A Feasible Algorithm for Locating Concave and Convex Zones of Interval Data and Its Use in Statistics-Based Clustering

Vladik Kreinovich[1], Eric J. Pauwels[2], Scott A. Ferson[3], and Lev Ginzburg[3]

[1]Department of Computer Science, University of Texas at El Paso
El Paso, TX 79968, USA, vladik@cs.utep.edu

[2]Centre for Mathematics and Computer Science CWI
Kruislaan 413, NL-1098, SJ Amsterdam, The Netherlands
eric.pauwels@cwi.nl
and
Katolieke University Leuven, P.O. Box 94079, NL-1090
GB Amsterdam, The Netherlands,
[3]Applied Biomathematics, 100 North Country Road
Setauket, NY 11733, USA, {scott,lev}@ramas.com

Abstract

Often, we need to divide n objects into clusters based on the value of a certain quantity x. For example, we can classify insects in the cotton field into groups based on their size and other geometric characteristics. Within each cluster, we usually have a unimodal distribution of x, with a probability density $\rho(x)$ that increases until a certain value x_0 and then decreases. It is therefore natural, based on $\rho(x)$, to determine a cluster as the interval between two local minima, i.e., as a union of adjacent increasing and decreasing segments. In this paper, we describe a feasible algorithm for solving this problem.

Keywords: Clustering; Statistical methods; Kolmogorov-Smirnov statistic; Convex segments of interval data

AMS Classification: 91C20, 65G20, 52A10, 62D05

1 Formulation of the Practical Problem

In each area of interest, we are studying certain classes of objects: e.g., in astronomy, we study stars, galaxies, etc.; in entomology, we study insects. To study different objects, we perform different measurements on these objects; as a result, each object is characterized by the values $x_1, \ldots, x_d$ of the measured quantities.

Usually, for each of these variables x_i, there are physical bounds that bound its possible values, and within these bounds, all values are, in principle, possible. In practice, however, there are a few sub-intervals that contain the vast majority of the objects, and values in between these sub-intervals are rare. In other words, most objects belong to *clusters*, with few objects in between.

This fact is very useful in practice: if we want to control objects from a given class, then, instead of designing different control strategies for different values $x = (x_1, \ldots, x_n)$, we can use the fact that objects within each cluster are similar to each other, and design a single control strategy for all the objects from a cluster.

Practical example: cotton contains insects. Some of these insects destroy the cotton crop; to preserve the crop, farmers use insecticides. Among other crops, cotton is especially vulnerable to insects; as a result, world-wide, more insecticides are used on cotton than on any other crop. The problem is that it is often difficult to distinguish between harmful and harmless insects; as a result, insecticides are used even when only harmless insects are present, thus destroying the (often useful) insects and, in general, polluting the environment. It is therefore desirable to be able to distinguish between useful and harmful insects. To characterize insects, we can measure different geometric characteristics. In principle, we can have insects with all possible values of these characteristics within certain intervals; however, in practice, most insects fall into a few clusters; these clusters crudely correspond to species. Some clusters consist of harmful insects, some other clusters only contain harmless ones. Therefore, to distinguish between harmful and harmless insects, it is sufficient to subdivide all the insects into clusters, and then decide, for each cluster, whether its insects are harmful or not (for details, see, e.g., [12]).

It is therefore important to classify objects into clusters. There exist many clustering techniques. In some cases, when we have a large number of objects, we can use statistical techniques to get a statistically validated classification; see, e.g., [10].

In many real-life situations, however, there is not enough data to apply these statistically validated techniques. Instead, practitioners use heuristic clustering techniques that use fuzzy logic, neural networks, etc.; see, e.g., [2, 3, 5, 13]. The problem with these methods is that their results are not completely justified; moreover, most of these methods require that we choose certain parameters, and different choices of these parameters lead to different subdivision into clusters. So, there is a need to design justified clustering methods.

Such methods are described in this paper. In this description, we use several ideas first announced in [9, 14, 15].

