Visually Controllable Data Mining Methods

Kai Puolamäki, Panagiotis Papapetrou, and Jefrey Lijffijt
Department of Information and Computer Science
Aalto University, Finland

***Abstract*—A large number of data mining methods are, as such, not applicable to fast, intuitive, and interactive use. Thus, there is a need for visually controllable data mining methods. Such methods should comply with three major requirements: their model structure can be represented visually, they can be controlled using visual interaction, and they should be fast enough for visual interaction. We define a framework for using data mining methods in interactive visualization. These data mining methods are called "visually controllable" and combine data mining with visualization and user-interaction, bridging the gap between data mining and visual analytics. Our main objective is to define the interactive visualization scenario and the requirements for visually controllable data mining. Basic data mining algorithms are reviewed and it is demonstrated how they can be controlled visually. We also discuss how existing visual analytics tools fit to the proposed framework. From a data mining perspective, this work creates a reference framework for designing and evaluating visually controllable algorithms and visual analytics systems.**

***Keywords*-data mining; visual analytics; interactive visualization; visually controllable data mining.**

I. Introduction and Related Work

A central problem in information visualization is how to navigate efficiently in information spaces. The problem arises in applications where data as a whole is too large to fit into a single view, or when a single view cannot efficiently show all interesting features of the data. This problem is also known as the *focus and context problem*.

Furnas [5] introduced an elegant formalization of this problem in terms of graphs, according to which one can view any information structure as a graph where nodes correspond to different *views* and directed links represent possibilities to move from one view to another via interaction. Furnas argued that this *logical structure graph* should enable the user to navigate between different views in short time without getting lost.

The necessary and sufficient conditions are *traversability* and *navigability*.

- *Traversability.* The views must be small enough, because it would otherwise be not possible to show them on a screen. Additionally, there must exist a short path between all nodes in the structure. A logical structure graph satisfying this property is called *traversable*.
- *Navigability.* It is not enough that a short path between two nodes exists, but the user must also be able to find that path. Navigability is related to the concept of *information scent* in the information foraging theory [15], [16] according to which as animals rely on "scents" to guide them to promising areas where they can find their prey, so do humans rely on various cues in the information environment to get similar answers. A logical structure graph that satisfies this property is called *navigable*.

A logical structure graph is *effective view navigable* if it is both traversable and navigable, and this property is called *effective view navigability* (EVN). If the logical structure graph is EVN it is possible for the user to access any view in a short time from any initial condition. Conversely, it is quite obvious that the user may get lost or it will take a long time to traverse the graph if either of these two requirements is violated.

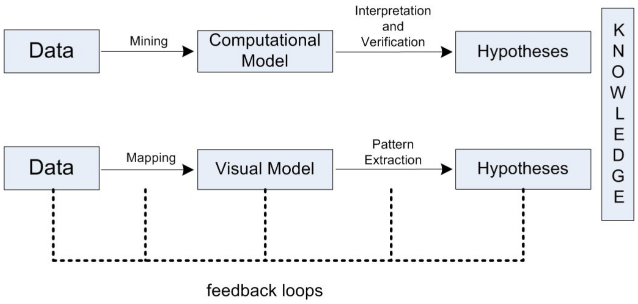


Figure 1. Comparing traditional data mining (top) and information visualization (bottom) analytic processes [3].

EVN is not directly applicable to many problem settings in the emerging field of visual analytics [20]. In EVN the user should be able to find a particular view efficiently. Nonetheless, the goal in interactive visualization or visual analytics may often be explorative. Users do not necessarily need to find any particular view—often they may not even know which view they would like to see. One can imagine that, instead of looking for any specific aspect of the data, the user is performing some task that could (at least in principle) be described by some utility function. The task could be, for example, to find outlier points in the data or to find the model that describes the data. In addition to straightforward data transformations and selections, the user may want to use some knowledge discovery and data mining (KDD) method during the task, such as clustering or regression.

There exist several visual analytics software tools to explore or analyze information using data mining methods, such as Matlab[1] , R[2], IBM SPSS Statistics[3], WEKA[4] and KNIME[5]. The first three have been developed and are being used for statistical and mathematical computing and visualization, though they support limited user interaction. A collection of KDD algorithms for data mining tasks is contained in WEKA and KNIME; user interaction is, however, limited.

We propose a novel framework for interactive visual exploration when a data mining algorithm is used. To our knowledge, there is yet no formalization for the exploration task. Several process models, like the ones shown in Figure 1, have been developed for data mining and visualization, respectively, but there is little work on combining the data mining and visualization tasks together; see Bertini et al. [3] for review and references of the current state of the art.

A well known and one of the most common data mining process models is CRISP-DM: CRoss Industry Standard Process for Data Mining[6]. CRISP-DM contains several phases, and it is iterative, but the timescales of the models are different from our viewpoint: we are interested in interactive timescales (like 10 seconds or less), while CRISP-DM describes a whole data analysis process which involves steps with timescales of several days or even months. One drawback of the CRISP-DM cycle is the limited user-control on the data mining process. The CRISP-DM model allows to move from one phase to another and roll back and forth to improve the quality of the results, but it does not address the interactivity discussed in this paper.

