Learning from Users' Interactions with Visual Analytics Systems

A dissertation

submitted by

Eli T. Brown, B.S., M.S.

In partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

in

Computer Science

TUFTS UNIVERSITY

August 2015

Advisor: Remco Chang

For my family: the one that made me and the one that I made.

Learning from Users' Interactions with Visual Analytics Systems

Eli T. Brown

Advisor: Remco Chang

Experts in disparate fields from biology to business are increasingly called upon to make decisions based on data, but their background is not in data science, which is itself a separate field requiring years to master. Machine learning approaches tend to focus on finding a black-box answer, which the user may not understand or trust. Visualization on its own can leverage the power of human insight, but may miss out on the computational power available with automated analysis. Visual analytics researchers aim to provide tools for domain experts to find the patterns they need in their data, and have recently been interested in systems that combine the two approaches. One promising method is to blend the best of visualization and machine learning by building systems that provide interfaces for users to explore their data interactively with visual tools, gather their feedback through interaction mechanisms, and apply that feedback by using machine learning to build analytical models. In this dissertation, I discuss my research on such systems, showing techniques for learning from user interactions about the data and about the users themselves. Specifically, I first describe a prototype system for learning distance functions from user interactions with high-dimensional data. These distance functions are weighted Euclidean functions that are human-readable as the relative importance of the dimensions of the data. Observing that users of such systems may be required to review large amounts of data to be effective, I propose an algorithm for better leveraging user efforts in this interactive context. Next, I show an adaptation of the interactive learning prototype for text documents, with a study showing how to make use of the vector representation of the distance functions for numerically examining the analysis processes of the participants. Turning the focus of the learning back onto the user, I provide a proof-of-concept that shows how models of users as opposed

to data can be learned from user interactions. Finally, I introduce the sketch of a framework for future systems that will empower data stakeholders to find the answers they need without leaving their comfort zone.

Acknowledgments

It has long been an ambition of mine to earn a Ph.D. I dove in without being sure how the whole process was supposed to unfold, never mind how it actually would. Right from the beginning, my adviser Remco Chang has been there to help me ponder the process and possibilities, to shepherd hare-brained ideas into quality research, and to strengthen my clarity of thought, my writing, and my scholarship, bringing me to the point where I could write this dissertation. I am forever grateful to him for his dedication.

Of course, I never would have been in a position to meet Remco and pursue a graduate career in the first place without the love and support of my family and my wife. My parents have provided me with an invaluable foundation of curiosity and perseverance, and I have been spurred on by the stories of family greatness from both sides, often told by Tantie Julia or Grammy Miriam. My sister is perhaps the only person alive who understands how my brain works at its best and worst. And as for my wife — I could not possibly express my gratitude to Rose, whose excellence inspires me and whose endless support and kindness has lifted me up from all the dark corners of this Pandora's box that is graduate study. I am forever grateful to them for their love. I would also like to express my appreciation for some dear friends, Derek Kreymer and Nick Finch, who checked-in on me and worried over me, even when unnecessary.

I learned a tremendous amount during my internships working with exceptional people at the Pacific Northwest National Lab (PNNL) including Alex Endert, Bill Pike, Kris Cook, and Russ Bertner. My funding from MIT Lincoln Labs came with insightful and inventive collaboration from Stephen Kelley. Finally, my internship at Microsoft Research gave me the opportunity to learn from an extraordinary group of people including my mentor Helen Wang, along with Michael Gamon, Ashish Kapoor, Sumit Basu, Saleema Amershi, Steve Drucker and Danyel Fisher.

I would like to thank the many researchers in the Visualization community who have been so welcoming in the halls of conferences, and provided me with fantastic advice, including Ben Shneiderman, Chris North, Michael Gleicher and Ross Maciejewski. I would also like to thank the friends I have made along the way to commiserate victories and failures, including Chad Stolper, Danielle Szafir, Michael Corell, Megan Monroe, Sarah Turnbull and Francis Williams. In the computer science community at Tufts there have been three professors whose mentorship has been invaluable: Rob Jacob, Lenore Cowen, and Norman Ramsey — I am sincerely appreciative of their efforts and skill at guiding me during my graduate career.

