VAT for Preference Data

ANUKARSH KHANDELWAL
ENROLMENT NUMBER 16116007
DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
Email:akhandelwal@ec.iitr.ac.in

YASH AGARWAL
ENROLMENT NUMBER 16116078
DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
Email:yagarwal@ec.iitr.ac.in

***Abstract*—Visual Assessment Tendency(VAT) is a method used for visually assessing the number of clusters in a given dataset. Conventional VAT fails to give correct output on many cases of preference data. In this project, we aim to modify VAT to deal directly with preference data. We use Chu–Liu/Edmond's Algorithm for reordering preference matrices, which is an essential step for getting VAT.**

I. INTRODUCTION

Clustering is a method of partitioning a bunch of data points or population into different groups such that data points in each of the separated groups have similar features incorporated within them. Clustering helps in understanding the relations between our data points and, at the same time, differences existing between the data points. In simple words, the aim is to segregate groups with similar features and assign them into clusters. Clustering has played a part in a vast number of fields, which include bioinformatics, statistics, information retrieval, data mining, psychology, machine learning, computer graphics, etc.

In today's time, where data is available in every form and factor, understanding these data helps in providing meaningful information for making better decisions. There are various methods of clustering for different types of datasets.We are focusing on the Visual Assessment Tendency(VAT) method for clustering in the preference type dataset.

The conventional method is visually assessing the clustering tendency of a group of data points when their representation is in the form of object vectors or in terms of dissimilarity values among each pair of the data points considered. Data points are reordered to form an updated matrix, which is then shown as an intensity image of dark blocks clustered along the diagonal.

The above VAT is for the dataset that in which each of the data points has some n features associated with them, and the datapoint can be plotted in an n-dimensional plane where each of the dimensions corresponds to the feature vector. Our aim is to use this method on the preference dataset.

Preference dataset is a type of dataset where only relative information is given between some/all pairs of indexes. A particular cell (p_{ij}) in a preference matrix (P) represents the preference of datapoint **'i'** over datapoint **'j'**.

II. THEORY OF VAT FOR THE CONVENTIONAL DATASET

To understand VAT, we have created a dataset having 2 featured data points to visually show the procedural steps in the algorithm.

Fig.1 is a scatterplot for a randomly generated 2-dimensional synthetic dataset having a total of 1,000,000 points, distributed among 4 clusters with ground truth values.

Fig.2 shows the Minimum Spanning Tree generated by VAT calculated using Euclidean distance between data points. The three longest edges, which are used for separating clusters, are shown in green. This MST is used by VAT to reorder dissimilarity matrices for this dataset.

Fig.3 is the output of VAT on this dataset, is an image matrix where the color of a cell is a reflection of value in its respective cell. The clustering tendency can be easily assessed from the received image. Broadly, it can be divided into two clusters. Looking closely in the bigger square, three different dark squares can be seen.

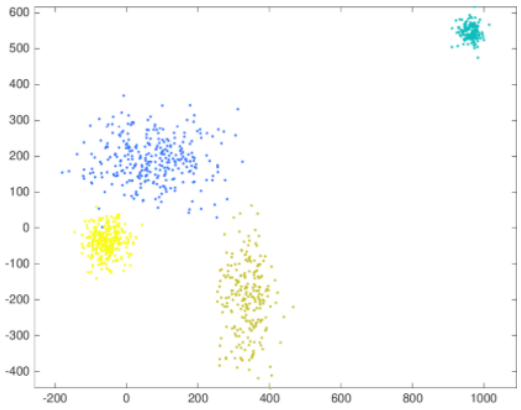

Fig. 1. Scatter Plot of dataset plotted as considering X and Y axis as features.

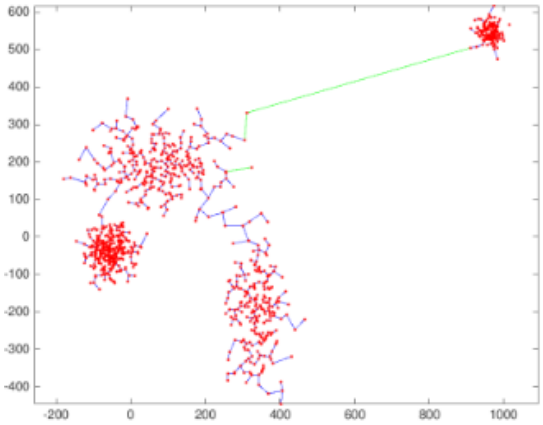

Fig. 2. MST of the dataset.Longest 3 edges are shown green.

