|  |  |  |
| --- | --- | --- |
|  | BAN CƠ YẾU CHÍNH PHỦ  “HỌC VIỆN KỸ THUẬT MẬT MÃ” | Mẫu 2 |

BÁO CÁO CHUYÊN ĐỀ SỐ 4.4.3

“Phương pháp sinh đặc trưng K-means”

NHIỆM VỤ: “Nghiên cứu và ứng dụng nền tảng học sâu để xây dựng hệ thống phát hiện mã độc trực tuyến”.

Mã số: 06/2022/CB.

Cơ quan chủ trì: Học viện Kỹ thuật Mật mã

Chủ nhiệm: ThS. Lê Đức Thuận

Hà Nội - 2023

|  |  |  |
| --- | --- | --- |
|  | BAN CƠ YẾU CHÍNH PHỦ  “HỌC VIỆN KỸ THUẬT MẬT MÃ” |  |

BÁO CÁO CHUYÊN ĐỀ SỐ 4.4.3

“Phương pháp sinh đặc trưng K-means”

NHIỆM VỤ: “Nghiên cứu và ứng dụng nền tảng học sâu để xây dựng hệ thống phát hiện mã độc trực tuyến”.

Mã số: 06/2022/CB.

Cơ quan chủ trì: Học viện Kỹ thuật Mật mã

Chủ nhiệm: ThS. Lê Đức Thuận

|  |  |
| --- | --- |
| **Người thực hiện chuyên đề** | **Cơ quan chủ trì** |
| *(Họ tên và chữ ký)* | *(Họ tên và chữ ký)* |

Hà Nội - 2023

MỤC LỤC

[MỤC LỤC 1](#_Toc129698723)

[DANH MỤC HÌNH ẢNH 1](#_Toc129698724)

[PHƯƠNG PHÁP SINH ĐẶC TRƯNG K-MEANS 2](#_Toc129698725)

[1.1. Thuật toán K-Means 2](#_Toc129698726)

[1.2. Phân nhóm đặc trưng dựa trên K-mean 3](#_Toc129698727)

[1.3. Tạo đặc trưng cải tiến 3](#_Toc129698728)

[1.4. Code 4](#_Toc129698729)

# DANH MỤC HÌNH ẢNH

[Hình 1. Thuật toán K-means 3](#_Toc129698631)

# PHƯƠNG PHÁP SINH ĐẶC TRƯNG K-MEANS

## Thuật toán K-Means

K-Means là thuật toán rất quan trọng và được sử dụng phổ biến trong kỹ thuật phân cụm. Tư tưởng chính của thuật toán K-Means là tìm cách phân nhóm các đối tượng (objects) đã cho vào K cụm (K là số các cụm được xác đinh trước, K nguyên dương) sao cho tổng bình phương khoảng cách giữa các đối tượng đến tâm nhóm (centroid ) là nhỏ nhất.

Thuật toán K-Means được mô tả ở Hình 2, được thực hiện qua các bước chính sau:

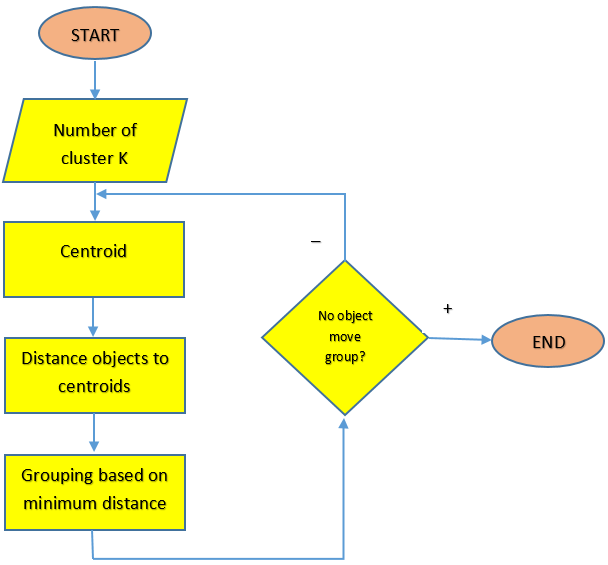
1. Chọn ngẫu nhiên K tâm (centroid) cho K cụm (cluster). Mỗi cụm được đại diện bằng các tâm của cụm.

2. Tính khoảng cách giữa các đối tượng (objects) đến K tâm (thường dùng khoảng cách Euclidean)

3. Nhóm các đối tượng vào nhóm gần nhất

4. Xác định lại tâm mới cho các nhóm

5. Thực hiện lại bước 2 cho đến khi không có sự thay đổi nhóm nào của các đối tượng



Hình 1. Thuật toán K-means

## Phân nhóm đặc trưng dựa trên K-mean

Sử dụng thuật toán K-means để chia các đặc trưng trong tập dữ liệu thành K cụm, nghĩa là tìm mối quan hệ giữa các đặc trưng trong mỗi cụm, từ đó đề xuất thêm K đặc trưng mới chính là điểm trung tâm của K cụm đó.

