Notas em Clusterização Espectral

Um Encontro de Teoria dos Grafos e Álgebra Linear

Matheus do Ó Santos Tiburcio
Instituto de Computação – Universidade Federal do Rio de Janeiro
matheusost@ic.ufrj.br

20 de julho de 2024

Resumo

Este texto busca introduzir um método de aprendizado de máquina não-supervisionado chamado clusterização espectral. A ideia é ir desde a especificação do problema reduzido a $k = 2$ até o algoritmo generalizado. No fim aplicações são mostradas e um pouco de teoria sobre um limitante inferior para k é mostrada também.

Sumário

Introdução

Modelos de classificação são amplamente estudados e utilizados nas mais diversas áreas. Seja classificação de doenças, dígitos escritos à mão, determinar qual time de futebol ganharia uma partida ou até para segmentar imagens, separando de algum modo seus elementos. Então não é de se surpreender que se haja tanto esforço em se desenvolver novos modelos de classificação ou aprimorar já existentes. Esses modelos podem, em geral, ser classificados em duas grandes categorias: os *supervisionados* e *não-supervisionados*. A grosso modo, no momento de treinamento, o primeiro possui as classificações das instâncias à priori e talvez o exemplo mais famoso seja o de redes neurais ou de regressões como a linear e logística. Enquanto no segundo não é sabido qual classe cada instância pertence e provavelmente o exemplo mais famoso é o de clusterização, que é exatamente onde a clusterização espectral se encontra.

Clusterização Espectral é um modelo, dentre vários existentes, da família de modelos de clusterização. Mais especificamente, faz parte da família de modelos de clusterização em grafos. Foi originalmente desenvolvido por Jianbo Shi e Jitendra Malik [7] em um contexto de segmentação de imagens. O problema de segmentação de imagens consiste em separar elementos ou objetos distintos em uma imagem. O artigo original traz consigo alguns exemplos desse modelo e recomendo a leitura. Este não é o primeiro modelo em grafos, mas certamente um de extrema importância e com grande impacto e certamente influenciou a área de visão computacional, sendo utilizado até hoje, mais de 20 anos após a publicação do artigo seguindo sendo aprimorado.

O objetivo desse trabalho é inicialmente explicar o que é um modelo de clusterização, assim como mostrar a teoria por detrás desse modelo desde sua construção e modelagem utilizando conceitos de teoria dos grafos até sua transformação e resolução em álgebra linear. Alguma familiaridade com grafos e conceitos de álgebra linear podem ser de ajuda para o entendimento da teoria, mas, na medida do possível, os conceitos são explicados ao decorrer do caminho conforme são utilizados. Em seguida algumas aplicações reais com o modelo são mostradas. É um modelo riquíssimo em teoria e que utiliza de ferramentas de inúmeras áreas da matemática e computação e espero conseguir mostrar ao menos uma pequena parte disto neste trabalho.

Como mencionado na seção anterior, clusterização é um nome dado à uma família de modelos não-supervisionados e tem como objetivo agrupar um conjunto de elementos em uma dada quantidade de grupos ou *clusters*. Certamente um método famoso dessa família é o *kmeans*, que a partir de um chute de pontos centrais de cada cluster os atualiza iterativamente e classifica cada instância como pertencente ao cluster cujo ponto central esteja mais próximo.

Em especial o $k$means é limitados a métricas lineares, isto é, caso os dados estejam separados por uma curva não-linear, o método de $k$means passa a ter um pouco de dificuldade. Na parte de aplicações isso será melhor visto. Uma vantagem da clusterização espectral, e isso será melhor trabalhado, é o fato de padrões não-lineares também serem reconhecíveis usando o método.

Especificação Para o Caso $k = 2$

A clusterização espectral é um método de clusterização em grafos. Os grafos que são tratados são grafos não-direcionados cujas arestas representam o quão similar duas instâncias são, sendo assim quanto maior o peso da aresta mais parecidos são as instâncias conectadas. Um detalhe importante também é que estas similaridades são valores não-negativos, portanto o grafo possui somente arestas não-negativas representando o grau de similaridade entre pontos. O argumento para ser um grafo não-direcionado é que se um ponto x se parece w com outro ponto y, é natural pensar que y se parece exatamente w com x também.

A técnica consiste em, dado um grafo, achar o melhor corte neste grafo que separe "bem"os pontos em k clusters. No linguajar de teoria dos grafos, o método busca encontrar uma k-partição deste grafo que separe "bem"os pontos.

Definição Dado um grafo $G = (V, E)$, uma **k-partição** de G é uma separação do conjunto de vértices V de G em k partes disjuntas, isto é, $(A_1, A_2, ..., A_k)$ é uma k-partição de G se $A_1 \cup A_2 \cup .... \cup A_k = V$ e $A_i \cap A_j = \emptyset$ para todo par de conjuntos distintos. Essa definição garante que todo vértice pertença a algum cluster e que um vértice pertença a somente um cluster. Exatamente o que se deseja.

Separar "bem"é uma ideia muito vaga e pode variar de método para método, a clusterização espectral parte de duas premissas para definir uma boa partição. Partindo do pressuposto que o grafo passado possui como arestas o grau de similaridade entre pontos, seria interessante duas coisas:

- Minimizar a soma do peso das arestas entre elementos de diferentes partições
- Maximizar a soma do peso das arestas entre elementos de uma mesma partição

O primeiro item garante que pontos de partições distintas sejam bem dissimilares, os mantendo afastados uns dos outros. O segundo garante que elementos de uma mesma partição, ou cluster, sejam bastante similares, os mantendo próximos uns dos outros. A clusterização espectral então assume que minimizar o primeiro caso e maximizar o segundo seria uma boa forma de encotrar um bom corte no grafo.

O algoritmo então tem como entrada a matriz de adjacências de um grafo não-direcionado G e como saída as k-partições desse grafo.

A teoria aqui será desenvolvida para $k = 2$, ou seja, será desejado uma bipartição (A, B) do grafo que minimize e maximize os itens 1 e 2 respectivamente. Mas certamente pode ser expandinda, assim como já foi, para $k \geq 2$ arbitrário [5].

Minimização entre Elementos de Conjuntos Distintos

Como discutido acima, ter arestas de baixo peso entre vértices de partições distintas é desejável e categoriza bem uma partição boa. Para conseguir isto uma definição de teoria dos grafos será usada, o corte ou *cut*. Então dada uma bipartição (A, B) de G, o *cut* entre A e B é definido como

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v) \tag{1}$$

Onde $w(u, v)$ indica o peso da aresta entre o vértice u e v no grafo. Note então que o *cut* é exatamente a soma das arestas **entre** os grupos, exatamente o que quero minimizar! A figura 1 foi retirada diretamente do artigo original [7]. Considere que os vértices à esquerda da linha tracejada no meio da imagem sejam da partição A e os à direita da partição B, o $cut(A, B)$ soma o peso das arestas que **cruzam** a linha tracejada.

