Practical Machine Learning Assignment Report: Unsupervised Learning Methods Comparison

Matei Bejan

January 11^{th}, 2020

1 Goal

My goal for this project is to compare five clustering algorithms by applying them on the World Development Indicators dataset. The algorithms are: KMeans, DBSCAN, Mean Shift, Hierarchical Clustering and Birch Clustering.

The libraries used in this project are numpy, scipy, pandas, sklearn, fancyimpute and matplotlib.

2 Dataset

The data set I have used for this project is the World Development Indicators published by World Bank and forwarded as a .csv file through kaggle. The data set contains 1346 social and economic indicators for 247 countries and geopolitical groups, collected over 55 years, from 1960 to 2015. Thus, it presents a total of 13585 samples.

Please note that the data set's details given above are determined after the preprocessing step, as the data requires some prior restructuring. The initial data will discussed in **3. Restructuring the data**.

3 Restructuring the data

As specified earlier, the initial dataset is unsuitable for modelling and thus requires some prior restructuring. As it can be seen in figure 1, the initial data presents one sample per feature per country per year. Moreover, there is a noticeable amount of unnecessary data, namely *CountryCode* and *IndicatorCode*.

My first action was to parse the data set and to group all features by year and country. For this I have dropped the *CountryCode* and *IndicatorCode* columns and added a column for every feature. Next, I have discarded the *Value* column and placed the sample in its respective column. This way each sample will contain all indicators observed in a certain year, as in figure 2.

	CountryName	CountryCode	IndicatorName	IndicatorCode	Year	Value
0	Arab World	ARB	Adolescent fertility rate (births per 1,000 wo...	SP.ADO.TFRT	1960	1.335609e+02
1	Arab World	ARB	Age dependency ratio (% of working-age populat...	SP.POP.DPND	1960	8.779760e+01
2	Arab World	ARB	Age dependency ratio, old (% of working-age po...	SP.POP.DPND.OL	1960	6.634579e+00
3	Arab World	ARB	Age dependency ratio, young (% of working-age ...	SP.POP.DPND.YG	1960	8.102333e+01
4	Arab World	ARB	Arms exports (SIPRI trend indicator values)	MS.MIL.XPRT.KD	1960	3.000000e+06
5	Arab World	ARB	Arms imports (SIPRI trend indicator values)	MS.MIL.MPRT.KD	1960	5.380000e+08
6	Arab World	ARB	Birth rate, crude (per 1,000 people)	SP.DYN.CBRT.IN	1960	4.769789e+01
7	Arab World	ARB	CO2 emissions (kt)	EN.ATM.CO2E.KT	1960	5.956399e+04
8	Arab World	ARB	CO2 emissions (metric tons per capita)	EN.ATM.CO2E.PC	1960	6.439635e-01
9	Arab World	ARB	CO2 emissions from gaseous fuel consumption (%...	EN.ATM.CO2E.GF.ZS	1960	5.041292e+00

Figure 1: Initial structure of the WDI data set.

The script that handles the restructuring was written separately and can be accessed here. The reason for which I took the decision of separating the two concerns was that, although simple and pretty straight-forward, the restructuring process is time-consuming, as it requires parsing over 5.5 million samples from the initial WDI dataset.

	CountryName	Adolescent fertility rate (births per 1,000 women ages 15-19)	Age dependency ratio (% of working-age population)	Age dependency ratio, old (% of working-age population)	Age dependency ratio, young (% of working-age population)	Birth rate, crude (per 1,000 people)	CO2 emissions (kt)	CO2 emissions (metric tons per capita)	CO2 emissions from liquid fuel consumption (% of total)	CO2 emissions from liquid fuel consumption (kt)	Death rate, crude (per 1,000 people)
0	Afghanistan	162.7380	101.094930	4.438311	96.656619	49.029	2676.910	0.221827	71.506849	1914.174	15.555
1	Afghanistan	163.3270	101.007981	4.464953	96.543028	48.896	2493.560	0.194971	71.176471	1774.828	15.008
2	Afghanistan	163.9160	100.880590	4.496947	96.383643	48.834	1426.463	0.103776	67.352185	960.754	14.524
3	Afghanistan	164.1812	100.881545	4.537744	96.343801	48.839	1375.125	0.092761	67.200000	924.084	14.101
4	Afghanistan	164.4464	100.847802	4.589312	96.258490	48.898	1320.120	0.083184	67.222222	887.414	13.736
5	Afghanistan	164.7116	100.476266	4.654247	95.822019	48.978	1268.782	0.075646	67.052023	850.744	13.421
6	Afghanistan	164.9768	101.231924	4.662978	96.568946	49.039	1199.109	0.068592	66.972477	803.073	13.146
7	Afghanistan	165.2420	101.976746	4.651200	97.325547	49.036	1114.768	0.061814	66.776316	744.401	12.894
8	Afghanistan	161.4432	102.539894	4.624247	97.915647	48.930	1056.096	0.057051	67.013889	707.731	12.651
9	Afghanistan	157.6444	102.925568	4.588172	98.337396	48.699	832.409	0.043723	60.352423	502.379	12.406

Figure 2: Restructured WDI data in a modelling-compatible format.