Để thực hiện điều đó, chúng tôi hoán vị ma trận dữ liệu trong bộ dữ liệu ban đầu (file .csv), nghĩa là hoán đổi hàng thành cột, cột thành hàng. Mỗi hàng thu được biểu diễn toàn bộ giá trị của một cột đặc trưng nhị phân ban đầu. Mục đích là biến các đặc trưng thành các instance, còn mỗi mẫu trở thành một attribute.

Chúng tôi chạy thuật toán K-means với bộ dữ liệu hoán vị đó, với từng giá trị K xác định. Kết quả thu được thông tin về K cụm, mỗi cụm gồm những đặc trưng (ban đầu) nào, và mỗi điểm trung tâm của cụm đó.

## Tạo đặc trưng cải tiến

Đặc trưng nhóm là đặc trưng được tổng hợp từ một nhóm các đặc trưng khác, thể hiện mối quan hệ đồng hiện giữa các đặc trưng của nhóm.

Theo công thức (1), hàm sinh đặc trưng nhóm được ký hiệu là fc(G). Chúng dựa trên quan hệ xuất hiện đồng thời thể hiện mối liên hệ giữa các đặc trưng trong các mẫu dữ liệu, chúng tôi xây dựng fc(G) theo công thức (2).

|  |  |
| --- | --- |
|  |  |

Tập đặc trưng cải tiến được tính bằng hợp của tập đặc trưng ban đầu và tập các đặc trưng nhóm

## Code

from \_\_future\_\_ import print\_function

import numpy as np

import matplotlib.pyplot as plt

import ekg\_data

from sklearn.cluster import KMeans

def sampling\_data(data, leg\_size, window\_size):

# This function is for time series data sampling

# leg\_size : Length of splited data

# window\_size : Sliding window size

samples = []

for pos in range(0,len(data),window\_size):

sample = np.copy(data[pos:pos+leg\_size])

if len(sample) != leg\_size:

continue

samples.append(sample)

return samples

def plot\_waves(waves,step):

# waves : waves to plot

# step : sampling rate to plot

plt.figure()

n\_graph\_rows = 3

n\_graph\_cols = 3

graph\_n = 1

wave\_n = 0

for \_ in range(n\_graph\_rows):

for \_ in range(n\_graph\_cols):

axes = plt.subplot(n\_graph\_rows, n\_graph\_cols, graph\_n)

axes.set\_ylim([-100,150])

plt.plot(waves[wave\_n])

graph\_n += 1

wave\_n += step

plt.tight\_layout()

plt.show()

def reconstruct(data, window, clusterer):

# reconstruct data with centeroid of clusterer

# data is input data

# window is window function that make (input data's starting point and ending point) to be zero

# Clusterer : scikit-learn Cluster model

window\_len = len(window)

slide\_len = window\_len/2

# data spliting with sliding lenght window\_len/2 ( some datas are overlapping )

segments = sampling\_data(data, window\_len, slide\_len)

reconstructed\_data = np.zeros(len(data))

# find nearest centroid among clusters for reconstruction

for segment\_n, segment in enumerate(segments):

segment \*= window

segment = np.reshape(segment,(1,window\_len))

nearest\_match\_idx = clusterer.predict(segment)[0]

nearest\_match = np.copy(clusterer.cluster\_centers\_[nearest\_match\_idx])

pos = segment\_n\*slide\_len

reconstructed\_data[pos:pos+window\_len] += nearest\_match

return reconstructed\_data

if \_\_name\_\_ == "\_\_main\_\_":

n\_samples = 8192

window\_len = 32

# to avoid too many data

data = ekg\_data.read\_ekg\_data('a02.dat')[0:n\_samples]

# create anomalous data

data\_anomalous = np.copy(data)

data\_anomalous[210:215] = 0

# window function is sin^2

window\_rads = np.linspace(0,np.pi,window\_len)

window = np.sin(window\_rads)\*\*2

segments = sampling\_data(data,window\_len,2)

segments\_anomalous = sampling\_data(data\_anomalous,window\_len,2)

windowed\_segments = []

for segment in segments:

segment \*= window

windowed\_segments.append(segment)

print("Clustering...")

clusterer = KMeans(n\_clusters=150)

clusterer.fit(windowed\_segments)

print("Reconstructing...")

reconstruction = reconstruct(data\_anomalous,window,clusterer)

error = reconstruction - data

print("Maximum error is ",max(error))

plt.figure()

n\_plot\_samples = 300

plt.plot(data[0:n\_plot\_samples],label="Anomalous data")

plt.plot(reconstruction[0:n\_plot\_samples],label="reconstruction")

plt.plot(error[0:n\_plot\_samples],label="error")

plt.legend()

plt.show()