LUDDY
SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

EMD–Based Consensus Clustering

Team Members: Noah Probst, Mentor: Md Taufique Hussain

What is Consensus Clustering?

Consensus clustering is a method of aggregating (potentially conflicting) results from multiple clustering algorithms. Another term for this is called cluster ensembles.

Figure 1

For this research project we used a dataset called Cassini. This was most viable to us because it was two dimensional (Which is uncomplicated to plot… see Fig. 1)

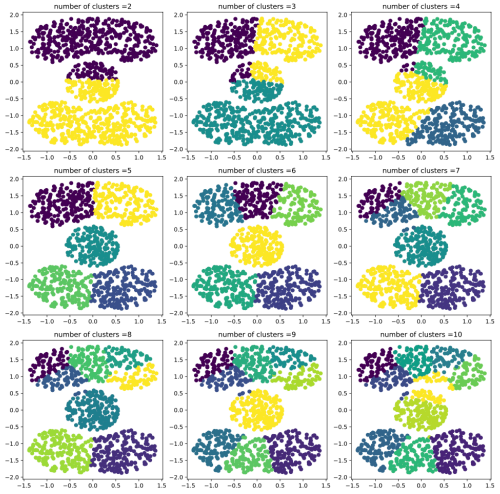

Figure 1: Different partitioning of Cassini dataset using k-means with different number of clusters centers

- Cassini dataset has 1000 datapoints which are drawn from uniform distribution on 3 structures. We generated the above list of partitions by running k-means clustering 9 times with different ‘k’ values (clusters).
- From the plots of the k-means, we can see that the quality changes with the change in number of clusters.
- Our research comes in when we need to find a consensus plot (1), from the original (9) in figure 1.

Our Method

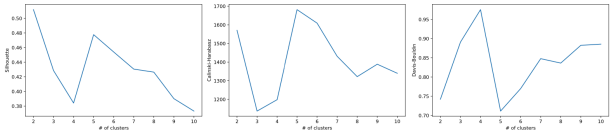

Figure 2: Quality of different partitionings of Cassini dataset. For first two, more is better. For the third, less is better.

- Below are the algorithms we used to come up with our new method of EMD-Based CC.

Algorithm 1 An overview of consensus clustering of two input partitions.
Input and Output: Inputs are two partitions P_1 and P_2 and output is a consensus partition P_c of the same data

procedure ConsensusTwo(P_1, P_2)
  $F \leftarrow EMD(P_1, P_2)$ ▷ Get flow by solving EMD
  Prune insignificant edges from F
  Form P_c by merging each connected component to form a cluster
  return P_c

Algorithm 2 An overview of consensus clustering of multiple input partitions.
Input and Output: Input is a list of partitions P_1, P_2, ..., P_n and output is a consensus partition P_c of the same data

procedure ConsensusAll($P_1, P_2, ..., P_n$)
  while Length of the partition list is two or more **do**
    Find closest two partitions P_a and P_b in the partition list according to EMD
    $P_c \leftarrow$ ConsensusTwo(P_a, P_b) ▷ Get consensus of two closest partitions
    Remove P_a and P_b from the pool
    Add P_c to the pool
  return only remaining partition in the pool

What is EMD?

To understand our algorithms, one must first understand Earth Movers Distance. EMD is a method to evaluate dissimilarity between two multi-dimensional distributions in some feature space where a distance measure between singles features, which we call ground distance is given. EMD essentially just optimizes the work needed to be done. When put into the algorithm, it ‘prunes’ the 9 plots into 1. In figure 3 (Top of third column on poster), the right plot is the final output. This is the plot we return in the two algorithms.

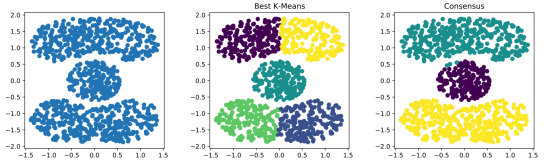

Figure 3: Scatter plot of Cassini data and clustering. First plot represents the structure of the data. Second plot represents clustering structure of best k-means clustering. Third plot represents clustering structure of consensus of many k-means clusterings.

Conclusion and Possible Next Step:

- In **conclusion**, and looking at our research project overall, we first took a dataset (Cassini), and took the k-means of it nine times.
- Then, we used two algorithms and EMD to ‘prune’/narrow down those nine into one consensus clustering plot in figure 3.
- Looking into the future and thinking of the next step possibly for this research project, we look to compare our method of consensus clustering with other similar ones.
- One we have considered to try and compare against is a function in a package called CLUE. This function is in ‘R’ code and is named cl_consensus. If we were to use the same dataset on the cl_consensus algorithm, then we could do a clean comparison and improve our method.
- In doing this, there is a problem because our method is in python, and cl_consensus is in ‘R.’ This problem can be solved however by using rpy2 in python to import ‘R’ libraries. Once there, we can use k-means the same on the same dataset, and then compare the outputs.
- Another next step idea is to look at the different parameters and test different ones out.


```
import rpy2
print(rpy2.__version__)
```