Performance Evaluation of Consensus Clustering Algorithms

STAT 598 Progress Report

Derek Chiu

August 24, 2015

Contents

1 Preface

This progress report fulfills the UBC Science Co-op requirement to submit a work term report at the end of every four month period. BC Cancer Agency (BCCA) is a not-for-profit organization that aims to provide care for cancer patients and conduct innovative cancer research. Our department, OvCaRe, is the Ovarian Cancer Research team tasked with studying ovarian cancers of many types. The objective of the project I am working on is to discover a viable classifer for ovarian high-grade serous carcinoma (HGSC). My role is to help devise a clustering algorithm that can statistically partition subtypes of HGSC without knowledge of the pathological properties of each sample. The progress report will evaluate the method we are using, consensus clustering, on a publicly available data set as well as simulated data sets. The final technical report will contain results of our method applied on HGSC data from our own cohort.

2 Introduction

Unsupervised learning is the process of inferring something about a data structure without knowing its true class labels. Cluster analysis is an unsupervised learning method of assigning entities into different groups based on one or more of their attributes. It is unsupervised because we do not know the true partitions of the entities. The objective is to place similar objects together in the same cluster and separate dissimilar objects into different clusters. For example, in genomics studies, we frequently try and cluster patient samples measured on a large number of molecular features. When we get a clustering assignment from an algorithm, we often want to evaluate its performance. Ideally, a good clustering algorithm is able to differentiate entities without knowledge of the true class labels. In addition, we want the algorithm to arrive at a stable and optimal number of clusters. The choice of the number of clusters is not trivial in some cases.

3 Methods

3.1 Clustering Algorithms

There are many clustering algorithms, each approaching the clustering problem in a different way. It is most important to note the advantages and limitations of each algorithm. There are some definitions to take note of:

- **Compactness**: how close the objects are in each cluster
- **Connectivity**: how connected the objects are in the feature space

3.1.1 Hierarchical Clustering

This clustering algorithm is very popular because of its intuitive representation using dendrograms (trees). Based on a distance matrix, the objects/features are clustered based on a linkage type. More similar objects are joined near the bottom of a dendrogram whereas less similar objects are joined at a higher tree height. A linkage criterion determines the distances amongst a set of objects/features using the pairwise distances. For example, an average linkage would use the average pairwise distances. In this way, a dendrogram with all objects/features can be made by recursively linking larger and larger sets of observations together.

CITATION: it turns out that single linkage works very well for data sets exhibiting connectivitiy but not compactness. An example of this would be a tree rings. Clusters are circles, and objects that are far away are in the same cluster compared to objects that are close. On the other hand, average linkage works well for data sets exhibiting compactness.

3.1.2 k-Means Clustering

First, k means are randomly initialized in the multi-dimensional object/feature space that we wish to cluster. Assigning each object/feature to its closest mean forms the clusters. The k means are re-calculated based on the centroids (center points) within each cluster. This process is repeated until the centroids converge.

There are two caveats to note when using k-means. First, the cluster assignments are unstable because they depend on the random initialization of the means. We would preferably want to repeat this process many times to see whether the clusters change drastically. Secondly, the choice of k is not arbitrary. Cross-validation using an appropriate loss function is a popular method for choosing k.

3.1.3 k-Medoids Clustering

Very similar to k-means except that we initialize a set of random data points as medoids instead of random means that are generally not real data points. The most common version of k-medoids is PAM.

3.1.4 Nonnegative Matrix Factorization

Given a non-negative data matrix A, we can factor it into two matrices W and H, which are also non negative. W and H have important properties. Suppose A has genes as rows and samples as columns. If we are interested in clustering samples, then H has a reduced gene space of meta genes that fully explain the samples. Samples are clustered based on the metagene they are most associated with. If we are interested in clustering genes, then W has a reduced sample space of metasamples that fully explain the genes. Genes are clustered based on the metasample they are most associated with.

In gene expression data, it is common to standardize the genes. Doing so would disrupt the nonnegativity of A required for NMF. A simple remedy can solve this problem. We append the matrix −A to the bottom of A, preserving the same number of columns, and set all the negative entries to 0. The computational complexity has been doubled as a result. NMF takes a long time to run, but offers the most stable clustering assignments.