Visualization is considered necessary by the data mining community, but there is no principled approach to build algorithms for visual analytics. A typical data mining contribution would evaluate the "goodness" of a method using some measure, such as classification accuracy. There are currently no good measures to evaluate the suitability of an algorithm to a visual analytics task.

The lack of a principled framework or evaluation criteria makes it difficult for the KDD community to evaluate data mining contributions to visual analytics. The integration of data mining and visual analytics methods and communities is currently under active discussion, see, e.g., the recent ACM SIGKDD Workshop on Visual Analytics and Knowledge Discovery (VAKD '09) [17].

We propose a novel class of *visually controllable data mining algorithms* that can be used for interactive analysis. This idea was originally briefly mentioned, but not further discussed, in Puolamäki et al. [18]. We essentially require that the model structure of the data mining method can be represented and controlled visually, and that the algorithms are fast enough so that visual interaction is feasible in real time. Some basic data mining algorithms are reviewed and it is shown what is required from them to be visually controllable.

This formalization can be used as a theoretical starting point in designing and evaluating visual analytics systems and data mining methods that are suitable for such systems.

The remainder of this paper is organized as follows: in Section II, we formalize the proposed framework by providing formal definitions, introducing and analyzing several properties of the framework, and present some preliminary examples. In Section III, we provide a set of examples on how some existing data mining methods can be visually controlled. In Section IV, we present several existing visual analytics tools and discuss how they fit to the proposed framework. Finally, Section V concludes the paper and suggests directions for future research.

II. Framework

In this section, we formally define visually controllable data mining and explain why it is needed. We define the minimal set of requirements that should be satisfied by a system that enables to visually explore information structures using a data mining method.

A. *Effective View Analyzability*

We extend the definition of the effective view navigability (EVN) [5] to situations where information is explored using interactive visualization and data mining methods.

The EVN property applies to navigation tasks where the objective is to find a specific view. A typical analytics task is different and explorative: users may not be looking for a specific view. It is difficult to give a general formulation for all analytics tasks. We assume that the user tries to optimize some unknown *objective* or *utility function*. This objective function could be related to finding interesting information in the data, or forming some hypothesis about the data. For example, if the task would be to find outliers, the objective function could be the number of outlier points visible in a view. We can assume that each set of views is associated with a cost and the user tries to optimize the objective function by finding one or more views that minimize the cost. It would, however, be very difficult to formulate such an objective function explicitly, even if we knew the motivation of the user. In this paper, we do not try to find or formulate such an objective function. Instead, we try to construct a system that would be usable for any reasonable objective function.

The purpose here is, thus, not just to enable navigability, but to make it easy for the user to find the nodes with high utility. In fact, as far as an analytics task is concerned, EVN is not strictly required. It does not matter if the user is

[1] http://www.mathworks.com/

[2] http://www.r-project.org/

[3] http://www.spss.com/statistics/

[4] http://www.cs.waikato.ac.nz/ml/weka/

[5] http://www.knime.org/

[6] http://www.crisp-dm.org/

unable to reach nodes where the objective function has a low value. We define a new concept, called *effective view analyzability* (EVA), that differs from EVN by taking the new task description into account (optimizing an unknown objective function vs. finding a specific view).

It is useful to consider an analogue to numerical optimization algorithms, such as gradient ascent [2], conjugate gradient [8], Newton's method, or simulated annealing [11]. Typically, these algorithms require access to a method that evaluates the objective function at a given point. The algorithms try to find a set of parameter values such that the value of the objective function is maximized. Many optimization algorithms also use first or second derivatives of the objective function. An optimization algorithm does not typically have access to the "full" objective function: it only knows the value of the objective function at the points where the optimization algorithm chooses to evaluate it.

The analogue to our scenario is that in EVA the user (instead of a numerical optimization algorithm) makes a decision to move from one point or view to another in a way that the expected utility is maximized. The visualization must give the user necessary hints to make this decision: for example, in analogue with gradient ascent, for each possible navigation step (out-links in the logical structure graph), the user should be able to guess whether the objective function is expected to increase or decrease for a view that would follow the navigation step. The idea behind EVA is to display the information so that the user can efficiently find optimal values of his or her objective function, i.e., the user can act as an optimization algorithm. The logical structure graph, the views, and the transitions described by links must be such that they allow the user to find high values of the objective function.

