A Physically Inspired Clustering Algorithm: to Evolve Like Particles

Teng Qiu¹, Kaifu Yang¹, Chaoyi Li², Yongjie Li¹*

¹Key Laboratory of NeuroInformation, Ministry of Education of China, School of Life Science and Technology, University of Electronic Science and Technology of China, Chengdu, 610054, China
²Shanghai Institutes of Biological Sciences, Chinese Academy of Sciences, Shanghai, 200031, China
*Corresponding author. Email: liyj@uestc.edu.cn

Abstract: Clustering analysis is a method to organize raw data into categories based on a measure of similarity. It has been successfully applied to diverse fields from science to business and engineering. By endowing data points with physical meaning like particles in the physical world and then leaning their evolving tendency of moving from higher to lower potentials, data points in the proposed clustering algorithm sequentially hop to the locations of their transfer points and gather, after a few steps, at the locations of cluster centers with the locally lowest potentials, where cluster members can be easily identified. The whole clustering process is simple and efficient, and can be performed either automatically or interactively, with reliable performances on test data of diverse shapes, attributes, and dimensionalities.

Main Text: Clustering aims to divide the whole data set, usually featuring high dimension and huge volume, into several groups with a close similarity within groups and large difference between them. Cluster analysis is widely used by researchers from various disciplines to analyze multivariate data (1). Although many clustering algorithms have been proposed (1, 2), clustering still remains quite challenging (1).

Our clustering algorithm is inspired by a question: what if data points had mass? They would evolve like particles in the physical world. Take two-dimensional (2D) data points for instance and imagine the 2D space as a horizontally stretched rubber sheet (3). Intuitively, the rubber sheet would be curved downward when the data points with mass lie on it. This in turn could trigger the evolution of the points, moving from higher to lower potential areas (i.e., the “descending direction”) and ultimately gathering at several locations of locally lowest potentials (Fig. 1F), where cluster members are then identified.

Instead of seeking analytical solutions for the movement trajectories of all data points, we approximate each of them by a zigzag path consisting of a sequence of “short hops” (4). The first hop is basically towards the location of the nearest point along the descending direction of potentials. See step 4 in table S1 for more explanations. Potentials at data points are defined by step 2 in table S1. Consistent with the situation on the rubber sheet, points in denser areas are more likely to have lower potentials.

After the first round of hops of all points, a special “graph”, an in-tree (5), is constructed (6) as shown in Fig. 1B. An in-tree is a directed tree with a special vertex or point, called the root, such that every other vertex has an unique directed path to reach it. This makes the next hops much easier by just following the directions of the edges (lines that connect vertices) of the graph.

However, before the next hops, some undesired edges should be first cut off (i.e., removed) to make the evolutions of different groups independent just like the physical
situation on the rubber sheet. Since, according to the criterion for the first hops, each of the undesired edges generally starts from the vertex with the locally minimal potential in the dense area of one group and ends to the vertex with lower potential in another group, the undesired edges are unusual and generally much longer compared with the connections of vertices within groups, and thus the edge length can serve as a reliable reference to identify those undesired edges. A simple way is to cut the $K$ longest edges, whereas $K$ should be prespecified. To avoid this requirement, the cutting process can be performed in sequence under the supervision of a few labeled data, starting from the longest one and stopping when data with different labels are distributed in different sub-graphs, i.e., a supervised cutting method (detailed in fig. S1). Also, if data can be visualized, a simple interactive cutting method can be used (see principle in fig. S2).

After cutting, all points are guaranteed to gather at their destinations, the roots, after $S = \lceil \log_2(H) \rceil$ rounds of hops (fig. S3 and step 6 in table S1). $H$ is the maximum number of edges of all directed paths in the graph after cutting. In Fig. 1, $H = 7$ (the edge number on the blue path in Fig. 1C) and $S = 3$ (Fig. 1, D to F). See analysis on other test data sets in table S2. The root vertices are then seen as cluster centers or exemplars (7) and the points with same roots are assigned into same clusters.

We first applied interactive cutting on synthetic data (8) in Fig. 2A, where the undesired edges are distinguishable. Based on the user’s simple interactive operations recorded by the red crosses, the undesired edges beside the red crosses can then be accurately identified by computer and removed (Fig. 2B). By virtue of our powerful visual system, the interactive method can effortlessly cluster complex shapes of data sets (fig. S4) which are usually difficult for certain clustering algorithm, and is insensitive (fig. S5) to the only free parameter $\sigma$ involved in step 2 of table S1.

For non-visible mushroom data set (9) containing 8124 mushrooms, each with 22 character attributes, they can first be transformed to visible 2D data points (fig. S6) by ISOMAP (4), one of many excellent non-linear dimensionality reduction techniques (10). Then, clustering can be easily completed based on interactive cutting. An error rate of 3.35% was achieved, comparable to other approaches (2, 11).

Since interactive cutting is performed in a visible environment, the clustering results can be consistent with the structures perceived by us and thus are very reliable. Furthermore, the participation of human users can provide guideline to clustering procedure to obtain more meaningful results, highlighted also in (12).

Compared with interactive cutting, the supervised cutting used in the following experiments is fully automatic and blind to the dimensionality and attribute of data sets (i.e., no data transformation is needed). Also, the cutting process can be quite reliable due to the supervision of known information, i.e., a few labeled data.

For synthetic data (13) in Fig. 2D, five salient dense groups were well clustered in Fig. 2F when 20 labeled data were given. Compared with the method proposed in (13), our method worked well for a much larger range of $\sigma$ values (fig. S7).

For the mushroom data again, we tried various values for parameter $\sigma (= 0.001, 0.01, 0.5, 1, 2, \ldots, 20, 30, 50, 100, 500, 1000)$. For each value, we repeated the test 20 times, all under the supervision of 200 randomly selected data labeled according to their true annotations. The average clustering error rates were all below 1% and the average cluster numbers for most of them were closely around 7, with very little deviation (Fig. 3). In fact, as the number of labeled data was increased to 800, the performance became almost perfect (fig. S8).

