作业 4
CS540 F24

作业目标

- 处理真实世界的数据
- 实施分层聚类
- 可视化聚类过程

您要对来自不同国家的社会经济数据进行分层聚类。每个国家由数据集中的一行定义。我们将使用一个数据子集，用一个特征向量来表示每个国家。在这项作业中，您必须用 6 个统计数据来表示一个国家："人口"、"净移民"、"GDP（人均美元）"、"识字率（%）"、"电话（每 1000 部）"和 "婴儿死亡率（每 1000 名新生儿）"。

将每个国家表示为 6 维特征向量（x_1 , ... , x_6 ）后，需要使用层次聚类（HAC）对国家进行聚类。通过聚类，我们可以直观地看出哪些国家具有相似的社会经济状况。**严禁使用 scipy.cluster.hierarchy.linkage()，否则本作业将得零分。**除此之外，您可以使用 python 内置标准库、scipy(除了 hierarchy.linkage)、numpy 和 matplotlib 中的任何内容。

计划概述

CSV 格式的数据可在文件 countries.csv 中找到。注意，本作业没有启动代码。如果您觉得有困难，请参考以往作业中的启动代码。您必须为本作业编写几个 python 函数。**(注意：函数的名称必须与写法完全一致**）以下是每个**函数**的高级描述，供参考：

1. load_data(filepath) - 接收包含 CSV 文件路径的字符串，并以 dict 列表的形式返回数据点。第 0.1 节
2. calc_features(row) - 从上一个函数加载的数据中获取一行字符串，然后按照上面的指定计算该国家的相应特征向量，并以形状为（6,）的 NumPy 数组形式返回。数组的 dtype 应为 float64。第 0.2 节

3. hac(features) - 使用（x_1，...，x_6）特征表示对国家进行单链路分层聚类，并返回一个表示聚类结果的 NumPy 数组。第 0.3 节

4. fig_hac(Z, names) - 可视化国家特征的分层聚类。第 0.4 节

5. normalize_features(features) - 接收特征向量列表并计算归一化值。输出应该是一个格式与输入相同的归一化特征向量列表。第 0.5 节

您可以根据需要实现其他辅助函数，但这些都是我们要测试的函数。尤其是，您的最终 python 文件只是一套函数，您不应该在函数之外运行代码。为了测试代码，你可能需要一个 "main "方法将所有代码整合在一起。请确保，要么删除任何运行在函数之外的测试代码，要么用 if name ==" main ": 将其包裹起来。这将在第 0.6 节中详细讨论。

计划详情

0.1 load_data(filepath)

总结。[10 分］

- 输入：字符串，要读取文件的路径。
- 输出：列表，其中每个元素都是一个 dict，代表读取的文件中的一行。

详细信息。

1. 读入参数 `filepath` 指定的文件。Python `csv` 模块中的 `DictReader` 对读取文件很有用。请注意，根据您的 Python 版本，它可能会返回 `OrderedDicts` 而不是普通的 `dicts`，因此如果有必要，请确保将它们转换为普通的 `dicts`。不会对数据进行类型转换；所有值都保留为字符串。
2. 返回一个字典列表，其中数据集中的每一行都是一个字典，列标题为键，行元素为值。我们会原样保留 CSV 文件中的所有列。
3. 您可以假定文件已经存在，并且是格式正确的 CSV。

0.2 calc_features(row)

总结。[10 分］

- 输入：代表一个国家的二进制数。
- 输出：形状为 (6,) 的 NumPy 数组，dtype 为 float64。第一个元素为 x_1 ，以此类推，第六个元素为 x_6 。

详情该函数将代表一个国家的 dict 作为输入，并计算出特征表示（x_1 , ... , x_6 ）。具体来说

1. x_1 = "人口
2. x_2 = "净移民

3. x_3 = "国内生产总值（人均美元）
4. x_4 = "识字率 (%)
5. x_5 ="电话（每 1000 部）
6. x_6 ="婴儿死亡率（每千名新生儿）"。

注意，dict 中的这些统计量可能不是浮点类型。在计算每个 x_i 时，确保将每个相关统计量转换为浮点型。返回一个 NumPy 数组，其中每个 x_i 的顺序为： x_1 , . ..$_6$该数组的形状应为 (6,)。数组的 dtype 应该是 float64。请记住，此函数只适用于一个国家，而不是您在 load_data 中加载的所有国家。请确保输出的数据结构与指定的数据类型完全一致，否则有可能导致点数大幅减少。

0.3 hac(features)

总结。[50 分]

- 输入：形状为 (6,) 的 NumPy 数组列表，其中每个数组都是第 0.2 节中计算的 (x_1 , ... , x_6) 特征表示。特征向量的总数，即输入列表的长度，为 *n*。请注意，我们会在不同的 *n* 上测试你的代码（如第 0.6 节所述）。

