IEEE Int. Conf. on Data Mining (ICDM'03), Melbourne, FL, USA, November 2003

Tree-structured Partitioning Based on Splitting Histograms of Distances

Longin Jan Latecki
Computer and Inf. Sciences Dept.
Temple University, Philadelphia, PA 19122
latecki@temple.edu

Rajagopal Venugopal
Computer and Inf. Sciences Dept.
Temple University, Philadelphia, PA 19122
vrajagop@temple.edu

Marc Sobel
Dept. of Statistics
Temple University, Philadelphia, PA 19122
sobel@sbm.temple.edu

Steve Horvath
Dept. of Human Genetics and Biostatistics
Univ. of California, Los Angeles, CA 90095-1772
SHorvath@mednet.ucla.edu

Abstract

We propose a novel clustering algorithm that is similar in spirit to classification trees. The data is recursively split using a criterion that applies a discrete curve evolution method to the histogram of distances. The algorithm can be depicted through tree diagrams with triple splits. Leaf nodes represent either clusters or sets of observations that can not yet be clearly assigned to a cluster. After constructing the tree, unclassified data points are mapped to their closest clusters. The algorithm has several advantages. First, it deals effectively with observations that can not be unambiguously assigned to a cluster by allowing a "margin of error". Second, it automatically determines the number of clusters; apart from the margin of error the user only needs to specify the minimal cluster size but not the number of clusters. Third, it is linear with respect to the number of data points and thus suitable for very large data sets. Experiments involving both simulated and real data from different domains show that the proposed method is effective and efficient.

1 Introduction and Related Work

Clustering is a division of data into groups of similar objects. Each group (or cluster) consists of similar objects; objects in different clusters are dissimilar. Here we discuss clustering under the assumption that clusters are connected regions with a relatively high density of points separated from other clusters by sparse regions [2]. Common clustering methods include: k-means clustering, k-medoid also known as partitioning around medoids (PAM) clustering [7], self-organizing maps, hierarchical clustering, and mixture model clustering. For a survey of different clustering algorithms see for example [12].

We briefly review data partitioning algorithms. These algorithms divide data into several subsets (clusters) based on optimizing an objective function that often is a function of pairwise distances, e.g. it may measure inter- or intra-cluster relations. Because checking all possible subset systems is computationally infeasible, greedy heuristics are used to iteratively optimize the objective function. This results in different relocation schemes that iteratively re-assign points between the k clusters. In iterative schemes pairwise computations may quickly become computationally too expensive. Using unique cluster representatives (centroids) resolves the problem: now computation becomes linear in the number of objects. Depending on how centroids are constructed, iterative optimization partitioning algorithms are subdivided into k-medoid (PAM) and k-means methods. A medoid is the data point that minimizes the sum of its distances to all other data points. Representation by medoids has the advantage that it presents no limitations on the attribute type and only requires the specification of a distance measure. When the features are quantitative one may also represent a cluster by the mean of its points. In k-medoid (PAM) clustering, one minimizes the median of the distances between cluster centers and the constituent elements of their associated clusters. Most partitioning algorithms do not have built-in margins of error allowing for the possibility of uncertain cluster assignments at various stages of the algorithm. For example, PAM clustering must assign an observation to a particular cluster, irrespective of its distances from the different cluster medoids. Here we propose a partitioning algorithm that succeeds in building in a margin of error by allowing for 'unclassified' nodes.

