HW05

February 26, 2022

DSCC 465

Assignment 5

Uzair Tahamid Siam

---

[1]:
```
import numpy as np
import pandas as pd
import random
import matplotlib.pyplot as plt
```

Q1

[2]:
```
df = pd.read_excel('country_information.xlsx')
```

0.1 a)

The attributes in the dataset each represent some metric that can help us compare the economic or the civil state of the countries. For example,

1) `gini_index` represents the wealth gap between the richest and the poorest in a nation. Having a high value for `gini_index` means there is more chance to have civil unrest but also that it is likely that the economy of such a country is worse than that with a lower `gini_index`.

2) `effective_coverage_of_health_services_index` essentially represents the quality of universal healthcare in the country. In most cases (barring the US whose healthcare is a joke) 1^{st} world countries should have a higher value for this relative to 3^{rd} world countries. But a lower value could at the same time also mean that there is more 'civil unrest' in the people who might want better healthcare.

This trend can be seen throughout all the attributes and there is an interesting pattern that can be observed throughout the entire dataset that relates to the differences in lifestyle quality/economic/civil state of the countries in the data.

0.2 b)

[3]:
```
from sklearn.preprocessing import MinMaxScaler

mms = MinMaxScaler()
df.iloc[:, 1:] = mms.fit_transform(df.iloc[:, 1:])
```

```
df.to_excel('country_information_normalized.xlsx')
```

[4]:
```
df.head()
```

[4]:
```
       country  gini_index  corruption_perceptions_index  freedom_house  \
0  Afghanistan    0.415418                      0.092105            0.0
1      Albania    0.239518                      0.315789            0.5
2      Algeria    0.096609                      0.315789            0.0
3    Argentina    0.487058                      0.394737            1.0
4    Australia    0.270142                      0.855263            1.0

        hdi  press_freedom  democracy_economist  populism  \
0  0.000000           0.25             0.000000  0.441839
1  0.666667           0.50             0.333333  0.562931
2  0.666667           0.25             0.333333  0.550136
3  1.000000           0.50             0.666667  1.000000
4  1.000000           0.75             1.000000  0.683351

   effective_coverage_of_health_services_index  trust_in_news_media  \
0                                     0.109375             0.501952
1                                     0.593750             0.306081
2                                     0.515625             0.257206
3                                     0.453125             0.459459
4                                     0.890625             0.594595

   trust_in_government  trust_in_science  colonized
0             0.743412          0.153846        0.0
1             0.519163          0.153846        1.0
2             0.738070          0.057692        1.0
3             0.153846          0.365385        1.0
4             0.323077          0.596154        1.0
```

Q2

0.2.1 Initialization method for K-Means++

Choose one center uniformly at random among the data points.

For each data point x not chosen yet, compute D(x), the distance between x and the nearest center that has already been chosen.

Choose one new data point at random as a new center, using a weighted probability distribution where a point x is chosen with probability proportional to $(D(x))^2$.

Repeat Steps 2 and 3 until k centers have been chosen.

Now that the initial centers have been chosen, proceed using standard k-means clustering.

0.2.2 The 5 Steps in K-means++ Clustering Algorithm

Step 1. Get the initialized centroids from method above

Step 2. Find the distance (Euclidean distance for our purpose) between every data point with the K centroids.

Step 3. Assign each data point to the closest centroid according to the distance found.

Step 4. Update centroid location by taking the average of the points in each cluster group.

Step 5. Repeat the Steps 2 to 4 till the centroids don't change.

[5]:

```python
# initialization algorithm
def initialize(data, k):
    '''
    Initializes the centroids for K-means++

    Parameters
    -----------
    data:
        numpy array of data points
    k:
        number of clusters

    Returns
    -----------
    centroids:
        The initial centroid positions
    '''
    random.seed(265)
    centroids = []

    # randomly initialize the first centroid
    centroids.append(data[random.randrange(0,len(data),1)])

    ## compute remaining k - 1 centroids
    for c_id in range(k - 1):
        # for every iteration find the shortest eucledian distance between centroids and points
        dist = np.array([min([np.inner(c-x,c-x) for c in centroids]) for x in data])

        # find the probability associated with the distances
        probs = dist/dist.sum()

        # use the probability to find the the centroid position
        centroids.append(max(random.choices(data, probs)))

    return centroids
```