We further applied the supervised cutting method to the Olivetti Face Database (14). It involves 40 subjects, each containing 10 grayscale faces of $112 \times 92$ pixels. In
this experiment ($\sigma = 100$), 3 faces, e.g., the 1st, 5th, and 9th ones, of each subject were selected as labeled data (faces from the same subject were labeled identically). Supervised with these 120 labeled faces, the cutting process was terminated when the cluster number reached 99, with only 3 faces falsely clustered. 271 faces (30 of them were labeled faces) were correctly assigned to the 30 largest clusters, better than the result (161 faces) in (13). In fact, the 40 largest clusters contained as many as 315 faces. Also, there were 41 singleton clusters, each having only one face. These faces may be isolated as marginal points or noises in Fig. 2F. After re-finding the roots for them based on their initial connections before cutting, the number of false assignments was degenerated a little from 3 to 10. However, the number of faces contained in the 30 largest clusters (22 of them are shown in Fig. 4) was increased to 286 and the cluster number was reduced to 58 (fig. S9). Furthermore, the clustering results changed very little when the number of labeled faces in each subject was reduced from 3 to 2 (fig. S10), or when the value of $\sigma$ varied greatly (table S3).

For the related density-based clustering methods, e.g., mean-shift (15, 16) and DENCLUE (17), the density peaks are searched by solving the extreme-value problem of a continuous function, which involves a time-consuming iterative process and is only applicable to data defined by a set of coordinates (13). However, our approach, learned from physics, can simply yet effectively achieve the similar goal of converging all points to the roots with locally lowest potentials, meanwhile avoiding the limitations of the above methods. Although the potential variable in our approach has some density-like feature, the clustering performance is not largely rely on how well such potential is consistent with the underlying density of data (or in other words the performance is insensitive to the only parameter $\sigma$).

Our approach also has several advantages over related graph-based techniques. Minimal spanning tree (MST) clustering (18) first constructs a tree with the minimal sum of edge length and then removes the undesired edges. However, it is usually not easy to identify those edges, since they are in general the connections of the marginal points of different clusters, lacking the salient feature in length especially when clusters are close to each other or contaminated by noise (fig. S11). Directed tree clustering (19) requires that the parent (with the same role as the transfer point in our approach) of each vertex should be determined among the vertices within a fixed radius. However, choosing an appropriate radius is non-trivial, a drawback not involved in our method (see step 4 in table S1). Also, the algorithm in (19) requires additional effort to avoid generating cycles in the graph. In contrast, a cycle-free network is naturally obtained in our approach as a result of following the law of nature.

We learnt from the probable evolving behavior of particles an effective combination of two variables, the hidden one (potential) and apparent one (pair-wise distance), and consequently all data points are well organized into an in-tree structure. Similar combination has also benefited (13) to represent each point just with 2 quantities in the 2D feature space where cluster centers can be fast identified. Interestingly, we can merge the methods in (13) into our in-tree structure to form together a pool of powerful clustering methods (see text S6 and fig. S12).

References and Notes:
1. A. K. Jain, Pattern Recognit. Lett. 31, 651 (2010).
2. R. Xu, D. Wunsch, IEEE Trans. Neural Netw. 16, 645 (2005).
3. It's a standard way to illustrate Einstein's General Relativity, the core of which is often summarized as "matter tells space-time how to curve and space-time tells matter how to move". See http://einstein.stanford.edu/SPACETIME/spacetime2

4. J. B. Tenenbaum, V. De Silva, J. C. Langford, Science 290, 2319 (2000).

5. J. L. Gross, J. Yellen, Handbook of Graph Theory. (CRC press, Boca Raton, FL, 2004).

6. See supporting material for the proof.

7. B. J. Frey, D. Dueck, Science 315, 972 (2007).

8. P. Fränti, O. Virmajoki, Pattern Recognit. 39, 761 (2006).

9. From http://archive.ics.uci.edu/ml/.

10. J. A. L. a. M. Verleysen, Nonlinear dimensionality reduction. (Springer, New York, 2007).

11. A. Gionis, H. Mannila, P. Tsaparas, ACM Trans. Knowl. Discovery Data 1, 4 (2007).

12. B. Shneiderman, Science 343, 730 (2014).

13. A. Rodriguez, A. Laio, Science 344, 1492 (2014).

14. F. S. Samaria, A. C. Harter, in Proceedings of 1994 IEEE Workshop on Applications of Computer Vision (IEEE, New York, 1994), pp. 138-142.

15. K. Fukunaga, L. Hostetler, IEEE Trans. Inf. Theory 21, 32 (1975).

16. Y. Cheng, IEEE Trans. Pattern Anal. Mach. Intell. 17, 790 (1995).

17. A. Hinneburg, D. A. Keim, in Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, R. Agrawal, P.E. Stolorz, G. Piatetsky-Shapiro, Eds. (AAAI Press, Menlo Park, CA, 1998), vol. 98, pp. 58-65.

18. C. T. Zahn, IEEE Trans. Comput. 100, 68 (1971).

19. W. L. Koontz, P. M. Narendra, K. Fukunaga, IEEE Trans. Comput. 100, 936 (1976).

Acknowledgments: We thank Hongmei Yan, Ke Chen for stimulating suggestions and Shaobing Gao, Yezhou Wang for technical support. We also thank Nancy Westanmo-Head for assistance in smoothing the English. This work was supported by the Major State Basic Research Program (2013CB329401), the Natural Science Foundations of China (61375115, 91120013, 31300912). The work was also supported by the 111 Project (B12027) and PCSIRT of China (IRT0910).
Fig. 1. An illustration for the evolution of 2D data point system. The original data points, sampled from 2 elongated Gaussian distributions, are in a state of rest in (A). Evolution begins with points hopping to the locations of their first transfer points according to step 4 in table S1. Consequently, an in-tree is constructed in (B). Colored point denotes the root of the in-tree. Before evolution continues, the undesired edge (purple) should be cut off to obtain two independent in-trees as shown in (C). In (D), (E) and (F), points hop to the locations of their next transfer points according to the arrows in (C), (D) and (E), respectively. Evolution stops (all points return to a state of rest) in (F) where data points gather at different roots. Points gathered in the same root are thus assigned to the same cluster.
Fig. 2. Clustering two synthetic data by interactive cut (A) and supervised cut (D), respectively. (A) The constructed in-tree (arrows are ignored) of the first synthetic data (points). Fourteen red crosses denote user’s selections, close to the undesired edges. (B and C) The cutting and clustering results, respectively. (D) The constructed in-tree of the second synthetic data. Twenty colored triangles represent different labeled data. (E) The cutting result under the supervision of labeled data. (F) Clustering result. Points in the dense areas are well clustered into five groups. In (C) and (F), different colors represent different clustering assignments, and black circles denote the places where the root vertices in (B) and (E) locate.
Fig. 3. Clustering mushroom data by supervised cutting method over different parameters. For each value of $\sigma$, 200 (out of 8124) mushroom data are randomly selected as supervised data. The obtained cluster number versus $\sigma$ is plotted in blue and error rate of assignment versus $\sigma$ in red. The points indicate the means and the error bars indicate the standard errors over 20 random tests.

