Model-Driven Visual Analytics

Supriya Garg Julia Eunju Nam I.V. Ramakrishnan Klaus Mueller

Computer Science Department Stony Brook University

ABSTRACT

We describe a Visual Analytics (VA) infrastructure, rooted on techniques in machine learning and logic-based deductive reasoning that will assist analysts to make sense of large, complex data sets by facilitating the generation and validation of models representing relationships in the data. We use Logic Programming (LP) as the underlying computing machinery to encode the relations as rules and facts and compute with them. A unique aspect of our approach is that the LP rules are automatically learned, using Inductive Logic Programming, from examples of data that the analyst deems interesting when viewing the data in the high-dimensional visualization interface. Using this system, analysts will be able to construct models of arbitrary relationships in the data, explore the data for scenarios that fit the model, refine the model if necessary, and query the model to automatically analyze incoming (future) data exhibiting the encoded relationships. In other words it will support both model-driven data exploration, as well as data-driven model evolution. More importantly, by basing the construction of models on techniques from machine learning and logic-based deduction, the VA process will be both flexible in terms of modeling arbitrary, user-driven relationships in the data as well as readily scale across different data domains.

KEYWORDS: Visual Analytics, Knowledge Discovery, Visual Clustering, Machine Learning, Grand Tour, High-dimensional Data, Network Security.

INDEX TERMS: H.5.2 [Information Interfaces and Presentation]: User Interfaces—Graphical user interfaces; I.2.6 [Artificial Intelligence]: Learning—Concept Learning; I.5.3 [Pattern Recognition]: Clustering—Similarity Measures.

1 INTRODUCTION

Modern day enterprises, be they commerce, government, science, engineering or medical, have to cope with voluminous amounts of data. Effective decision making based on large, dynamic datasets with many parameters requires a conceptual high-level understanding of the data. Acquiring such an understanding is a difficult problem, especially in the presence of incomplete, inconsistent, and noisy data acquired from disparate real-world sources. To make progress on this problem one must draw on the complementary strengths of computing machinery and human insight.

Recognizing this promising human-computer synergy, Visual Analytics (VA), defined as the science of analytical reasoning facilitated by interactive visual interfaces [22], has become a major development thrust. It seeks to engage the fast visual circuitry of the human brain to quickly find relations in complex data, trigger creative thoughts, and use these elements to steer the underlying computational analysis processes towards the extraction of new information for further insight. VA has widespread applications, such as homeland security, the financial industry and internet security among others. Research papers and tools related to VA are beginning to emerge (see e.g. [1][20][23]).

However, thus far the main emphasis in VA has been mostly on visualization, data management, and user interfaces (see e.g. [5] and other work mentioned in the following). As far as analytical computing goes, VA research has mainly focused on relatively low-level tasks, such as image [24] and video [12] analysis and database operations [21]. In today's VA systems, it is the human analyst who performs the actual reasoning and abstraction. Obviously this type of workflow is of limited scalability in terms of reasoning chain complexity. While there has been recent promising work on explicit knowledge management [25], this system has been mainly designed to help users shape and keep track of emerging insight, but not to derive higher-level models from the data. In this paper, we present a framework that allows human analysts to off-load some of the reasoning to a computational analyst. Collaboration with this machine agent occurs by pointing out interesting data patterns discovered in the visual interface.

The VA infrastructure we describe is rooted on techniques in *machine learning* and *logic-based deductive reasoning*. They assist analysts in making sense of large data sets by facilitating the generation and validation of models representing relationships in the data. By basing the construction of models on techniques from machine learning and logic-based deduction, the VA process will both be flexible in terms of modeling arbitrary, user-driven relationships in data and it will readily scale across different data domains.

Locating patterns in high-dimensional space via visual inspection can be challenging. The method of parallel coordinates (PC) [8], which has seen various refinements in recent years (see e.g.,[17]), maps a high-dimensional point into a piece-wise linear line that spans the vertical dimension axes. With PC, clusters can be isolated by brushing the appropriate data axes. There are, however, certain limits on the spatial complexity of the clusters. Scatter plots, on the other hand, provide an intuitive way to observe possibly arbitrarily complex relationships. However, in most applications, scatter plots are confined to projections onto two data axes at a time, and in order to view higher-order relationships spanning more than two dimensions, the associated projections must be serialized, viewed as a matrix, or be stacked [10].

We seek a visual interface that allows users to point out arbitrary high-dimensional patterns in a more direct manner. An early high-dimensional exploration paradigm providing a combined scatter plot of an arbitrary number of variables is the Grand Tour (GT) [2], which is part of the GGobi package [19]. The user is guided through hyper-space along a trajectory that is decided along the way by selecting the maximum of a given projection pursuit metric. The user stops when the current view point is the local maximum of this metric. While traveling, the user views projections (in form of scatter plots) of the data onto a 2D hyperplane. In GT the motion parallax resulting from the quick but

email: {sgarg, ejnam, ram, mueller}@cs.sunysb.edu

IEEE Symposium on Visual Analytics Science and Technology
October 21 - 23, Columbus, Ohio, USA
978-1-4244-2935-6/08/$25.00 ©2008 IEEE

continuous transition of hyper-plane projections provides a very compelling experience for getting a sense of the high-dimensional structure of the data. We present an autonomous high-dimensional navigation interface augmented with intuitive navigation aids that allows the user to fluently control the orientation of the projection hyper-plane, utilizing the resulting motion parallax to explore high-dimensional neighborhoods in an insightful manner.

Our paper is structured as follows. In Section 2, we discuss existing VA work relevant to ours. Section 3 provides technical preliminaries and gives an overview of our approach. In Section 4 we describe our model learning framework, while Section 5 describes our high-dimensional visual data navigation and pattern painting interface. Section 6 demonstrates the use of our VA framework for the learning of new models from user-suggested data examples within a few representative application scenarios. Section 7 provides details on how to generalize our approach to a wider field of applications and ends with conclusions.

