Cluster Sculptor, an interactive visual clustering system

P Bruneau, P Pinheiro, B Broeksema, B Otjacques

Cluster Sculptor, an Interactive Visual Clustering System

P. Bruneau[a,], P. Pinheiro[a], B. Broeksema[a], B. Otjacques[a]

[a] *Centre de Recherche Public - Gabriel Lippmann, 41 rue du Brill, L-4422 Belvaux (Luxembourg)*

Abstract

This paper describes Cluster Sculptor, a novel interactive clustering system that allows a user to iteratively update the cluster labels of a data set, and an associated low-dimensional projection. The system is fed by clustering results computed in a high-dimensional space, and uses a 2D projection, both as support for overlaying the cluster labels, and engaging user interaction. By easily interacting with elements directly in the visualization, the user can inject his or her domain knowledge progressively, crafting an updated 2D projection and the associated clustering structure that combine his or her preferences and the manifolds underlying the data. Via interactive controls, the distribution of the data in the 2D space can be used to amend the cluster labels, or reciprocally, the 2D projection can be updated so as to emphasize the current clusters. The 2D projection updates follow a smooth physical metaphor, that gives insight of the process to the user. Updates can be interrupted any time, for further data inspection, or modifying the input preferences. The interest of the system is demonstrated by detailed experimental scenarios on three real data sets.

Keywords: interactive clustering, dimensionality reduction, visual clustering

1. Introduction

Clustering algorithms are extensively employed in various domains such as data mining, information retrieval and bio-informatics. They provide means to

Email addresses: bruneau@lippmann.lu (P. Bruneau), pinheiro@lippmann.lu (P. Pinheiro), broeksem@lippmann.lu (B. Broeksema), ojacque@lippmann.lu (B. Otjacques)

classify unlabeled multivariate items of various data types in an unsupervised manner. Among other use-cases they are used to find genome-wide expression patterns [1], patterns in trajectories [2, 3] and similar documents in a text corpus [4]. In order to exploit the full potential of these algorithms [5], interactive visual representations are required for both analysis and communication purposes.

The high-dimensional spaces real-world data sets often lie in are typically harmful to clustering algorithms. In particular, most well-known clustering algorithms (e.g., k-means [5], spectral clustering [6], or EM for the Gaussian mixture [7, Chapter 9]) rely on the Euclidean distance, or some transform of the latter. Unfortunately, this kind of distance suffers from the curse of dimensionality: as the dimensionality increases, the distribution of pairwise distances is shifted towards high values, while its variance remains almost unchanged (see Figure 1). Pairwise Euclidean distances thus tend to become indistinguishable when the dimensionality increases [7, Section 1.4]. The use of some adaptive [8] or locally sensitive [9] measure may alleviate the problem, the study of which remains central in the machine learning community.

On the other hand, usage of clustering results as a communication tool is promising, but also affected by the reference to these high-dimensional spaces. The latter are indeed challenging for representation, and user understanding. For effective communication and presentation, the results of clustering algorithms are thus often combined with a Dimensionality Reduction (DR) technique. These techniques [10, 11] project a high dimensional data set to a lower dimensional space, for visualization in two or three dimensions. The resulting dimensionality reduced data set can be visualized using scatterplot techniques. Cluster labels are then overlaid on this low dimensional projection, using a mapping to glyphs or category colors, as illustrated in Figure 2.

Such visual representations can be used both for building and adjusting a mental map of the data at hand, or as an entry point for finer data inspection using brushing techniques [12]. This need may occur when considering the organization of image collections [13], in the context of multimedia retrieval

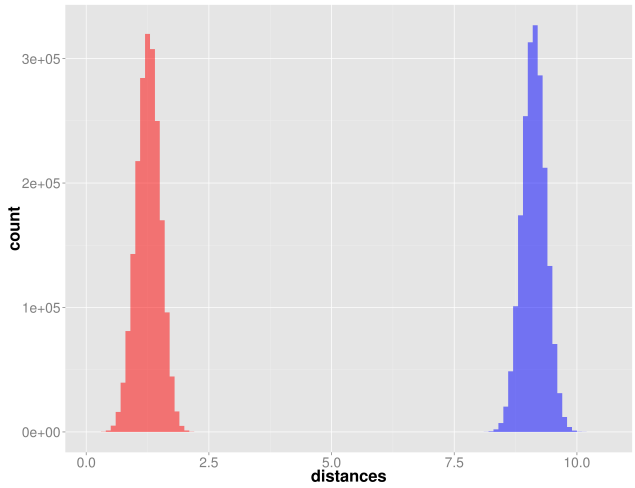

Figure 1: Distribution of pairwise distances computed from 2000 d-dimensional elements generated uniformly in $[0, 1]$, *in red*: for $d = 10$, *in blue*: for $d = 500$

Figure 2: t-SNE 2D projection of the COIL-20 image collection. The result from the spectral clustering algorithm is mapped to categorical colors for the point glyphs.

engines, or when reporting public or medical data. In many cases though, the initial clustering of a data set is deemed to be imperfect with regard to a ground truth, or user expectations. Decision bounds may be unsatisfactory, the ground truth clusters may be multimodal in the 2D space or exhibit outliers. Worse, data attributes might be badly chosen, noisy, or combined in an inappropriate distance function.

Tackling all these issues at once is certainly not realistic, but at least should the user be allowed to interact with the clustering and DR results in order to investigate them, and partly cope with them. It is thus possible to let the user manually amend a clustering structure by selecting and labeling points in the projection. However, with naive rectangle or lasso selections this might result in density gaps within the visual representation of the clusters, which is not consistent with the intuitive meaning of a cluster. Also, when analyzing or communicating clustering results, a user may have preferences in the arrangement of clusters in the visualization space (e.g., highlighting semantic regions). Rearrangement of the clusters' relative positions should thus be supported to some extent. However, clustering and DR techniques usually do not allow such fine tuning.

The purpose of this work is to support the cluster analysis in a visual, semi-automatic way. We largely build upon the t-SNE DR technique [14]. After a review of the related work in Section 2, we recall its use in a batch setting to build an initial 2D projection of the data in Section 3. Then in Section 4, we give a first glance of the Cluster Sculptor interface and its data inspection facilities. An interactive 2D scatterplot view and legend support the efficient input of user preferences, and is augmented by controls to parametrize and initiate updates of the visualization and the associated clustering. To avoid tedious and error-prone manual grouping and relabeling of data elements, we derive semi-automated label diffusion techniques, described to a greater extent in Section 5. The user simply has to select few elements (i.e., *seeds*, in the remainder), then used as a basis to interactively refine the clusters. Beyond minimizing the amount of effort the user has to spend, this choice accounts for,

and even actively uses, the data distribution in the 2D space. In Section 6, we describe how the underlying dissimilarity matrix is updated behind the scenes to emphasize the clustering structure. Section 7 then shows how the visualization is smoothly and interactively adapted to reflect this update. Detailed experimental scenarios on well known benchmark image collections (COIL-20 [15], MNIST [16]) and an actual biological data set, presented in Section 8, illustrate tasks that can be conducted with Cluster Sculptor, and demonstrate the interest of the system. After a critical discussion of our proposition in Section 9, we give a summary of our findings, and draw some perspectives for future work, in Section 10.

2. Related Work

This paper contributes essentially to the interactive and visual clustering state of the art, by assembling ideas taken from the existing literature (i.e., DR technique, clustering, information visualization and label propagation), with crucial contributed parts (e.g., dissimilarity adaptation scheme) in an interactive system. Therefore, in this section we focus on relating our work to the existing interactive clustering literature, and enrich it with references from several connected domains, such as DR and semi-supervised approaches.

The visual and interactive clustering literature covers a variety of work, that differs essentially from the pursued objectives.

Seo and Shneiderman propose the Hierarchical Clustering Explorer (HCE) [17]. This system contains various linked views to get oversight and detailed views of hierarchically clustered data, obtained in the context of genetic analysis. They provide means to make cluster comparisons and dynamic query controls to eliminate uninteresting clusters. Their approach is not only focusing on hierarchical clusters, they also seem to assume that the clusters faithfully represent a ground truth as they provide no means to change the clusters.

Turkay et al. [18] also proposed an interactive system to analyze clustering results. With the cluster tendency view and the parallel cluster view, they

put an emphasis on the comparative analysis of several clustering results. The parallel cluster view visualizes where data points appear in different clusterings using a visualization based on parallel sets, which allows to comparatively find stable structures. As a complement, the cluster tendency view visualizes the similarity matrix of a brushed subset of elements. This allows for validating if the selected data points are likely to be clustered. In our approach we go one step further and provide fine-grained interactive control of the clustering of a selection of data points. Not only do we provide more control over clustering, we also update the projection in order to visually separate the clusters more clearly.

Rinzivillo et al. [19] and Adrienko et al. [3] propose an interactive clustering method for large spatio-temporal data. Initially a density based clustering algorithm is applied to a subset of the data points with a suitable distance function. Next, prototypes are selected for each cluster which, in combination with a cluster distance threshold, form classifiers. Classifiers may be refined by the user by adapting the initial clusters. For example, subclusters can be excluded, turned into new clusters or dissolved among other subclusters. These refinements are supported by visual representations of the clusters and subclusters, which allow the user to interact with both to perform classifier refinement operations. Finally, the obtained classifiers are used to infer the class of the remaining data points. This approach differs from ours in that it does not have to deal with the placement of clusters as these are fixed by their geo-spatial coordinates. As a result, not much can be done in this case to make clusters visually more separated.

In the context of document topic modeling, the iVisClustering [4] tool allows to inspect and interact with textual data and a related latent Dirichlet allocation topic model. Its coordinated views comprise a force-directed layout, based on the similarity of document topic distributions. The topic structure is used to derive cluster summary nodes. The tool is used to identify documents poorly reflected by the current model, or refine vague topics and derive hierarchies of nested topics. The cluster structure is either updated via model parameters tun-

ing or hierarchical refinements, significantly differing from our approach, where a plain structure is non-parametrically adapted according to user interactions.

Schreck et al. [2] also propose a method for cluster analysis of trajectory data, but base their approach on Kohonen maps. Traditionally, Kohonen maps are unsupervised, as the initial grid is determined automatically based on, for example, random initialization or principal component analysis of the input data. In [2] a user guided approach is proposed where some of the grid elements are drawn by the user while the remaining elements are interpolated from the user-specified ones. The reasoning for this approach is very similar to our goal, starting with a sensible initial layout which next can be iteratively refined by the user, although we use the unsupervised t-SNE to get an initial map. Also, very similar to what we do is the visualization of the iterative process of the Kohonen map training process. For each step they update the color coding of the cell, which allow the user to visually inspect how the learning process evolves and when it starts to converge. We do something similar by updating the positions of the data points in the map after each t-SNE iteration, also allowing the user to interact with the mapping process, and visually estimating when it is stable enough.

ManiMatrix by Kapoor et al. [20] provides an interactive way to change the trade-off in the errors of a classifier. In a visual confusion matrix the user can change the distribution of classification errors based on the requirements for the scenario at hand. Although our approach is in the domain of clustering, it somewhat echoes the approach of Kapoor et al. The ManiMatrix approach starts with initial classifiers which are refined iteratively by the user. Our work presents a similar iterative workflow, but applied to a clustering model. Additionally, reminiscent to how interactions in ManiMatrix result in an updated classifier model, the update of the clustering in our approach leads to a modified mapping of the data points.

Contributions in visual clustering also emerged in literature from the DR domain. For example, Broeksema et al. [21] combine dimensionality reduction and clustering in their tool. The applied clustering is based on a Voronoi parti-

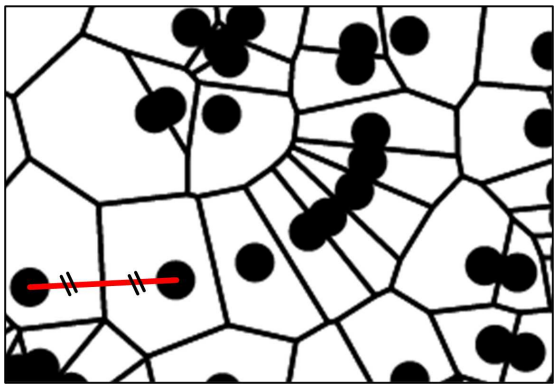

Figure 3: Voronoi cells examples. The path between two elements is highlighted in red, emphasizing that the respective Voronoi cell border is exactly halfway between them. The glyph in the cell is thus the closest among all glyphs to any points in the cell.

tioning of the points in the projection space. The Voronoi cell of a data element is the subset of the projection space such that the data element glyph is closer than any other glyph to all points in the cell (see Figure 3). Starting at the smallest cell, each cell is merged with its neighbors as long as these neighbors are within a user configured distance. Their approach does not allow for updating the clusters nor the projection to reflect changes in the clustering. Moreover, they focus on categorical data, whereas the scope is set on multi-dimensional numerical data in this paper.