2 Main Idea: How to Reformulate this Practical Problem in Precise Mathematical Terms

2.1 It Is Sufficient to Be Able to Solve 1-D Case

How can we solve the above practical problem? First, let us mention that in general, we have many parameters x_i that describe different objects from our class, so clustering in a multi-dimensional problem. There are cases when we have only a single parameter x_1 and when clustering is thus a 1-D problem, but in general, we have to process multi-D data, and processing multi-dimensional data is usually much more complex than processing 1-D data.

In many practical multi-D cases, however, it is possible to find a *single* parameter – it can be one of the parameters x_i or a combination of these parameters – that is sufficient to classify objects into clusters. In some cases, one parameter is not sufficient: e.g., when we classify insects by size, we get several classes, but some of these classes actually contain several different clusters. In such cases, in order to be able to solve most practical multi-D clustering problems, it is sufficient to be able to solve a 1-D problem; then, by applying this solution one or more times, we will be able to determine all the desired clusters.

In view of this comment, in this paper, we will only show how to solve a 1-D clustering problem.

In this problem, we have a single parameter x that characterizes different objects from our class, and we have several (n) objects with different values $x^{(1)}, \ldots, x^{(n)}$ of this parameter. We want to divide these objects into clusters.

2.2 Traditional Statistical Approach to Clustering

The value of the parameter x depends not only on the cluster to which this object belongs, but also on many other factors. For example, the size of an insects is determined not only by its species, but also on the weather conditions, on the environment, on the presence or absence of chemicals that are damaging to these insects, etc. As a result, different objects within the same class exhibit random variations from the average value corresponding to this class.

It is therefore reasonable to consider, for each cluster, a probability distribution that describes how frequently different values x occur for objects from this cluster. This is the main idea behind statistical clustering methods: for each class, we measure a lot of objects, determine the corresponding probability distribution; then, for each new object with the value x, we get the probability (actually, probability density) $\rho_i(x)$ that x is from cluster i, and we assign the object x to the cluster i_0 for which this probability is the largest, i.e., for which $\rho_{i_0}(x) \geq \rho_i(x)$ for all $i \neq i_0$.

2.3 Towards a Formal Definition of a Cluster

In this paper, we consider situations in which we do not have enough observations to determine all the distributions $\rho_i(x)$. Instead, all we observe is a sample $x^{(1)}, \ldots, x^{(n)}$, and we do not know which object corresponds to which cluster. If we knew the probability distributions $\rho_i(x)$ and the frequency p_i of objects from each cluster, then we could say that the observed data are a sample from a mixture distribution, with the density $\rho(x) = \sum p_i \cdot \rho_i(x)$. Since we do not know neither the densities $\rho_i(x)$ nor the frequencies p_i, all we can say is that the values $x^{(k)}$ are a sample from *some* probability distribution with an unknown density $\rho(x)$.

Let us relate this observation with the above (informal) description of a cluster: clusters are sub-intervals that contain the vast majority of the objects, and values in between these sub-intervals are rarer than inside them. A natural corollary of this description is that immediately outside the cluster sub-interval $[\underline{a}, \overline{a}]$, the density is smaller than inside, i.e., that the density function $\rho(x)$ is increasing for $x = \underline{a}$ and decreasing for $x = \overline{a}$. A continuous function that increases at $\underline{a}$ and decreases at $\overline{a} > \underline{a}$ must attain a (local) maximum inside the interval $[\underline{a}, \overline{a}]$; vice versa, if a function has a local maximum, then a sufficiently narrow interval around this maximum is a cluster in this sense.

Therefore, if, within an interval, there are at least two local maxima, this means that we can form at least two clusters. Thus, it is natural to identify clusters with local maxima of the probability density function (pdf) $\rho(x)$. To be more precise, clusters are neighborhoods of local maxima, neighborhoods that go both ways until the corresponding local minimum – the point at which decreasing changes to increasing or vice versa.

In other words, it is natural to define a cluster as the interval between two consequent local minima of the probability density function (pdf) $\rho(x)$, i.e., as a union of adjacent increasing and decreasing segments. Within each cluster, we have a unimodal distribution of x, with a probability density $\rho(x)$ that increases until a certain value x_0 and then decreases.

