DATA CLUSTERING USING THE NATURAL BÉZIER FUNCTIONS

P. Venkataraman
Rochester Institute of Technology
Department of Mechanical Engineering
76, Lomb Memorial Drive
Rochester, NY 14623-0887, USA
Tel: 585 475 6975; Fax: 585 475 7710
pneveme@rit.edu

ABSTRACT
An unorthodox and effective non-iterative procedure for spherical clusters is demonstrated in this paper. It uses natural Bézier functions to determine initial cluster locations using the content of the data. The natural Bernstein-Bézier functions are very robust in representing data through continuous functions in the application of functional data analysis. This paper demonstrates that they are equally robust at resolving data clusters in classification problems. The original data is scaled and segmented. A natural Bézier function is fitted for each segment and the initial clusters are centered at the function extremums that are distinctly located. A self-selection process based on least distance is used to assign the data to these initial cluster centers. A minimum membership count is imposed and nearby clusters are combined to reduce these initial cluster centers based on visual clues. Centroid recalculation and data reassignment can be used for centroid convergence. The method is demonstrated for two dimensional spherical clusters. This approach requires no iteration. Four examples from clustering benchmark datasets are used to showcase the method. The data include different numbers of clusters, different data density for the clusters, as well as different levels of overlap. This method is new and different from other data clustering methods available in the literature. It is better than the standard k-means clustering method since it does not require information on the number of clusters or cluster membership count. The method is non-iterative and does not require random initialization or distance optimization.

INTRODUCTION
Machine learning and artificial intelligence have significant influence in decision making in every discipline today. Machine learning essentially classifies data into patterns, groups, or labels. Machine learning problems are broadly categorized into supervised and unsupervised learning. Supervised learning involves training the algorithm, testing it, and validating it using portions of the same data. Unsupervised learning is simpler and used for finding patterns in data. Data clustering is a primary example of unsupervised learning [1].

Data clustering is the task of assigning data to groups that have similar properties and differ from other groups. This is a common technique for data mining. Clusters themselves are not uniquely defined. A cluster could be a clump of data with small distances amongst group members, it could involve members that define a dense area of data space, or a collection of data that have a particular statistical distribution [2]. This makes cluster analysis or cluster algorithms different based on the general task they are required to solve as well as the nature of the data. These algorithms differ on defining clusters but also how to efficiently find them using a particular model.

Conventional cluster algorithms classify data by using some properties of the data. Data clusters are not all the same and the idea of a cluster cannot be accurately defined. Different properties require different models and
different algorithms to identify different types of clusters. There are algorithms for connectivity models, centroid models, distribution models, density models, subspace models, group models, graph based models, neural models [3]. In addition there are partitioning clusters with and without outliers, hierarchical clustering, and overlapping clusters. Partitional clustering uses distance or density criteria and can be based on defining a global distance minimization problem. This paper deals with partitional clustering.

Clustering is often part of a larger solution to problems in pattern recognition, image analysis, and other fields in science and engineering [4]. Clustering can be postulated as a multi-objective optimization problem. The important answers in a clustering problem is the number of clusters and where they are located. Most algorithms require some parameters to start the search and their improved versions are associated with reducing their dependence on these parameters. Since the solution clusters are not uniquely defined most methods stop with an acceptable solution. From an optimization perspective this is an approximate solution instead of the global solution. In general, each clustering algorithm in the same category can produce different number of clusters with different membership. Different clustering results may also be possible using the same clustering algorithm but using different parameter values. Robustness of clustering in the presence of noise is an important area of research and is not addressed in this paper as the primary motivation is the presentation of a new procedure.

One of the earliest well known clustering algorithm is the k-means algorithm. It is part of centroid based clustering models and the original algorithm is also known as the Lloyd's algorithm [5]. It discovers approximate clusters. It requires information on the number of clusters and is sensitive to initial parameter choices. Clusters are expected to be of similar size and shape. Several variations of the k-means algorithm have been developed to overcome its limitation. They include k-means++ [6] where initial centers are less randomly chosen as in this paper. K-median [7], k-medoids, and fuzzy c-means [2] are also variations on centroid based clustering. It is believed that k-means cannot represent density based clusters nor can handle non convex clusters.

