Interactive Visual Clustering

Marie desJardins *James MacGlashan*
Department of CS&EE
Univ. of Maryland Baltimore County
Baltimore, MD 21250 USA
+1-410-455-3967
{mariedj, jmac1}@umbc.edu

Julia Ferraioli
Bryn Mawr College
101 North Merion Ave.
Bryn Mawr, PA 19010
+1-610-526-5358
jferraio@brynmawr.edu

ABSTRACT
Interactive Visual Clustering (IVC) is a novel method that allows a user to explore relational data sets interactively, in order to produce a clustering that satisfies their objectives. IVC combines spring-embedded graph layout with user interaction and constrained clustering. Experimental results on several synthetic and real-world data sets show that IVC yields better clustering performance than alternative methods.

ACM Classification: I2.6 [Artificial Intelligence]: Learning. H5.2 [Information interfaces and presentation]: Graphical user interfaces.

General terms: Algorithms, experimentation.

Keywords: Clustering, constraints, interaction.

MOTIVATION
The goal of this research is to develop interactive clustering methods, which allow a user to partition a data set into clusters that are appropriate for their interests. Traditional automated clustering partitions data into clusters with high intra-cluster similarity and low inter-cluster similarity. However, the "best" clusters may also depend on the user's goals. For example, given a collection of student data, an admissions officer may be looking for patterns in student performance, whereas a registrar might want to track enrollment patterns.

Constrained clustering allows users to provide partial knowledge about the nature of the clusters, typically in the form of pairwise constraints on cluster membership. Ideally, the user would provide a few initial constraints to "seed" the clusters, then add constraints as necessary to adjust and improve the resulting clusters. However, identifying useful constraints can be difficult, particularly in high-dimensional domains.

In some domains, there may be other relational information in addition to the pairwise constraints. These relations are generally weaker than constraints: they do not strictly imply shared cluster membership, although they may indicate a cluster correlation. Most clustering algorithms take into account either attribute or relational information, but not both.

Our goal is to allow a user to explore a large relational data set interactively, in order to produce a clustering that satisfies their objectives. We present a novel approach called Interactive Visual Clustering (IVC). In IVC, the relational data is initially displayed using a spring-embedded graph layout. The user can then move groups of instances together in order to form initial clusters. A constrained clustering algorithm is applied to combine the attribute information with the constraints implied by the instances that have been moved. The resulting clusters are then used to generate additional graph edges, leading to a new layout in which instances are relocated closer to the clusters to which they appear to belong. The user can then identify instances that are "misplaced" and move these instances into the correct clusters.

We show experimentally, using several synthetic and real-world data sets, that IVC converges to a target clustering significantly faster than either manual adjustment, spring-embedded layout alone, or clustering alone.

BACKGROUND
Our work incorporates force-directed graph layout and the PCK-Means [1] constrained clustering method.

We use a type of force-directed layout called *spring embedding* [3], as implemented in the Prefuse graph visualization system [9]. In spring embedding, nodes in a graph act on each other with two kinds of simulated forces: a *node repulsion force* emitted from each node and an attractive *spring force* that "pulls" along the edges between the nodes. The spring-embedded layout is determined iteratively, by computing and summing all of the forces on each node, then moving the nodes incrementally in the direction of the net resulting force. This "settling" process is repeated until the layout reaches an equilibrium.

Constrained clustering allows users to provide additional information about the nature of the clusters, typically in the form of pairwise constraints, indicating that two points should be in the same cluster (*must-link* constraints), or should be in different clusters (*cannot-link*). A popular standard clustering technique is the K-means algorithm [7], which iteratively assigns points to the nearest cluster, then recomputes cluster centroids, until a stable clustering is reached. Basu and Mooney [1] extended K-means to use a weighted penalty function for constraint violations. PCK-Means searches for the cluster assignment that maximizes cluster coherence while minimizing the penalty for constraint violations.

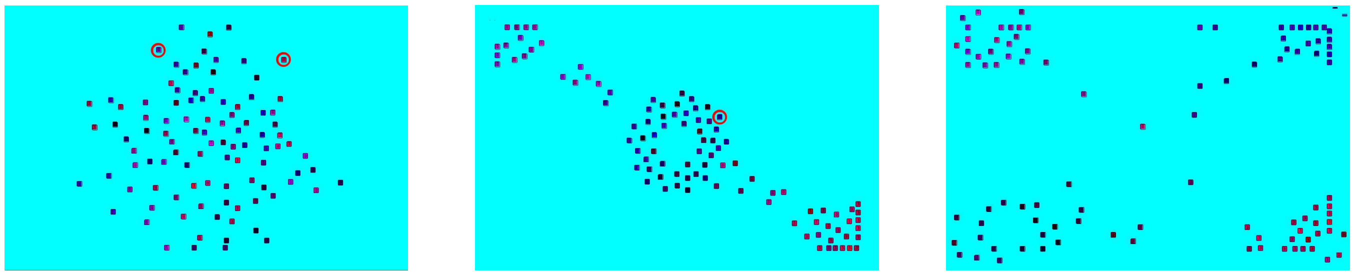


Figure 1: Overlapping Circles data set: (a) Initial display. The circled instances will be moved first. (b) After two instances have been moved. (c) After 14 instances have been moved.

