Aprendizado de Máquina - Exercício 4

Thiago Amendola - 148062

Introdução

Foi feita a leitura de 5417 linhas de dados com 3 dimensões distintas. Foi aplicado o algoritmo de K-means sobre estes dados, adotando número de restarts igual a 5. Para a busca do valor k para o número de clusters, foram adotadas duas soluções: uma métrica interna, utilizando índice de Dunn das distâncias entre os núcleos dos clusters, e uma métrica externa, utilizando índice de Rand e um conjunto fornecido de classes para os dados. Estas métricas pontuaram os melhores valores de k, que posteriormente foram utilizadas na plotagem de dois gráficos, um para cada métrica utilizada, apresentando os pontos reduzidos via PCA, os centros dos clústeres encontrados e seus limiares.

Resolução

Começamos importando as bibliotecas que serão utilizadas nesta aplicação:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.cluster import KMeans
from sklearn.neighbors import DistanceMetric
from sklearn.decomposition import PCA
from sklearn import metrics
```

É então feita a leitura dos dados dos arquivos fornecidos:

```
raw_data = pd.read_csv('cluster-data.csv', ',')
raw_labels = pd.read_csv('cluster-data-class.csv', ' ')
datas = raw_data.values
labels = raw_labels.values
```

Começamos então a métrica interna por índice de Dunn. A configuração de índice adotada é calculada com a distância máxima entre os pontos e os centros dos clusters aos quais pertencem dividido pela distância mínima entre dois cluesteres quaisquer:

```
    # ============ Internal metric: Dunn
print("=> Starting Kmeans with internal metric using Dunn of distances")
# Get best value for k
best_dunn = 0
best_k = 2
for k in range(2,11):
    inter, intra = 0, 0
    km = KMeans(n_clusters=k, n_init=5)
```

```
        km.fit(datas)
        # Calculate Dunn Index of center distances
        dist = DistanceMetric.get_metric('euclidean')
        # INTRA
        intraGroup = []
        for i in range(k):
            intraGroup.append(0)
        for i in range(len(datas)):
            X = [datas[i],km.cluster_centers_[km.labels_[i]]]
            intraGroup[km.labels_[i]] += float(dist.pairwise(X)[0,1])
        intra = max(intraGroup)
        #INTER
        interGroup = []
        for i in range(k): # For each cluster
            for j in range(i+1,k):
                X=[km.cluster_centers_[i], km.cluster_centers_[j]]
                interGroup.append(float(dist.pairwise(X)[0,1]))
        inter = min(interGroup)
        #DUNN
        dunn=inter/intra
        if(dunn>best_dunn):
            best_dunn = dunn
            best_k = k
```

Para o cálculo da métrica externa, foi utilizado o índice de Rand. Este índice, assim como outros de métrica externa, recebem como entrada as classes dos dados fornecidos. O índice é então cálculado com base nas classes para os dados fornecidas e os clústeres estimados pelo algoritmo de K-means:

```
    # ============ External metric: Indice Rand
    print("=> Starting Kmeans with external metric using Rand Index")
    # Get best value for k
    best_score = 0
    best_k = 2

    for k in range(2,11):
        km2 = KMeans(n_clusters=k, n_init=5)
        km2.fit(datas)
        score = metrics.adjusted_rand_score(labels[:,0], km2.labels_)
        if(score>best_score):
            best_score = score
            best_k = k
```

Para cada uma das métricas utilizadas, foram plotados gráficos apresentando os pontos dos dados de entrada, dos centros dos clústeres e os limiares entre clústeres

```
    # Plot
    reduced_data = PCA(n_components=2).fit_transform(datas)
    km2 = KMeans(n_clusters=best_k, n_init=5)
    km2.fit(reduced_data)
```

```
    # Plot the decision boundary. For that, we will assign a color to each
    x_min, x_max = reduced_data[:, 0].min() - 1, reduced_data[:, 0].max() + 1
    y_min, y_max = reduced_data[:, 1].min() - 1, reduced_data[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
    # Obtain labels for each point in mesh. Use last trained model.
    Z = km2.predict(np.c_[xx.ravel(), yy.ravel()])
    # Put the result into a color plot
    Z = Z.reshape(xx.shape)

    plt.figure(1)
    plt.clf()
    plt.imshow(Z, interpolation='nearest',
               extent=(xx.min(), xx.max(), yy.min(), yy.max()),
               cmap=plt.cm.Paired,
               aspect='auto', origin='lower')

    plt.plot(reduced_data[:, 0], reduced_data[:, 1], 'k.', markersize=2)

    centroids = km2.cluster_centers_
    plt.scatter(centroids[:, 0], centroids[:, 1],
            marker='x', s=169, linewidths=3,
            color='b', zorder=10)
    plt.title('K-means - External evaluation by Rand Index (data reduction via
PCA)\n')
    plt.xlim(x_min, x_max)
    plt.ylim(y_min, y_max)
    plt.xticks(())
    plt.yticks(())
    plt.show()
```

Resultados

Para a execução com avaliação interna por índice de Dunn, o melhor valor de índice foi de *0.0019005162887162077*, com k igual a *10*. O gráfico para esta execução mostrou um uso demasiado de clústeres, não englobando com precisão dados mais próximos. Uma possível causa para esta imprecisão pode ser a escolha de valores para a composição do índice de Dunn. O uso de outros índices, ou mesmo do índice de Dunn utilizando outros métodos de cálculo, podem fornecer valores melhores que o encontrado.

Na execução com com avaliação externa por índice Rand, o melhor valor de índice foi de *0.5250334201*, com k igual a *4*. O gráfico para esta execução apresentou-se bem mais preciso que o anterior, separando o conjunto de clústeres de maneira bem definida.