Interactive Data Summarization: An Example Application

Neal Lesh
Mitsubishi Electric Research Laboratories
201 Broadway
Cambridge, MA, 02139
lesh@merl.com

Michael Mitzenmacher*
Harvard University
Computer Science Department
Cambridge, MA, 02138
michaelm@eecs.harvard.edu

ABSTRACT

Summarizing large multidimensional datasets is a challenging task, often requiring extensive investigation by a user to identify overall trends and important exceptions to them. While many visualization tools help a user produce a single summary of the data at a time, they require the user to explore the dataset manually. Our idea is to have the computer perform an exhaustive search and inform the user about where further investigation is warranted. Our algorithm takes a large, multidimensional dataset as input, along with a specification of the user's goals, and produces a concise summary that can be clearly visualized in bar graphs or linegraphs. We demonstrate our techniques in a sample prototype for summarizing information stored in spreadsheet databases.

Categories and Subject Descriptors

H.1.2 [**Models and Principles**]: User/Machine systems; H.5.2 [**Information Interfaces and presentations (e.g., HCI)**]: User Interfaces

General Terms

Algorithms

Keywords

interactive data exploration, data mining, information visualization

1. INTRODUCTION

Summarization is a common and powerful though often time-consuming approach to analyzing large datasets. For example, suppose one wants to investigate census data in order to understand the relationship between level of education and salary in the United States. A very compact summary of the census can be viewed by plotting the *average* salary by education level. This summary will be sufficient for some purposes, but others may require more time to achieve a better understanding of the data. A simple example would be to include the standard deviation information along with the averages. Further, it may be more revealing, for example, to break down the average salaries by age group, or to exclude outlying salaries. Additionally, the relationship between salary and education may be different for men and woman, or may vary by race or geography. In general, effective summarization involves both identifying overall trends and important exceptions to them.

Many tools exist to help users summarize data by allowing them to repeatedly generate and visualize different summaries [4, 6]. In this approach, the user must essentially perform an exhaustive search to visualize every potentially interesting summary. In our example of census data, the user would have to examine the data across each dimension, such as age, race, geography, and gender, as well as different aggregations, such as averaging by mean or by median. Note the importance of examining each combination to ensure that no salient observations are missed. For example, she would want to know if an observed trend using medians would not hold when using means. For lower-dimensional data, this process is tedious and error prone, and it does not scale well to higher-dimensional data.

Our idea is to have the computer do the exhaustive search rather than the user, and inform the user about where further investigation is and is not warranted. While summarization is highly subjective and context-dependent, the fundamental insight we leverage is that the user spends much of her time looking for and evaluating how the summary *changes* as she manipulates the data or the visualization. For example, if the computer indicates that the summary will change noticeably if the data is restricted by geographic region, then the user will likely attend to this variation. Similarly, and at least as importantly in practice, if the user can rely on the system to alert her whenever choosing a different aggregation method would change the summary significantly, then she can safely ignore this choice when she is not so alerted.

Figure 1 previews our approach on a simple database of automobile characteristics. It shows a series of screenshots, which will be described in greater detail below, of our prototype in which the user has a set of controls for manipulating which summary to view next. When the user requests

*Supported in part by NSF CAREER Grant CCR-9983832 and an Alfred P. Sloan Research Fellowship. This work was done while visiting Mitsubishi Electric Research Laboratories.

