An Experiment with Distance Measures for Clustering*

Ankita Vimal, Satyanarayana R Valluri, Kamalakar Karlapalem

International Institute Of Information Technology
Gachibowli, Hyderabad, India
{ankita@students., satya@, kamal@}iiit.ac.in

Abstract

Distance measure plays an important role in clustering data points. Choosing the right distance measure for a given dataset is a non-trivial problem. In this paper, we study various distance measures and their effect on different clustering techniques. In addition to the standard Euclidean distance, we use Bit-Vector based, Comparative Clustering based, Huffman code based and Dominance based distance measures. We cluster both synthetic datasets and one real life dataset using the above distance measures by employing k-means, matrix partitioning and dominance based clustering algorithms. We analyse the results of our study using a real life dataset of cricket and compare the accuracy of various techniques using synthetic datasets.

1 Introduction

Clustering is an important data mining technique that has a wide range of applications in many areas like biology, medicine, market research and image analysis among others. Clustering is the process of partitioning a set of objects into different subsets such that the data in each subset are similar to each other.

The similarity between various objects is defined by a distance measure. The distance measure plays an important role in obtaining correct clusters. For simple datasets where the data is multidimensional, Euclidean or Minkowski distance measures[1] are employed. But as the dimensions of the dataset increase, where each dimension denotes a specific aspect of the dataset, these simple distance measures may not be the right functions to compute the distance between the data objects.

In this paper, we experiment with different distance measures. We validate our results on synthetic datasets and one real life dataset of cricket. Results of our experiments show that different distance measures behave differently, and our Dominance measure, as described in section 4.6, captures the notion of distance more accurately, as the results are semantically better then those by the distance measures mentioned above and also by k-means[2] and DBSCAN [2].

In section 2, we mention related work; section 3 describes the datasets; in section 4, various distance measures are defined; section 5 describes various algorithms used; section 6 shows the results; in section 7, we interpret the results; and section 8 concludes the paper.

2 Related Work

Many distance measures have been proposed in literature for data clustering. Most often, these measures are metric functions; Manhattan distance, Minkowski distance, Hamming[3] are such common functions. Jaccard index, Cosine Similarity[4] and Dice Coefficient[5] are also popular distance measures. For non-numeric datasets, special distance functions are proposed. For example, edit distance[6] is a well known distance measure for text attributes.

3 Datasets

In this paper, we study Euclidean distance and four new distance measures which we define. We used a cricket dataset along with synthetic datasets generated using Syndeca[4] to verify the results.

Syndeca - Various datasets are generated using the Syndeca software[7]. The toolkit takes as input the number of points (integer), the number of dimensions (integer), the number of clusters (integer), maximum range for all the dimensions (real) and (0,1), 1 if subspace clusters are to be computed, 0 otherwise. Syndeca by default generates noise, but we have removed noise and considered only the cluster points.

Cricket - From the web[8], we downloaded a dataset of

*A full version of this paper is available [3]

International Conference on Management of Data
COMAD 2008, Mumbai, India, December 17–19, 2008

[1] http://en.wikipedia.org/wiki/Data_clustering
[2] http://en.wikipedia.org/wiki/K-means
[3] http://en.wikipedia.org/wiki/Data_clustering
[4] http://en.wikipedia.org/wiki/Jaccard_index
[5] http://en.wikipedia.org/wiki/Dice_coefficient
[6] http://en.wikipedia.org/wiki/Edit_distance
[7] http://cde.iiit.ac.in/syndeca
[8] http://www.cricinfo.com

350 players . The number of dimensions considered are six. The attributes of the players and sample tuples in the input file are as in the technical report [3]. A 6-tuple is formed for every player by adding the corresponding values of all the attributes in the file for that player.

4 Distance Measures

The number of points in a dataset is denoted by N. Each point is denoted by P_i, P_j and so on. k denotes the number of clusters and d denotes the number of dimensions of a point. D denotes the set of dimensions and D_{i_x}, D_{j_y} represent the subsets of dimensions of the points P_i and P_j respectively. l,m and x are simply used as indices.

