SHIM: Semantic Hierarchical clustering with Interactive Machine learning

Fang Cao* Yuanwei Tu† Eli T. Brown‡

DePaul University

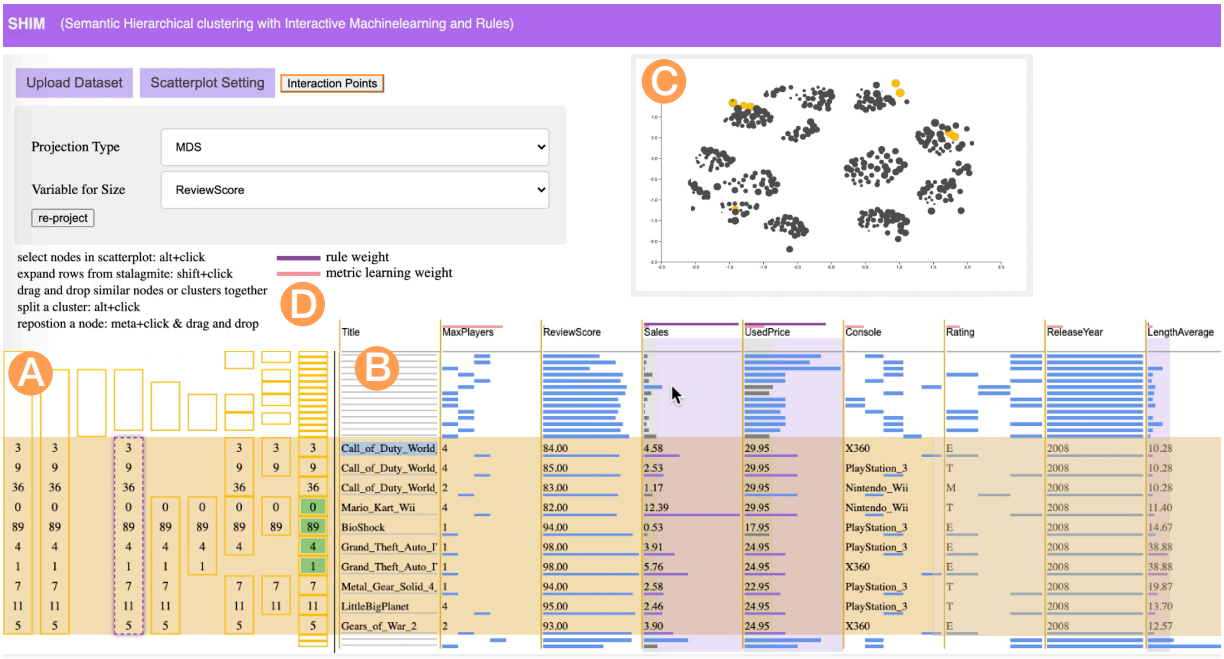


Figure 1: There are three main components of the SHIM prototype: (A) *stalagmite* visualization of hierarchical clusters providing several interaction types (B) associated table view to show data detail and feedback from the machine learning back-end, and (C) a scatterplot showing relationships between points in projected space. The (D) marks the legend for the bars that show the importance score of columns based on the rule learner and the metric learning respectively.

Abstract

Clustering is widely used, and there are available techniques for using semantic interactions to incorporate human feedback. Hierarchical agglomerative clustering produces complete substructure, offering rich possibilities for user interactions. However, currently available human-in-the-loop techniques miss out on the advantages of hierarchical agglomerative clustering. In this paper, we present a technique, SHIM, for semantic, interactive hierarchical clustering that helps users understand their data through exploring and modifying clusters and cluster substructure with semantic interactions and automatic descriptive rules. The proposed solution includes a compact visualization of hierarchical clustering customized for this context and connected to a table view that shows data detail and descriptive rules simultaneously. We assess our design decision and usability with expert feedback, and use simulations to test whether the metric learning method will improve a model with limited interactions.

*e-mail: fcao1@depaul.edu
†e-mail: ytu4@depaul.edu
‡e-mail: eli.t.brown@depaul.edu

1 Introduction

Clustering is one of the core types of machine learning algorithms, used across every variety of data mining application. A growing number of people need to take advantage of clustering analysis, but advanced skills are required and gaining proficiency is difficult and time-consuming. One approach is a semantic interaction, [20] human-in-the-loop system, where the user interacts with a visual system, engaging with their data through semantics that are reasonable for their domain. Behind the scenes, machine learning is used to power the analytic capabilities. These systems aim to get the best of both human and machine by playing to their strengths: machines for raw number crunching power, humans for real sense-making and insight. Clustering has been addressed in this context (e.g., [7, 13, 19, 32]), however, previous work has missed out on the opportunities afforded by hierarchical clustering (typically *hierarchical agglomerative clustering*, or HAC).

The advantages of HAC include complex cluster boundaries with detailed substructures for a fuller picture of the data groupings. However, current human-in-the-loop techniques have not fully taken advantage of HAC. First, due to the substructure, there is a potential to investigate a richer set of interaction mechanisms beyond the assignment of each data item to a cluster. SHIM supports interactions at any level of HAC clusters and adapts those interactions into metric learning to further improve the user-based model. Second, due to the complex cluster boundaries, tools to assist the user's understanding are crucial. We employ automatic descriptive rules to provide interactive visualizations of what data attributes are im-

portant to groups and what ranges of those attributes matter most. Third, HAC can be visualized through how leaf nodes are merged into one whole cluster. In this process, there are many sub-clusters formed. Current visualizations of HAC, especially dendrograms, are not organized to facilitate interaction with all levels of the clustering. We need to make sure that users can see not only the big picture of the clustering but also sub-clusters with data entity information. SHIM's stalagmite (detailed in Sect. 3.2) meets this requirement. Users can see a compact clustering with each cluster as a box that can be expanded at any level of HAC to see the detailed information through corresponding table rows as shown in Fig. 1 A and B.

In this paper, we present a semantic interaction technique that helps users develop and understand subsets of interest by directly interacting with sub-clusters (even leaf level nodes) to tune a clustering based on their preferences. The flexibility in interaction types provided by hierarchical clustering's substructure is implemented by creating metric learning constraints for each type of interaction. Our stalagmite visualization (Sect. 3.2) associated with an interactive data table with descriptive rules facilitates pattern discovery while keeping overview and detail tightly integrated. The integration between the table and stalagmite features the ability to quickly see which attributes are important for explaining a given cluster, and see which clusters are most affected by a selected attribute. A customizable scatterplot helps users (1) see an overview of the dataset through different projections and (2) further project several attributes learned from descriptive rules based on the selected sub-cluster. We detail the technique, the available interactions, and how they are translated for the metric learning, plus the integration with rule learning and other visualization features. A prototype implementation supports a usage scenario demonstrating the effectiveness of the workflow on real-world data. Machine learning experiments demonstrate that a small number of interactions can result in a quality model. Finally, we provide feedback from visualization experts on the design and usability of the prototype as an illustration of the feasibility of the technique.