Liu et al. [11] framed clustering as a supervised learning problem that uses decision trees to distinguish observed observations from synthetic observations, which are drawn from the null distribution of no cluster structure. We pursue a different strategy which is reminiscent of using classification trees but it does not involve synthetic observations or class labels. The processing flow of our algorithm is similar to the flow of classification tree algorithms [1]. At each level of the tree, the data is partitioned into subsets (also called nodes); splits (also called branches) describe how 'parent' nodes are partitioned into 'child' nodes. The node without a parent is called the root node; nodes without child nodes are called leaf nodes or terminal nodes [17]. Classification trees recursively split the data by selecting an optimal feature and corresponding cut-off value. Tree construction involves starting with a split at the root node and continuing the splits of resulting child nodes until splitting stops; the child nodes which have not been split at the

end of this process become terminal or leaf nodes. Commonly used node splitting criterion are the Gini index [1] or the information gain criterion [13]. These splitting criteria partition the data by hyperplanes that are perpendicular to the coordinate axes in the feature space. We propose a different splitting criterion which does not correspond to perpendicular hyperplanes. Instead of splitting based on a single feature, our splitting criterion considers all features simultaneously by taking as input distances. Thus we propose a novel heuristic for relocating points which is based on applying a discrete curve evolution method to the histogram of distances between cluster members and centroids. Even if the histogram of distances contains more than two modes the discrete curve evolution robustly finds a cut-off point that divides data into two parts. The cut-off point is found based on the shape of the histogram function as 'the most significant local minimum' of the histogram function. The discrete curve evolution has been successfully applied in Computer Vision to obtain shape descriptors [10] and to characterize video trajectories [9]. Note that alternative statistical methods such as a mixture model involving two Gaussians may fail to divide data into two meaningful parts when there are more than 2 modes.

2. Tree-structured Partitioning

The proposed clustering algorithm consists of two major steps:

1. **Recursive Splitting Step:** A discrete curve evolution method (see below) is applied to histograms of distances to assign observations to two or one temporary cluster. In case of two temporary clusters, we use a Voronoi splitting with a margin to obtain two clusters and a margin data set of unclassified points. If we have only one temporary cluster, the data is not split, and this cluster becomes a leaf node.

2. **Remapping Step:** After the tree is built, ambiguous observations are mapped to the closest cluster centroids. This is similar to clustering by k-medoids algorithms such as PAM (Partitioning Around Medoids) [7].

We will now present our tree-structured partitioning algorithm. Section 2.2 describes the new node splitting criterion. Finally, Section 2.3 describes how unclassified data points are remapped. In binary classification trees a parent node gets split into two child nodes. Our algorithm follows the same pattern but splits a parent node into three child nodes. The reason for this modification is to deal with ambiguous observations (outliers), which cannot yet be clearly assigned to a cluster. Let us be more specific. The parent node is split into left, right and unclassified (uc) child nodes. At least two of the three child nodes must be present or the parent node will not be split. The unclassified node is a terminal node, which contains the ambiguous data points. If the left and right child nodes are terminal nodes, they contain all the data points belonging to a particular cluster.

Each node (except the root node) in the resulting tree structure has two attributes: (a) the membership information of the data points and (b) the distances of all the data points in that node from the representative of the parent node. For the root node the centroid of the data is selected as the representative. For each other node N, we select the representative point with respect to its parent node representative pr as follows. We first compute the mean of the distances of data points in N from pr. The representative data point for N is the data point in N whose distance to pr is closest to this mean. Ties are broken by random sampling. For the version of our algorithm that takes a distance matrix as input, we compute the medoid instead of the mean. Recall that the medoid is defined to be the data point that minimizes the sum of its distances to all other data points.

2.1. Algorithmic Details

The input to our algorithm is either a set of attributes or a distance matrix that contains all pairwise distances between the data points. Let $A = \{a_i\}$, for $i = 1, .., n$, be a set of data points in an m dimensional feature space, i.e., each feature is denoted by index a_{ij} for $j = 1, ..., m$. Let $a \in A$ be one of the data points. *The distance projection* function d_a is defined as $d_a(x) = d(a, x)$, which denotes the distance between x and $a \in A$. Thus a distance projection function represents the distance of the data points from a singled out data point.

Algorithm $Createtree(A)$

1) Normalize the data A (optional)

2) Compute the centroid (mean or medoid) $\bar{A}$ of the data A.

3) Define the distance projection with respect to $\bar{A}$: $d(a_i) = d(\bar{A}, a_i)$ (for $i = 1, ..., n$).

