

Spectral Clustering e Aplicações

Pedro Hiroshi Ely Ito
João Francisco Zani de Arruda

Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo

12/07/2022

1 Introdução

- Clustering
- Spectral Clustering
- K-means

2 Aplicação em imagens

- Criando o Grafo
- Clustering
- Resultados

Clustering refere-se a um conjunto de métodos que buscam subconjuntos(clusters) de um dataset que computem alguma forma de similaridade: elementos num mesmo cluster são parecidos, enquanto elementos de clusters diferentes não.

Um método comum nesse tipo de tarefa é o Spectral Clustering. Nele, criamos um grafo a partir do dataset e usamos os autovalores do Laplaciano para encontrar uma representação 2d desse grafo. A partir dessa representação, aplicamos k-means.

Definition

Spectral Clustering:

1. Calcule o Laplaciano L .
2. Calcule os k menores autovalores.
3. Crie uma matriz a partir dos k primeiros autovetores.
4. Aplique k-means usando a matriz.

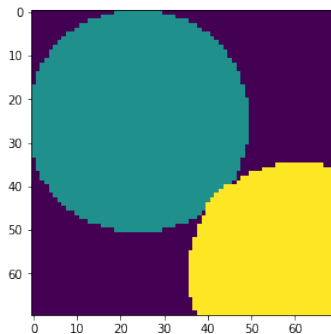
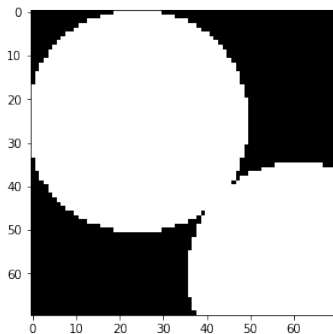
Definition

K-means:

1. Selecione k "médias" aleatórias.
2. Associe cada ponto do dataset à média mais próxima.
3. Atualize as médias para os centróides de cada cluster.
4. Repita até a convergência.

Segmentação de objetos

Vamos aplicar Spectral Clustering na segmentação de objetos em imagens: dada uma imagem com n objetos, queremos delimitar as regiões correspondentes a cada objeto.



De imagem para grafo

Vamos trabalhar com imagens em nível de cinza, representada por uma matriz com entradas no intervalo $(0, 255)$, onde 0 é preto e 255 branco.

Exemplo:



```
array([[255, 255, 255, 255, 255, 0, 0, 0, 0, 0, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 0, 0, 0, 0, 0, 0, 0, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 0, 0, 170, 170, 0, 170, 170, 255, 255, 255, 255, 255],
       [255, 255, 255, 0, 170, 0, 0, 170, 170, 170, 170, 170, 255, 255, 255, 255],
       [255, 255, 0, 0, 170, 170, 170, 170, 170, 0, 170, 170, 170, 255, 255, 255],
       [255, 255, 0, 0, 170, 170, 170, 170, 0, 0, 0, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 0, 170, 170, 170, 170, 170, 255, 255, 255, 255, 255],
       [255, 255, 255, 255, 0, 85, 0, 0, 0, 85, 255, 255, 255, 255, 255, 255],
       [255, 255, 255, 0, 0, 85, 0, 0, 0, 85, 0, 255, 255, 255, 255, 255],
       [255, 255, 0, 0, 0, 85, 0, 0, 0, 85, 0, 255, 255, 255, 255, 255],
       [255, 255, 0, 0, 0, 255, 85, 85, 85, 255, 0, 0, 255, 255, 255, 255],
       [255, 255, 170, 170, 170, 85, 85, 85, 85, 85, 170, 170, 170, 255, 255, 255],
       [255, 255, 255, 170, 170, 85, 85, 85, 85, 85, 170, 170, 255, 255, 255, 255],
       [255, 255, 255, 85, 85, 85, 85, 85, 255, 85, 85, 85, 255, 255, 255, 255],
       [255, 255, 255, 0, 0, 0, 255, 255, 255, 0, 0, 0, 255, 255, 255, 255],
       [255, 255, 0, 0, 0, 0, 255, 255, 255, 0, 0, 0, 255, 255, 255, 255]]
```

A partir dessa imagem, criamos um grafo onde cada nó representa um pixel. Cada pixel é ligado aos pixels vizinhos(cima, baixo, esquerda e direita). O peso das arestas é dado por:

$$W_{pq} = e^{-\frac{|I_p - I_q|^2}{2\sigma^2}} \quad (1)$$

Onde I_p é a intensidade do pixel p e σ é um "fator de escala", quanto menor for σ , menores são os clusters formados.

```
def neighbors(img):  
    right = np.vstack([img[:, :-1].ravel(), img[:, 1:].  
        ravel()])  
    below = np.vstack([img[:, -1, :].ravel(), img[1:, :].  
        ravel()])  
    neighbors = np.unique(np.hstack([right, below]), axis  
        =1)  
  
    return neighbors
```

```

def create_nxGraph(img):
    m,n = img.shape
    sigma = np.std(img)
    img_num = (np.arange(m*n).reshape(m,n))
    adj = neighbors(img_num)

    G = nx.Graph()
    G.add_nodes_from(np.arange(m*n))

    I = img.flatten()[adj[1]] - img.flatten()[adj[0]]
    weights = np.exp(-(np.abs(I))**2)
    weights = weights/(2*sigma**2)

    edges = zip(adj[0,:], adj[1:], weights)
    G.add_weighted_edges_from(edges)
    nx.set_node_attributes(G, img.flatten(), "intensity")

    bool_img = img.astype(bool)
    mask = img_num[bool_img]

    return G.subgraph(mask)

```

Aplicando Spectral Clustering

```
from sklearn.cluster import spectral_clustering
import matplotlib.pyplot as plt

img = Image.open('imagem.png')
img = img.convert('L') #converte para grayscale
img = np.asarray(img) #converte para array

G = create_nxGraph(img) #cria o grafo
A = nx.to_numpy_array(G) #retorna a matriz de adjacencia
A = np.nan_to_num(A) #substitui infinitos

mask = img.astype(bool) #consideramos apenas os objetos
labels = spectral_clustering(A, n_clusters=3,
                             eigen_solver="arpack")
label_im = np.full(mask.shape, -1.0) #array do resultado
label_im[mask] = labels #preenche os resultados
```

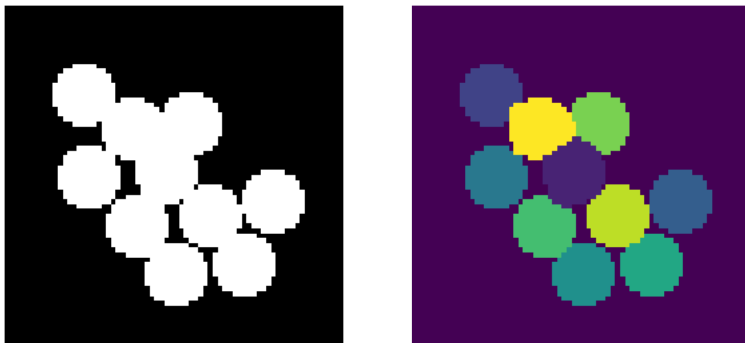


Figura: $n_{clusters} = 10$



Figura: $n_{clusters} = 3$



Figura: $n_{clusters} = 2$



Figura: $n_{clusters} = 2$



Figura: $n_{clusters}=3$

Muito Obrigado!