Porém note que, assumindo essa fórmula como a desejada a se minimizar, algumas situações ruins podem ocorrer. Na figura 1 mesmo, os cortes ótimos retornados seriam os vértices isolados à direita, pois no caso de possuírem poucas arestas, o somatório da definição de $cut(A, B)$ seria pequeno. Seria interessante então **normalizar** essa quantificação, podendo assim priorizar mais as partições maiores. Para isso a definição de associação será feita. Dado um subconjunto A de V, a associação entre A e V é dada por

$$assoc(A, V) = \sum_{u \in A, v \in V} w(u, v) \tag{2}$$

Supondo que A é o mesmo conjunto de vértices discutido acima e analisando a figura 1, $assoc(A, V)$ agora contabiliza todas as arestas que **cruzam** a linha tracejada e todas as arestas **dentro** do conjunto A. A normalização desejada então seria

$$\frac{\text{arestas que } \textbf{cruzam} \text{ a linha tracejada}}{\text{arestas que } \textbf{cruzam} \text{ a linha tracejada + arestas } \textbf{dentro} \text{ do conjunto}} \tag{3}$$

Isso responde a pergunta de "qual a proporção da soma das arestas do conjunto A que, de fato, cruzam a linha tracejada?". Agora pegue os vértices isolados de anteriormente, a fração daria

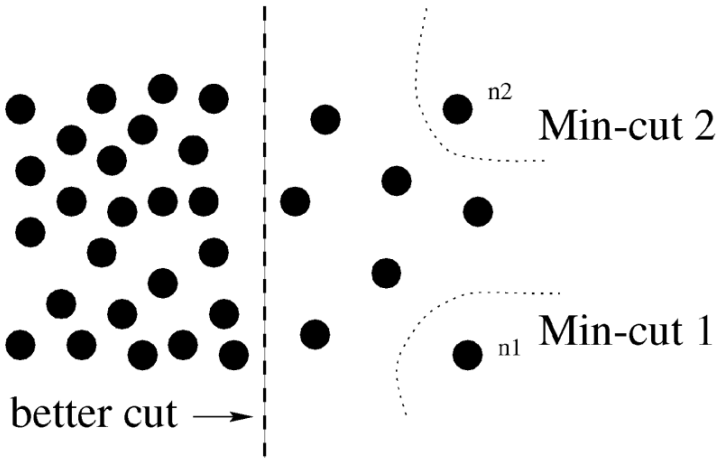

Figura 1: Grafo de exemplo.

exatamente 1, visto que não há arestas dentro do conjunto. De modo geral, para um conjunto grande muitas arestas internas ocorreriam, o denominador da razão cresceria e o número como um todo diminuiria. Usando essa razão, o **corte normalizado**, ou ***normalized cut***, $Ncut$ entre A e B é definido como

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(B,A)}{assoc(B,V)} \tag{4}$$

E é esta fórmula que será minimizada.

Maximização entre Elementos de Mesmo Conjunto

Definida a minimização, resta definir a maximização. Usando a definição de *assoc* já é possível ter uma ideia do que seria maximizar, verificando a associação de A consigo mesmo.

$$assoc(A,A) = \sum_{u \in A, v \in A} w(u,v) \tag{5}$$

Isso é exatamente a soma dos pesos das arestas **dentro** do conjunto A. Mas repare que, semelhante ao problema da minimização, deixar isso não-normalizado pode causar problemas. Grandes conjuntos com arestas internas de alto peso, mas também com arestas externas de alto peso poderiam ocorrer. Novamente a ideia de proporção pode ser útil aqui. Uma razão plausível seria

$$\frac{\text{arestas } \textbf{dentro} \text{ do conjunto}}{\text{arestas que } \textbf{cruzam} \text{ a linha tracejada + arestas } \textbf{dentro} \text{ do conjunto}} \tag{6}$$

Assim muitas arestas que **cruzam** a linha tracejada implicaria em um denominador maior e, por consequência, um número menor. Mas agora queremos maximizar! Então faz sentido diminuir o peso das arestas externas. Relembre que $assoc(A,V)$ contabiliza **todas** as arestas de A, internas e externas. A associação normalizada, $Nassoc$, de A e B pode ser definida como

$$Nassoc(A,B) = \frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)} \tag{7}$$

Que expressa exatamente a razão discutida. E, de fato, é isto que se desejará maximizar.

Minimizar é igual a Maximizar

Agora tanto a função a ser minimizada quanto a função a ser maximizada estão definidas. Mas minimizar uma e maximizar outra ao mesmo tempo parece ser um problema um pouco difícil. Ocorre que, minimizando uma se maximiza a outra e realmente isso é um fato intuitivo ou que "deveria ser verdade"dado o modo como estas funções foram construídas.

Primeiro, colocarei as duas razões (3) e (6) que foram utilizadas para a minimização e maximização, respectivamente.

$$\frac{\text{arestas que } \textbf{cruzam} \text{ a linha tracejada}}{\text{arestas que } \textbf{cruzam} \text{ a linha tracejada + arestas } \textbf{dentro} \text{ do conjunto}}$$

$$\frac{\text{arestas } \textbf{dentro} \text{ do conjunto}}{\text{arestas que } \textbf{cruzam} \text{ a linha tracejada + arestas } \textbf{dentro} \text{ do conjunto}}$$

Note que maximizar as arestas dentro do conjunto minimiza a primeira razão e minimizar as arestas que cruzam a linha tracejada maximizam a segunda. Não somente isso, mas a soma das razões dá exatamente 1! Como, de fato, deveria ser, visto que uma contabiliza a proporção de peso das arestas externas e outra a proporção de peso das arestas internas. A primeira pode ser reescrita então como

$$1 - \frac{\text{arestas } \textbf{dentro} \text{ do conjunto}}{\text{arestas que } \textbf{cruzam} \text{ a linha tracejada + arestas } \textbf{dentro} \text{ do conjunto}} \tag{8}$$

Como se deseja **maximizar** esta razão que aparece com um sinal de subtração acima, segue que maximizá-la minimiza a fórmula como um todo. Mas note que isso poderia ser alcançado manipulando a definição de $Ncut$ também

$$\begin{aligned}
& \tfrac{cut(A,B)}{assoc(A,V)} + \tfrac{cut(B,A)}{assoc(B,V)} \\
= \quad & \{ \text{ reescrita de } cut(A,B) \ \} \\
& \tfrac{assoc(A,V)-assoc(A,A)}{assoc(A,V)} + \tfrac{assoc(B,V)-assoc(B,B)}{assoc(B,V)} \\
= \quad & \{ \text{ álgebra } \} \\
& 2 - \left(\tfrac{assoc(A,A)}{assoc(A,V)} + \tfrac{assoc(B,B)}{assoc(B,V)} \right) \\
= \quad & \{ \text{ definição de } Nassoc \ \} \\
& 2 - Nassoc(A,B)
\end{aligned}$$

O mesmo fato aparece. Se deseja minimizar o $Ncut$ e, dada a forma final obtida, isso é obtido maximizando a $Nassoc$, que também é desejado. Como minimizar o $Ncut$ ou maximizar a $Nassoc$ traz o mesmo resultado, a escolha feita para o restante do texto será a de minimizar o $Ncut$.

De Teoria dos grafos para Álgebra Linear

Até agora toda a discussão foi feita em alto nível utilizando conceitos de Teoria dos Grafos, mas a princípio não fica claro **como** minimizar o $Ncut$. Para facilitar um pouco esse processo, traduzirei o problema, em linguagem de Teoria dos Grafos, para a linguagem de Álgebra Linear. Antes de isso ser feito, algumas pequenas definições precisam ser feitas.