Fig. 4. The 22 largest clusters for the face data by supervised method after merging the singleton clusters. Faces (originally grayscale images) that are assigned to the same cluster are marked with the same color.
Supplementary Materials:

Texts S1-S6
Figures S1-S14
Tables S1-S4
Table S1.  Computational steps of the proposed clustering algorithm

The proposed clustering algorithm takes as input the pair-wise distance between all pairs of data from data set \( \{X_1, X_2, \ldots, X_N\} \). The only free parameter \( \sigma \) appears in step 2, but to which our model is not too sensitive, because what we rely on in step 4 is the difference of potentials, not their absolute values. \( W_t \) denotes the length of edge \((i, l_i')\) that starts from point \( i \). The end point \( l_i' \) is namely the transfer point of point \( i \) in the \( t \)-th round of hops. Step 4+ is a supplementary operation to step 4 aimed at coding the algorithm more easily.

| Step | Description | Implementation |
|------|-------------|----------------|
| 1    | Compute pair-wise distance | \( D_{i,j} = d(i,j), \quad i,j = 1,2,\ldots,N \) |
| 2    | **Compute potential:** The total potential at each point is approximated by the sum of potentials exerted from all points. Each point is assumed to generate an isotropic potential around it with negative value whose magnitude declines exponentially with distance. | \( P_i = -\sum_{j=1}^{N} \frac{D_{i,j}}{\sigma}, \quad i = 1,2,\ldots,N \) |
| 3    | **Compute potential difference** | \( C_{i,j} = P_i - P_j, \quad i,j = 1,2,\ldots,N \) |
| 4    | **Determine the first transfer points:** Each point will select as the only transfer point the nearest one or the one with the smallest index among the several nearest ones from such candidate point set, if non-empty, that contains points with smaller potentials or the same potential but smaller data indexes (illustrated in fig. S14). | The candidate point set of every point \( i \) is defined by \( J_i = \{j | C_{j,i} < 0\} \cup \{j | C_{j,i} = 0 \& j < i\} \). If \( J_i \) is non-empty, then set the first \((t = 1)\) transfer point of point \( i \) and the length of the edge between them respectively as \( l_i' = \min_{j \in J_i} \arg \min_{j} D_{i,j}, \quad W_{i}^{l} = D_{i,l_i'} \). |
| 4+   | For the point (root) with an empty optional point set, its transfer point is set as itself. | If \( J_i \) is empty, then define \( l_i' = i, \quad W_{i}^{l} = -\infty \). |
| 5    | **Cut undesired edges:** Edges that connect different clusters can be cut off either (a) interactively, or automatically according to the edge length (in decreasing order) under supervision of (b) labeled data or (c) predefined cluster number. | For any identified undesired edge \((i, l_i')\), the transfer point of point \( i \) (new root) and the length of the edge between them are respectively changed to \( l_i' = i, \quad W_{i}^{l} = -\infty \). |
| 6    | **Find roots:** All points iteratively hop to the next transfer points based on the edge directions of the graph obtained from the previous round of hops until converging at the roots. | In each round, the transfer points of all points are updated in parallel by \( l_{i}^{t+1} = l_{i}^{l} \) until \( l_{i}^{t+1} \) equals \( l_{i}^{t} \) for all points. |
S1. Details for the distance metrics in step 1 of table S1

We use the simple Euclidean metric $d(i,j) = \|x_i - x_j\|$ for the data with numerical attributes, and $d(i,j) = \sum_m 1\{x_{im} = x_{jm}\}$ for the data with categorical attributes (e.g., the original mushroom data), where $1\{x_{im} = x_{jm}\}$ equals 1 if the $m$th attribute for point $i$ and $j$ are matched, else 0.

S2. Proof for the in-tree structure constructed by step 4 of table S1

Before proving, some definitions (D) and facts (F) (1, 2), together with the requirements (R) of step 4 in table S1, are listed as follows.

D1: A digraph $G$ is a directed graph $G$, a directed edge of which is also called an arc. $V(G)$ and $E(G)$ denote respectively vertex set and arc set; $n(G)$ and $m(G)$ denote respectively the number of vertices and arcs.

D2: The indegree, outdegree and degree of a vertex $v$ are respectively the number of directed edges directed to, directed from and connected with it, and are denoted respectively as $d^-(v)$, $d^+(v)$ and $d(v)$.

D3: A directed cycle of a directed graph is a cycle that for any vertex $v$ on the cycle, travelling along the directions of the arcs, it can return to itself. Thus, $d^-(v) + d^+(v) = 1$.

D4: A connected graph is a graph of which any two vertices are connected by a sequence of edges.

D5: A tree is a connected graph without cycle.

D6: An in-tree, also called in-arborescence or in-branching, is a digraph that meets the following conditions: (a) only one vertex has outdegree 0; (b) any other vertices has outdegree 1; (c) there is no cycle in it; (d) it is a connected graph.

F1: If undirected graph $G$ is a tree, $m(G) = n(G) - 1$.