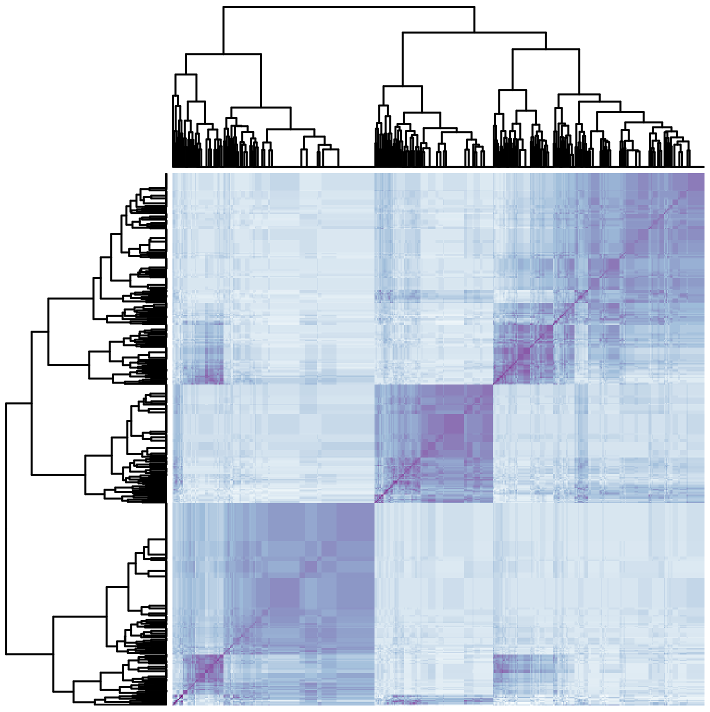
3.2 Consensus Clustering

Consensus clustering is as much an algorithm itself as it is a way of combining realizations of other clustering algorithms. For this, it deserves its own section. Algorithms like k-means and PAM are unstable, as the clustering assignments depend on the initialization of centroids and medoids, respectively. Consensus clustering compares results from repeated runs of a clustering algorithm. Typically, these runs use different subsamples of the data to model sampling variability. A final clustering assignment is determined based on agreement across replicates.

A consensus matrix is a significant aspect of consensus clustering. All entries range from 0 to 1, and represent the proportion of replicates in which two objects/features were clustered together. A perfect clustering would consist of a consensus matrix with only 0s and 1s. More importantly, we can perform hierarchical clustering on a consensus matrix, resulting in a heat map with a diagonal block structure.

Repeating this process for different values of k (number of clusters) and looking at the corresponding heatmaps, we can see which value of k provides the most stable clustering assignment. Certainly, using a performance measure such as PAC would be a more formal method of assessment that doesn't rely on potentially subjective visualizations.

Consensus clustering attempts to stabilize results from randomness introduced by subsampling and the clustering algorithm. A natural extension is sometimes called meta consensus clustering, where we want to compare results across different algorithms. For instance, we may combine the consensus clustering assignments from 10 different algorithms and come up with a final clustering.



The proportion of cases with at least 0.6 agreement is 0.2170958.

The confusion matrix is shown below, as well as different metrics for each class.

	C1	C2	C4	C5
C1	104	10	18	7
C2	1	37	10	1
C4	1	60	91	21
C5	3	0	16	109

	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
Class: C1	0.9541	0.9079	0.7482	0.9857	0.2229	0.2127	0.2843	0.931
Class: C2	0.3458	0.9686	0.7551	0.8409	0.2188	0.07566	0.1002	0.6572

	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	Prevalence	Detection Rate	Detection Prevalence	Balanced Accuracy
Class: C4	0.6741	0.7684	0.526	0.8608	0.2761	0.1861	0.3538	0.7212
Class: C5	0.7899	0.9459	0.8516	0.9197	0.2822	0.2229	0.2618	0.8679

4 External Evaluation

External evaluation usually refers to the case when we compare our clustering assignments to true class labels, or have some gold standard to compare to. In applications, this might be the published clustering result. The downside of using external evaluation is that the reference classes may not be correctly clustered themselves, and we are treating these as the norm. None the less, we can explore a few metrics.

4.1 Kappa Statistic

The unadjusted kappa statistic is 0.5927477 and the weighted kappa statistic is 0.7717546 for the final meta consensus cluster.

4.2 Adjusted Rand Index

The larger the better.

Algorithms	ARI
NMF (Divergence)	0.4799
NMF (Euclidean)	0.4435
PAM (Spearman)	0.427
HC (Diana)	0.4221
KM (Spearman)	0.4049
PAM (Euclidean)	0.3434
HC (Euclidean)	0.3275
KM (Euclidean)	0.2559
PAM (MI)	0.07465
KM (MI)	0.07369

4.3 Mutual Information

The larger the better.