A primeira é a de matriz de adjacências. Seja G um grafo de n vérices, sua **matriz de adjacências** será uma matriz $W_{n\times n}$ cujas entradas w_{ij} serão exatamente $w(v_i, v_j)$, o peso da aresta entre o vértice i e o vértice j. Adicional a isso, defino o grau d_i de um vértice i como

$$d(u) = \sum_{v \in V} w(u, v) \tag{9}$$

Ou seja, a soma do peso de todas as arestas que saem de i. Todos os graus dos vértices ficarão contidos em uma matriz diagonal D cuja definição é

$$d_{ij} = \begin{cases} d(v_i), \text{ se } i = j, \\ 0, \text{caso contrário} \end{cases} \tag{10}$$

Uma matriz A é **simétrica** se $A^T = A$. Logo, por definição, D é simétrica e pelo fato de G ser não-direcionado W também é, pois $w_{ij} = w_{ji}$ para quaisquer par de vértices.

Uma observação é que $assoc(A, V)$ contabiliza a soma do peso das arestas de todos os vértices pertencentes a A, isso é exatamente $\sum_{u \in A} d(u)$. Para simplificações futuras, defino o volume de A, $vol(A)$, como sendo $assoc(A, V)$. Com isso $Ncut$ fica como

$$Ncut(A, B) = \frac{cut(A, B)}{vol(A)} + \frac{cut(B, A)}{vol(B)} = \frac{cut(A, B)(vol(B) + vol(A))}{vol(A)vol(B)} \tag{11}$$

A última definição necessária é a de um **vetor indicador de clusters** f de dimensão $n \times 1$ cuja definição é

$$f_i = \begin{cases} \sqrt{\frac{vol(B)}{vol(A)}}, \text{ se o vértice } i \text{ pertencer a } A \\ -\sqrt{\frac{vol(A)}{vol(B)}}, \text{ se o vértice } i \text{ pertencer a } B \end{cases} \tag{12}$$

Um leve detalhe de que não há nada de especial neste vetor indicador e, de fato, quaisquer dois valores distintos poderiam ser escolhidos para representar se um vértice i está em A ou B. Esses dois valores foram escolhidos somente para facilitar os cálculos que se seguirão.

A ideia agora é, usando o que foi definido, chegar na definição de $Ncut$. Isso pode ser obtido calculando o valor de $f^T Df$ e $f^T(D - W)f$. Para $f^T Df$

$$\begin{aligned}
& f^T Df \\
= & \quad \{ \text{ expansão em somatório } \} \\
& \sum_{i=1}^{n} f_i d_{ii} f_i \\
= & \quad \{ \text{ separação em vértices de } A \text{ e vértices de } B \} \\
& \sum_{i \in A} d_{ii} f_i^2 + \sum_{j \in B} d_{jj} f_j^2 \\
= & \quad \{ \text{ definição de } f \} \\
& \sum_{i \in A} d_{ii} \left(\sqrt{\frac{vol(B)}{vol(A)}} \right)^2 + \sum_{j \in B} d_{jj} \left(-\sqrt{\frac{vol(A)}{vol(B)}} \right)^2 \\
= & \quad \{ \text{ definição de } vol \text{ e álgebra } \} \\
& vol(A) \frac{vol(B)}{vol(A)} + vol(B) \frac{vol(A)}{vol(B)} \\
= & \quad \{ \text{ álgebra } \} \\
& vol(B) + vol(A)
\end{aligned}$$

Agora a derivação de $f^T(D - W)f$

$$\begin{aligned}
& f^T(D - W)f \\
= & \quad \{ \text{ distributividade } \} \\
& f^T Df - f^T Wf \\
= & \quad \{ \text{ expansão em somatório } \} \\
& \sum_{i=1}^{n} d_{ii} f_i^2 - \sum_{i=1}^{n} \sum_{j=1}^{n} f_i f_j w_{ij} \\
= & \quad \{ \text{ reescrita } \} \\
& \frac{1}{2} \left(\sum_{i=1}^{n} d_{ii} f_i^2 - 2 \sum_{i=1}^{n} \sum_{j=1}^{n} f_i f_j w_{ij} + \sum_{j=1}^{n} d_{jj} f_j^2 \right) \\
= & \quad \{ \text{ álgebra } \} \\
& \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} (f_i - f_j)^2 \\
= & \quad \{ \text{ separação de casos em } f_i = f_j \text{ e } f_i \neq f_j \} \\
& \frac{1}{2} \left(\sum_{f_i = f_j} w_{ij} (f_i - f_j)^2 + \sum_{f_i \neq f_j} w_{ij} (f_i - f_j)^2 \right) \\
= & \quad \{ \text{ simplificação } \} \\
& \frac{1}{2} \sum_{f_i \neq f_j} w_{ij} (f_i - f_j)^2 \\
= & \quad \{ \text{ distributividade em } (f_i - f_j)^2 \text{ e definição de } f \} \\
& \frac{1}{2} \left(\sqrt{\frac{vol(B)}{vol(A)}} + \sqrt{\frac{vol(A)}{vol(B)}} \right)^2 \sum_{f_i \neq f_j} w_{ij} \\
= & \quad \{ \text{ definição de } cut \}
\end{aligned}$$

$$
\begin{aligned}
& \tfrac{1}{2} cut(A,B) \left(\sqrt{\tfrac{vol(B)}{vol(A)}} + \sqrt{\tfrac{vol(A)}{vol(B)}} \right)^2 \\
= & \quad \{ \text{ álgebra } \} \\
& \tfrac{1}{2} cut(A,B) \tfrac{(vol(B)+vol(A))^2}{vol(A)vol(B)}
\end{aligned}
$$

Com ambas as derivações feitas, o $Ncut$ pode ser mais uma vez reescrito.

$$Ncut(A,B) = \frac{cut(A,B)(vol(B)+vol(A))}{vol(A)vol(B)} = 2\frac{f^T(D-W)f}{f^TDf} \tag{13}$$

Um último detalhe, que será importante futuramente, é o fato de $f^TD\mathbf{1} = 0$.

$$
\begin{aligned}
& f^TD\mathbf{1} \\
= & \quad \{ \text{ expansão em somatório } \} \\
& \textstyle\sum_{i=1}^{n} d_{ii}f_i \\
= & \quad \{ \text{ separação em vértices de } A \text{ e vértices de } B \ \} \\
& \textstyle\sum_{i\in A} d_{ii}f_i + \sum_{j\in B} d_{jj}f_j \\
= & \quad \{ \text{ definição de } vol \text{ e } f \ \} \\
& vol(A)\sqrt{\tfrac{vol(B)}{vol(A)}} - vol(B)\sqrt{\tfrac{vol(A)}{vol(B)}} \\
= & \quad \{ \text{ álgebra } \} \\
& \sqrt{vol(A)vol(B)} - \sqrt{vol(B)vol(A)} \\
= & \quad \{ \text{ álgebra } \} \\
& 0
\end{aligned}
$$

Onde $\mathbf{1}$ é um vetor com todas as entradas são iguais a 1. Isso será incorporado como restrição do problema de minimização. Com tudo isso discutido, finalmente o problema de otimização pode ser completamente descrito.