5) Create a root node (r), with attributes $\bar{A}$ and d.

6) Apply the function $splitnode(r)$ (see Section 2.2) to non-terminal nodes.

7) Let N be a non-terminal node computed in (6) that is either (lc) or (rc) type node. Let $pr(N)$ be the representative point of the parent node of N:

a) Find the mean m of distances $d(pr(N), x)$ of the data points $x \in N$

b) Select a data point $a_i \in N$ for which $|d(pr(N), a_i) - m|$ is the smallest as the *representative* of N and label it $r(N)$.

c) Define the distance projection function $dnew$ of all the data points in N with respect to $r(N)$, i.e., $dnew() = d_{r(N)}()$.

d) Replace d with $dnew$ for that node.

8) Go back to line (6) substituting non-terminal nodes for the input parameter

9) Compute the centroids for all the clusters

10) Data points in the terminal node that contains the unclassified (ambiguous) observations get assigned to the cluster that corresponds to the closest centroid.

In lines 1-5 a root node is created. For each of the non terminal nodes returned by the split function (line 6), the distance values are updated in lines 7a),b),c),d). Each of the updated non terminal nodes are split in step 8. Lines 6,7,8 are repeated until all leaf nodes are found. In lines 9,10 data points labelled unclassified are mapped to the cluster closest to them.

2.2. Node Splitting by Discrete Curve Evolution

Let $N = \{x_i\}$, for $i = 1, .., z$ be a set of data points in a node (N) and d_i , for $i = 1, .., z$ be a set of distances $d_i = d(pr(N), x_i)$, where $pr(N)$ is the representative point of the

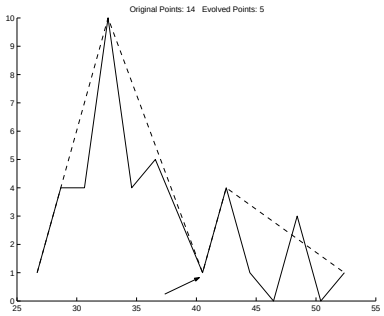

Figure 1: The initial cut-off point obtained by the discrete curve evolution for NIC41 data described in Section 3.

parent node of N. The algorithm for splitting a node in the tree is outlined below. It computes the function $splitnode(N)$.

There are two user defined inputs mnp and $percentage$. The input mnp defines the minimum number of data points the user wants to see in a cluster and acts as criterion to stop splitting. The parameter $percentage$ defines the area in between the two representative points as "danger zone". All data points falling in this area are considered to be outliers and become an unclassified terminal node.

The user does not need to specify the number of clusters since the algorithm determines the number of clusters on the basis of mnp and $percentage$. We have found that $percentage$ has minor effect on the resulting number of clusters. In our experiments we set $percentage$ to vary from 0.01 (1%) to 0.1 (10%) of the data in a given node. A good default setting is 0.1. But it is obvious that mnp will in general have a major effect on the resulting number of clusters. For example setting $mnp = 500$ will result in clusters with a minimum cluster size of 500.

The basic idea of splitting a node consists of three steps

1. Find an initial cut-off point: We first plot the histogram of distances $\{d_i\}$. We use discrete curve evolution (DCE) [10, 9] to find a temporary cut point. We treat the graph of the distance histogram as a polygonal curve P. DCE ellows us to recursively delete the vertices of P until only 5 vertices remain that best interpolate the shape of P Figure 1 shows a sample histogram (frequency plot) and the temporary cut point (marked with the arrow). The histogram plot of the distances of all the data points in a node from the representative is shown as a solid line and the evolved curve resulting from using the discrete curve evolution is shown as a dashed line. The first local minimum point in the evolved curve is chosen as the cut point.

2. Adjust the data partition: After splitting the data temporarily into two child nodes (*child1* and *child2*) the final split is found based on three criteria.

I) If the number of data points in one of the child nodes *child1* or *child2* is zero, the parent node N is labelled as a cluster and no further splitting is performed.