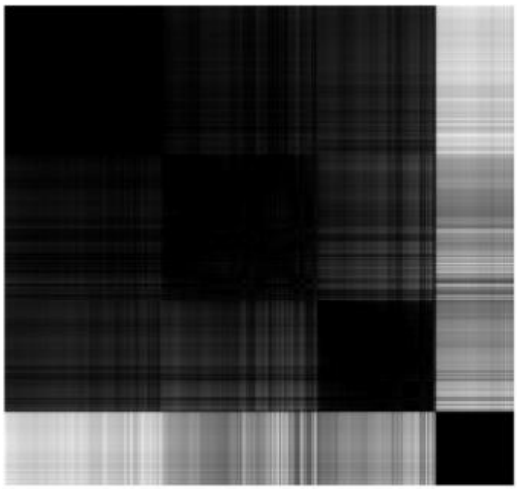

Fig. 3.VAT of the reordered dissimilarity matrix.

III. PREVIOUS APPROACH APPLIED

The previous methodology applied to the preference dataset has been discussed in this section.

Given a group of n objects, a fuzzy preference relation is specified by a preference matrix of size n x n, as shown in Fig.4, where each of the matrix element p_{ij} specifies the preference of data point **'i'** over **'j'** in terms of numerical value.

$$P = \begin{pmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{n1} & \cdots & p_{nn} \end{pmatrix}$$

Fig. 4 .Preference matrix

There is a defined range of values of preference, which is $p_{ij} \in [0, 1]$, in which $p_{ij} = 0$ states that there is no preference of datapoint 'i' over datapoint 'j', $p_{ij} = 1$ states an absolute preference, and pij = ½ states equal prefer.

A straightforward way to convert such a reciprocal additive fuzzy preference matrix into a symmetric dissimilarity matrix is by $d_{ij} = \max(p_{ij}, p_{ji}) - 0.5$

We have applied this approach to two examples and have classified them as 'Good' and 'Bad' examples.

Good Example: Considering a preference matrix P_1, we can calculate its dissimilarity matrix D_1 using the aforementioned formula. P_1 can be translated to preference graph G1, as shown in Fig. 5.

$$P_1 = \begin{pmatrix} 0.5 & 1 & 1 & 1 \\ 0 & 0.5 & 0.5 & 0.5 \\ 0 & 0.5 & 0.5 & 0.5 \\ 0 & 0.5 & 0.5 & 0.5 \end{pmatrix} \quad D_1 = \begin{pmatrix} 0 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 \end{pmatrix}$$

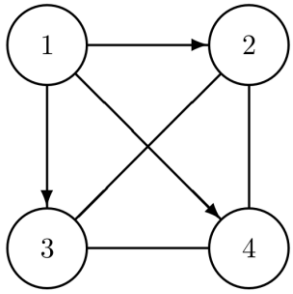

Fig. 5. Preference Graph G1

VAT results for the P_1 and D_1 are shown in Fig. 6 and Fig. 7 respectively. Fig. 6 fails to give any useful information, whereas Fig. 7 clearly shows the presence of two clusters, which is the expected outcome.

Fig. 6 shows VAT for preference matrix P_1

Fig. 7 shows VAT for dissimilarity matrix D_1.

Bad Example: Now, considering a preference matrix P_2 and D_2 is made using the previous formulae, we get Fig. 9 as a result of P_2 and Fig. 10 as a result of D_2.

$$P_2 = \begin{pmatrix} 0.5 & 0.5 & 1 & 0.5 \\ 0.5 & 0.5 & 0.5 & 1 \\ 0 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 & 0.5 \end{pmatrix} \quad D_2 = \begin{pmatrix} 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 \\ 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \end{pmatrix}$$

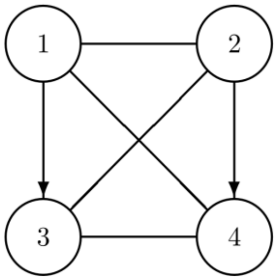

Fig. 8. Preference Graph G2

Both Fig. 9 and Fig. 10 do not yield expected results. Many more such cases can be made where VAT does not yeild images that appropriately reflect the preference structure.

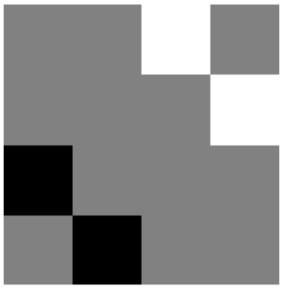

Fig. 9 shows VAT for preference matrix $\mathbf{P_2}$

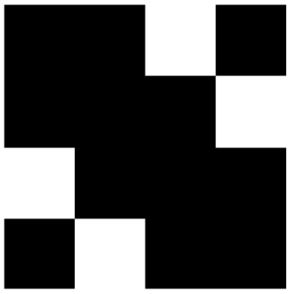

Fig. 10 shows VAT for dissimilarity matrix $\mathbf{D_2}$.

IV. METHODOLOGY

In conventional VAT, we have values of n features for every node where the pairwise euclidean distance between any two nodes can easily be calculated, whereas in the preference matrix, we don't have any ground truth and only relative information is available.