4 Missing data imputation

I applied the SoftImpute function in order to infer the missing data. This imputation step is preceded by a bi-scaling procedure. Bi-scaling is equivalent to a double normalization of the data through iterative estimation of row/column means and standard deviations.

Before imputation, however, I dropped those features that contained only NaN values.

5 Dimensionality Reduction

Given the high dimensionality of the data - over 1300 features - I chose to PCA it down to only the most informative ones. Other reasons for using PCA are the fact that this clustering process is not part part of a larger architecture, thus there isn't a goal-feature in respect to which we could filter out features, and the extremly long time it took to explore the parameter grid for each algorithm and to extract the best parameters.

I've used PCA to reduce the data's dimensions down to 65. These offer an overall explained variance of 0.85. Adding more features does not lead to a better variance-dimensionality trade-off.

6 Modelling

I've used four approaches towards computing the optimal number of clusters for each algorithm:

1. **Elbow point.** An heuristic method which looks at the percentage of variance explained as a function of the number of clusters. Given a variance vs. no. of clsuters plot, the marginal gain given by increasing the no. of clsuters will decrease at some point, signaling an elbow point.

2. **Silhouette.** A measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). Silhouette takes values in $[-1, 1]$, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \ ,$$

   where $a(i)$ is the mean distance from i to all j, $i, j \in C(i)$ and $b(i)$ is the minimum distance between i and j, $i \in C(i), j \notin C(i)$.

3. **Intracluster point scatter.** Points within the same cluster must be close to one another, $W(C)$ should be minimised:

$$W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} d(x_i, x_{i'})$$

4. **Intercluster point scatter.** Points from different clusters must be far from each oth er, $B(C)$ should be maximised:

$$B(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')\neq k} d(x_i, x_{i'})$$

We can reach our optimum by maximising $B(C)$ or minimising $W(C)$. I tried maximizing the following combined metric: $B(C) + \log W(C)^{-1}$. This way we take into consideration both metrics, but place more emphasis on $B(C)$.

6.1 K-Means

The K-Means algorithm that partitions the data points into a fixed number of clusters K. Being a centroid-based method, each cluster is represented by a centroid and every point is assigned to the cluster with respect to the nearest centroid. K-Means computes the clusters by minimizing the sum of Euclidean distances between feature vectors the sample points and the centroids.

Tested K-Means with $k \in [2, 100]$ and chose the best number of clusters through via both elbow, silhouette and intercluster-intracluster distance methods.

6.1.1 Elbow Method

As we can observe in the figure below, the elbow point seems to be around $k = 25$. The metrics are:

- Silhouette: 0.12396712521164671
- Interclsuter: 2653262374.296015
- Intracluster: 84178022.83436516

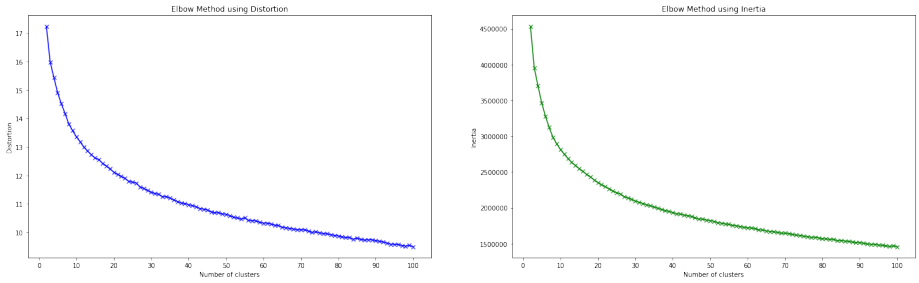

Figure 3: Elbow point graphs of distortion and inertia for K-Means.

In figure 16 one can observe two types of clusters: tail clusters, which present an elongated shape, and more bloated head clusters. There is also a considerable outlier count in between clusters.

The head clusters' centroids seem to be rather equidistant when compared to their tail counterparts. The border between the head and tail contains "flat" groupings.

6.1.2 Intercluser-Intraclsuter Metric Method

Optimizing the intercluser-intracluster distance has yielded an optmimum of $k = 98$. The metrics are:

- Silhouette: 0.1164869467002419

- Interclsuter: 2718419162.859649
- Intracluster: 19034206.50818126

Figure 17 represents our clusters and their respective centroids. Clusters are much more compact; the elongated shapes show a clear structural trend in the tail's data, meanwhile the head contains much more accurate outlier clusters. There is also a much more well-defined transition from the tail trend to the head trend in the middle clusters.