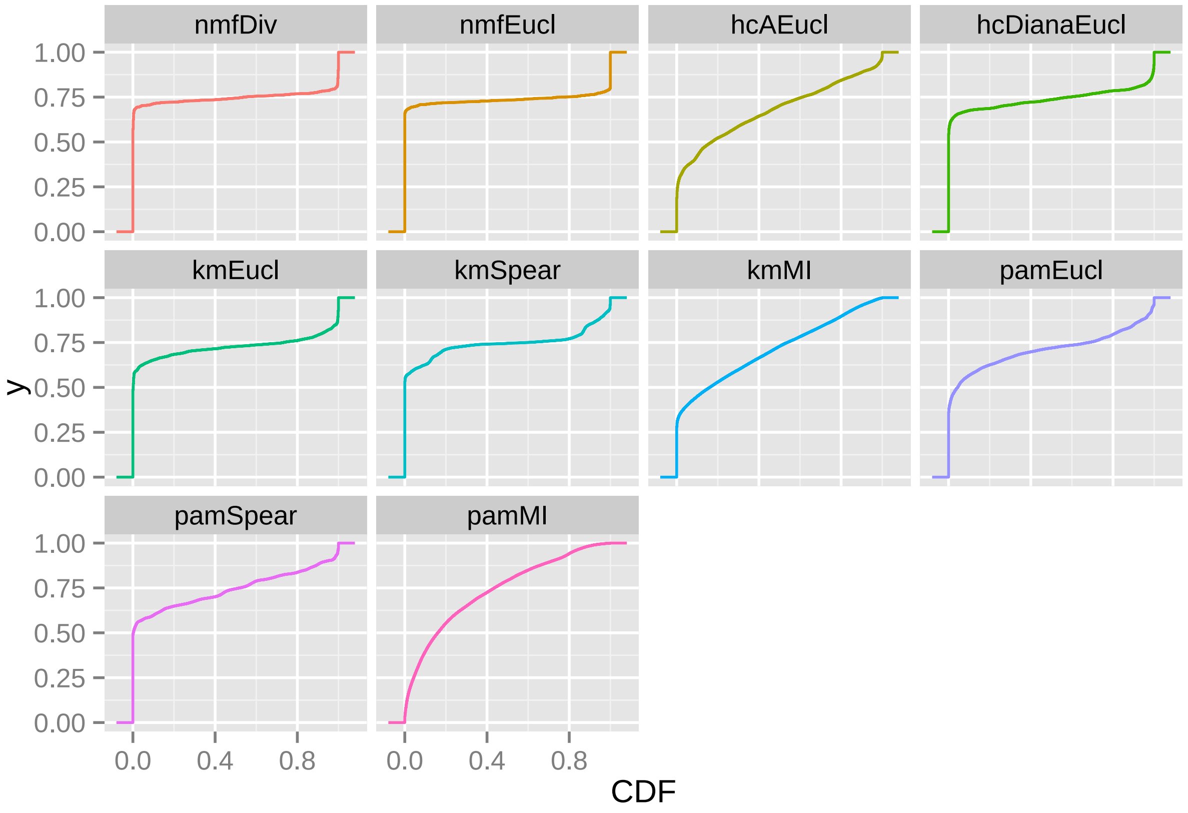
Algorithms	MI
NMF (Divergence)	0.6723
NMF (Euclidean)	0.6416
PAM (Spearman)	0.621
HC (Diana)	0.5976
KM (Spearman)	0.5889
PAM (Euclidean)	0.546
HC (Euclidean)	0.481

Algorithms	MI
KM (Euclidean)	0.4598
PAM (MI)	0.1652
KM (MI)	0.1169

The mutual information for the meta consensus clustering is 0.5987537.

5 Internal Evaluation

5.1 Cumulative Distribution Function



5.2 Proportion of Ambiguous Clusters

The lower the better.

Algorithms	PAC
NMF (Euclidean)	0.1406
NMF (Divergence)	0.3267
HC (Diana)	0.3829
KM (Spearman)	0.425
KM (Euclidean)	0.4398
PAM (Spearman)	0.4748
PAM (Euclidean)	0.5984

Algorithms	PAC
KM (MI)	0.7167
HC (Euclidean)	0.7836
PAM (MI)	0.9638

The PAC for the meta consensus matrix is 0.7114804.

5.3 Davies-Bouldin Index

For DBI, the lower the better.