F2: For digraph $G$, $\sum_{v \in d^+(G)} d^+(v) = \sum_{v \in d^-(G)} d^-(v) = m(G)$ and $d(v) = d^-(v) + d^+(v)$

R1~R3: Each point (or vertex) will (R1) select as the only transfer point (or directed vertex) the nearest one or the one with the smallest index among the nearest ones from such point set, (R2) if it’s non-empty, (R3) that contains points with smaller potentials or the same potential but smaller data indexes.

Proof. (i) There should be one global minimum for the potential values of all vertices. If only one vertex has the minimum potential value, then it will be the only vertex with outdegree 0 (without directed vertex), since R2 is not met. If several vertices are of the minimum potential, then since the vertex indexes are unique, there is still only one vertex among them having the smallest index, for which R2 is not satisfied and thus the outdegree is 0. Hence, for all cases, the condition (a) is met.

(ii) Let vertex $r$ be the only vertex with outdegree $d^+(r) = 0$. Then, for any other vertex, say $v(\neq r)$, there will be at least one vertex, e.g., $r$, meeting R3, and thus R2 is satisfied. Then according to R1, $d^+(v) = 1$. Therefore, the condition (b) is met.

(iii) Suppose there is a cycle $C$ in digraph $G$.

If $C$ is a directed cycle $v_i \rightarrow v_j \rightarrow \cdots \rightarrow v_i$, then according to R3, the potential values for these vertices should meet $P_i \geq P_j \geq \cdots \geq P_i$, which is true only when $P_i = P_j = \cdots = P_i$. Then according to R3 again, the data indexes for these vertices should meet $i > j > \cdots > i$, resulting in $i > i$, an obvious contradiction.

If $C$ is not a directed cycle, we claim that there exists at least one vertex of $C$, say $w$, with indegree 2 or outdegree 2 otherwise $C$ is a directed cycle. However, if $d^-(w) = 2$, then it will contradict the condition (b); If $d^+(w) = 2$, since every vertex
v of a cycle has degree 2, i.e., 
\[ d^+(v) + d^-(v) = 2, \]
and according to F2
\[ \sum_{v \in \mathcal{P}(C)} d^+(v) = \sum_{v \in \mathcal{P}(C)} d^-(v), \]
there must be at least one vertex of C having outdegree 2, contradicting the condition (b) again.

Therefore, there is no cycle in digraph \( G \), i.e., the condition (c) is met.

(iv) Suppose \( G \) is not connected, e.g., containing \( M \) connected sub-graphs \( G_1, G_2, \ldots, G_M \). Since digraph \( G \) has no cycle, there should be no cycle in each subgraph as well. Hence, each subgraph is at least a tree. Assume the vertex \( r \) of outdegree 0 is in \( G_i \), then vertices in any other subgraph \( G_k \) \((k \neq i, k \in \{1, 2, \ldots, M\})\) are all with outdegree 1, thus, according to F2,
\[ m(G_j) = \sum_{v \in \mathcal{P}(G_j)} d^+(v) = \sum_{v \in \mathcal{P}(G_k)} 1 = n(G_k), \]
which contradicts F1. Therefore, digraph \( G \) is connected, i.e., the condition (d) is met.

According to (i), (ii), (iii) and (iv), the digraph \( G \) is an in-tree.

S3. Notations to the term of data index involved in step 4 of table S1

In step 4 of table S1, the data indexes (i.e., the order in which data points are processed), together with the potential values, serve to provide references for each point to choose a suitable transfer point. The role of the data indexes is significant to guarantee the in-tree structure, especially when the potential values for all points are the same. This may account for why most of the test data sets have consistent results for a large range of values for parameter \( \sigma \). Also, the elaborate design for data index can avoid the unexpected “singleton cluster phenomenon” after cutting operation (discussed in fig. S14).

However, involving data index term does not mean that the result is sensitive to it. The influence is very limited. The reason can be analyzed as follows:

(i) If point \( i \) is not with the local minimal potential, then since its candidate point set \( J_i \) consists of two groups of points, i.e., the points with smaller potentials (group A) and the points with the same potential but smaller data indexes (group B), only points in group B is affected by data index. The points in group B are so special that they do not always exist. Even if group B is non-empty, according to the “constrained proximity criterion” that only the nearest point or the point with the smallest index among the nearest ones in \( J_i \) will be selected as the transfer point, the selected transfer point \( I_i \) is at least very close to point \( i \). Therefore, the change of the data indexes only affects the connections or edges in local areas, whereas these local edges, however they are changed, are in general much shorter than those undesired edges among groups and thus less likely to be cut off. Consequently, the near points connected by these local edges can be clustered into the same groups.

(ii) Otherwise, the candidate points with lower potential values in set \( J_i \) will come from other groups. So, no matter how data indexes are assigned to those points, the edge between point \( i \) and its transfer point \( I_i \) will in general be much longer than the edges among points within group, and thus will be cut off later. This is what we expect. Consequently, the clustering result will not be largely affected.

The above is further demonstrated by a test with 50 different randomly permutations to a data set \((N = 5000)\), as shown in fig. S13.

For the clustering results of face data with \( \sigma = 100 \) shown in Fig.4 and fig. S9, the number of vertices with different potential values is 354 out of 400, i.e., only 6 vertices were directly affected by data indexes and the clustering result changed very little after randomly permuting the original data set. See table S3 for details. Table S3
also lists the results for other $\sigma$ values, for which all vertices have different potential values and the change of data indexes makes no change to the clustering results. For mushroom data, after randomly permuting the arrangement of the original data set, the clustering results also change very little (fig. S8).

S4. Notations to step 4+ and 5 in table S1

In order to make the evolution or iteration in step 6 of table S1 go in the simplest form, we introduce in step 4+ and 5 in table S1 such technical operations $I_i^j = i$; $W_i^j = -\infty$, equivalent to adding self-loops on root vertices. This brings the convenience that the rooted vertices are not necessary to be specially treated in step 6. They can also evolve like other points, while, in effect, they are still in the state of rest. Although adding self-loops on root vertices means that the in-tree structure is slightly destroyed, this makes no effect on clustering result, since there still be one and only one path for other points to reach their corresponding root vertices. So, each of the subgraphs after step 4 is still called in-tree, ignoring the above operations on root vertices.