APPROACH

Our visual clustering paradigm consists of four steps: (1) initialize the display (using Prefuse's spring embedding); (2) interpret user actions; (3) perform constrained clustering (using PCK-Means); and (4) update the display.

Interpreting User Actions. When the user moves an instance, it is "pinned" in place. The constrained clustering process begins after the user has moved two instances. To generate the constraints, the screen distance between each pair of moved instances is computed. If the instances are at least δ units apart (where δ is a user-adjustable parameter), they are considered to be in different clusters, and a *cannot-link* constraint is added between them. If they are less than ϵ units apart (where the parameter $\epsilon \leq \delta$), then they are considered to be in the same cluster, and a *must-link* constraint is added. If the screen distance is greater than ϵ but less than δ, the situation is ambiguous, and no constraints are generated.[1]

PCK-Means is then run on the data, using the constraints, and a new clustering is produced. Note that the distance metric for PCK-Means is Euclidean distance in the attribute space, *not* the screen distance used to generate constraints.

Updating the Display. After clustering, the display is updated to reflect the groupings inherent in the new clustering. We adapt an approach described by Brockenauer and Cornelson [2] for visualizing clusters in graphs. First, a new "dummy" node is generated to represent the center of each cluster. Next, a *cluster edge* is added between this cluster center and each instance assigned to that cluster. The relational edges use Prefuse's default spring constant (2.0×10^{-5}). Cluster edges are set to have a spring constant equal to twice the default (4.0×10^{-5}). The spring-embedded layout is then invoked on the combined graph. (Only the relational edges are shown to the user, and the cluster center nodes are not drawn.)

Simulating the User. We have not yet run formal experiments on human users. For the experiments reported here, we simulate user behavior using one of two heuristics for instance selection: *Random* and *Farthest-First*. The Random instance method selects a random instance to move at each step. Farthest-First selects the instance that is farthest (on the screen) from its correct cluster. The intuition behind the latter heuristic is that the user will be most likely to notice anomalous instances — that is, instances that appear farthest from where they should be. For both node heuristics, we use predefined locations (near the screen corners) for the cluster centers. In the experiments with force-directed layout, after each instance is moved, the layout is allowed to "settle" to an equilibrium before the next instance is moved.

[1]In our experiments, ϵ is set to 227 pixels, and δ is set to 450 pixels.

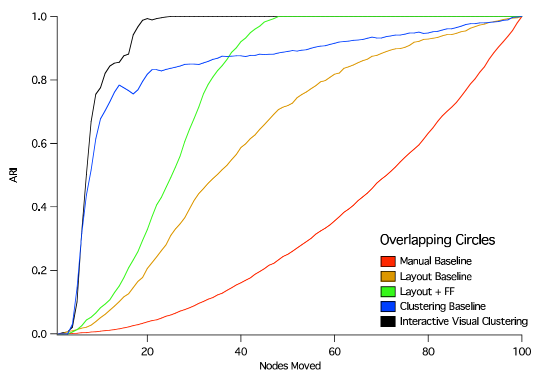
System Operation. Figure 1(a) shows the initial display of the synthetic Overlapping Circles data set (described in "Data Sets," below). For the purpose of illustrating the process, the colors of the nodes and numeric labels indicate the "true" (target) cluster membership. To simplify the displays, the relational edges are not shown. Notice that nodes from all of the clusters are interspersed in the display. The circled nodes in Figure 1(a) are the first two nodes chosen by the user. The resulting display is shown in Figure 1(b). Here, the upper left and lower right clusters (where the first two nodes were placed) are starting to become apparent. As more nodes are moved, the clusters gradually become more distinct. Figure 1(c) shows the display after 14 nodes have been moved. At this point, as seen in the results in Figure 2(a), most of the instances are grouped correctly into their target clusters. Visually, the clusters are very distinct, with only a few nodes scattered between the clusters.

METHODOLOGY

We compared five approaches: Manual Baseline, Layout Baseline, Layout+FF, Clustering Baseline, and Interactive Visual Clustering. The "Layout" methods and IVC use force-directed layout; in the others, only the nodes explicitly moved by the user are repositioned. The "Clustering" approaches apply constrained clustering after each instance is moved, and use the new cluster edges in the layout. The Layout+FF and Interactive Visual Clustering methods use the Farthest-First heuristic to select which node to move next; the other methods use the Random heuristic.