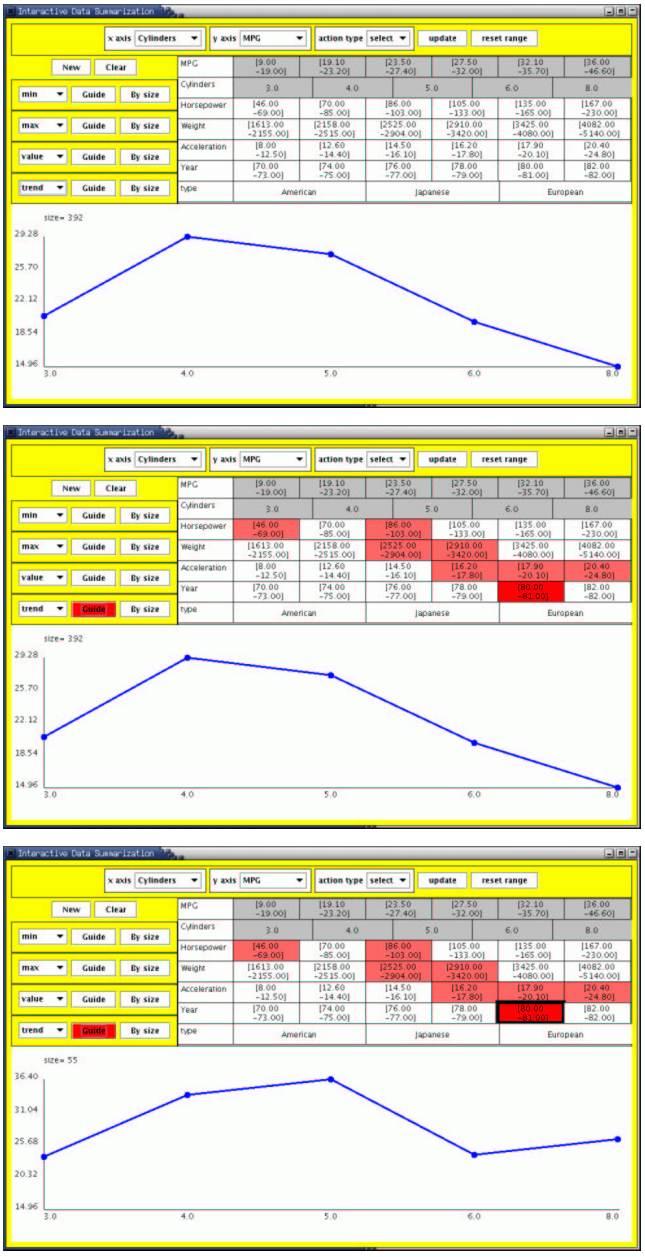

Figure 1: Illustrative example: The top screen shows the prototype after the user has chosen attributes for the x and y axes. The buttons on the upper left are controls the user can manipulate to restrict the data and thus view different summaries. The middle screen shows the prototype after the user has asked for computer assistance to find summaries with different trends between adjacent points than are in the current summary. The shadings of each button indicate how much the graphs will change if that button is used to restrict the data. The user knows, for example, that pressing the lightly shaded buttons will not change the graph significantly. The bottom screen shows the prototype after the user has pressed one of the darkly shaded buttons, which produces a graph with very different trends.

guidance, the prototype shades the different controls relative to how much the graph will change when those controls are used. This provides a great deal of information that the user would otherwise have to acquire by tedious exploration, and helps the user decide what to investigate next.

2. EXAMPLE APPLICATION

We demonstrate Intelligent Data Summarization (IDS) in a sample application for helping users explore correlations between attributes. Specifically we present LIDS, a line graph IDS prototype, implemented in Java.

2.1 Datasets

LIDS takes as input a simple table, such as can be produced by saving an Excel spreadsheet in text format. The first row indicates the names of each attribute, and each subsequent row is a record providing a value for each attribute. An attribute can either be numerical or can hold a collection of values, i.e., each value is interpreted as a string. LIDS treats an attribute is numerical if all the values for that attribute in the given dataset can be read as numbers.

In order to illustrate the use of LIDS, we use a dataset describing cars from the UC Irvine Machine Learning Repository[1] which has six attributes (miles per gallon (MPG), number of cylinders, horsepower, weight, acceleration, year, and the country in which it was produced) and 392 records with a value for each attribute. In our examples, the user investigates the correlation between horsepower and MPG, as well as between the number of cylinders and MPG.

2.2 Controls

In LIDS, summaries of a given dataset are represented as line graphs. In general, however, summaries might also be a single number, a histogram, or multiple line graphs. The range of possible summaries is defined by a set of *controls* that can be applied to the dataset. Each control can be assigned a value from a predefined set. For example, there may be a control to restrict the data used in the summary by year, and the possible values could include single years, a range of years, or allowing all years. Another control might select the aggregation function, with values such as average, average and standard deviation, median, minimum, or maximum.