S5. Notations to the labeled data

Ideally, only one labeled data in each cluster is enough for supervised cutting method if they are from the dense part of each cluster. In practice, such requirement is hard to satisfy. However, we can employ a little more labeled data to increase the probability that some of them may be located in the dense parts. This will burden the data collection, whereas the result will be more reliable and the over-cutting phenomenon, i.e., some large groups are divided into several smaller ones, can be alleviated, since more labeled data helps some small clusters containing the same labeled data to be merged into larger ones.

S6. Detailed comparison with the work of Science-14 (3)

It is essential yet challenging in pattern recognition to extract effective features from raw data. The success of Science-14 (fig. S12A) owes a lot to two effective features, the local density $\rho$ and the distance $\delta$ from points of higher density, which are discovered based on a simple assumption that “the cluster centers are characterized by a higher density than their neighbors and by a relatively large distance from points with higher densities”. These two features are so effective that only cluster centers, actually density peaks, have large $\delta$ and relatively high $\rho$, thus popping out as outliers and being separable from the remaining points in the “Decision Graph” (a scatterplot of features $\delta$ and $\rho$ for all points). Consequently, the problem of finding density peaks is reduced to the problem of detecting outliers, rather than the mathematical problem of solving the extreme values of a continuous function. In addition, it is extremely easy to detect those outliers (i.e., cluster centers), since the 2D Decision Graph provides a visualized circumstance so that the detection process can be performed just by human user’s simple interactive operation (denoted as $\text{Int-DCC}$ in fig. S12A) instead of by some complicated outlier detection algorithms.

Thus, the effectiveness, both in quality and quantity, of the features derived from a simple assumption makes the clustering procedure in Science-14 simple and effective for general cases.

However, the characteristics of real-world data is more complicated than a simple assumption. The expected salient images, with some points popping out from the Decision Graphs, may not always be obtained, e.g., when clustering faces (tested in Science-14) or documents (4) (fig. S12F). Also, the expected salient images rely on
a free parameter and in general the range of the suitable values for this parameter is limited (e.g., fig. S7). Facing the less-than-ideal Decision Graphs, human users may be placed in the dilemma of how to make their interactive choices reliable, since the feature space lacks any other useful information of the original data set that they can rely on. Although *Science-14* has further proposed a method by taking the $K$ data points with the highest values of $\gamma_i = \rho_i \delta_i$ as cluster centers, such strategy (denoted as $K$-DCC in fig. S12A) has its own limitations: (i) the value of $K$ is usually hard to specify in advance, since the plot of $\gamma_i$ sorted in decreasing order cannot always reveal a clear threshold; (ii) points with high $\gamma$ are not always the best cluster center candidates, or in other words, cluster centers cannot be guaranteed to be more separable in this 1D feature space shaped by $\gamma$, since according to the assumption for the cluster centers in *Science-14*, both $\rho$ and $\delta$ in essence are only of locally high value (this in turn shows the advantage of constructing the Decision Graph directly by these two features $\rho$ and $\delta$ in a 2D space, which in general makes the cluster centers more separable).

Our approaches (fig. S12B) are inspired by the clustering phenomenon when a swarm of particles evolve, and implemented based on an in-tree structure from graph theory. By following the law of nature, i.e., the evolving tendency of particles, our clustering procedure is also endowed with order, certainty and efficiency. By utilizing the in-tree structure, the real implementation is not only vivid but also with simplicity. Although some edges are undesirably produced, these undesired edges, like density peaks in *Science-14*, are also separable from others and thus easy to be identified and then removed, since their edge lengths are in general much longer than their surroundings; However, unlike the Decision Graph in *Science-14*, the in-tree structure also contains other direct information of data points which can play important auxiliary roles in identifying the undesired edges, e.g., (i) for interactive cutting (Int-Cut), the structure information that all data points show can aid human users to make a more reliable and meaningful interactive choice to remove the undesired branches; (ii) for supervised cutting (Sup-Cut), the labeled data points can help machine automatically identify the undesired edges with a certain extent of reliability, and provide reference (e.g., how fast and well these labeled data are clustered) for us to adjust the only one free parameter $\sigma$ to achieve better results. When clustering very complicated data, the above auxiliary roles of the information of raw data will be more valuable.

In fact, since the edge length $W$ and potential value $P$ in our methods are computationally to some degree similar to feature $\delta$ and $\rho$ in *Science-14*, respectively, the methods, Int-DCC and K-DCC, in *Science-14* can be merged into our in-tree-based clustering structure, respectively resulting in the new cutting methods, Int-DCC-Cut and K-DCC-Cut, as shown in fig. S12C. Int-DCC-Cut serves to remove or cut the directed edges that start from the points interactively selected as cluster centers from the scatterplot of $W_i$ and $|P_i|$ for each point. K-DCC-Cut works to remove the edges that start from the points with the $K$ (prespecified) largest values of $W_i \times |P_i|$. After cutting, benefited from the in-tree structure, the parallel root finding process in our methods can be used to perform clustering assignment (shown as FR in fig. S12, B and C), instead of a sequential assignment starting from the points that directly take the identified cluster centers as the nearest points of higher density (shown as FGM in fig. S12A).

Among the five cutting methods (a “Method Pool” based on in-tree structure) in fig. S12C, K-Cut and K-DCC-Cut are simple and automatic, whereas the biggest problem for them is that $K$ is usually hard to specify before clustering. Int-Cut is
intuitive and reliable since the point distribution is visible when cutting interactively, whereas it needs transformation for the invisible data (e.g., high dimensional or non-numerical data). Int-DCC-Cut has no need for transformation and thus is very convenient (fig. S12D), whereas it would become helpless when facing some non-salient Decision Graphs (fig. S12 E and F). Sup-Cut is automatic, with no above need of (i) specifying the cluster number, (ii) transforming the original data, (iii) obtaining a salient intermediate result (i.e., Decision Graph) and (iv) performing additional interactive operations. Also, The whole process for Sup-Cut is to some degree reliable due to the supervision of the labeled data (whereas no training process is involved). Compared with interactive cutting methods Int-Cut and Int-DCC-Cut, the automatic characteristic of Sup-Cut make the machine have the self-learning potential just like a child, i.e., from the finite information (told by teachers or parents), the raw data in his brain derived from his past experience will be well organized and associated with such known information and thus be of meaning. With the massive raw data (from experience) and finite known information (from learning) being alternatively acquired, he will gradually become knowledgeable and smart. Similarly, the machine (computer) can become more intelligent (data are better organized) as more labeled and unlabeled data are gradually added in. However, for Sup-Cut, the performance of clustering results rely on how effective the supervised data are. In addition, when facing some complicated data, an appropriate value of the free parameter $\sigma$ is expected to alleviate the over-cutting phenomenon (though the merging process based on the label data can repair it to some degree).