Numerical optimization algorithms use various strategies to choose the next point to evaluate the objective function. For example, the gradient ascent chooses the direction where the value of the objective function increases most. Simulated annealing includes a "temperature" parameter in the computation. It is based on the idea that the original objective function can be thought of as the "cold" solution. In "hot" temperature the details are blurred out and only large scale features are left in the system. The objective function is first optimized in hot temperature and then the temperature is gradually decreased, or relaxed, towards the original objective function. The advantage of this approach is that this way it is more likely to find a global optimum solution, instead of getting stuck into a local optimum. The visualization analogue of simulated annealing is the focus and context approach. The temperature parameter indicates the detail level that is visualized: high temperature corresponds to showing the context, while small temperatures show more detail.

So far we have been referring to the concept of "view" without explicitly defining it. By *view* we mean the graphical presentation of data and data mining model. The view may include a possibility to select other types of presentations, or to modify some parameters of the data mining model. These choices will result in different views. The definition of a view can be considered identical to the generic definition in EVN [5], the views can be represented as nodes in the logical structure graph and possibilities to move to other views (e.g., by modifying the parameters of the data mining model). We give some concrete examples in Section III.

For a given objective function it is necessary that the user can efficiently find the views with high values of the objective function. It does not matter if the path to views with small values of the objective function are unreachable. However, if we assume that the objective function is unknown, the logical structure graph should be such that all views can be reached efficiently, because it is a priori possible that any view could have a high value of the objective function.

We therefore, like in EVN, require that the logical structure graph is *traversable*, i.e., the views are small enough and that there exists a relatively short path between any two views. We can, however, relax on the condition of navigability. The difference to EVN is that we are not interested in a specific view, but just in the value of the objective function associated with the views. It is enough if we can find a short path which leads to large values of the objective function. At minimum, it is required that each node in the structure graph contains information about the change of the objective function in the set of nodes to which its out-links lead. The out-link information should also be small enough in order to fit to the current view of the user. For example, in analogue with gradient ascent, the out-link info could contain information from which the user is able to infer the expected change in the objective function if the link were followed.

The optimization task differs from the navigation task in the sense that it is more important to be able to recover from wrong choices. For the navigation task it is sufficient to find the next view that is closer to the target view and there is no need to backtrack the steps (at least in the simplest navigation task of finding some pre-defined view). In the optimization task it is more likely to make bad navigation choices. It is possible that the selections lead to a local optimum, and if the out-link information is imperfect it may be that the navigation choice does not really increase the objective function. In both cases, the user would like to backtrack. We therefore require that the logical structure graph is symmetric, that is, it is always possible to backtrack the navigation steps. In practical terms, this means that the user must be able to try a view and return to the original view if necessary.

A logical structure graph that satisfies the following three requirements is called *optimizable*:

- *Requirement EVO1 (out-link info).* The out-links must contain information from which the user can infer knowledge about value of the objective function in the set of nodes to which the out-link leads.

- *Requirement EVO2 (small views).* Out-link info must be "small", i.e., it must be easy to represent it on a computer screen.
- *Requirement EVO3 (symmetry).* The logical structure graph must be symmetric, i.e., the user must be able to backtrack any navigation step.

We call a logical structure graph *effective view analyzable* (EVA) if it is both *traversable* and *optimizable*.

B. Visually Controllable Data Mining

It is possible to have an EVA structure with traditional visualization techniques that use smartly designed data transformations or selections, but do not incorporate any computational data mining methods or other sophisticated data transformations. An essential part of our contribution is how to integrate data mining methods in the EVA framework. The views can present, in addition to the data, the model structure of the data mining method, and navigation steps can be defined as modifications of the KDD model structure (as in our examples in Section III).

Definition 1: Visually controllable data mining method. A data mining method is visually controllable if it satisfies the following three properties: the parameters of the method as well as the extracted data mining models and original data should be visually representable (VC1), the method should be controllable via visual interaction (VC2), and the method should be fast enough to allow visualization and visual interaction (VC3).

Property 1: VC1 (visual representation). The first property of a visually controllable data mining method is that the involved parameters as well as the produced models are visually representable. This essentially places two sub-requirements: (i) there should be a way to visualize the model space and the data, and (ii) the visualization should convey an intuitive mapping between the model and data space, it should be conceptually sound, and comprehensible to the end-user so that it can be used for further analysis. From the EVA point of view, we require that the system is EVA and satisfies EVO2, i.e., it must be possible to form a view of the data and the model.

Example 1: An example of a visually representable model space is a linear regressor (e.g., Section III-A2). There is an intuitive mapping between the model space and the data space. An example of a non-representable regressor is a multi-layer perceptron (MLP), where the connection of the model space can, in general, not be intuitively linked to the data.

Property 2: VC2 (visually controllable). The second property is that there should be an intuitive visual interaction that enables the user to modify the model space. In addition, due to the requirement EVO1 the visualization should give the user an idea whether the interaction may lead to nodes with high values of the objective function. The interaction should be predictable. It follows from EVO3 that there should exist a one-to-one mapping between the views and the model space. Any navigation step should be retractable. For example, if the user chooses to "split" a cluster, the user must be subsequently able to choose to "merge" that cluster (i.e., undo the splitting and return to the original view).