This dissertation represents collaborative efforts with an exceptional group of researchers without whom it would have been impossible: from the Visual Analytics Laboratory at Tufts (VALT), Jordan Crouser, Alvitta Ottley, Lane Harrison, Fumeng Yang, Shaomeng (Samuel) Li, Jieqiong (Helen) Zhao and Quan Lin; from the Tufts HCI Lab, Dan Afergan, Evan Peck and Beste Feliz; from the Computer Science Department at Tufts University, Jingjing Liu and Carla Brodley; from the Math Department at Tufts University, Misha Kilmer; and from internships and collaborations, as mentioned above.

Finally, I would like to thank the members of my committee for agreeing to be part of this process: Professors Robert J.K. Jacob, Greg Crane, and Misha Kilmer of Tufts University, and Professor Chris North of Virginia Tech. I appreciate their valuable time considering and measuring my contribution.

This work has been supported by the National Science Foundation, Draper Labs, MIT Lincoln Labs, the Pacific Northwest National Lab, and Microsoft Research. The results and findings expressed here are not a representation of the views of these entities.

ELI T. BROWN

TUFTS UNIVERSITY

August 2015

Contents

List of Tables

List of Figures

Chapter 1

Introduction

People in disparate fields and professions from government work to medicine, education to biology, marketing to engineering are expected to make decisions with data. Because of this growing cultural trend, data is collected on myriad aspects of life and in staggering quantity. But the people who must depend on the insights encoded in the data are not trained in how to extract them. As an example of the growing problem, in the scientific community, retractions are on the rise [CORE08] and one of the main causes of retractions is mistaken statistics [Eco13]. It is asking too much of professionals who are studied and steeped in the knowledge of their domain to learn the mathematical or statistical modeling required to do the analysis necessary for their complex experiments.

Approaches to this problem can involve training the domain experts or hiring data analysts separately and training them in the domain. Both options require sorting through communication issues between groups with disparate training, and both options can be expensive in both time and money. There is already high demand for the data analysts. Davenport and Patel wrote in the Harvard Business Review that the sexiest profession of the 21st century will be "data scientist" [DP12]. But there is no consensus about what that job entails or its qualifications. Listings under that title include jobs for bachelors in the sciences, PhDs in statistics, and marketing majors with an analytics focus. The fact is that most people not only are unsure about how to do good data science, but about what it is in the first place. Further obstacles to hiring help to deal with "the math" include the expense, and the communication gap — data scientists cannot also be expected to be experts in their

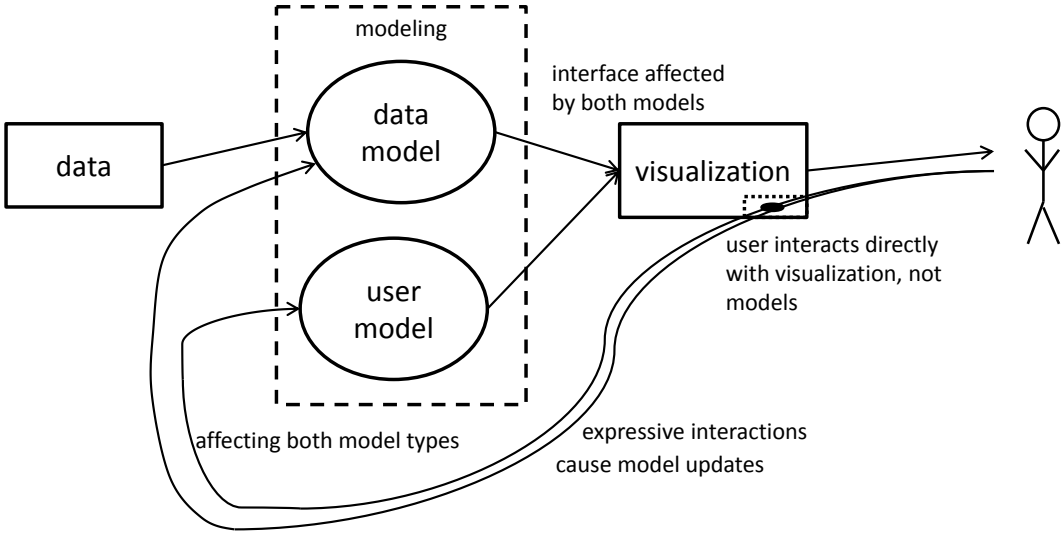


Figure 1.1: This diagram shows the user affecting models of both the data and the user, while only interacting directly with a visualization. A user's interactions might tell us something about the data, but we can also create a model of the user. Note that both affect the visualization. The data model may influence what data gets displayed, how it is laid out, or what we can suggest to the user. The user model might allow adaptations of what options are available, or directing the user toward the data of his or her expertise.

employers' domains.