2.4 A Similar Problem with Known Solution and Why We Cannot Use It

If we could determine bounds on $\rho(x)$ based on the empirical data (i.e., on the values $x^{(1)}, \ldots, x^{(n)}$), then we would be able to use the known algorithms for finding local minima and local maxima of interval-valued functions; see, e.g., [17]. The problem is that, based on empirical data, we cannot find bounds on the pdf.

2.5 Enter Kolmogorov-Smirnov Bounds

What we can find is bounds on the *cumulative density function* (CDF) $F(x)$. Specifically, based on the sample values $x^{(1)}, \ldots, x^{(n)}$, we can determine an *empirical* CDF $F_{\rm emp}(x)$: for each x, $F_{\rm emp}(x)$ is defined as the ratio $\#\{k \mid x^{(k)} \le x\}/n$. This empirical distribution is the easiest to compute if we first sort the values $x^{(k)}$ in the increasing order $x^{(1)} \le x^{(2)} \le \ldots \le x^{(n)}$; then:

- $F_{\rm emp}(x) = 0$ for $x < x^{(1)}$;
- $F_{\rm emp}(x) = k/n$ for $x^{(k)} \le x < x^{(k+1)}$;
- $F_{\rm emp}(x) = 1$ for $x \ge x^{(n)}$.

Kolmogorov-Smirnov theorem (see, e.g., [19]) states that if the actual (unknown) PDF $F(x)$ is located on a known interval, then, for any given confidence level α, we can find the value ε for which, with this confidence, we have $\max\limits_x |F_{\rm emp}(x) - F(x)| \le \varepsilon$. Thus, with this given confidence level, we know that for every x, we have

$$F(x) \in \mathbf{F}(x) = [\underline{F}(x), \overline{F}(x)] \stackrel{\rm def}{=} [\max(F_{\rm emp}(x) - \varepsilon, 0), \min(F_{\rm emp}(x) + \varepsilon, 1)]. \tag{1}$$

Based on this interval information, we want to find out whether it is possible that $\rho(x)$ is increasing or decreasing on a given interval of values x. The function $\rho(x) = dF(x)/dx$ is decreasing (increasing) if and only if $F(x)$ is concave (convex). Thus, to solve our problem, we must determine concave and convex zones of the interval-valued function $\mathbf{F}(x)$.

In this paper, we propose a $O(n \cdot \log(n))$ time algorithm for determining such zones.

3 Proposed Algorithm

Our algorithm uses known $O(n \cdot \log(n))$ time incremental convex hull algorithms [4, 7, 8]; these algorithms, given n points $p_1, \ldots, p_n$ on the 2-D plane, find the convex hull of these points. These algorithms are called *incremental* because in the time $O(n \cdot \log(n))$, they not only find the convex hull of all n points $p_1, \ldots, p_n$, but also, for all k from 1 to n, convex hulls of the sets $\{p_1, \ldots, p_k\}$.

The algorithm for checking whether there exists a convex function $F(x)$ within given bounds is as follows:

- First, we sort all the values $x^{(k)}$ in the increasing order; this sorting takes $O(n \cdot \log(n))$ time. Without losing generality, we will therefore assume that the values $x^{(k)}$ are already sorted, i.e., that $x^{(1)} \le x^{(2)} \le \ldots \le x^{(n)}$.
- We compute the values $\underline{F}_k \stackrel{\rm def}{=} \max(k/n - \varepsilon, 0)$ and $\overline{F}_k \stackrel{\rm def}{=} \min((k-1)/n + \varepsilon, 1)$ ($\overline{F}_1 = 0$); this computation takes $O(n)$ time.

- Then, we use an incremental convex hull algorithm to compute, for every k from 1 to n, the convex hull C of the points $\overline{p}_1, \ldots, \overline{p}_k$, where $\overline{p}_k \stackrel{\text{def}}{=} (x^{(k)}, \overline{F}_k)$, and check whether the points $\underline{p}_1, \ldots, \underline{p}_k$, where $\underline{p}_k \stackrel{\text{def}}{=} (x^{(k)}, \underline{F}_k)$, are outside the interior of this convex hull (i.e., on or below the piecewise-linear curve $F_{\text{conv}}(x)$ describing the lower boundary of this convex hull); this requires $O(n \cdot \log(n))$ time.