2 Related Work

Learning models from patterns is an active research topic in various branches of computer vision. But there the pattern examples in most cases originate directly from image analysis, promoting unsupervised learning where subtle anomalies (the unexpected data) or new families of patterns which the model parameters cannot capture often go undetected or are misunderstood. Visual analytics, on the other hand, aims to be more flexible in the data constellations encountered, appealing to the complex pattern recognition apparatus of humans and their intuition, creativity, and expert knowledge to point out unusual configurations for further testing and model refinement. Papers recognizing this potential have been sparse, but are currently emerging. Janoos et al. [9] used a visual analytics approach to learn models of pedestrian motion patterns from video surveillance data, in order to distinguish typical from unusual behavior in order to flag security breaches in outdoor environments. Their semi-supervised learning approach in which users interact with video stream data improves upon the standard unsupervised learning schemes that are typically used in these scenarios (see for example [13]) allowing personnel to better train the models used for anomaly detection. The patterns encountered are fairly straightforward to visualize: motion trajectories of pedestrians visualized as flow fields. In contrast, our system attempts to learn from high-dimensional data.

Xiao et al. [23] describe a visual analytics system for network traffic analysis. In their system, users manually assemble and construct declarative knowledge rules based on example patterns they mark in the visual interface. These rules are then available for the classification of incoming new network traffic data. However, the manual assembly of these declarative rules can be laborious, especially in the context of high-dimensional data. Our system automates this process by replacing the manual rule assembly by machine learning and logic-based reasoning, based on the patterns specified by the user.

3 Technical Preliminaries and Overview

An important aspect of high-level semantic understanding of data is the identification of entities that are present in the data and relationships among them. Computational Logic offers an elegant and powerful vehicle to facilitate this kind of understanding. In particular Logic Programming languages such as Prolog offer a platform to encode and compute with such relations. A logic program consists of a data base of facts representing background domain knowledge and logical relationships (encoded as rules) that describe the relationships which hold in the data. Rather than running a program to obtain a solution, the user poses queries and the run time system searches through the data base of facts and rules to determine (by logical deduction) the answer[11].

Inference rules in Prolog, are of the form 'Head :- Body', meaning "conclude Head if Body holds". Facts are rules with empty bodies. Below is a specification of a path in a graph:

```
path(X,Y)  :- edge(X,Y).
path(X,Y)  :- edge(X,Z),path(Z,Y).
edge(a,b). edge(b,c). edge(a,d). edge(c,d).
```

Rule 1 says there is a path from X to Y if there is an edge from X to Y. Rule 2 says that there is a path from X to Y if there is an edge from X to Z and a path from Z to Y. Note that X, Y and Z are variables which are instantiated with actual nodes in the graph. The terms path(X,Y), edge(X,Y) are called *predicates*. An instance of a graph is specified by the facts. In the program above the four facts edge(a,b), edge(b,c), edge(a,d) and edge(c,d) specify a graph with edges from a to b, b to c, a to d and c to d respectively. A query '?path(X, d)' to the program will return one answer 'true' with X bound to "a" using resolution-based deduction. One can also get two more answers with X bound to "b" and "c".

In our approach the relations in the data are encoded as logical rules in Prolog. They constitute the *model* of the relations. Rather than encoding these rules manually we learn them from examples of data using Inductive Logic Programming (ILP) methods [15]. Typically ILP methods use a top-down approach of refining a very general rule, where the refining process involves adding a body predicate one at a time with the purpose of "better fitting" the given training examples, i.e., eliminate negative examples while still maintaining the coverage of positive examples. The added predicates can be base predicates, previously learned predicates, or arithmetic constraints (such as $|X-Y| < \delta$). ILP methods have produced impressive practical results (see e.g., [15] and are now well established as mainstream machine learning technology. The strength of ILP methods is that learning is done in the presence of prior background knowledge encoded as facts.

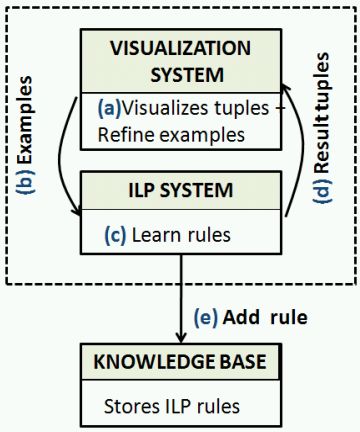


Figure 1: The overview of our system:
a. Visualize the dataset
b. Provide examples.
c. Learn rules.
d. Return and display results.
Steps b, c, d continue in a loop till the user is satisfied with the rule.
e. Add rule to the knowledge base.

Figure 1 provides an overview of how our VA system works: the analyst starts off by visualizing the dataset in the visualization interface, identifies interesting patterns and provides them as examples to the ILP system, and waits for the results. Once the ILP system has learned the rules, the analyst can visualize the tuples returned by the system. These tuples correspond to the relation encoded by the learnt rule (e.g. path (a, b), path (a, c), path (a, d) in the path predicate above). A single loop of "visualize-supply examples-learn rules-visualize returned tuples" constitutes an iteration of the VA cycle. Based on the returned results in the next iteration the analyst can supply addi-

tional positive and/or negative examples so that the ILP system can refine the rules to correct the accuracy of the learnt relation. Finally, when the rules encompass the patterns deemed relevant by the analyst, the new rule is added to the knowledge base.

With the user actively providing the examples from which to learn, an effective visual interface is crucial. One of our demo applications is the learning of models from sensor-rich high-dimensional network traffic data. Locating patterns in this high-dimensional space via visual inspection can be challenging. In the most general case these patterns can form high-dimensional manifolds. Current techniques, for the most part, can only reveal clusters and patterns of rather regular topology, such as quadrics. As mentioned, scatter plots provide a powerful means to visualize irregular patterns, but most applications only provide scatter plots along axis dimensions. This limits abilities to discover joint relationships in high dimensions. Instead, we provide an interface in which analysts can mark interesting example patterns directly onto arbitrary projections of hyper space, navigated via an intuitive flight control interface. But high-dimensional projections are always ambiguous, and thus overall quality assessments need to be made. For this, we couple our N-D projection display with a spectral plot that is used to fine-tune the patterns before sending them to the learning engine. The same spectral plot is also utilized for the visualization of tuples retrieved from the data on queries over the learnt model, which allows further model refinement.