2 Related Work

Finding structure in data is a key component to gaining an understanding of data (sensemaking). Because of this need, clustering is a common, well-studied tool in machine learning and in visual analytics. Where machine learning and visualization come together, in human-in-the-loop (HIL) analytics[1], researchers develop techniques where one may use the machine learning methods without controlling the model-building algorithms directly. HIL systems have been described in several frameworks [8, 22, 27, 41, 42].

A subset of HIL is using *semantic interactions* [21] for *modelsteering*: the user interacts with a system designed with visualizations and interactivity suited to their domain knowledge (even if the domain is fairly general data analysis) and behind the scenes a model is built using algorithmic machine learning tools. The interactions are converted to machine learning inputs and when a model is trained, the result is used to update the interface so the user may benefit. This iterative refinement of the model through continued interactions is what closes the *loop* of human-in-the-loop.

2.1 Clustering and Interactive Clustering

SHIM uses semantic interactions to help users make sense of the structure of groups of similar items in their data. Semantic interaction has been applied to the task of clustering before. Examples include iCluster [7], which provides a grouping tool that uses a learned distance function to make suggestions for points to add, and ClusterSculptor [33], which iteratively tunes parameters to derive a classification hierarchy. Another with nearly the same name, Cluster Sculptor [10] allows interactively updating cluster labels, and Clustrophile 2 [12] helps choose clustering parameters and compare results. None of these uses hierarchical clustering, so they do not take advantage of substructures in the clusters.

On the machine learning side of SHIM, we need a clustering algorithm that can accept user feedback. Since clustering is unsupervised (no feedback), we need a semi-supervised version, of which several (see survey [5]) are available. Additional knowledge can be introduced by constraints, as in Ankerst, et al. [3], and in Wagstaff, et al. [44]. Okabe and Yamada [35] introduced a tool for human active learning in constraint clustering. In addition to constraints assignment, the tool provides functions including incremental distance metric learning. In their work, Lu, et al. [30], built constrained clustering based on expectation maximization. However, none of this work is for hierarchical clustering.

Because hierarchical clustering works based on a set of distance measurements between all points (pairwise distances), instead of changing the algorithm itself, we can simply change the distances. Our solution, explained in Sect. 3, uses metric learning (for a survey see [46]) to control the hierarchical clustering. There are several varieties of metric learning algorithms, but a common approach is to input constraints instead of labels. Rather than specify a label for a given point, we specify pairs of points that should be close together (must-link constraints) and pairs that should be separated (cannot-link constraints). Furthermore, user interactions can be translated into these metric learning constraints. In respect of semantic clustering interactions, [45] taxonomizes interactions and states the challenge of interpreting those interactions. Unlike their work focusing on projected clustering, we specify four main interactions for the agglomerative hierarchical clustering visualized with SHIMat both leaf node and group levels as shown in Fig. 3 One reason metric learning is popular in HIL systems is that it can be effective with a small number of user inputs. Examples include distance function learning on its own [9], clustering [7], and domain applications [2, 23].

2.2 Rule Learning

We also need machine learning to create automatic descriptions of clusters. There are several approaches to descriptive rules, including learning them directly from data [14, 15, 24], or building a separate model and then extracting them [1, 36, 43]. Based on the semantic interaction rule-learning in DRIL [11], we use Quinlan's C5.0 [36, 40], which learns a decision tree and then performs optimization steps to extract IF-THEN rules. Though this technique is intended for predictive modeling, it can be used for descriptive rules by creating labels that separate groups of interest (clusters) from other data, as explained in Sect. 3.3.

2.3 Related Interactive Visualizations

In SHIM, we need to visualize data, descriptive rules, and an HAC clustering in an integrated manner. Table Lens [37] applies a focus+context (fisheye) technique that fits tabular data well. One convenient aspect of Table Lens for our work is that the ranges of values in each data column are clearly represented. We take advantage of this to highlight ranges that are important based on the generated descriptive rules.

Another helpful aspect is that the table structure means every data point appears in a predictable spot that can be lined up with another visualization. Specifically, we need to align it with a hierarchical visualization, similar to John et al. [26], which integrated a data analysis step into Table Lens to show self-organizing maps and hierarchical clustering. We take inspiration from that as well as Icicle Plots [28], which are commonly used for hierarchical clustering. Unlike dendrograms, icicle plots have boxes to visualize each sub-cluster, so that users are able to interact with clusters at any

[1] Closely related terms used in various communities of HCI and machine learning include *interactive machine learning* and *human-centered machine learning*.

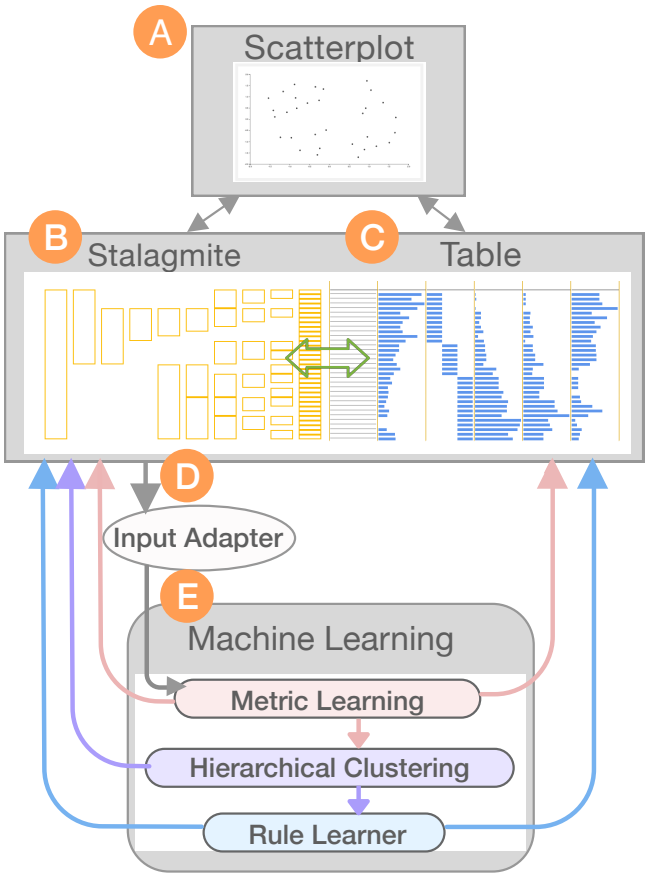


Figure 2: System structure: we have three main interface components, coordinated with each other and connected to a machine learning backend. The interactions from the interface are passed into an input adapter to translate them to metric learning constraints (gray arrow). The dissimilarity matrix learned from metric learning is into hierarchical clustering. The rule learner models descriptive rules for each cluster. The machine learning results from all components are visualized in stalagmite and the table for further exploration and interaction.

level of hierarchical clustering. Our *stalagmite* approach is related, but shows clusters growing from the bottom up, in contrast to the bottom-down icicle metaphor. This choice creates some open space in the visualization, but it means we can align individual points in the hierarchical data visualization with the table so the correspondence is always clear between the data detail view and the structural information of the clustering. More detail on the visualization is in Sect. 3.2.