In practice, according to the characteristics of raw data, we can choose an appropriate method from the Method Pool based on the advantages and disadvantages of all methods to obtain more accurate clustering results. Sometimes, a combined use of multiple methods may be promising to cluster very challenging data. For example, we can use Int-DCC-Cut (by observing whether there are some points slightly popping out in Decision Graph) to obtain a relatively good estimate of parameter $\sigma$ for Sup-Cut. Also, we can replace the referred feature $W_i$ by $W_i \times |P|$ in Sup-Cut, which may be helpful to terminate the cutting process in fewer steps in some applications. In fact, all the methods in the Method Pool can be implemented with a few lines of MATLAB code.
Fig. S1.  Principle of supervised cutting

Iteratively, the current longest edge is cut off, followed by a root finding process (blue block) to find the roots only for the supervised data, during which the graph structure is fixed, in contrast with the root finding process after cutting (red block) during which the roots for all data points should be identified and graph structure is updated (see step 6 in table S1). This iterative cutting is terminated when all supervised data with different labels have different roots.
Fig. S2. Principle of interactive cutting

Red cross records user’s interactive selection (e.g., pressing the mouse button beside the undesired edge). For all directed edges, if the start vertices first go across the red cross then to the end vertices, different deflection angles would be produced, e.g., $\alpha$ for edge (2,3) and $\beta$ for edge (1,2). Therefore, the edge identified and removed by computer is edge (2,3), the one with the smaller angle. The closer the red cross lies rightly beside the undesired edge (not the extension line), the smaller the deflection angle is, and thus the more accurate the judgment of computer is.
Fig. S3. An illustration for root finding phase

After cutting process, all sub-graphs are still in-trees, where every vertex has one and only one directed path to reach its corresponding root. Since all vertices can search for their root vertices in parallel, the total number of steps needed in root finding phase equals the number of steps needed for the furthest vertex to reach its root. Suppose the directed path in (A) is the longest one (with the max edge number, $H = 8$) in all sub-graphs, since the total shifting distances (in vertex number) for the farthest vertex “9” in (B–D) is a geometric sequence (respectively are 2, 4, 8 with the common ratio $q = 2$), thus, it needs $S = \lceil \log_2(H) \rceil$ steps for all vertices to reach their root vertices in root finding phase. “$\lceil x \rceil$” denotes the smallest integer not less than $x$. This is well validated in table S2.
Table S2. Analysis of $H$ and $S$ for root finding in different graphs

$H$ is the max number of edges of all directed paths in the graph after cutting and $S$ denotes the number of steps required for root finding. The results in this table show that $S$ and $H$ for each data set indeed meet with the relationship obtained in fig. S3. Note that in practice, the computations of $H$ and $S$ are not needed, because they are not relevant to the iteration termination condition. See details in step 6 of table S1. Computer used here is Intel Core i5, 3GHz with 8GB RAM. For mushroom data ($N = 8124$), the computation time for finding root is almost negligible (0.001s) and in fact the total time for step 2 ~ 6 is in several seconds if the supervised cutting method is chosen in step 5. $N$: number of data vectors; $n$: number of supervised data; $t$: the total time cost (in second) during root finding phase. Dim: dimension of data; Attr: attribute ("real" or "char"); char: character. Real data sets are denoted by "*". The remaining are synthetic data sets.

| Cutting method | Data set | Figure | Attr | Dim | $N$ | $H$ | $S$ | $t$ |
|----------------|----------|--------|------|-----|-----|-----|-----|-----|
| Interactive    | S4       | Fig. 2A| real | 2   | 5000| 15  | 4   | 0.0008 |
|                | SCI-14   | fig. S4A| real | 2   | 600 | 14  | 4   | 0.0002 |
|                | Aggregation | fig. S4B| real | 2   | 788 | 16  | 4   | 0.0002 |
|                | Compound | fig. S4C| real | 2   | 399 | 31  | 5   | 0.0002 |
|                | D31      | fig. S4D| real | 2   | 3100| 20  | 5   | 0.0006 |
|                | Flame    | fig. S4E| real | 2   | 240 | 20  | 5   | 0.0001 |
|                | Jain     | fig. S4F| real | 2   | 373 | 43  | 6   | 0.0002 |
|                | S1       | fig. S4G| real | 2   | 5000| 15  | 4   | 0.0006 |
|                | Spiral   | fig. S4H| real | 2   | 600 | 14  | 4   | 0.0002 |
|                | Mushroom* (transformed) | fig. S7| real | 2   | 8124| 165 | 8   | 0.002 |
| Supervised     | SCI-14 (n = 20) | Fig. 2D| real | 2   | 2000| 48  | 6   | 0.0002 |
|                | Mushroom* (n = 200) | Fig. 3| char | 22  | 8124| 12  | 4   | 0.001 |
|                | Face data* (n = 120) | Fig. 4| real | 10304| 400 | 4   | 2   | 0.002 |
Fig. S4. Interactively clustering different shapes of synthetic data
The first column of (A–H) lists the in-trees constructed for the data from (3), (5), (6), (7), (8), (9), (10), (11), respectively. The original data in (B–H) are available at http://cs.joensuu.fi/sipu/datasets. Red crosses record the places where the user clicks the mouse. The second column lists the results after cutting the edges identified by the red crosses. The last column lists the corresponding clustering results after searching for the roots (denoted by black circles) for all points based on the graphs shown in the second column.
Fig. S5. Interactive cutting with different values of parameter $\sigma$