Training domain experts and hiring data experts are both expensive and difficult, but there is another solution to this pervasive data analysis problem. It is the one I explain and advocate for in this thesis: create tools that domain experts can use to perform the data analysis themselves. Experts could interact with their data in a comfortable way, using tools that are tailored to their understanding of their own data, and the necessary models could be built for them automatically, behind-the-scenes, to protect them from needing to study a completely separate discipline. This final approach underpins my research and is the broad, guiding motivation of this dissertation. There are two prongs — learning about data from user interactions, and learning about the users themselves from their interactions. Figure 1.1 shows this approach pictorially. A user can interact directly with a visualization as opposed to interacting with models or parameters, and the interactions are fed to modeling in the back-end. Some of these interactions are expressive and we respect the semantics of the user's intentions to update a model of the data. The broader set of interactions can be used to learn about the user him- or herself. Either or both of these models can be used

to in-turn affect the visualization or interface so that the user's intentions manifest in an improved experience and improved data model.

In this dissertation, I discuss the possibilities for how systems with the automatic data modeling goal can be constructed and narrow the focus sufficiently to explain my contribution. The remainder of this introduction provides context for the chapters: (1) an exposition of related academic work, (2) presentation of a prototype system that learns from user interactions with high-dimensional numerical data, (3) a method for making the user experience more efficient by guiding the analysis from the machine learning, (4) an application of that technology to text analytics for intelligence with a discussion of how to look for user interaction patterns with these types of systems, (5) an example of what we can learn when we turn the model building exercise upon the users themselves to predict performance and even personality, and finally (6) a discussion of how the fields of machine learning and visualization can work together. Note that although I lead the research efforts described in this document, I use the pronoun "we" when describing the work, as is common practice in academia, to acknowledge the contributions of my collaborators.

1.1 Learning About Data

The well-known problem of the increasing volume of data and our increasing expectations for its power is tackled by both the visualization and machine learning communities. The machine learning community offers sophisticated algorithms for building models and making predictions. These algorithms may involve complex, opaque parameters, and present themselves to data-stakeholders as black-boxes. Expertise in these algorithms is well outside the realm of a data stakeholder's domain expertise. They may be glad not to have to understand how the result is achieved, but they may also not inherently trust a result handed to them from an opaque process. Further, many problems are beyond the scope of what a computer can handle automatically, especially for cases in which ground-truth, expertly-labeled data examples are scarce.

Human reasoning is better suited to those open-ended problems, where the target is shifting and the desired pattern is unknown. The visualization community enables data

stakeholders to take control of their own analytics process by providing techniques and systems that allow them to leverage their expertise to explore the data and discover patterns on their own. However, visual analytics systems generally do not build and export useful computational models for explaining patterns and re-using on other data.

While machine learning research focuses on leveraging the machine's raw computing power to discover patterns, visual analytics works to leverage the natural visual perception ability of humans. Leveraging the best of each could provide powerful results. A spectrum of approaches exists for this combination. On one end, a machine learning algorithm uses a person as a source of information about the data, directly asking the user to label the data. Some of this work can be classified as interactive machine learning, a subfield in which machine learning is applied to general software tasks and the user may not be aware of providing labels, e.g. a spam filter. On the other end, visualization systems adapt toward machine learning by showing the results and providing an interactive mechanism for changing parameters of the algorithms (e.g. iPCA [JZF+09a]). Our approach comes closer to the middle — the user works with an interactive visual system for data analysis, with tools that are appropriate and comfortable for the data at hand, and those interactions are converted into feedback for a machine learning algorithm that can run behind-the-scenes. This algorithm learns a model about the data that encapsulates the user's feedback, progressively refined by iterating. The visualization reflects the machine learning model, so as the user feedback improves the model, the user can see the interactions improving the visual representation. After the user is satisfied with the visual representation, the software can export the learned model as an analytic representation that models the insight supplied by the user during the analysis. This approach requires machine learning algorithms that can be used incrementally at an appropriate speed for an interactive context. It requires visualization methods that enable exploration and discovery, and interaction methods that capture the feedback needed by the machine learner.