We now define the controls in LIDS. Two controls, presented along the top of the interface shown in Figure 2, allow the user to choose attributes for the x- and y- axes. The y-axis must be a numerical attribute. When the user presses the 'update' button, the x and y axes are updated.

The remaining controls are set so that initially the summary produced is a line graph summarizing all the data. For each value x_i of the x-axis attribute, all the records in the dataset with that value are found, and the average y_i of all the values of the y-axis attributes in those records is computed to determine the point $\langle x_i, y_i \rangle$.

The large control panel on the top right of the interface allows the user to select subsets of the dataset by setting the appropriate controls. We call these controls *value constraints*. These controls allow the user to easily restrict the records used for the summary to a subset of the database, based on attribute values.

[1]Available at http://24.147.185.177/viz/datasets.htm.

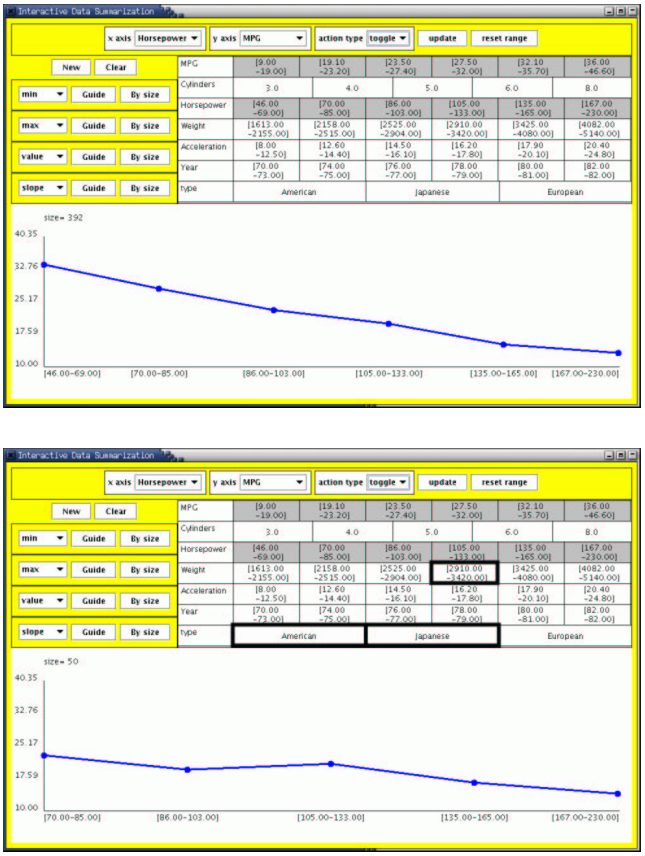

Figure 2: Our IDS prototype before guidance is invoked by user. The top image shows LIDS after the user has chosen to plot the horsepower against MPG on all the data. On the bottom, the user has restricted the data to only include records in which the weight is between 2910 and 3420 pounds and the car was made in either America or Japan. Our prototype immediately redraws the graph when the value-constraint buttons are pressed, and animates the transition from one graph to another.

Each row corresponds to a control for one of the attributes of the dataset. The row contains all the values that an attribute can take on in the dataset. For attributes for which a large numbers of values occur in the dataset, we currently perform very simple histogramming which creates a fixed number of ranges, each containing roughly the same number of unique values from the dataset.

As demonstrated in Figure 2, if one or more values of that attribute are selected, then any record with a non-selected value for that attribute is excluded in the resulting summary. For example, if 'American' is selected for the country attribute, then the resulting summary graph is computed based only on the records of American cars. If none of the values of an attribute are selected, then no record is excluded on the basis of that attribute. The controls for the current x and y axes are grayed out to indicate that they are the values being plotted. If the user selects values on these attributes, it restricts the graph, effectively zooming in on some portion of the x or y range.

Whenever a button is pressed, LIDS animates the transition from the current graph to the one for the new dataset. We found this animation extremely helpful for understanding the relative difference between two graphs.

2.3 IDS guidance

We have developed a simple interface to allow the user to communicate her interest by configuring one or more of the widgets in the top left of the interface. An example widget is shown in Figure 3.