3 Interactive Hierarchical Clustering for Sensemaking

Although hierarchical clustering is a common technique for producing groups (clusters) from raw data, especially the most common type, hierarchical agglomerative clustering (HAC), current human-in-the-loop techniques have not fully taken advantage of HAC. SHIM takes advantage of HAC to help users explore subsets of interest through a user-focused machine learning model by their iterative interactions. Following the diagram of Fig. 2, we will explain how SHIM helps users explore and understand how their interactions tune the clustering visualized through three main components closely connected to backend machine learning so that they can incrementally understand how their interactions improve the clustering.

Hierarchical clustering has advantages over other methods like k-means [29, 31] and EM [18] clustering, including: (1) modeling complex cluster shapes, (2) providing not only a clustering of data but a complete cluster substructure going all the way down to individual data points, and (3) operating with only pairwise distances between points.

The first item is helpful in general, but the latter two have particular importance for this work. Perhaps the most common type, hierarchical agglomerative clustering (HAC) [38] works by initializing each point as a cluster and then progressively merging the closest clusters, building up a few large clusters. This nested construction produces a subcluster-level structure, providing detail about relationships within clusters. Our work takes advantage of our visualization called stalagmite (detailed in Sect. 3.2) that shows the whole structure of how the subclusters merged, with tools to connect it directly to the data.

Second, the fact that only pairwise distances are necessary to compute HAC means that the main input to the algorithm can be controlled by a custom distance function. When it comes to incorporating incremental user feedback into machine learning, metric learning algorithms are popular because they can produce good results with small amounts of user labels. These metric learning algorithms can take user feedback and produce exactly what is needed to control HAC: a custom distance function.

The concept of this work is to take advantage of these properties to create an interactive hierarchical clustering experience that maintains the advantages of HAC in terms of complexity of modeling, but allows a rich set of semantic interactions. The concept uses clustering as a means of data exploration. Rather than assisting the user in perfecting a full clustering explicitly, we focus on finding and refining a group of interest. To achieve this, we rely on user interactions. Our technique provides flexible interactions at both subcluster level and leaf node level (see detail in Sect. 3.1) using our *stalagmite* interface (see detail in Sect. 3.2). SHIM also applies rule learning to explain the HAC subclusters, linking descriptions to interactive, coordinated visualizations (see detail in Sect. 3.4).

In Fig. 2, the diagram shows how the components of this technique are connected. The visualization components (scatterplot, stalagmite, and table) show the clustering tightly connected to the data (stalagmite to table), the descriptive rules (on the table), and the spatial relationships between groups of points (scatterplot) for overview. These interactive components are connected to a machine learning back-end via an input adapter (see detail in Sect. 3.3). The table in Fig. 3 shows the set of interactions and how the input adapter translates interactions to metric learning constraints. These constraints are passed to learn a new metric model, which provides a distance function that reflects user feedback. Based on the model, the back-end produces a new dissimilarity matrix to re-run the hierarchical clustering. With the new hierarchical clustering, the rule learner models descriptive rules for each cluster. As long as a cluster from HAC has more than one data entity, we learn descriptive rules algorithmically to provide explanations of how groups are distinct from the rest of the data with respect to particular attributes and their value ranges.

These models are all integrated back into the visualizations to show the clustering updated based on feedback, and new descriptive rules to reflect those changes (described in Sect. 4).

3.1 Flexible Interactions with Hierarchical Clustering

One advantage of hierarchical clustering and the substructure it produces for clusters is the possibility of a rich set of semantic interactions. Based on work describing a full set of interactions used in topic modeling to adjust hierarchical topics [25], we support four main interactions as shown in the table of Fig. 3. We assume users would like to find some subsets of points they are interested in. SHIM provides flexibility to interact directly on either a cluster or a

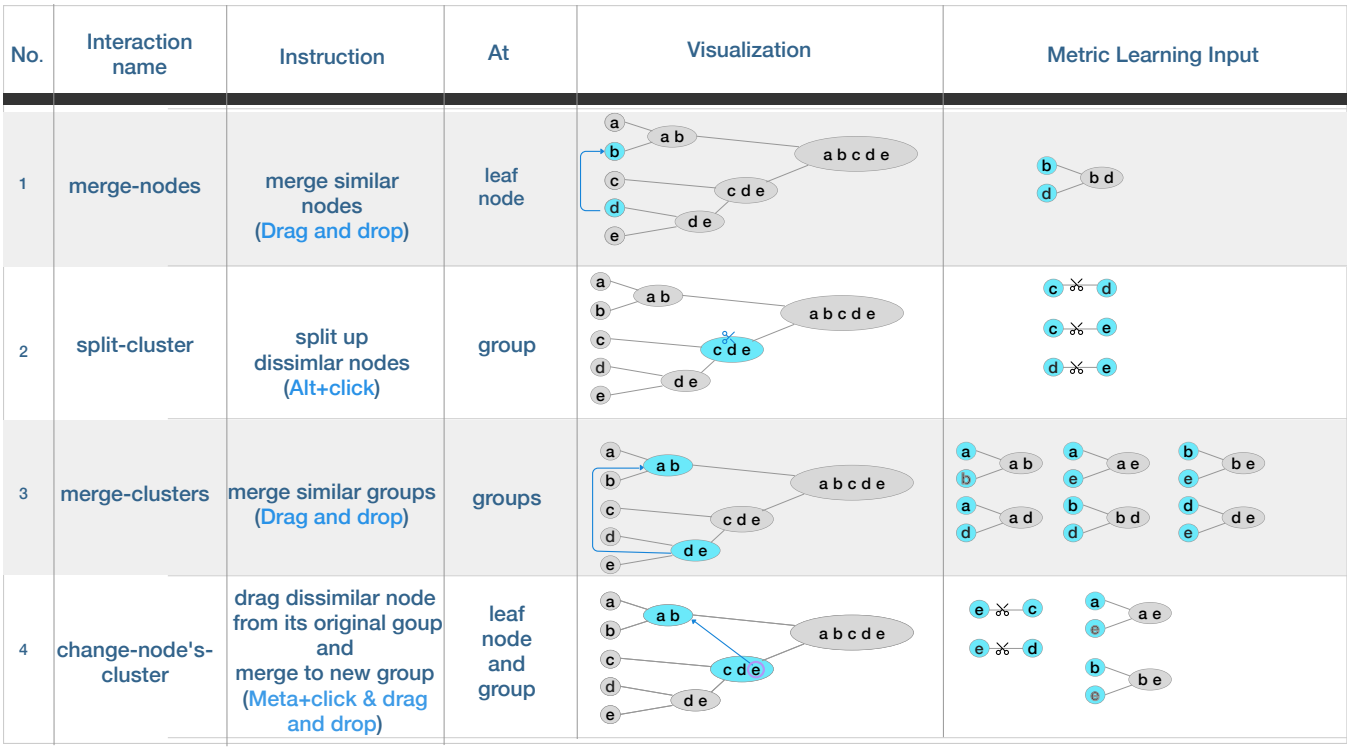


Figure 3: The interactions from the interface will be passed into an input adapter to translate them to metric learning constraints.