Dado um grafo não-direcionado G, quero uma bipartição (A,B) que minimize o corte mínimo sujeito à f definido em (12) e à $f^TD\mathbf{1} = 0$. Matematicamente:

$$\tilde{f} = arg\min_f \frac{f^T(D-W)f}{f^TDf} \tag{14}$$

sujeito à

$$f_i = \begin{cases} \sqrt{\tfrac{vol(B)}{vol(A)}}, \text{ se o vértice } i \text{ pertencer a } A \\ -\sqrt{\tfrac{vol(A)}{vol(B)}}, \text{ se o vértice } i \text{ pertencer a } B \end{cases}$$

$$f^TD\mathbf{1} = 0$$

Como se deseja o **argumento mínimo**, 2 pode ser retirado da função. Porém note que devido à definição de f encontrar a solução **exata** deste problema se torna um problema NP-Completo. Todas as entradas de f podem assumir um valor entre dois possíveis, isso daria exatamente 2^n possibilidades! Uma prova mais detalhada pode ser encontrada no artigo original [7]. Por sorte relaxando f para assumir quaisquer valores reais em suas entradas é possível encontrar uma solução em tempo polinomial. Reforço que a solução com f relaxado é uma solução **aproximada** para o problema original. Reescrevendo o problema de otimização se obtém

$$\tilde{f} = arg\min_f \frac{f^T(D-W)f}{f^TDf}$$

sujeito à

$$f^TD\mathbf{1} = 0$$

Resolver esse problema não parece tão direto devido a matriz D no denominador, a fim de removê-la uma reescrita será feita. Defina g como $D^{\frac{1}{2}}f$, a minimização então se torna

$$\tilde{f} = arg\min_g \frac{g^TD^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}}g}{g^Tg} \tag{15}$$

A próxima seção detalha melhor como resolver esse problema.

Otimização

Com o problema relaxado, uma solução aproximada é possível de ser obtida. Apesar disso, a solução dessa minimização não é tão direta e um pouco de teoria vai ser necessária para que eu consiga solucioná-la.

A Laplaciana e Matrizes Simétricas Positivas Semi-Definidas

Para de fato conseguir resolver esse problema de otimização, precisarei de alguns lemas e propriedades. Deixei aqui algumas provas, mas nem tudo foi provado.

A primeira coisa que posso usar e irei provar é um fato importante sobre minimização.

Lema 1 $y = min_x \frac{||Ax||^2}{||x||^2} \Leftrightarrow y = min_{||x||^2=1}||Ax||^2$

A direção interessante aqui é $\Rightarrow$ cuja prova é esboçada abaixo.

Reescreva x como $||x||u$, onde $||u|| = 1$. Pela linearidade de A, tem-se que $A(k \cdot x) = k \cdot Ax$, segue que $Ax = ||x||Au$ e $||Ax|| = ||(||x||)Au|| = ||x||||Au||$. Reescrevendo x, $\frac{||Ax||^2}{||x||^2}$ se torna $\frac{||x||^2||Au||^2}{||x||^2} = ||Au||^2$. Mas por definição $||u||^2 = 1$. Segue que $min_x \frac{||Ax||^2}{||x||^2} = min_{||x||^2=1}||Ax||^2$. □

O próximo passo é mostrar que $D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$ pode ser quebrada em uma forma Z^TZ, pois note que com isso sendo possível $||(D - W)D^{-\frac{1}{2}}g||$ assumirá uma forma parecida com a do lema provado acima.

A matriz $(D-W)$ é a famosa Laplaciana de um grafo e a denotarei por L. Uma de suas propriedades é o fato de ser simétrica positiva semi-definida, isto é, existe uma matriz Z tal que $L = Z^TZ$. Outra caracterização de uma matriz simétrica positiva semi-definida é

$$x^TLx \geq 0 \tag{16}$$

Para todo vetor x.

Lema 2 Seja A é matriz simétrica com todos seus autovalores não-negativos, então A é simétrica positiva semi-definida.

Usando a definição (16), segue que

$$\begin{aligned}
& x^TAx \geq 0 \\
= & \quad \{\ SVD \text{ de } A\ \} \\
& x^TVDV^Tx \geq 0 \\
= & \quad \{\ \text{definição de tranposta}\ \} \\
& (V^Tx)^TDV^Tx \geq 0 \\
= & \quad \{\ \text{reescrita de } V^Tx \text{ para } y\ \} \\
& y^TDy \geq 0 \\
= & \quad \{\ \text{expansão em somatório}\ \} \\
& \sum_{i=1}^n d_{ii}y_i^2 \geq 0
\end{aligned}$$

Dado o fato de que todos os autovalores de A são não-negativos, então todos os elementos da diagonal de D são também não-negativos. Por qualquer real ao quadrado ser não-negativo, segue que $\sum_{i=1}^n d_{ii}y_i^2 \geq 0$. □

Lema 3 Seja A e B matrizes simétricas positivas semi-definidas, então existe matriz $A^{\frac{1}{2}}$ tal que $A = A^{\frac{1}{2}}A^{\frac{1}{2}}$ e $A^{\frac{1}{2}}BA^{\frac{1}{2}}$ é simétrica positiva semi-definida.

A primeira parte pode ser provada via SVD. Por A ser simétrica, pode ser diagonalizada em VDV^T, logo defina $A^{\frac{1}{2}}$ como $VD^{\frac{1}{2}}V^T$, o que é possível pois todos os seus autovalores são não-negativos pelo **lema 2**. Segue que

$$A = A^{\frac{1}{2}}A^{\frac{1}{2}} = VD^{\frac{1}{2}}V^TVD^{\frac{1}{2}}V^T = VDV^T$$

Resta mostrar que $A^{\frac{1}{2}}BA^{\frac{1}{2}}$ é simétrica positiva semi-definida. Usando a definição (16) se tem que $x^TBx \geq 0$ para todo vetor x. Segue que

$$\begin{aligned} & x^TA^{\frac{1}{2}}BA^{\frac{1}{2}}x \geq 0 \\ = & \quad \{ \text{ definição de transposta e simetria de } A^{\frac{1}{2}} \ \} \\ & (A^{\frac{1}{2}}x)^TBA^{\frac{1}{2}}x \geq 0 \\ = & \quad \{ \text{ reescrita de } A^{\frac{1}{2}}x \text{ para } y \ \} \\ & y^TBy \geq 0 \end{aligned}$$

Por B ser positiva simétrica semi-definida a última inequação vale. □

A Junção das Peças e Solução da Minimização

Pelo **lema 2** D é positiva semi-definida. E pelo **lema 3** e o fato de L ser simétrica positiva semi-definida é possível concluir que a matriz $D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$ é também simétrica positiva semi-definida, ou seja, existe Z tal que $D^{-\frac{1}{2}}LD^{-\frac{1}{2}} = Z^TZ$, portanto pelo **lema 1**

$$\tilde{f} = arg\min_g \frac{||Zg||^2}{||g||^2} \Rightarrow \tilde{f} = arg\min_{||g||^2=1} ||Zg||^2 = g^TZ^TZg = g^TD^{-\frac{1}{2}}LD^{-\frac{1}{2}}g$$

O problema de minimização reescrito fica, portanto

$$\tilde{f} = arg \min_{||g||=1} g^TD^{-\frac{1}{2}}LD^{-\frac{1}{2}}g \tag{17}$$

sujeito à

$$g^TD^{\frac{1}{2}}\mathbf{1} = 0$$

É possível resolver esse problema usando ideia de multiplicadores de Lagrange. Tratando $||g|| = 1$ como a restrição, o problema pode ser reescrito como

$$\tilde{f} = arg\min_{g\neq 0} g^TD^{-\frac{1}{2}}LD^{-\frac{1}{2}}g + \lambda(||g|| - 1) \tag{18}$$