Each widget defines what we refer to as a *discrepancy function* that is used to indicate what aspects of the summaries are currently most important to the user. A discrepancy function takes as input two summaries and returns a non-negative real number indicating how different those two summaries are; returning a zero means perfectly similar, and larger numbers imply greater dissimilarity. In this way, the discrepancy function indicates how interesting or unexpected one summary is relative to another. Of course, the discrepancy function is context- and user-dependent and may vary and evolve over the course of a session.

We have implemented several basic discrepancy functions that can be selected by the user. Intuitively, these widgets allow the user to indicate that she is interested in conditions in which the minimum point of the graph changes, the maximum changes, the values of the graph change in any direction, or the relative value of adjacent points changes. More formally, suppose that we have summaries S_1 and S_2, which correspond to sequences of points $\{\ldots\langle x_{1,i}, y_{1,i}\rangle\ldots\}$ and $\{\ldots\langle x_{2,i}, y_{2,i}\rangle\ldots\}$, respectively. Our implementation includes:

- min:

$$|\text{xOfMin}(S_1) - \text{xOfMin}(S_2)|$$

  where $\text{xOfMin}(\{\ldots\langle x_i, y_i\rangle\ldots\})$ returns x_i for the $\langle x_i, y_i\rangle$ with minimum y_i. (Our current implementation does not handle multiple minimums).

- max can be defined similarly:

$$|\text{xOfMax}(S_1) - \text{xOfMax}(S_2)|$$

- value: under the assumption that $|S_1| = |S_2| = N$,

$$\sum_{i:1\ldots N} |y_{1,i} - y2, i|.$$

  That is, we sum the absolute value of the difference between the y-values of the two summaries for each x-value.

- trend: under the assumption that $|S_1| = |S_2| = N$,

$$\sum_{i:1\ldots N-1} compare(y_{1,i} - y_{1,i+1}, y_{2,i} - y_{2,i+1}).$$

  where $compare(a, b)$ returns 2 if one of a or b is positive and the other is negative, 1 if one of a or b is zero and the other is not, or 0 otherwise.

Our platform is designed to make additions or changes to the available discrepancy functions straightforward with little programming overhead. Our implementation is modular so that adding a new type of discrepancy function simply

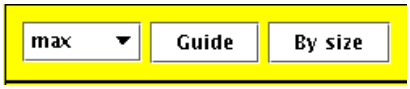

Figure 3: Example discrepancy function widget.

involves naming it and providing a function for computing the distance between two graphs.

Initially, LIDS generates four discrepancy widgets for the user, each with a different default type. The user can create more widgets by pressing the 'new' button. If there are too many to fit into the window that holds the widgets, they are put into a scrollbar window.

The user invokes the IDS guidance by pressing the 'Guide' or 'By Size' buttons on an discrepancy widget. Intuitively, the 'Guide' button directs the computer to find ways to manipulate the controls that will cause the visualized summary to change maximally for the given discrepancy function. In some contexts, however, the user will not be interested in large changes to the summary if it means restricting the data to a very small subset of the graph. Pressing the 'By Size' button directs the computer to factor in the amount of data that is being summarized in addition to how much the summarization changes.

More specifically, pressing either guide button causes the IDS prototype to compute a value v_i for each of the value-constraint buttons, described in Section 2.2. First, each v_i is assigned the value of the distance between the currently visualized graph and the graph that would result from pressing that value-constraint button. We normalize these values by subtracting the minimum v_i from all the values, and then dividing by the sum of the values. The values therefore add up to 1. If the 'By Size' button was chosen, we then multiply each value by the number of records that would be included after the value-constraint button is pressed, and renormalize the resulting values.

When the user presses a 'Guide' or 'By Size' button, that button is shaded in fully each of the constraint buttons are shaded proportionally to the v_i values, as shown in the second and third figures in Figure 1.

3. DISCUSSION

We now discuss limitations of and planned extensions to LIDS, as well as future directions of research in this area.

We plan to implement multiple aggregation functions, including average, median, min, and max. We plan to treat the choice of aggregation function as a user control, very similar to the value constraint buttons.