4.1 Euclidean Distance

An N x N matrix M_e is calculated. For points with d dimensions, the Euclidean distance $M_e(P_i, P_j)$ between two points P_i and P_j is defined as follows:

$M_e(P_i, P_j) = \sqrt{\sum_{x=1}^{d} (P_{i_x} - P_{j_x})^2}$

where P_{i_x} and P_{j_x} represent the x^{th} dimension values of P_i and P_j respectively. Also, M_e is a symmetric matrix.

4.2 Bit-Vector Distance

An N x N matrix M_b is calculated. Each point has d dimensions and $M_b(P_i, P_j)$ is determined as a d-bit vector. This vector is obtained as follows:
If the numerical value of the x^{th} dimension of point P_i is greater than the numerical value of the x^{th} dimension of point P_j, then bit x of $M_b(P_i, P_j)$ is set to 1 and bit x of $M_b(P_j, P_i)$ is set to 0. All the bit vectors in M_b are then converted to integers.

4.3 Comparative Clustering Distance

An N x N matrix M_c is calculated. The set of data points are sorted in increasing order along every dimension. Thus, we generate d sorted lists of the data points, one along each dimension. For each point, the set of ranks it is assigned in the d sorted lists is obtained. The score of a data point is defined as the summation of its rank in the d sorted lists multiplied by a constant normalization factor. In our experiments, we used the normalization factor of 10. Based on these scores, the matrix M_c is generated as:
$M_c(P_i, P_j) = M_c(P_j, P_i) = |\text{score}(P_i)\text{-score}(P_j)|$
It can be seen that the matrix M_c is symmetric.

4.4 Huffman Codes

Huffman Encoding[9] is a method to encode data in the form of bits based on the frequency of various values. The Huffman encoding used here is ordered, that is, less the number of bits in the code, greater is the frequency of that particular value and more the number of bits, less is the frequency.
By determining the maximum value of each dimension, we divide the $[0 - maximum]$ range into a suitable number of bins. Each of the bins is assigned a unique string value. Based on the values of the dimensions of the points, the strings are assigned a frequency value. The frequencies of the strings are used to get the Huffman codes for each bin for every dimension. Hence, for each dimension of every point, a Huffman code is assigned based on the frequency of the string representing the bin to which it belongs to. Then, these Huffman codes are converted to integers. When two bins of a dimension have the same frequency, the integer representation of the Huffman codes reflects the relative values of the dimension for two different points. So, for every point, the original data values are mapped to a new set of integers.
The distance between two points is computed using Euclidean distance where Huffman encoded integers are considered instead of the actual dimension values.

[9] http://en.wikipedia.org/wiki/Huffman_encoding

4.5 Substring Matching

The Huffman codes for each point are obtained with respect to every dimension. In this manner we have a string of 0s and 1s for each point, with the Huffman codes of each dimension joined by a hyphen(-). For example, the following shows the values of attributes of a data point and the corresponding Huffman codes:

Attributes	297	11025	3	134
Huffman codes	11101110	1111101	00	11101100

The string 11101110-1111101-00-11101100- is formed by concatenating the Huffman codes of the attribute values above.
Using this string as a representation of each point, we determine the similarity using the substring matching method[1].
The function to compute similarity is described as follows:

- Given two strings, find the maximum lengths of the two strings which match.
- Count the number of characters which match. Now match the right and left unmatched parts of the strings and add the number of characters which match in each part to the previous count.
- This process goes on recursively till no substrings are found to match.

We get the total number of characters matching in both strings and obtain the percentage similarity (approx.) S_{sub} as follows :

$S_{sub} = \frac{(2 \times \text{no. of chars matched})}{(\text{total no. of chars in both strings})} \times 100$

In this method, we do not find clusters, but for every point we get a ranking of the other points based on this similarity percentage S_{sub}. Results of the same can be seen in the technical report [3].

4.6 Dominance measure

Let P_i and P_j be two points and P_{i_x}, P_{j_x} denote their x^{th} dimensions; if $P_{i_x} > P_{j_x}$, point P_i dominates point P_j with respect to dimension x. For every point, a vector of vectors called as *domination vector* is obtained, which gives information about the points it dominates and over which dimensions.