Algorithms	DBI
HC (Euclidean)	1.689
NMF (Euclidean)	1.702
NMF (Divergence)	1.702
PAM (Spearman)	1.72
PAM (Euclidean)	1.725
KM (Spearman)	1.725
HC (Diana)	1.731
KM (Euclidean)	1.741
PAM (MI)	1.935
KM (MI)	1.972

5.4 Dunn Index

For DI, the larger the better.

Algorithms	DI
HC (Euclidean)	1.04
PAM (Euclidean)	1.028
NMF (Euclidean)	1.003
NMF (Divergence)	0.999
KM (Spearman)	0.9926
HC (Diana)	0.9866
PAM (MI)	0.9821
PAM (Spearman)	0.9492
KM (MI)	0.9262
KM (Euclidean)	0.9238

5.5 Silhouette Average Width

For SAW, the larger the better.

Algorithms	SAW
HC (Euclidean)	0.1226
PAM (Euclidean)	0.1171
NMF (Divergence)	0.1111

Algorithms	SAW
NMF (Euclidean)	0.1084
PAM (Spearman)	0.1069
KM (Spearman)	0.1059
HC (Diana)	0.09062
KM (Euclidean)	0.08707
PAM (MI)	-0.0061
KM (MI)	-0.007504

5.6 Rousseeuw's Silhouette

For Rousseuw's Silhouette internal cluster quality index (RS), the larger the better.

Algorithms	RS
HC (Euclidean)	0.1226
PAM (Euclidean)	0.1171
NMF (Divergence)	0.1111
NMF (Euclidean)	0.1084
PAM (Spearman)	0.1069
KM (Spearman)	0.1059
HC (Diana)	0.09062
KM (Euclidean)	0.08707
PAM (MI)	-0.0061
KM (MI)	-0.007504

5.7 C-Index

For CI, the lower the better.

Algorithms	CI
KM (Euclidean)	0.2816
NMF (Divergence)	0.3023
PAM (Euclidean)	0.31
HC (Euclidean)	0.3129
HC (Diana)	0.3212
NMF (Euclidean)	0.3369
PAM (MI)	0.3376
PAM (Spearman)	0.3489
KM (MI)	0.3529
KM (Spearman)	0.356

5.8 Baker and Hubert Index

For BHI, the larger the better.

Algorithms	BHI
HC (Euclidean)	2.051
PAM (Euclidean)	1.754

Algorithms	BHI
HC (Diana)	1.746
PAM (Spearman)	1.715
KM (Euclidean)	1.676
NMF (Euclidean)	1.666
KM (Spearman)	1.627
NMF (Divergence)	1.617
KM (MI)	-3.142
PAM (MI)	-3.358

5.9 Calinski-Harabasz Index

For CHI, the larger the better.

Algorithms	CHI
NMF (Divergence)	80.36
NMF (Euclidean)	79.26
KM (Spearman)	78.71
PAM (Spearman)	77.19
PAM (Euclidean)	75.71
KM (Euclidean)	75.38
HC (Diana)	74.95
HC (Euclidean)	66.49
PAM (MI)	13.23
KM (MI)	7.322

5.10 Summary

Here is a summary of all the indices for each algorithm, in unsorted order.

Algorithms	ARI	MI	PAC	DBI	DI	SAW	RS	CI	BHI	CHI
HC (Diana)	0.4221	0.5976	0.3829	1.731	0.9866	0.09062	0.09062	0.3212	1.746	74.95
HC (Euclidean)	0.3275	0.481	0.7836	1.689	1.04	0.1226	0.1226	0.3129	2.051	66.49
KM (Euclidean)	0.2559	0.4598	0.4398	1.741	0.9238	0.08707	0.08707	0.2816	1.676	75.38
KM (MI)	0.07369	0.1169	0.7167	1.972	0.9262	-0.007504	-0.007504	0.3529	-3.142	7.322
KM (Spearman)	0.4049	0.5889	0.425	1.725	0.9926	0.1059	0.1059	0.356	1.627	78.71
NMF (Divergence)	0.4799	0.6723	0.3267	1.702	0.999	0.1111	0.1111	0.3023	1.617	80.36
NMF (Euclidean)	0.4435	0.6416	0.1406	1.702	1.003	0.1084	0.1084	0.3369	1.666	79.26
PAM (Euclidean)	0.3434	0.546	0.5984	1.725	1.028	0.1171	0.1171	0.31	1.754	75.71

Algorithms	ARI	MI	PAC	DBI	DI	SAW	RS	CI	BHI	CHI
PAM (MI)	0.07465	0.1652	0.9638	1.935	0.9821	-0.0061	-0.0061	0.3376	-3.358	13.23
PAM (Spearman)	0.427	0.621	0.4748	1.72	0.9492	0.1069	0.1069	0.3489	1.715	77.19

6 Ranked Indices

The table below shows the ranking of algorithms for performance on a clustering index, for each index. There is an additional column that shows the propoportion of indices where an algorithm was ranked **first or second**.