The first prong of this thesis demonstrates that this middle approach is feasible. In Chapter 3, I present a prototype system that works with high-dimensional numerical data. Users interact directly with data points in a visualization while the system, Dis-Function, automatically learns a distance function that encapsulates what the user has expressed about

the data through interactions. The visualization consists of a two-dimensional projection of the high-dimensional data, and captures the relationships between each pair of points in terms of their distances from each other. We start with a naive notion of distance, i.e. the Euclidean distance. As the user progressively, iteratively provides feedback about what inter-point relationships are being visualized inconsistently with his or her understanding of the data, this feedback is processed by a machine learning algorithm. The algorithm learns a distance function which warps the data space to enforce the updated relationships. We apply the new distance function to calculate new relationships between the points and then generate a new visualization for another round of feedback. The user is protected from learning how the algorithm works and from manipulating model or algorithm parameters. The experiments in Chapter 3 show that the system can be used to iteratively produce an effective distance function and corresponding visualization. We also show that the updated distance functions can be computed in a timeframe that allows using this in an interactive system for data sizes of approximately 200 points.

1.2 Extensions and Applications

This dissertation continues with the goal of improving techniques for building models about data with visual interaction beyond the Dis-Function technique itself (see Chapter 3). We provide an extension that could make the interactive learning mechanism more efficient by encouraging users toward the most effective feedback for the model learning. In addition, we provide an adaptation of the Dis-Function technology to text data and demonstrate its application to intelligence data. Finally, we consider the trail of models that the interactive process produces and what it may tell us about the analytic processes of the users.

1.2.1 Active Learning

Despite the power of the interactive learning techniques we have presented, a data domain expert may find the amount of data they are required to review to be too great. In particular, the analyst is required to compare sets of data points. In the worst case, this could require reviewing every pair each iteration. The data points may each have numerous facets

to consider, or may be related in deep and subtle ways. For example, a doctor researching effects of disease on different patients may consider each patient a data point and may need to read two full medical charts for every patient comparison. One potential mitigation for this problem is to allow the machine learning algorithm a means of communicating to the user to create a more efficient experience. The machine learning community devoted to studying machine learning algorithms that work with limited input from users by communicating what input would be best is called *active learning* (see the Discussion in Chapter 7 for more information). In Chapter 4, we present work in progress that explains an approach to using this type of learning algorithm, but built for the interactive model learning context of Chapter 3. Unlike most active learning algorithms which focus on areas of uncertainty for the model, ours is guided by the user. He or she starts a direction of inquiry by choosing a data point of interest. The algorithm responds by helping to narrow the user's focus on points for comparison, aiming to help the user make the strongest update to the model. Our work on this algorithm is still in progress. Though the algorithm itself remains unproven, we present evidence of its likely effectiveness. Further, we believe that the approach we present, i.e. a mechanism for a user-guided active metric learner, is worth presenting as it represents a departure from the general active learning literature.

In our active metric learning algorithm, EigenSense, we consider the matrix of pairwise distances of all the data to be a representation of the relationships between the data under a given model. It encapsulates both the distance metric model and the data in one matrix. We look at this distance matrix like the transition matrix of a graph and, modeled after the approaches of population studies in biology and the PageRank algorithm, we use the dominant eigenvector as an approximation of its structure. When a user has selected a point of interest, we examine the sensitivity of that dominant eigenvector with respect to the chosen point in order to predict what user inputs could most affect the model structure. Our evaluation compares the algorithm's selection of those strong changes against ground truth of how much those inputs actually change the model. We find that EigenSense predictions are correlated to the quantity they are estimating, the actual amount of model change.

1.2.2 Intelligence Application

While Chapter 3 focuses on high-dimensional real-number data, there are other types of data for which this approach is germane. One example is text data, which abounds and which is of critical interest to multiple parties, including for intelligence and anti-terrorism purposes. This common type of data provides some advantages, especially visually, but also presents challenges. Representations of text data are generally per-document vectors (see bag-of-words models as a starting point [MRS+08b]), which means these data can be plugged into many learning algorithms. However, caution is necessary because bodies of text tend to yield extremely high-dimensional spaces with sparse vectors. Whereas options for visualization on numerical data are myriad with no certain standard, for text data there is a more limited number of very strong visualizations. Text is convenient for representation because so much information about the meaning of the data is encompassed by the raw data's most obvious visual representation: the words themselves. A user seeing a word on the screen does not just see the data point that the word may represent, but understands immediately what that data point is about.