This is analogues to conventional VAT with a difference that the edges here are directed. We know that a node is preferred over another by some value. Similar to conventional VAT, we can find MST for reordering, and VAT can be visualized. Edmonds' algorithm or Chu–Liu is an algorithm for finding a spanning arborescence(for a vertex u called the root and any other vertex v, there is exactly one directed path from u to v) of maximum weight. It is the directed analogue of the minimum spanning tree problem.

The preference matrix is now converted into a graph, and we can apply Edmond's algorithm on it considering the fact all other nodes can be visited from the root which is always true because in our dataset, each of the data points have incoming and outgoing edges with each of the other data points. So there is no constraint of applying Edmond's algorithm in our experiments.

A. Edmond's algorithm:

In graph theory, this algorithm is used for finding the shortest arborescence of a directed graph. In other words, it is used to find the minimum spanning tree for a directed graph.
We make use of the fact that the final MST every node except root will have exactly one incoming edge, and the root node will have zero incoming edge. Intuitively, for every node, we select an incoming edge with minimum value greedily. If we encounter a cycle at any point, we must add an incoming edge to a node of the cycle and remove an edge to cycle. The selection of edges to be added and removed is also done greedily by considering the difference of value that will occur after the procedure.

Preprocessing for this algorithm to work requires removal of all incoming edges to the ROOT, this is done to make sure that ROOT will be root for any solution. Also, if there are multiple edges between any pair nodes in the same direction, only the highest scoring edge should be considered, and the rest of the edges should be removed.
This algorithm works in two stages: contracting and expanding.

Contracting Stage
For this stage, we essentially need two more data structures to store the best incoming edge and edge that is being kicked out. Let the names of these data structures be bestEdge and kicks. Following steps take place in this stage:
> For every node v that is not root, the edge with the highest scoring value is set as bestEdge[v].
> Continue doing this step for the node connecting the edge found in the previous step until a cycle is formed.
> If at any step a cycle C is formed:
All the nodes of cycle C should be contracted into a new node N. All the edges connecting nodes in cycle C should be modified, and N should replace all these nodes. During this modification, the value of concerned edges is decreased by the value of the corresponding edge that will be kicked out. The edge being kicked out is set against the edges connecting N in data structure kicks. This will give us a new graph with no cycle.
> Repeat above mentioned steps there is an incoming edge for every node except the root. Also ensure that there is no cycle formed.

Expanding Stage
After the previous stage, bestEdge data structure will hold the value of edge with the best incoming score against the names of nodes, including the new nodes formed because of cycle formations. In this stage, we just traverse the bestEdge data structure in the reverse direction of the order values that were added to it, and we fix the edge e and replace all the edges in kicks[e] from bestEdge with e.

For this particular case, we consider every entity as a node, and every couple of nodes have two edges with opposite directions. For preprocessing the given preference matrix, each edge is given a weight. The weights between any two nodes 'a' to 'b' were replaced by a value equal to $\mathbf{x / |x - 0.5|}$ where **x** is the preference value of 'a' over 'b', this value always lies between 0 and 1. The graph of the implemented function is shown in Fig. 11. This change in weight assigns edges connecting two equally preferred edges high values, and absolutely preferred edges will have the least values. For x = 0.5, we have replaced the value with 55.

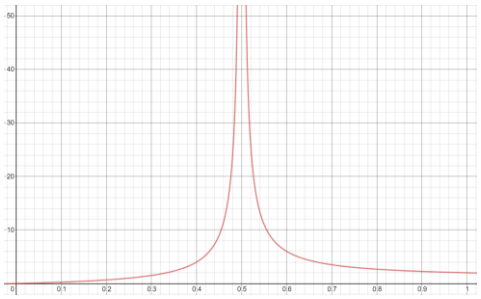

Fig. 11. Shows the graph of our assigned function to the edge values.

Before starting the recursion to find MST, the least preferred node is required to be found. We will be starting Edmond's algorithm recursion from this particular node. For every pair of nodes, we increment the score of the node, which is less preferable using the preference matrix. The node with the highest score is selected as the first node for finding MST.

We start recursion for using the least preferred node until all nodes except one get the highest scoring incoming edge. We use edmond's algorithm to solve cycles, and after expanding stage, we get the required MST.

To apply VAT algorithm to this, we need the reordered matrix. For this step, we used another data structure called the priority queue. Starting from the root obtained from the MST, all the edges connected to the node are added to the priority queue. Edge with the smallest value is then popped from the priority queue, and all edges connected to this new node are added to the priority queue. This is repeated until we pop every node and get an order of nodes.
This order is used to generate reordered matrices. We use the resultant reordered matrix to find a pairwise distance between every node, and a heatmap is plotted according to this value, where 0 is the darkest and 1 is the lightest color. The resultant image can be visualized for the assessment of clustering tendency.

