Interactive Tools for Pattern Discovery

Tin Kam Ho
Bell Labs, Lucent Technologies
700 Mountain Ave., 2C425, Murray Hill, NJ 07974, USA
tkh@research.bell-labs.com

Abstract

Real-world pattern recognition problems contain rich context that must be taken into account in solution development. Beyond the core classification tasks, there are several common challenges including the need to correlate non-comparable groups of variables while preserving data proximity relationships within each group, visualization of data geometry, rapid trial of many methodological alternatives, and integration with existing infrastructure. We explore the effective methods and tools to meet with these challenges. We describe a graphical software, Mirage, as an experiment to address such concerns.

1. Introduction

In the process of searching for patterns in application data, we often run into many challenges beyond the core classification tasks. These include many issues in data preparation, formulation of the classification problems, and interpretation of the results. In this paper we describe our efforts in designing a software tool that assists in several stages throughout the development of complete solutions. We first present two cases leading to the tool's early development.

Design of optical fibers

Our explorations began with a rather atypical problem in pattern recognition involving the design of optical fibers. Properties of a cylindrical optical fiber depend primarily on its refractive index profile that specifies the change of the refractive index as a function of radius. Detailed mathematical models exist on the physics of waveguides that can predict the transmission properties of fibers with a given refractive index profile. Traditional designers of optical fibers draft a profile that they believe, by experience or by computation using these models, will produce the desired transmission properties. The profile is translated into precise specifications of the chemical mixture to be deposited to form each layer of the preform (by the MCVD, or modified chemical vapor deposition process[4]), which is then drawn into a fiber. There are several problems with such a trial-and-error based design process: the process is slow and expensive, the chance of success relies heavily on the designer's experience, which often means that improvements are incremental and it is difficult to obtain breakthrough innovations that are far away from prior experience.

A new design strategy is to use massive-scale virtual experiments to systematically search a space of design parameters. This can minimize the need for expensive sample manufacturing, and more importantly, it enables explorations of unfamiliar corners of the design space. However, while the models of physics give accurate prediction of properties for each alternative, the challenge of design is in the inverse problem: i.e., given specific demands on the fiber properties, which designs can best achieve such objectives? The models of physics are not directly invertible. A mathematical optimization procedure embedding the forward model can produce good candidate designs corresponding to different local optima. Yet there are still thousands of possibilities, representing different tradeoffs on the desired properties. Some profiles appear similar but straddle large discrepancies in the resulting properties, others appear dramatically different yet end up mapped to small neighborhoods in the property space.

Hence there is a clustering problem, i.e., a need to understand the similarities of the choices, in each of the relevant spaces: the space of design parameters, the space of profile shapes, and the space of resulting transmission properties. Moreover, there is a need to understand how clusters in each space relate to those in another. Each space has its own unique scale: while profile shapes can be compared using a standard norm, they are not directly comparable to the transmission properties which consist of several subgroups with different units.

Stars or galaxies

An astronomical survey on deep sky objects [7] returned images of millions of very faint objects. The primary scientific goal requires careful characterization of the distribution of observed shape distortions of far away galaxies. But before that, those galaxies need to be distinguished from the foreground stars which could be at similar apparent brightness. The discrimination task depends on tiny differences in color and shape.

However, a difficulty remains: the survey is to an unpreceded depth, meaning that most objects have never been observed before and nobody knows their true classification. How does one build confidence in the result of the classifier? There is some prior knowledge one can count on, such as the expected count of objects of each type in each bin of magnitude, or the luminosity function. But how can one use this knowledge in the development of the classifier? Here we are again concerned with several perspectives on the same data: the characterization of the objects in the color space, in the shape parameters, and in the brightness statistics.