Some interactive DR techniques are also closely related to our work. For instance, Philippeau et al. [22] propose an interactive DR technique for organizing multimedia documents in a 2D visual space. A subset of the documents is placed in the visual space by the user. Based on this placement, a similarity measure is trained which is next used to place the remaining documents in the visual space. A similar approach is taken by Mamani et al. [23], where the differences between default positions of sampled elements and those set by the user feed a feature transform optimization scheme. Those approaches are different to ours in that the clustering is implicit, i.e., based on placements by the user.

Aupetit emphasizes that all but trivial data sets lie on high-dimensional

(HD) manifolds, impossible to map faithfully to a two-dimensional (2D) space, as recommended for rendering scatterplots on-screen. In practice, this causes two types of *projection artifacts* [24]:

- *tears*: two elements close in the HD space are rendered too far in the 2D space,
- *false neighborhoods*: two elements far in the HD space are rendered as neighbors in the 2D space.

Aupetit then illustrates that each DR technique tends to favor a kind of artifacts, e.g., PCA [25] is prone to false neighborhoods, whereas CCA [26] is likely to tear manifolds of the HD data. To make these patterns apparent to a viewer, the author suggests that hovering over a reference element in the 2D space interactively maps the HD distances with respect to other elements to a gray scale coloring of the respective Voronoi cells in the 2D space. This initial work was then extended to ProxiViz [27] by the use of the Shepard interpolation [28] to smoothen the Voronoi cells, and timer techniques to avoid the flickering that often occurs with false neighbors. This important point pertains to our approach, and we thus included a simplified version of ProxiViz in our system, as shown in Section 4. Along with other tools, such as a data inspector, its use as a support for taking informed decisions before modifying the visualization or the clustering structure is illustrated in Section 8.

Martin et al. [29] propose an approach that has some similarities to our own. In their approach, a user can interactively pose additional constraints on the position of data items in the projected space. These constraints will be taken into account in a next iteration of the dimensionality reduction. They assume, like we do, that the user can inject knowledge on the similarity of objects. However, they do not concern themselves with providing the users with means to actually cluster objects, which is an explicit goal of our work. Akin to the latter is Dis-Function [30], where user constraints are injected in a feature weight optimization. Full interactive controls are provided, but each

iteration requires a full optimization to proceed before the user can see the result of his actions. The authors indicate that a couple of seconds are needed to update the visualization of data sets with barely a hundred elements, thus not pleading for the scalability of the method's interactivity. Alternatively, we ground upon the physical metaphor underlying t-SNE, to engage the user in following the progressive update consecutive to his or her actions.

Some semi-supervised techniques also use constraints between elements, not necessarily user-specified, to improve a classification function [31]. In this paper, we actually adapt a technique from the semi-supervised literature, the label propagation [32], and incorporate it as a label diffusion tool. To this respect, our work distinguishes from Dis-Function and the approach by Martin et al.: whereas seeds selected by the user directly affect the distance function and thus the visualization in their work, label diffusion from seeds only modifies the clustering structure in ours. The visualization then uses the clustering to guide its updates more globally.

The present work is roughly a follow-up on the proposition of Bruneau and Otjacques [33]. However, a greater care in now taken about providing consistent user experience, with the use of t-SNE [14], overviewed in the next section, to allow a smoothly evolving mental map.

3. A Summary of the t-SNE Projection Technique

Let us consider a set of N elements, which is fed as input to Cluster Sculptor. We assume it is defined by d numerical features, with $d > 3$ for high-dimensional data sets. Each element is stored as a row of the $N \times d$ matrix $\mathbf{X}$.

Multidimensional projections have the intrinsic capability to represent a HD data set in a visual space (consisting of either two or three dimensions) while, to some extent, preserving pairwise distances. In this work, we focus on 2D projections. The goal is thus to estimate the $N \times 2$ matrix $\mathbf{Y}$ of the respective elements in $\mathbf{X}$. We use t-SNE [14] DR technique, already shown as resilient to severe tearing and false neighborhood artifacts (see Section 2 for definitions).

We motivate this choice by its suitability for the data sets and task at hand, and the fact that the resulting 2D projections are especially useful to emphasize clusters and the respective low-dimensional local manifolds [34].

Let us define $\mathbf{P}$ (respectively $\mathbf{Q}$) as the matrix formed by the probabilities $\mathbf{P}_{nn'}$ (respectively $\mathbf{Q}_{nn'}$) of elements n and n' being neighbors in $\mathbf{X}$ (respectively $\mathbf{Y}$). $\mathbf{D}$ (respectively $\mathbf{G}$) is a $N \times N$ matrix of dissimilarities computed from a function of pairs of elements in $\mathbf{X}$ (respectively $\mathbf{Y}$). These dissimilarities are initialized with normalized Euclidean distances, but any valid dissimilarity (i.e., in the unit interval range) could be used instead. t-SNE estimates $\mathbf{Y}$ by optimizing the Kullback-Leibler divergence KL($\mathbf{P}||\mathbf{Q}$), with $\mathbf{P}$ and $\mathbf{Q}$ given as:

$$\mathbf{P}_{nn'} = \frac{1}{2}\left(\frac{\exp(-\frac{\mathbf{D}^2_{nn'}}{2\sigma^2_n})}{\sum_{m\neq n}\exp(-\frac{\mathbf{D}^2_{nm}}{2\sigma^2_n})} + \frac{\exp(-\frac{\mathbf{D}^2_{nn'}}{2\sigma^2_{n'}})}{\sum_{m\neq n'}\exp(-\frac{\mathbf{D}^2_{n'm}}{2\sigma^2_{n'}})}\right) \tag{1}$$

$$\mathbf{Q}_{nn'} = \frac{(1+\mathbf{G}^2_{nn'})^{-1}}{\sum_{m\neq m'}(1+\mathbf{G}^2_{mm'})^{-1}} \tag{2}$$

In Equation (1), the σ_n can be interpreted as reflecting the extent of the neighborhood of element n, and is determined automatically with a binary search, purposely avoiding overly peaked distributions for $\mathbf{P}_{n\cdot}$ vectors. In the literature, the functional form $\exp(\mathbf{D}_{nn'}/\sigma)$ is used extensively, e.g., in the context of kernel learning [7] or neural networks [35]. It is often referred to as the *Radial Basis Function* (RBF). Setting σ there has also been studied independently of t-SNE, and alternatives to the binary search, e.g., using nearest neighbors, have been proposed [36, 9].

The gradient of KL($\mathbf{P}||\mathbf{Q}$) with respect to $\mathbf{Y}$ is available in closed form, and can be used to find a local optimum for $\mathbf{Y}$ [14]. Equation (1) is symmetrized to smooth computational issues induced by outliers.

The terms on the right hand side of Equation (1) are actually unnormalized Gaussian distributions, while the right hand side of Equation (2) is the unnormalized heavy-tailed Cauchy distribution. This characteristic ensures that elements sharing a common neighborhood in 2D are also closely related in the high dimensional space. Furthermore, using a heavy-tailed distribution for $\mathbf{Q}$

leads pairwise distances in $\mathbf{Y}$ exceeding close neighborhood to be relatively insensitive to their counterpart in $\mathbf{X}$. In other words, the method is focused on extracting local manifolds in data, with little respect to higher-order structures in the HD space. Actually, van der Maaten and Hinton noted that targeting the faithful modeling of all distance ranges in a DR method causes a *crowding problem* [14]: all elements tend to be packed in the center of the visualization, increasing the risk of false neighborhood. The principles underlying t-SNE alleviate this risk, and encourage an efficient use of the 2D space.

Two successive phases happen in the optimization algorithm. The initial phase is inspired from a simulated annealing optimization, and allows large movements in the search space so that elements form rough regions. After a significant amount of iterations (100 in the R implementation of the method), the algorithm switches to the classical gradient steps, that perform the local optimization of element positions. This local optimization can be interpreted as the temporal evolution of a 2D graph completely linked by a system of springs. This relates t-SNE to spring-based graph layout algorithms [37], already employed as a DR method in the literature [22, 13, 4].

Figure 4 shows the result eventually obtained for the COIL-20 data set (see Section 8 for a more detailed introduction). It illustrates the ability of the t-SNE method to emphasize local manifolds of data, and distribute them more freely in the 2D space. For example, a class in COIL-20 is formed by a set of images representing the full rotation of an object (see r.h.s. of Figure 4), and translates as a linear or circular component on the l.h.s. of Figure 4. The methods favors the even distribution of these manifolds in the 2D space, facilitating the visual identification of clusters.

Clustering usually takes place in the HD space, and its results are then overlaid on a 2D projection for off-line inspection. In this paper, we propose to go beyond this static approach, by rather considering HD clustering results as an initial input, and allowing a user to amend it using interactive tools. Alternatively, the user may want to keep the state of his or her clustering results, and improve the fit of the visualization to the latter. This task is also supported

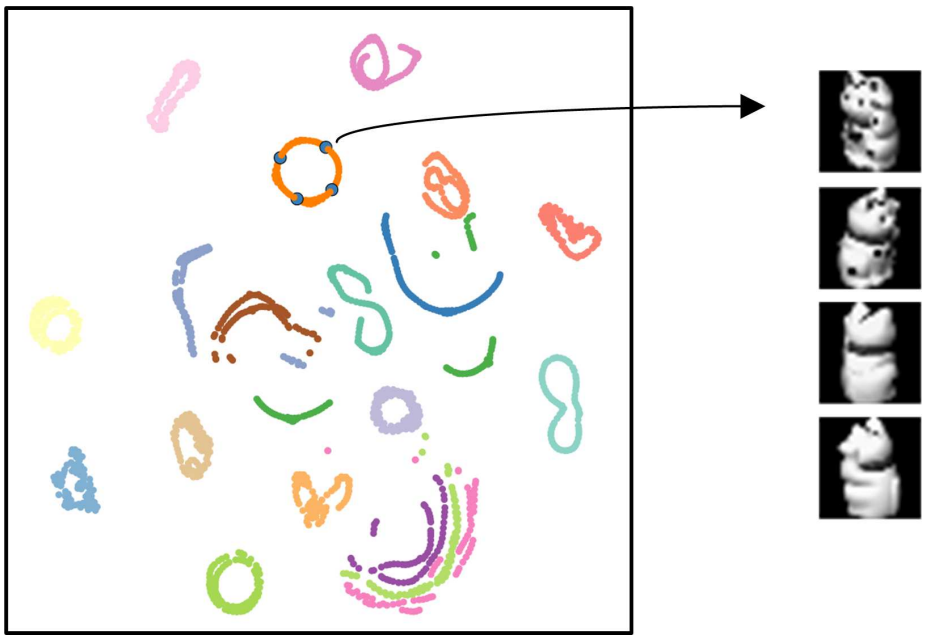

Figure 4: *Left*: t-SNE 2D projection of the COIL-20 image collection. Ground truth classes are mapped to the point glyphs as categorical colors. *Right*: The highlighted glyphs, and the images they map, illustrate the ability of t-SNE to recover the manifold implied by the rotation of the class object.

by the interactive set of tools overviewed in Section 4.

As exposed further in the remainder, Cluster Sculptor is not a clustering algorithm *per se*: HD clustering results are used as an input, and amended interactively without requiring the full execution of an actual clustering algorithm. An extensive presentation of clustering algorithms is thus outside the scope of the paper. In Section 8, results from k-means [5] and spectral clustering [6] are used, but Cluster Sculptor is agnostic of a specific algorithm, and just requires to be fed with a set of labels mapping the data collection under consideration.