V. EXPERIMENTAL RESULTS

A. Example 1 (Artificial dataset)

To verify our algorithm, we made an Artificial dataset. We intentionally made the dataset such that it contains 4 clusters. The preference matrix is shown below in Fig. 12 where any element a_{ij} is the winning probability of team **'i'** over team 'j'. The preference matrix is made such that :
1)Teams 11,3,4 are the best teams.
2)Teams 1,9 are the next best teams.
3)Teams 2,5,6,10,12 are average.
4)Teams 7,8 are the worst teams.

	1	2	3	4	5	6	7	8	9	10	11	12
1	0.5	0.63	0.38	0.4	0.67	0.7	0.77	0.79	0.55	0.65	0.35	0.6
2	0.37	0.5	0.23	0.24	0.55	0.58	0.7	0.72	0.37	0.53	0.23	0.49
3	0.62	0.77	0.5	0.55	0.8	0.84	0.91	0.94	0.68	0.78	0.45	0.74
4	0.6	0.76	0.45	0.5	0.81	0.82	0.9	0.94	0.66	0.77	0.4	0.75
5	0.33	0.45	0.2	0.19	0.5	0.6	0.67	0.68	0.32	0.44	0.18	0.46
6	0.3	0.42	0.16	0.18	0.4	0.5	0.66	0.67	0.3	0.41	0.15	0.43
7	0.23	0.3	0.09	0.1	0.33	0.34	0.5	0.53	0.22	0.33	0.08	0.28
8	0.21	0.28	0.06	0.06	0.32	0.33	0.47	0.5	0.25	0.3	0.05	0.27
9	0.45	0.63	0.32	0.34	0.68	0.7	0.78	0.75	0.5	0.65	0.3	0.61
10	0.35	0.47	0.22	0.23	0.56	0.59	0.67	0.7	0.35	0.5	0.21	0.48
11	0.65	0.77	0.55	0.6	0.82	0.85	0.92	0.95	0.7	0.79	0.5	0.75
12	0.4	0.51	0.26	0.25	0.54	0.57	0.72	0.73	0.39	0.52	0.25	0.5

Fig. 12. Preference Matrix of the artificial dataset

Result
Clusters formed from top left to bottom right in Fig. 13:
Cluster 1 : Teams(11,3,4)
Cluster 2 : Teams(1,9)
Cluster 3 : Teams(2,5,6,10,12)
Cluster 4 : Teams(7,8)

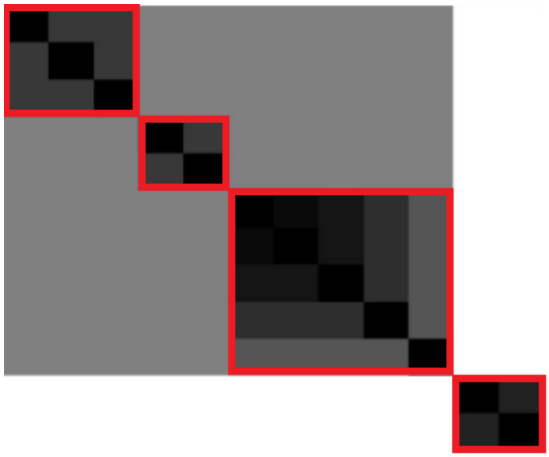

Fig. 13. iVAT of the artificial dataset. Clusters formed are marked with red-colored squares.

Observations
The result obtained is perfectly fine as the expected 4 clusters are clearly seen.

For getting results on a real-life dataset, we have chosen to implement our algorithm on the top 3 football leagues(Barclays Premier League, La Liga, and Serie A), cricket league of India IPL(Indian Premier League) and Youtube Comedy dataset.

B. Example 2 (Barclays Premier League):

Dataset : We have chosen 22 teams that played in the Premier League games from 2006-2007 season till 2016-2017 season. To get a rough idea of how teams perform overall ,the stats are shown in Fig. 14. They are sorted according to their score, which is calculated by **[total wins of team + (total draws/2)]/[overall matches played by team]**.

The total number of matches of each team may be different because BPL has relegation policies that at the end of the season bottom 3 teams are relegated to the second division league, and the top 3 teams from the second division get promoted to Barclays Premier League.

Preference matrix: Each element a_{ij} in preference matrix is calculated as [total wins of the team **'i'** over **'j'** + 0.5*(draws between team **'i'** and **'j'**)] / [total matches played between team **'i'** and team **'j'**]