This paper presents a unique but novel approach to two-dimensional clustering. It does not require information on the number of clusters or its membership count. It seeks to use the content of the data to initialize the cluster centers. If all data have the same weight then a large group of data located close by would suggest a maximum for a function that represents the data distribution, particularly if the function was continuous. This approach works well in functional data analysis (FDA) where the data are modeled to be continuous and smooth [8, 9] unlike the data space in clustering where continuity is an obscure expectation. For this paper, the representations of bunched discrete data by continuous function was also required to facilitate the recognition of the function maximums or extremums. One category of functions that has served the author well in this task is the Bézier functions [10]. Recently a simpler version of these functions, called the natural Bézier function, was shown to be very effective in representing the data and its derivative smoothly [11]. Using a continuous function to represent data in clustering is definitely an unusual and original approach.

The procedure is detailed after the introduction of the natural Bézier function below. In the following sections the natural Bézier function is first defined. It is followed by the description of the procedure developed for the paper along with the parameters involved. All computations are done in MATLAB using array processing. Four examples from a cluster database are explored using the same technique. Finally the paper is concluded detailing the advantages of this approach and identifying future work to follow.

**NATURAL BERNSTEIN-BÉZIER FUNCTIONS**

A regular Bézier curve is a continuous, smooth, bounded polynomial that can simulate any continuous function over a reasonable range. In this paper the term regular Bézier curve is the same as the standard curve defined in the literature [12]. This curve is formally defined by invoking the Bernstein polynomials. A regular Bézier curve is a parametric curve that is completely determined by a set of vertices or control points.

\[
[x(t) \ y(t)] = \sum_{i=0}^{n} B_{i} J_{n,i}(t) , \ 0 \leq t \leq 1
\]

\[
J_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i}
\]

Here \(J_{n,i}(t)\) represents the Bernstein basis in the parameter \(t\). \(n\) is the order or degree of the polynomials. \(B_i's\) are the vertices or control points of the polygon (a pair of values \(a_i\) and \(b_i\)) that determine the curve in two-dimensional space. Gordon et. al. [13] establish several properties of the curve and their definition using matrices. Many of these properties are useful in functional data analysis. These properties also make the curve suitable as solutions to differential equations. The Bézier function is an excellent candidate for mathematical problems that require a continuous solution over the whole domain. These Bézier curves are special cases of B-splines.

**The Natural Bernstein-Bézier Function**

The natural Bernstein-Bézier function, defined by the author, is the regular Bézier function where the vertices of
the independent variable are located at equidistant values between their known limits \([x_{\text{min}}, x_{\text{max}}]\). They will be referred as the natural Bézier functions in this paper. For the natural Bézier function the vertex/control point location of the independent variable \(x\), that is \(a_0, a_1, a_6\) is known explicitly. This constraint causes both \(t\) and \(x\) to vary linearly between their limits. This makes \(t\) is a linear function of \(x\)

\[
\Delta x = (x_{\text{max}} - x_{\text{min}}); \quad x_{\text{min}} \leq x \leq x_{\text{max}}
\]

\[
t = \frac{x - x_{\text{min}}}{\Delta x} \quad \text{OR} \quad t(x) = c x + d \quad (2)
\]

This brings the problem from the parameter space to the function space

\[
y(t) \Rightarrow y(x) \quad (3)
\]

The dependent variable \(y\) can be expressed as a function of \(x\) instead of \(t\) in the regular Bézier function.

\[
y(x) = \sum_{i=0}^{n} B_i J_{n,i}(t(x)) \quad , \quad x_{\text{min}} \leq x \leq x_{\text{max}} \quad (4)
\]

Here, \(B = [b_0, b_1, b_2, ... , b_n]\) is a simple vector array.

This dependence can be further simplified to

\[
y(x) = y(b_0, b_1, ..., b_n); \quad x_{\text{min}} \leq x \leq x_{\text{max}} \quad (5)
\]

In addition to the direct dependence on the variable \(x\), the natural Bézier polynomials in the \(x\) domain have the same features as the regular Bernstein polynomials in the \(t\) domain. In fact all of the properties in the parametric space are available in the two-dimensional function space. Figure 1 compares the polynomials in the parameter space to the polynomials in the two-dimensional space for a fourth order Bézier function. We use this correspondence to postulate the behavior of the function is the same in the parametric space and the function space.