Currently, LIDS only considers changing a single control from the current settings when guiding the user. However, changing two or more controls might produce an important deviation from the current summary. The user may not not see this because neither change by itself is significant. A natural and efficient approach is an exhaustive, local search: first consider changes in two controls, then three controls, and so on as time permits. We plan to implement this search for guidance in the near future. Our current plan is to present the options from this search as a scrollable list of combinations of control changes, sorted in decreasing order of how much the graph will change using those changes. However, this list may be cluttered by closely-related options that essentially all do the same thing. These same problems have arisen in data mining, and we hope to borrow some of their solutions [1, 5].

We are also interested in extending our system to types of graphs other than simple line graphs. An example is to extend LIDS to show bar graphs or pie charts. A more interesting extension would be to graphs containing multiple lines, such as would be useful for comparing the performance of two different optimization algorithms over time on benchmark problems. This will involve developing additional controls and discrepancy functions (that operate on pairs of graphs which each contain multiple lines) for this task. For example, the user is likely to want to express interest in the relative position of the two lines and in how often they intersect.

For larger, more complex datasets, we expect more complex discrepancy functions than our current ones will be required. As an example, if the values one of the attributes can take on are arbitrary real values, then a natural visualization approach would be to create a histogram of the data in some way. It is possible that histogram-creation could be part of the summarization process. In this case, the discrepancy function would act on histograms. Another possibility is that the summaries are left as raw data, and the discrepancy function itself determines an appropriate histogram for the data.

In our experiments with our prototype, we have not encountered large computational requirements. Of course, as we expand the complexity of our techniques and apply our system to larger datasets, computational demands will increase. However, people often want to summarize datasets that are small enough to be extensively processed by computers quickly, and so it is reasonable to develop computationally intensive techniques, even if they are restricted to smaller datasets. Additionally, we expect that our IDS techniques (though not our current implementation of them) could be scaled effectively to large datasets. Note that the summaries, by their nature, are are likely to be compact and so many could be precomputed and stored. Furthermore, the system does not have to wait for the user to press a guidance button, but can perform a local search from the current visualization as a background computation.

4. RELATED WORK

We are not aware of prior research on the task of helping users quickly and reliably summarize data that offers computer-generated guidance resembling that proposed in this paper. Nor are we aware of any techniques which leverage the type of information about the user's current interest that our discrepancy functions elicit.

However, a great number of techniques and systems have been developed to help people navigate, explore, and produce summaries of multidimensional data. An important example is providing multiple, correlated views of some aspect of the data [2]. This is especially effective in conjunction with *brushing*, which allows the user to highlight a subset of the data [3, 10]. This technique allows the user to visualize patterns that are difficult or impossible to detect with a single view, and thus is more powerful investigative tool than the single view currently provided in our prototype. However, it does not fundamentally address the same issue that we have considered in this paper of helping the user decide what to investigate next.

One widely studied visualization technique that helps users

find correlations in large datasets is Parallel Coordinates ([7, 8]). This approach represents the data on a graph consisting of a number of parallel (typically vertical) axes, one for each attribute in the dataset. Each record in the dataset is drawn as a distinct line, intersecting each axis at the point corresponding to the value of that attribute in the record.

Parallel Coordinates are very useful for viewing the relationship between attributes. However, they also do not eliminate the need for computer assistance for the task of summarization. For example, when using a Parallel Coordinate system it is rather difficult for the user to know if she has missed something important. Similarly, there is no analogue in Parallel Coordinate systems to the discrepancy functions in LIDS for specifying which features of the data are currently of interest to the user. Another important difference is that Parallel Coordinates does not naturally allow the user to employ aggregation functions such as averaging by mean or median. We believe, however, that many of the ideas we have developed for guided summarization could be combined with these techniques.

MultiNav offers another approach for browsing multidimensional data [9]. Indeed, our method of restricting data using the value-constraint buttons described in Section 2.2 was inspired by MultiNav. MultiNav provides much more functionality than our control buttons, including that each record is individually represented in the attribute windows, though they are grouped into ranges. The size of the range indicates how many records have values in that range. These sizes change dynamically as the data is restricted by selecting various ranges. Furthermore, the users can interact with the system by sliding rods to reveal relationships between attributes. It seems likely that our guidance techniques could likely be extended to MultiNav given that our basic application is similar, though simpler, than MultiNav.