2. Challenges for an Analysis Tool

The above two cases highlighted several issues that are peripheral but critically important to classification, including the existence of noncomparable groups of variables, the need to relate clusters arising from different perspectives, and validation of results with side evidences. We see such issues occurring in many other application contexts, including other simulation studies for understanding optical networks, performance diagnosis for wireless communication systems, and analysis of biomedical data. Generalization of these experiences pointed to several crucial needs of a pattern discovery tool:

Separate treatment of noncomparable groups of variables

Many application problems contain several groups of variables that are not necessarily comparable on the same scale. For astronomical objects these can be images, spectra, light curves, surface temperature, orbital period, and velocity. Similarly, a database of people may include portraits, audio clips, finger prints, as well as age, height, weight, education records, etc. Proximity between the objects in the original, natural scale of each group has important meaning and need to be preserved, but at the same time we need to examine the correlation between different groups of attributes. Thus artificially combining all variables into the same feature space with a standardized scale may not be the best treatment. Classification needs to be applied to each relevant subspace, even if it means that many variables will be treated singly in its own scale.

Versatile visualization utilities allowing many perspectives

It is well known that visualization can be a great aid in the early stages of pattern discovery, to help "sanitary checking" in data preparation, verify correctness of preprocessing steps, clean up undesirable artifacts, and choose relevant samples. Visualization also enables an initial exploration to spot explicit patterns, select potentially useful features, try different normalization schemes, and suggest choices of classifiers, clustering algorithms, or trend models.

Understandably it is difficult for a single visualization method to satisfy all these needs. Ideally, data visualization should allow large flexibility in the choice of perspectives, ranging from basic statistical graphics such as histograms and scatter plots, to more sophisticated views of high dimensional spaces, multimedia signals, maps, and auxiliary data structures like graphs and trees. Many innovative data views have been produced in visualization research. However, most visualization systems are implemented as a toolkit, i.e., a loose collection of a library of plotting modules. Better uses of these can be made if there is a connecting architecture where data relationship can be easily tracked between modules.

Mechanism for rapid and convenient feedback

It often happens that classification tasks in an application can be set up in different ways, by defining the classes differently, choosing different features, and applying different feature transformations. Coupling this with many choices of methods available for each task and different parameter options within each method, there can be a huge number of alternative results that need to be examined. An example is in text categorization. The definitions of categories can be very broad or very narrow, with direct influence on classification accuracy. The tradeoff between specificity and accuracy can be manipulated to serve different needs depending on the purpose of the application system.

Thus there is a need for an application developer to quickly apply different alternatives and examine their results for highest validity and easiest interpretation. While a specific processing pipeline can always be constructed using general-purpose languages and scripting facilities to semi-automatically try out the alternatives and display the results, this need is common enough among many pattern recognition practices that a generic, reusable tool will be most desirable.

Support for data access, collaborative development, and continuous growth

A good data analysis tool should provide linkage to existing data management infrastructure with minimum effort, support easy sharing of exploration procedures and results among a collaborative team, and enable continuous growth by addition of new visualization or analysis tools.

Having been exposed to such common problems from highly diverse application backgrounds, we set out to study how to construct a tool that would satisfy these needs and can become a key utility in the process of developing pattern recognition solutions. Our efforts resulted in an implementation called *Mirage* that has been in public circulation

for over a year [1] [2][3]. We describe below the key features in Mirage that are motivated by the identified needs.

3. Tracking Data Correlation with Mirage

Mirage is a Java-based tool that is organized around a command interpreter which receives action commands from textual input or a graphical user interface. The action commands are for loading data, incremental import of new entries and new attributes, simple attribute manipulation, and activating several embedded classification routines. While the tool can be run in the background as a scripting facility for the supported activities, the most important functionalities are on data visualization provided through a graphical display.

