A, negativos em $\overline{A}$, por exemplo). Para $k > 2$, um processo similar é utilizado para chegar na clusterização espectral, e o K-Means é utilizado por fim, ao invés da análise dos sinais.

7.2. Visualizando a Transposição

Uma forma de entender o impacto da transposição por autovetores da Laplaciana em um conjunto de pontos é visualizando a transformação que ocorre. Na Figura 7, a transposição é demonstrada. Para ficar claro, os pontos estão pintados apenas para ser possível acompanhar o deslocamento deles pelo plano; não houve nenhum agrupamento nesse caso, apenas a transposição.

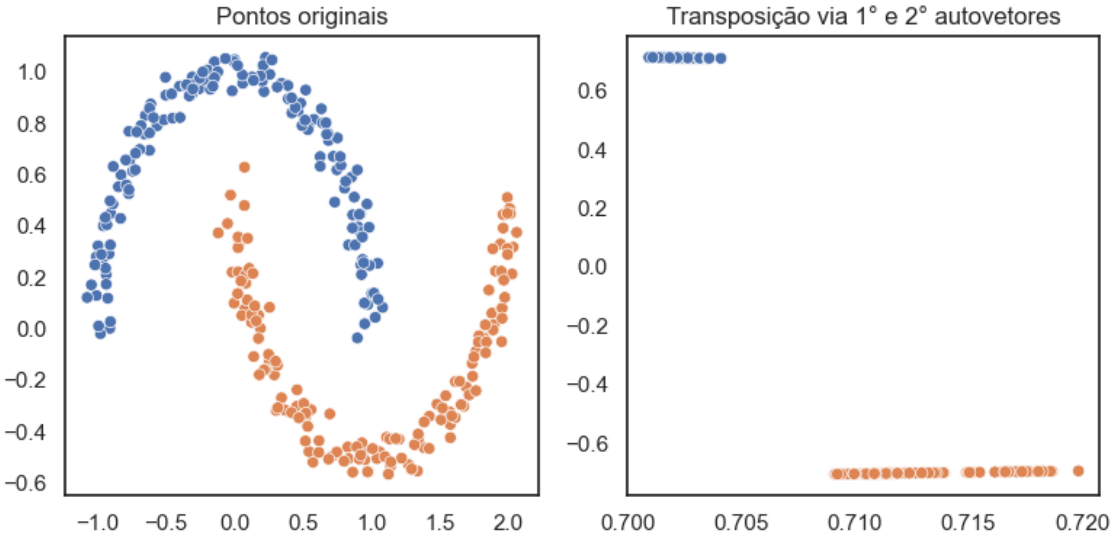

Figura 7. Visualização do efeito dos autovetores da Laplaciana

Observe que, apesar de o primeiro e segundo autovetores terem sido utilizados, o autovetor que de fato diferencia os pontos é o segundo, correspondente ao eixo y. Fica claro também que o algoritmo K-Means encontrará a clusterização desejada nesse caso, pois os autovetores separaram muito bem os pontos.

8. Conclusões

A clusterização parte de uma ideia simples e intuitiva, a de maximizar a afinidade interna dos clusters e minimizar a afinidade entre eles, para um algoritmo que também é simples, possui poucos passos e é fácil de acompanhar, mas que por trás carrega ideias complexas da teoria espectral de grafos.

Um problema desse método é o tempo de execução. Um dos critérios que levei em conta quando escolhi os *datasets* foi o número de pontos (que no caso não passaram de 1300), pois com um número elevado de dados o algoritmo demorava muito. Isso certamente é um problema quando o número de pontos chega as dezenas de milhares, como demonstrado na Figura 6, que possui 12800 pontos e demorou 51 minutos para ser gerada.

De qualquer modo, as ideias por trás do algoritmo se revelam extremamente eficientes na descoberta de grupos nos conjuntos de pontos, como pode ser visto na Figura 5, que agrupa quase que perfeitamente os quatro conjuntos selecionados.

Referências

Joonas (2022). Clustering exercises. `https://www.kaggle.com/datasets/joonasyoon/clustering-exercises`.

Ng, A., Jordan, M., and Weiss, Y. (2001). On Spectral Clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, volume 14. MIT Press.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416.

Wagner, D. and Wagner, F. (1993). Between min cut and graph bisection. In Borzyszkowski, A. M. and Sokołowski, S., editors, *Mathematical Foundations of Computer Science 1993*, Lecture Notes in Computer Science, pages 744–750. Springer.

Zelnik-manor, L. and Perona, P. (2004). Self-Tuning Spectral Clustering. *Advances in Neural Information Processing Systems*, 17.