Unsupervised Machine-Learning Final Assignment

Guy Van-Dam

Abstract

Clustering 2 data-sets using 5 different **Clustering Algorithms**, showing results and analyzing performances for each, in an attempt to find the best algorithm for both **clustering** and **anomaly detection** for the given data-sets.

Finding the distance based *K-Means* and *Fuzzy-C-Means* to be the best performing in terms of both external and internal clustering measures. And *GMM* to be the best performing for anomaly detection tasks.

All source code can be found in the GitHub repository

Unsupervised-Learning

1 Introduction

When building a model, we need to experiment with different Machine-Learning methods, as different methods are built for different tasks. When we do not have a lot of information about our data, this step is crucial if not necessary for building an effective and efficient model. Trying different algorithms and analyzing and comparing their performances against each other will guide us towards building the right model for the task at hand.

When is comes to Unsupervised Machine-Learning, where we rarely have external labels, and we do not have a lot, if any information about our data. And so, we have to try different clustering algorithms and compare their results for classifying our given data-set in the most accurate way

2 Methods

2.1 Data-Sets And Dimension Reduction

2.1.1 Data-Set 1 - MoCap Hand Postures

Data-set consisting of $\approx 78,000$ points representing recordings of 36 markers - 12 for each (x, y, z) axis. Positioned on the left hand of a user performing 5 known hand gestures.

Due to computational constraints, this data-set was reduced to $\approx 14,000$ points and the last marker of each axis removed due to high non-values percentage. We remove all of the points with less than 95% non-values, and fill the missing values with the median of each component. After sampling, the number of classes in the external classifier was 3 instead of the original 5.

2.1.2 Data-Set 2 - HTRU2

Data-set consisting of ≈ 17000 *Pulsar* candidates collected during the *HTRU* survey. This data-set stayed whole, as it doesn't contain any non-values, and it's of manageable size. It contain 2 classes.

2.1.3 Principal Component Analysis

Although we eliminated some columns, our data-sets were all above 3 dimensions, so a dimension reduction algorithm needed to shorten computational time. The dimension reduction algorithm of choice was **Principal Component Analysis**, or **PCA** with a the fixed random seed mentioned below.

After normalizing the data using functions from the *Scikit* library, it is passed through the PCA algorithm, reducing the data to around 10% of the original size, as the average number of dimensions of one point is 15 on average.

We normalize and reduce the data dimensions using the *reduceDimensions()* method of the *DimReductionAlgorithm* object in *DimReductionAlgorithm.py*

After sampling, deleting, and normalizing. the data is *prepared* for clustering.

This whole process is done by calling the method *prepareDataset()* method found in the *Dataset* object in *Dataset.py*

2.2 Clustering Algorithms and Random States

5 clustering algorithms were used with the following parameters:

- Agglomerative Clustering
  - *Ward* linkage criterion which minimizes the **variance of the clusters being merged**
- Fuzzy-C-Means
- Gaussian Mixture Models - GMM
  - separate co-variance matrix Σ_k for each cluster C_k
- K-Means
- Spectral
  - affinity matrix constructed using the *radial basis kernel function*
  - partitioning with *K-Means*.

2.2.1 Random State

Many of the functions that were used were randomly initialized, so a fixed random state was required for reproducibility. I used **0** as a fixed random seed. While the list of random states I used when running the statistical tests can be found found in *Code/GlobalParameters.py*

2.3 Statistical Tests - Defining an Order Across Different Random States.

Due to the random initialization of algorithms' parameters, all but *Agglomerative* can be sensitive to random states, so running on different random states and performing a statistical test is vital when trying to get information from these algorithms.

2.3.1 Two-Sample One-Tailed T-Test

The *one-tailed two-sample t-test* is a statistical test for difference between the 2 samples population means. We can use this test for testing if the population mean of the first sample is greater than the population mean of the second sample. To get the *p-value* for the *Null Hypothesis* $H_0 : \mathbb{E}[b] \geq \mathbb{E}[a]$ we manipulate the output *p-value* from the two-sided test as

$$\textit{p-value} = \begin{cases} 1 - \frac{p_{output}}{2}, & \text{if } t - statistic_{output} < 0 \\ \frac{p_{output}}{2}, & \text{otherwise} \end{cases}$$

If we get a *p-value* lower than 0.05, we can reject the null hypothesis, and say that $\mathbb{E}[a] > \mathbb{E}[b]$

With this in mind we can easily define a measure of order in samples representing results across different random states, by using samples **population means** as our order.