In Chapter 5, we extend the techniques of Chapter 3 to collections of text documents and discuss further ways to use the type of model being built by this method. When analysts work with text data corpora, they must make sense (sensemaking) of piles of documents by looking for relationships and trying to discover themes and topics to compare against existing mental models. One way that tools help analysts perform these tasks is by visualizing document collections in spatializations [WTP+95]. These visual representations show many documents at a time, each represented by some glyph, and place the documents relative to each other based on their relationship (generally by words they have in common).

Systems may include techniques that allow the analyst to change or specify those relationships by engaging in model-steering to update that model that calculates the distances. For example, highlighting words across several documents might affect the model's information about the importance of those words, and thus affect the spatial layout. To work with documents based on such a layout, though, an analyst must read through a large

amount of text. Each data point in the layout is a stand-in for a potentially long document that the user may have to review for each comparison. One way around this is the use a keyword-centric spatialization. Instead of using a glyph to represent an entire document, a keyword-centric layout visualizes a set of keywords extracted from the document corpus directly, positioning them to encode their relationships based on documents that contain them. In Chapter 5 we present a prototype system that shows such a keyword layout and allows users to manipulate keywords and implicitly and automatically learn new models. The keywords in that chapter are extracted from a dataset of short documents provided by the intelligence community that contain a fictitious terrorist plot. They are positioned based on their co-occurrence in different documents, and the constructed model of the importance of each document in that calculation. In a study with thirteen participants, we started with a model that evenly weights all documents in importance, but as the participants provided feedback on the inter-keyword relationships, the model was adjusted. The participants successfully used the tool to uncover the fictitious plot.

1.2.3 Model Trails

In addition to proposing a tool for finding patterns in text data, we begin to consider the paths the analysts took in their thinking process. In the second contribution of Chapter 5, we leverage the fact that the models created during each participant's process are vectors in a high-dimensional space and thus we can visualize their relationships with a projection into a two-dimensional spatialization, just as we do with the data themselves. Such a visualization enables us to explore the actions of all the participants and see how their investigations unfolded, providing a novel method for studying provenance analytically. We visualize each model that any participant created as a point in the space of possible models, and layout the points based on their similarity to each other. Connecting the points for a given user, we draw lines that reveal (by mouseover) the actions of the user that lead to the transition from the start state to the end state of the line. This visual tool allows us to compare and contrast different participants' investigations. In doing so, we are able to ask questions about the user behaviors, and transition our discussion to modeling the users themselves.

1.3 Learning About Users

So far, we have discussed means to help data stakeholders discover insights in their data. We enable humans to communicate with the machine by empowering their interactions with data to build analytic models. However, the choice of data that users interact with, and the way they interact could tell us about them individually. There is an area of research in the visualization community called *provenance* devoted to this study. In particular, researchers in this field believe that the process of an analysis is as important as the result [NCE+11]. A current of studying the analytic process of the user (i.e. the provenance of the analysis) runs through this entire work. First, in Chapter 3, we choose a model-building method because it is human-readable. The distance functions that result from the analysis process are artifacts of the user's thinking. They encode the mental model and they are readable to the extent that they reveal what the user found most important overall. Second, in Chapter 5, we consider the provenance of the models more deeply, not only considering the final model, but examining the models created during the process and the user's paths between them. By making use of the fact that the models are vectors, we visualize the analysts' progress through "model space", a numerical representation of their analytic provenance. In Chapter 6, my interest in learning about users comes full circle. We use interaction data to learn not about data but *about the users* themselves.

Data analysis is best served by a human-computer collaboration - the human is in control and provides the actual insight, but the computer serves the human's needs to find and present data. While the computer can use a full complement of visual (and audio) mechanisms to convey information to the user, the person driving the process has a limited means of communicating with the machine. Just telling the machine what to do does not give the software a chance to help the user get the most out of the analysis. Research had shown that interactions can reveal a user's reasoning process before the work of Chapter 6, but accomplished only through painstaking manual analysis of logs. To be able to create systems that help individual users, technology is needed that can learn models based on the user interactions automatically.