6.1.3 Silhouette Method

The number of clusters to maximize the Silhouette score has been computed automatically. The computation has yielded $k = 2$. The metrics are:

- Silhouette: 0.27215402846032716
- Interclsuter: 1418772371.597972
- Intracluster: 1318680997.7698603

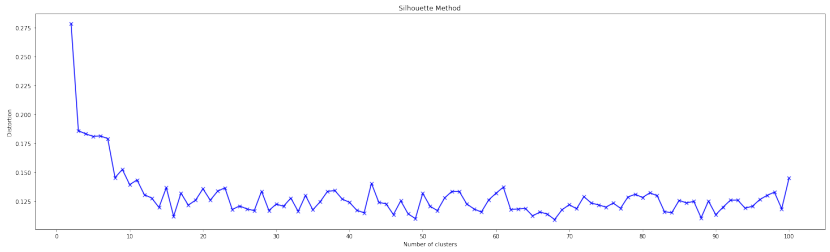

Figure 4: Silhouettes for K-Means.

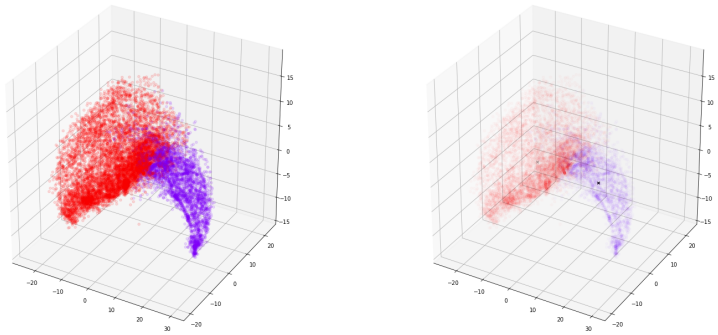

Figure 5: K-Means Silhouette clusters.

6.2 BDSCAN

Compared to K-Means, density-based clustering algorithms looks for how many neighbors a point within has within a given radius, thus not being dependent on being given an apriori number of clusters.

For DBSCAN I've used the Intercluster-Intracluster distance to compute the optimum epsilon value, and it yielded $eps = 10$. The metrics are:

- Interclsuter: 770415164.1513163
- Intracluster: 1851433050.8059065

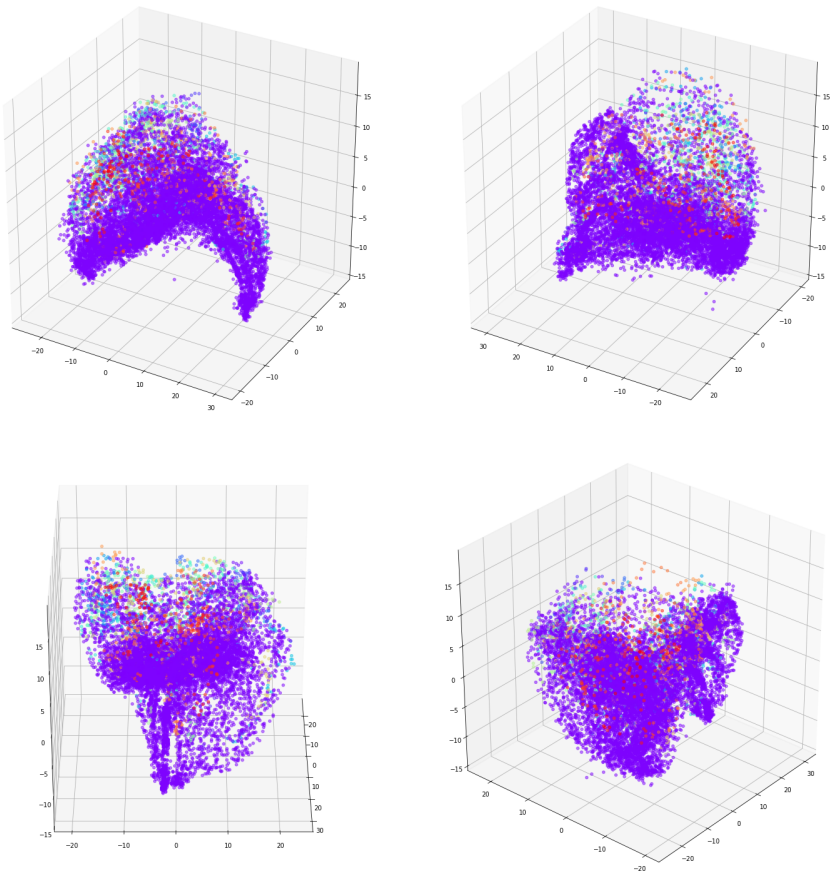

Figure 6: DBSCAN clusters from 300, 220, 360 and 120 degrees viewpoints.