We first use *ANOVA*, a parametric statistical test with a *Null Hypothesis* $H_0 : \mathbb{E}[sample_1] = \mathbb{E}[sample_2] = ... = \mathbb{E}[sample_n]$ for $n \geq 2$. As a preview test for differences in population means between all of our samples. If *ANOVA* fails, we can resort to our comparison function

$$Comp(sample_a, sample_b) = \begin{cases} 0, & t - test(sample_a, sample_b) \text{ returns } p > \alpha = 0.05 \\ 1, & t - test_{One-Tailed}(sample_a, sample_b) \text{ returns } p > \alpha = 0.05 \\ -1 & otherwise \end{cases}$$

With this comparison function we can easily sort any list of samples based on **population mean** with statistical evidence. And thous getting order within our samples.

2.4 Fitment To External Classification And Optimal Number Of Clusters

Using **Kullback–Leibler Divergence** - $D_{\mathrm{KL}}(P\|Q) = \sum P(x)\log\left(\frac{P(x)}{Q(x)}\right)$ we measure fit between the prediction labels given to us by the algorithm and the ground truth labels in the data-set. For each of our 30 different random states, we cluster the data and compute the $D_{\mathrm{KL}}(P(\text{External classifier})\|P(\text{prediction labels}))$ for that random state. With the a list of 30 D_{KL} values for each algorithm, we sort the algorithms with the order defined above.

By running *FitExternalClass.py* the *CSV* file is created and saved.

2.4.1 Silhouette Score

The *Silhouette Score* is an internal measure of performance of data clustering. We start by defining a_i to be the mean distance of point x_i from the other points in **its** cluster and b_i to be min$\{$mean distance of point x_i from the points in the **other** clusters$\}$.

We can now define the *Silhouette Score* for a point x_i as $s_i = \frac{b(i)-a(i)}{\max\{a(i),b(i)\}}$. And thous, a point with a silhouette score closer to 1 is a which fits well within its cluster while a point with a score closer to -1 is a point which doesn't fit well, due to it being classified or anomalous.

For number of clusters n ranging from 3-6 in data-set 1 and 2-5 in data-set 2 we cluster the data and compute the *Silhouette score*. We repeat this process for 30 different random states, and using our order get the optimal n. We repeat that for all of our algorithms.

This is can be done by running *OptimalNClusters.py*

2.5 Anomaly Detection - Comparing performances

2.5.1 Clustering Based Anomaly Detection

Anomaly detection can be performed using our clustering algorithms, by clustering the data, classifying all the points with a negative *Silhouette score* as anomalous points, and re-cluster the data until no negative scored points are found.

In addition to that, I have also used ***Local Outlier Factor*** with k neighbors = 20.

We start by clustering all of the data using *K-Means* and computing the silhouette score as a benchmark to the other algorithms. We than remove all of the anomalous points using our algorithms, and re-cluster the "clean" data with *K-Means*. We repeat this process for all of our random states, and sort the samples to get the best performing anomaly detection algorithm.

Unfortunately, due to computational constraints, I was not able to preform this test with the *Agglomerative* and *Spectral* algorithms.

3 Results

3.1 Optimal Number Of Clusters

Table 1: **Optimal Number Of Clusters** ranging from 3-6 for Data-Set 1 and 2-5 in Data-Set 2 With 30-Random-State Statistical-Test Sorting

Data-Set	**K-Means**	**GMM**	**Fuzzy-C-Means**	**Agglomerative**	**Spectral**
Data-Set 1	3	4	3	3	3
Data-Set 2	4	3	4	3	3

Table 2: **Optimal Number Of Clusters** For Data-Set 1 Sorted From High To Low by **Silhouette Score Samples** With 30-Random-State Statistical-Test Sorting

Place In Sorted Order	**K-Means**	**GMM**	**Fuzzy-C-Means**	**Agglomerative**	**Spectral**
2	3	4	5	4	5
3	5	2	3	5	4
4	2	5	2	2	2

While we don't get a definitive answer to what is the number of classes for each data-set, we can assume that data-set 1 is fitted pretty well with 3 classes, which it's external class in the first place. while data-set 2, which presumably has 2 classes, is shown to have more than that, with 2 classes coming last for all but one of the algorithms.