Algorithms	ARI	MI	PAC	DBI	DI	SAW	RS	CI	BHI	CHI	Top
HC (Euclidean)	7	7	9	1	1	1	1	4	1	8	50%
NMF (Divergence)	1	1	2	3	4	3	3	2	8	1	50%
NMF (Euclidean)	2	2	1	2	3	4	4	6	6	2	50%
PAM (Euclidean)	6	6	7	5	2	2	2	3	2	5	40%
KM (Euclidean)	8	8	5	8	10	8	8	1	5	6	10%
HC (Diana)	4	4	3	7	6	7	7	5	3	7	0%
KM (MI)	10	10	8	10	9	10	10	9	9	10	0%
KM (Spearman)	5	5	4	6	5	6	6	10	7	3	0%
PAM (MI)	9	9	10	9	7	9	9	7	10	9	0%
PAM (Spearman)	3	3	6	4	8	5	5	8	4	4	0%

If we were to conduct a meta consensus clustering, a weight for each algorithm needs to be assigned. One such way of doing so is using the inverse rank sums. We can sum the ranks for each algorithm, and assign higher weight for consistently high ranked methods (1st or 2nd) and vice versa. This is shown below:

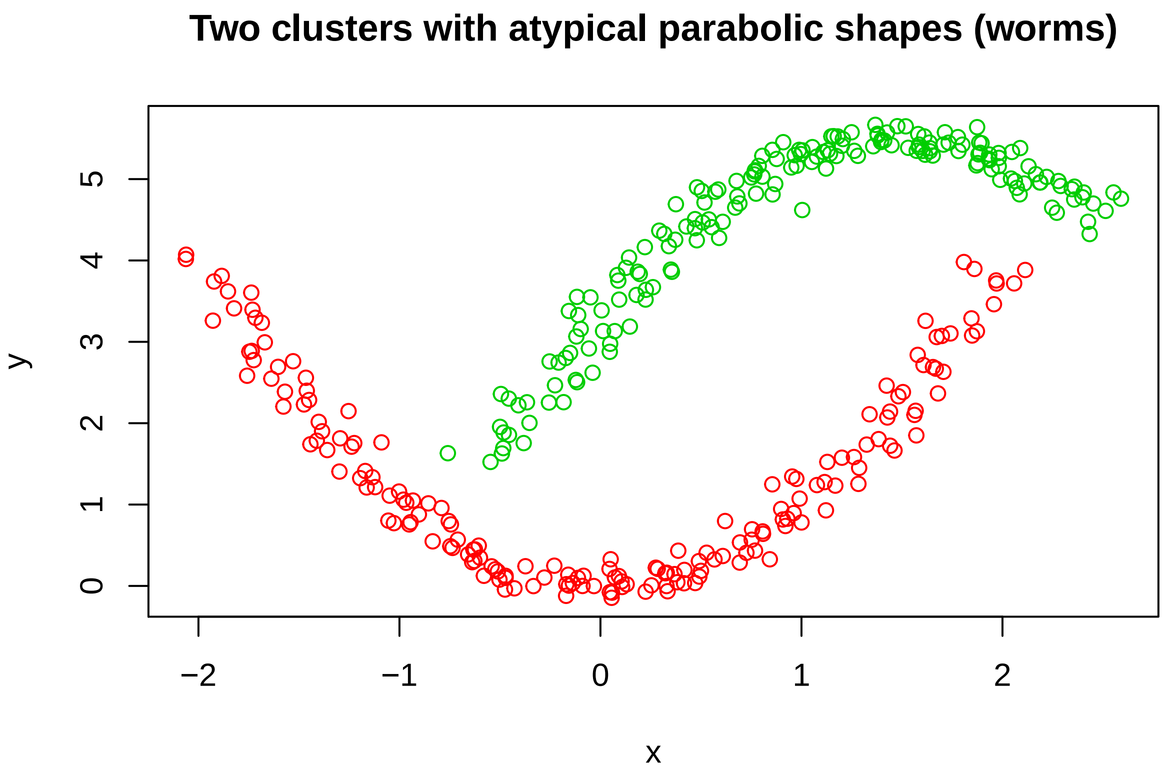
Algorithms	Top	Sum	Weight
NMF (Divergence)	50%	28	16.99%
NMF (Euclidean)	50%	32	14.87%
HC (Euclidean)	50%	40	11.89%
PAM (Euclidean)	40%	40	11.89%
PAM (Spearman)	0%	50	9.52%
HC (Diana)	0%	53	8.98%
KM (Spearman)	0%	57	8.35%
KM (Euclidean)	10%	67	7.1%
PAM (MI)	0%	88	5.41%
KM (MI)	0%	95	5.01%