II) If criterion **I)** is not satisfied, then two representative points c_1, c_2 of the temporarily split data are computed. c_1 is a closest data point in *child1* to the mean of the distances $d(pr(N), a_i)$ of points a_i in *child1* to the representative of the parent node $pr(N)$. Similarly we determine c_2.

III) We declare a point x_i in N ambiguous, if it is is within the margin region

$$|d(x_i, c_1) - d(x_i, c_2)| \leq percentage * d(c_1, c_2).$$

If it is not ambiguous and closer to c_1 than to c_2, then it is mapped to cluster *newchild1* else to *newchild2*.

3. Tests on minimal number of points:

Finally we eliminate clusters that have less points than mnp.

2.3. Remapping

This is the final stage of our algorithm. We arrive here when all nodes are terminal leaf nodes. The centroids of all terminal leaf nodes (lc) and (rc) are computed and the data points from the leaf nodes labeled unclassified (uc) are mapped to their closest centroids in the feature space.

3 Comparison to PAM Clustering

Here we apply our partitioning method to 3 different data sets. On all of these data sets the proposed method performs better than PAM clustering. For our analysis we used the PAM function in the cluster library of the freely available software R (url: http://cran.r-project.org/). PAM takes as input a dissimilarity matrix between the observations and requires that the user specify the number of clusters k to be generated. In all of our analysis we used the Euclidean distance metric between the observations as dissimilarity matrix.

Our first test data set will be referred to as *ChallengeIII*. This simulated data set contains 2 clusters. Each cluster contains 50 observations. There are 4 features but only the first 2 features contain a signal while features 3 and 4 are random draws from a standard normal distribution. For cluster 1 and cluster 2 observations, the first 2 features are random draws from a beta distribution with shape parameters (2,5) and (5,2), respectively. When choosing input parameters $mnp = 30$ and $percentage = 0.05$, our proposed method misclassifies only one point. The adjusted Rand index [14] between assigned clusters and ground truth is .96 which reflects the very high agreement between cluster label and ground truth.

When using the same distance matrix in $k = 2$ medoid (PAM) clustering we find that 3 observations are misclassified which lowers the adjusted Rand index to .88. Thus our proposed method outperforms PAM on this data set.

The second data set is a subset of the NCI60 gene expression data from the National Cancer Institute (see [15] and http://genome-www.stanford.edu/nci60). We considered only 6 kinds of cancers and used the gene expression values of the 1375 genes that were available in the data set referred to as "T matrix". Our data set which could be referred to as NCI41 consisted of 41 cancer cell lines (a classical multi-dimensional scaling (MDS) plot is shown in Figure 2): 6 central nervous system (denoted by CN), 7 colon (CO), 6 leukemia (LE), 8 melanoma (M), 6 ovarian (O), and 8 renal (R) cancer cell lines.

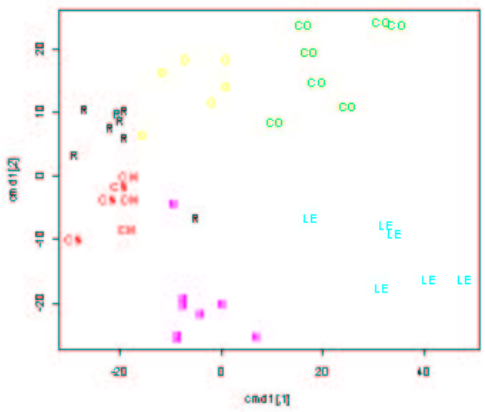

Figure 2: Classical MDS plot of the 41 samples in NIC41.

	CN	CO	LE	M	O	R
1	5	0	0	0	0	0
2	0	6	0	0	3	0
3	0	0	6	0	0	0
4	0	0	0	7	0	0
5	0	1	0	0	4	0
6	1	0	0	1	2	8

Figure 3: Our algorithm disagrees with the ground truth information for NIC41 in only five points.