Example 2: Consider for example the standard hierarchical clustering approach where data is clustered in different levels and then, depending on the demanded degree of granularity, one can go to lower or higher levels in the hierarchical tree and get various cluster representations of the data. This example is discussed in more detail in Section III-B2.

Property 3: VC3 (speed). The third property requires that visual interaction should be fast. This does not necessarily mean that the global solution must be found fast, or that the algorithm should be fast for arbitrarily large data sets. What is required is that a change induced by a typical visual interaction can be implemented in a sub-second response. It is possible that preprocessing takes some time, but a single user interaction can be computed fast. As far as scalability is concerned, it might not make sense to visualize very large data sets. This of course depends on the nature of the application domain. Therefore, it may be sufficient for the algorithm to be fast enough for data sets up to a certain size depending on the application.

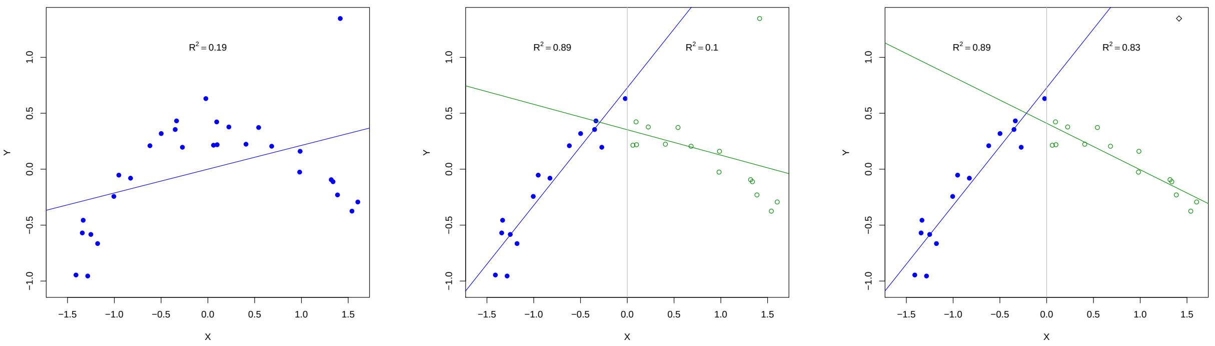
It should be noted that all properties discussed above are just a reformulation of the EVA property from the data mining viewpoint. Effectively this implies that EVA and visually controllable data mining refer to the same concept. Their main difference is that EVA is a generic formulation and visually controllable data mining specifies the requirements in terms of the data mining algorithm.

III. Examples

The main target of this paper is to define the interactive visualization scenario and the requirements for visually controllable data mining. In this section, we use the formulation of Section II to study some commonly used data mining methods via illustrative examples and discuss the applicability of the proposed framework. The example applications are taken from the fields of supervised (regression, classification) and unsupervised learning (clustering, dimensionality reduction). Besides the studied algorithms, there are most likely many other algorithms which are or could be made visually controllable.

A. Supervised Learning

The KDD technique for deducing a function from training data is called *supervised learning*. Next, we discuss the applicability of the proposed framework to several existing supervised learning tasks.



(a) Initial least-squares linear regression. The fit is rather poor, as indicated by the R^2 value.

(b) Split the model into two parts at $x = 0.0$. The fit for the left part is good, but the on the right the fit is poorer than previously.

(c) Remove the outlying data point in the top right corner. The fit is now quite good and we are satisfied with the constructed model.

Figure 2. Three steps during interactive analysis using *linear regression*. We build a simple model for a small data set by splitting the data into two parts and removing an outlier.