leaf node. Instead of using the child or siblings concept, as in Hoque, et al. [25], SHIM focuses on the groups of points in the clusters or leaf nodes that a user interacts with. Therefore, we assign metric learning constraints to all data entities in each interacted cluster. We directly translate the interactions into metric learning inputs through these transformations:

1. **Merge similar nodes** (*merge-nodes*): if two entities are closely related to each other, the user can merge similar nodes at the leaf level. This produces a must-link constraint between them.
2. **Split up dissimilar nodes** (*split-cluster*): if a cluster of entities should not be grouped together, the user can break it apart. This produces a cannot-link constraint between each pair of its nodes.
3. **Merge similar groups** (*merge-clusters*): if users consider one cluster similar to another, they can merge the two. This produces pairwise must-link constraints for items in the two groups.
4. **Move an item** (*change-node's-cluster*): if an item does not belong in its group, move it to another. This produces must link constraints between the item and the new cluster members and cannot-link constraints between the item and its old cluster.

After the constraints are generated and a new metric model is learned, the hierarchical clustering can be rerun, producing a new clustering that reflects the user's feedback.

3.2 Stalagmite Visualization

For this technique, we need a visualization of the hierarchical clustering that enables the user to interact with the subclusters to provide feedback. It must also be connected closely with the actual data for a detailed view. We propose *stalagmite*, which is related to an icicle plot, but essentially flipped, the way a stalagmite is a geological feature grown from dripping minerals but appearing to grow from the ground. In an icicle plot, the top level of the clustering, with all points merged together, is represented as a box along one edge of a rectangle. Layers of boxes follow, showing the substructure level by level of the hierarchy. However, the structure does not have a uniform depth, so leaf nodes of the hierarchy may be shown at multiple levels of the plot. This arrangement seems to emphasize the splitting of the whole rather than the merging from individuals. In our case, we want all the individual points and their merges emphasized and connected to our table view of the data. To solve this, we put all the leaf nodes together into the right-most column so that these nodes representing an individual data point are lined up next to the table view on the right. In that position, they can be coordinated with the table view that also maintains a lineup of data points (detailed in Sect. 4). We use a simplified version of TableLens [37] where reordering is not possible because it would affect the hierarchy.

The visualization of hierarchical clustering also must be compact, collapsible and expandable to match the fisheye effect of the table view, otherwise it would be impossible to see individual items. Stalagmite places a thin node representing each data point along one column, positioned to align with the data in the table. It shows the full history of how nodes were merged during clustering into larger groups with successive rectangles that cover the vertical bounds of all elements within them. For example, a cluster with two nodes covers the vertical space associated with those two points, but sits next to them. As the cluster grows in size, the rectangles grow to the right, but they are kept as far left as possible for compactness. Because we know the merge order of the data points, it is possible to put the nodes in order such that aligning the rectangles with the data points never conflicts.

3.3 Learning from Interactions

First, we suppose there is a set of n points and each point has m attributes $\{x_i\}_{i=1}^{n} \subseteq \mathbb{R}^m$ and users interact with those points on stalagmite with four main interactions as shown in the table of Fig. 3. The interactions from the interface will be passed into an input adapter to translate them to metric learning constraints. Metric learning uses optimization to learn a distance function (or metric) that takes into account external constraints. Common constraints include "these two points are similar" and "these two points are dissimilar", known as *must-link* and *cannot-link* constraints. Based on which type of interaction the user performed, different sets of constraints are constructed and passed to the machine learning. Here we provide details of these constraints, corresponding to Sect. 3.1 and Fig. 3. In all cases, we codify the constraints in sets S and D.

$$S = \{(x_i, x_j) | must\_link(x_i, x_j)\} \quad (1)$$

$$D = \{(x_i, x_j) | cannot\_link(x_i, x_j)\} \quad (2)$$

As shown in Fig. 3, for *Merge-nodes*, a user merges similar nodes by dragging and dropping a leaf node d close to another leaf node b, so the metric learning input, adding on to any existing constraints from previous interactions, will be

$$S = S \cup \{(b,d)\} \quad (3)$$

For *Split-cluster*, a user splits a sub-cluster that has data entities c, d, e. Cannot links are assigned to all the pairs of the data entities in the selected sub-cluster, so the metric learning input will be

$$D = D \cup \{(c,d),(c,e),(d,e)\} \quad (4)$$

For *Merge-clusters*, a user merges similar groups by dragging and dropping a sub-cluster (e.g., containing d and e) close to another sub-cluster (e.g., containing a and b), so the metric learning input will link all pairs across the two and all pairs within each original cluster:

$$S = S \cup \{(a,b),(a,d),(a,e),(b,d),(b,e),(d,e)\} \quad (5)$$

For *Change-node's-cluster*, a user drags a dissimilar node (e) from its original group (e.g., containing c, d, and e) and then merges it into a new group (e.g., containing a, b). So the metric learning input will be

$$S = S \cup \{(a,e),(b,e)\} \quad (6)$$

and

$$D = D \cup \{(e,c),(e,d)\} \quad (7)$$

Given these constraints, we use metric learning (ITML [16], see Sect. 6) to learn a distance metric, which is a function d_A giving the distance between points x_i and x_j. In this case, it takes the form of a Mahalanobis distance function, which is a linear distance parameterized by a learned matrix A.

$$d_A(x_i, x_j) = (x_i - x_j)^T A (x_i - x_j) \quad (8)$$

The dissimilarity matrix, A, learned from metric learning provides distances between all pairs of points, which is exactly the required input for hierarchical clustering. The new clustering will reflect the feedback the user provided with their interactions because it is dependent on the learned distances.

Further, the rule learner models descriptive rules for each new sub-cluster, completing the machine learning update based on the interactions. For every subcluster c, we create a labeling Y of all n data points such that $y_i = 1$ if $x_i \in c$. Thus for every subcluster, we use these labels in a rule-learning algorithm [39] to discover the descriptive rules that will be used to visualize the importance of variables and key ranges. The machine learning results will all be visualized into both stalagmite and the table for further exploration and interaction.

3.4 Connecting Visualizations and Interactions

As shown in Fig. 2, we have three main interfaces components connected to a machine learning backend. We will first explain how those components are connected and why we designed SHIM as a human-in-the-loop technique with those connections considering user needs.

Through stalagmite, users can see the structure of hierarchical clustering and explore data detail by expanding a subcluster (and even a leaf node) to highlight it with an orange box and display it in the associated table view as shown in Fig. 1 A and B. Both stalagmite and the table view are closely connected to the scatterplot. All the selected data entities in stalagmite and the table view will be highlighted in the scatterplot as well. Users can select a projection and customize the scatterplot based on their domain knowledge. Selection in the scatterplot, as shown in Fig. 1C, will be highlighted in both stalagmite and the table view accordingly.