In figure 6, we can see the core points in blue and the border points in red. DBSCAN has also determined noise points in the form of 3 outliers groups, which we can see in cyan, orange and green towards the head of the horn.

Note how the inner section of the horn is considered by DBSCAN to be the center area, with the back section containing most noise points.

I've searched stackoverflow a bit and xdecided to use a value of 10 for epsilon and a minimum number of four points for the neighborhood.

6.3 Mean Shift Clustering

MeanShift is a centroid-based algorithm. Given a set of data points, the Mean Shift algorithm iteratively assigns each data point towards the closest cluster centroid. The direction to the closest cluster centroid is determined by where most of the points nearby are at. So each iteration each data point will move closer to where the most points are at, which is or will lead to the cluster center.

In sklearn, MeanShift depends heavily on the *bandwidth* parameter, which is computed based on a *quantile* parameter.

6.3.1 Elbow Method

I've used the elbow method to compute the optimal number of clusters, searching for a quantile value in $[0, 1]$.

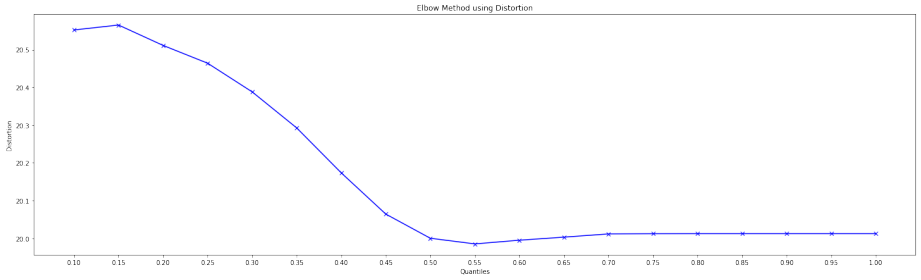

Figure 7: MeanShift distortions.

The Mean Shift algorithm, however, computes a single cluster, regardless of the quantile parameter's value.

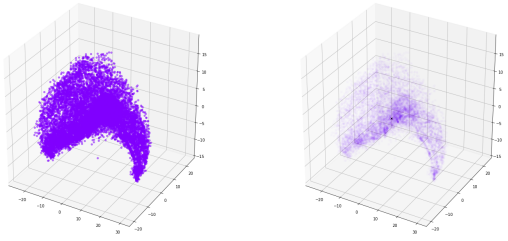

Figure 8: The MeanShift cluster.

6.3.2 Intercluster-Intracluster Metric Method

I've used the Intercluster-Intracluster distance to compute the optimum epsilon value, and it yielded $quantile = 10$. It isn't necessary to compute any metrics since the process has yielded but one cluster.

The resulted quantile led to a similar clsutering as the Silhouette search, only now the centroid has shifted.

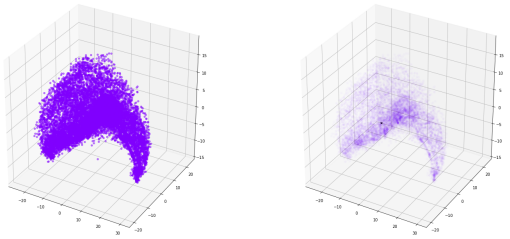

Figure 9: The MeanShift cluster.

6.4 Agglomerative Clustering

The agglomerative clustering algorithm start with each point in its own cluster and then merges pairs of clusters until only one cluster is formed.

For the agglomerative clustering method, I've used both the best k found during the K-Means process and the a k that optimizes the Silhouette metric and the Intracluster-Intercluster metrics, respectively.

6.4.1 K-Means *k*

The agglomerative clusters in figure 9 are clearly more compact than those constructed via K-Means.

Compared to the K-Means clusters, the agglomerative clusters are much more compact, with the tail clusters gaining an elongated shape. Also, the head clusters present a higher degree of sparsity. The metrics are:

- Silhouette: 0.09696238365307305
- Interclsuter: 2647809555.3413167
- Intracluster: 89643814.02650206