- 输出：对于任意 *i*，*Z*[*i,* 0] 和 *Z*[*i, 1*] 表示聚类算法*第 i 次*迭代中合并的两个聚类的索引。然后，*Z*[*i,* 2] = *d*（*Z*[*i,* 0]，*Z*[*i, 1*]）是在*第 i 次*迭代中合并的两个聚类之间的单链距离（这将是一个实数值，而不是像其他量那样的整数）。最后，*Z*[*i,* 3] 是合并后形成的新聚类的大小，即该聚类中的国家总数。需要注意的是，原国家群组的索引为 0, .在算法的*第 i 次*迭代（*i* ≥ 1）中构建的聚类的聚类索引为 (*n* - 1) + *i*。

距离使用单链，执行讲座中详细介绍的分层聚类算法。使用标准欧氏距离函数计算两点之间的距离。你可以实现自己的距离函数或使用 numpy.linalg.norm()。其他距离函数可能无法达到预期效果，因此请在 Gradescope 上检查您的结果！如果您使用了软件包中的距离函数而导致分数减少，请自行承担责任。

概要。以下是实现 hac() 的一种可能路径

1. 给每个起始数据点编号，从 0 到 *n* - 1。这就是它们的原始群集编号。

2. 创建一个 (*n* - 1) × 4 数组或列表。逐行遍历该数组/列表。对于每一行

(a) 确定最接近的两个群组，并将它们的编号放入第一个群组中。

和第二行元素 $Z[i, 0]$ 和 $Z[i, 1]$。所列的第一个元素 $Z[i$、
0] 应该是两个群组索引中较小的一个。

(b) 两个聚类之间的单链距离归入第三行元素 $Z[i, 2]$ 中。

(c) 第四个元素 $Z[i, 3]$ 代表该组国家的总数。

如果合并一个包含多个国家的分组，其索引（第一或第二行元素的索引）为 n + 创建分组时的行索引。

3. 在返回数据结构之前，如果还不是 NumPy 数组，则将其转换为 NumPy 数组
。

当 n 较大时，为使该方法高效运行，应在函数开始时保留一个距离矩阵，以避免重新计算各点之间的距离。在合并 i 和 j 聚类的迭代之后，你可以

1. 在末尾添加一列和一行，以存储到新群集 {i, j} 的距离：使用单链距离，从现有群集 k 到新群集 {i, j} 的距离是群集 k 到 i 的距离和群集 k 到 j 的距 离 之间的最小值；

2. 删除 i 和 j 列和行（或直接将这些列和行中的值改为-1）；

3. 找出距离最小的两个聚类（它们是更新后的距离矩阵中最小非负非对角项的列和行索引），并在下一次迭代中合并这两个聚类。

您应该能够通过 CSV 文件中的所有国家高效地运行 HAC。例如，如果您有一个名为 distance_matrix 的 NumPy 数组，那么 distance_matrix[3,4] 就等于特征输入中索引 3 和索引 4 的国家之间的欧氏距离。你可以将输出结果与 scipy.spatial.distance_matrix 进行比较，但请注意我们在这项任务中使用的是欧氏距离。

打破平局。在选择下两个要合并的聚类时，我们会选择单链距最小的一对。如果多对聚类的单链距离相同，我们就需要额外的标准来进行选择。为此，我们在指数上采用了如下的决胜规则：假设 (i_1, j_1)，...。,(i_h, j_h) 是一对距离相等的群集指数，即 $d(i_1, j_1) = \cdots = d(i_h, j_h)$，并假设所有 t 中 $i_t < j_t$（因此每对指数都是排序的）。如果存在多个第一索引为 i 的配对，我们需要进一步区分它们。假设这些配对是（i，t_1），（i，t_2），.....等等。为了在这些配对中打破平局，我们要选出第二索引最小的配对，即这些配对中 t 值最