We visualize machine learning results for users to easily understand and interact to tune the model further. Domain experts could assume one or a few attributes are important based on their domain knowledge. SHIM provides the capability of selecting attributes in the table to explore clustering. By selecting the attribute, the most related clusters will be shown by a heatmap in stalagmite, so that users can assume the darker colored clusters are more related to the selected attribute (see the Usage Scenario, step 14 in Fig. 4, for an example). This helps users focus on fewer but more related clusters to explore. When users are exploring subclusters, they are able to see visualized descriptive rules explaining why the clusters are formed in the table view. These rule learner results are visualized in two ways: (1) the rule lists are shown in the table view with the background colors of relevant columns (detailed in Sect. 4, example in Fig. 1B, the purple and gray highlighted ranges in the columns), and (2) attribute weights from rule learning will be shown as bars in the header of the table view, so that the user can see which attributes are highly important.

When a user has just provided a round of feedback to the backend, they will see an updated visualized clustering in stalagmite. Now that there is a metric learning model, SHIM shows its attribute weights as another set of bars in the table header. Rule attribute weights do not show again until a cluster is selected in stalagmite. In Fig. 1 B (legend in D) we see both together. In this way, users can see which attributes are most impacted in the metric learning based on their interactions. Metric learning returns a dissimilarity matrix, which is the learned input for hierarchical clustering. It also means the attribute weight change from metric learning is based on accumulated user interactions directly. Different from the rule weights based on a selected cluster, the metric learning weights are based on the dissimilarity matrix behind the hierarchical clustering.

4 The Prototype and Usage Scenario

4.1 Prototype

In this section, we explain our prototype implementation of SHIM. There are three main components as seen in Fig. 1: (A) the stalagmite hierarchical cluster view, (B) a table view, and (C) a scatterplot. These three components are closely connected. Selecting a cluster in stalagmite zooms the table so details of data entities in that group are available. Stalagmite and the table are aligned at the item level, even when expanded. The bars shown under the data values in the table make it possible to investigate data trends in the clusters. The scatterplot gets highlighted based on selections in stalagmite, and vice versa, which is helpful for visualizing the clusters in the context of the data. The user can choose to graph by data attributes or choose a projection (MDS, t-SNE, and UMAP are available), and size can be mapped to an additional attribute.

A user can directly interact with these three components to apply their knowledge to improve hierarchical clustering. The four interactions (detailed in Sect. 3.1 and enumerated in the table of Fig. 3)

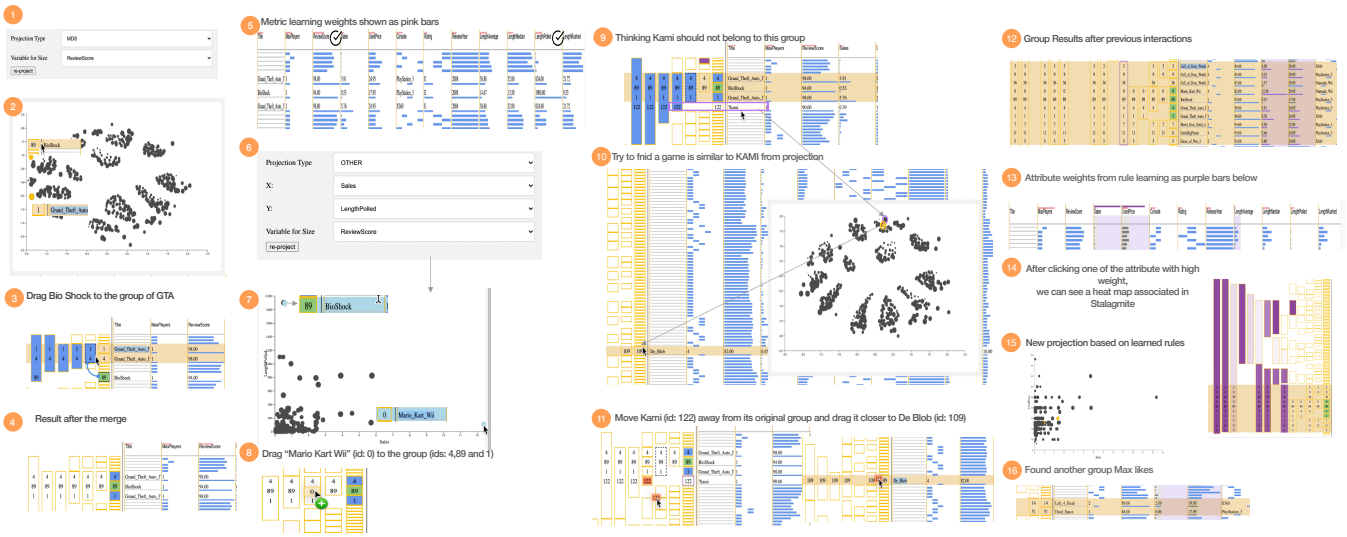


Figure 4: Usage Scenario - Max used SHIM to find a group or groups of games he is interested in. Also, he found some hidden rules of his preferences, which he can apply for his future game search. See Sect. 4

.

tell stalagmite to send metric learning input to the machine learning backend. For merging similar data nodes together, users can drag a cluster to a similar cluster at any hierarchical level (*Merge-nodes* and *Merge-clusters*). For separating dissimilar data, the user can split up a group (*Split-cluster*), and users can Meta+click to make each data point in a group draggable to move to a new group by *Change-node's-cluster*. For *Merge-nodes* and *Merge-clusters*, the dragged entity or group will be highlighted in green and the other entity or group will be highlighted in blue. For *Split-cluster*, the lines will be added to split up all the entities in that group. For *Change-node's-cluster*, the different entities will be highlighted in orange to separate from their original group, and the entities in the new group will be highlighted in blue. All the parent groups which contain those entities will be highlighted as well to show the structure of the changes.

Users may also use relationships between clusters and attributes via stalagmite and the table. As detailed in Sect. 3.4, the table shows the importance of each attribute in the metric learning model and in the rules for a selected cluster. The user can click an attribute to see a heatmap across stalagmite of the influence on clusters. These features simplify further exploration of the clusters of interest and their attributes.

4.2 Usage Scenario

We present a usage scenario with step numbers in Fig. 4 to show how a user can use the SHIM prototype to find a group of games based on user preferences. Max likes to play games and used the SHIM prototype to find a group or groups of games he wants to play. First, he uploaded a game dataset [6][2]. As a starting entry (step 1), he projected the data points with MDS and sized the nodes by "Review Score". In this way, he can explore some groups projected by MDS and select data nodes with larger sizes which means those data nodes have higher review scores.