7 Simulations

To confirm the clustering results from using the TCGA dataset, we need to try out the algorithms on a few simulated datasets, designed to test the robustness of each method. The `clusterSim` package provides very good built-in examples to use.

7.1 Worms Dataset

The following two-cluster dataset is our first simulation:



The ranked indices and weights tables are shown below:

Algorithms	ARI	MI	PAC	DBI	DI	SAW	RS	CI	BHI	CHI	Top
KM (Euclidean)	1	1	6	2	1	1	1	1	1	1	90%
PAM (Euclidean)	2	2	9	1	4	2	2	2	2	2	80%
HC (Euclidean)	5	5	1	7	5	5	5	5	5	5	10%
HC (Diana)	5	5	1	7	5	5	5	5	5	5	10%
KM (Spearman)	5	5	1	7	5	5	5	5	5	5	10%
PAM (Spearman)	5	5	1	7	5	5	5	5	5	5	10%
NMF (Euclidean)	4	4	7	4	2	4	4	4	4	4	10%
KM (MI)	9	9	10	6	10	10	10	10	10	9	0%
PAM (MI)	9	9	5	5	9	9	9	9	9	10	0%

Algorithms	ARI	MI	PAC	DBI	DI	SAW	RS	CI	BHI	CHI	Top
NMF (Divergence)	3	3	8	3	3	3	3	3	3	3	0%

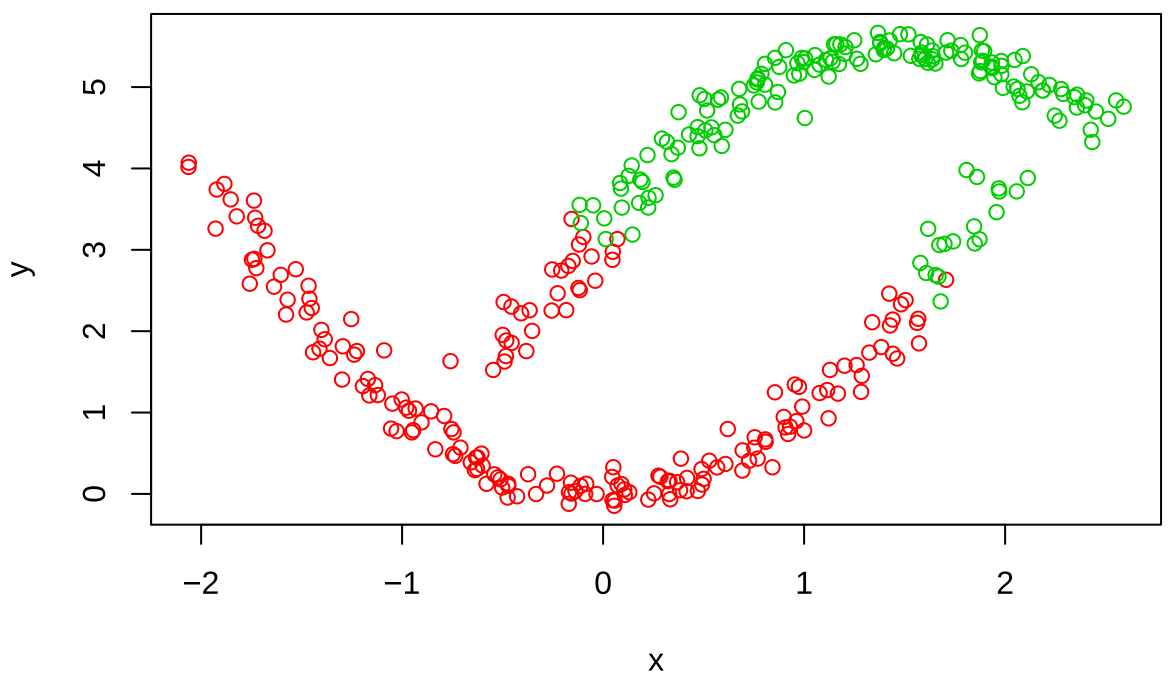
Algorithms	Top	Sum	Weight
KM (Euclidean)	90%	16	24.29%
PAM (Euclidean)	80%	28	13.88%
NMF (Divergence)	0%	35	11.1%
NMF (Euclidean)	10%	41	9.48%
HC (Euclidean)	10%	48	8.1%
HC (Diana)	10%	48	8.1%
KM (Spearman)	10%	48	8.1%
PAM (Spearman)	10%	48	8.1%
PAM (MI)	0%	83	4.68%
KM (MI)	0%	93	4.18%

