SOFTEX/UNIFEI - Curso de Introdução à Inteligência Artificial

3. Aprendizado de Máquina

Aula 06 - Tutorial notebook 2 - Aprendizado Não-Supervisionado

Autor: Rafael Frinhani

Tópico: Processo de Machine Learning - Desenvolvimento e Validação do Modelo

Objetivo: Este notebook mostra as etapas de "Desenvolvimento do Modelo" e "Avaliação e Teste do Modelo" do processo de *Machine Learning*, mais especificamente a aplicação de algoritmos de aprendizado Não-Supervisionado para a tarefa de agrupamento de dados.

Organização do Notebook: Os exemplos a seguir consideram um conjunto de dados biológicos bem conhecido na literatura. Na forma original o conjunto de dados possui os rótulos de classe de todos os objetos. Em um cenário real de aplicação de algoritmos de agrupamento os rótulos não são conhecidos. Para fins didáticos, nos exemplos relacionados ao entendimento dos dados foram considerados os rótulos de classe dos objetos. Na parte relacionada ao desenvolvimento do modelo de agrupamento tais rótulos foram desconsiderados, sendo utilizados apenas para verificação dos resultados.

Conjunto de Dados

O conjunto de dados utilizado neste notebook é chamado Iris, cujos arquivos e demais informações constam no UCI Machine Learning Repository. O cenário do problema associado ao conjunto de dados é o de uma classificação taxonomica de três espécies da flores Iris (Íris-Setosa, Íris-Versicolor e Íris-Virginica), que embora sejam visualmente semelhantes possuem, possuem diferenças anatômicas em relação a medidas de comprimento e largura das sépalas e das pétalas. O conjunto de dados possui um total de 150 flores de íris, sendo 50 amostras para cada uma das espécies.

Fonte: Wikipedia (Flores de Iris)

O conjunto de dados pode ser importado diretamente do repositório da UCI, sua carga é feita em um *DataFrame* Pandas, seguida da visualização dos primeiros registros conforme a seguir:

```
#import warnings
#warnings.filterwarnings("ignore")

import pandas as pd
df = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data", header=None)
df.head()
```

Cada registro corresponde a uma flor. Os atributos das colunas 0 e 1 são respectivamente as medidas de comprimento e largura da sépala, e os atributos 2 e 3 representam o comprimento e largura da pétala. O atributo 4 representa os rótulos das classes (espécie da flor). Para uma melhor interpretação dos dados, as colunas podem ser renomeadas:

```
df.rename(columns = {0:'comp_sepala',
                     1:'larg_sepala',
                     2:'comp_petala',
                     3:'larg_petala',
                     4:'classe'}, inplace = True)
df
```

Para um melhor entendimento de como os objetos do conjunto de dados estão distribuídos, é possível gerar um gráfico de dispersão para visualizar como esses atributos estão relacionados. Para isso, será utilizada a biblioteca Seaborn, que possui recursos para plotagem gráficos em Python, sendo útil para explorar com mais detalhes como os dados se comportam numa base de dados.

O comando a seguir plota diversos gráficos a partir da combinação dos atributos, sem considerar o atributo 4 (classe). Este atributo foi propositalmente desconsiderado para remeter a uma situação real, na qual não se conhece a classe do objeto.

```
import seaborn as sns
sns.set_theme() #Define aspectos do tema visual para todos os gráficos
sns.pairplot(df[['comp_sepala', 'larg_sepala', 'comp_petala', 'larg_petala']])
```

O comando a seguir considera o atributo classe. Pelos gráficos de dispersão, percebe-se que determinadas combinações de atributos resultam em distribuições que facilitam identificar as divisões entre as classes.

Na maioria das combinações, a classe Iris-setosa é a que mais se diferencia das outras duas classes. Já Iris-versicolor e Iris-virginica possuem valores que se sobrepõem independente da combinação de atributos. Esta situação ocasiona em maiores dificuldades para particionar os objetos dessas classes.

```
import seaborn as sns
sns.set_theme() #Define aspectos do tema visual para todos os gráficos
sns.pairplot(df[['comp_sepala', 'larg_sepala', 'comp_petala', 'larg_petala', 'classe']], hue="classe", diag_kind="kde")
```

Desenvolvimento de um modelo de Agrupamento

Considerando que a tarefa de agrupamento é realizado quando não se conhece o rótulo das classes dos objetos, nos exemplos a seguir será adotado o conjunto de dados Iris sem considerar o atributo 4 (classe). Este atributo será utilizado apenas para validação do exemplo, ou seja, o método de aprendizado não receberá informação dos rótulos.