For the data set (5), we test a large range of values of $\sigma$: 0.001, 1, 1000. For all cases, we can always interactively cut off the undesired edges based on the visualized data structure without too much effort. This also applies to other data sets in fig. S4.
Fig. S6. Interactively clustering the transformed Mushroom Dataset.
Data points in (A) are the result after transforming the original mushroom data from high- to two-dimensional space by non-linear dimensionality reduction method \( K\text{-ISOMAP} \ (K = 90) \) (12). Unlike linear dimension methods such as multidimensional scaling (MDS) and principal component analysis (PCA), \( K\text{-ISOMAP} \) can maximally preserve the underlying non-linear structure in the original data set, as shown in (A) where several clusters can be distinguishable after reducing the dimensionality. Based on the interactive cutting (denoted by the red crosses), these clusters can also be easily identified by our clustering algorithm as shown in (B).
Fig. S7. Robustness of supervised cutting to the value of parameter

A comparison is made between *Science*-14 (3) (A and B) and our supervised clustering (C). (A) Decision Graphs with different $\sigma$ (from left to right, $\sigma = 10^i$, $i = -4, -2, -1, 0$). $\sigma$ is the parameter of kernel function used to compute the density feature in *Science*-14. Colored points in the red boxes are selected as cluster centers. (B) the corresponding clustering results. Five well clustered groups are obtained only for the first two panels, corresponding to the salient Decision Graphs (having some pop-out points) in (A). However, the clustering results in (C) by our supervised method under the supervision of the same data as Fig.2D can always identify 5 main dense groups. Though many small isolated clusters exist in our results (each black circle represents a cluster center), they just occur in marginal areas, thus making negligible influence on the dense parts. Actually, for this data, our method can identify those 5 main dense groups with any $\sigma$. 
(A) The clustering results with different parameter $\sigma$, all under the supervision of 800 randomly selected labeled data points. For most $\sigma$, the average cluster number is around 3 and the error rate is almost 0, all with very small standard deviation over 20 different tests. The performance is almost perfect considering that this data set is of 2 classes (i.e., edible or poisonous). The results change very little, as shown in (B), after randomly permuting the original data, indicating that data indexes exert little influence on the results. This works even for the 2 smallest $\sigma$ for which all vertices are with the same potentials as shown in (C). **Abbreviation:** ANVDP, number of vertices with different potentials;
Fig. S9. Clustering faces by supervised cutting ($\sigma = 100$)

Faces (originally in gray-scale) assigned in the same clusters are dyed in the same colors. In particular, each of the faces labeled with yellow points is wrongly assigned to one of the other subjects.
Table S3 Clustering faces by supervised cutting with different $\sigma$ values.

For this data set, $N = 400$ and the average Euclidian distance is 5578. It shows that for different $\sigma$ values, the clustering results are quite comparable, even if the arrangements of the original data are randomly permuted. **Abbreviation:** BM, before merging (clustering results before merging the singleton clusters); AM, after merging (clustering results after merging the singleton clusters); NVDP, number of vertices with different potentials; CN, cluster number; NFAF, number of falsely assigned faces.

| Test | Parameter ($\sigma$) | NVDP | Clustering result | Before permutation | After permutation |
|------|----------------------|------|-------------------|-------------------|------------------|
| 1    | 100                  | 354  | CN 99 58          | BM 99 AM 58       | BM 99 AM 58      |
|      |                      |      | NFAF 3 10         | BM 3 AM 12        | BM 3 AM 12       |
| 2    | 200                  | 400  | CN 120 62         | BM 120 AM 62      | BM 120 AM 62     |
|      |                      |      | NFAF 3 10         | BM 3 AM 10        | BM 3 AM 10       |
| 3    | 1000                 | 400  | CN 110 61         | BM 110 AM 61      | BM 110 AM 61     |
|      |                      |      | NFAF 3 13         | BM 3 AM 13        | BM 3 AM 13       |
| 4    | 2000                 | 400  | CN 101 58         | BM 101 AM 58      | BM 101 AM 58     |
|      |                      |      | NFAF 3 14         | BM 3 AM 14        | BM 3 AM 14       |
Fig. S10. Clustering faces with different numbers of supervised data
The red and blue plots, the element number (in decreasing order) versus cluster \( n \), correspond to the clustering results supervised by 3 and 2 labeled data point in each subject, respectively. The two comparable plots indicate that when the supervised data in each subject is reduced from 3 to 2, the result changes a little.
Fig. S11.  Minimal spanning trees of two data sets
Compared with the results of our method in fig. S4E and Fig. 2A, the undesired edges in these two graphs are much less salient.
Fig. S12. Method pool based on In-tree structure

(A) The framework of Science-14. Two methods are optional for determining cluster centers: Int-DCC, interactively determining cluster centers based on the scatterplot of variable $\rho$ and $\delta$ for all points; K-DCC, determining $K$ (prespecified) cluster centers based on the variable $\rho_i \times \delta_i$. Abbreviations: FGM, find group members (based on the neighborhood relationship R).