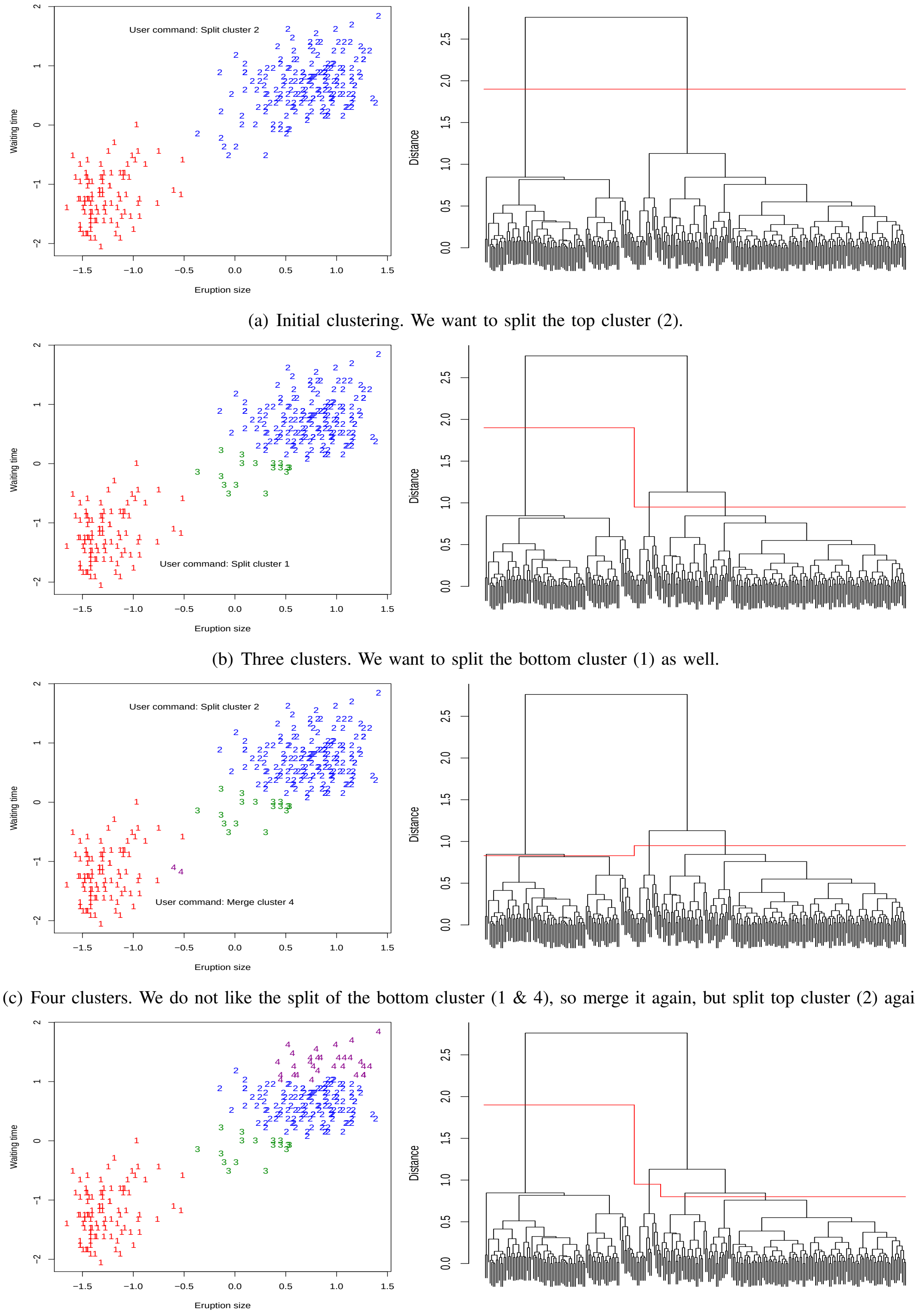
1) Methods: A large variety of supervised learning tasks exist in the data mining literature, including linear regression and classification. For a review of supervised learning see Kotsiantis et al. [12].

Linear Regression. The main task in linear regression is, given some input data, to find the linear model or set of linear models that best fit the data. From the perspective of our framework, linear regression could be used to visually fit the data, detect outliers, and further adjust the regression parameters, for example, the number of linear models to be used to fit the data, and thus, decrease the number outliers while defining a user-controlled fitting of the data. One main challenge here is to properly identify the appropriate domains for the linear regressor (i.e., number of linear models or regressors to fit the data).

Classification. In classification, the aim is to predict the label of some input object. We can have two types of classification: binary and multi-class. The former requires discerning between two classes, whereas the latter assigns an object to one of several classes. A simple classifier is the Naive Bayes classifier, where conditional independency of data features is assumed. From our viewpoint, the user could for example vary the weight of the variables or the data points. This will allow for efficient visual user-interaction, where the user can adjust the weights according to his/her objective function. Similarly, classification trees and k-nearest neighbor classifier could be visually controllable. The multi-layer perceptron (MLP) is a more complex classifier. As the number of hidden nodes/levels increases visualization becomes less and less feasible (VC1 is violated), and computation may require significant amount of time (VC3 is also violated). Finally, there are techniques to combine several weak classifiers to build a single strong classifier [13]. In these case, the user could visually obtain a strong classifier by interactively adding/removing weak classifiers.

2) Linear Regression Example: We conduct a case study to show how linear regression can be used in visual interactive analysis of a data set. We use a small artificially generated two-dimensional data set and try to build a model that explains the data. In Figure 2(a) we observe the data in a scatter plot over the two dimensions of the data, along with the simple linear fit, where each data point has equal weight. In the view we also see the R-squared value of the fit, to give a measure on the goodness of linear model. As users, we immediately observe that that the data cannot be explained well by a straight line. Suppose our out-links to other views consist of ignoring data points and splitting the data into parts using straight lines, and the respective reverse operations (unignoring data points and removing splits). A user may hypothesise that the data actually consists of two parts, instead of one as in the initial view. So, we introduce a split of the data points at $x = 0.0$. The new view is given in Figure 2(b). The model for the left part is reasonably convincing, but the right part is not. This is probably due to the point in the top right corner. The user may decide to ignore this outlier, hence obtaining the view in Figure 2(c). The user is now very happy with the current view and underlying model so decides to stop. As a result, we obtained a model for the data and identified one outlier point.