Preparação do Conjunto de Dados

Os objetos do conjunto de dados possuem quatro atributos, correspondendo a um problema cujos dados estão distribuídos espacialmente em quatro dimensões. Para uma melhor visualização dos clusters gerados, neste exemplo serão considerados apenas os atributos comprimento da pétala e comprimento da sépala.

A partir do conjunto de dados original (*DataFrame* df), o comando a seguir obtém um conjunto X com os atributos 'comp_petala' e 'comp_sepala', e um conjunto Y com os rótulos das classes de cada objeto.

```
X = df[['comp_petala','larg_petala']] # Obtém um dataframe X apenas com os atributos comprimento e largura da pétala
Y = df['classe'] # Obtém um dataframe Y que contém apenas os rótulos de classe
print("Atributo (X):", '\n', X, '\n')
print("Classes (Y):", '\n', Y)
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
Y = le.fit_transform(Y)
Y
```

Para o agrupamento de dados será utilizado o algoritmo ***K-Means***, que em termos gerais executa os seguintes passos:

0. Definir a quantidade *k* de clusters;
1. Selecionar *k* amostras aleatórias dos dados como centroides;
2. Atribuir todas as amostras ao centróide mais próximo;
3. Recalcular a posição dos centróides dos grupos recém-formados;
4. Repetir os passos 3 e 4 até a convergência do método.

Definir a quantidade de grupos

Quando não se conhece a quantidade de classes de um conjunto de dados, existem formas de encontrar um valor *k* da quantidade de grupos que seja próximo ao ideal. O metodo do Cotovelo (***Elbow Method***) pode ser usado para recomendar a quantidade de grupos, baseando-se na construção de diversos modelos do algoritmo com diferentes valores de *k* (ex. 2 a 30). Em seguida, gera um gráfico mostrando o valor de inércia do modelo conforme *k*. A inércia é a soma das distâncias quadradas das amostras até o centro do grupo mais próximo. Os valores posicionados no "cotovelo" do gráfico são os mais indicados para *k*. No exemplo a seguir foi considerado um range de 1 a 10 clusters.

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
vetor_inercia = []

for i in range(1, 10): # Testados modelos de 1 a 10 clusters
    kmedias = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 42)
    kmedias.fit(X)
    vetor_inercia.append(kmedias.inertia_) # Calcula a soma do quadro das distâncias intra-cluster (Elbow Method)
```

```
plt.figure(figsize = (8,6))
plt.plot(range(1, 10), vetor_inercia, marker='o')
plt.title('ELBOW METHOD')
plt.xlabel('Quantidade de grupos k')
plt.ylabel('Inércia')
plt.show()
```

Pelo gráfico, nota-se que os melhores valores possíveis para o modelo seriam 2 ou 3. Como já se tem uma noção da quantidade de classes do conjunto, no exemplo a seguir será considerado *k* = 3. No caso de um conjunto de dados com rótulos desconhecidos, poderia ser testado *k* = 2 e *k* = 3, sendo selecionado o valor que produzir os clusters mais satisfatórios.

Ao final da execução do K-Means cada objeto de dados possui um valor numérico que corresponde a uma classe.

```
# Função 'fit_predict()' computa os centros dos clusters obtem o id do cluster de cada objeto
kmedias = KMeans(n_clusters = 3, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 42)
X_kmedias = kmedias.fit_predict(X)
X_kmedias
```

O comando a seguir permite visualizar os objetos de dados e seus respectivos *clusters* e centroides. Os *clusters* são rotulados conforme o rótulo de classe: 0 = Iris-virginica; 1 = Iris-setosa; 2 = Iris-versicolor.

```
plt.figure(figsize = (10,8))
plt.scatter(X[X_kmedias == 0]["comp_petala"], X[X_kmedias == 0]["larg_petala"], s = 100, c = 'green', label='Iris-virginica')
plt.scatter(X[X_kmedias == 1]["comp_petala"], X[X_kmedias == 1]["larg_petala"], s = 100, c = 'blue', label='Iris-setosa')
plt.scatter(X[X_kmedias == 2]["comp_petala"], X[X_kmedias == 2]["larg_petala"], s = 100, c = 'orange', label='Iris-versicolor')

# Mostra o centroide de cada grupo na cor vermelha
plt.scatter(kmedias.cluster_centers_[:, 0], kmedias.cluster_centers_[:,1], s = 100, c = 'red', label = 'Centroides')
plt.legend()
```

O agrupamento obtido pelo K-Means podem ser comparado com o agrupamento dado pelos rótulos do conjunto de dados.