4. Cluster Sculptor Overview

In Figure 5, we overview the Cluster Sculptor interface, and the provided inspection and interaction tools. The front-end is implemented as a one-page AJAX application, and R runs as a server in the back-end. A node.js [38] middleware manages routine calls by the front-end, and all the required data exchanges. R workspace files act as databases, and maintain a consistent state for the application. This section is aimed at giving a first glance of the interface, and the actions that can be performed using it. Details about label diffusion, dissimilarity transforms, or interactive t-SNE updates, are presented afterwards, respectively in Sections 5, 6 and 7.

The interface is centered on a scatterplot view (Figure 5e), that displays the $N \times 2$ matrix $\mathbf{Y}$ output by t-SNE steps. When a data set is selected using the data loading controls (Figure 5a), the data structure needed for t-SNE to run (e.g., the $\mathbf{P}$ matrix from Equation (1)) is initialized asynchronously in the back-end, and the user is informed of the currently pending operation. The user may then run or stop the t-SNE process on demand. A label vector, selected from the available clustering results in the data loading controls, triggers the overlay of categorical colors (defined following the recommendations of Harrower and Brewer [39]) on the current scatterplot. The controls also feature several convenient overloads, such as cached t-SNE results, or updated dissimilarity matrices.

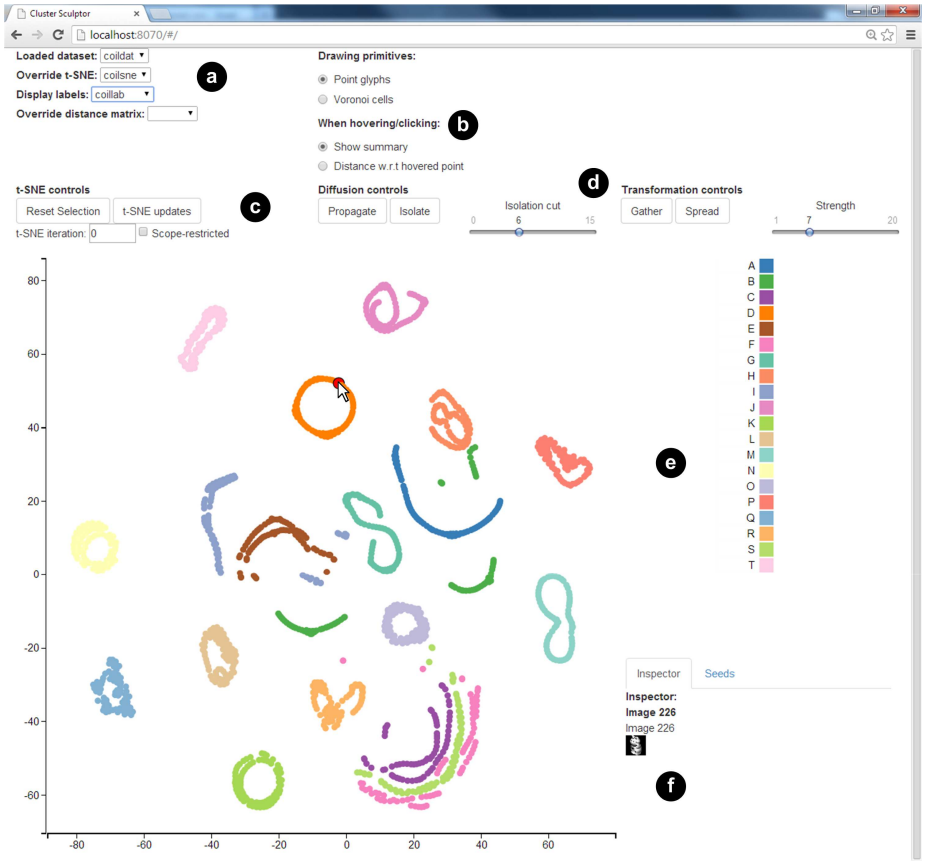

Figure 5: Overview of the Cluster Sculptor interface. *a*: Controls to load data from the back-end. *b*: Controls of the element appearance in the scatterplot, and the behavior when hovering. *c*: Scatterplot update controls. *d*: Label diffusion and dissimilarity transform controls. *e*: The scatterplot, and the associated interactive legend. *f*: The data inspector, stacked to the interactive list of currently selected seeds.

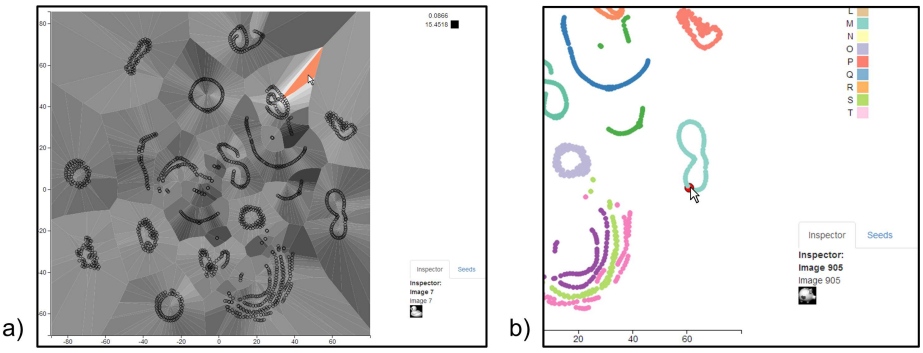

Figure 6: *a)* Voronoi diagram view of the COIL-20 2D projection. The ProxiViz tool is activated, and overlays distances in the HD space w.r.t the hovered point. *b)* Classical scatterplot view, supported by the data inspection tool. The hovered point is highlighted, and the data inspection panel is refreshed interactively.

The controls from Figure 5b influence the appearance and behavior of the scatterplot view content. The user may switch any time between a classical scatterplot view, as shown in Figure 5e, or a Voronoi diagram view (e.g., Figure 6a and 19b). Both visual primitives support two hovering interactions: either displaying a summary of the hovered point in the data inspector (Figure 6b), or map the pairwise distances in the HD space relative to the hovered point (Figure 6a). This interaction in largely inspired by the ProxiViz tool [27], and is a valuable asset to assess the faithfulness of the visualization. We incorporated slight modifications to the tool, by mapping the cluster categorical color to the hovered point (instead of white in ProxiViz), and by dynamically substituting the categorical legend by the distance bounds when the mouse is inside the scatterplot (see Figure 6a).

The behavior of the t-SNE updates can be controlled via the scatterplot update controls (Figure 5c). Specifically, the t-SNE phase (i.e., initial simulated annealing, or local optimization after a sufficient convergence) depends on the iteration count, that can be set when the t-SNE process is not currently running. A check-box controls the restriction of t-SNE updates to the current *scope*; this notion is explained to a greater extent below.

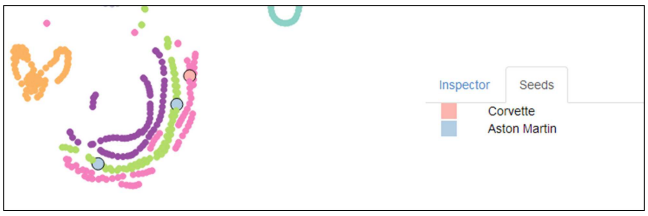

Figure 7: Panel listing the currently selected seeds. The labels can be interactively edited.

The label diffusion and dissimilarity transform controls are the prime tools for amending the currently displayed clustering and visualization (Figure 5d). The label diffusion techniques use the distribution of the data in the 2D space to reshape the current set of labels. The dissimilarity transform facilities mirror them, by triggering the update of the HD-related dissimilarity matrix $\mathbf{D}$ underlying the t-SNE computation (e.g., see Equation (1)), based on the current set of labels. The outcome of such transforms can be immediately visualized using the ProxiViz tool, and affect subsequent interactive t-SNE steps.

In addition to hovering, element glyphs can be clicked to define label diffusion *seeds*. The currently selected seeds are listed (see Figure 7), and can be interactively edited, for example to define multiple seeds for a single label value. A legend is adjoined to the scatterplot view, and lists the mapping between glyph colors and cluster labels. Labels can be interactively edited, e.g., to give a semantic value to a cluster, or to trivially merge clusters when typing an already mapped label. The color patches are also interactive: clicking them highlights the associated cluster in the visualization. Several clusters may be selected simultaneously, and the selection is persistent beyond modifications of the label vector. This can be useful to restrict the comparative analysis of several clustering results (see Figure 8).

Beyond implementing a simple visual filter of the data according to the cluster labels, the selection mechanism also implicitly defines a *scope*, i.e., a restriction of the elements according to the selected clusters. When a scope is active, operations such as label diffusions, dissimilarity transforms, or even t-SNE up-

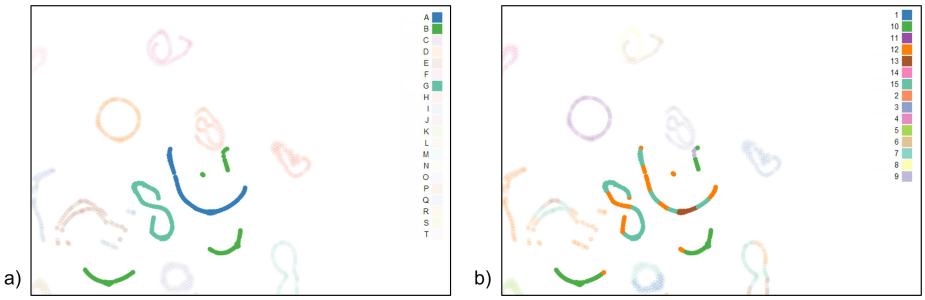

Figure 8: *a)* Example of scope restriction. The selected clusters are highlighted by lowering the alpha channels of non-selected ones in the scatterplot and in the legend. *b)* When updating the label vector, the current scope is maintained. The user may reset the selection when this specific scope is not needed any more.

dates (if the control in Figure 5c is checked) occur on this restriction, without affecting other 2D elements, or their underlying dissimilarity information. With this mechanism, a user can work iteratively on parts of the projection, without excessive discontinuities with respect to his mental map, and limiting the cognitive burden and visual clutter. This also limits the computational burden of diffusion and transform operations.

5. Label Diffusion From Selected Seeds

As envisioned in the previous section, reference elements, i.e., *seeds*, can be selected directly in the scatterplot. They can be seen as tentative labels, associated to a single element at the moment of the selection, and that can be later diffused in a chosen scope, according to a specific mechanism. We see them as a semi-automatic alternative to the classical rectangular and lasso selectors, that uses the 2D distribution of elements to update the clustering structure. We implemented two diffusion mechanism, that are of complementary use, as shown experimentally in Section 8. The probabilistic *label propagation scheme*, described in Section 5.1, is inspired by the semi-supervised learning literature, and defines a comprehensive diffusion of the seeds in the current scope. The *isolation scheme* (Section 5.2) uses the Minimum Spanning Tree of the current

scope. A breadth-first exploration in cuts of the tree induces a limited diffusion of the seeds.

5.1. Label Propagation

This operation is an interactive adaptation of the label propagation technique [32], taken from the semi-supervised learning literature. In this section, for mathematical convenience the seeds selected in the current scope are assumed to take values in $1 \dots C$. This set of seeds is described here as the labeled set, whereas the remainder of the scope is the unlabeled set. The goal of the operator is to propagate the known labels exhaustively.

Consistently to the definitions in Section 3, let us define $\mathbf{Y}^s$ as the restriction of $\mathbf{Y}$ to the current scope, and $\mathbf{Y}^s_L$ (respectively $\mathbf{Y}^s_U$) the labeled (respectively unlabeled) subset of l (respectively u) data elements. For further convenience, elements in $\mathbf{Y}^s$ are permuted so that:

$$\mathbf{Y}^s = \begin{bmatrix} \mathbf{Y}^s_L \\ \mathbf{Y}^s_U \end{bmatrix} \tag{3}$$

Each row in $\mathbf{Y}^s$ has a respective counterpart in $\mathbf{Z}^s$, the probabilistic labels for the elements in the scope. The C columns of $\mathbf{Z}^s$ mirror the C seed values, with $\mathbf{Z}^s_{nc}$ the probability of element n having label c, and $\sum_{c=1}^{C} \mathbf{Z}^s_{nc} = 1$. Values in $\mathbf{Z}^s_L$ are thus set to binary values, so as to reflect the seeds set by the user (see Figure 7 for an example).

