|  |  |
| --- | --- |
| Lab 2 – Parallel K-means | Name: SHUYUE JIAID: 56846018 |

1. Python scikit-learn package is one of the most popular package in data mining and machine learning.
2. In particular, its k-means function has been widely adopted for data clustering:

<http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>

1. The following is the example code snippet from GitHub in running k-means:

import numpy as np

from sklearn.cluster import KMeans

from sklearn.datasets import make\_blobs

# Creating a sample dataset with 4 clusters

X, y = make\_blobs(n\_samples=100, n\_features=3, centers=4)

# Initializing KMeans with 4 clusters

kmeans = KMeans(init="k-means++", n\_clusters=4, n\_init=12)

# Fitting with inputs

kmeans = kmeans.fit(X)

# Labelling the clusters

labels = kmeans.labels\_

print(labels)

# Getting the cluster centers

C = kmeans.cluster\_centers\_

print(C)

1. Fill in the underlined space. Please feel free to try the code in your own machine for experiments.

Answer:

Python 2.7.16 (default, Aug 30 2021, 14:43:11)

[GCC Apple LLVM 12.0.5 (clang-1205.0.19.59.6) [+internal-os, ptrauth-isa=deploy on darwin

Type "help", "copyright", "credits" or "license" for more information.

>>> import numpy as np

>>> from sklearn.cluster import KMeans

>>> from sklearn.datasets import make\_blobs

>>> X, y = make\_blobs(n\_samples=100, n\_features=3, centers=4)

>>> kmeans = KMeans(init="k-means++", n\_clusters=4, n\_init=12)

>>> kmeans = kmeans.fit(X)

>>> labels = kmeans.labels\_

>>> print(labels)

[0 0 1 0 3 1 0 1 0 1 0 3 0 3 3 1 1 3 1 1 3 3 0 3 2 2 1 1 1 1 0 2 3 1 2 0 3

2 2 3 2 0 1 0 2 0 1 3 3 1 0 2 3 2 2 1 3 1 0 2 0 3 0 2 1 0 3 3 1 3 2 0 2 0

1 1 2 1 0 2 3 3 1 3 2 3 0 2 1 0 3 2 0 2 2 2 0 2 2 3]

>>> C = kmeans.cluster\_centers\_

>>> print(C)

[[-8.21396128 -1.83811964 8.72595069]

[-2.11273252 8.05654272 0.73589464]

[-3.98548093 -4.20428332 1.48986906]

[-0.23661739 1.16590183 5.41562605]]

1. If you read below carefully, you can see that there is an argument called “n\_jobs”. What is it?

<http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>

<https://github.com/scikit-learn/scikit-learn/blob/master/sklearn/cluster/k_means_.py>

Answer:

“n\_jobs” means that we can use parallel computation to execute one or more programs faster.

The basic principle of this “n\_jobs” is to use multiple threads to assign each sample to its closest center which may speed up the computation n times than single thread (one job).

1. Unfortunately, the argument “n\_jobs” does not give you the function to parallelize the algorithmic core of K-means which should be straightforward to be parallelized.
2. Therefore, we aim at parallelizing it for big data in this lab.
3. You are given the Python k-means code “Lab2given.py” and data “xclara.csv” in CANVAS.
4. It is observed that there are two code sections which can be parallelized as tabulated below.
5. Please read the documentation of the “joblib” package in Python and use it to parallelize k-means.
6. <https://github.com/joblib/joblib>

|  |  |
| --- | --- |
| Sequential Version | Parallel Version using joblib package |
| # Assigning each value to its closest cluster  for i in range(len(X)):  distances = dist(X[i], C)  cluster = np.argmin(distances)  clusters[i] = cluster | from joblib import Parallel, delayed  def assignment(x, C):  distances = dist(x, C)  cluster = np.argmin(distances)  ci = cluster  return ci  # Assigning each value to its closest cluster  clusters = Parallel(n\_jobs=2)(delayed(assignment)(X[i], C) for i in range(len(X))) |
| # Finding the new centroids by taking the average value  for i in range(k):  points = [X[j] for j in range(len(X)) if clusters[j] == i]  C[i] = np.mean(points, axis=0) | from joblib import Parallel, delayed  def centroidcompute(X, clusters, i):  points = [X[j] for j in range(len(X)) if clusters[j] == i]  return np.mean(points, axis=0)  # Finding the new centroids by taking the average value  b = Parallel(n\_jobs=2)(delayed(centroidcompute)(X, clusters, i) for i in range(k))  for i in range(k):  C[i] = b[i] |

1. Once you have finished the above, please feel free to try your parallel code on your machine.

~/Desktop » python Lab2given.py 1 ↵

Input Data and Shape

(3000, 2)

Initial Centroids

[[60. 59.]

[77. 53.]

[30. 21.]]

/Library/Python/2.7/site-packages/joblib/externals/loky/backend/semlock.py:217: RuntimeWarning: semaphore are broken on OSX, release might increase its maximal value

"increase its maximal value", RuntimeWarning)

/Library/Python/2.7/site-packages/joblib/externals/loky/backend/semlock.py:217: RuntimeWarning: semaphore are broken on OSX, release might increase its maximal value

"increase its maximal value", RuntimeWarning)

Final Centroid Values

From scratch done by us:

[[ 40.683628 59.715893]

[ 69.92419 -10.119641]

[ 9.478045 10.686052]]

From scikit-learn package:

[[ 40.68362784 59.71589274]

[ 9.4780459 10.686052 ]

[ 69.92418447 -10.11964119]]

1. (Optional) Alternatively, there is also a native python package called “multiprocessing” which could be interesting to try and replace the “joblib” package in Python for parallel data processing.
2. This is the end; please upload this sheet with your answers to the submission system.