By selecting a big data node from one of the groups on the left top (step 2), he noticed a row highlighted in both stalagmite and the table. He expanded it from stalagmite to see the detail of the data row which is the game "Bio Shock" in the table. To explore a little bit more, he selected another big node from the group right below his previous selection in the scatterplot. He found the game "Grand Theft Auto" (GTA) which he likes as well. He moved up one more hierarchical level to see if he may like any other similar ones, and then he saw another GTA which is the same game for another platform. Since he likes "Bio Shock" and "GTA", he simply dragged "Bio Shock" to the "GTA" group to tell the machine learner they are similar (step 3). After submitting those metric learning inputs, the hierarchical clustering is updated. As shown in stalagmite, those three games are grouped together (step 4).

Looking at the metric learning scores with pink bars, Max noticed that his interactions are mostly related to "Review Score" and "Length Polled" (step 5). Although it shows "sales" is not highly related, Max thought sales might be an important feature to find the games he likes. To apply those features, he used the scatterplot again by setting the x-axis to "Sales", the y-axis to "Length Polled" and mapped size to "Review Score" (step 6). With this projection setting, he selected two outliers (step 7) with large dots meaning high review scores: (1) a game with low length polled but high sales and (2) a game with high length polled but low sales, which he can discern from the table view. From the associated highlight in the table view, he found that the first game is "Mario Kart Wii" and the second one is "Bio Shock", which is already in his group. Thus, he dragged "Mario Kart Wii" into his group (step 8). Meanwhile, he went one hierarchy level up to see games in his parent group, and then he noticed "Kami" which is the type of game he does not like (step 9). He switched the projection back to MDS and selected a game close to "Kami" in the projection (step 10). The selected game "De Bob" is also a type of game he does not like, so he Meta + clicked his cluster so he could drag "Kami" (highlighted in orange) out of his group and drop it with "De Bob" (step 11). This *Change-node's-cluster* interaction marked these two games similar and "Kami" dissimilar from its original group.

After submitting those metric learning inputs, he got updated groups and those dragged games were highlighted in green. He expanded his group at a higher hierarchy level to see if there were any other games he might like. Happily, he found that he likes all of them except "Little big Planet" (step 12). He was very satisfied with this regrouping, so he wanted to see the rules of this group. By clicking this new group, he saw bars of rule weights at the table header and rule ranges highlighted in the columns which have high rule learning weights (step 13). By clicking one of the attributes with

[2]This dataset has removed some special characters with data cleaning. For example, the game "Ôkami" is incorrectly stated as "Kami"

high weights, he can see a heatmap in the stalagmite. The cluster in darker purple is more related to that attribute (step 14). Max explored the groups with darker purple and rule ranges, and found that he could see his preference is highly related to certain ranges of the attributes (as opposed to seeing only which attributes). Therefore, he found out some hidden rules he had not realized before: his choices are mostly based on two attributes: "Sales" and "Used Price". The rules' ranges show that he likes games with a high amount of sales and the used price is around 25 dollars. Max was curious if based on these two learned rules, he could find more games he likes. He turned back to the scatterplot and set the x-axis and y-axis to "sales" and "used price" respectively (step 15). Not surprisingly, he quickly found two more games ("Left 4 Dead" and "Dead Space") that he likes based on these two rules (step 15). These hidden rules are learned from his preferences, and he can apply these rules for his future game search.

5 Expert Feedback

We collected feedback from five visualization experts (2 female, 3 male; 3 faculty, 2 graduate students) to assess some of our design decisions and the usability of our prototype. We gave these participants (P1-P5) a tour of the prototype and asked them to think aloud as they developed a cluster of countries[3].

All of them started from the scatterplot, looking for outliers or groups of interest. P4 tried different projection methods (MDS, t-SNE, and UMAP) to explore the clustering and mentioned that the projection was "extraordinarily helpful for [his] exploratory work." They all used the table to review data in detail. Perhaps thanks to the fact that stalagmite lays out boxes representing individual points right next to table rows, all but one of our experts found the connection straightforward. In the remaining case (P5), the expert was unclear on hierarchical clustering in general. P5's difficulty is instructive because we aim to target a broad audience with future work (detailed in Sect. 7) and will develop ways to clarify HAC and stalagmite to new users.

All participants used some of the interactions to change clusters and found the drag-and-drop interactions for similar clusters straightforward. The interactions for separating items (*Split-cluster* and *Change-node's-cluster*) were less obvious because they required keystrokes of which participants needed reminders. Alternate interactions, like a context menu, were suggested and we will use that to improve future work. Overall, participants recognized that the backend machine learning worked to change the groups based on their feedback. One noted that it is not always clear exactly what the effect was, referring to the fact that the model updates the entire clustering, and items may not end up exactly where they were placed. This is expected behavior, but that feedback points towards interesting research in a hybrid clustering model with both metric learning similarity constraints and must-cluster constraints across varying levels of the hierarchy.

The rules were perhaps the most popular interface feature, including the value range displays in the table columns, the attribute weight bars for rule importance in the table columns, and the coloring in stalagmite to show clusters where a selected attribute is important. The column weights guided participants toward variables of interest once they established a group of interest. The cluster coloring got a unanimous appreciation for helping drill down, calling attention to certain clusters, and seeing how the clustering was relying on certain attributes.

6 Empirical Machine Learning Evaluation

We need to ensure that our learning method is capable of producing valuable subsets of interest for users with their limited interactions. We assume the user is consistent but only able to provide a limited number of metric learning constraints and wants to ensure they get helpful results from a relatively small amount of effort. To evaluate our model's performance, we simulate users' interactions, and see how different interactions enhance the basic HAC model by measuring how performance evolves as more constraints are assigned. We apply our method to some common tabular data: Ionosphere, Parkinsons, and Breast cancer [4].

6.1 Metrics of Success

We use two metrics to calculate model performance scores. To measure the clustering quality, we compare the learned cluster labels with ground truth labels in our *linear match score*. We match each data point's learned label with its ground truth label (clusters are first re-numbered to align with labels) and count 1 point if they are matched, otherwise, 0 points. A normalized score between 0.0 and 1.0 is derived by dividing by the total number of data points. During the simulation, the learning algorithm does not get access to the ground truth, but we use the labels to evaluate its performance.

Since the learning is based on metric learning (ITML [17]), we evaluate the model's quality by testing the accuracy of the distance function at k-Nearest-Neighbor (kNN) classification with *cross validation*. We compare the cross-validation score using the distance function to the score without it (unweighted Euclidean distance).

6.2 Interaction Simulations

6.2.1 Constraint Assignment

Our technique assumes that users have sufficient domain knowledge to provide valuable information about similarity and dissimilarity relationships between groups, so we use ground truth cluster labels to create constraint assignments in our simulations.

To perform *Merge-nodes*, we randomly pick one pair of data points with different HAC labels that actually share the same ground truth label. We then assign a must-link constraint to the pair and run the metric learning. For *Split-cluster*, the pair of data points will be two data points with different ground truth labels but with the same HAC label. The constraint assigned will be a cannot-link constraint accordingly.