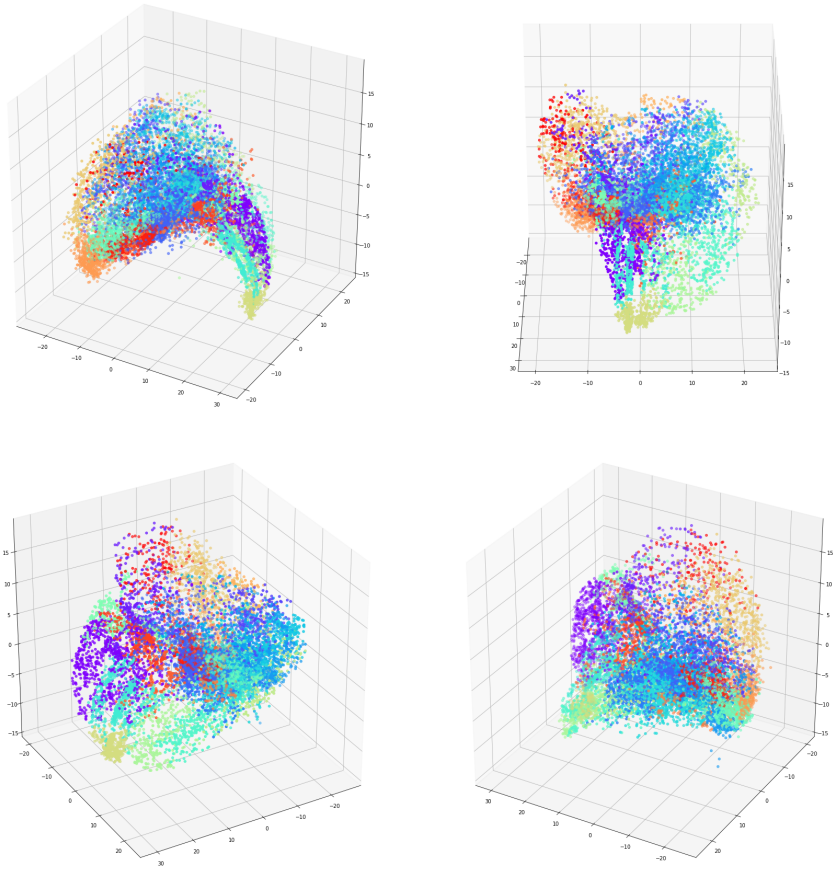

Figure 10: Agglomerative clusters from 300, 360, 60 and 120 degrees viewpoints.

6.4.2 Intercluster-Intracluster Distance

Optimizing the intercluser-intracluster distance has yielded an optmimum of $k = 99$. The metrics are:

- Silhouette: 0.12173056403924244
- Interclsuter: 2716626927.5763545
- Intracluster: 20826441.791468382

Just like in the case of K-Means, the clusters are more well-defined and well-separated, with compact and elongated clusters on the tail and more dispersed clusters towards the head, where the ouliers are more obvious. This time the intermediate clusters are much more elongated.

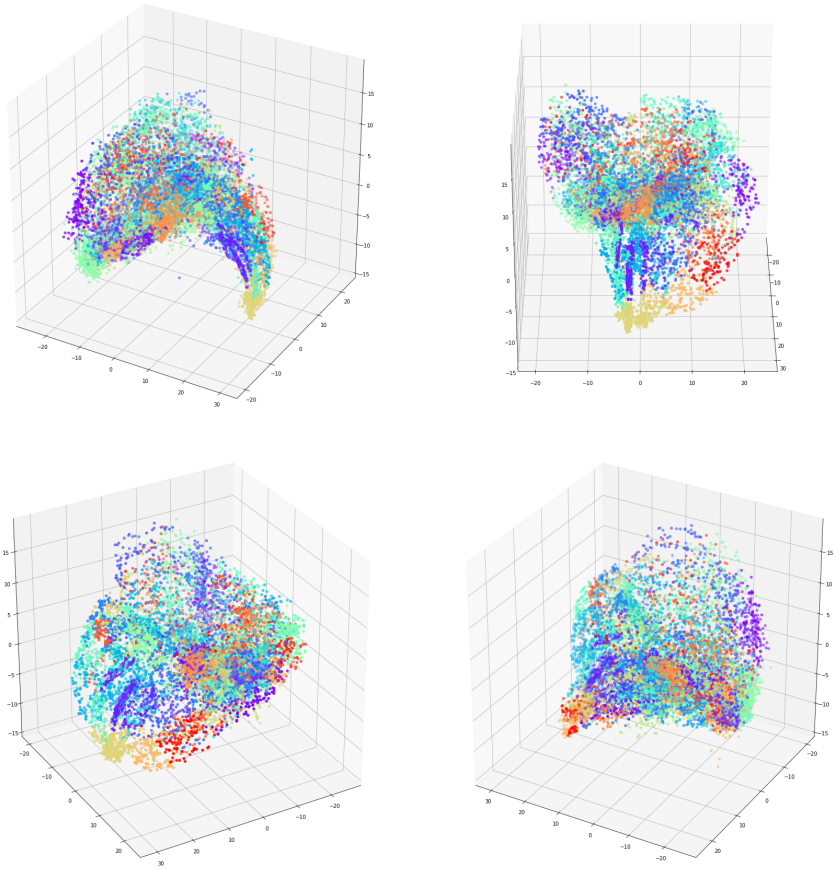

Figure 11: Agglomerative clusters from 300, 360, 60 and 120 degrees viewpoints.