The propagation scheme metaphorically lets labels *jump* from element to element. It follows the intuitive idea that similar elements are likely to have similar probabilistic labels. Instead of elements described in a vector space, the propagation algorithm thus uses similarity values, ranging in $[0, 1]$, such as obtained by a RBF applied on rows of $\mathbf{Y}^s$. We note $\mathbf{S}$ the matrix in which $\mathbf{S}_{nn'}$ is the RBF of the difference between rows $\mathbf{Y}^s_{n.}$ and $\mathbf{Y}^s_{n'.}$. To compute $\mathbf{S}$, the RBF function is parameterized using the method of Karatzoglou et al. [9]. We define a probabilistic transition matrix as:

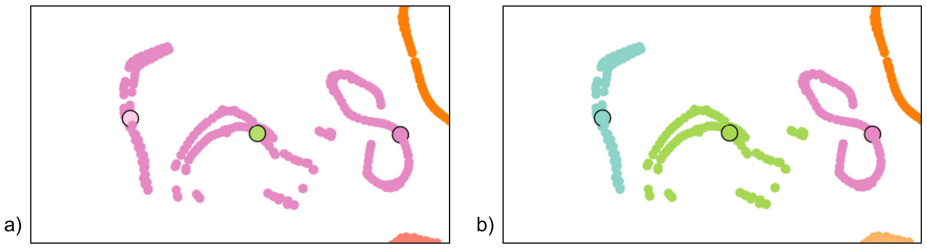

Figure 9: Example of propagation of 3 seeds. A subsample from the t-SNE projection of COIL-20 is used.

$$\mathbf{T}_{nn'} = P(n' \to n) = \frac{\mathbf{S}_{nn'}}{\sum_{m=1}^{l+u} \mathbf{S}_{mn'}}, \tag{4}$$

with $\mathbf{T}_{nn'}$ being the probability of jumping from element n' to element n. Mirroring the permutation defined by Equation (3), $\mathbf{T}$ has the following block structure:

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}_{ll} & \mathbf{T}_{lu} \\ \mathbf{T}_{ul} & \mathbf{T}_{uu} \end{bmatrix} \tag{5}$$

To avoid notation clutter in later steps, $\mathbf{T}$ is assumed to have its rows normalized to a unit sum in the remainder. The propagation then proceeds by iterating $\mathbf{Z}^s = \mathbf{T}\mathbf{Z}^s$ until $\mathbf{Z}^s$ converges. In [32], the authors showed that this algorithm converges to a unique fixed point, and that $\mathbf{Z}^s_U$ can be initialized arbitrarily without influence on this fixed point. Specifically, the converged solution is shown to be:

$$\mathbf{Z}^s_U = (\mathbf{I} - \mathbf{T}_{uu})^{-1}\mathbf{T}_{ul}\mathbf{Z}^s_L \tag{6}$$

An example application of this operator is shown in Figure 9. Applying Equation (6), the operator thus diffuses the C seeds exhaustively to the data set $\mathbf{Y}^s$, following the topological information provided in $\mathbf{S}$.

5.2. Isolating Manifolds Using the Minimum Spanning Tree

Whereas the propagation operator described in Section 5 diffuses exhaustively the seeds to the scope, Cluster Sculptor also features a more exclusive

tool, that *isolates* local manifolds in the scope. It also proceeds from the seeds, and similarly to the label propagation scheme, diffuses them according to the vicinity between elements in the visualization, as implied by the pairwise distances in the 2D space.

Let us consider the complete graph over the elements in the scope, and, borrowing notations from Section 5.1, define $E_{nn'}$ the edge between elements $\mathbf{Y}^s_{n.}$ and $\mathbf{Y}^s_{n'.}$. This edge is weighted with $w_{nn'}$, the Euclidean distance between $\mathbf{Y}^s_{n.}$ and $\mathbf{Y}^s_{n'.}$. The Minimum Spanning Tree is then defined as the subgraph that connects all elements with minimal summed weight (see Figure 10a). It is usually obtained using Kruskal's algorithm [40], or Prim's algorithm [41]. We also define a cut of this graph at w_{cut} as its restriction such that:

$$E_{nn'} \in \text{cut} \leftrightarrow w_{nn'} \leq w_{\text{cut}} \tag{7}$$

Prior to triggering the *isolate* control in the interface, the user may adjust w_{cut} using the *isolation cut* control (Figure 5d). Updating the cut of a MST is fast, and can be performed interactively (see Section 9 for notes on MST-related computational complexity).

A set of connected components can then be extracted from the MST cut using simple breadth-first explorations. Depending on the seed parametrization, triggering the *isolate* control has a variable effect:

- if no seed is selected, root nodes for the breadth-first search are chosen randomly in the scope as long as the scope has not been completely processed. All extracted connected components become new clusters, replacing those that defined the scope,
- if at least a seed is selected, new clusters are formed with the connected components extracted by using the respective seed as root nodes of the search. Depending on user preferences as set in the controls, the remaining elements keep their label as prior to scope selection, or are regrouped in a new cluster.

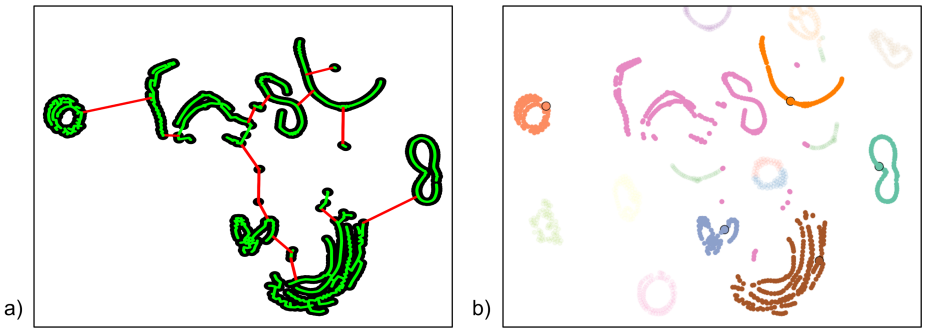

Figure 10: *a)* MST of a subset of a COIL-20 t-SNE projection. The cut for $w_{\text{cut}} = 5$ is shown in red. *b)* For the highlighted set of seeds, the resulting new clusters are shown. According to the user parametrization in this case, connected components with no associated seed are gathered in a cluster.

The result of an isolation cut is shown in Figure 10. Be they defined as in the original clustering results, or resulting from label diffusions such as shown in this section or in Section 5.1, the current cluster labels serve as cues for updating the dissimilarity matrix underlying the visualization. This mechanism is presented in the next section.

6. Dissimilarity Updates

After updating the cluster labels using tools described in the previous sections, or simply on the account of the clustering results initially loaded, a user may wish to update the 2D projection, so as to better reflect the clustering information. Multiple reasons can be invoked:

- the clusters have too shallow borders,
- clusters may be multimodal in the 2D space,
- clusters may have unsatisfactory neighborhoods, irrelevant to user knowledge.

Overall, the user could either want to inject knowledge absent from the raw features used to compute the dissimilarities underlying the 2D projection,

or emphasize cluster boundaries. In this paper, we choose to translate these preferences in updates to the dissimilarity matrix $\mathbf{D}$ (see Equation (1)). As t-SNE is exactly about mapping the distribution of values in this matrix to $\mathbf{Y}$, modifying the dissimilarity matrix is expected to be progressively reflected in the 2D projection as a side effect of t-SNE updates.

In this section, we focus on the dissimilarity modification process, and identify two ways a user would want to update the distribution of the clusters in the visualization space: *gathering* them, i.e., making the clusters in the current scope closer to each other, and *spreading* them, i.e., emphasize the separation between the clusters in the scope.

Let us consider the complete weighted graph implicitly defined by the matrix $\mathbf{D}$. We choose to restrict dissimilarity updates to the bipartite subgraphs between clusters in the current scope. This choice preserves the topology of components in the visualization, and the efficient use of the 2D space by t-SNE, which prevents elements from excessively packing together.

In Cluster Sculptor, both operations are implemented as different parameterizations of the cumulative beta distribution function, $P_{\text{beta}(\alpha,\beta)}$. Considering a scope with two label values l_1 and l_2, this function is applied on the bipartite graph between l_1 and l_2 (i.e., its edges link elements with label l_1, to elements labeled with l_2) weighted by the respective matrix cells in $\mathbf{D}$. Generalizing this principle to any number of labels, $\mathbf{D}$ is filtered according to $P_{\text{beta}(\alpha,\beta)}$:

$$\mathbf{D}^{\text{new}}_{nn'} \leftarrow P_{\text{beta}(\alpha,\beta)}(\mathbf{D}_{nn'}), \tag{8}$$

for n, n' referring to an edge in the bipartite graph. To preserve the structure of local manifolds initially learned by t-SNE, we enforce monotonous updates of the dissimilarities (i.e., $\mathbf{D}_{nn'} > \mathbf{D}_{mm'} \rightarrow \mathbf{D}^{\text{new}}_{nn'} > \mathbf{D}^{\text{new}}_{mm'}$), by constraining either α or β to be 1. The resulting family of functions is shown in Figure 11. This family of functions bijectively maps the unit domain to itself. While maintaining the order of dissimilarities, this guarantees valid dissimilarity values after the filter application. As Figure 11 shows, curves for $\alpha > 1$ (resp. $\beta > 1$) tend

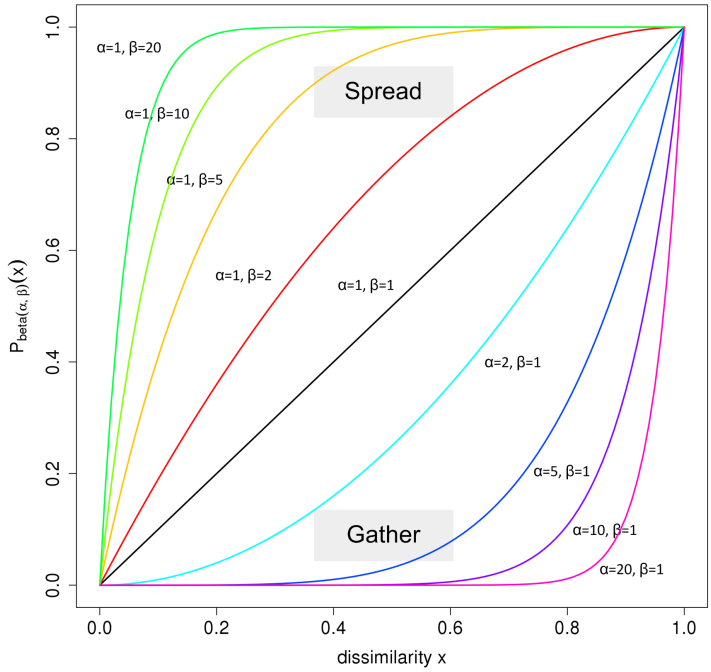

Figure 11: The family of cumulative beta distribution functions with either α or β set to 1. The identity (both parameters set to 1) is indicated as a reference.

to reduce (resp. increase) the dissimilarity between elements, thus effectively *gathering* (resp. *spreading*) them.