To find zones of convexity and concavity, we do the following:

- First, we run the above algorithm until we find the last value k^+ for which the values $\underline{p}_k$ are outside the interior of the convex hull, i.e., for which it is still possible to have a convex function $F(x) \in \mathbf{F}(x)$ for $x \le x^{(k^+)}$.
- Then, we similarly process the values starting with $x^{(n)}$ backwards and find the smallest value k^- for which it is still possible to have a concave function $F(x) \in \mathbf{F}(x)$ for $x \ge x^{(k^-)}$.
- If $k^- \le k^+$, this means that it is possible to have a unimodal distribution $F(x)$, and $[x^{(k^-)}, x^{(k^+)}]$ is the interval of possible locations of its mode. In cluster terms, it means that the data is consistent with having only one cluster, with a "center" at some point $x \in [x^{(k^-)}, x^{(k^+)}]$.
- If $k^- > k^+$, this means that there are several clusters; to find these clusters, we apply the same algorithm to data starting with $(k^+ + 1)$-st point.

4 Justification of the Proposed Algorithm

One can easily see that the above algorithm requires $O(n \cdot \log(n))$ time. Let us show that this algorithm works correctly. Our algorithm is based on the ability to check convexity, so it is sufficient to show that the algorithm for checking convexity is correct.

1°. Let us first show that if there is a convex function $F(x) \in \mathbf{F}(x)$, then none of the points $\underline{p}_k$ are inside the interior of the convex hull C.

Indeed, let us assume that there is a convex function $F(x) \in \mathbf{F}(x)$. Then, the area $G \stackrel{\text{def}}{=} \{(x, y) \,|\, y \ge F(x)\}$ above this function is convex. The function $\overline{F}(x)$ is constant on each interval $[x^{(k-1)}, x^{(k)})$; thus, for arbitrary small $\delta > 0$, we have $\overline{F}(x^{(k)} - \delta) = \overline{F}(x^{(k-1)})$. From the definition of G, we conclude that $(x^{(k)} - \delta, \overline{F}(x^{(k)} - \delta)) = (x^{(k)} - \delta, \overline{F}(x^{(k-1)})) \in G$. In the limit $\delta \to 0$, we conclude that $\overline{p}_k = (x^{(k)}, \overline{F}(x^{(k-1)})) \in G$. Also, by definition of the set G, the pairs $\underline{p}_k = (x^{(k)}, \underline{F}(x^{(k)})$ are not inside this area.

By definition, the convex hull C is the intersection of all the convex sets that contain given points, thus, $C \subseteq G$; hence the points $\underline{p}_k$ cannot be inside the interior of C either.

2°. Vice versa, let us show that if all the points $\underline{p}_k$ are not inside the interior of the convex hull C, then there exists a convex function $F(x) \in \mathbf{F}(x)$.

We will prove that we can take $F_{\text{conv}}(x)$ as the desired function $F(x)$. Indeed, as a lower bound for a convex set, this function is convex.

Since the points $\underline{p}_k = (x^{(k)}, \overline{F}(x^{(k)}))$ are not inside the interior of the convex hull, we conclude that $\underline{F}(x^{(k)}) \le F_{\text{conv}}(x^{(k)})$. Since the function $\underline{F}(x)$ is constant on the interval $[x^{(k)}, x^{(k+1)})$, and the function $F_{\text{conv}}(x)$ is non-decreasing on this interval, we conclude that $\underline{F}(x) = \underline{F}(x^{(k)}) \le F_{\text{conv}}(x^{(k)}) \le F_{\text{conv}}(x)$ – i.e., $\underline{F}(x) \le F_{\text{conv}}(x)$ for all x.