VC1 is satisfied, because the view gives an accurate and understandable view of the both the data and the regression model. VC2 is also satisfied, because the effects of the parameters of the model (ignoring points and splitting the data) are intuitive and a user can predict quite well what happens when changing between views. Linear regression in general satisfies VC3, even if we simply recompute the whole solution for each new view.



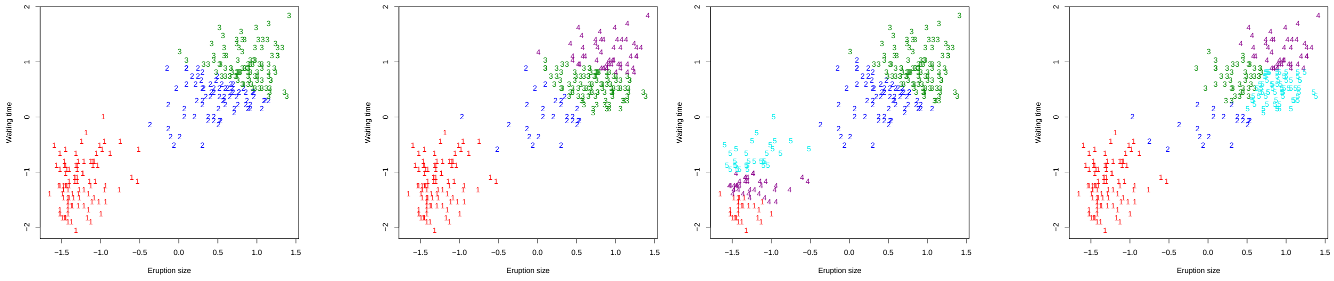
(a) Initial clustering. We want to split the top cluster (2).

(b) Three clusters. We want to split the bottom cluster (1) as well.

(c) Four clusters. We do not like the split of the bottom cluster (1 & 4), so merge it again, but split top cluster (2) again.

(d) Final four clusters, we do not want to split any further.

Figure 3. Four steps in the process of interactive analysis of the Old Faithful data set [6] using *agglomerative hierarchical clustering*. We show a scatter plot of the data over the two variables in the data set (left), and the corresponding dendrogram representing the distances between the data points (right). The red line indicates the cut-off corresponding to the commands given by the user, which defines the clustering shown in the corresponding left figure.



(a) Three clusters. We want to increase the number of clusters.

(b) Four clusters. We want to increase the number of clusters.

(c) Five clusters. The obtained clustering is very different from the previous clustering.

(d) Five clusters. Rerunning the algorithm with the same parameters can give very different results.

Figure 4. Possible steps during interactive clustering of the Old Faithful data set [6] using *k-means clustering*.

B. *Unsupervised Learning*

Unsupervised learning is a KDD task where, as opposed to supervised learning, the objective is to learn a model of the data. Clustering and dimensionality reduction are examples of unsupervised learning tasks.

1) Methods: Dimensionality Reduction is the process of reducing the number of data features under consideration by focusing on those that convey the most interesting information. Typically, a dimension reduction method reduces a data set to a form that can be plotted to two dimensions. Several dimensionality reduction methods exist. The most common ones include Principal Component Analysis (PCA), Factor Analysis (FA), Independent Component Analysis (ICA), Multidimensional Scaling (MDS), ISOMAP, Laplacian eigenmap, etc. Different methods try to preserve different features. For example, PCA and MDS try to preserve large distances, while spectral methods such as ISOMAP and Laplacian eigenmap try to preserve local neighbourhoods. Projection pursuit methods such as PCA, FA and ICA find a linear projection of the data to a direction that maximizes some measure, such as variance (PCA and FA) or non-gaussianity (ICA). For a thorough survey see Fodor [4], and Lee et al. [14] for a recent review on non-linear methods. The models of dimensionality reduction can often be parametrized by several parameters, for example, we can weigh different data points or attributes differently. Dimensionality reduction looses always some information of original data and this is why the interesting features may be visible in an alternatively parametrized view.

Clustering. Clustering is one of the basic data analysis methods. The objective in clustering is to partition the data into subsets in a way that the data items within the subset are similar to each other and, conversely, different across clusters. There are a large number of approaches based on different similarity and optimization criteria, using various algorithms approaches, see Jain et al. [9] for a review. We will discuss two clustering examples in the next subsection.

2) Clustering Examples: In this section we discuss visual interactive analysis using two clustering methods. First we study hierarchical clustering, which satisfies all three requirements well, and second we study k-means clustering, which we show to have some undesirable properties as a visually controlled data mining method. We us the well known Old Faithful geyser data set [6] and GNU R [19] in both examples. Our objective is to find if this data has some interesting cluster structure.