In summary, the natural Bézier function is now defined completely by the \(y\)-control point array \(B = [b_0, b_1, b_2, ... , b_n]\), accompanied by the limits on the independent variable \([x_{\text{min}}, x_{\text{max}}]\) and the order of the function \((n)\). The location of the \(x\)-control points is automatically defined. While this is a different Bézier function it is also a simpler one. This function is less flexible than the original Bézier function, which can be made up by increasing the order of the function.

**NATURAL CLUSTERING**

The term natural clustering is coined for the first time by the author to provide a reference to the unique procedure documented here using the natural Bézier functions. This work considers two-dimensional data and spherical clusters. It does not require prior knowledge on the number of clusters or the cluster membership count. The process is implemented as a series of steps. The significant effort takes place initially where a continuous natural Bézier function is fit to the discrete data using least squared error between the function and the data. This is solved through a matrix equation and not as an iterative optimization problem [14]. The extremum of this function through the first derivative is used to discover the initial cluster centers. The remaining exercise involves modifying the clusters by adjusting the cluster centroids until the sequence is completed. The clusters themselves are recognized using a minimum Euclidean distance measure. The data select their nearest cluster centers.

The procedure relies on the following parameters:
- Number of data segments/levels (integer - \(k\))
- Order of the fitted natural Bézier function (integer - \(m\))
- Minimum number of data points for a cluster (integer - \(n\))
- Minimum distance between clusters (real - \(d\))
The parameters can be selected based on the properties of the original data, particularly if a visual plot of the original data is available. For example the first parameter for the examples included here varied between 1 and 7. The second parameter, the order of the Bézier function, was not very critical. A choice of 10 or higher produces sufficient flexibility to accommodate several curvature changes of the function. The minimum number of data points in a cluster, and the minimum distance between clusters can be obtained by visual cues from the plot of the original data or its properties. Many of the integer parameters can be selected by varying them between a range of values and choosing the best using some internal or external criteria. It is possible to evolve this procedure to be strictly machine implemented without human interaction. For this initial investigation some visual cues are used to execute the procedure. In this paper a minimum Euclidean distance is used to assign the data to the cluster centers.

Proximity information can be derived from visual clues or be based on minimum distance between the initial clusters.

**Step 5.** Determine the centroid of the current clusters and reassign data to clusters based on these centroids. These are the final data clusters. Step 5 can be extended to converge the centroids.

Data is clustered at the end of these five steps. No iterations are necessary. No knowledge of the number of clusters are required. There are no random initializations. There is no optimization problem. Data fitting in step 2 is non-iterative. Where possible the computations are done as array operations for faster execution in MATLAB [15].

**EXAMPLES**

Four examples are discussed in this section. They vary with the number of clusters, different density clusters and clusters with overlap. The original data is obtained from the web site “Clustering Benchmark Datasets” [16] maintained by the school of computing at the University of Eastern Finland.

**Example 1**

The actual data set is from Fränti [17] where an iterative method is introduced to solve this problem. The data is synthetic. The scaled data set shown in Figure 3a comprises of 5000 data points in 15 Gaussian clusters.

**Step 1.** Normalize the data and estimate the parameters based on the initial plot of data. Figure 2 is the initial data for Example 1. The first parameter is to select the number of levels of the data along the ordinate. Figure 2 indicates $k = 4$ equal segments or levels based on the initial plot of the data.

**Step 2.** In each segment/level fit the data with a natural Bézier function. This is artificially illustrated in Figure 2 for the first level. Once the function is available the distinct extremums are obtained. These are the initial cluster centers. Assign the data to these initial clusters.

**Step 3.** Reduce the number of clusters using the minimum population count. This can be based on the total number of data points and the initial number of clusters, or visual clues.

**Step 4.** Merge clusters that are in close proximity and reassign the data to the remaining cluster centers.

**Step 5.** Determine the centroid of the current clusters and reassign data to clusters based on these centroids. These are the final data clusters. Step 5 can be extended to converge the centroids.

Data is clustered at the end of these five steps. No iterations are necessary. No knowledge of the number of clusters are required. There are no random initializations. There is no optimization problem. Data fitting in step 2 is non-iterative. Where possible the computations are done as array operations for faster execution in MATLAB [15].