$P_i :<< P_{i_1}, D_{i_1} >, < P_{i_2}, D_{i_2} > ... < P_{i_l}, D_{i_l} >>$

$P_j :<< P_{j_1}, D_{j_1} >, < P_{j_2}, D_{j_2} > ... < P_{j_m}, D_{j_m} >>$

P_i denotes a point, D_{i_x} denotes the dimensions over which P_i dominates point P_{i_x} and D_{j_y} denotes the dimensions over which P_j dominates point P_{j_y}.

Similarity measure S_d between two points is defined as :

$$S_d(P_i,P_j) = \frac{\sum_{\forall P_{i_x}=P_{j_y}} \frac{|D_{i_x} \bigcap D_{j_y}|}{\frac{|D_{i_x}|+|D_{j_y}|}{2}}}{\frac{(l+m)}{2}}$$

Let Q be a common point that occurs in the domination vectors of both P_i and P_j. Let D_{i_x} and D_{j_y} be the subsets of the dimensions of Q in both the vectors respectively. We then count the number of common dimensions of D_{i_x} and D_{j_y} in D. Divide this by the average number of dimensions of D_{i_x} and D_{j_y}. The numerator denotes the summation of such values computed for all possible such Q points. The denominator denotes the average number of points in both the domination vectors P_i and P_j.

5 Algorithms

5.1 Matrix Partitioning and Splitting

An algorithm is developed on the lines of the Vertical Partitioning Algorithm[5]. The rows and columns of the value based matrices M_e, M_b and M_c are first reordered based on an affinity based algorithm. The reordered matrix is then split into k parts by using a greedy binary partition algorithm. Refer to the technical report [3] for more details.

5.2 Dominance based method

This method is applied to points when the Dominance distance measure, as discussed in section 4.6, is used. A threshold of similarity is provided as T.

Algorithm

- The algorithm takes as input parameters the dataset file and the percentage similarity measure T. The number of clusters are not specified. Increasing the value of T increases the number of clusters and vice-versa.
- A point is grouped with another point when $S_d \geq T$. A random point is chosen at first. The other points are compared with the points already clustered.
- In this manner, clusters are formed with each cluster having points which are at least T percent similar to each other. When a point can go to two or more groups, it is put in the cluster where it has the maximum value of S_d.

However, the algorithm can be modified to generate a particular number of clusters. Based on the current number of clusters formed, the appropriate value of T is found using binary search till the required number of clusters are generated. But this also increases the complexity of the algorithm which originally is independent of the number of clusters needed.

6 Results

The results (% accuracy) as seen in table 1 are analysed on the datasets generated by Syndeca. We generated ten datasets using the Syndeca data generator with the number of points (N) varying between 15-40, the number of dimensions (d) varying between 2-6, the number of clusters (k) varying between 2-7. We ran the matrix partitioning algorithm using Euclidean, Bit-Vector and Comparative distance measures. k-means algorithm was run with the Euclidean and Comparative distance measures on the original dataset and Euclidean on the Huffman encoded dataset. We compared the accuracy of the above algorithms with that of the dominance based method which uses the Dominance measure. k-means algorithm using the Euclidean measure and the dominance based method exhibit high accuracy in almost all the datasets.

We compared the results obtained by Dominance based method along with those of k-means and DBSCAN. We manually classified the 140 players in the cricket dataset into four categories - batsmen, bowlers, wicket keepers and all rounders. The values of all attributes are normalised to fall into the range of $0 - 1$. Given a grouping of players, we compute the goodness of the solution by quantifying the difference of the features of all the pairs of players who belong to the same category (batsmen or bowlers etc.) but are put into two different groups. We use different sets of attributes for different categories - for batsmen, we consider total matches played and total runs made; for bowlers, total matches played ,total wickets taken and total runs given; for wicket keepers, total matches played and total stumpings made and for all rounders, the union of the attribute sets of batsmen and bowlers is considered. The summation of the absolute difference of their respective attributes is computed across all the pairs of players as defined before, and the sum value is then divided by the total number of such pairs. Thus, the smaller the value, the better the solution that is obtained. Table 2 shows that the solutions obtained by the Dominance based method are better compared to those by k-means and DBSCAN. Refer to technical report [3] for details.