[6]:

```
from scipy.spatial.distance import cdist

def kmeans(x,k=6, no_of_iterations=100):
    '''
    Returns the labels associated to every data point

    Parameters
    -----------
    x:
        numpy array of data points
    k:
        number of clusters
    no_of_iterations:
        max number of iterations to reach convergence

    Returns
    -----------
    closest_centroids:
        The labeled dataset
    '''
    centroids = initialize(x, k)

    #finding the distance between centroids and all the data points
    distances = cdist(x, centroids ,'euclidean') #Step 2

    #Centroid with the minimum distance
    closest_centroids = np.array([np.argmin(i) for i in distances]) #Step 3

    #Repeating the above steps for a defined number of iterations while the prev␣
↪and new centroids are the same
    for _ in range(no_of_iterations):
        centroids = []
        for cluster_label in range(k):

            #Updating Centroids by taking mean of Cluster it belongs to for each␣
↪cluster
            cluster_mean = x[closest_centroids==cluster_label].mean(axis=0)
            centroids.append(cluster_mean)

        centroids = np.vstack(centroids) #Updated Centroids

        distances = cdist(x, centroids ,'euclidean')
        new_centroids = np.array([np.argmin(i) for i in distances])

        # checks convergence
        if (closest_centroids == new_centroids).all():
            break
```

```
        else:
            closest_centroids = new_centroids


    return closest_centroids
```

Q3

0.3 a)

[7]:
```
df['kmeans_label'] = kmeans(df.iloc[:, 1:].values)
```

[8]:
```
df.head()
```

[8]:
```
       country  gini_index  corruption_perceptions_index  freedom_house  \
0  Afghanistan    0.415418                      0.092105            0.0
1      Albania    0.239518                      0.315789            0.5
2      Algeria    0.096609                      0.315789            0.0
3    Argentina    0.487058                      0.394737            1.0
4    Australia    0.270142                      0.855263            1.0

        hdi  press_freedom  democracy_economist  populism  \
0  0.000000           0.25             0.000000  0.441839
1  0.666667           0.50             0.333333  0.562931
2  0.666667           0.25             0.333333  0.550136
3  1.000000           0.50             0.666667  1.000000
4  1.000000           0.75             1.000000  0.683351

   effective_coverage_of_health_services_index  trust_in_news_media  \
0                                     0.109375             0.501952
1                                     0.593750             0.306081
2                                     0.515625             0.257206
3                                     0.453125             0.459459
4                                     0.890625             0.594595

   trust_in_government  trust_in_science  colonized  kmeans_label
0             0.743412          0.153846        0.0             5
1             0.519163          0.153846        1.0             2
2             0.738070          0.057692        1.0             3
3             0.153846          0.365385        1.0             0
4             0.323077          0.596154        1.0             1
```

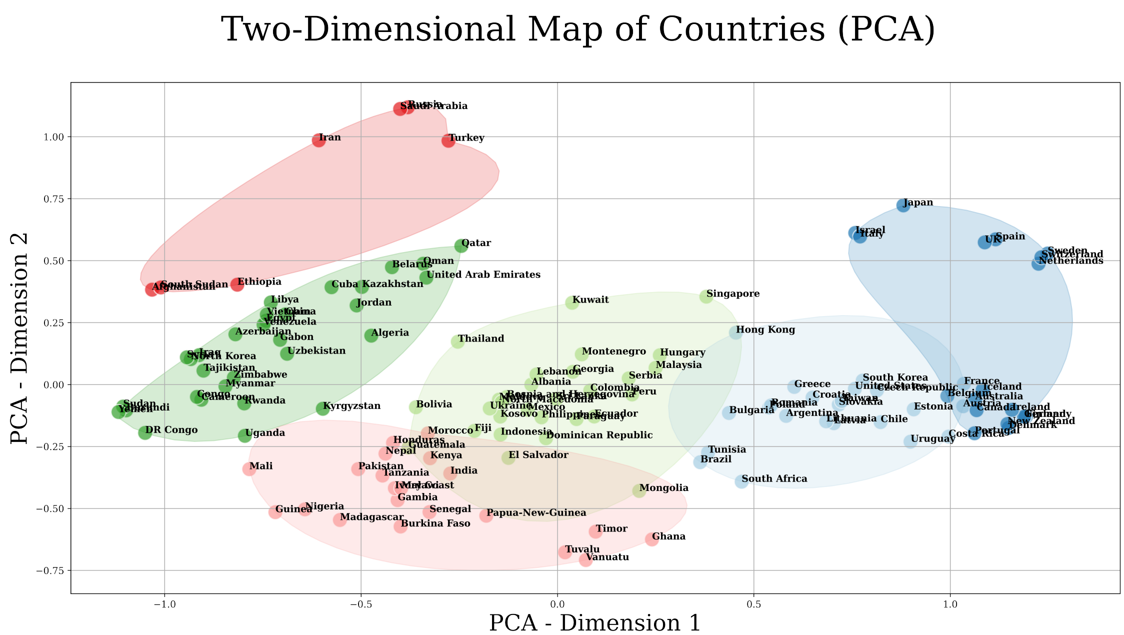
0.4 b)