小的配对。请注意，这种打破平局的策略可能不会产生与 linkage() 完全相同的结果。

0.4 羽状复叶

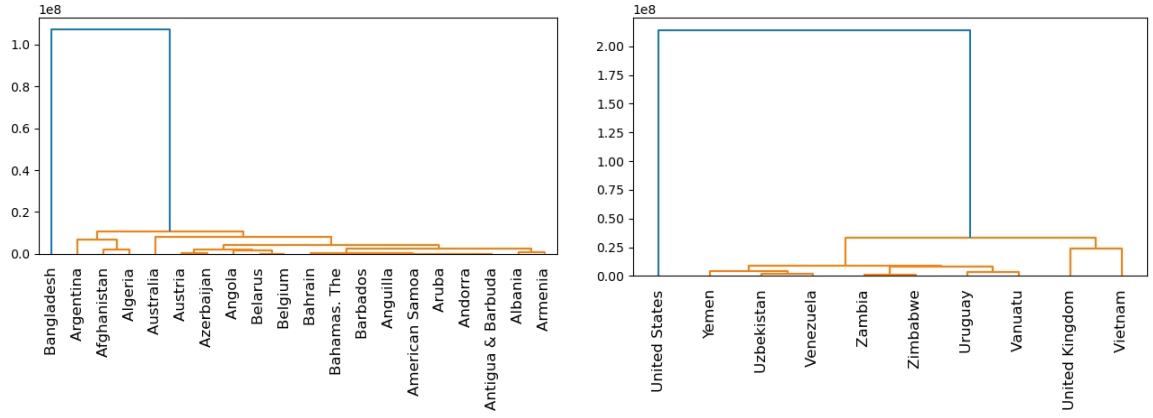
总结。[10 分]

- 输入hac 输出的 NumPy 数组 *Z*，以及长度为 *n* 的国名对应字符串列表。
- 输出：Matplotlib 图，其中的图形可视化分层聚类。

详细信息。首先用 "fig = plt.figure() "初始化一个图形。然后使用 dendrogram - SciPy v1.14.1 Manual 中 SciPy 模块中的 dendrogram，使用 `labels=names` 和 `leaf_rotation = ??`。您的绘图可能会切断 x 标签。要使图形看起来像下面的示例，需要在图上调用 tight_layout()。

前 20 个国家和最后 10 个国家的可视化情况

N = 20N = 10



讨论。请注意，上述数据几乎完全是按人口对国家进行分组的。人口的数值远远大于数据中其他列的数值。因此，人口对特征向量之间欧氏距离的贡献不成比例。理想情况下，所有六个统计量都应有助于聚类，因此您将在下一节创建一个函数来对数据进行归一

化处理。

0.5 normalize_features(features)

总结。[20 分]

- 输入：此函数的输入将是 calc_features 输出的特征向量列表。每个特征向量都是一个 NumPy 数组，其形状为 (6,)，dtype 为 float64。
- 输出输出格式应与输入格式相同：形状为 (6,) 和 dtype 为 float64 的 NumPy 数组列表。不过，特征向量中的统计值应替换为归一化值。

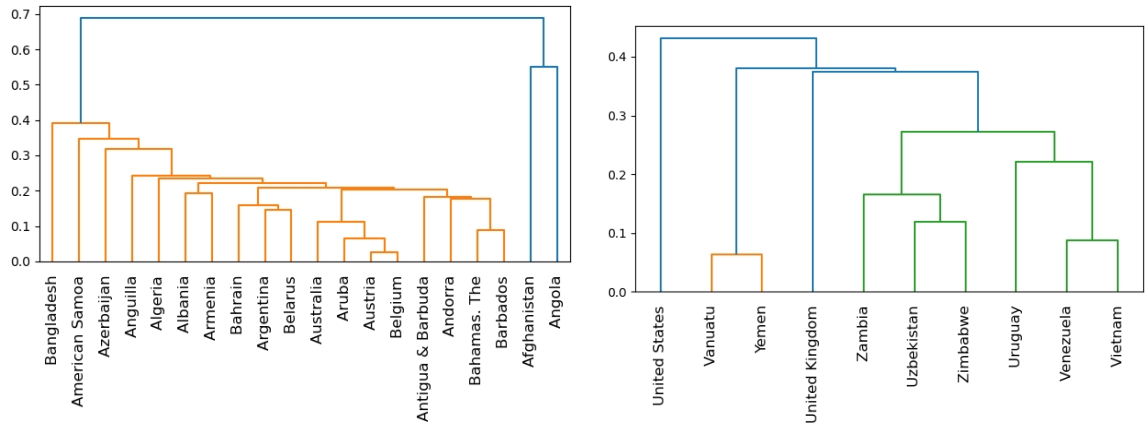
详细信息。对于 6 个统计量中的每一个，使用下面的公式计算每个数据点的归一化特征值。

$$\frac{x - col\_min}{col\_max - col\_min}$$

x 表示原始值，*col_min* 表示该列的最小值，*col_max* 表示该列的最大值。因此，统计量中的每个值都会在 [0, 1] 之间。在归一化数据上应用 HAC 后，应绘制出下面的曲线图。

前 20 个国家和最后 10 个国家的可视化情况

N = 20N = 10



讨论情况。这些结果似乎将社会经济地位相似的国家聚集在一起。对数据进行归一化处理后，所有统计数据的取值范围相似，因此它们对欧氏距离的贡献相同。

0.6 测试

要测试你的代码，请尝试在主方法或 Jupyter 笔记本中运行以下几行，并选择不同的 n，注意如果你想要最终的 n 行，应该使用 [-n:] 替换 [:n]：

```
data = load_data("countries.csv")
country_names = [row["Country"] for row in
data] features = [calc_features(row) for row in
data] 国家名称 = [row["Country"] for row in data

features_normalized = normalize_features(features)
n = ?
Z_raw = hac(features[:n])
Z_normalized =
hac(features_normalized[:n]) fig =
fig_hac(Z_raw, country_names[:n])
plt.show()
```