3.2 Comparing Performances - Silhouette Score

Table 3: **Clustering** Comparison Based On a 30-Random-State Statistical-Test Sorting Showing Sample Mean *Silhouette Score*

Data-Set	**First**	**Second**	**Third**	**Forth**	**Fifth**
1	0.395 KMeans	0.390 FuzzyCMeans	0.387 Spectral	0.354 Agglo	0.328 GMM
2	0.404 Spectral	0.402 KMeans	0.400 FuzzyCMeans	0.376 GMM	0.364 Agglo

For both data-sets the differences between the algorithms are not all that different. especially when comparing *Spectral* to *K-Means* to *Fuzzy-C-Means*. While *GMM* and *Agglomerative* are behind significantly. With that in mind, we can put *Fuzzy-C-Means* and *K-Means* in first and second place as they are very similar in results which indicates a similarity in the prediction clusters they produce.

Statistical tests values can be found in tables 6 and 7

3.3 Fitment To External Classification

Table 4: **External Classification Fitment** Comparison Based On a 30-Random-State Statistical-Test Sorting Showing Sample Mean *K-L Divergence*

Data-Set	**First**	**Second**	**Third**	**Forth**	**Fifth**
1	0.083 KMeans	0.087 FuzzyCMeans	0.100 Spectral	0.176 GMM	0.207 Agglo
2	0.058 FuzzyCMeans	0.059 KMeans	0.061 Spectral	0.068 GMM	0.088 Agglo

While in data-set 1 there was a lot of difference in performance, in data-set 2 we see a much more subtle results, almost to a point of equal performance, which is even more significant looking at the results from table 3 showing silhouette score results. We can probably assume circular clusters on both data-sets, due to the close fitment of the circular assuming algorithms to the ground-truth labels.

Statistical tests values can be found in tables 8 and 9

3.4 Comparing Performances - Anomaly Detection

Table 5: **Anomaly Detection** Performance Comparison Based On a 30-Random-State Statistical-Test Sorting Showing Sample Mean *Silhouette Score*

Data-Set	**First**	**Second**	**Third**	**Forth**	**Fifth**
1	0.434 GMM	0.406 FuzzyCMeans	0.406 KMeans	0.395 KNN	0.395 AllData
2	0.469 GMM	0.416 FuzzyCMeans	0.411 KMeans	0.402 AllData	0.402 KNN

From the results, we see *GMM* has a big advantage over the other algorithms which produced pretty similar results. It might be thanks to the was *GMM* treats points which doesn't not assign to a cluster indefinitely. It will ”accept” anomalous points into the cluster, and when computing silhouette scores, this points will be removed due to their negative score.

Statistical tests values can be found in tables 10 and 11

4 Discussion

While the external classifier in data-set 1 is pretty in-line to the results giving to us by the algorithms, with 3 classes for the sampled data-set. Data-set 2 proves to have more classes from what the external classifier tells us, With 2 classes coming last in the ranking for almost every algorithms. Further more, this test shows us that for the general clustering and classifying use, the distance based algorithms which are *K-Means* and *Fuzzy-C-Means* performed better than the other algorithms, are far faster and computationally cheaper, which makes them a great choice compared to the alternatives. Finally, we find *GMM* to be the best anomaly detector from the bunch, with it's unique, normal distribution approach which proves to be better than the distance and density approaches with the anomaly detection technique used.

Table 6: P-Value and T-Statistics Values For 30-Random-State Statistical-Test Sorting Of **Silhouette Score** Of Different **Clustering** Algorithms For Data-Set 1