6.4.3 Silhouette Method

It is no surprise that, in this scenario, the Silhouette is maximized by $k = 2$. Note how close this grouping is to the K-Means Silhouette clustering. The metrics are:

- Silhouette: 0.1164869467002419
- Interclsuter: 2718419162.859649
- Intracluster: 19034206.50818126

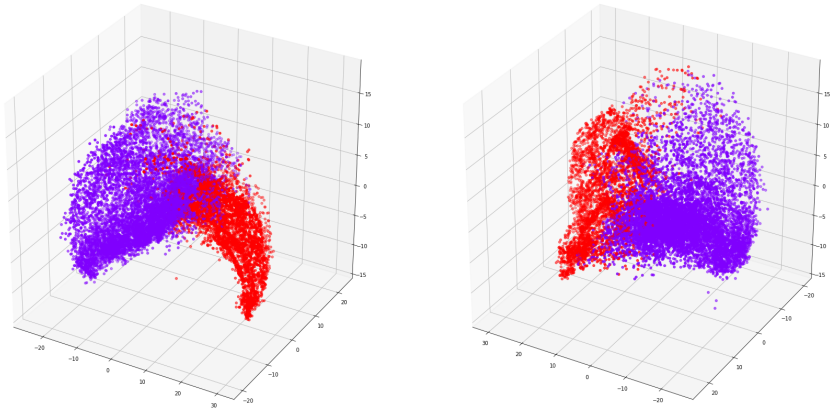

Figure 12: Agglomerative clusters from 300 and 120 degrees viewpoints.

6.5 Birch Clustering

BIRCH, or Balanced Iterative Reducing and Clustering using Hierarchies, deals with large datasets by first generating a more compact summary that retains as much distribution information as possible, and then clustering the data summary instead of the original dataset.

The Bitch algorithm depends primarily on the *threshold* and *number of clusters* parameters. For the latter I've used the K-Means elbow optimum, as well as a grid search over a set of clusters and thresholds via Silhouette and Intercluster-Intracluster distance.

Unfortunately, since a parameter grid search lasts too long, I've been forced to

6.5.1 K-Means *k*

The birch tail cluster resemble their agglomerative counterparts in that they're compact and elongated, although there are more of them in this case. The head clusters, however, are pretty compact. The metrics are:

- Silhouette: 0.09472825960721697
- Interclsuter: 2634713554.5883026
- Intracluster: 102739814.77952224

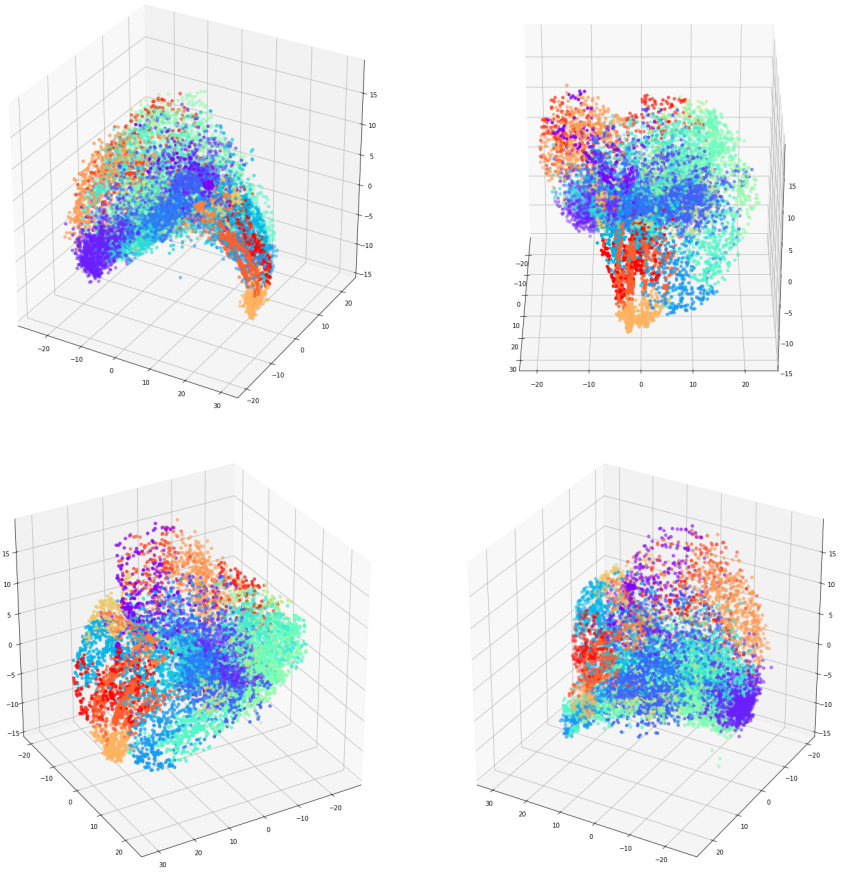

Figure 13: Birch clusters from 300, 360, 60 and 120 degrees viewpoints.