We hypothesize that the Farthest-First instance heuristic will improve performance faster than moving random instances; clustering will improve performance faster than without clustering; and force-directed layout will improve performance faster than manual layout.

To measure clustering performance, we use the Adjusted Rand Index (ARI). The ARI evaluates how close a given clustering is to the "correct" or target clustering, by comparing the proportion of clustering matches (i.e., pairs of instances that are grouped together in both the learned and the target clustering, or grouped separately in both the learned and the



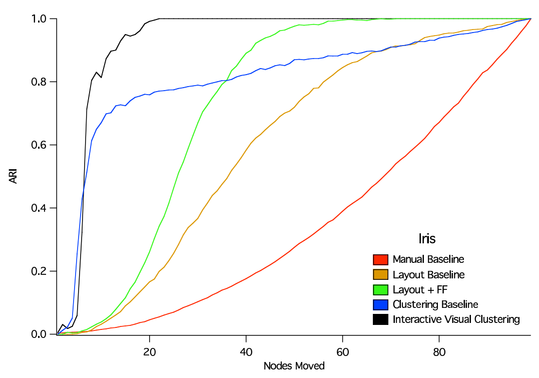


Figure 2: (a) Results on the Overlapping Circles data set. (b) Results on the Iris data set.

target clustering) to the expected number of matches [4]. An ARI of 1 means that all instances are correctly clustered. In the results, clustering performance is always shown as a function of the number of instances moved. Both the layout and the cluster assignments use a random initialization step, so we show average performance over 20 runs.

Data Sets

We used five data sets, as described below. We tested several methods of edge generation for the synthetic and Iris data sets. Space does not permit us to describe these methods in detail; the results are given for the version of each data set that yielded the best performance for each method.

Circles. The synthetic Circles data set includes 120 instances in two distinct clusters, generated by positioning circles of radius 50 at [50,50] and [150,150] on the (x, y) plane. Fifty points are randomly selected from inside each circle, and assigned to the corresponding cluster. Twenty additional "outlier" instances are generated by randomly sampling between the bounding circles. These outliers are then assigned to the nearest cluster.

Overlapping Circles. The synthetic Overlapping Circles data set includes 100 instances in four overlapping clusters. This data is generated by creating random points from a uniform distribution within the radius of four overlapping circles—corresponding to the four clusters—whose centers lie on another circle's radius at each 45-degree mark.

Iris. The Iris data set is a widely used classification database from the UC Irvine Machine Learning Repository [8]. The original data set consists of 150 instances; we chose 33 instances randomly from each of the three clusters. Each instance is described by four numeric attributes. The clusters correspond to three different species of irises. This data set is known to be a difficult one for most clustering algorithms, because two of the classes are linearly separable from each other, but the third is not.

Amino Acid Indices. The amino acid data set is a subset of the AAIndex Version 6.0 database [5], which consists of 494 *indices*, each of which measures a chemical property of amino acids. Each instance in the AAIndex database has twenty attributes, corresponding to the values of this index for each of the twenty amino acids in the standard genetic code. Tomii and Kanehisa [10] identified six clusters of indices in AAIndex. We use 100 indices, selected randomly from two of the clusters (A (measures of alpha and turn propensities) and H (hydrophobicity)). Edges in this data set were determined by measuring the correlations between the instances, then finding a minimum spanning tree.

Amino Acid. In this data set, the attributes and instances are inverted from the Amino Acid Index data set. The Amino Acid data set includes twenty instances—one for each amino acid—whose attributes are twenty-five of the 100 indices from the Amino Acid data set. Since many of the indices are variations of the same basic measurement, we asked a domain expert to select 25 indices that measured relatively independent properties. Edges were added to the data set based on three properties of amino acids: acidic side chains, basic side chains, and cyclic hydrocarbons. Edges were placed between pairs of instances that share one or more of these properties. The target clustering has three clusters, also manually identified by our domain expert: polar (asymmetrical electron charge on the side chain), non-polar (symmetrical electron charge), and both (long side chains with both polar and non-polar regions).

RESULTS AND DISCUSSION

Overall, our experimental results support our claim—that Interactive Visual Clustering provides improved clustering performance, compared to the alternative approaches we tested. However, the Amino Acid Index data set does not yield the expected results, highlighting some of the open challenges.

Circles. The results on this data set (omitted for space) are as predicted. Manually moving the instances shows the slowest improvement as a function of the number of instances moved; the Layout Baseline shows significant improvement; Layout+FF is still better; and IVC performs the best. However, the clustering baseline yields nearly identical performance to IVC. This is not surprising: the instances are well separated, making this a fairly easy clustering problem.