	p value	stat	test	result
1	0.00E+00	8609968.36	KMeans = GMM = Fuzzy = Agglo = Spec	FALSE
2	5.79E-160	-3361.61	GMM = KMeans	FALSE
3	1.00E+00	-3361.61	GMM < KMeans	TRUE
4	3.53E-158	3138.98	FuzzyCMeans = GMM	FALSE
5	1.77E-158	3138.98	FuzzyCMeans < GMM	FALSE
6	3.53E-158	3138.98	FuzzyCMeans = GMM	FALSE
7	1.77E-158	3138.98	FuzzyCMeans < GMM	FALSE
8	7.50E-141	-1614.36	FuzzyCMeans = KMeans	FALSE
9	1.00E+00	-1614.36	FuzzyCMeans < KMeans	TRUE
10	2.75E-211	-24095.89	Agglomerative = FuzzyCMeans	FALSE
11	1.00E+00	-24095.89	Agglomerative < FuzzyCMeans	TRUE
12	3.95E-135	1296.10	Agglomerative = GMM	FALSE
13	1.98E-135	1296.10	Agglomerative < GMM	FALSE
14	1.41E-161	3576.43	Spectral = Agglomerative	FALSE
15	7.04E-162	3576.43	Spectral < Agglomerative	FALSE
16	5.12E-98	-311.87	Spectral = FuzzyCMeans	FALSE
17	1.00E+00	-311.87	Spectral < FuzzyCMeans	TRUE

Table 7: P-Value and T-Statistics Values For 30-Random-State Statistical-Test Sorting Of **Silhouette Score** Of Different **Clustering** Algorithms For Data-Set 2

	p value	stat	test	result
1	1.60E-78	394.90	KMeans = GMM = Fuzzy = Agglo = Spec	FALSE
2	4.08E-13	-9.23	GMM = KMeans	FALSE
3	1.00E+00	-9.23	GMM < KMeans	TRUE
4	6.16E-12	8.53	FuzzyCMeans = GMM	FALSE
5	3.08E-12	8.53	FuzzyCMeans < GMM	FALSE
6	6.16E-12	8.53	FuzzyCMeans = GMM	FALSE
7	3.08E-12	8.53	FuzzyCMeans < GMM	FALSE
8	8.25E-47	-43.04	FuzzyCMeans = KMeans	FALSE
9	1.00E+00	-43.04	FuzzyCMeans < KMeans	TRUE
10	1.74E-196	-13654.22	Agglomerative = FuzzyCMeans	FALSE
11	1.00E+00	-13654.22	Agglomerative < FuzzyCMeans	TRUE
12	1.35E-15	-10.74	Agglomerative = GMM	FALSE
13	1.00E+00	-10.74	Agglomerative < GMM	TRUE
14	1.74E-14	10.05	Spectral = GMM	FALSE
15	8.71E-15	10.05	Spectral < GMM	FALSE
16	4.90E-130	1065.97	Spectral = FuzzyCMeans	FALSE
17	2.45E-130	1065.97	Spectral < FuzzyCMeans	FALSE
18	5.00E-51	50.82	Spectral = KMeans	FALSE
19	2.50E-51	50.82	Spectral < KMeans	FALSE

Table 8: P-Value and T-Statistics Values For 30-Random-State Statistical-Test Sorting Of K-L Divergence Of **External Classifier** And Prediction Labels For Data-Set 1

	p value	stat	test	result
1	2.00E-307	895160.16	KMeans = GMM = Fuzzy = Agglo = Spec	FALSE
2	6.48E-117	854.03	GMM = KMeans	FALSE
3	3.24E-117	854.03	GMM < KMeans	FALSE
4	6.29E-117	-854.50	FuzzyCMeans = GMM	FALSE
5	1.00E+00	-854.50	FuzzyCMeans < GMM	TRUE
6	2.75E-69	120.27	FuzzyCMeans = KMeans	FALSE
7	1.38E-69	120.27	FuzzyCMeans < KMeans	FALSE
8	6.29E-117	-854.50	FuzzyCMeans = GMM	FALSE
9	1.00E+00	-854.50	FuzzyCMeans < GMM	TRUE
10	2.51E-226	76784.98	Agglomerative = FuzzyCMeans	FALSE
11	1.26E-226	76784.98	Agglomerative < FuzzyCMeans	FALSE
12	5.57E-92	306.28	Agglomerative = GMM	FALSE
13	2.78E-92	306.28	Agglomerative < GMM	FALSE
14	1.59E-89	276.84	Spectral = FuzzyCMeans	FALSE
15	7.95E-90	276.84	Spectral < FuzzyCMeans	FALSE
16	5.09E-105	-523.65	Spectral = GMM	FALSE
17	1.00E+00	-523.65	Spectral < GMM	TRUE