As mentioned, the Grand Tour allows users to explore the N-D space by projecting the N-D data into a general hyperplanes. Such projections have the potential to reveal complex high-dimensional relationships that axis-wise projections may not. However, the curse of dimensionality makes self-guided tours a daunting exercise, and to provide some guidance the method of projection pursuit [7] was devised. It allows users to 'chase' pre-defined or 'interesting' patterns and cluster configurations. We describe here an interface that allows users to interactively navigate a given N-D subspace, using motion parallax to defuse the limitations of N-D to 2-D projections. This subspace may either be encountered during a projection pursuit-based guided exploration or during an unguided space exploration, initialized by human analyst intuition in our view-setup interface and refined using the flight controls.

4 Model Learning Framework

For our model learning framework, we use *Aleph* [26], a Prolog based ILP system. It can learn rules which are disjunctions of conjunctions like the path predicate shown earlier.

To learn rules in *Aleph,* we need background knowledge in the form of Prolog facts and rules, and some positive and negative examples corresponding to patterns that are in the relation and not in it respectively. Further, the program should know which background rules the new rule can depend on. Sometimes the user knows this and can restrict the rule's dependency to a subset of attributes. The user can add domain specific rules as well.

4.1 Rule Learning

The positive and negative examples are all provided through the visual interface. An example consists of either a single tuple, or a set of tuples. This is automatically converted into example files for *Aleph* which expects them to be in the form of Prolog facts as well. Note that when we say we pass tuples as examples, we only pass their unique indices.

In our implementation, we modified some of the *Aleph* parameters to facilitate the handling of large data sets. These include:

- Clause length (*clauselength)*: This specifies the length of the rule, including the head (the rule name). We increased it from the default value of 5 to 10, to learn longer rules
- Evaluation function (*evalfn)*: It is an expression which determines the importance of a clause based on the number of positive and negative examples it satisfies. We set this to 'pos' only in the first round of learning; this helps Aleph learn rules in the absence of negative examples.
- Number of nodes (*nodes)*: This denotes an upper bound on the number of nodes to explore while searching for the rule. We increased the default value of 5000 to 100,000.
- Layers of new variables (*i)*: This represents an upper bound on the layers of new variables. Each layer represents sets of variables occurring adjacent to the variables in the previous layer. Layer 0 contains the variables in the head of the rule. The value of this parameter was increased from the default value of 1.

With the above changes to the default values, *Aleph* was better suited to handle large data sets.

Once the rules are generated, we need to return patterns satisfying these rules. To do so the rule are evaluated over the KB using Prolog-style evaluation to materialize the tuples of the relation and return them to the visualization system.

Observe that the learnt rules can generate patterns in new datasets in the same domain. In this manner, the models created by one analyst can be shared with other analysts.

5 Visual High-Dimensional Data Interaction Interface

In the following, we use the term 'projection plane' to denote a subspace of arbitrary dimensions d in an N-dimensional data space, where $2 \leq d \leq N$.

5.1 High Dimensional Visual Interface: Navigator

Figure 2 below shows a screenshot of our interface. It is composed of (a) a scatter plot window, (b) a touch-pad like navigation interface that allows interactive control of the two orthogonal projection plane axis vectors (called *PPA-vectors* for short), (c) a window that shows the projections of all data axes onto the current projection plane, and (d) a display that plots their N-D coordinates as a bar chart histogram.

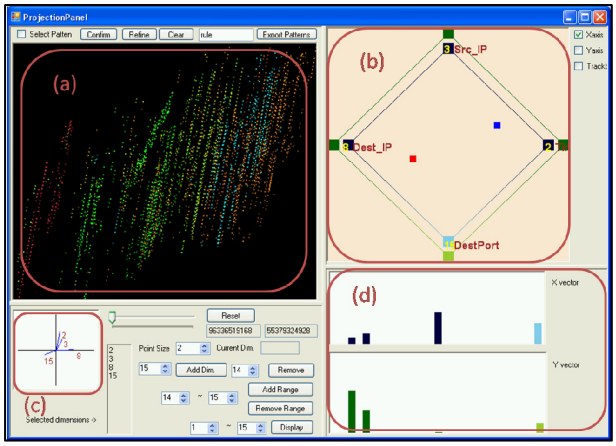


Figure 2: High Dimensional Navigation interface (a) Scatter plot (b) Touch Pad Navigator (c) Dimension axis display (DA-display) (d) Histogram display.

The user starts by choosing d dimensions of interest among the N dimensions available (this choice can be altered at any time). The touch pad interface will then be configured into a d-gon (an equilateral polygon with d vertices) for navigation (Figure 2(b)), where each polygon vertex stands for one dimension axis. Similar to a mouse touch pad, the user may move the pointer (indicated by a small colored dot and addressed by the mouse) inside the polygon, which transforms the projection plane of the scatter plot within the d-dimensional sub-space. In fact, we provide two such pointers, one each to represent the x-axis (red) and the y-axis (blue) of the PPA-vectors. We found that in many cases only the dynamics of motion can reveal interesting high-dimensional patterns, which, in fact, is part of the general idea of the Grand Tour paradigm used in GGobi. However, in the Grand Tour, in conjunction with projection pursuit, users are mostly confined to watching motions until interesting projections appear. While users do have some level of interactive navigation control, they cannot change the projection plane orientations in arbitrary ways. Our interface provides significantly more freedom in that respect, but it is probably most suited for expert users who are familiar with the space they are navigating in and can use informed intuition on the type of patterns and their location they seek to explore. Our direct navigation allows them to easily transform to these interesting projections and at the same time it also gives them a sense of location and orientation, which is important for navigation tasks.