Overlapping Circles. Figure 2(a) shows the experimental results for the Overlapping Circles data set. Again, the methods perform as expected, with IVC outperforming the other methods. In this case, IVC does provide a noticeable improvement beyond the Clustering Baseline, indicating that

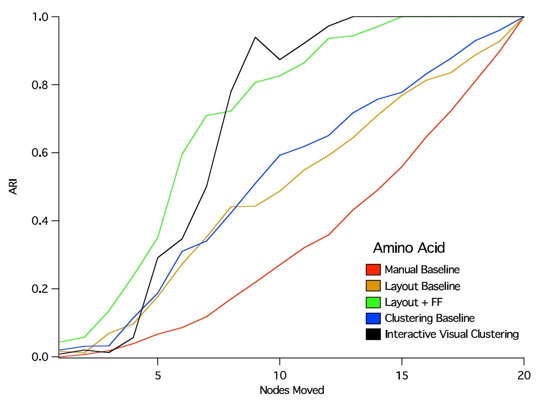


Figure 3: Results on the Amino Acid data set.

the Farthest-First heuristic is helpful in identifying important instances for repairing the clustering.

Iris. As seen in Figure 2(b), IVC also yields the best performance of any of the methods we tested on the Iris data set. The improvement provided by IVC is significant: after only 10 instances, with IVC, the clusters are nearly perfect, with an ARI close to 1.0. The next-best method (Clustering Baseline) has only reached an ARI of 0.75 at this point.

Amino Acid Indices. In the Amino Acid Indices data set (omitted for space), clustering does not help, and actually appears to hinder performance. The best performance is given by Layout + FF. IVC is only slightly better than the Clustering Baseline and the Layout Baseline. IVC also shows much more variability than the other methods: it appears that for this data set, slight variations in the layout (resulting in different selected nodes) yield significantly different clusters. Also, the clusters in this data set are not well separated in Euclidean space. Therefore, the underlying assumptions of the clustering method are violated. This observation led us to develop the alternative (Amino Acid) data set.

Amino Acid. The results for the Amino Acid data set are shown in Figure 3. IVC outperforms most of the other methods, but the Layout + FF approach is comparable. The latter method slightly outperforms IVC when only a few nodes have been moved, but IVC is slightly better for more nodes; however, these differences are not statistically significant. The Clustering Baseline and Layout Baseline perform about equally, both outperforming the Manual Baseline.

We conclude that force-directed layout and Farthest-First help the user to find the desired clustering. However, the clustering itself does not provide an additional benefit. Again, in this data set, the Euclidean distances between instances are not strongly related to the true clustering.

RELATED WORK

Lesh *et al.* [6] also present an interactive clustering method that used force-directed layout. However, their underlying clustering method is purely graph-based, not attribute-based. Also, rather than using constrained clustering, their approach uses the modified clusters produced by the user as seeds for local heuristic search. Their results do show that similar interactive approaches may be useful even for much larger data sets than we have studied. In the constrained clustering literature, there has been some work on active (automatic) selection of constraints, but we are not aware of any previous work on interactive methods for enabling the user to select appropriate constraints more effectively.

FUTURE WORK AND CONCLUSIONS

We have shown that IVC can improve clustering performance by integrating force-directed layout with user interaction and constrained clustering. IVC is only the first step towards a more user-centered, relational approach to clustering. We are currently designing a user study to test the hypothesis that users will be able to identify anomalous (misplaced) instances in the display, and therefore converge more quickly to the correct clustering than without force-directed layout. We are also developing relational constrained clustering algorithms, which cluster the data in attribute space and relational space simultaneously.

ACKNOWLEDGEMENTS

Thanks to Adam Anthony, Blaz Bulka, Donald MacGlashan, and Penny Rheingans for their assistance and inputs. This work was partially supported by NSF award #0325329 and #00000923, and by the Distributed Mentor Project (DMP) of the Computing Research Association's Committee on the Status of Women in Computing Research (CRA-W).

REFERENCES

1. S. Basu, A. Banerjee, and R. Mooney. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, pages 333–344, April 2004.

2. R. Brockenauer and S. Cornelsen. Drawing clusters and hierarchies. In Michael Kaufmann and Dorothea Wagner, editors, *Drawing Graphs: Methods and Models*, pages 193–227. Springer, 2001.

3. P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

4. L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1988.

5. S. Kawashima and M. Kanehisa. AAindex: Amino acid index database. *Nuc. Acids Res.*, 28(1):374, 2000.

6. N. Lesh, J. Marks, and M. Patrignani. Interactive partitioning. In *International Symposium on Graph Drawing*, pages 31–36, 2000.

7. J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Symposium on Math, Statistics, and Probability*, volume 1, pages 281–297, 1967.

8. D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases, 1998.

9. prefuse.org. Prefuse: Interactive information visualization toolkit, 2006.

10. K. Tomii and M. Kanehisa. Analysis of amino acid indices and mutation matrices for sequence comparison and structure prediction of proteins. *Protein Engineering*, 9:27–36, 1996.