6.5.2 Intercluster-Intracluster Distance

The intracluster-intercluster distance optimization has yielded an optimum of $k = 15$ and an optimum threshold of 0.1. The metrics are:

- Silhouette: 0.10225087785037541
- Interclsuter: 2586900961.446753
- Intracluster: 150552407.92106754

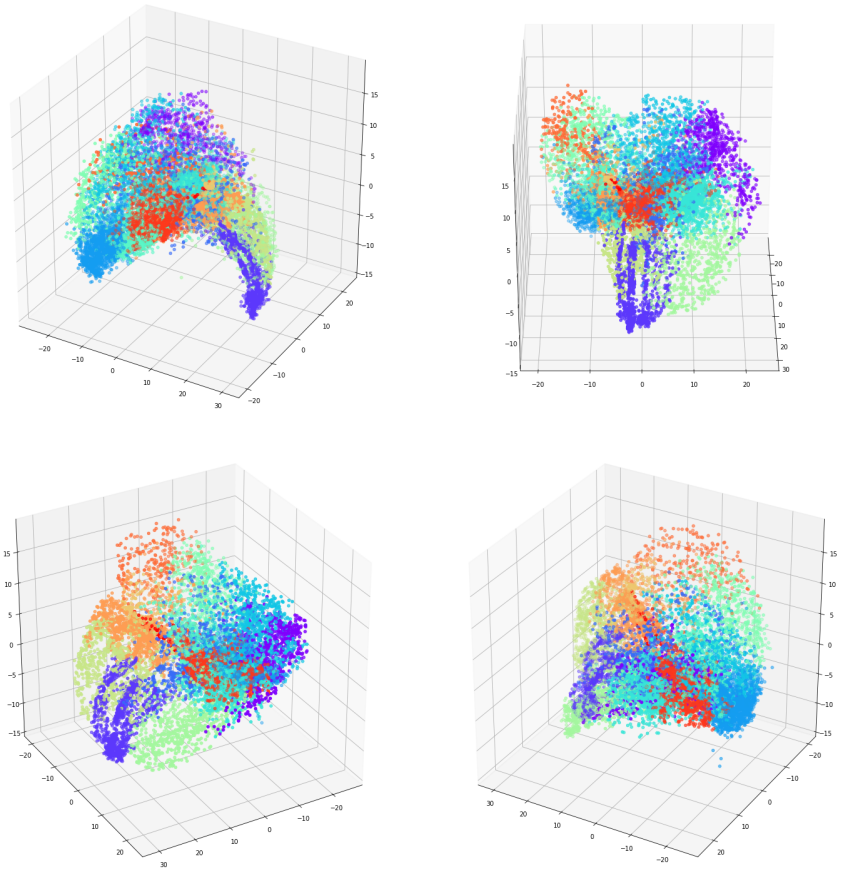

Figure 14: Birch clusters from 300, 360, 60 and 120 degrees viewpoints.

6.5.3 Silhouette

The Silhouette method has yielded an expected optimum of $k = 2$ clusters and an optimal threshold of 0.15. The metrics are:

- Silhouette: 0.27758729449712133
- Interclsuter: 1412347829.2406697
- Intracluster: 1325105540.1271546

When compared to other 2-cluster grouping, the Birch tail cluster seems to intrude more on the head's territory at the back of our data space.

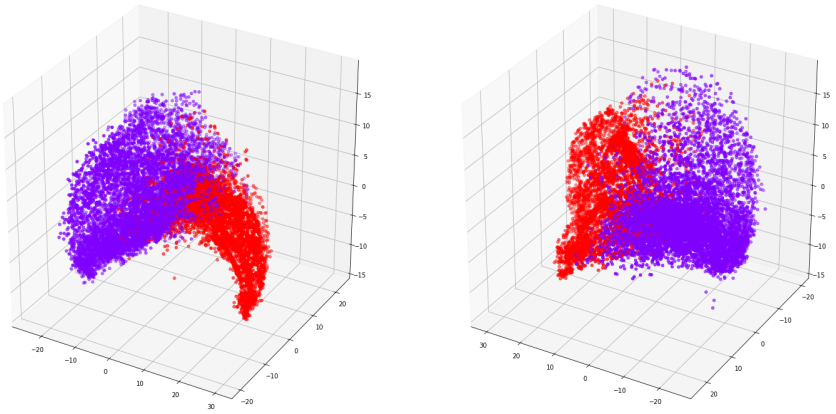

Figure 15: Birch clusters from 300 and 120 degrees viewpoints.