	TEAMS	WINS	DRAW	LOST	TOTAL MATCHES	SCORE
1	Manchester United	222	72	66	360	0.716667
2	Chelsea	209	86	65	360	0.7
3	Arsenal	189	91	80	360	0.651389
4	Manchester City	192	76	92	360	0.638889
5	Liverpool	179	99	82	360	0.634722
6	Tottenham Hotspur	163	94	103	360	0.583333
7	Everton	142	107	111	360	0.543056
8	Stoke City	89	82	129	300	0.433333
9	Aston Villa	86	94	128	308	0.431818
10	Fulham	68	74	110	252	0.416667
11	Newcastle United	87	72	141	300	0.41
12	Blackburn Rovers	53	51	88	192	0.408854
13	West Ham United	89	87	152	328	0.403963
14	West Bromwich Albion	67	81	120	268	0.401119
15	Bolton Wanderers	51	43	98	192	0.377604
16	Sunderland	75	80	151	306	0.375817
17	Burnley	27	30	55	112	0.375
18	Wigan Athletic	51	63	108	222	0.371622
19	Watford	29	22	59	110	0.363636
20	Norwich City	27	31	62	120	0.354167
21	Middlesbrough	23	40	61	124	0.346774
22	Hull City	30	37	87	154	0.314935

Fig. 14. Statistics of the teams in the dataset considered

Result

Clearly, there are mainly 2 clusters seen in Fig. 15 marked with red squares, but some subclusters in Cluster 2 are marked with yellow-colored squares.

Clusters formed from top left to bottom right in Fig. 15:

Cluster 1: Teams(Chelsea, Manchester United, Manchester City, Tottenham Hotspur, Arsenal, Liverpool)

Cluster 2 :

Subcluster 1: Teams(Stoke City, West Ham United)

Subcluster 2: Teams(Wigan Athletic, Norwich City, West Bromwich Albion, Hull City, Middlesbrough, Sunderland, Burnley, Aston Villa, Fulham, Watford, Blackburn Rovers)

Subcluster 3: Teams(Newcastle United)

Subcluster 4: Teams(Bolton Wanderers, Everton)

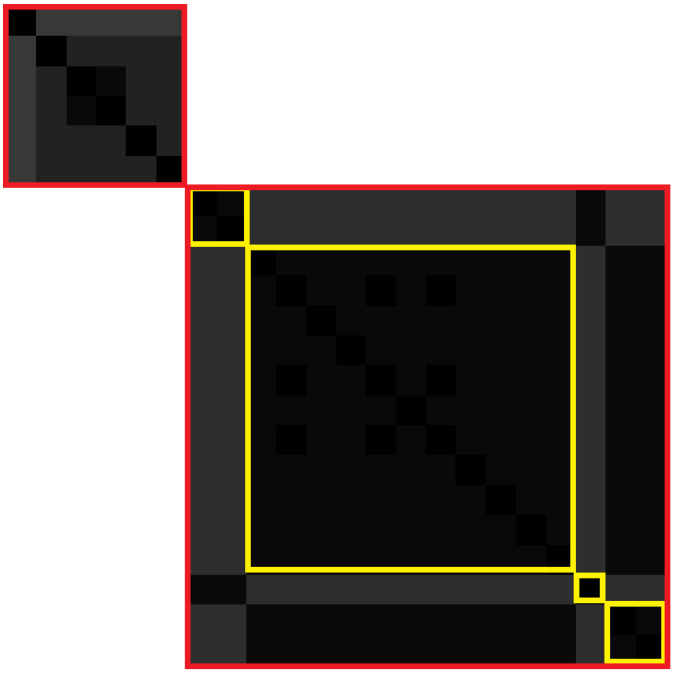

Fig. 15. iVAT of the artificial dataset. Clusters formed are marked with red colored squares.Subclusters marked with yellow squares.

Observation

Top teams of the league are together in Cluster 1. Apparently, the 6 teams clustered together are also the top 6 teams in the statistics table we have.

Cluster 2 comprises the average teams in the league. Teams in Subcluster 1 and 4 have close competitions with each other.

The result looks good because Cluster 1 consists of teams that have a score of 0.58 or greater, and Cluster 2 has teams having a score of 0.43 and less except Everton(score of 0.54).Although there are too many teams in Cluster 2, the range of scores of teams in cluster lies between 0.43 to 0.34, and the difference is not even 0.1, which makes it clear why all these teams are clustered in a single cluster. The same goes for teams in Cluster 1 where the maximum difference in the range of values is 0.13.

C. Example 3 (La Liga)

Dataset: We have chosen 25 teams which played from season 2006-2007 to 2017-2018. Team stats are shown in Fig. 16. Again the teams are sorted according to their scores using the same score formulae as in our previous case of Barclays Premier League. It should be noted that ordering has a partial say in the clustering because there might be teams that may be far away in rankings, but they may have close competition with each other.

Preference matrix: Calculated using the formula as for the Barclays Premier League.