We normalized the features so that they have mean zero and variance one. Our algorithm with the parameters $mnp = 3$ and $percentage = 0.01$ is very close to the ground truth: it misclassified 5 points resulting in an adjusted Rand index of .82; see Figure 3, where the rows correspond to the obtained clusters and the columns show the ground truth.

We also applied our partitioning method to the Euclidean distance matrix of the normalized feature space. With the parameters $mnp = 1$ and $percentage = 0.05$ the clustering method misclassified 9 points which resulted in an adjusted Rand index of .58. When applying PAM clustering (k=6) to the same distance measure we found that PAM misclassifies 12 observations, which lowered the adjusted Rand index to .47. Thus our proposed method outperforms PAM on this data set.

The third data set is the classical data set called Irises. It is composed of 150 observations with 4 feature measurements. As ground truth we have three clusters, each with 50 points. There is one clearly separated cluster A and two clusters B and C that are hard to distinguish. We applied our algorithm to the distance matrix of the data computed without feature normalization. It misclassified 10 observations from clusters B and C with a corresponding adjusted Rand index of .82. The results obtained with the parameters $mnp = 25$ and $percentage = 0.01$ are summarized in the table shown in Fig. 4, where the rows correspond to the obtained clusters and the columns show the ground truth.

When using the same distance measure in PAM $k = 3$ clustering, we find 16 misclassifications and a the adjusted Rand index was lowered to .73. Again, our proposed method outperforms PAM. Matlab source code of our algorithm can be found at `www.cis.temple.edu/~latecki/Clustering`.

	1	2	3
1	50	0	0
2	0	49	9
3	0	1	41

Figure 4: Our algorithm disagrees with the ground truth information for Irises data in 10 points. Adjusted Rand index is .82.

References

[1] Breiman, Leo, Friedman, J.H., Olshen, R.A., and Stone, C.J. *Classification and Regression Tree's.* Wadsworth, 1984.

[2] B.S.Everitt Cluster analysis *Heinemann, London* 1974.

[3] C. Hennig and L. J. Latecki. The choice of vantage objects for image retrieval *Pattern Recognition* 36, pp. 2187-2196, 2003.

[4] Hand, D.J. *Discrimination and Classification.* Wiley, 1981.

[5] Hand, D.J. *Construction and Assessment of Classification Rules.* Wiley, 1997.

[6] Hartigan, J. *Clustering Algorithms.* Wiley, 1975.

[7] L. Kaufman and P.J. Rousseeu. *Finding Groups in Data.* Wiley, New York, 1989.

[8] Kohonen, T. *Self-Organizing Maps.* Springer, Berlin, 1995.

[9] L. J. Latecki and D. de Wildt. Automatic Recognition of Unpredictable Events in Videos *Proc. ICPR*, Vol. 2, 2002.

[10] L. J. Latecki and R. Lakamper Shape Similarity Measure Based on Correspondence of Visual Parts *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)* 22, pp. 1185-1190, 2000.

[11] Bing Liu, Yiyuan Xia and Philip S. Yu. CLTree - Clustering through decision tree construction *IBM Research Report* RC21695, 20/3/2000.

[12] B. Pavel. Survey of clustering data mining techniques. *Accrue Software Inc* 2002.

[13] J. R. Quinlan. C4.5: program for machine learning *Morgan Kaufmann* 1992.

[14] W.M. Rand. Objective criteria for the evaluation of clustering methods. *J. of the American Statistical As.* 66, 846-850, 1971.

[15] D. T. Ross, U. Scherf, M. B. Eisen, C. M. Perou, P. Spellman, V. Iyer, S.S. Jeffrey, M. V de Rijn, M. Waltham, A. Pergamenschikov. Systematic variation in gene expression patterns in human cancer cell lines. *Nat Genet* 24, 227–234, 2000.

[16] Classification Trees *Electronic Statistic Textbook, Statsoft Inc. "http://www.statsoftinc.com/textbook/ stathome.html"*

[17] Classification Trees: Slide notes *"http://medg.lcs.mit.edu/hamish/6872LECT/sld001.htm"*