The impact of the operation can be more clearly interpreted in Figure 12, showing the absolute increase or decrease of dissimilarity caused by applying P_{beta} for a range of α and β values. Figure 12a highlights that enforcing valid similarities causes the functions to be bounded from above and below. Then Figure 12b displays the *closeness to the bound* of the dissimilarities after the transform, i.e., the relative influence of the operation. As gathering and spreading functions are mirroring, these closeness profiles are similar in both cases. From this plot, it is clear that increasing α or β tightens a wider range of similarities to the minimal or maximal valid dissimilarity. The choice of the P_{beta}

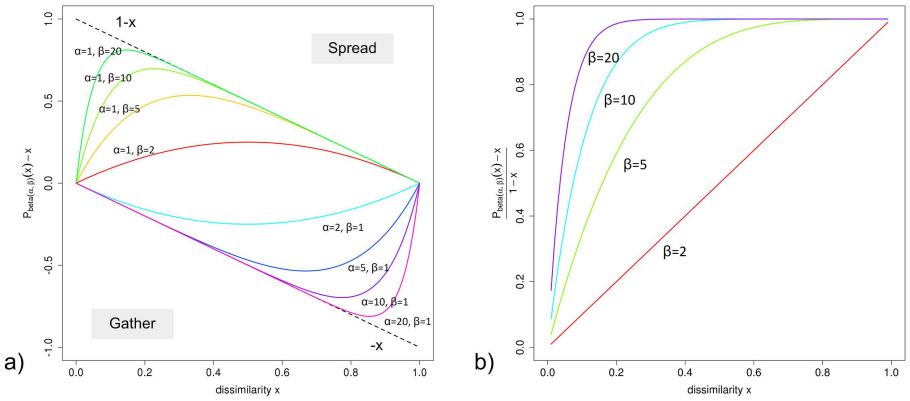

Figure 12: *a)* Plot of $P_{\text{beta}(\alpha,\beta)}(x) - x$ with respect to x. $1 - x$ and $-x$ bound this family of functions from above and below. These bounds ensure transformed similarities remain valid. *b)* Closeness to the bound of the spreading functions. The curves for the gathering functions are similar, up to varying α, setting $-x$ in the ratio denominator, and thus varying from 1 to 0.

family of functions is motivated by the behavior at the vicinity of $x = 0$ in Figure 12b: trying to spread elements that are extremely similar can be disruptive for the visualization. P_{beta} smooths this problem, and ensures the manifolds underlying $\mathbf{D}$ are preserved to some extent. α or β can then be seen as the *sharpness* of this smoothing.

In Figures 11 and 12, dissimilarities are assumed to range in the unit interval, which may be overly restrictive in practice. Considering an arbitrary domain $[a, b]$, $P_{\text{beta}(\alpha,\beta)}$ can be rescaled according to Equation (9). This operation preserves the required properties, i.e., it bijectively ranges in $[a, b]$, and preserves the order of dissimilarities, with minimal and maximal bounds respectively at a and b.

$$P^{[a,b]}_{\text{beta}(\alpha,\beta)}(x) = (b-a)P^{[0,1]}_{\text{beta}(\alpha,\beta)}\left(\frac{x-a}{b-a}\right) + a \tag{9}$$

In practice, we use this rescaling to ensure two desirable properties:

- the modified dissimilarity should not exceed the current maximal dissim-

ilarity in $\mathbf{D}$,

- when gathering two clusters, the decreased dissimilarity should account for the internal cohesion of the clusters.

The first property amounts to set b statically for a given data set. When spreading clusters, a plays a marginal role, and can be left to 0. When gathering clusters, we choose to set a to some quantile of the distribution of dissimilarities internal to the clusters (i.e., outside the bipartite graph). In the experiments described in Section 8, we use the 5% quantile.

The next section describes how we adapt the t-SNE algorithm to discontinuous changes in the matrix $\mathbf{D}$ such as described above, and smoothly render this change to the user in an animated fashion.

7. Updating the t-SNE

Being a gradient-based method, t-SNE is quite expensive to compute afresh [34]. For instance, it took approximately 11 minutes, on an 8 core machine, to compute 1000 gradient steps for the COIL-20 data set (1440 elements, 30 principal PCA features). Assuming a perturbation of the dissimilarity matrix $\mathbf{D}$ such as described in Section 6, computing a t-SNE projection from the ground up at each user interaction is clearly not acceptable.

The available R implementation [42] is a classical batch algorithm. We adapted it to support independent step executions, and maintain an internal state for the algorithm on the R server, as our interactive scheme requires. Convenience accessors are also implemented to support updates of any part of the internal state, e.g., modifying its parametrization, updating the dissimilarity matrix (see Section 6), or overloading the current 2D projection (see Figure 5a). Let us note that updating $\mathbf{D}$ triggers the re-computation of the $\mathbf{P}$ matrix, along with the re-estimation of the σ_n parameters.

In its local optimization phase, t-SNE can be qualified as an *anytime* method, with each iteration being a smooth update of the preceding 2D layout, following

a physical metaphor. Consequently, discontinuous updates of $\mathbf{D}$ are supported by continuously updating the 2D position matrix $\mathbf{Y}$ towards its new convergence point, ensuring the visual stability of the projection. Furthermore, the convergence of gradient-based methods such as t-SNE is typically difficult to assess with automatic means. A user is actually much more qualified to assess visually when the projection is stable enough, and can take this decision using the t-SNE controls.

Spring-based layouts are prone to get stuck in local optima [13]. Worse, a complete graph is involved in t-SNE, which causes a high effort against any live update of $\mathbf{D}$. In the context of the batch t-SNE, this problem is handled by the simulated annealing phase (i.e., *early exaggeration and compression* in [14]). Though efficient in the latter case, doing so when live updating $\mathbf{D}$ would be visually disruptive, and thus inadequate.

Instead, we implemented a *scope restriction* of the gradient steps, i.e., the gradient sums are performed only over elements in the selected scope, ignoring the rest of the data set. This option can be activated on user demand (see t-SNE update controls in Figure 5c). As experimentally shown in Section 8, this option is effective at quickly updating the position of clusters in the scope, but their new positions may be conflicting with others outside the scope. The user then has to find a satisfactory configuration by alternating scope-restricted and classical update sequences. If necessary, this might be supplemented by including the conflicting cluster in the scope, and using an additional spread operation.

Via detailed experimental scenarios on real data, the next section shows how the assembly of tools described above can be used to amend an initial 2D projection, and associated clustering results. We specially put an emphasis on how the tools are used to harness the differential information carried respectively by the 2D projection and the HD clustering results.

8. Experimental Scenarios

This section describes three scenarios of a user that iteratively amends initial 2D projections and their associated clustering using Cluster Sculptor. Rather than focusing on low-granularity tasks, we show on a higher level how a user can use the tools to get some insight of the data, and update the visualization and clustering structure according to his or her findings.

We use the two image collections: COIL-20 and MNIST handwritten digits. COIL-20 contains 1440 images, each having 32x32=1024 pixels. The images are photographs of 20 objects rotated by all possible angles modulo 5°. A class is thus made of the 72 images of a given object. The MNIST collection is made of 60000 images, with each 28x28=784 pixels. The images are variants of handwritten digits, thus forming 10 even classes.

Additionally, we used a biological data set, describing 31483 bacterial DNA fragments over 8 numerical features. These fragments were sampled from digesters, where biodegradable waste is stored, and slowly degraded by bacteria, to eventually generate biogas. This data set was handed by biologists, in the context of a beginning collaborative exploratory analysis of the data. We further refer to this data set as *biogas*.

As MNIST and biogas are too big for the current Cluster Sculptor implementation (see Section 9 for a discussion on complexity issues), we sampled randomly respectively 1500 and 3000 elements in these data sets. After removing dimensions that carry no information in each collection (i.e., zero variance), SVD is applied on these, thus removing effects caused by potential correlations among variables. Van der Maaten and Hinton prescribe to retain the coordinates on the 30 principal axes as element representatives [14]. For COIL-20, this amounts to retaining 88% of the data set variance, as indicated by the singular values. For MNIST, 30 principal axes summarize less than 80% of the variance. We thus extended the size of the axes set to 50, that explain 83% of the variance. biogas has only 8 variables, so all axes can be retained.

The next subsections present interaction scenarios specific to each data set.

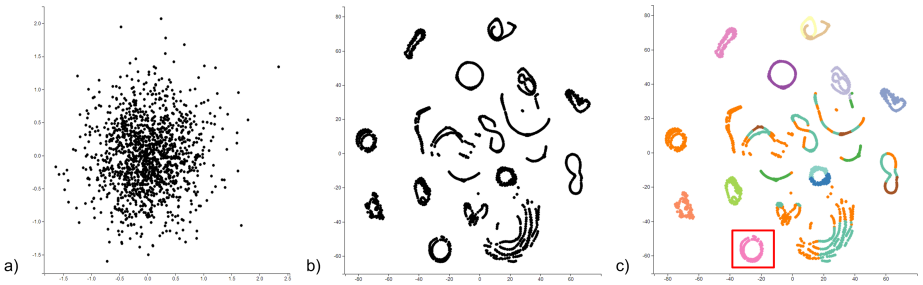

Figure 13: *a)* Initial configuration of t-SNE. *b)* 2D projection after 1000 t-SNE iterations. *c)* Overlay of the spectral clustering result. Cluster 14 is highlighted in red.

In the scenarios, we refer to clusters with integer IDs, as we have no prior on their meaning. If desirable, these default IDs can directly be edited in the interactive legend (see Section 4).

8.1. COIL-20

Selecting COIL-20 in the data loading controls (see Figure 5a) triggers the initialization of the t-SNE internal state. The initial projection is eventually displayed (see Figure 13a). The user then clicks the *t-SNE updates* button to start iterating t-SNE steps, that proceed until the *t-SNE stop* button is clicked. The result after 1000 iterations is shown in Figure 13b.

The user then selects a label vector obtained with the spectral clustering algorithm [6], on the HD representation of the data set. At the time of the clustering process, the user had no clue about an adequate number of clusters, and parametrized his algorithm with 15 clusters (Figure 13c).

The clustering algorithm visibly captured some clear structure in the data set, e.g., Cluster 14 highlighted in Figure 13c, but also fails to recover clear patterns discovered by the t-SNE projection. Maybe resulting from a bad parametrization, some clusters aggregate several unrelated groups (e.g., see Clusters 12, 13 and 15 on Figure 14a). The adjusted Rand index [43] of the clustering w.r.t. the ground truth labels, of 33%, reflects these observations to some extent. The user decides to use the tool to improve this initial guess.

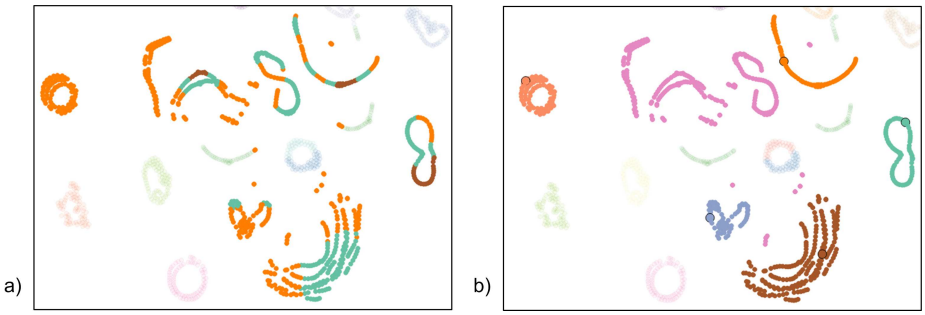

Figure 14: *a)* Highlight on Clusters 12 (orange), 13 (brown) and 15 (green). *b)* Resulting clusters after isolation of manifolds from the highlighted seeds. The remaining elements in the scope become Cluster 18 (in purple).

Before modifying the 2D projection, the user wants to quickly reorganize Clusters 12, 13 and 15 using the label diffusion tools. He or she defines a scope for this operation, by clicking the related color patches in the legend. He or she then selects 5 seeds, purposely to isolate the 5 manifolds these seeds lie on (Figure 14b). The remainder of the scope is to be regrouped in Cluster 18. After adjusting the *isolation cut* slider, and observing interactively the resulting isolations, the users retains 5 for this parameter, leading to the result shown in Figure 14b. It is worth noting that the short sequence of actions taken until now (a dozen of clicks and few slider interactions) led the adjusted Rand index of the clustering to 69%.

The user then notices Cluster 10, with 3 distinct manifolds in the current clustering. To restrict his or her attention, the user defines a scope on Cluster 9, 10 and 18. Elements of the 3 components are first hovered over, using the data inspector as a confirmation that they indeed are related to the same ground truth object (see Figure 15a). The user then activates the Voronoi diagram view, along with the ProxiViz tool. In the scope restriction, he or she searches potential tears that would suggest a re-unification of the 3 components of Cluster 10. The lighter the gray shade, the closer the mapped element in the respective Voronoi cell is to the reference element in the HD space. Using Cluster 10 and

Figure 15: *a)* Scope containing Clusters 9 (orange), 10 (green) and 18 (purple). For each of the three components of Cluster 10, we attach the respective data inspector to one of its elements, highlighted in brown. *b) and c)* ProxiViz view of the scope, for two reference (hovered) elements, with Cluster 10 components highlighted in green. The scope extent is indicated using alpha blending. The Voronoi cells of non-hovered elements in the scope are mapped with gray shades, indicating the distance in the HD space w.r.t. the reference element. A tear occurs if an element is far to the reference element in the projection, but close in the HD space.

the neighboring ones as a cue for an inspection with the ProxiViz tool, manifold tear artifacts are clearly identified (Figure 15b and c). The data inspector is then used to confirm the separated components are indeed related to a single object. The user decides to amend the 2D projection, so that Cluster 10 is displayed as a single manifold.