A smooth transition function for the projection plane is the key to intuitive observance of dynamic patterns in N-D. Furthermore, we wish to be able to select a single dimension by simply moving over the polygon vertex it is associated with. Both can be achieved by interpolating the PPA-vectors via *generalized barycentric coordinates* [14], which calculate barycentric coordinates from a 2-simplex to a planar n-sided convex polygon. This mechanism has found frequent use for the interpolation of high-dimensional properties (see e.g.,[18]), and we extend its use here for the control of hyper-dimensional scatter plots. For a given PPA-vector, manipulated by the pointer in the touchpad d-gon, the barycentric coordinates determine the weight that each of the dimensions has on this vector. The weight of dimension d_3 (Src_IP) for the x-axis PPA-vector (the red point p) is (see Fig, 3):

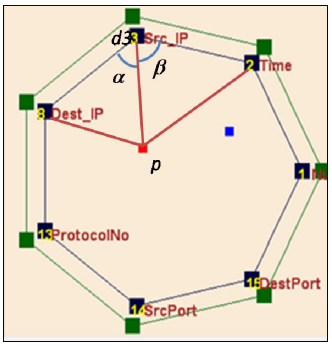


Figure 3: Computing weights of PPA-vectors using barycentric coordinates.

$$\overrightarrow{w_3} = \frac{\cot(\alpha)+\cot(\beta)}{\| \vec{p} - \vec{d_3} \|^2} \quad (1)$$

After computing the weights for all d dimensions, they are first normalized and then form the components of the respective PPA-vector. In essence, by moving the point closer to a certain dimension, the dimension's weight will grow larger in the vector, and thus this 'attracts' that dimension axis to the PPA-vector (that is, either the x or the y axis of the scatter plot). Each PPA-vector is displayed in the dimension histogram, to convey the prominent data dimensions in the current view to the user (Figure 2(d)). Finally, the data point positions in the scatter plot (Figure 2(a)) are given by the dot products between the PPA-vectors and the data points.

But even if users are permitted to freely move about space, choosing PPA-vectors via the touch pad interface, we must still enforce that the two PPA-vectors remain mutually orthogonal. To ensure this, for example, when moving the y-axis PPA-vector, we project this vector onto a vector y' that is closest to a vector orthogonal to the current x-axis PPA-vector (or vice versa):

$$y' = y - (x \bullet y) \cdot x \quad (2)$$

Below the scatter plot, we visualize the orientation and weight that each dimension axis has in the scatter plot (see Figure 2(c)). This type of dimension axis display (or *DA-display*) is also part of GGobi. It very intuitively shows which dimension the scatter plot visually separates (those whose axes are spread apart in this DA-display) and which ones it does not (those whose axes are close or even coincide). The user can increase the spread with the touchpad interface, by moving the mouse to reduce or increase the dimension's weight in one of the projection plane vectors. It also shows which dimensions are not or only mildly expressed in the scatter plot (those whose axes are very short in the axis plot).

We also provide a more radical means to separate the visual effect of two dimensions. Our interface allows users to flip one dimension's weight to a selected PPA-vector. This mirrors the data along this dimension with respect to this PPA-vector only, and can yield the desired separation (see Figure 4). While in some cases this alone is still not sufficient (which is easily seen in the DA-display), further mouse-movements can eventually achieve the goal. The order of the vertices along the touch pad polygon

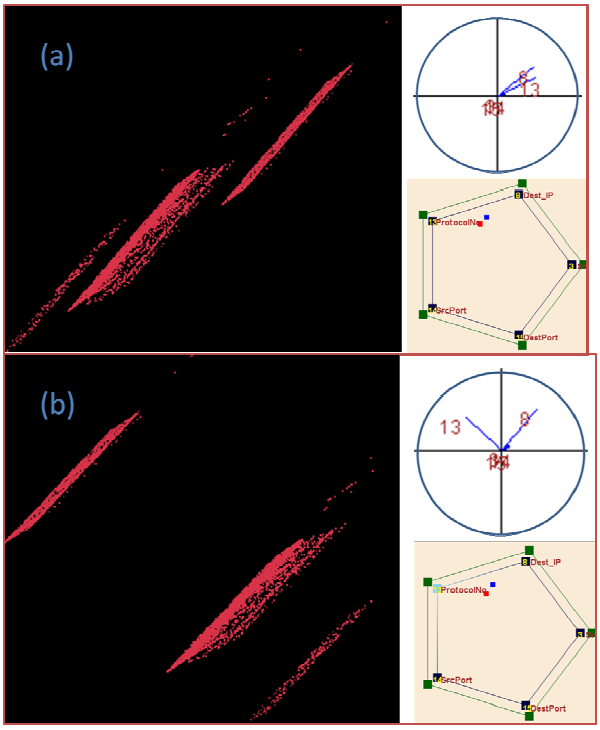


Figure 4: Two dimensions (8 and 13) which were close to each other (a) exhibit a broader gap in the projection plane by flipping the 13th dimension axis from positive to negative (b). Data points in the scatter plot will be spread out between the 8th and the 13th dimension. A negative flipping (negative weights) is conveyed on the touch pad navigator by coloring the axis poles with less saturation. (a) Navigating only with positive weights. (b) Flip the 8th dimension with respect to the x-axis and the 13th dimension with respect to the y-axis of the projection plane.

strictly determines the sub-space that can be explored. Users can set up a new sub-space for visualizing other desired relationships among dimensions by manipulating the vectors in the DA-display. Here, orthogonal orientation of the two PPA-vectors is again enforced using a method similar to the one described above.

5.2 High Dimensional Visual Interface: Point Map

Even with the dynamic motion display, the scatter plot projections can still present ambiguities. Therefore, we also provide a spectral plot (called *point map*) which represents D-dimensional data completely undistorted and with no projection ambiguities (Figure 5 (a)). The point pap displays each data vector (the spectrum) as a horizontal line of colored pixels, one per axis direction. The color mapping is according to value, normalized from 0 to 1, and the color is mapped from blue to red. The point lines are grouped by the smallest cluster they reside in (the leaf nodes of the hierarchy). The dendrogram (Figure 5(b)) represents the current status of the cluster hierarchy. The user may double-click on any such cluster to make it the active cluster. The scatter plot will then display the information of this active cluster, and the color of the points in the scatter plot corresponds to the dendrogram cluster color. The interface also features weighted k-means clustering and sorting.