[9]:
```
from sklearn.decomposition import PCA

pca = PCA(n_components=2, random_state=265)
principalComponents=pca.fit_transform(df.iloc[:, ~df.columns.isin(['country',␣
↪'kmeans_label'])])
df['pca_dim_1'], df['pca_dim_2'] = principalComponents[:,0],␣
↪principalComponents[:, 1]
```

Q4

[10]:
```
import matplotlib.pyplot as plt
import matplotlib as mpl
import seaborn as sns

mpl.rcParams['figure.dpi'] = 600
plt.rcParams['figure.figsize'] = (20.0, 10.0)
plt.rcParams['font.family'] = "serif"
plt.rcParams['font.size'] = 10

df = df
pal = sns.color_palette("Paired")[:len(set(df['kmeans_label']))]
p1 = sns.scatterplot(x="pca_dim_1", y='pca_dim_2', hue='kmeans_label', palette =␣
↪pal, data=df, s=250, alpha=0.7, legend=False)

#For each point, we add a text inside the bubble
for line in range(0,df.shape[0]):
     p1.text(df.pca_dim_1[line], df.pca_dim_2[line], df.country[line],␣
↪horizontalalignment='left', size='medium', color='black', weight='semibold')

plt.suptitle('Two-Dimensional Map of Countries (PCA)', fontsize=36)
plt.xlabel('PCA - Dimension 1', fontsize=24)
plt.ylabel('PCA - Dimension 2', fontsize=24)

from scipy import interpolate
from scipy.spatial import ConvexHull

for i in df.kmeans_label.unique():
    # get the convex hull
    points = df[df.kmeans_label == i][['pca_dim_1', 'pca_dim_2']].values
    hull = ConvexHull(points)
    x_hull = np.append(points[hull.vertices,0],
                       points[hull.vertices,0][0])
    y_hull = np.append(points[hull.vertices,1],
                       points[hull.vertices,1][0])
```

```
    # interpolate
    dist = np.sqrt((x_hull[:-1] - x_hull[1:])**2 + (y_hull[:-1] - y_hull[1:])**2)
    dist_along = np.concatenate(([0], dist.cumsum()))
    spline, u = interpolate.splprep([x_hull, y_hull],
                                    u=dist_along, s=0)
    interp_d = np.linspace(dist_along[0], dist_along[-1], 50)
    interp_x, interp_y = interpolate.splev(interp_d, spline)
    # plot shape
    plt.fill(interp_x, interp_y, '--', c=pal[i], alpha=0.2)


plt.grid()
plt.show()
```



Q5

[11]:
```
countries = {i+1: df[df['kmeans_label']==i].country.values for i in range(5)}
countries_labeled_df = pd.DataFrame({ key:pd.Series(value) for key, value in␣
 ↪countries.items() })
countries_labeled_df
```

[11]:
```
             1           2                        3            4  \
0    Argentina   Australia                  Albania      Algeria
1      Austria     Belgium                  Bolivia   Azerbaijan
```