XGobi is a excellent example of a powerful visualization tool for high dimensional data [4]. It is a freely available and well engineered system. XGobi allows the user to visualize scatterplot and line drawings (primarily). The user can zoom, rotate, or tour the plot, as well as invoke an impressive range of powerful tools on it. XGobi's approach is to give the user realtime, interactive control of the visualization, allowing the user to continually refine the visualization in order to search the data effectively. Additionally, XGobi contains employs many methods, such as the use of motion, to provide information-rich visualizations to help the user spot interesting patterns for further investigation. This is quite different in spirit from our approach, which is to augment the user's search by having them communicate their intentions, and letting the computer perform most of the search. We see these as complementary lines of research.

Finally, it is worth pointing out here how summarization differs from other related methods for understanding large, multidimensional data sets. Tools for data exploration allow users to explore, rather than summarize, the data in order to find interesting or significant patterns. Data mining (e.g.,[1]) can be seen as an attempt to automate this process. We view summarization as a related but distinct task. In summarization, the goal is to determine the most important observations on the data, up to some level of detail. Again, this involves not only including important observations, but also not excluding more important observations. Summarization can be viewed as a kind of lossy compression, keeping exactly the most important observations.

5. CONCLUSION

A primary goal of this paper has been to propose and demonstrate a new kind of interaction in the area of interactive data visualization. Our techniques leverage the fact that if one is using a visualization to analyze data, then what one learns during exploration comes from how the visualization does, or does not, change when different data is visualized or is visualized differently. It follows that it is valuable to provide the users cues as to where they can find change.

Another goal of this project has been to focus attention on the problem of summarization as opposed to the exploration task that is the primary focus of most previous visualization systems and data mining techniques. In summarization the goal is to comprehend as fully as possible some aspect of a dataset. Thus, there is a need to develop techniques to alert people to important observations they might otherwise miss and, similarly, help give them confidence that they have found the information that is most important to them.

6. ACKNOWLEDGMENTS

We thank Joe Marks for his wisdom, wit, hyphens, and helpful suggestions on earlier drafts.

7. REFERENCES

[1] R. Agrawal and R. Srikant. Mining Sequential Patterns. *Intl. Conf. on Data Engineering*, pp. 3-14, 1995.

[2] M.Q.W. Baldonado, A. Woodruff, and A. Kuchinsky. Guidelines for using multiple views in information visualization. In *Advanced Visual Interfaces (AVI 2000)*, pp 110-119, 2000.

[3] R. Becker and W. Cleaving. Brushing scatterplots. *Technometrics*, 29(2), pp.127-142, 1987.

[4] A. Buja, D. Cook, and D. Swayne. Interactive high-dimensional data visualization. *Journal of Computational and Graphical Statistics*, vol 5, pp. 78–99, 1996.

[5] P. Domingos and G. Hulten. A General Method for Scaling Up Machine Learning Algorithms and its Application to Clustering. In *Eighteenth International Conference on Machine Learning*,pp. 106-113, 2001.

[6] H. Hauser, F. Ledermann, and H. Doleisch. Angular brushing of extended parallel coordinates. In *IEEE Visualization 2002 Proceedings. IEEE Computer Society*, 2002.

[7] A. Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1(2):69-92, 1985.

[8] A. Inselberg and B. Dimsdale. Parallel Coordinates A Tool for Visualizing Multivariate Relations. *Human-Machine Interactive Systems*. Plenum Publishing Corporation, 1991, pp. 199-233.

[9] T. Lanning, K. Wittenburg, M. Heinrichs, C. Fyock, and G. Li. Multidimensional Information Visualization through Sliding Rods. In *Advanced Visual Interfaces (AVI 2000)*, pp. 173-180, 2000.

[10] A. Martin and M. Ward. High dimensional brushing for interactive exploration of multivariate data. In *IEEE Visualization Proceedings*, IEEE Computer Society, pp. 271-278, 1995.