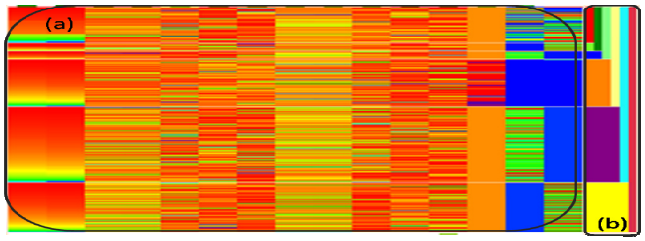


Figure 5: High Dimensional Visualization Interface (a) Spectrum Point Map (b) Tree-like Dendrogram.

5.3 Interaction with the ILP System: Pattern Painting

Upon discovering some interesting patterns in the N-D projection, the analyst may paint those patterns directly in the scatter plot, to be subsequently exported to the ILP. The selected patterns will be highlighted, after which the user moves further about in the high dimensional space to confirm that these patterns are indeed correct and not just due to projection effects. In the point map these sets can then be separated as different clusters. Since different clusters are represented by colors in both dendrogram and scatter plot, analysts can quickly gain insight into the selections and possibly refine the existing or even select additional patterns.

The selected pattern sets are exported to the ILP system. That in turn produces new sets of patterns from the data (by evaluating the rules) which can be visualized in a similar fashion. Imported pattern sets will be represented as clusters and can be viewed in different colors. Users can refine positive examples or indicate a pattern as a negative example, and then export these new sets of examples to the learning system.

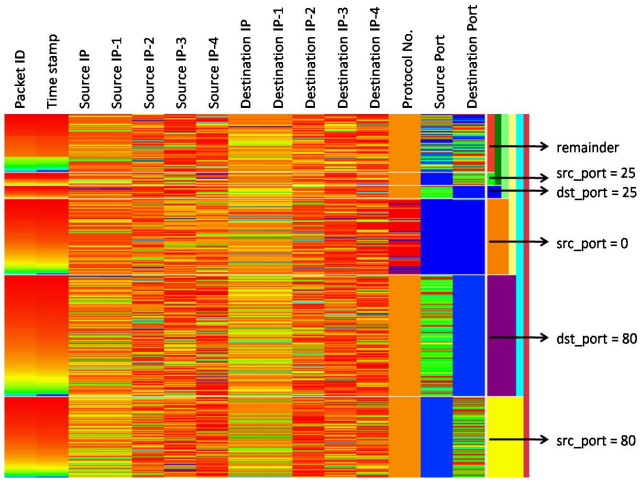


Figure 6: Color map showing the clusters formed after filtering based on port values.

Table.1: Example of patterns selected to learn the rule webpage-load.

index	src_ip	dest_ip	src_port	dest_port	time
10	ip_9	ip_14	80	2659	0.1
…	...	…	…	...	...
44	ip_9	ip_14	80	2663	9.2
57	ip_11	ip_18	80	2646	25
…	…	…	…	…	…
68	ip_11	ip_18	80	2675	32

6 Model-Driven VA in Action

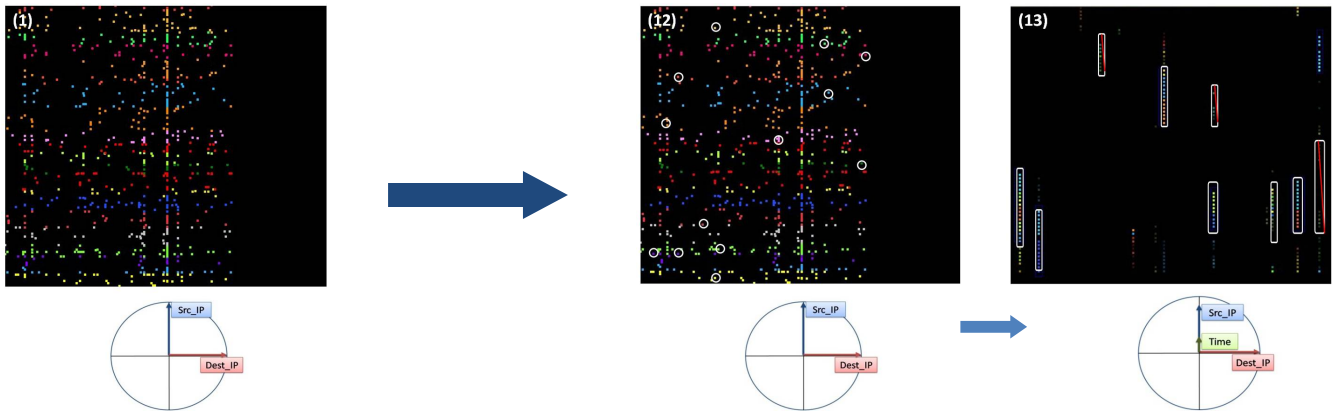
In this section we illustrate the operational aspects of our VA framework by building models of relations in two data sets, namely network traffic and census data.

6.1 Network Traffic

This dataset was obtained from the MAWI Working Group Traffic Archive [27]. It represents traffic data captured over an hour, in tcpdump format. We extracted the required fields in a comma separated file, and preprocessed it in the format needed by our VA system. The fields extracted were packet number, source IP, destination IP, source port, destination port, protocol number, and time stamp. Any unknown fields were given the value of 0.

As the dataset is huge, to see an overall structure in it, we filtered it on basis of various port values (Figure 5). Here, the fields *Source IP-I* and *Destination IP-I*, where I=1, 2, 3, 4, represent the 8-bit partitions of the source and destination IP respectively. The filtering was done based on the source or destination port of the packets. As seen in Figure 6, it is done in the following order: (i) source port 80, (ii) destination port 80, (iii) source port 0 (unknown source port), (iv) destination port 25, and (v) source port 25. In this way, we characterized the network traffic at a coarse level. In the figure, the yellow and purple clusters at the bottom contain packets involving transfer on port 80 – the http port. Since they make up more than half of our dataset, we can easily confer that most of our traffic is http traffic. This means that a great number of webpage requests/loads are happening. To find out which destinations IPs have many webpage loads, we select the cluster corresponding to source port 80 – the yellow cluster.