(B) The framework of our work. After constructing the in-tree structure, three methods are optional for cutting the undesired edges: Int-Cut, interactive cutting; Sup-Cut, supervised cutting; K-Cut, cutting the $K$ (prespecified) longest edges; Variables involved: $W$, edge weights; $P$, potentials of vertices; $I$, edges of in-tree; $Id$, identities (or indexes) of vertices; $L$, labels of supervised vertices. Abbreviations: FR, find root. (C) A clustering “Method Pool” containing five methods for cutting undesired edges. Method 1 and 2 are two variants from the methods in (A). Int-DCC-Cut, cut directed edges that start from the cluster centers interactively determined with reference to the scatterplot of $W$ and $|P|$; K-DCC-Cut, cutting $K$ (prespecified) edges that start from the cluster centers determined by referring to variable $W_i \times |P_i|$. 3~5 are three methods in (B). Process in the red block represents the step of root constructing, blue blocks denote different cutting methods, and purple blocks correspond to the same root finding method. (D~F) “Decision graphs” corresponding to Int-DCC-Cut method on Mushroom Dataset (D and E) and News Dataset (F). Like Int-DCC, a salient image emerged in (D) and after interactively selecting (red box) the pop-out points (blue), 23 clusters was obtained with no cluster error. However, when the parameter was not ideal (E) or when the real dataset is very complex as 20 news dataset (7505 test documents, each transformed by us with TF-IDF values) in (F) or face dataset mentioned in Science-14, either method Int-Dcc-Cut or Int-DCC will be helpless to clustering.
Take data set in Fig. S4G ($N = 5000$) for instance. In order to show the effect of data index, we should first produce lots of vertices with identical potentials, since only for these vertices, their data index may serve as a reference for identifying transfer points. We set $\sigma$ as 10, much less than the average pair-wise distance ($4.3316 \times 10^5$). Consequently, only 151 potential values are unique, which means that most vertices are with the same potentials. We randomly changed the data index for 50 times. At each time, we just set the cluster number as 15, i.e., using the last cutting method (K-Cut) in step 4 of Table S1, and the largest 14 edges for the in-tree were cut off. Two of the 50 runs are shown here: respectively, (A) and (B) represent the in-trees and (C) and (D) represent the corresponding clustering results after cutting. It is clear that for the edge points in the red rectangles of (A) and (B), local difference occurs in their neighborhood relationships, whereas this difference makes very little effect on the final clustering results. In fact, (i) for the in-trees of all 50 tests, the longest 100 edges are the same; (ii) for the clustering results of all 50 tests, if we take the first clustering result as the benchmark to compare with the clustering results of other 49 tests, relative clustering error rate were very small, $0.0074 \pm 0.0041$ (mean $\pm$ SD), which means that, on average, 37 points out of 5000 are in different assignments. To summarize, although step 4 of Table S1 involves data index item, negligible influence occurs on the final clustering result.
Fig. S14. An illustration for step 4 in table S1
Consider five points from certain data set with $R_1 > R_2 = R_3 = R_4 > R_5$, where point 4 is overlapped with point 3. The computation of the transfer points (directed neighbors) of all points are described in table S4. Since point 3 has smaller data index than point 4, point 3 serve as a representative to build up linkages with other neighboring points (e.g., point 1 and 5). It is quite meaningful to define efficient rules to identify the transfer points for the points with same potentials, especially for the overlapped cases which may quite often occur for the massive number of data in practice. For example, suppose point 4 is connected to point 5 instead of point 3. Then the edge between point 4 and 5 would more likely be cut off and point 4 would become an singleton cluster. As the number of such overlapped points increases, the number of such singleton clusters also increases. However, unlike outliers, these singleton clusters should not be simply discarded and thus additional operations would be required to merge them into other groups. The method proposed here, however, can avoid the above operations, since point 4 is connected to point 3 and the edge connecting them is as small as 0 in length and thus will not be cut off.
Table S4. Determining the transfer points and connection weights in fig. S14

We define $J_i = K_i \cup L_i$, $K_i = \{ j \mid C_y > 0 \}$, $L_i = \{ j \mid C_y = 0 \land j > i \}$.

| $i$ | $K$ | $L$ | $J_i$ | $I_i = \min_{j \in \{2,3,4,5\}} \min_j D_{i,j}$ | $W_i$ |
|-----|-----|-----|-----|---------------------------------|-----|
| 1   | $\{2,3,4,5\}$; $L = \emptyset$ | then $J_1 = \{2,3,4,5\} \neq \emptyset$ | $I_1 = \min_{j \in \{2,3,4,5\}} \min_j D_{1,j}$ | $\min \{3,4\} = 3$ | $W_1 = D_{1,3}$ |
| 2   | $\{5\}$; $L = \emptyset$ | then $J_2 = \{5\} \neq \emptyset$ | $I_2 = \min_{j \in \{5\}} \min_j D_{2,j}$ | $\min \{5\} = 5$ | $W_2 = D_{2,5}$ |
| 3   | $\{5\}$; $L = \{2\}$; then $J_3 = \{2,5\} \neq \emptyset$ | $I_3 = \min_{j \in \{2,5\}} \min_j D_{3,j}$ | $\min \{5\} = 5$ | $W_3 = D_{3,5}$ |
| 4   | $\{5\}$; $L = \{2,3\}$; then $J_4 = \{2,3,5\} \neq \emptyset$ | $I_4 = \min_{j \in \{2,3,5\}} \min_j D_{4,j}$ | $\min \{3\} = 3$ | $W_4 = D_{4,3}$ |
| 5   | $\emptyset$; $L = \emptyset$ | then $J_5 = \emptyset$ | $I_5 = 5$ | $W_5 = -\infty$ |
References of supplementary material:
1. J. L. Gross, J. Yellen, *Handbook of Graph Theory*. (CRC press, Boca Raton, FL, 2004).
2. J. L. Gross, J. Yellen, *Graph Theory and Its Applications*. (CRC press, Boca Raton, FL, ed. 2, 2005).
3. A. Rodriguez, A. Laio, *Science* **344**, 1492 (2014).
4. http://qwone.com/~jason/20Newsgroups/
5. A. Gionis, H. Mannila, P. Tsaparas, *ACM Trans. Knowl. Discovery Data* **1**, 4 (2007).
6. C. T. Zahn, *IEEE Trans. Comput.* **100**, 68 (1971).
7. C. J. Veenman, M. J. T. Reinders, E. Backer, *IEEE Trans. Pattern Anal. Mach. Intell.* **24**, 1273 (2002).
8. L. Fu, E. Medico, *BMC Bioinf.* **8**, 3 (2007).
9. S. A. Greibach, *Lecture Notes in Computer Science*. (1975).
10. P. Fränti, O. Virmajoki, *Pattern Recognit.* **39**, 761 (2006).
11. H. Chang, D.-Y. Yeung, *Pattern Recognit.* **41**, 191 (2008).
12. J. B. Tenenbaum, V. De Silva, J. C. Langford, *Science* **290**, 2319 (2000).