Repare que esta forma é equivalente à anterior pelo fato de o lado direito da soma ser minimizado quando $||g|| = 1$. A restrição de $g \neq 0$ foi adicionada, mas não altera o problema, pois na modelagem original se tinha $||g|| = 1$, condição que o vetor nulo não satisfaz. Derivando em função de g e igualando a 0, se chega na forma

$$D^{-\frac{1}{2}}LD^{-\frac{1}{2}}g + \lambda g = 0$$

$$D^{-\frac{1}{2}}LD^{-\frac{1}{2}}g = -\lambda g$$

Com isso, g é autovetor de $D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$. Um fato interessante é que $D^{\frac{1}{2}}\mathbf{1}$ é autovetor de $D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$.

$$\begin{aligned} & D^{-\frac{1}{2}}LD^{-\frac{1}{2}}D^{\frac{1}{2}}\mathbf{1} \\ = & \quad \{ \text{ produto de matrizes } \} \\ & D^{-\frac{1}{2}}L\mathbf{1} \\ = & \quad \{ \text{ definição de } L \ \} \\ & D^{-\frac{1}{2}}(D - W)\mathbf{1} \\ = & \quad \{ \text{ distributividade } \} \\ & D^{\frac{1}{2}}\mathbf{1} - D^{-\frac{1}{2}}W\mathbf{1} \\ = & \quad \{ \text{ soma da linha } i \text{ de } W = \text{grau de } v_i \ \} \end{aligned}$$

$$
\begin{aligned}
& D^{\frac{1}{2}}\mathbf{1} - D^{-\frac{1}{2}}D\mathbf{1} \\
= & \quad \{ \text{ álgebra } \} \\
& 0
\end{aligned}
$$

Não somente isso, mas por $D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$ ser simétrica positiva semi-definida, todos seus autovalores são não-negativos e, portanto, $D^{\frac{1}{2}}\mathbf{1}$ é o seu menor autovetor! Segue que g é perpendicular à $D^{\frac{1}{2}}\mathbf{1}$ e, portanto, é autovetor de $D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$ também, pois pelo fato de esta matriz ser simétrica, existe uma base ortonormal de seus autovetores. Como o interesse é no **argumento mínimo**, isto é, g que seja perpendicular a $D^{\frac{1}{2}}\mathbf{1}$ e minimize a função, então g será exatamente o autovetor associado ao segundo menor autovalor. Isso pode ser demonstrado manipulando a função de minimização também. Como (17) e (18) se equivalem, utilizarei (17)

$$
\begin{aligned}
& g^T D^{-\frac{1}{2}}LD^{-\frac{1}{2}}g \\
= & \quad \{ \text{ definição de autovalor e autovetor } \} \\
& g^T \lambda g \\
= & \quad \{ \text{ deslocando } \lambda \} \\
& \lambda g^T g \\
= & \quad \{ \ ||g|| = 1 \ \} \\
& \lambda
\end{aligned}
$$

O menor λ associado a um autovetor que seja perpendicular ao menor autovetor é exatamente o segundo menor autovalor.

Algoritmo

Toda a especificação do problema foi discutida e a solução da minimização determinada. Resta agora comentar sobre o algoritmo em si. Existem algumas liberdades para diferentes tomadas de decisão no algoritmo e isso será discutido nessa seção de detalhes do algoritmo. Mas a grosso modo, para o caso $k = 2$, o algoritmo seria

Algoritmo 1: Clusterização Espectral $k = 2$

Entrada: $W_{n\times n}$
Saída: $f_{n\times 1}$
1 $D \leftarrow graus(W)$
2 $L \leftarrow D - W$
3 $f \leftarrow segundoMenorAutovetor(D^{-\frac{1}{2}}LD^{-\frac{1}{2}})$
4 $f \leftarrow clusteriza(f)$
5 retorne f

A função de graus calcula o grau de cada vértice i (soma da linha i de W) e coloca na posição i, i de uma diagonal. A função *clusteriza* é a responsável por discretizar o vetor f. Lembre que este assume qualquer valor real nas entradas, mas o interesse é em classificar os pontos em 2 clusters, então algum método de discretização precisa ser feito. As subseções de transformações e cálculo de autovetores expressam um detalhamento de alguns passos do algoritmo e sua leitura é opcional. Posteriormente é discutido sobre a função *clusteriza* e a generalização do algoritmo para $k \geq 2$.

Transformando dados em Grafo

A primeira coisa a se discutir é: "Dado um conjunto de dados quaisquer, como os transformo em um grafo?". Como foi discutido inicialmente, o algoritmo de clusterização espectral recebe como entrada uma matriz de adjacências de um grafo G, mas não necessariamente essa matriz existe a priori para um conjunto de dados qualquer e, de fato, em muitas implementações do algoritmo sua entrada na verdade são os dados e um passo adicional de transformá-los em um grafo é feito.

Isso pode ser feito de diversas maneiras, o modo mais comum é utilizar o conhecido kernel RBF. Uma matriz S de similaridades é calculada como $s_{ij} = e^{-\gamma||v_i - v_j||^2}$, onde γ pode ser alternativamente definido como $\frac{1}{2\sigma^2}$. σ pode, de fato, ser o desvio padrão dos dados ou um hiperparâmetro

livre. Em alguns casos, antes de tratar S como a matriz de adjacências, uma filtragem pode ser feita, utiliza-se um parâmetro r para determinar as entradas de W. A definição então fica

$$w_{ij} = \begin{cases} s_{ij}, \text{ se } s_{ij} < r, \\ 0, \text{ caso contrário} \end{cases} \tag{19}$$

Outra forma é usar o kNN(k-Nearest Neighbours), ligando os vértices aos seus k vizinhos mais próximos. Tendo a matriz de adjacências pronta, a Laplaciana $L = D - W$ pode ser calculada, visto que d_{ii} é somente a soma da linha i de W. O próximo passo é obter alguns autovetores de $D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$.

Computando os Autovetores de L

Mais para a frente, na generalização do problema, será visto que caso se deseje k clusters, os k menores autovetores de $D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$ serão necessários. O problema principal é o tamanho de $D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$. Como comentado na introdução, o método de clusterização espectral foi desenvolvido inicialmente para segmentação de imagens [7], onde cada vértice seria um pixel. O tamanho de $D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$, que depende do número de pixels, pode explodir muito rápido, tornando impraticável encontrar seus autovetores através, por exemplo, da resolução de um sistema linear. A complexidade seria $O(n^3)$, n sendo o número de pixels!

Por sorte, $D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$ é simétrica e, em muitos casos, esparsa e devido à essa sua estrutura há diferentes métodos mais rápidos para computar seus autovetores. Uma forma de se encontrar isso é através do algoritmo de Lanczos, que parte da ideia que uma matriz simétrica S pode ser expressa por uma ortogonal Q e uma tridiagonal T na forma $S = QTQ^T$. A ideia é encontrar aproximações de Q e T, visto que os autovalores de T são os mesmos de S. Conforme o algoritmo itera, os autovalores extremais, isto é, os menores e maiores, das aproximações de T convergem para os de T. Para obter os autovetores de S, basta multiplicar os autovetores da aproximação de T pela aproximação encontrada de Q^T. Para explicações detalhadas veja as referências [2][4].