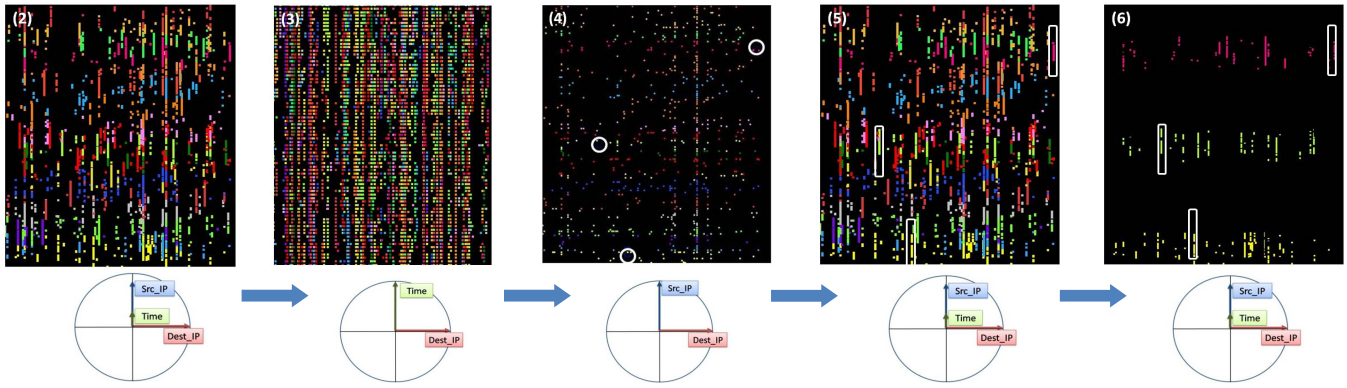
For further analysis, we shift our focus to the navigation on the high dimensional data space. Here, if all packets are displayed with the same color, it will be difficult to follow the patterns formed by the points sharing some attributes. For e.g. the size of the packets exchanged between two IPs might be increasing with time. In this case, we want to follow the number of packets exchanged between two IPs in a short time range. To address this, we cluster the dataset into 16 sets based on source IP. Adjacent clusters are assigned highly different colors, so that we can differentiate between points belonging to them easily.



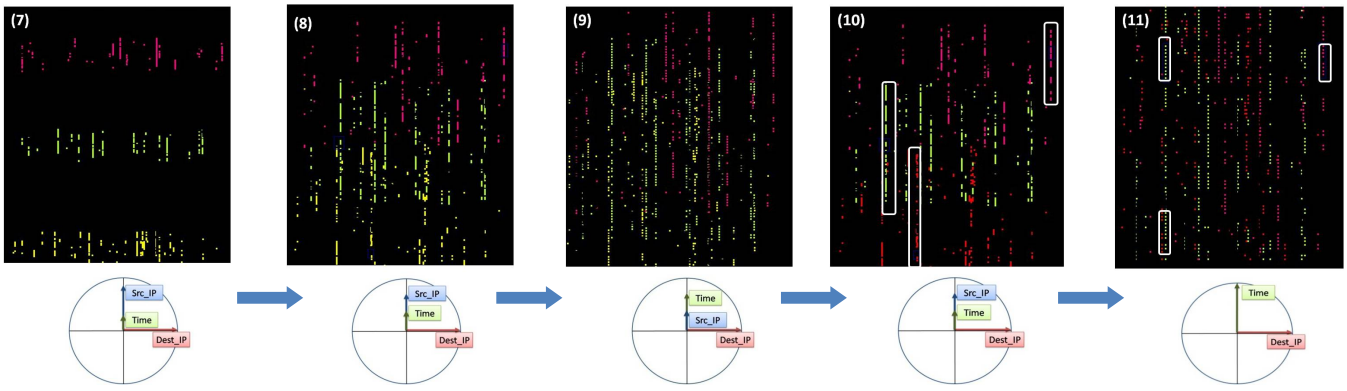
First overview panel

Results

Panel 1 on the left shows the initial projection view during rule learning - source IP vs. destination IP; it shows the IPs which exchanged packets with each other. Panels 12 and 13 on the right show the final results returned by the learning program. In panel 13, we look only at the results, and select positive and negative examples.



In panels 2 and 3, we move the Y-axis from source IP to time. We see the points spread in space showing that the corresponding source IP-destination IP pairs exchange multiple packets. In panels 4, 5 and 6 we select three points, and follow their path as the Y-axis is again changed. Due to the image getting cluttered, we select the clusters containing the points in panel 6.



In the panels above, we are looking only at the three selected clusters. In panels 7-9, we move the Y-axis in a fashion similar to panels 2 and 3. Due to the yellow and green being very similar, it is difficult to follow the paths traced by the points. To overcome this, we recolor the yellow cluster to red in panel 10, and finally select examples in panel 11 when the Y-axis represents time.

Figure 7: Thirteen panels showing the network traffic scenario.

We follow the rest of this scenario in Figure 7, which contain 13 panels telling the story from the initial projection in panel 1, to the display and refining of results in panels 12 and 13.

For an initial view, we project the packets into the domain of source IP against destination IP (Figure 7(1)), showing us the IPs which exchanged packets. To follow the exchange of packets between two IPs, we vary the Y-axis of our projection between source IP and time. Figures 7(2) and 7(3) clearly show us a lot of points spreading, revealing one or even multiple webpage-loads during the motion.

In Figures 7(4) and 7(5), we try to follow a few points, as the Y-axis is being changed. But since this scatterplot has too many points, we just select the three clusters which contain these points (Figure 7(6)). This step makes our task of selecting a few positive examples much easier.

In Figures 7(7)-7(11), we look at a smaller subset of the data, hoping to follow patterns more easily. However, as we move the Y-axis between source IP and time, we notice that the yellow and green clusters clash (Figures 7(8) and 7(9)), and being quite similar, they are difficult to differentiate. So we change the color of the yellow (lower) cluster to red, and continue our investigation as in Figures 7(10) and 7(11).