Since all the points $\overline{p}_k = (x^{(k)}, \overline{F}(x^{(k-1)})$ are inside the convex hull, we conclude that $F_{\text{conv}}(x^{(k)}) \le \overline{F}(x^{(k-1)})$; similarly, since the function $\overline{F}(x)$ is constant on the interval $[x^{(k-1)}, x^{(k)})$, and the function $F_{\text{conv}}(x)$ is non-decreasing on this interval, we conclude that $F_{\text{conv}}(x) \le \overline{F}(x)$ for all x. Thus, $F(x) \in \mathbf{F}(x)$ for all x.

5 What if Measurements Come With Interval Uncertainty?

The above algorithm can be used to check whether there is a convex function $F(x)$ within arbitrary piecewise-constant bounds $\mathbf{F}(x) = [\underline{F}(x), \overline{F}(x)]$. An important case of this general situation stems from the fact that the values $x^{(k)}$ come from measurements, and measurements are never 100% accurate. As a result, the measured values $\widetilde{x^{(k)}}$ are, in general, different from the actual (unknown) values $x^{(k)}$ of the measured characteristics. Usually, we know the upper bound Δ for the (absolute value of) the measurement error $\Delta x_k \stackrel{\text{def}}{=} \widetilde{x^{(k)}} - x^{(k)}$; thus, instead of the exact value of $x^{(k)}$, we only know the *interval* $\mathbf{x}_k = [\underline{x}_k, \overline{x}_k] = [\widetilde{x^{(k)}} - \Delta, \widetilde{x^{(k)}} + \Delta]$ of possible values of $x^{(k)}$.

In this case, for every x, we have $F(x) \in [\underline{F}(x), \overline{F}(x)]$, where $\underline{F}(x) = \max(\underline{F}_{\text{emp}}(x) - \varepsilon, 0)$, $\overline{F}(x) = \min(\overline{F}_{\text{emp}}(x) + \varepsilon, 1)$, $\underline{F}_{\text{emp}}(x) = \#\{k \,|\, \overline{x}_k \le x\}/n$, and $\overline{F}_{\text{emp}}(x) = \#\{k \,|\, \underline{x}_k \le x\}/n$. Similarly to the above case, we have piecewise-constant bounds for $F(x)$ that change values only at $\underline{x}_k$ and $\overline{x}_k$. Thus, we can apply the above algorithm and find convexity zones in $O(n \cdot \log(n))$ time. (A similar algorithm for detecting monotonicity zones is described in [11].)

6 How to Parallelize Our Algorithm

Although our algorithm is pretty fast, its running time still grows with the number of points n. So, when the number of points is large, it is desirable to speed it up. A natural way to speed up an algorithm is to run it in parallel. There exist algorithms that compute the convex hull of n points in $O(\log(n))$ time on n processors [1] (see also [6]). If we use this algorithm, we can check, for every k, in $O(\log(n))$ time, whether the values $x^{(1)}, \ldots, x^{(k)}$ are consistent with the convexity of $F(x)$. We can use this check in two different ways:

- we can run n checks in parallel; thus, by using $O(n^2)$ processors, we detect the desired zones in $O(\log(n))$ time;
- alternatively, we can use bisection on the interval $[1, n]$ to find the last value k that is still consistent with convexity; binary search requires $\log(n)$ class, so we find the zones in $O(\log^2(n))$ times by using n processors.

It is worth mentioning that a similar drastic speed-up is possible if we use parallel computations in a similar problem of detecting areas of monotonicity [18].

Acknowledgments

This work was supported in part by Belgian Fund for Scientific Research grant G.0366.98, by NASA under cooperative agreement NCC5-209 and grant NCC2-1232, by the Future Aerospace Science and Technology Program (FAST) Center for Structural Integrity of Aerospace Systems, effort sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grants F49620-95-1-0518 and F49620-00-1-0365, by NSF grants CDA-9522207, EAR-0112968, EAR-0225670, and 9710940 Mexico/Conacyt, by IEEE/ACM SC2001 and SC2002 Minority Serving

Institutions Participation Grants by Small Business Innovation Research grant 9R44CA81741 to Applied Biomathematics from the National Cancer Institute (NCI), a component of the National Institutes of Health (NIH), and by a research grant from Sandia National Laboratories as part of the Department of Energy Accelerated Strategic Computing Initiative (ASCI).