In Figure 3(a) we see on the left a scatter plot with all data points in the two dimensional space and on the right the dendrogram produced by the agglomerative hierarchical clustering algorithm. In the scatterplot each data point is represented by the number of the cluster that it belongs to. The dendrogram is cut by a line, which corresponds to the same clustering as shown in the scatter plot. We allow the user to split and merge clusters, where merging is just the symmetric operation of splitting.

The first operation we want to try is splitting the big cluster at the top. In Figure 3(b) we find a small collection of points (in the center) is separated from cluster two. We are pleased with the result, so let us also try to split cluster one. In Figure 3(c) we observe two points end up in a separate cluster. We do not want to get two loose points, so we reverse the operation by merging cluster four again. We also split cluster two again to see what happens. In Figure 3(d) we observe the new situation. We decide to stop the analysis and conclude this data set is perhaps best described by two or three clusters, because the views with four clusters were not very convincing. Let us also look at splitting the data into more clusters with the k-means algorithm.

In Figure 4(a) we find the three cluster solution for the same data set, using k-means [7] instead of hierarchical clustering. The top right part of the data set is split right in the middle to two clusters, which is quite different from the previous clustering. In ordinary k-means we can only increase or decrease the number of clusters and restart the algorithm. We increase k to four and obtain the view in Figure 3(b). This is quite similar to the four cluster solution in the previous example. Note the cluster model obtained from k-means is well represented by the view. It is easy to understand and gives full overview of the model, hence VC1 is satisfied. However, we should notice it is not directly obvious which points change from one cluster to another between Figures (a) and (b). Let us increase k to five. In Figure 3(c) we see the top cluster suddenly merged again and the bottom cluster was split into three parts. A user might become very confused because of the large change in the view. Also, if we restart the algorithm with the same parameters, we may obtain the view in Figure 3(d), which is very different from (c). These are both violations of the requirement of predictability, thus VC2 is not satisfied in this scenario. VC3 is hard to assess in general for k-means, because it is known for certain data the time to reach convergence is super-polynomial, but at least for moderately sized data sets the algorithm is in practice fast enough [1].

IV. Evaluation of Existing Visual Analytics Tools

A large variety of visual data mining and visual analytics tools have been developed for different tasks. In this section we discuss three representative tools and connect them to the proposed framework.

A. HCE: Hierarchical Clustering Explorer

HCE[7] is a visual analytics tool is used for interactive exploration of multidimensional data. The application domain of this tool is on Genome research, where cluster analysis is used to find meaningful groups in microarray data. HCE provides an overview of the whole dataset at any time by using several types of 2-D plots (such as bar charts, etc). Its key feature is that it performs hierarchical clustering without a predetermined number of clusters, and then enables users to determine the natural grouping with interactive visual feedback (dendrogram and color mosaic) and dynamic query controls that enable EVA. Users have the option to cluster with respect to a set of data attributes and choose between different distance measures and linkage methods. The clustering distance cut-off can be adjusted by the user and thus navigate through the dendrogram. In addition, users can compare clustering results produced by different clustering settings using either K-means or hierarchical clustering.

Nonetheless, HCE suffers from some major limitations. Firstly, some navigation features could be improved. For instance, when zooming in a cluster, the overall view of the clustering is poorly represented—zooming out is not possible. Also, navigation within the dendrogram is always performed at the same level for all sub-trees. A better version would be to only zoom in at some selected subtree(s). Finally, the user can not interactively change the number of clusters.

Thus, all three properties (VC1-VC3) are satisfied by HCE. However, there is room for improvement as regards VC1 and VC2.

B. Expression Profiler

Expression Profiler [10] is an extensible web-based collaborative platform for microarray gene expression, sequence, and PPI data analysis. It includes several data analysis components for gene expression: data selection (the user can get a brief statistical overview of the data), data transformation (e.g., K-nearest neighbor imputation to fill in missing values), similarity Search (the user specifies one or several genes, chooses a similarity measure and receives those genes most closely co-expressed with the selected genes within the dataset), clustering (provides both hierarchical and partitioning-based clustering methods). The hierarchical clustering tree produced for large datasets can be difficult to comprehend all at once, thus the user has the option of displaying a quick high-level overview of the clustered data by selecting the percentage of major tree-branches that should be displayed. In addition, significant gene finding, between group analysis, and other statistical components are available.

A major limitation of this tool concerns navigability: the user should always be on the same window in order to be aware of the overall clustering structure. Also, this tool is only designed for microarray data.

Again, all three properties (VC1-VC3) are satisfied. However, there is room for improvement as regards VC1.

C. Visual Data Mining Platforms

Several visual data mining platforms exist, such as KNIME, Weka, and RapidMiner. KNIME[8] is a modular data exploration platform that enables the user to visually create data flows, selectively execute some or all analysis steps, and later investigate the results through interactive views on data and models. The key advantage of KNIME is its inherent modular workflow approach, which documents and stores the analysis process in the order it was conceived and implemented, while ensuring that intermediate results are always available. Similar to KNIME, Weka[9] is a collection of machine learning algorithms for data mining tasks, which allows to create pipelines in order to perform data preprocessing, classification, regression, clustering, association rules, and visualization. RapidMiner[10] is an environment for machine learning and data mining tasks, which allows users to create data flows, perform data preprocessing and visualization, and integrate several learning schemes and attribute evaluators.