Table 9: P-Value and T-Statistics Values For 30-Random-State Statistical-Test Sorting Of K-L Divergence Of **External Classifier** And Prediction Labels For Data-Set 2

	p value	stat	test	result
1	1.11E-85	572.81	KMeans = GMM = Fuzzy = Agglo = Spec	FALSE
2	7.34E-11	8.01	GMM = KMeans	FALSE
3	3.67E-11	8.01	GMM < KMeans	FALSE
4	1.95E-11	-8.36	FuzzyCMeans = GMM	FALSE
5	1.00E+00	-8.36	FuzzyCMeans < GMM	TRUE
6	5.78E-29	-21.73	FuzzyCMeans = KMeans	FALSE
7	1.00E+00	-21.73	FuzzyCMeans < KMeans	TRUE
8	9.72E-132	1571.08	Agglomerative = KMeans	FALSE
9	4.86E-132	1571.08	Agglomerative < KMeans	FALSE
10	5.00E-24	17.21	Agglomerative = GMM	FALSE
11	2.50E-24	17.21	Agglomerative < GMM	FALSE
12	4.71E-72	134.82	Spectral = KMeans	FALSE
13	2.35E-72	134.82	Spectral < KMeans	FALSE
14	4.34E-07	-5.72	Spectral = GMM	FALSE
15	1.00E+00	-5.72	Spectral < GMM	TRUE

Table 10: P-Value and T-Statistics Values For 30-Random-State Statistical-Test Sorting Of **Silhouette Score** Of Different **Anomaly Detection** Algorithms For Data-Set 1

	p value	stat	test	result
1	4.65E-298	6.58E+05	AllData = KMeans = GMM = FuzzyCMeans = KNN	FALSE
2	4.91E-80	1.87E+02	KMeans = AllData	FALSE
3	2.46E-80	1.87E+02	KMeans < AllData	FALSE
4	0.00E+00	1.54E+15	GMM = KMeans	FALSE
5	0.00E+00	1.54E+15	GMM < KMeans	FALSE
6	0.00E+00	-1.41E+15	FuzzyCMeans = GMM	FALSE
7	1.00E+00	-1.41E+15	FuzzyCMeans < GMM	TRUE
8	0.00E+00	1.14E+14	FuzzyCMeans = KMeans	FALSE
9	0.00E+00	1.14E+14	FuzzyCMeans < KMeans	FALSE
10	0.00E+00	-1.41E+15	FuzzyCMeans = GMM	FALSE
11	1.00E+00	-1.41E+15	FuzzyCMeans < GMM	TRUE
12	1.43E-74	-1.50E+02	KNN = KMeans	FALSE
13	1.00E+00	-1.50E+02	KNN < KMeans	TRUE
14	5.80E-01	5.57E-01	KNN = AllData	TRUE

Table 11: P-Value and T-Statistics Values For 30-Random-State Statistical-Test Sorting Of **Silhouette Score** Of Different **Anomaly Detection** Algorithms For Data-Set 2

	p value	stat	test	result
1	0.00E+00	60623857.57	AllData = KMeans = GMM = FuzzyCMeans = KNN	FALSE
2	1.11E-154	4035.30	KMeans = AllData	FALSE
3	5.56E-155	4035.30	KMeans < AllData	FALSE
4	6.52E-264	360113.81	GMM = KMeans	FALSE
5	3.26E-264	360113.81	GMM < KMeans	FALSE
6	1.54E-171	-8071.08	FuzzyCMeans = GMM	FALSE
7	1.00E+00	-8071.08	FuzzyCMeans < GMM	TRUE
8	3.30E-75	153.54	FuzzyCMeans = KMeans	FALSE
9	1.65E-75	153.54	FuzzyCMeans < KMeans	FALSE
10	1.54E-171	-8071.08	FuzzyCMeans = GMM	FALSE
11	1.00E+00	-8071.08	FuzzyCMeans < GMM	TRUE
12	9.25E-170	-7501.75	KNN = KMeans	FALSE
13	1.00E+00	-7501.75	KNN < KMeans	TRUE
14	9.57E-43	39.78	KNN = AllData	FALSE
15	4.79E-43	39.78	KNN < AllData	FALSE