Há outras alternativas talvez mais famosas como o método da potência [5], que encontra os maiores autovetores, mas com um pouco de manipulação é possível encontrar os menores e até o ARPACK, usado em bibliotecas como o *scikit-learn* [6].

Discretização de f

Como já discutido, f do problema relaxado assume quaisquer valores reais, mas como f é um vetor indicador de clusters seria interessante que este possuísse um número determinado de valores distintos que de fato indicassem à qual cluster cada vértice pertence. Isso pode ser feito de diversas maneiras, aqui algumas são discutidas. A mais direta é escolher um valor m qualquer, por exemplo 0 ou a mediana de f, e definir que caso $f_i < m$, então v_i pertence a A, caso contrário a B. Um modo de definir um bom número de partição m é testar vários valores no intervalo de valores de f, isso é feito no artigo original [7].

Outra forma, e será a utilizada na forma generalizada do algoritmo, é aplicar um método de clusterização nesse vetor. Como se tem somente um vetor, seria uma clusterização em R^1 e qualquer método de clusterização, como o $k$means, por exemplo, pode ser utilizado. f no fim então seria um vetor discretizando, indicando exatamente a qual cluster cada vértice pertence.

Algoritmo para $k \geq 2$

A prova mais formal da generalização pode ser vista em [5], mas o resultado é que para se obter k clusters, basta tomar o k menores autovetores de L. Intuitivamente, após obter o segundo menor, chamarei de v_2, uma nova restrição à otimização pode ser adicionada exigindo g tal que g seja perpendicular ao menor autovetor $\mathbf{e}$ a v_2. Por um argumento semelhante ao usando no caso $k = 2$ a solução desse otimização será exatamente o terceiro menor autovetor.

Logo, de maneira geral, se k clusters são exigidos e $\lambda_1 < \lambda_2 < .... < \lambda_n$ são os autovalores de L, a solução será os autovetores $\{v_2, v_3, ..., v_{k+1}\}$ com v_i autovetor associado a λ_i. O algoritmo generalizado pode agora ser escrito.

Algoritmo 2: Clusterização Espectral

Entrada: $W_{n\times n}$, k
Saída: $f_{n\times 1}$
1 $D \leftarrow graus(W)$
2 $L \leftarrow D - W$
3 $f \leftarrow MenoresAutovetores(D^{-\frac{1}{2}} L D^{-\frac{1}{2}}, k)$
4 $f \leftarrow clusteriza(f)$
5 retorne f

Repare que $MenoresAutovetores$ retornaria, para k clusters, $\{v_1, v_3, ..., v_k\}$, o que difere da teoria original, mas parece ser o que se é feito e acredito, sem provas, que no geral não deve diferir tanto. A ideia geral então seria levar o vértices do grafo para R^k onde a base seria justamente os k menores autovetores e, nesta dimensão reduzida, aplicar um método qualquer de clusterização. Note então que o papel que a clusterização espectral faz é de **reduzir a dimensão** dos dados para que um método de clusterização seja aplicado nos dados em dimensão reduzida. Isso explicita semelhanças desse método com outros que também reduzem como o MDS, isomap e outros.

Exemplos de Aplicação

Essa seção foca em exemplos de aplicações do método. O primeiro que gostaria de mostrar é uma imagem retirada do scikit mostrando diferentes modelos de clusterização representada na figura 2. Note como o $k$means tem grandes dificuldades de acertas padrões não-lineares enquanto métodos como a clusterização espectral e DBSCAN costumam acertar.

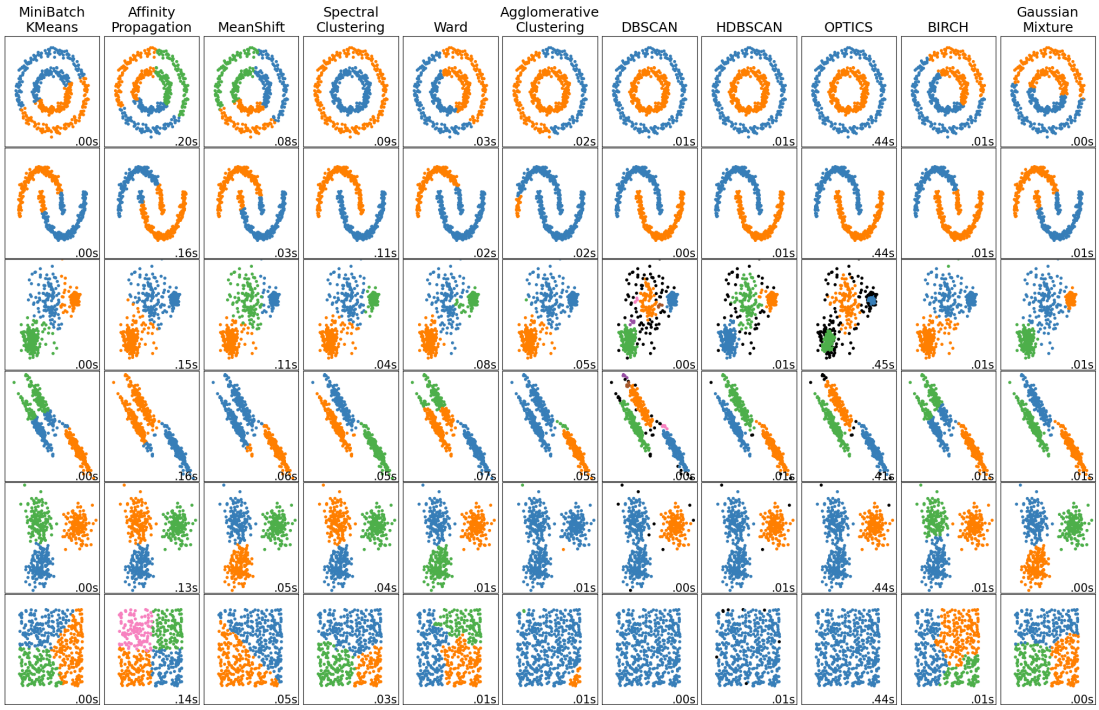

Figura 2: Imagem retirada de `https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html`

O algoritmo também é utilizado em segmentação de imagens. Abaixo mostro algumas imagens no qual testei. Os resultados estão nas figuras 3, 5 e 6. Essencialmente usei a biblioteca *scikit-learn* para aplicar o método. Todos os grafos foram produzidos tomando o gradiente de cada pixel para

cada pixel. Em seguida cada coordenada foi transformada usando a função $e^{\frac{-\beta w_{ij}}{\sigma}} + \varepsilon$, σ desvio padrão dos dados do grafo e ε hiperparâmetro de valor arbitrário. Abaixo trato γ como $\frac{\beta}{\sigma}$. Os testes abaixo usaram o $k$means para a clusterização na dimensão reduzida, embora o "discretize"do scikit-learn trouxesse resultados parecidos.