	TEAMS	WINS	DRAW	LOST	TOTAL MATCHES	SCORE
1	Barcelona	297	75	44	416	0.804087
2	Real Madrid	294	60	62	416	0.778846
3	Atletico de Madrid	224	82	110	416	0.637019
4	Villarreal	169	93	116	378	0.570106
5	Valencia	187	94	135	416	0.5625
6	Sevilla	188	88	140	416	0.557692
7	Atletico de Bilbao	157	98	161	416	0.495192
8	Real Sociedad	111	79	124	314	0.479299
9	Mallorca	84	62	102	248	0.46371
10	Espanol	136	107	173	416	0.455529
11	Malaga	113	86	149	348	0.448276
12	Celta de Vigo	80	54	106	240	0.445833
13	Betis	93	81	138	312	0.427885
14	Santander	57	65	88	210	0.42619
15	Getafe	117	93	174	384	0.425781
16	Zaragoza	64	54	96	214	0.425234
17	Rayo Vallecano	62	28	92	182	0.417582
18	Deportivo	85	107	150	342	0.404971
19	Eibar	37	32	63	132	0.401515
20	Valladolid	45	51	80	176	0.400568
21	Osasuna	85	83	148	316	0.400316
22	Levante	86	79	151	316	0.397152
23	Almeria	55	48	105	208	0.379808
24	Sporting de Gijon	54	49	107	210	0.37381
25	Granada	49	50	115	214	0.345794

Fig. 16. Statistics of the teams in the La Liga dataset considered

Results

Clusters are mentioned in order from top left to bottom right in Fig. 17.

Cluster 1 : (Barcelona, Real Madrid0
Cluster 2 : (Rayo Vallecano)
Cluster 3 : (Levante)
Cluster 4 : (Granada,Mallorca)
Cluster 5 : (Atletico de Madrid, Villarreal)
Cluster 6 : (Eibar, Malaga, Almeria, Valladolid,
Sporting de Gijon, Zaragoza,
Sevilla, Getafe, Santander, Real Sociedad
Espanol, Celta de Vigo)
Cluster 7 : (Osasuna, Betis, Deportivo)
Cluster 8 : (Atlético de Bilbao , Valencia)

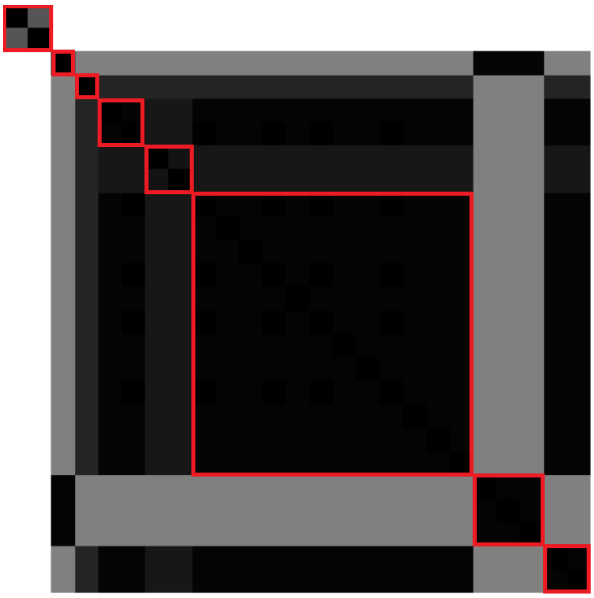

Fig. 17. iVAT of the artificial dataset. Clusters formed are marked with red-colored squares.

Observations

The top teams Barcelona and Real Madrid are dominant(only the two teams which have a score greater than 0.75) to all other teams form a single cluster(Cluster 1).
The above-average teams like Valencia and Athletic Bilbao are clustered together in Cluster 8.
The same goes for Sevilla and Real Sociedad, which are above average teams in the same cluster(Cluster 5).
Teams clustered in Cluster 6 are average.
Below average teams, Osasuna, Betis, and Deportivo, are clustered together in Cluster 7.
Granada and Mallorca which are far apart in the statistics table are clustered together as they have equal head to head competitions(2 draws between them and 1 win for each of the teams).
Overall the results are good. The teams like Barcelona and Real Madrid, which have high preference values against all other teams except themselves, are together clustered in the same group. Average teams are clustered together.

D. Example 4 (Serie A):

Dataset: We have chosen 24 teams that played from season 2000-2001 to 2012-2013. The scores are calculated using the same formulae as used for the previous two leagues. The teams, along with their statistics, are shown in Fig. 18.

	TEAMS	WINS	DRAW	LOST	TOTAL MATCHES	SCORE
1	Juventus	218	111	61	390	0.701282
2	Milan	231	112	81	424	0.676887
3	Inter	229	107	88	424	0.666274
4	Roma	213	110	101	424	0.632075
5	Napoli	97	68	77	242	0.541322
6	Lazio	166	120	138	424	0.533019
7	Fiorentina	147	98	125	370	0.52973
8	Udinese	162	105	157	424	0.505896
9	Palermo	114	86	118	318	0.493711
10	Genoa	75	56	85	216	0.476852
11	Sampdoria	102	92	118	312	0.474359
12	Parma	126	109	151	386	0.467617
13	Chievo	104	112	144	360	0.444444
14	Bologna	90	99	131	320	0.435938
15	Catania	72	73	105	250	0.434
16	Cagliari	87	96	135	318	0.424528
17	Atalanta	88	103	139	330	0.422727
18	Brescia	41	59	76	176	0.400568
19	Reggina	61	84	113	258	0.399225
20	Siena	69	100	143	312	0.38141
21	Empoli	40	40	78	158	0.379747
22	Livorno	40	51	85	176	0.372159
23	Torino	39	69	90	198	0.371212
24	Lecce	54	74	126	254	0.358268

Fig. 18. Statistics of the teams in the Serie A dataset considered

Results

Clusters are mentioned in order from top left to bottom right in Fig. 19.