As far as visually controllable data mining is concerned, all three platforms use existing data mining methods

[7] http://www.cs.umd.edu/hcil/hce/.

[8] http://www.knime.org/

[9] http://www.cs.waikato.ac.nz/ml/weka/

[10] http://rapid-i.com/

and tools and allow users to visually interact with them. Nonetheless, they do not interfere at all in the details of data mining process; they rather employ data mining methods as black boxes that connect to each other allowing for visual interaction.

Clearly, VC1 is satisfied as all tools as well as their subcomponents (data mining models produced by the methods) are visually representable. VC2 is satisfied, as users can visually modify the model space. However, this can be achieved by adding or deleting several components from an existing data flow. There is room for improvement as regards VC2 by adding visual interaction capabilities to the data mining process rather than using these methods as black boxes. Finally, VC3 is satisfied as well.

V. Discussion and Conclusion

We have defined a formal framework for using data mining methods in interactive visual analysis as well as the interactive visualization scenario and the requirements for visually controllable data mining. Our intention is not to rank data mining methods or visualization systems, but present necessary and sufficient conditions for a visual analytics system that uses KDD methods. Thus, we present a reference framework for evaluation and design of visually controllable algorithms and visual analytics systems. There are several interactive visualization tools in which KDD methods are implemented, though not at all of them are optimized for interactive analysis. One reason for this is that visual analytics systems and data mining methods have been designed separately, and data mining algorithms have not usually been designed with the interactive visualizations in mind. Our key objective in this work is to bridge the gap that currently exists between the KDD and visual analytics communities.

Further topics for research would be to study if the current systems satisfy these criteria and if they could be improved. It would also be interesting to study which of the commonly used KDD methods satisfy or could be adapted to satisfy the visual controllability criteria.

References

[1] D. Arthur and S. Vassilvitskii. How slow is the k-means method? In *Proceedings of the twenty-second annual symposium on Computational geometry*, page 153. ACM, 2006.

[2] M. Avriel. *Nonlinear Programming: Analysis and Methods*. Prentice-Hall, Englewood Cliffs, N.J., 1976.

[3] E. Bertini and D. Lalanne. Investigating and reflecting on the integration of automatic data analysis and visualization in knowledge discovery. *SIGKDD Explorations - Special Issue on Visual Analytics and Knowledge Discovery*, 11(2):9–18, 2009.

[4] I. Fodor. A survey of dimension reduction techniques. Technical Report UCRL-ID-148494, Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, 2002.

[5] G. Furnas. Effective view navigation. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 367–374. ACM, 1997.

[6] W. Härdle. *Smoothing techniques: with implementation in S*. Springer, New York, 1991.

[7] J. Hartigan and M. Wong. Algorithm AS 136: A K-means clustering algorithm. *Applied Statistics*, 28(1):100–108, 1979.

[8] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49, 1952.

[9] A. Jain, M. Murty, and P. Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[10] M. Kapushesky, P. Kemmeren, A. C. Culhane, S. Durinck, J. Ihmels, C. Krner, M. Kull, A. Torrente, U. Sarkans, J. Vilo, and A. Brazma. Expression profiler: next generation–an online platform for analysis of microarray data. *Nucleic Acids Research*, 32:465–470, 2004.

[11] S. Kirkpatrick, J. Gelatt, C. D., and M. P. Vecchi. Optimization by simulated annealing. *Science 220*, 4598:671–680, May 1983.

[12] S. Kotsiantis. Supervised machine learning: a review of classification techniques. *Informatica*, 31:249–268, 2007.

[13] L. Kuncheva. *Combining pattern classifiers: methods and algorithms*. Wiley-Interscience, 2004.

[14] J. A. Lee and M. Verleysen. *Nonlinear Dimensionality Reduction*. Springer, New York, NY, USA, 2007.

[15] P. Pirolli and S. Card. Information foraging. *Psychological review*, 106(4):643–675, 1999.

[16] P. L. T. Pirolli. *Information Foraging Theory: Adaptive Interaction with Information*. Oxford University Press, 2007.

[17] K. Puolamäki, editor. *VAKD '09: Proceedings of the ACM SIGKDD Workshop on Visual Analytics and Knowledge Discovery*, New York, NY, USA, 2009. ACM.

[18] K. Puolamäki and A. Bertone. Introduction to the special issue on visual analytics and knowledge discovery. *SIGKDD Explorations*, to appear. http://www.hiit.fi/vakd09/si.html.

[19] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2009. ISBN 3-900051-07-0.

[20] J. Thomas and K. Cook, editors. *Illuminating the path: The research and development agenda for visual analytics*. IEEE Computer Society, 2005.