Using hierarchical clustering with Wald's method on the meta-consensus matrix across the ten algorithms, we can obtain meta-consensus classes. The following table shows how the different methods compare, based on the number of matches to the true class labels.

	Matches
KM (Euclidean)	311
PAM (Euclidean)	306
Meta	304
NMF (Divergence)	291
NMF (Euclidean)	287
PAM (MI)	181
KM (MI)	179
HC (Euclidean)	136
HC (Diana)	136
KM (Spearman)	136
PAM (Spearman)	136

Let's visualize how the best algorithm, K-Means using euclidean distance, separates the clusters:

K−Means (euclidean) clustering of two atypical parabolic shapes (wor

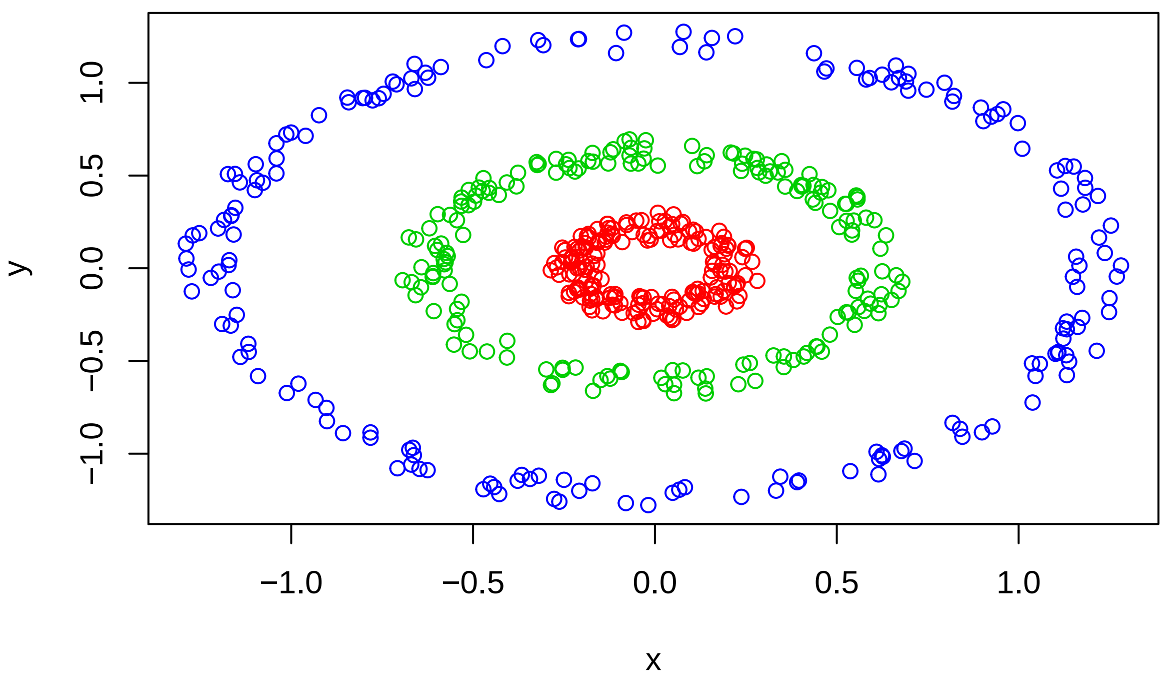


It appears that even the best algorithm cannot make a parabolic division in the feature space.