First, as some confusion errors seem to affect Cluster 10, the user first selects it, along with Clusters 9 and 18, also involved in the confusion. Three seeds then mark the components of Cluster 10, whereas one is sufficient for Cluster 9. The user isolates Clusters 9 and 10 after having adjusted the isolation cut, the remainder of the scope being again assigned to Cluster 18 (Figure 16a).

Setting the sole Cluster 10 as the scope, the user tries to run restricted t-SNE steps to see if the distinct components can be reunited this way. Two of the three components merge quickly, after approximately 50 iterations. However, the algorithm stabilizes to a configuration with still two distinct components, with one of the latter in conflicting positions with another cluster (Figure 16b). Striving for the reunion of the two components, the user defines a seed on each component, and diffuses them with the propagation tool. A gathering trans-

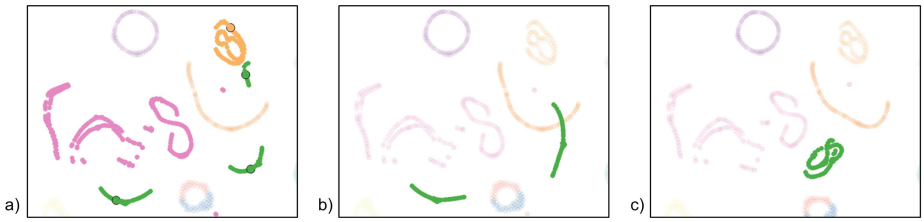

Figure 16: *a)* Cluster 10 after correction of the clustering errors using the label diffusion tools from three seeds. *b)* Result after 50 scope-restricted t-SNE steps, without a dissimilarity transform. *c)* Resulting projection after a gathering transform and 50 more scope-restricted steps.

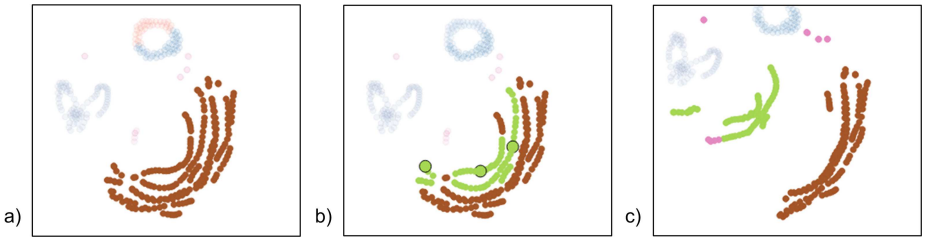

Figure 17: *a)* Highlight on Cluster 13. *b)* Isolation of a subset of the cluster using three seeds. *c)* Resulting projection and clustering, with the clearly separated Clusters 13 and 19. An outlying subset of Cluster 18 was spontaneously gathered to the new cluster.

form is then applied to this scope, with moderate sharpness (3), to encourage the merger of the temporary clusters as a single manifold. Subsequent t-SNE steps on this scope indeed quickly lead to a reunion, with its internal structure reflecting the underlying data topology. The temporary clusters are then trivially merged, and restored as Cluster 10, using simple edits in the legend (see Section 4). Few switches between scope-restricted and unrestricted t-SNE steps lead to a proper cluster separation (Figure 16c).

The user then notices Cluster 13, standing out from the others with its distinct shape (Figure 17a). It actually regroups the images of three cars. As car images differ more by their pose than by the patterns overlaid on them, t-SNE has laid them in close manifolds.

Distinguishing the three cars happens to be difficult for clustering algorithms, leading to a single cluster. This layout does not conform to user preferences, rather expecting a single car in each cluster. The user thus decides to modify the clustering and 2D projection to have one of the cars standing out from the others. By setting three seeds, and testing several isolation cuts with slider adjustments, he is able to derive a close to correct cut of one of the cars in the original Cluster 13 (see Figure 17b). He or she then applies a spread dissimilarity transform to make the separation clearer. Interestingly, as a side effect, a part of Cluster 18 referring to the same car as the new Cluster 19, but initially lying out of Cluster 13, is automatically connected to the new cluster (see Figure 17c). Label 19 can then be quickly diffused, by isolating a seed on the part of Cluster 18 highlighted in Figure 17c, and performing a trivial merger by setting 19 as the legend ID of the freshly isolated component.

8.2. MNIST Handwritten Digits

Distinctly from the COIL-20 scenario, the user loads a cached t-SNE projection, that has been computed off-line on the server. The spectral clustering algorithm has also been processed off-line using the HD representation of the MNIST collection, and the result is overlaid on the projection (see Figure 18a).

Using the inspector, the user sees that t-SNE is rather good at identifying regions in the data set, and some pieces of manifolds (e.g., of the digit *1* at the center of the projection, or *7* in Cluster 12, see Figure 18b). However, maybe due to the high variability of digit drawings, there is no clear boundaries between clusters. The visualization could thus be improved. In addition, the HD clustering performs badly, with initially 13% as Rand index.

Some clusters are clearly irrelevant, because their elements are scattered almost evenly in the visualization, and inspection confirms their elements are rather unrelated (e.g., Clusters 1, 2, 8, 9 and 15, see Figure 19b).

The user first focuses on the center of the projection, that looks more densely populated. A scope is thus defined by the selection of Clusters 11 and 3. Interestingly, despite a bad general performance of the clustering, these two specific

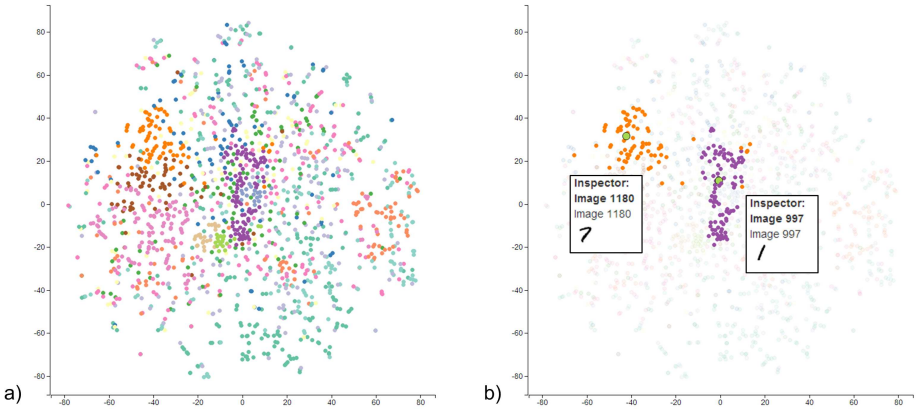

Figure 18: *a)* t-SNE projection of the MNIST image collection, with the spectral clustering result overlaid as categorical colors. *b)* Cluster 11 (purple, in the center) contains almost exclusively the digit *1*. Cluster 12 (orange, on the left) is mostly populated by samples of the digit *7*.

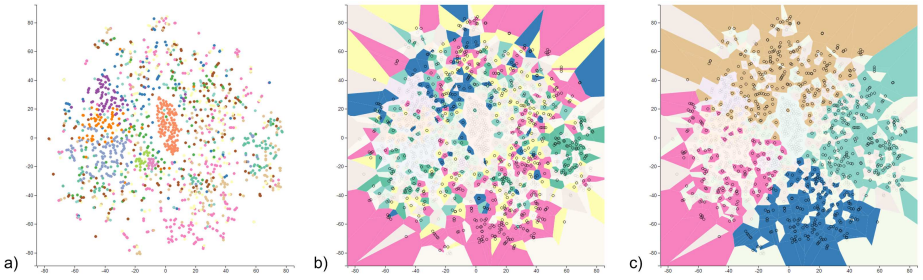

Figure 19: *a)* 2D projection after the trivial merger of Clusters 11 and 3. The spreading transform emphasizes the boundaries of the new cluster (center of the projection) *b)* Highlight of Clusters 1, 2, 8, 9 and 15. The Voronoi visualization emphasizes the distribution in the 2D projection. *c)* The same scope, after propagation of 4 seeds.

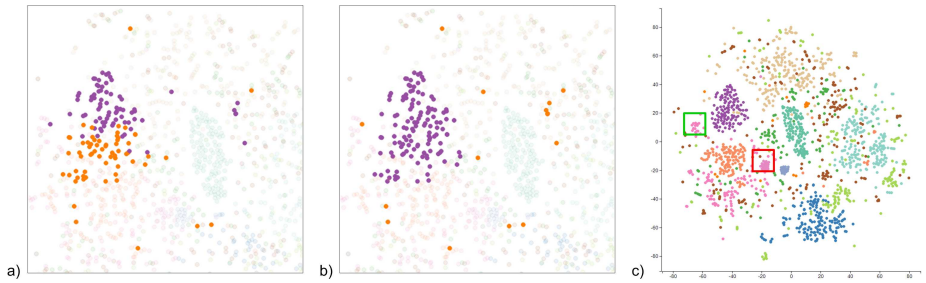

Figure 20: *a)* Clusters 12 (purple) and 13 (orange) before isolating the central manifold. *b)* The central manifold becomes Cluster 12 (purple), and the remainder populates Cluster 13 (orange). *c)* Visualization after applying a general spreading transform, and 50 t-SNE iterations. Cluster 6, highlighted in red, contains exclusively *1* digits, and the subset of Cluster 2 highlighted in green gathers samples of the digit *4*.

clusters have captured a pattern absent from the projection: they seem to delimit a region for the digit *1*, that was not clear at all from the sole projection. The user wants to emphasize this pattern. First, Clusters 11 and 3 are trivially merged via a simple legend edit. He or she then applies a spreading transform, with a sharpness of 5, to this new cluster against all others (Figure 19a).

Judging that most of the clustering appears as visual clutter, the user then focuses on Clusters 1, 2, 8, 9 and 15, that exhibit a poor locality w.r.t. the projection. In a scope defined by the selection of the latter, he or she uses the inspector, and the marginal distribution of the scope in the visualization, to define the seeds for four rough regions, and diffuses them using the propagation tool, leading to more localized clusters (see Figure 19c).

The user then notices Clusters 12 and 13, with both a more densely populated part and few outliers. Beyond their apparent proximity, inspection reveals they are both related to the same ground truth digit (*7*). After trivially merging them with a legend edit, the user isolates the central manifold from the outliers with a single seed selection. The new merged cluster takes the label 12, and the remainder populates its own small *residual* cluster 13 (Figures 20a and b).

Finally, to emphasize the borders between the current clusters, the user

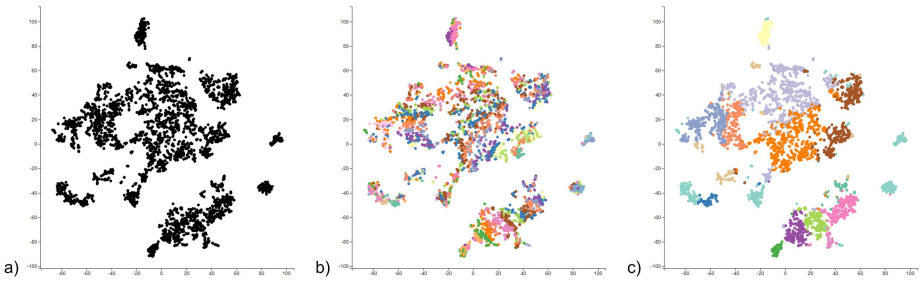

Figure 21: *a)* t-SNE projection of the *biogas* data set *b)* Overlay of the k-means clustering results. *c)* Overlay of the spectral clustering results.

applies a general spread transform (i.e., of all clusters w.r.t. all others) with a significantly high sharpness (10). After 50 iterations, the visualization looks much clearer, with a stronger emphasis given to manifolds and potential outliers (Figure 20c). Actually, this operation emphasized patterns unnoticed beforehand, e.g., Cluster 6, and dense regions in Cluster 2 (see Figure 20c). The few interactions led to a significant improvement of the Rand index, now reaching 20%. This leaves an important margin for progression though.