Now, the patterns are much clearer, and as the Y-axis is aligned with time, we can easily pick some examples. At this point we see that the red point we had selected did not give rise to any webpage load as the packets have been exchanged over long periods of time (see Figure 7(11)). However the pink point and the green point provide us with one and two examples, respectively. Thus, one source-destination IP pair had in fact multiple webpage loads.

We can look for further examples in a similar fashion. After selecting the examples, we assign a name to the relation that will be learnt based on these examples send it over to the *Aleph* system for learning the rules. Table 1 is a fragment of the examples used. Here, two sets of packets have been selected as examples. As expected, the source and destination IPs are the same for all packets in an example. In *Aleph*, these examples are loaded along with the background facts. In the presence of only positive examples, *Aleph* learns the following rule, with X representing a set of packets:

```
webpage_load(X) :- same_src_ips(X),same_dest_ips(X),
                   same-src_port(X, 80).
```

This rule states that a webpage contains packets which share the same source IP, destination IP and source port 80. Now we generate the tuples which satisfy the above result. Being a general rule, generating all the results will both take a long time and lead to clutter on the screen. To avoid this, we only generate some results, which add up to a certain threshold. This is done by generating random subsets of the packet space, and checking if they satisfy the given rule. Further, we restrict the length of these subsets roughly to the range of the length of the positive examples. For e.g. if the examples have a length between 6 and 10, we allow subsets of length 4 to 12. Note that these estimations are not required if the rule itself provides a clause specifying the length.

The results are now presented to the user, and we show them along the initial source IP vs. destination IP axes in Figure 7(12). This gives us an overall idea of which packets are involved in a webpage load (and which are not). At this point, we just choose to see the results, and each result is assigned a distinct color. This helps us to see multiple webpage loads between the same pair of IPs. It provides us with a chance to further guide the learning system by refining these results. This can be done by marking some of the results as negative is they do not seem correct, or merging/splitting multiple results to form a correct example. In Figure 7(13) we see the results where the Y-axis is a mix of time and source IP, and the X-axis is destination IP. Here, we merge 6 sets of examples, to form a correct example. Two examples are marked as negative because they are too short, and a third one is marked as negative because its packets have been exchanged over a long time span.

After three such iterations of rule refining, Aleph learns the following rule:

```
webpage_load(X) :- same_src_ips(X),same_dest_ips(X),
                   same-src_port(X, 80),length(X, L),
                   greaterthan(L,8),timeframe_upper(X,10).
```

This rule contains extra constraints when compared to the previous one. Apart from the packets sharing source and destination IPs, and the source port 80, they contain greater than 8 packets (length(X,L), greaterthan(L,8)), and occur within a time frame of 10 seconds (timeframe_upper(X, 10)). Since in Panel 13 we crossed out a few results based on their length, and time span, we can clearly see that Aleph was able to make use of these negative examples in learning the extra parameters in the rule.

6.2 Census Dataset

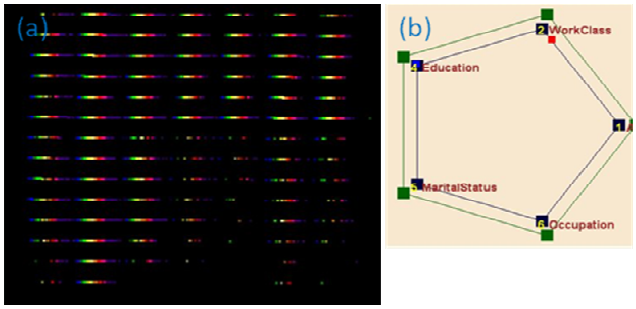


Figure 8: Scatterplot showing the distribution of age for education versus working class. The Y-axis is Education, and the X-axis is a mixture of WorkClass and Age.

The Census dataset [25], extracted from the 1994 Census data contains population characteristics including age, education, religion, marriage, profession, annual income etc.

Looking at this dataset, we can identify traits of people, who have "high" earning potential or work for a certain kind of organizations, etc. We use our system to learn models for these kinds of characteristics. First, we sorted the dataset based on the age. Then we clustered it into ten sets by the age field. Since each cluster is assigned a unique color, this helps us identify age groups in the scatter plots.

Now we switch to the navigation interface and see a scatterplot of *education* vs. *working class*. As we make the Y-axis a mixture of *working class* and *age*, we see that for most education-working class combinations, people of all ages are present (Figure 8). We now learn a model for older men with high capital gains. We select some examples satisfying these criteria. In these examples we know that age and high savings are attributes already common to them. To learn other attributes common to them – e.g. education, hours of work etc, we mark the attributes on which the rule should depend. By default, the new rule depends on all the predicates existing in the database, and if the user does not select the attributes on which the rule should depend, the rule will simply state that *saves_more(A)* holds for old men with high capital gains,

learning nothing new. Hence, using the user's pointers, the background file used by Aleph is modified, and it learns a rule which is the disjunction of the following four rules:

1. saves_more(A) :- education_level(A,B), gteq(B,13), capital_loss(A,0), native_country(A,'United_States').
2. saves_more(A) :- race(A,'White'), education(A,'HS_grad'), marital_status(A,'Married_civ_spouse').
3. saves_more(A) :- race(A,'White'), workclass(A,'Private'), education_level(A,B), greater_than_equal_to(B,13).
4. saves_more(A) :- marital_status(A,'Married_civ_spouse'), occupation(A,'Exec_managerial').

The quoted strings in the above rule come from the original dataset. The first rule holds true for people with a minimum education level of 13 (Bachelors in this example), no capital loss, and US nativity. The second rule holds for white HS grads that have the marital status 'Married_civ_spouse'. The third rule holds for whites working in the private sector, who have the minimum education level of Bachelors (13). The last rule accepts people whose marital status is 'Married_civ_spouse' and work as an executive manager.

In contrast to the network dataset, here our rule is a disjunction of conjunctions. Also compared to [23], our rules are neither hand constructed, nor restricted to pure conjuncts.