7 Conclusion

It was interesting to see how different approaches reveal several latent structures in our data, such as K-Means, DBSCAN have. It was interesting namely how DBSCAN seaprated the inner edge of the horn from the outer edge, which it considered as noise. As a side note, it also seemed as, with a small number of principal components, the DBSCAN couldn't retrieve much information about our data's structure, just like in the case of MeanShift.

Another interesting insight upon the data's structural nature was given by the centroid-based methods - K-Means, Agglomerative and Birch - and how they shape their clusters, the latter two defining more compact and elongated grouping, at least in the tail of the horn.

We've also seen that Silhouette maximization ended up yielding an optimum of 2 clusters. This however, does not help out intracluster or intercluster metrics at all, the Silhouette-chosen clusters yielding the worse metrics.

Contents

Figure 16: K-Means clusters and their respective centroids from 300, 360, 60 and 120 degrees viewpoints. Optimum K determined via distortion graph analysis.

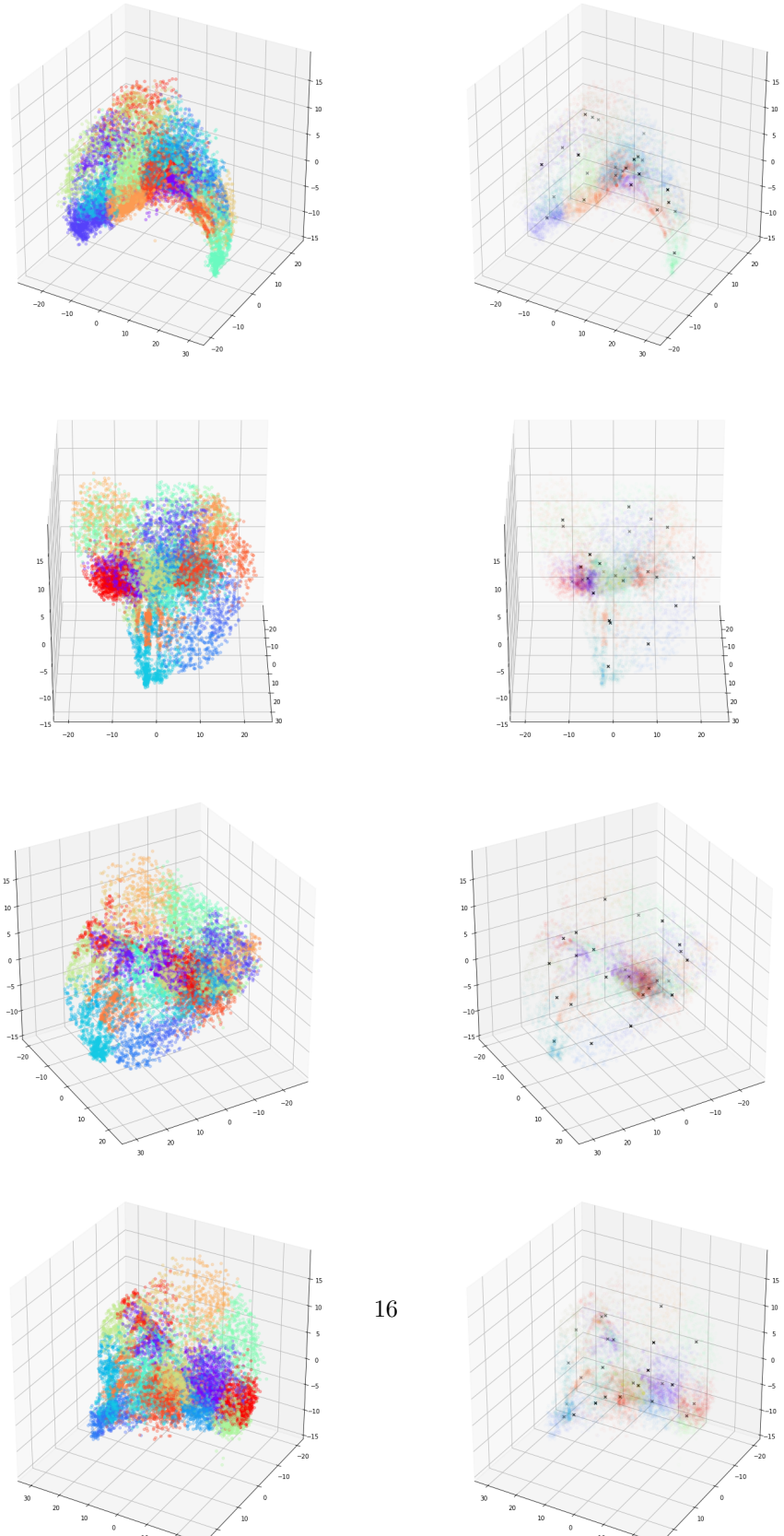

Figure 17: K-Means clusters and their respective centroids from 300, 360, 60 and 120 degrees viewpoints. Optimum K determined via within-cluster and between-cluster distance.