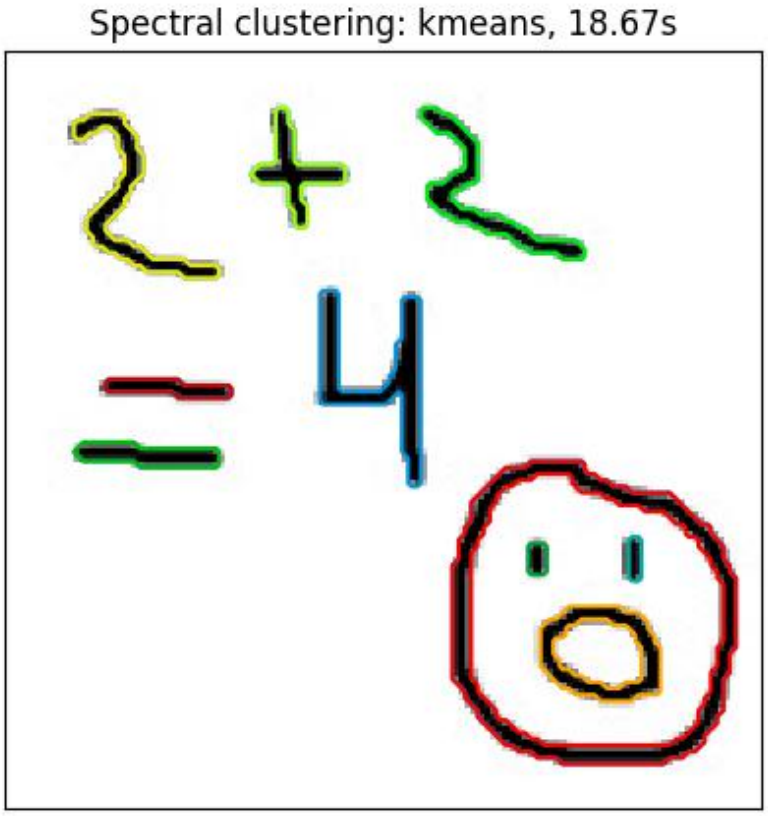

Figura 3: Classificação da clusterização espectral com $k = 13$ e $\gamma = \frac{10}{\sigma}$, σ desvio padrão da imagem, em um desenho feito usando o mouse no Paint. Um fator $\varepsilon = 10^{-6}$ foi somado no peso de todas as arestas.

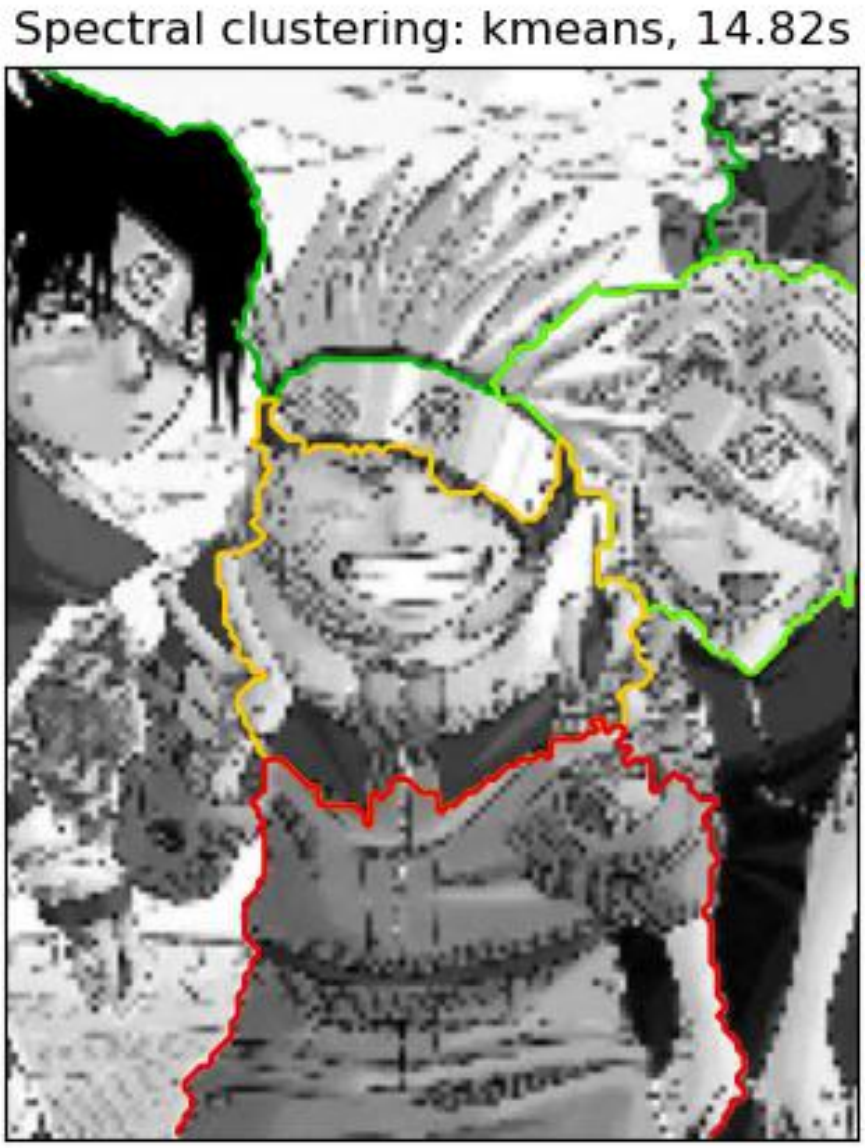

Figura 4: Classificação da clusterização espectral com $k = 13$ e $\gamma = \frac{5}{\sigma}$, σ desvio padrão da imagem, em uma imagem do mangá de Naruto. Um fator $\varepsilon = 10^{-3}$ foi somado no peso de todas as arestas.

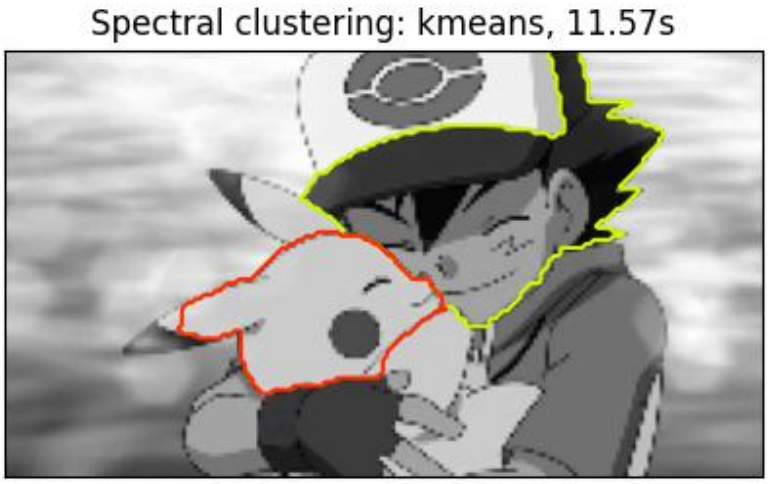

Figura 5: Classificação da clusterização espectral com $k = 11$ e $\gamma = \frac{5}{\sigma}$, σ desvio padrão da imagem, em uma imagem de Pokémon. Um fator $\varepsilon = 10^{-3}$ foi somado no peso de todas as arestas.