**EXAMPLES**

Four examples are discussed in this section. They vary with the number of clusters, different density clusters and clusters with overlap. The original data is obtained from the web site “Clustering Benchmark Datasets” [16] maintained by the school of computing at the University of Eastern Finland.

**Example 1**

The actual data set is from Fränti [17] where an iterative method is introduced to solve this problem. The data is synthetic. The scaled data set shown in Figure 3a comprises of 5000 data points in 15 Gaussian clusters.
The second parameter is order of the Bézier functions. Since this is an integer, once again a range of values can be considered and best chosen through a least squared error in the function fit. In numerical experiments the method always picked the largest permitted order. It is also possible to use the same distance measure to choose the best power. In this example we pick \( m = 10 \). This choice is not very critical and generally 10, 15 or 20 worked well with the examples in this paper. While tenth order polynomial is normally considered excessive, these orders are pretty stable in obtaining the extremums. It takes high orders to allow for flexibility in the curvature of the function to locate the extremums, particularly if there a large number of clusters.

**Parameters:** \( k = 4, m = 10, n = 110, d = 0.07 \)

**Step 2.** This is the important step of fitting a continuous natural Bézier function to the data. The set of Figures 3b – 3e illustrate the power of approximating the discrete data by a continuous natural Bézier function. This is a tenth order polynomial. It is remarkable to see that the function is able to identify the cluster centers in each segment. Shown on the figures along with the function are the distinct extremums established through the derivative of the natural Bézier functions using very fine spacing. They are the yellow circular markers. The figures indicate that many of the extremums are located close to the cluster centers justifying the idea behind this approach. Those extremums not located close to the cluster centers are sparsely populated.

Number of extremums in each level:  6 5 6 5
Total number of initial cluster centers:  22

**Figure 3b.** Example 1 - Bézier function with extremums – level 1

**Figure 3c.** Example 1 - Bézier function with extremums – level 2

**Figure 3d.** Example 1 - Bézier function with extremums – level 3

**Figure 3e.** Example 1 - Bézier function with extremums – level 4

Figure 4a is the complete collection of these initial cluster centers along with the original data. Unlike the k-means method this does not require information on the number of clusters. It is clear that these clusters must be reduced further through population count and proximity consideration. The data is initially clustered using these cluster centers.
Step 3. The next step is to remove clusters with few data points. The minimum number of data points for a cluster to exist was rounded to be 110. This is around one half the number of points per cluster based on the initial number of clusters. This reduced the number of clusters to 15. The new cluster centers are shown in Figure 4b.

Step 4. The next step is to collapse clusters which are in close proximity. This requires distance information and is the final parameter with a value of 0.07. This parameter can be based on a multiple of the minimum distance between the initial cluster centers. This did not result in further decrease in the number of clusters.

Step 5. Calculate the centroid of the current clusters and use these as the new cluster centers. Reassign data to these new cluster centers using a different color for each cluster. The result is shown in Figure 4c. The clusters are shown along with the centroid based on the latest membership.

This completes the data clustering activity. The idea is very simple. Use the content in the data itself for the calculation of the initial clusters. The clusters are separately colored and are visibly confirmed. The sum of the squared distance of all the data points from their respective cluster centroid is 10.308. This distance can be used to implement a complete machine learning approach.

Each data point is assigned to the closest cluster center. The final number of clusters using the natural clustering technique is 15 and this is the number of clusters in the original data. The number of clusters and the membership has been determined successfully by visual comparison.

Figure 4c. Example 1 - final clusters and the centroid.

Figure 4d is the same example clustered using the K-means++ algorithm from scikit-learn [18] using Python on the same laptop. The code is run through the spyder IDE.

The execution time for the natural clustering using MATLAB was 0.528 seconds while it took 1.524 seconds for the K-means++ algorithm. Natural clustering did not require the number of clusters while it was necessary for K-means++ implementation.

Example 2

Example 2 [19] has 8 Gaussian clusters with 6500 data points and is seen in Figure 5a. The clusters are not balanced as they have different densities. The dense clusters all had 2000 points each and the remaining 500 was distributed among the 5 clusters as seen in the figure. In the reference they were discussed in the context of
developing external validity measures that should not depend on the nature of the data or its distribution. Figure 5b is obtained from the same reference (Figure 23) indicating the weak performance of the k-means and the single link algorithm. The natural clustering technique of this paper identifies the clusters correctly.