8.3. biogas

Similarly to the MNIST scenario, the user first loads a cached t-SNE projection of the *biogas* data (Figure 21a). For this data set, two clustering results are available: a k-means output of 30 clusters, handed by the experts, and a spectral clustering output with the parametrization used in the two previous sections (15 clusters). Both results are overlaid on the 2D projection as categorical colors, respectively in Figure 21b and 21c.

Using the data loading controls to compare the two clustering results, the user immediately notices a clear distinction between them. Whereas spectral clustering results are quite closely related to the manifolds found by t-SNE, k-means finds a different kind of patterns, with clusters splitting across components (see Figure 22a). As the experts provided the k-means results, the user chooses to use them as a starting point, and expects to use his or her findings

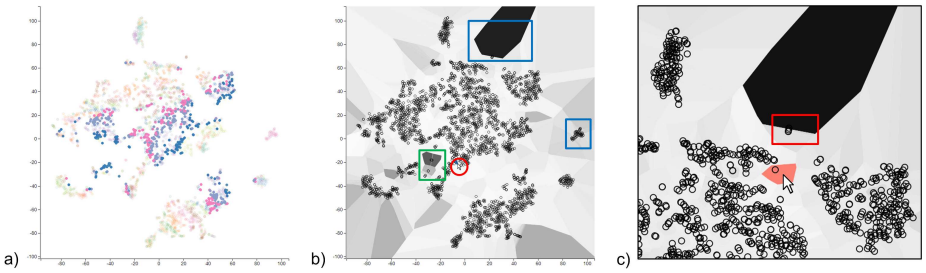

Figure 22: *a)* Highlight of Clusters 1 (blue), 4 (light blue) and 7 (pink). *b)* ProxiViz distance pattern when hovering over the element highlighted by a red circle. Cluster 14 is highlighted in green. Cluster 8, highlighted in blue, contains the component on the right, and the outlying element on top of the visualization. *c)* Among the four closely mapped elements (highlighted in red), from his dark shade, only one is clearly standing out.

as a support for further interactions with the experts.

Before actually amending the visualization, the user wants to inspect the projection using the ProxiViz tool along with the Voronoi diagram visualization (see Figure 22b). The user immediately identifies some outlying patterns, by their significantly darker shade (i.e., higher distance in the HD space), irrespectively of the hovered element (i.e., the element is far from everyone in the HD space). Quoting the experts, such outliers are likely to be representing viral DNA. With its three elements clearly standing out, Cluster 14 has already been identified as such using other visualization techniques, such as the parallel coordinates (see Figure 22b). The user found another pattern, more subtle, that could serve expert hypothesis formulation: with its dark shade, an element looks very clearly as an outlier, but was mapped very closely to three other elements that clearly do not look as such (Figure 22c). The user also notes that the suspect element belongs to Cluster 8, that looks a bit outcast in the visualization, as confirmed by the ProxiViz view (Figure 22b). Cluster 8, and the three elements mapped close to the suspect element, are interesting cues for further analysis. The user is tempted to gather the components of Cluster 8, but owing to the heterogeneity of the elements highlighted in Figure 22c, chances are their position is due to a subtle equilibrium with many other elements in

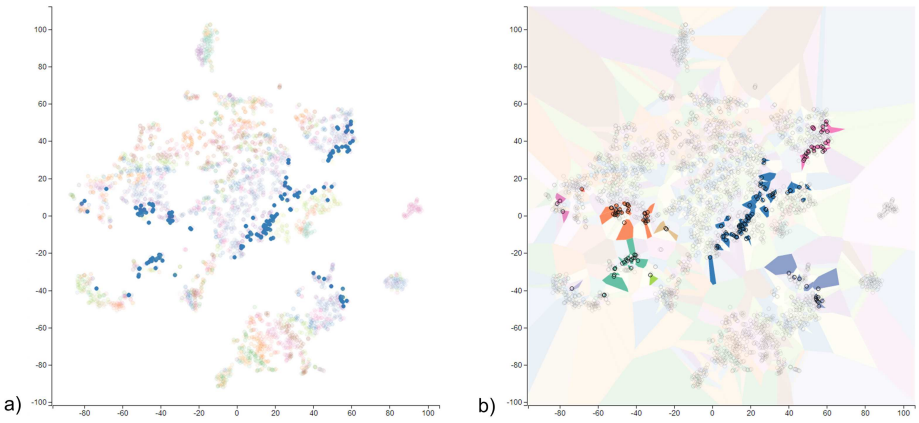

Figure 23: *a)* Scope containing Cluster 1 (blue). *b)* Voronoi visualization of the components isolated by the MST cut.

the 2D projection. In this context, gathering components in a restricted scope would require a very high sharpness, and could be widely disruptive.

The user then concentrates on Cluster 1, and wants to use the label diffusion and dissimilarity transform tools to regroup its components (see Figure 23a). After having defined Cluster 1 as the current scope, the user adjusts the isolation cut control to reflect the spread he or she wants to reduce (Figure 23b). No seed has been selected before, so the isolation effectively defined all components under the cut as new, temporary clusters (see Section 5.2). The user then gathers this new scope with a moderate sharpness (5). The temporary clusters that served as a cue for the gathering transform are then trivially restored as a single cluster (see Figure 24a). The user is not satisfied with the resulting visualization though, and would like to have the new Cluster 1 more clearly separated from its neighbors, Clusters 4 and 11. Using spread transforms parametrized with a medium sharpness (10), the visualization shown in Figure 24b is eventually obtained.

This scenario is part of a preliminary stage to the exploratory analysis. The user notes that the *biogas* data set lacks a summary representation for its elements. Plugging such representations in the data inspection tool would give

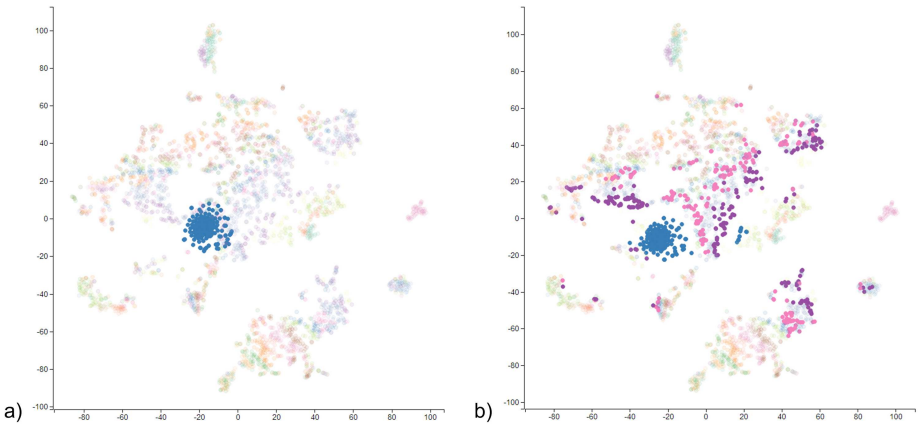

Figure 24: *a)* Cluster 1 (blue) after the gathering transform, and 200 t-SNE iterations. *b)* Cluster 1 after being spread from conflicting neighbors (Clusters 4 (pink) and 11 (purple)).

a crucial cue for further insight. The design of such a representation is thus to be shortly discussed with the experts.

9. Discussion

In the context of interactive systems, the computational complexity of the involved tools is critical. Four potential bottlenecks exist in Cluster Sculptor:

- the computation of the Minimum Spanning Tree (MST),
- the interactive update of the MST cut,
- the dissimilarity transform,
- the t-SNE updates.

The graph underlying t-SNE is complete, causing MST computation to be $O(N^2)$. This is not excessively harmful, as the MST has to be updated only after a dissimilarity transform. Also, as using an optimal MST is not critical to the application, increased speed could be obtained by filtering the highest range of values from the graph implicit to $\mathbf{D}$.

Updating the cut is linear w.r.t. the number of edges in the MST, thus linear w.r.t. the number of elements in the scope, as the MST of a graph defined over N elements has $N-1$ edges [40, 41]. This operation is generally quite fast, but can become slow if no scope is defined, and the number of elements exceeds 5000. However, the current implementation is naive, and dramatic improvements could be made using a properly sorted data structure.

The dissimilarity transform is $O(N^2)$ in the worst case. However, this step just applies P_{beta} on graph edges in the selected scope (see Section 6): each operation is thus considerably faster than those of the MST computation. For a higher interactivity, in case of very large data sets, piecewise computation could also be considered.

t-SNE updates are $O(N^2)$ [14]. Unlike dissimilarity transforms, they occur frequently, and their speed is the basis for the interactivity of the system: a user will perceive the succession of updates as a smooth move only if its frequency is sufficiently high. Some studies argue that a system would be perceived as interactive only if the response time is under 100 ms [12]. Unfortunately, in the current implementation, updating the position of 2000 elements in the 2D projection takes approximately 1 second. However, we did not investigate improvements to the baseline t-SNE steps, such as a random-walk approximation [14], or more recently $O(N \log N)$ variants based on the Barnes-Hut algorithm [44, 45]. Using this implementation, we could for example easily process the complete biogas data set (see Section 8).

A major issue in our system is to keep the user engaged when making changes to the projection. Heer and Robertson found that careful animation design has significant advantages for graphical perception of changes in the context of statistical plots [46]. When applying a dissimilarity transform, the visual tracking of objects is facilitated by the physical metaphor t-SNE follows (see Section 3). However, if the frame rate of t-SNE updates is too low, the sequence may not be perceived as a smooth move. This issue could be potentially alleviated by using interpolation and animation features available in d3.js [47], a library already widely used in the Cluster Sculptor prototype.

In this paper, we emphasized the use of the label diffusion tools, instead of classical lasso or rectangular selectors. In the current prototype state, this is the only supported selection mode. This design choice was motivated by our intuition, and should eventually be validated by a proper user study. In the meantime, classical selection modes could be implemented, as we sense the most efficient tool should depend on the properties of the selection to be performed.

We purposely chose to apply dissimilarity updates on the basis of the set of labels in the scope and not specific pairwise constraints, such as seen in metric learning approaches [48, 30]. We found that these methods require too high amounts of user-specified constraints to be really effective, and wanted to spare user efforts. For example, Martin et al. require approximately 30 pairwise constraints for their technique to be effective [48] and at least 20% of the collection has to be labeled in Basu et al. [31]. In Dis-Function, the authors choose to bias their objective function and give higher importance to the user specified constraints. We alternatively aimed at implementing smooth, non disruptive updates to the visualization, in a more robust fashion. Yet, a comparative analysis, or a way of combining the approaches should be investigated.

Our approach has a strong weakness: unlike the metric learning approaches, it is not currently able to generalize to new data points. This can be problematic in the context of data streams and further work is needed to support them.

Another important point regards the updated dissimilarity matrix, $\mathbf{D}$. It is explicitly stated as a dissimilarity matrix as its transform by P_{beta} may invalidate its initial Euclidean metricity. Fortunately, t-SNE is not limited by the metricity of values used in Equation (1), and is able to cope with any valid dissimilarity matrix.

In principle our approach is DR-technique agnostic and could be thus combined with any DR technique, as long as the above mentioned concerns are addressed. The choice of technique also depends on various factors such as the input format, distance matrix or coordinates in original space or the trade-off between computational complexity and projection quality [49, 50, 51].

Finally, related work such as iVisClustering [4] and Dis-Function [30] demonstrate the effective use of multiple coordinated views for a better insight and understanding of the data and model at hand. This direction has not been explored in our work, though some needs have been identified in Section 8.3 (e.g., an adapted inspector view or complementary insight by parallel coordinates).