The display presents a stack of canvas pages each of which can be subdivided arbitrarily, via horizontal or vertical splits, into rectangular cells (Figure 1). Each cell can be loaded with any particular data view module via simple drag-and-drop operations. Each view module shows the data in a specific way, which can be a table, a simple histogram on one attribute, an x-y scatter plot of two numerical attributes, a tree of clusters, or position markings of each data object on top of an image or map background. Each module provides its own control commands to manipulate the specific method of data presentation. In addition, all view modules implement the same Java Interface `ActivePanel` containing the following commands:

getSelected()	clearSelected()
highlightDataEntry()	colorDataEntry()
clearHighlights()	clearColors()
changeToMonochrome()	changeToColor()

Simple as they are, these several commands provide the essential "brushing" links between *all* data views. Coordinated sequences of selection and highlighting can be defined with a module and broadcast to other views via these commands. Coupled with the flexibility of data views, the interface becomes a very powerful tool for studying correlations. Several example packages of operations provided by Mirage are as follows:

- The histogram view module supports a bin walking action, which selects each consecutive bin in turn and broadcasts the selection to all other views that are visible. A histogram is essentially a trivial cluster structure on a single attribute, and traversal of this structure allows tracking of correlation between that particular attribute and others.
- A graph view module includes a layout tool that can project a cluster tree computed in arbitrary dimensions onto 2D. The graph provides a skeletal summary of the data distribution in the relevant feature space in the form of a set of discrete principal curves. A broadcast walk along each curve enables tracking how that particular trend relates to other attributes (Figure 2).

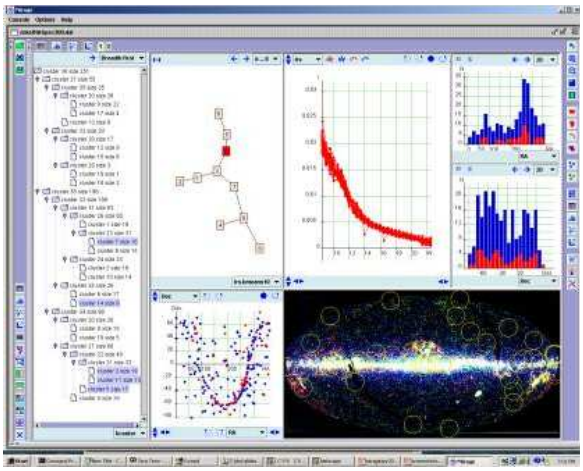

Figure 1. A Mirage screenshot, showing several linked views. A subset of data is selected in a cluster graph and broadcast to other views.

- A tree view module includes standard depth-first or breadth-first traversals of a cluster tree. When broadcast to other views, they show the effect of adopting cuts at different levels of a cluster hierarchy.
- Distinct coloring of each piece of a pie chart for one label variable, when broadcast to other views, shows instantly the effects of class definition according to that label. Degree of separation of those classes in the concerned attribute spaces can be examined. Dynamic importing of a new class label allows classes to be redefined and their effects rapidly checked.
- High dimensional vectors can be shown in parallel coordinates. A cycling feature with broadcasting enables users to examine each vector individually, and at the same time its location in the context of other variables. This is particularly useful for sparse data sets where individual samples are very expensive to obtain and warrants detailed investigation.

Any user can supply a new module implementing the same interface and include it into Mirage with an entry in a customization file to specify its name and icon. With such a simple interface, we encourage users to develop new data views that can support their unique visualization needs yet at the same time can be easily linked to other generic tools. The utility of the interface has been demonstrated by two contributed custom views, one for displaying astronomical images featuring sophisticated image processing commands [3], another for visualizing computer networks featuring an intelligent graph layout algorithm.

The unifying theme for the data exploration operations in Mirage is the analysis of correlations of data proximity

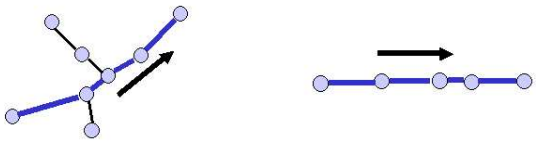

Figure 2. A walk following a principal curve in one space may or may not synchronize with a uni-directional walk in another space depending on correlation between the two spaces.

in different projection subspaces [2]. We observe that many data mining and pattern discovery tasks are covered under this theme. A graphical tool implementing this theme enables a highly intuitive understanding of core tasks in pattern recognition, which benefits both the solution developers and the domain experts. It also allows domain experts to participate in the process easily, often with immediate feedback on their opinions in feature selection, class definition, and proposal of trend models.