**Step 1.** The data is clustered using the same approach used in Example 1 but with 3 levels, Bézier functions of order 10, a population cut-off of 50 points, and a proximity distance of 0.1.

**Parameters:** \( k = 3, m = 10, n = 50, d = 0.1 \)

**Step 2.** Figure 6a – 6c illustrates the natural Bézier function for the three segments and the extremums identified. The figures illustrate that the expected cluster centers are effectively captured by the high order continuous function, even extending the data region beyond the initial bounds. The second level is particularly challenging but appears to hold in spite of the function fit. It was not necessary to constrain the fitting function since these locations will drop off due to the minimum population requirements for the cluster.

Number of extremums in each level: 2 7 6
Total number of initial cluster centers: 15
Step 3. Figure 7a is the consolidation of the initial clusters and the initial data. There are 15 of them. Figure 7b are the cluster centers after removal of low population cluster centers. This reduces the count to 9.

Step 4. Figure 7c illustrates the set of clusters after those nearby have been combined. This reduces the number of clusters to 8.

Step 5. In this final step the centroid of the local clusters are identified and the data is clustered around them. This is the final result. Figure 7d indicates the final clusters and the end of the exercise. The sum of the squared distance of the data points from their respective cluster centroid is 4.247. The dense clusters each have 2000 data points. The coarse clusters have 100 points each as designed. The clusters are visually confirmed and the membership counts matches the information found in Reference 16. The method has not changed between Examples 1 and 2.

Example 3
Example 3 [17] is accessed through the dataset collection [16]. This is a synthetic data set with 15 clusters and 5000 data points. The data is more diffuse than previous examples with different degrees of overlapping. The original paper used an iterative approach to shrink the number of clusters until the desired number is reached. The algorithm used a local optimization method with a crossover method and a genetic algorithm. This paper identifies the clusters using the same non-iterative approach in the previous examples.

Step1. The normalized original data is in Figure 8a. The data is clustered using the same approach used in Example 1 but with 7 levels, Bézier functions of order 20, a population cut-off of 80 points, and a proximity distance of 0.1.

Parameters: k = 7, m = 20, n = 100, d = 0.1
Step 2. For this example only the consolidated results are presented. Figure 8b is the complete collection of initial clusters generated using the natural Bézier function in all of the seven segments.

Initial clusters/level:    7     3     6     3     6     4     2
Total number of initial cluster centers: 31

Step 3. Figure 8c is the cluster centers after removal of low population clusters. The number of clusters is reduced to 18.

Step 4. The proximity clusters are combined to yield Figure 8d. The number of clusters is reduced to 14. This is the final number of cluster centers determined by the method of this paper.

Step 5. The centroid based on the cluster distribution is determined and data are assigned to these centroids. This is the final cluster and is shown in Figure 8e. In this figure the original cluster centers (green circle markers) are also shown to illustrate the change in going from cluster centers to centroids (yellow square markers). Figure 8f is the clusters with centroids alone. It is possible to converge the centroids to the data by additional sequences of reclustering and with new centroid determination.

The number of designed clusters for this problem is 15. This is one more than the number of clusters determined by natural clustering. Figure 8g is the centroids projected on the original data. Visually it can be seen that all of the centroids are near the denser distribution of data except for the points within the black circular region represented by a single cluster instead of two. The centroid appears equidistant from two very similar clusters where the diffusion of data is pretty uniform so that the data in that region is represented by a single cluster instead of two clusters (circular markers). The method is not aware of the number of clusters it is supposed to seek and it has done a respectable job in most of the region. Since most practical clustering problems do not have a unique solution, this approximate solution may be acceptable.
Example 4
This is a synthetic data set with 5250 data points at 35 clusters with 150 data points per cluster [20]. The data was used to present a dynamic local search algorithm that is significantly faster than a brute force approach. It was based on a sophisticated evaluation function that included the number of clusters as a parameter. In this paper we continue to solve the problem using the same approach used in earlier examples. This approach has no knowledge of the number of clusters or cluster membership count.