7 Conclusions

We described a VA infrastructure for analyst-guided discovery of relations present in datasets. The analyst merely supplies patterns from the visual interface and the system learns the relations automatically based on these patterns. Another important aspect of the infrastructure is that it is readily scalable over different datasets - all that is needed is encoding the background knowledge about the new data set; all the other aspects of model building remains the same. We demonstrated the feasibility of our approach by implementing a prototype VA system and illustrated it by building models of relations in two different data sets. In the future we plan to expand the system to deal with inconsistent and noisy data sets by integrating statistical learning in logic programming and generalizing ILP methods to learn probabilistic rules from data.

8 Acknowledgements

The research of I.V Ramakrishnan was partially supported by NSF grants: CNS-0721665, 0627447, IIS-0534419. S. Garg, E. Nam, and K. Mueller were partially supported by NSF grant ACI-0093157, NIH grant 5R21EB004099-02, and an equipment grant from the NVIDIA Professor Partnership program.

References

[1] M. Amherst, M. Ester, H.-P. Krieger, "Toward an Effective Cooperation of the User and the Computer for Classification," *ACM Knowledge Discovery and Data Mining,* pp. 179-188, 2000.

[2] D. Asimov, "The Grand Tour: A Tool for Viewing Multidimensional Data," *SIAM Journal on Scientific and Statistical Computing* 6(1):128-143, 1985.

[3] A. Asuncion, D. Newman, UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: UC School of Information and Computer Science. 2007.

[4] W. Bethel, S. Campbell, E. Dart, K. Stockinger, K. Wu, "Accelerating network traffic analytics using query-driven visualization," *IEEE Symp. Visual Analytics Sci. and Techn. (VAST),* pp. 115-122, 2006.

[5] S. Card, J. Mackinlay, B. Schneiderman. *Readings in Information Visualization: Using Vision to Think*, Morgan Kaufman Publ., 1999.

[6] R. Eccles, T. Kapler, R. Harper, W. Wright, "Stories in GeoTime", *IEEE Symp. Visual Analytics Sci. and Techn. (VAST)*, 2007.

[7] J. Friedman, J. Tukey, "A projection pursuit algorithm for exploratory data analysis," *IEEE Trans. on Comp. C-23*, (9):881–890, 1974.

[8] A. Inselberg, B. Dimsdale, "Parallel Coordinates: A tool for visualizing multi-dimensional geometry," *IEEE Vis.*, pp. 361-378, 1990.

[9] F. Janoos, S. Singh, O. Irfanoglu, R. Parent, R. Machiraju, "Activity analysis using spatio-temporal activity maps in surveillance applications," *IEEE Symp. Vis. Anal. Sci. Techn. (VAST)*, pp. 3-10, 2007.

[10] J. LeBlanc, M.O. Ward, N. Wittels, "Exploring n-Dimensional Databases*," IEEE Visualization*, pp. 230-239, 1990.

[11] J. Lloyd, *Foundations Logic Programming, 2nd Ed.,* Springer. 1988.

[12] H. Luo, J. Fan, J. Yang, W. Ribarsky, S. Satoh, "Analyzing Large-Scale News Video Databases to Support Knowledge Visualization and Intuitive Retrieval," *IEEE Symp. Visual Analytics Sci. and Techn. (VAST)*, pp. 107-114, 2007.

[13] D. Makris, T. Ellis, "Learning semantic scene models from observing activity in visual surveillance," *IEEE Trans. Systems, Man, and Cybernetics, Part B* 35(3):397-408, 2005.

[14] M. Meyer, H. Lee, A. Barr, M. Desbrun, "Generalized Barycentric Coordinates on Irregular Polygons", *J. Graphics Tools*, pp 1086-7651, 2002

[15] S. Muggleton, L. De Raedt,"Inductive Logic Programming: Theory and Methods," *J. Logic Programming*, pp. 629-679, 1994.

[16] E. Nam, Y. Han, K. Mueller, A. Zelenyuk, and D. Imre. "ClusterSculptor: A visual analytics tool for high-dimensional data," *IEEE Symp. Visual Analytics Sci. Techn. (VAST)*, pp. 75-82, 2007.

[17] E. Rundensteiner, M. Ward, Z. Xie, Q. Cui, C. Wad, D. Yang, S. Huang, "Xmdvtool: quality-aware interactive data exploration," *Proc. SIGMOD*, pp. 1109-1112, 2007.

[18] R. Smith, R. Pawlicki, I. Kókai, J. Finger, T. Vetter, "Navigating in a Shape Space of Registered Models," *IEEE Trans. Vis. Comput. Graph*. 13(6):1552-1559, 2007.

[19] D. Swayne, D. Lang, A. Buja, D. Cook, "GGobi: evolving from XGobi into an extensible framework for interactive data visualization," *Comp. Statistics & Data Analysis*, 43(4):423-444, 2003.

[20] S. Teoh, K.-L. Ma, S. Wu, D. Jankun-Kelly,"Detecting flaws and intruders with visual data analysis," *.IEEE Computer Graphics and Applications,* 24(5):27-35, 2004.

[21] D. Tesone, J. Goodall, "Balancing Interactive Data Management of Massive Data with Situational Awareness through Smart Aggregation," *IEEE Symp. Vis. Anal. Sci. Techn. (VAST),* pp. 67-74, 2007.

[22] J. Thomas, K. Cook, (editors). *Illuminating the Path: The Research and Development Agenda for Visual Analytics*, IEEE, 2004.

[23] L. Xiao, J. Gerth, P. Hanrahan, "Enhancing Visual Analysis of Network Traffic using a Knowledge Representation," *IEEE Symp. Visual Analytics Sci. and Techn. (VAST)*, pp. 107-114, 2006.

[24] J. Yang, J. Fan, D. Hubball, Y. Gao, H. Luo, W. Ribarsky, M. Ward, "Semantic Image Browser: Bridging Information Visualization with Automated Intelligent Image Analysis," *IEEE Symp. Visual Analytics Sci. and Techn. (VAST)*, pp. 191-198, 2006.

[25] D. Yang, E. Rundensteiner, M. Ward, "Analysis Guided Visual Exploration of Multivariate Data," *IEEE Symp. Visual Analytics Sci. and Techn. (VAST),* pp. 83-90, 2007.

[26] http://web2.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/aleph.html

[27] MAWI Working Group Traffic Archive, http://mawi.wide.ad.jp/mawi/