Integration of Unsupervised Clustering, Interaction and Parallel Coordinates for the Exploration of Large Multivariate Data

Published in IEEE, Information Visualization, 2004
Citations: 21 IEEE (4)

Parallel coordinates are widely used in many applications for visualization of multivariate data. Because of the nature of parallel coordinates, the visualization technique is often used for data overview. However, when the number of tuples to be visualized becomes very large, this technique makes it difficult to distinguish the overall structure.

This paper uses a classification approach, the self-organizing map (an unsupervised learning algorithm), to solve this problem by creating an initial clustering of the data. By initially only visualizing the resulting representational clusters, the inherited global structure can be shown.

Using linked views and allowing the user to perform drill-down, filtering and zooming on these representations reveals the single data items without loss of context.

Focuses on the tight coupling between the SOM algorithm and the parallel coordinate technique. Goal is to find and catalogue patterns, trends and clusters or similarities in large multivariate data sets that normally would be hidden from us. Combine a zoomable interface with linked views to provide the foundation for a focus-plus-context design. By integrating SOM, parallel coordinates and linked views to create a dynamic platform for exploratory analysis of large multivariate data.

Like many clustering algorithms, SOM relies on concepts like distance, similarity and average. Because of this, the algorithm faces problems when the input data is based on representations, and data sets containing non-numerical data cannot be successfully used in the standard algorithm.

Focus+Context Visualization with Flip Zooming and the Zoom Browser

Published in Proceeding CHI EA ’97 CHI ’97 Extended Abstracts on Human Factors in Computing Systems
Citations: 66

FLIP ZOOMING

A dataset that is to be visualized is first divided into a number of equally sized parts, called pages. For a large document each page is a section of the text. The pages are then laid out as thumbnail sketches on the display in a left-to- right, top-to bottom order. When users want to view a certain page, they can click on it to bring it into focus. The page is then expanded to a size suitable for reading. All surrounding pages are shrunk and re-arranged to make place, keeping the left-to-right, top-to-bottom page ordering intact.

HIERARCHICAL ORDERING

To make it possible to handle larger data sets, hierarchical ordering of the pages might be introduced. For instance, when editing a programming project, pages belonging to a certain function could be hidden when it is not in focus, leaving only a page with the function name visible.

ZOOM BROWSER

The Zoom Browser is the first implementation of the flip zoom technique. It is a web-browser (currently text- only), which can be used to download and view HTML, documents on the WWW. It is implemented in Java, which makes it easy to integrate with other web-based services.

When a web document is loaded into the browser, it is split up into a suitable number of pages that are added to the display. Users can then navigate through the pages using the flip zoom technique, and click on links in the text to load new documents. Previously viewed documents remain visible, providing a kind of history mechanism, but can be removed if the display becomes crowded.

SCALABLE VISUAL HIERARCHY EXPLORATION

A Thesis Submitted to the Faculty of the Worchester Polytechnic Institute

Citations: 19

This work addresses two questions

- The first one is how to most effectively translate the visual exploration operations such as zooming, brushing, etc., into a database understandable language such as SQL.
- The second one is how to store and manage the results of the database requests into the main memory and make subsequent memory access operations as efficient as possible.

Approach for the above questions

Move part of the on-line computation into a pre-processing phase. In this case, the front-end operations are reducible to a particular class of recursive unions of joins and divisions that operate on hierarchies. By using adequate pre-computation (i.e., organizing the hierarchical structure as what we called a MinMax tree), the recursive processing of the operations in this class can be reduced to range queries.

To make the second step efficient, they employ a main memory strategy to support incremental loading of data into the main memory. To show that incremental loading is efficient, given the set of operations it needs to. To further reduce the response time of the system, they have designed a speculative non-pure prefetcher that brings data into memory when the system is idle. The prefetcher is based on the property of navigation systems that queries remains “local”, i.e., given the set of currently selected objects we have a small number of choices of which objects are next selected. The property provides therefore “implicit hints” to the system. Additional hints might be provided by the specific data and the user’s navigation preferences as well.

Hierarchical Navigation Interface: Leveraging Multiple Coordinated Views for Level-of-Detail Multiresolution Volume Rendering of Large Scientific Data Sets

Published in IEEE, Information Visualization, 2005
Citations: 8 IEEE (2)

A new hierarchical navigation interface for level-of-detail selection and rendering of multiresolution volumetric data. The interface consists of multiple coordinated views based on concepts from information visualization as well as scientific visualization literature. With key features such as brushing and linking, and focus and context, it gives the users full control over the level-of-detail selection when navigating through large multiresolution data hierarchies. The navigation interface can also be integrated with traditional level-of-detail selection methods for more effective visual data exploration.

Multiresolution Data Hierarchies

Describes how we build multiresolution data hierarchies for large three-dimensional steady and time-varying data sets, using the wavelet tree and the wavelet-based time-space partitioning tree respectively.

Hierarchical Navigation Interface for LOD Selection and Rendering

Introduces hierarchical navigation interface, and then describes in detail the key features and main interactions that support the LOD selection and rendering of large data sets. The interface consists of three main interactive components: the rendering window, the overview map, and the treemap.

Brushing and Linking

In hierarchical navigation interface, brushing is used to select a subset of nodes from the current LOD for further operations such as join or split. Brushing can be performed in any of the three views and the result of the selection is highlighted in all views. Using brushing and linking, all the three views are linked together and updated dynamically whenever one of the views changes. This allows the user to detect correspondences and correlations among the three different visual representations.

Polaris: A System for Query, Analysis and Visualization of Multi-dimensional Relational Databases

Published in IEEE Transactions on Visualization and Computer Graphics
Citations: 386

Polaris, an interface for exploring large multi-dimensional databases that extends the well-known Pivot Table interface. The novel features of Polaris include an interface for constructing visual specifications of table-based graphical displays and the ability to generate a precise set of relational queries from the visual specifications. The visual specifications can be rapidly and incrementally developed, giving the analyst visual feedback as they construct complex queries and visualizations.

Generating Graphics

The visual specification consists of three components: (a) the specification of the different table configurations, (b) the type of graphic inside each pane, and (c) the details of the visual encodings.

Table Algebra

A formal mechanism to specify table configurations, and have defined algebra for this purpose. When the analysts place fields on the axis shelves, they are implicitly creating expressions in this algebra.

Types of Graphics

After the table configuration is specified, the next step is to specify the type of graphic in each pane. One option, typical of most charting and reporting tools, is to have the user select a chart type from a predefined set of charts. Polaris allows analysts to flexibly construct graphics by specifying the individual components of the graphics. However, for this approach to be effective, the specification must balance flexibility with succinctness. We have developed a taxonomy of graphics that results in an intuitive and concise specification of graphic types.

Visual Mappings

Each record in a pane is mapped to a mark. There are two components to the visual mapping. The first component determines the type of mark. The second component encodes fields of the records into visual or retinal properties of the selected mark.