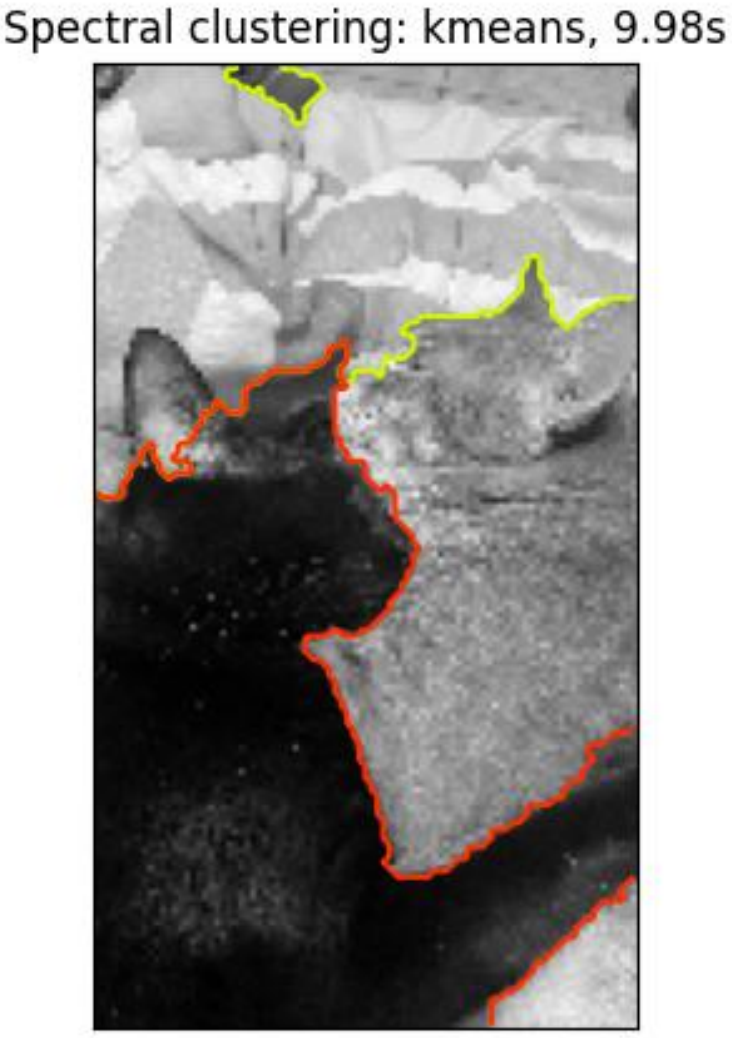

Figura 6: Imagem de dois gatos se abraçando. Utilizei $k = 5$, $\gamma = \frac{10}{\sigma}$, σ desvio padrão da imagem. Um fator $\varepsilon = 10^{-6}$ foi somado no peso de todas as arestas.

Bônus: Um Limitante Inferior Para o k Ótimo

O algoritmo exige a passagem de k como parâmetro e não prevê qual o melhor para a clusterização. Porém uma tentativa de estimar um limitante inferior pode ser possível. Como o dado trabalhado é um grafo, é natural pensar que se o grafo tem m componentes conexas, ao menos k deveria ser maior ou igual a m. Um teorema pode ajudar quanto a isso, mas antes a definição de componente conexa.

Definição Seja $G = (V, E)$ um grafo não direcionado, um subgrafo H de G é uma componente conexa se há pelo menos um caminho entre todos os vértices de H e não é subconjunto de nenhum outro subgrafo conexo.

Pode-se pensar em componentes conexas como "ilhas", não há arestas entre vértices de diferentes componentes conexas.

Teorema Seja G um grafo não-direcionado com todas as arestas com peso não-negativo e L sua matriz Laplaciana, o número de componentes conexas de G é exatamente o número de autovetores de L associados ao autovalor 0, onde os autovetores relacionados às componentes $\{A_1, ..., A_m\}$ serão múltiplos dos vetores $\{\mathbf{1}_1, ..., \mathbf{1}_m\}$ cuja definição é

$$(\mathbf{1}_j)_i = \begin{cases} 1, \text{ se }, v_i \in A_j \\ 0, \text{ caso contrário} \end{cases} \tag{20}$$

Isto é, cada componente tem um autovetor constante com 1 nas entradas dos vértices que pertencem à componente e 0 para os que não pertencem.

Farei apenas um esboço da prova, visto que nem tudo usado será detalhadamente provado. Uma prova para o teorema pode ser vista em [3].

Prova: Suponha f autovetor de L associado ao autovalor 0, segue que $f^T L f = 0$ pela definição de autovalores e autovetores. Para o caso $m = 1$, todo o grafo é conexo e como visto anteriormente outro modo de escrever $f^T L f$ é $\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$. Como todo peso de aresta é positivo, todas as parcelas desse somatório devem ser zeradas. No caso de $w_{ij} > 0$, $f_i = f_j$ e dois vértices conectados pertencem ao mesmo cluster. Olhando com mais cuidado, se há um caminho entre dois vértices v_i e v_j, então $f_i = f_j$. Para verificar isso melhor suponha um grafo caminho com somente 3 vértices, este possui duas arestas (v_1, v_2) e (v_2, v_3), pelo primeiro par $f_1 = f_2$ e pelo segundo $f_2 = f_3$. Como o grafo todo é conexo, f será $\mathbf{1}$.

Para o caso geral de $m \geq 2$ componentes conexas, ordene os vértices baseado em qual componente pertencem, a matriz W de adjcências seria então um matriz com blocos na diagonal, visto que vértices de componentes distintas não possuem arestas entre si.

$$W = \begin{bmatrix} W_1 & & & \\ & W_2 & & \\ & & \ddots & \\ & & & W_m \end{bmatrix}$$

Onde W_i é a matriz de adjacências da componente i. De modo similar, a Laplaciana de G será uma matriz diagonal com a Laplacina de cada componente do grafo.

$$L = \begin{bmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_m \end{bmatrix}$$

Para matrizes em bloco, é possível definir o determinante de maneira parecida com a que se fazia matrizes com elementos. Neste caso, por L ser diagonal, $det(L) = det(L_1) \cdot det(L_2) \cdot .... \cdot det(L_m)$, segue que $det(L - \lambda I) = det(L_1 - \lambda I) \cdot det(L_2 - \lambda I) \cdot .... \cdot det(L_m - \lambda I)$ e, portanto, os autovalores de

L são a união dos autovalores de L_i. Não somente isso, mas seus autovetores serão os autovetore de L_i com 0 nas posições fora de seu bloco. Como cada bloco L_i representa uma componente conexa, então cada bloco possui um autovetor constante nas posições dos vértices pertencentes à componente e 0 nos que não pertencem. Logo o número de autovetores de L associados a 0 é exatamente o número de componentes conexas do grafo G, assim como os autovetores são constantes para vértices na componente e tem valor 0 para os que não pertencem. □

Com esse teorema e encontrada a multiplicidade do autovalor 0 é possível determinar que há m "ilhas"no grafo e, portanto, m potenciais clusters.

Referências

[1] Francis R. Bach and Michael I. Jordan. Learning spectral clustering, with application to speech separation. *Journal of Machine Learning Research*, 7(71):1963–2001, 2006.

[2] Gene H. Golub and Charles F. Van Loan. *Matrix Computations - 4th Edition*. Johns Hopkins University Press, Philadelphia, PA, 2013.

[3] Ulrike Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, dec 2007.

[4] Kiran Kumar Matam and Kishore Kothapalli. Gpu accelerated lanczos algorithm with applications. In *Proceedings of the 2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications*, WAINA '11, page 71–76, USA, 2011. IEEE Computer Society.

[5] Danielle Middlebrooks and Kasso Okoudjou. Application of the spectral clustering algorithm. 2016.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[7] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22, 05 2002.

[8] Frederick Tung, Alexander Wong, and David Clausi. Enabling scalable spectral clustering for image segmentation. *Pattern Recognition*, 43:4069–4076, 12 2010.