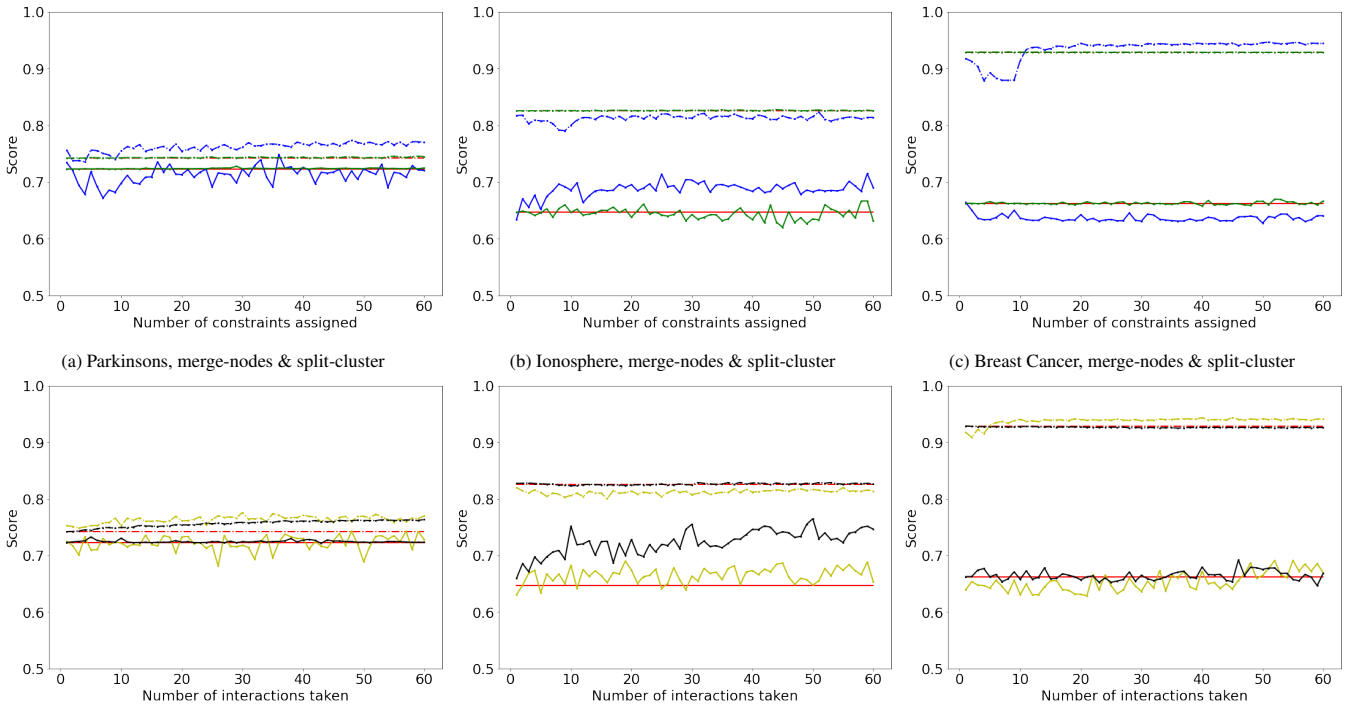
Merge-clusters takes two groups of data. We select two groups that share the same real cluster label while HAC assigns different labels to some of their data points. Must-link constraints are assigned to data points pairwise within the two groups.

For *Change-node's-cluster*, two groups of data are needed as a source group and a target group. In the source group, we randomly pick one data point which has a different HAC label from the others. Cannot-link constraints are assigned between it and the rest of the source group's data points. Must-link constraints are assigned between it and the target group's data points. Due to the different sizes of the groups chosen, each *Merge-clusters* and *Change-node's-cluster* can have a variable number of constraints provided to the model.

6.2.2 Experiment and Interpretation

We conduct experiments to evaluate four types of interactions individually with sets of varying sizes. Mean scores are collected from multiple runs. Results are shown in Fig. 5. Cross-validation scores are used to measure how well the constraints passed into ITML are helping it learn a distance function. Linear match scores are used to measure the hierarchical clustering (HAC) accuracy based on the learned cluster label. We observe steady improvement effects or fast convergences for both the cross-validation scores and the linear match scores after a small number, usually around 10, of interactions on various data sets of different sizes. The results suggest that the user can use the method to obtain a sufficient model with just an affordable amount of effort.

[3]Data were attributes of countries thought to impact happiness [34].



(a) Parkinsons, merge-nodes & split-cluster

(b) Ionosphere, merge-nodes & split-cluster

(c) Breast Cancer, merge-nodes & split-cluster

(d) Parkinsons, merge-clusters & change-node's-cluster

(e) Ionosphere, merge-clusters & change-node's-cluster

(f) Breast Cancer, merge-clusters & change-node's-cluster

Figure 5: Interaction Simulation Results

7 Future Work

Based on the expert feedback, we can improve the interface components to assist users' exploration and interactions better. First, we can get rid of some specific keyboard clicks for user interactions. We will still keep "drag and drop" for interactions based on the feedback. Besides that, we will apply a right click and show a menu for the other interactions. Second, we can include analytic provenance features so users can see and improve the feedback they have already provided. After many iterations of tuning the machine learning model, users might forget which data nodes or sub-clusters they have interacted with, or change their mind about a relationship. By retrieving previous interactions, users could better understand how their previous interactions impacted the model, maximizing the effectiveness of using the existing capabilities. Third, one of our participants (P5) is unclear on hierarchical clustering in general and had trouble understanding the structure of stalagmite. As a novice of hierarchical clustering, P5 did not quickly see how the sub-clusters at each level are connected. This might be because P5 did not have the background knowledge that hierarchical agglomerative clustering works by merging clusters from the bottom. Since we target a broader audience, we need to highlight the structure of hierarchical clustering. For example, when users mouseover a cluster or leaf node, its parent clusters will be highlighted. This will help the non-expert users quickly get the concept and be able to further explore.

In our experiments, we observed differences between the improvements from must-link constraints versus cannot-link constraints, with variation across datasets. We see a performance dip at a small number of constraints, as shown with some experimental data sets like ionosphere and breast cancer. This could be due to the instability of the model at an early stage of the learning process. Though the effect is unlikely to cause problems for users, as it happens with small amounts of randomly selected data, investigating the cause of this behavior could result in a refined understanding of how metric learning adapts based on particular types of input.

8 Conclusion

In this work, we presented a technique for semantic, interactive hierarchical clustering. We take advantage of the cluster substructure to create a rich set of interactions, facilitated by a compact visualization of the hierarchy (stalagmite). The interactions can be translated to metric learning constraints, which makes it possible to incorporate user feedback in hierarchical clustering. We use a prototype implementation to demonstrate the application to real data in a usage scenario, and show that this technique can be effective for customizing and comprehending groups. Experimental results from simulations demonstrate the performance of the underlying machine learning as applied to these interactions. Finally, the technique is further validated by expert feedback on the interface and functionality.

References

[1] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216, 1993.