为了帮助您测试代码，我们在 "output.txt "中提供了 hac 使用前 50 个国家规范化数据的正确输出。请注意，我们仍然使用所有国家/地区的数据对特征进行归一化处理，但我们为 hac 选择了前 50 个国家/地区的归一化数据。如果您想将输出结果与 "output.txt "进行比较，请严格按照上述测试代码进行。您也可以将 fig_hac 的输出结果与本文提供的图表进行比较。

对于 hac 函数，我们将测试 *n 的*大小值，最大值为 204。对于整个数据集，您的代码运行时间不应超过 15 秒。我们将测试 n
<= 30 为 fig_hac。

额外部分：在世界地图上展示它们（不计分）

为了更好地理解这种聚类，我们在 **function.py** 中提供了一个名为 *world_map* 的函数，用于在世界地图上直观显示聚类。完成作业的主要部分后，您可以将代码复制到 hw4.py 中。

要使用该功能，我们首先需要使用以下命令安装 *geopandas* 软件包。

```
pip install geopandas==0.14.4
```

注意：如果安装的版本 >= 1.0.0，则需要修改代码，因为 geopandas.dataset 模块已被弃

用，并在 1.0.0 中移除。

功能概述

- 输入hac 输出的 NumPy 数组 *Z*、长度为 *n* 的国名对应字符串列表以及聚类数 *K*

  。

- 输出：Matplotlib 图形，将聚类可视化 用不同的输入进行尝试，

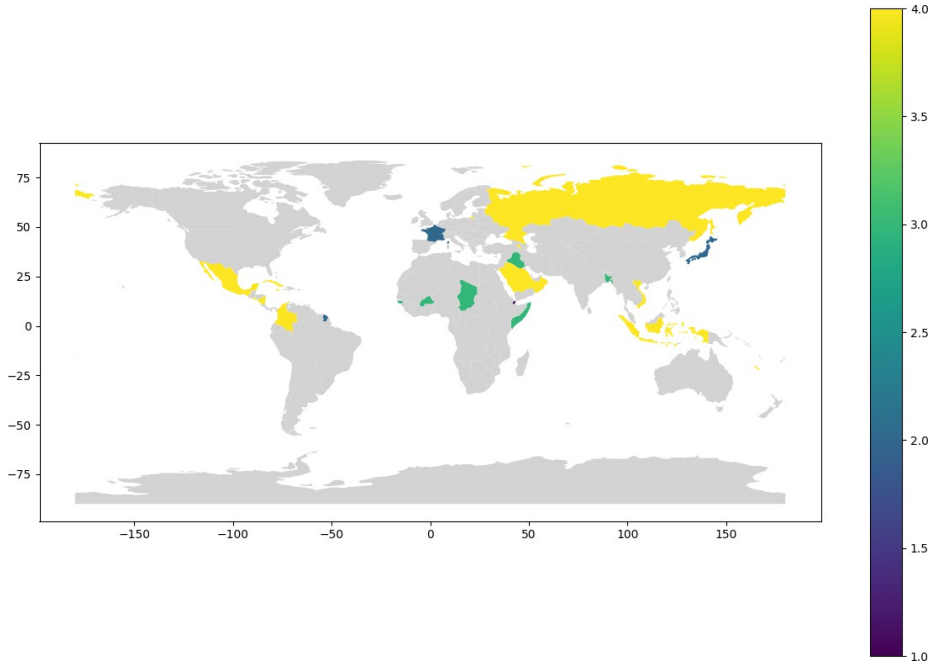
看看自己的情况！
例如，您可以使用以下代码随机选择国家并运行

功能

```
导入随机
random_indices = random.sample(range(0, len(names)), 30)

random_names = [names[i] for i in random_indices] 随机指数

random_features = [features_normalized[i] for i in

random_indices] Z = hac(random_features)

world map(Z, random names, 5)
```

注意：由于软件包中的国家名称集与我们提供的文件不完全相同，因此某些国家可能无法正确显示。您在使用代码时可以忽略这个问题。

以下是代码输出示例，请随意修改代码和 world_map 函数中的参数，以改变函数的颜色或外观。



提交详情

- 请在 Gradescope 上提交名为 hw4.py 的文件。

- 所有代码都应包含在函数或 "if _____名称___=="__main__":'
- 确保在提交前删除所有调试输出。

- 作业截止时间为 10 月 7 日星期一上午 9:30。