10. Conclusion and Perspectives

This paper extensively described Cluster Sculptor, a novel interactive clustering system. Our system takes clustering results as an input, and extensively uses the t-SNE projection technique, its iterative nature, and the physical metaphor underlying it. Its contributed components pertain essentially to a framework of interactive tools, i.e., label diffusion and dissimilarity transform. Cluster Sculptor also grounds in the ability of t-SNE to adapt smoothly to discontinuous similarity updates, thus preventing invasions in the user's mental map. Detailed experimental scenarios, using real data sets, showed how a user could combine the diverse features of the system to amend clustering results, and the associated 2D projection.

This work opens to numerous perspectives. The tools described in the paper are prototypical, and could be improved in many aspects. For example, we included the ProxiViz tool [52] to answer the most urgent needs, and did not account for ProxiLens, the most recent evolution of the tool[53]. It features an interactive lens, that animates tears and false neighborhoods when hovering over any point in the visualization. Its integration could be considered in future work. We are aware that the system lacks basic HCI implementations, such as a history of interactions, or at least undo and redo features. Their implementation would help reduce user frustration, render the system more usable, and to some extent give some further insight of the data via the post-hoc navigation in interaction sequences.

The discussion has shown openings for a better scalability. If very large and dense data sets are targeted, Cluster Sculptor could be supplemented by tools

for interacting with dense patterns, such as a zooming feature, or a fish-eye plugin [54].

In the experimental section, numerous examples of MST cuts are given. As a complement to the isolation cut parameterization, visualizing and interacting directly with these graphs could give increased control to the user.

The dissimilarity transform is currently applied to all clusters in the current scope. For example, when gathering the components underlying a cluster, or spreading a cluster w.r.t. a subset of its neighbors, this design leads to tedious relabeling. A set of predefined sequences could be derived, e.g. a gathering operator including propagation and isolation steps.

The generalization of Cluster Sculptor to more data types (e.g., categorical) and DR techniques could be investigated in future work. Also, as identified in the discussion, the relationship, and potentially complementarity, with methods based on learning a distance function using pairwise constraints, is an interesting direction of research. Detecting the local relevance of HD features would be an interesting perspective to limit the disruptive tendency of classical metric learning approaches. Accounting for the latter aspects could widen the applicability of the system, e.g. to heterogeneous data types, or lessening the importance of noisy features.

Finally, the experimental scenarios involve a variety of low-level tasks (e.g., isolating a cluster with a MST cut). As evoked in the experimental section, we chose to put an emphasis on scenarios of use: but these tasks should certainly be evaluated on a unitary basis in the context of a classical user study.

Acknowledgements

We would like to warmly thank the anonymous referees for their extensive reviews. They highlighted important points that largely contributed to the improvement of this work.

[1] M. B. Eisen, P. T. Spellman, P. O. Brown, D. Botstein, Cluster analysis and

display of genome-wide expression patterns, Proceedings of the National Academy of Sciences 95 (25) (1998) 14863–14868.

[2] T. Schreck, J. Bernard, T. von Landesberger, J. Kohlhammer, Visual cluster analysis of trajectory data with interactive kohonen maps, Information Visualization 8 (1) (2009) 14–29. arXiv:http://ivi.sagepub.com/content/8/1/14.full.pdf+html, doi:10.1057/ivs.2008.29.
URL http://ivi.sagepub.com/content/8/1/14.abstract

[3] G. Andrienko, N. Andrienko, S. Rinzivillo, M. Nanni, D. Pedreschi, F. Gianotti, Interactive visual clustering of large collections of trajectories, IEEE Symposium on Visual Analytics Science and Technology (2009) 3–10.

[4] H. Lee, J. Kihm, J. Choo, J. Stasko, H. Park, iVisClustering: An interactive visual document clustering via topic modeling, in: Computer Graphics Forum, Vol. 31, Wiley Online Library, 2012, pp. 1155–1164.

[5] A. K. Jain, Data clustering: 50 years beyond k-means, Pattern Recognition Letters 31 (8) (2010) 651–666.

[6] A. Y. Ng, M. I. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, NIPS.

[7] C. M. Bishop, Pattern recognition and machine learning, Springer, 2006.

[8] S. Lespinats, M. Aupetit, Classimap: a supervised mapping technique for decision support, in: Proceedings of the EuroVis Workshop on Visual Analytics using Multidimensional Projections, 2013, pp. 17–20. doi:10.2312/PE.VAMP.VAMP2013.017-020.
URL http://diglib.eg.org/EG/DL/PE/VAMP/VAMP2013/017-020.pdf

[9] A. Karatzoglou, A. Smola, K. Hornik, kernlab (R package) (2013).

[10] I. Fodor, A survey of dimension reduction techniques, Tech. rep., Lawrence Livermore National Laboratory (2002).

[11] L. Van der Maaten, E. Postma, H. Van Den Herik, Dimensionality reduction: A comparative review, Tech. rep., Tilburg University (2009).

[12] C. Ware, Information Visualization: Perception for Design, Elsevier, 2004.

[13] P. Bruneau, F. Picarougne, M. Gelgon, Interactive unsupervised classification and visualization for browsing an image collection, Pattern Recognition (2010) 485–493.

[14] L. van der Maaten, G. Hinton, Visualizing data using t-SNE, Journal of Machine Learning Research 9 (2008) 2579–2605.

[15] S. A. Nene, S. K. Nayar, H. Murase, Columbia object image library (coil-20), Tech. rep., Technical Report CUCS-005-96 (1996).

[16] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[17] J. Seo, B. Shneiderman, Interactively exploring hierarchical clustering results [gene identification], Computer 35 (7) (2002) 80–86. doi:10.1109/MC.2002.1016905.
URL http://dx.doi.org/10.1109/MC.2002.1016905

[18] C. Turkay, J. Parulek, N. Reuter, H. Hauser, Integrating cluster formation and cluster evaluation in interactive visual analysis, Proceedings of Spring Conference on Computer Graphics.

[19] S. Rinzivillo, D. Pedreschi, M. Nanni, F. Giannotti, N. Andrienko, G. Andrienko, Visually driven analysis of movement data by progressive clustering, Information Visualization 7 (3-4) (2008) 225–239. doi:10.1057/palgrave.ivs.9500183.
URL http://dx.doi.org/10.1057/palgrave.ivs.9500183

[20] A. Kapoor, B. Lee, D. Tan, E. Horvitz, Interactive optimization for steering machine classification, in: Proceedings of the SIGCHI Conference on

Human Factors in Computing Systems, CHI '10, ACM, New York, NY, USA, 2010, pp. 1343–1352. doi:10.1145/1753326.1753529.
URL http://doi.acm.org/10.1145/1753326.1753529

[21] B. Broeksema, A. C. Telea, T. Baudel, Visual analysis of multi-dimensional categorical data sets, Computer Graphics Forum 32 (8) (2013) 158–169. doi:10.1111/cgf.12194.
URL http://dx.doi.org/10.1111/cgf.12194

[22] J. Philippeau, J. Pinquier, P. Joly, J. Carrive, Dynamic organization of audiovisual database using a user-defined similarity measure based on low-level features, IEEE International Conference on Image Processing (2008) 33–36.

[23] G. M. H. Mamani, F. M. Fatore, L. G. Nonato, F. V. Paulovich, User-driven feature space transformation, Computer Graphics Forum 32 (3) (2013) 291–299.

[24] M. Aupetit, Visualizing distortions and recovering topology in continuous projection techniques, Neurocomputing (2007) 1304–1330.

[25] I. T. Jollife, Principal Component Analysis, Springer, 1986.

[26] P. Desmartines, J. Hérault, Curvilinear component analysis: a self-organising neural network for non-linear mapping of data sets, IEEE Transactions on Neural Networks (1997) 148–154.

[27] N. Heulot, M. Aupetit, J.-D. Fekete, Proxiviz: an interactive visualization technique to overcome multidimensional scaling artifacts, in: Proceedings of IEEE InfoVis, poster, 2012.

[28] D. Shepard, A two-dimensional interpolation function for irregularly-spaced data, in: Proceedings of the 23rd ACM national conference, 1968, pp. 517–524.

[29] L. Martin, M. Exbrayat, G. Cleuziou, F. Moal, Interactive and progressive constraint definition for dimensionality reduction and visualization, in: G. R. F. Guillet, D. Zighed (Eds.), Advances in Knowledge Discovery and Management Vol. 2 (AKDM-2), Studies in Computational Intelligence, Springer, 2012, pp. 121–136.
URL http://hal.archives-ouvertes.fr/hal-00595035

[30] E. T. Brown, J. Liu, C. E. Brodley, R. Chang, Dis-function: Learning distance functions interactively, in: IEEE Conference on Visual Analytics Science and Technology, 2012, pp. 83–92.

[31] S. Basu, A. Banerjee, R. Mooney, Semi-supervised clustering by seeding, Proceedings of 19th International Conference on Machine Learning.

[32] X. Zhu, Z. Ghahramani, Learning from labeled and unlabeled data with label propagation, Tech. rep., Technical Report CMU-CALD-02-107, Carnegie Mellon University (2002).

[33] P. Bruneau, B. Otjacques, A proposition of interactive visual clustering system, EuroVis 2013 Workshop on Visual Analytics using Multidimensional Projections.

[34] A. Gisbrecht, B. Hammer, B. Mokbel, A. Sczyrba, Nonlinear dimensionality reduction for cluster identification in metagenomic samples, 17th International Conference on Information Visualisation (IV) (2013) 174–179.

[35] I. T. Nabney, NETLAB: Algorithms for Pattern Recognition, Springer, 2002.

[36] L. Zelnik-Manor, P. Perona, Self-tuning spectral clustering, NIPS.

[37] T. Kamada, S. Kawai, An algorithm for drawing general undirected graphs, Information Processing Letters 31 (1989) 7–15.

[38] S. Tilkov, S. Vinoski, Node.js: Using javascript to build high-performance network programs, IEEE Internet Computing 14 (6) (2010) 80–83.

[39] M. Harrower, C. A. Brewer, ColorBrewer.org: An online tool for selecting colour schemes for maps, The Cartographic Journal 40 (1) (2003) 27–37.

[40] J. B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, in: Proceedings of the American Mathematical Society, Vol. 7, 1956, pp. 48–50.

[41] R. C. Prim, Shortest connection networks and some generalizations, Bell System Technical Journal (1957) 1389–1401.

[42] J. Donaldson, T-distributed Stochastic Neighbor Embedding for R (R package) (2012).
URL `http://cran.r-project.org/web/packages/tsne/index.html`

[43] L. Hubert, P. Arabie, Comparing partitions, Journal of Classification 2 (1985) 193–218.

[44] L. Van der Maaten, Barnes-hut-sne, Tech. rep., Delft University of Technology (2013).

[45] Z. Yang, J. Peltonen, S. Kaski, Scalable optimization of neighbor embedding for visualization, in: Proceedings of the 30th International Conference on Machine Learning, 2013, pp. 127–135.

[46] J. Heer, G. G. Robertson, Animated transitions in statistical data graphics, Visualization and Computer Graphics, IEEE Transactions on 13 (6) (2007) 1240–1247.

[47] M. Bostock, Data driven documents, URL: `http://d3js.org` (2013).

[48] L. Martin, M. Exbrayat, G. Cleuziou, F. Moal, Interactive and progressive constraint definition for dimensionality reduction and visualization, Advances in Knowledge Discovery and Management, Studies in Computational Intelligence 398 (2012) 121–136.

[49] I. Borg, Modern multidimensional scaling: Theory and applications, Springer, 2005.

[50] F. V. Paulovich, C. T. Silva, L. G. Nonato, Two-phase mapping for projecting massive data sets, IEEE Transactions on Visualization and Computer Graphics 16 (6) (2010) 1281–1290.

[51] S. L. France, J. Carroll, Two-way multidimensional scaling: A review, Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 41 (5) (2011) 644–661.

[52] N. Heulot, M. Aupetit, J.-D. Fekete, Évaluation de proxiviz pour la fouille visuelle de données multidimensionnelles, Atelier Fouille Visuelle de Données : méthodologie et évaluation, in ECG 2012.

[53] N. Heulot, M. Aupetit, J.-D. Fekete, Proxilens: Interactive exploration of high-dimensional data using projections, in: Proceedings of the EuroVis Workshop on Visual Analytics using Multidimensional Projections, 2013, pp. 11–15.

[54] M. Sarkar, M. H. Brown, Graphical fisheye views of graphs, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (1992) 83–91.