2	Brazil	Canada	Bosnia and Herzegovina	Belarus
3	Bulgaria	Denmark	Colombia	Burundi
4	Chile	Finland	Dominican Republic	Cameroon
5	Costa Rica	Germany	Ecuador	China
6	Croatia	Iceland	El Salvador	Congo
7	Czech Republic	Ireland	Fiji	Cuba
8	Estonia	Israel	Georgia	DR Congo
9	France	Italy	Guatemala	Egypt
10	Greece	Japan	Hungary	Gabon
11	Hong Kong	Netherlands	Indonesia	Iraq
12	Latvia	New Zealand	Kosovo	Jordan
13	Lithuania	Portugal	Kuwait	Kazakhstan
14	Poland	Spain	Lebanon	Kyrgyzstan
15	Romania	Sweden	Malaysia	Libya
16	Slovakia	Switzerland	Mexico	Myanmar
17	South Africa	UK	Moldova	North Korea
18	South Korea	NaN	Mongolia	Oman
19	Taiwan	NaN	Montenegro	Qatar
20	Tunisia	NaN	North Macedonia	Rwanda
21	United States	NaN	Paraguay	Sudan
22	Uruguay	NaN	Peru	Syria
23	NaN	NaN	Philippines	Tajikistan
24	NaN	NaN	Serbia	Uganda
25	NaN	NaN	Singapore	United Arab Emirates
26	NaN	NaN	Sri Lanka	Uzbekistan
27	NaN	NaN	Thailand	Venezuela
28	NaN	NaN	Ukraine	Vietnam
29	NaN	NaN	NaN	Yemen
30	NaN	NaN	NaN	Zimbabwe

	5
0	Burkina Faso
1	Gambia
2	Ghana
3	Guinea
4	Honduras
5	India
6	Ivory Coast
7	Kenya
8	Madagascar
9	Malawi
10	Mali
11	Morocco
12	Nepal
13	Nigeria
14	Pakistan
15	Papua-New-Guinea

```
16         Senegal
17        Tanzania
18           Timor
19          Tuvalu
20         Vanuatu
21             NaN
22             NaN
23             NaN
24             NaN
25             NaN
26             NaN
27             NaN
28             NaN
29             NaN
30             NaN
```

0.5 a)

Which countries seem to be similar? Why do you think these countries are clustered together?

To a large degree, my initial hypothesis in **Q1** is represented in the clustering above. One can see that countries that are thought alike in terms of their economy or their civil state (how happy or sad people with their government) are clustered together. E.g. In `Table 1` countries such as Germany, Japan, Switzerland, UK and Sweden are all grouped together and these countries are known to have a good economy. And you can also see that countries like *Nepal*, *India*, *Pakistan* alongside others in Cluster-2 are grouped together as they all have similar economic standing. On another note, countries like *North Korea*, *China*, *Syria*, *Zimbabwe* are grouped together which makes sense given that these countries have a similar type of leadership (weak in democracy).

0.6 b)

If you run the kmeans++ algorithm more than once, do you think the results will change?

Assuming that the random seed is not set to a specific value, the kmeans++ should definitely result in changes as we are still choosing the initial point randomly.

0.7 c)

(Subjectively speaking) Do you think this is an accurate clustering of the countries? Would the results change greatly if we had different social/economic variables?

Given my initial hypothesis that the dataset can be used to cluster countries using their different economic/civil states represented by the various attributes, some of the clusters definitely do not look accurate. E.g. In `Table 1` you can see that cluster 1 contains *United States* and *South Korea* but also *Brazil* and *Tunisia*. It is hard to imagine, given the attributes in the dataset, for these countries to somehow belong to the same cluster. However, if we look at Cluster 2, all the countries seem to be *fairly* similar in terms of their economic and civil states.

I think the results would most definitely change if we had other attributes given the clustering depends on the attribute values themselves.

0.8 d)

Do you think PCA may have affected the results at all? In other words, if we had a different number of principle components, would our visual interpretation be different?

PCA definitely affects the result we have plotted. Projecting higher dimensional data into a lower dimensional space casues the points to be closer together than they originally were and vice-versa. So, it is safe to say that if we were to have a different number of principle axes, n, we would get different results.

If n > 2, our points would be farther away from each other and if n < 2, our points would be closer to each other on the plot. If the points were already close to each other however, 'how much' the results would change would be minuscle and the same can be said for the other case.