4. Supporting Facilities

Mirage is written with strong commitment to modularity and extensibility. The Mirage tool can be invoked as a single Java class by a wrapper utility that provides access to existing databases [3], or implements dynamic data updating via messages [6]. Handles are available to the wrapper to pass commands into the Mirage interpreter for data loading and refreshing, including adding columns or rows to the existing data matrix, or replacing it entirely while preserving the choices of views. Columns added can show newly observed or computed attributes, including classification decisions or predictions from external algorithms. Rows added can be additional samples from the same source, which can be used to check the correctness of a conjecture or prediction. Continuous adding and removing rows turns the software into a monitoring tool.

In addition, Mirage has a slot for plugging in an "Action" panel, which can be used to encapsulate user-defined exploration operations on top of existing primitives. Either the wrapper handle or the Action panel can be used to embed Mirage into a more sophisticated decision support platform.

Finally, data analysis is seldom an individual effort. To enable easy repetition and sharing of exploration commands and results in a research team, Mirage provides text-driven commands and scripting facilities. Scripts can be passed around for replay, allowing users to reproduce colleagues' explorations. They can also be systematically constructed by simple programs to make animations.

5. Conclusions

We described a graphical data analysis software, Mirage, that supports many tasks in knowledge discovery under a unifying theme. We are in continuous experimentation on ways to improve it towards a rich tool for development of pattern recognition solutions in various degrees of automation.

A convenient visualization tool like Mirage can provide better insight into the data geometry, and to provide domain experts easy access to the analysis process. Mirage has been included as a key tool in the Virtual Observatory [3]. In our own environment Mirage has enabled discovery of unexpected clusters and outliers in several communication engineering data sets, e.g., the effects of design parameters on signal power evolution in an optical line system. The advantages of the tool are especially obvious for data sets involving spatial distribution of objects on an image or a map.

Early applications of Mirage confirmed the effectiveness of many of our designs, and suggested improvements in several ways. Programmable layouts, software state saving and restoration, detailed logging of operations performed, and more automated explorations are among the most desired. We are also investigating better ways to cooperate with external analysis code while maintaining a simple and modular core architecture, and methods to allow more flexible joins among multiple data sets [5]. Finally, viewing data in high dimensional spaces is by itself a difficult challenge. We hope that by providing an open, easy interface with many support structures, we can encourage experiments of other innovative data visualization and exploration methods.

Acknowledgements

The project has benefited from collaborations with Lawrence Cowsar, J. Anthony Tyson, David Wittman, Alex Szalay, Samuel Carliles, and Wil O'Mullane. Their insights to the application problems and suggestions of features are gratefully acknowledged.

References

[1] `http://www.cs.bell-labs.com/who/tkh/mirage`.

[2] T.K. Ho, Exploratory Analysis of Point Proximity in Subspaces, *Proc. of the 16th ICPR*, Quebec City, Canada, Aug 11-15, 2002.

[3] S. Carliles, T.K. Ho, W. O'Mullane, VO Enabled Mirage and the IVOA Client Package, *Proc. of Astro. Data Analysis Software & Systems XIII*, Strasbourg, France, Oct 12-15, 2003.

[4] S.R. Nagel, Fiber Materials and Fabrication Methods, *Optical Fiber Telecommunications II*, S.E. Miller, I.P. Kaminow (eds.), Academic Press, San Diego, CA, 1988.

[5] C. North, B. Schneidermann Snap-Together Visualization: A User Interface for Coordinating Visualizations via Relational Schemata, *Proc. Advanced Visual Interfaces 2000*, May 2000.

[6] M. Thottan, K. Swanson, M. Cantone, T.K. Ho, J. Ren, S. Paul, SEQUIN: An MPLS Network Monitoring System, *Bell Labs Tech. J.*, **8**, 1, 2003, 95-111.

[7] J.A. Tyson, I. Dell'Antonio, D. Wittman, The Deep Lens Survey, `http://dls.bell-labs.com`.