Cluster 1:(Livorno)

Cluster 2:(Genoa)

Cluster 3:(Palermo, Sampdoria, Catania, Cagliari, Siena)

Cluster 4:(Lazio)

Cluster 5:(Lecce, Florentina)

Cluster 6:(Inter, Milan, Roma, Juventus)

Cluster 7: (Palermo, Torino, Empoli, Brescia, Chievo, Bologna, Udinese, Reggina)

Cluster 8:(Napoli)

Cluster 9:(Atalanta)

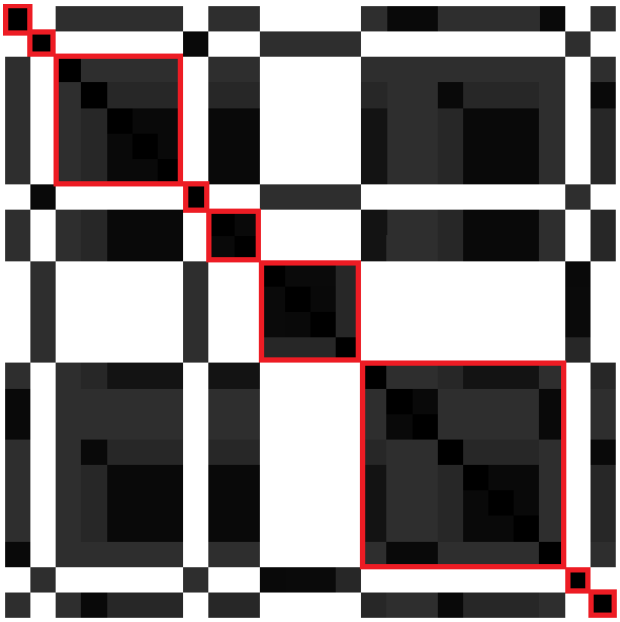

Fig. 19. iVAT of the Serie A dataset. Clusters formed are marked with red-colored squares.

Observations

Top teams Inter, Milan, Roma and Juventus are together in the Cluster 6.

Average teams are placed in Cluster 7.

One exception can be seen in Cluster 5 that Lecce, which is the last team in the table, is in the same cluster with Florentina which is 7th in ranking because they have equal head to head competitions with each other.

Again as seen from the previous results of the two leagues, top teams that have high scores in the statistics table are separately clustered together. Here also the top 4 teams have a score greater than 0.63, and the fifth is much below with a value of 0.54, so the top 4 teams are clustered together.

E. Example 5 (Indian Premier League)

Dataset: We have chosen 9 teams that played from season 2008 to 2019. The scores are calculated using the formulae **[total wins of team + (total NO RESULT/2)]/[overall matches played by team]**. The teams, along with their statistics, are shown in Fig. 20.

	TEAMS	WINS	NO RESULT	LOST	TOTAL MATCHES	SCORE
1	Chennai Super Kings	72	0	47	119	0.605042
2	Mumbai Indians	82	0	54	136	0.602941
3	Sunrisers Hyderabad	36	0	31	67	0.537313
4	Rajasthan Royals	55	1	51	107	0.518692
5	Kolkata Knight Riders	66	0	63	129	0.511628
6	Royal Challengers Bangalore	61	2	67	130	0.476923
7	Kings XI Punjab	58	0	71	129	0.449612
8	Delhi Daredevils	49	2	75	126	0.396825
9	Pune Warriors	10	1	30	41	0.256098

Fig. 20. Statistics of the teams in the IPL dataset considered

Results

Clusters are mentioned in order from top left to bottom right in

Fig. 21.
Cluster 1: Teams(Chennai Super Kings)
Cluster 2: Teams(Rajasthan Royals, Royal Challengers Bangalore, Kolkata Knight Riders)
Cluster 3: Teams(Delhi Daredevils, Pune Warriors, Kings XI Punjab, Mumbai Indians, Sunrisers Hyderabad)

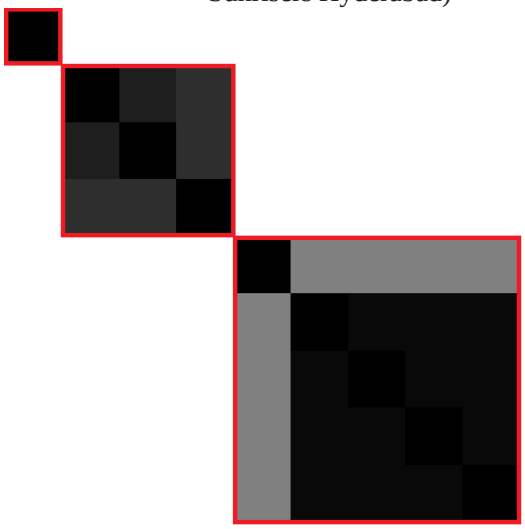

Fig. 21. iVAT of the IPL dataset. Clusters formed are marked with red-colored squares.