7 Interpreting Results

We make the following observations:

- The Bit-Vector method, used to get the distance between two points, gives unnecessary advantage to some attributes over others. This is because the

DataSet			Matrix Partitioning			K-Means			Dominance based Method
			Euclidean	Bit-Vector	Comparative	Euclidean		Comparative	
N	k	d				Standard	Huffman		
15	2	6	73.33	53.33	53.33	100	73.33	60	100
17	4	6	41.1	41.1	47	76.47	76.47	64.7	81.25
22	2	6	77.2	63.63	54.5	100	90.90	50	100
25	3	6	44	44	40	100	92	48	100
26	4	6	46.1	42.3	46.1	80.76	80.76	50	96.15
30	2	6	83.33	50	73.33	100	60	70	76.66
31	3	6	45.1	41.9	77.4	100	61.2	51.6	100
32	4	6	34.3	31.25	34.3	100	59.37	34.7	93.75
36	7	5	30.55	27.77	25	100	50	41.66	100
26	6	12	30.76	38.46	26.92	84.61	50	57.69	100

Table 1: Results analysed on Syndeca datasets

No. of Clusters	Dominance based Method		K-Means	DBSCAN			
	similarity threshold (%)	result		eps	minpts	noise pts.	result
9	67	0.266	0.278	2000	1	0	0.431
13	75	0.257	0.268	1500	1	22	0.425
23	85	0.245	0.249	1000	1	43	0.330

Table 2: Results analysed on Cricket dataset

integer value depends on the order of dimensions in the d-bit vector for d dimensions.

- The Comparative Clustering Distance measure does not perform as expected. This could be because the measure does not assign any weight to any dimension. Therefore, we modified the distance measure to consider weights for the dimensions. By employing a trial and error method, when appropriate weights are assigned to the dimensions, the accuracy of clustering increases. However, such a modified measure may not be useful because assigning the correct weights to the dimensions is a non-trivial problem.
- The 2D modified vertical partitioning algorithm has huge complexity and is time inefficient for large datasets. The majority of the clusters obtained by k-way splitting are singular. This could be due to the greedy heuristic employed.
- We observe that the Euclidean distance values computed using Huffman encoded integers are significantly different from those computed using actual attribute values. The reason is that the frequency is shown by the number of bits in the Huffman code and not its integer conversion. Two Huffman encodes, which represent two different frequencies, can have the same integer representation. For example, 000010 represents low frequency and 10 represents high frequency, but both of them have the same integer representation. Hence using the measure of Euclidean distance for Huffman codes is not a good idea.
- As seen in table 2, Dominance based method is less sensitive to parameters than DBSCAN. As the average error is minimum, the results are semantically better than those by k-means and DBSCAN.

8 Conclusion

In this work, we analysed various clustering algorithms when different distance measures are employed. We ran our experiments on both synthetic and a real life dataset of cricket. We made several interpretations about the algorithms and the distance measures based on the results. Future work involves running the experiments on larger and different kinds of datasets and extending our study to other distance measures and other clustering algorithms.

References

[1] John W. Ratcliff and David E. Metzener, *Pattern Matching: The Gestalt Approach*, 1998, p. 46.

[2] Martin Ester Hans-Peter Kriegel Jrg Sander and Xiaowei Xu, *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*, 1996, pp. 226–231.

[3] Ankita Vimal Satyanarayana R Valluri and Kamalakar Karlapalem, *An Experiment with Distance Measures for Clustering*, 2008, Technical Report:IIIT/TR/2008/132.

[4] J. R. Vennam and S. Vadapalli, *Syndeca: A tool to generate synthetic datasets for evaluation of clustering algorithms*, 2005, pp. 27–36.

[5] S. Navathe S. Ceri G. Wiederhold and J. Dou, *Vertical Partitioning Algorithms for Database Design*, 1984, pp. 680–710.