[2] S. Amershi, J. Fogarty, and D. Weld. Regroup: Interactive machine learning for on-demand group creation in social networks. In *Pro-*

ceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 21–30. ACM, 2012.

[3] M. Ankerst, C. Elsen, M. Ester, and H.-P. Kriegel. Visual classification: an interactive approach to decision tree construction. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 392–396, 1999.

[4] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[5] E. Bair. Semi-supervised clustering methods. *Wiley Interdisciplinary Reviews: Computational Statistics*, 5(5):349–361, 2013.

[6] A. C. Bart. Video games csv file, 2017. retrieved April 12 2021 from https://corgis-edu.github.io/corgis/csv/video_games/.

[7] S. Basu, S. M. Drucker, and H. Lu. Assisting Users with Clustering Tasks by Combining Metric Learning and Classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 394–400, 2010.

[8] E. T. Brown, A. Endert, and R. Chang. Human-machine-learner interaction: The best of both worlds. In *Proceedings of the CHI Workshop on Human Centred Machine Learning (HCML)*, 2016.

[9] E. T. Brown, J. Liu, C. E. Brodley, and R. Chang. Dis-function: Learning distance functions interactively. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 83–92. IEEE, 2012.

[10] P. Bruneau, P. Pinheiro, B. Broeksema, and B. Otjacques. Cluster Sculptor, An Interactive Visual Clustering System. *Neurocomputing*, 150:627–644, 2015. Special Issue on Information Processing and Machine Learning for Applications of Engineering Solving Complex Machine Learning Problems with Ensemble Methods Visual Analytics using Multidimensional Projections.

[11] F. Cao and E. T. Brown. Dril: Descriptive rules by interactive learning. In *2020 IEEE Visualization Conference (VIS)*, pages 256–260. IEEE, 2020.

[12] M. Cavallo and C. Demiralp. Clustrophile 2: Guided Visual Clustering Analysis. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):267–276, 2019.

[13] J. Choo, H. Lee, J. Kihm, and H. Park. ivisclassifier: An interactive visual analytics system for classification based on supervised dimension reduction. In *Visual Analytics Science and Technology (VAST), 2010 IEEE Symposium on*, pages 27–34. IEEE, 2010.

[14] P. Clark and T. Niblett. The cn2 induction algorithm. *Machine learning*, 3(4):261–283, 1989.

[15] W. W. Cohen. Fast effective rule induction. In *Machine learning proceedings 1995*, pages 115–123. Elsevier, 1995.

[16] J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon. Information-theoretic metric learning. In *Proceedings of Twenty-Fourth International Conference on Machine Learning (ICML)*, pages 209–216, 2007.

[17] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216, 2007.

[18] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[19] M. Desjardins, J. MacGlashan, and J. Ferraioli. Interactive visual clustering. In *Proceedings of the Twelfth International Conference on Intelligent User Interfaces*, pages 361–364. ACM, 2007.

[20] A. Endert, P. Fiaux, and C. North. Semantic interaction for visual text analytics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 473–482. ACM, 2012.

[21] A. Endert, P. Fiaux, and C. North. Semantic interaction for visual text analytics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 473–482. ACM, 2012.

[22] A. Endert, W. Ribarsky, C. Turkay, B. Wong, I. Nabney, I. Díaz Blanco, and F. Rossi. The state of the art in integrating machine learning into visual analytics. *Computer Graphics Forum*, 3 2017.

[23] J. Fogarty, D. Tan, A. Kapoor, and S. Winder. Cueflik: interactive concept learning in image search. In *Proceedings of the sigchi conference on human factors in computing systems*, pages 29–38. ACM, 2008.

[24] R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine learning*, 11(1):63–90, 1993.

[25] E. Hoque and G. Carenini. Interactive topic hierarchy revision for exploring a collection of online conversations. *Information Visualization*, 18(3):318–338, 2019.

[26] M. John, C. Tominski, and H. Schumann. Visual and analytical extensions for the table lens. In *Visualization and Data Analysis 2008*, volume 6809, page 680907. International Society for Optics and Photonics, 2008.

[27] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon. Visual analytics: Definition, process, and challenges. In *Proceedings of the IEEE Conference on Information Visualization*, pages 154–175. Springer Berlin Heidelberg, 2008.

[28] J. B. Kruskal and J. M. Landwehr. Icicle plots: Better displays for hierarchical clustering. *The American Statistician*, 37(2):162–168, 1983.

[29] S. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[30] Z. Lu and T. K. Leen. Semi-supervised learning with penalized probabilistic clustering. In *Advances in neural information processing systems*, pages 849–856, 2004.

[31] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[32] E. Nam, Y. Han, K. Mueller, A. Zelenyuk, and D. Imre. ClusterSculptor: A visual analytics tool for high-dimensional data. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 75–82. IEEE, 2007.

[33] E. J. Nam, Y. Han, K. Mueller, A. Zelenyuk, and D. Imre. ClusterSculptor: A Visual Analytics Tool for High-Dimensional Data. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology*, VAST '07, page 75–82. IEEE Computer Society, 2007.

[34] S. D. S. Network. World happiness report, 2019. retrieved 13 DEC 2020 from https://www.kaggle.com/unsdsn/world-happiness.

[35] M. Okabe and S. Yamada. An interactive tool for human active learning in constrained clustering. *Journal of Emerging Technologies in Web Intelligence*, 3(1):20–27, 2011.

[36] R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[37] R. Rao and S. K. Card. The table lens: merging graphical and symbolic representations in an interactive focus+ context visualization for tabular information. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 318–322, 1994.

[38] L. Rokach and O. Maimon. Clustering methods. In *Data mining and knowledge discovery handbook*, pages 321–352. Springer, 2005.

[39] Rulequest Research. Is see5/c5.0 better than c4.5?, February 2017. Retrieved from https://www.rulequest.com/see5-comparison.html August 24, 2020.

[40] Rulequest Research. Information on see5/c5.0, April 2019. Retrieved from https://www.rulequest.com/see5-info.html August 24, 2020.

[41] D. Sacha, M. Sedlmair, L. Zhang, J. Lee, D. Weiskopf, S. North, and D. A. Keim. Human-centered machine learning through interactive visualization: review and open challenges. In *In Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, 2016.

[42] P. Y. Simard, S. Amershi, D. M. Chickering, A. E. Pelton, S. Ghorashi, C. Meek, G. Ramos, J. Suh, J. Verwey, M. Wang, et al. Machine teaching: A new paradigm for building machine learning systems. *arXiv preprint arXiv:1707.06742*, 2017.

[43] A. K. H. Tung. *Rule-Based Classification*, pages 1–4. Springer New York, New York, NY, 2016.

[44] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl, et al. Constrained k-means clustering with background knowledge. 2001.

[45] J. Wenskovitch, M. Dowling, and C. North. With respect to what? simultaneous interaction with dimension reduction and clustering projections. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, pages 177–188, 2020.

[46] L. Yang and R. Jin. Distance metric learning: A comprehensive survey. *Michigan State Universiy*, pages 1–51, 2006.