This research was partly done when V.K. was a Visiting Faculty Member at the Fields Institute for Research in Mathematical Sciences (Toronto, Canada)

The authors are thankful to Rene Alt for his encouragement and to all the participants of SCAN'2002 for valuable discussions. We are especially thankful to Paul Zimmermann whose ideas helped improve the algorithms; he is truly a co-author of this paper.

References

[1] S. G. Akl and K. A. Lyons, *Parallel Computational Geometry*, Prentice Hall, Englewood Cliffs, 1993.

[2] J. C. Bezdek, J. Keller, R. Krisnapuram, and N. R. Pal, *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*, Kluwer, Boston, 1999.

[3] J. C. Bezdek and S. K. Pal (eds.) *Fuzzy models for pattern recognition*, IEEE Press, N.Y., 1992.

[4] J.-D. Boissonant and M. Yvinec, *Algorithmic Geometry*, Cambridge University Press, Cambridge, UK, 1998.

[5] H. Bunke and A. Kandel (eds.), *Neuro-Fuzzy Pattern Recognition*, World Scientific, Singapore, 2000.

[6] W. Chen, K. Wada, and K. Kawaguchi, "Robust algorithms for constructing strongly convex hulls in parallel", *Theoretical Computer Science*, 2002, Vol. 289, pp. 277–295.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, MA, and Mc-Graw Hill Co., N.Y., 2001.

[8] H. Eldelsbrunner, *Algorithms in Computational Geometry*, Springer-Verlag, Berlin, 1987.

[9] G. Frederix and E. J. Pauwels, "Segmentation based on principled self-supervising clustering", *Abstracts of the First SIAM Conference on Imaging Science*, Boston, Massachusetts, March 4–6, 2002, p. 25.

[10] K. Fukunaga, *Introduction to statistical pattern recognition*, Academic Press, San Diego, CA, 1990.

[11] J. Lorkowski and V. Kreinovich, "If we measure a number, we get an interval. What if we measure a function or an operator?", *Reliable Computing*, 1996, Vol. 2, No. 3, pp. 287–298.

[12] H. T. Nguyen, N. R. Prasad, V. Kreinovich, and H. Gassoumi, "Some Practical Applications of Soft Computing and Data Mining", In: A. Kandel, H. Bunke, and M. Last (eds.), *Data Mining and Computational Intelligence*, Springer-Verlag, Berlin, 2001, pp. 273–307.

[13] S. K. Pal and S. Mitra, *Neuro-Fuzzy Pattern Recognition: Methods in Soft Computing*, John Wiley, New York, 1999.

[14] E. J. Pauwels and G. Frederix, "Image Segmentation by Nonparametric Clustering Based on the Kolmogorov-Smirnov Distance", In: D. Vernon (ed.), *Proceedings of the 6th European Conference on Computer Vision ECCV'2000, Dublin, Ireland, June 26–July 1, 2000, Part II*, Springer Lecture Notes in Computer Science, Vol. 1843, 2000, pp. 85–99.

[15] E. J. Pauwels, G. Frederix, and G. Caenaen, "Image segmentation based on statistically principled clustering", *Proc. of 2001 International IEEE Conference on Image processing*, 2001, Vol. 3, pp. 66–69.

[16] Ramas website http://www.ramas.com, 2002.

[17] K. Villaverde and V. Kreinovich, "A linear-time algorithm that locates local extrema of a function of one variable from interval measurement results," *Interval Computations*, 1993, No. 4, pp. 176–194.

[18] K. Villaverde and V. Kreinovich, "Parallel algorithm that locates local extrema of a function of one variable from interval measurement results", *Reliable Computing*, 1995, Supplement (Extended Abstracts of APIC'95: International Workshop on Applications of Interval Computations, El Paso, TX, Febr. 23–25, 1995), pp. 212–219.

[19] H. M. Wadsworth, Jr. (eds.), *Handbook of statistical methods for engineers and scientists*, McGraw-Hill Publishing Co., New York, 1990.