Observations

Chennai Super Kings is at the top of the table and is clustered all alone.
Teams in Cluster 2 have close competitions with each other.
Teams in Cluster 3 have close competitions except for Pune Warriors, which mostly lost against Mumbai Indians and Sunrisers Hyderabad.

F. Example 5 (Youtube Comedy Dataset)

Dataset: YouTube Comedy dataset was an experiment conducted on TestTube, which is basically one of Youtube's experimental labs. The experiments were conducted between 2011 to 2012. For this experiment, a pair of videos were chosen randomly, and a user was asked to tell which video he found the funniest among the pair. Positions of the videos were chosen right and left randomly to cut out any bias. Videos were chosen from a set of weekly updated videos. Videos do not have any name or title associated with them
We have taken 32 videos among all the videos. Each video had at least a certain number of comparisons against each of the other videos.

Preference matrix: Each element $\mathbf{a_{ij}}$ in preference matrix is calculated as [Number of times video **'i'** is chosen over video **'j'**] / [Total comparisons between video **'i'** and video **'j'**]

Results

The resultant iVAT is shown in Fig. 22.

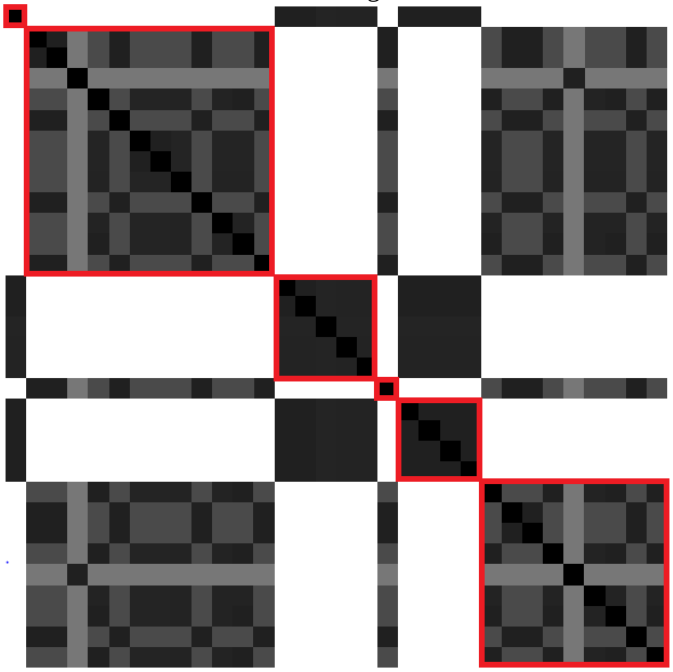

Fig. 22. iVAT of the Youtube Comedy Dataset. Clusters formed are marked with red-colored squares.

Observations

We have a clear indication that we have 4 major clusters in the i-VAT of the Comedy Dataset.
We may conclude two things from the result we have got.
First that each cluster may be at the same level of humor.
Second, there may be some elements of expression that may be common among them.This can be seen as unsupervised learning.

VI. Conclusion

The results are fine for any person who does not have any knowledge to get an idea about the teams in the competitive league.
For the artificial dataset, our algorithm is working perfectly fine. But in real life, we have cases where a top-performing team performs badly against some teams, which creates problems in clustering.
One result which is common in all the competitive league analysis we have done is that the top teams which are far better than other teams are clustered together in all the cases, which is an important step in cluster analysis.
After inspecting the datasets we have chosen, it is clear that there is no profound change in performances of the teams as we go down the score table. This is the reason that we do not find any cluster of poor performing teams specifically. Although in the first example of the Artificial dataset where we created a dataset such that there was a wide variety of performances among the teams, we did get clusters for poor performing teams. Teams in all the examples we have taken had preferences

with all other teams in their respective datasets. It is still to find the solution to the problem that when some teams do not have any comparisons with each other and are in the same dataset.

REFERENCES

[1] D. Kumar, J. Bezdek, M. Palaniswami, S. Rajasegarar, C. Leckie, and T. Havens, "A Hybrid Approach to Clustering in Big Data." IEEE Transactions on Cybernetics, vol. 46, no. 10, pp. 2372-2385, Oct. 2016.

[2] James C.Bezdek, Bonnie Spillman, Richard Spillman, "A fuzzy relation space for group decision theory."Fuzzy Sets and Systems Volume 1, Issue 4, October 1978, Pages 255-268.