Step 1. The normalized original data is in Figure 9. The data is clustered using 6 equal levels, Bézier functions of order 20, a population cut-off of 60 points, and a proximity distance of 0.07.

Parameters: \( k = 6, \quad m = 20, \quad n = 60, \quad d = 0.07 \)

This example is similar to Example 1 except that it has lot more clusters. Details of Step 2 are omitted.

Step 3. Figure 10a is the set of the 55 initial cluster centers for all the levels. Several cluster centers without data are clearly visible in the figure. Figure 10b is the set of clusters after elimination of low population clusters. The initial cluster center count is trimmed to 38 during this phase.

Step 4. The proximity clusters are combined to yield Figure 10c. There are now 35 clusters after application of this
process. This is the final number of clusters based on the application of the procedure.

**Step 5.** The centroid for the clusters in Figure 10c is determined. The data will now be distributed among these centroids to yield the final cluster as in Figure 10d along with a determination of the new centroid based on current membership.

In Figure 10d the clusters in the region between $0.45 \leq x \leq 0.7$ and $0.5 \leq y \leq 0.5$, affecting three clusters are not crisply defined as the remaining clusters. The total number of clusters in Reference 16 is 35 and this is the same as the number obtained in the exercise and shown in Figure 10d. The total distance of all clusters from their respective centroids is 8.2996.

This procedure can be extended another step by taking advantage of a self-adjusting process that is available as part standard k-means clustering. This is explained in Step 6.

**Step 6.** In this process there is a lag in assigning members to the cluster and the actual centroid of the cluster. By repeating these calculations it is possible to converge the members to the centroids. No new type of calculations are required. The calculations that are repeated:

(i) Establish the new centroid based on current membership.

(ii) Reassign data based on the new centroid.

(iii) Calculate the distance measure.

Figure 10e is the result at the end of five repetitions of the calculations or the application self-adjusting process. The data has been neatly clustered in the figure.

The sum of the distance of the data from the centroids progressively decreased as 8.2996, 8.1366, 7.634, 7.5495, and 7.5444. The numbers are converging.

The designed membership is 150 members per cluster. In Figure 10e there are 10 clusters with 150 members while the remaining clusters have counts between 148 and 151 with two outliers at 143 and 157.

**CONCLUSIONS AND FUTURE WORK**

Natural clustering as implemented in this paper uses natural Bézier functions to fit continuous functions to the discrete data in each level. The initial cluster centers are determined to be at the distinct extremums of these continuous functions. Since there are many inflection points for the fitted function there will be a cluster centers with few members. Some clusters may be split across levels giving rise to clusters in close proximity. The initial cluster centers are then trimmed to be distinctly located with more than a minimum membership count. This is the final number of clusters. The centroid of the members in the final clusters are calculated and the data is assigned to these centroids and the new centroid is calculated. The distance measure was the sum of the distance of the data from their nearest centroid. This completes the clustering process for Examples 1–3. For the fourth example the
lag in the centroid and the assignment of the members is used to iteratively improve clustering through only reassignment of the data.

Natural Bézier functions provide a new approach to data clustering for two dimensional spherical clusters. The method is simple and is driven by the properties of the data that have not been harnessed before. This unique method is shown to be effective using four examples with different properties for the data. The method requires four parameters which are currently based on the visual cues from the initial data. Give a data set made up of spherical clusters the procedure has the following advantages

- The numbers of clusters is not required.
- The membership distribution among the cluster is not required.
- Clusters with different data density can be recognized.
- Clusters with diffused members can be recognized.
- Large number of clusters can be determined.
- The distance measure is the sum of the shortest distance of the data from its closest center.
- There are no constraints imposed during the process except for the choice of the parameters which are suggested from the initial data.

Most of the parameters are integers and this will allow the development a machine learning version of this procedure to automate the clustering of data without visual cues. The procedure can also be improved by using different distance measures for other kinds of data.

The implementation is completely numerical and takes advantage of array processing in MATLAB. It executes Example 1 more quickly than the implementation of the k++-means in Python on the same laptop.

All data sets used are synthetic. Future investigations will include attempts to extend this idea to real data sets, non-convex clusters, shaped clusters, clusters with large data sets, and higher dimensional data sets. The robustness of this approach also needs to be explored.

REFERENCES