7.2 Rings Dataset

The following three-cluster dataset is our second simulation:

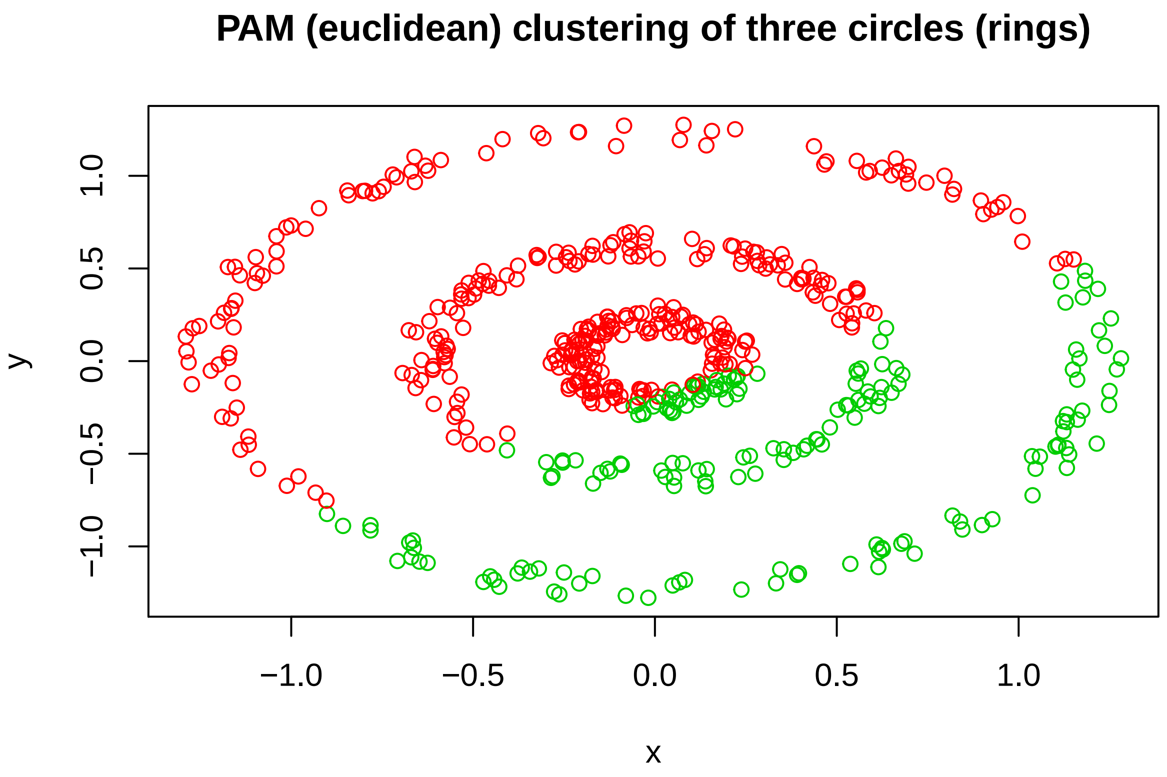
Three clusters with atypical ring shapes (circles)



Algorithms	PAC	DBI	DI	SAW	RS	CI	BHI	CHI	Top
NMF (Euclidean)	6	2	1	2	2	1	8	1	75%
KM (Euclidean)	7	9	9	4	4	2	2	9	25%
KM (MI)	8	1	8	9	9	9	1	8	25%
PAM (Euclidean)	9	4	3	1	1	7	3	7	25%
NMF (Divergence)	5	3	2	3	3	8	9	2	25%
HC (Euclidean)	1	5	4	5	5	3	4	3	12.5%
HC (Diana)	1	5	4	5	5	3	4	3	12.5%
KM (Spearman)	1	5	4	5	5	3	4	3	12.5%
PAM (Spearman)	1	5	4	5	5	3	4	3	12.5%

Algorithms	Top	Sum	Weight
NMF (Euclidean)	75%	23	15.84%
HC (Euclidean)	12.5%	30	12.14%
HC (Diana)	12.5%	30	12.14%
KM (Spearman)	12.5%	30	12.14%
PAM (Spearman)	12.5%	30	12.14%
PAM (Euclidean)	25%	35	10.41%
NMF (Divergence)	25%	35	10.41%
KM (Euclidean)	25%	46	7.92%
KM (MI)	25%	53	6.87%

	Matches
PAM (Euclidean)	201
KM (MI)	181
KM (Euclidean)	180
NMF (Euclidean)	180
HC (Euclidean)	169
HC (Diana)	169
KM (Spearman)	169
PAM (Spearman)	169
NMF (Divergence)	161
Meta	146



Again, we are only able to make linear separations.

8 References