```
from matplotlib.colors import ListedColormap
fig, ax = plt.subplots(nrows=2, ncols=1, figsize=(10,18))

ax[0].title.set_text('CLASSES CONFORME RÓTULOS DE CLASSE DO CONJUNTO ORIGINAL')
ax[0].scatter(X['comp_petala'], X['larg_petala'], c=Y, s = 100, cmap=ListedColormap(['blue', 'orange', 'green']))

ax[1].title.set_text('AGRUPAMENTO K-MEANS')
ax[1].scatter(X['comp_petala'], X['larg_petala'], c=X_kmedias, s = 100, cmap=ListedColormap(['green', 'blue', 'orange']))
```

Pelo gráfico nota-se que os objetos que tiveram os grupos trocados estão posicionados na região de sobreposição entre os grupos Iris-virgínica e Iris-versicolor. Para verificar a qualidade do agrupamento, o comando a seguir utiliza a métrica ARI (*Adjusted Rand Index*) que é uma medida externa que permite comparar os rótulos de classe originais de cada objeto (conjunto Y) com os rótulos obtidos pelo K-Means (conjunto X_kmedias). Valores de ARI próximos de 1 indicam um pareamento perfeito (verifique fazendo o teste considerando dois conjuntos iguais).

```
from sklearn.metrics.cluster import adjusted_rand_score

ARI = adjusted_rand_score(Y,X_kmedias)
print('%.3f' % ARI)
```

```
0.886
```

Agrupamento Hierárquico

Para o mesmo conjunto de dados, o código a seguir gera um agrupamento hierárquico (*Hierarchical Cluster Analysis*, HCA) e obtém o dendrograma do agrupamento obtido com estratégia aglomerativa e método de ligação Ward.

```
# Código adaptado do original disponível em https://scikit-learn.org/stable/auto_examples/cluster/plot_agglomerative_dendrogram.html)
import numpy as np
from scipy.cluster.hierarchy import dendrogram
from sklearn.cluster import AgglomerativeClustering

# Cria uma matriz de ligações e então plota o dendrograma
def plot_dendrogram(model, **kwargs):
    # criar as contagens de amostras em cada nó da árvore do dendrograma
    counts = np.zeros(model.children_.shape[0])
    n_samples = len(model.labels_)
    for i, merge in enumerate(model.children_):
        current_count = 0
        for child_idx in merge:
            if child_idx < n_samples:
                current_count += 1  # nó folha
            else: current_count += counts[child_idx - n_samples]
        counts[i] = current_count
    linkage_matrix = np.column_stack([model.children_, model.distances_, counts]).astype(float)
    # Plota o dendrograma correspondente
    dendrogram(linkage_matrix, **kwargs)

# Ajusta distance_threshold = 0 para garantir o cálculo de toda árvore
modelo_HCA = AgglomerativeClustering(distance_threshold=0, n_clusters=None, linkage='ward', compute_full_tree='auto')
agrupamento = modelo_HCA.fit(X)
plt.title("DENDROGRAMA DO AGRUPAMENTO HIERÁRQUICO")

# Plota os primeiros três níveis do dendrograma
plot_dendrogram(agrupamento, truncate_mode="level", p=3)
plt.xlabel("Número de pontos associados ao nó")
plt.show()
```

Referências

- McKINNEY, Wes. "Python para análise de dados: Tratamento de dados com Pandas, NumPy e IPython". Editora Novatec, 2018.
- KLOSTERMAN, Stephen. "Projetos de Ciência de Dados com Python: Abordagem de estudo de caso para a criação de projetos de ciência de dados bem sucedidos usando Python, Pandas e Scikit-Learn. Editora Novatec, 1ª Edição, 2020.
- GÉRON, Aurélien. "Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow". O'Reilly Media, Inc., 2022.
- Biblioteca Pandas Python
- Biblioteca Pandas-Profile Python
- Biblioteca Sciki-Learn - KMeans
- Biblioteca Sciki-Learn - Agglomerative Clustering