In Chapter 6, we test the question of whether or not computer software can be

programmed to automatically detect facets of its user. With an experiment that asked participants to perform a visual search task in the form of the children's game Where's Waldo, we verify that we could predict participant performance and even infer some aspects of personality by applying machine learning algorithms to interaction data. First we make the case for this approach by visualizing the interaction data to reveal strongly visually salient differences between the patterns of fast and slow user groups. Next, we take an analytic approach and encode the interaction data into four different sets of machine learning features. The encodings and techniques were chosen not just to seek accuracy of prediction, but also as a modeling exercise, seeking understanding of the nature of the interaction data. Each encoding takes into account different facets of the interaction experience: raw mouse movements incorporate the small unintentional movements while sequences of button clicks capture intentional user exploration, and two encodings based on the state of the software are focused on what the participant saw while performing the task. Making use of standard machine learning algorithms, we build classifiers for predicting a participant's performance at the Waldo task based on his or her interactions. We found that with between 62% and 83% accuracy, depending on the combination of machine learning technique and interaction encoding, we could predict if a participant would be among the faster or slower group overall. Beyond performance, we apply the same encodings and algorithms to personality information about the participants, as determined by surveys administered for the experiment. We found that, we can infer, to a lesser extent than performance, aspects of participant personality including locus of control, extraversion and neuroticism depending on the choice of technique, a result which is a consistent with existing literature on personality factors that affect interactions with visualization. Finally, we showed that we can attain strong results even if we limit the time that the algorithms could observe participants. In one case, limiting the learning algorithm to a quarter of the average task completion time still yields 95% of the final accuracy percentage. Overall, Chapter 6 demonstrates that there is potential to use interaction data to automatically infer information about users and possibly their processes or strategies. These technologies are a step forward in realizing a pathway for a computer to gather information about its human collaborator, paving the way for mixed-initiative visual analytic systems.

1.4 Future Analytics Systems

This dissertation marks progress in a number of technologies with strong promise for the future of visual analytic systems. Much work is needed to realize the potential of leveraging machine learning for learning about users and their data. Future work and innovation should be driven by applications. Applying these principles to solve real-world problems is the only way to demonstrate their power in leveraging human and machine at their best strengths. For the purpose of encouraging this future work, in the Discussion (Chapter 7), I provide a sketch of a framework for how machine learning and visualization techniques fit together. Finally, I explain the types of interaction mechanism and provide an overview of machine learning with a focus on how it can be integrated to benefit visual analytics.

Chapter 2

Related Work

In order to appropriately place this dissertation in context, we discuss related previous research. The two main branches of the work cover learning about data and learning about users. The connection between the two branches is work in machine learning and analytic provenance. We discuss these two areas first and then explore the related work in the area of machine learning and visualization together. To better frame our work with text data, we provide further background in text analytics. A subsection on inferring cognitive traits and strategies explains work related to building models of users. Finally, we explore the literature of active learning, a subset of machine learning that we use when managing the user's workload.

2.1 Machine Learning

Chapter 7 includes general background on machine learning, but here we focus on specific work that is closely related to the techniques presented in this dissertation. Many examples of interactive software using machine learning algorithms at their core use a subset of algorithms called *metric learning*. This powerful approach has been the subject of much research since 2003 [BHHSW05, BBM04b, DKJ+07, GRHS04, RF06, SKWH09, TL07, WBS06, WS08, XNJR02, YHC09, YL12] (see Yang's work [YJ06] for a survey) and has proven applicable in many real-world application domains including information retrieval, face verification and image recognition [CHL05, GVS09, HLLM06]. Methods in the ma-

chine learning literature assume that the machine learner is given additional information beyond the data itself, most often as pairwise constraints between data points, i.e. that certain pairs should or should not be close together. With that information, metric learning techniques learn a distance function optimized to produce relatively small distances between points that belong close together, and large distances between those that belong far apart [YL12]. Further, it is assumed that a domain expert can easily provide pairs of similar data points and pairs of dissimilar data points. An approximation to this information can be collected from the label information in supervised training datasets (by defining instances in the same class to be similar, and from distinct classes to be dissimilar).

Using this side information, existing methods seek to learn a distance metric such that the distance between similar examples should be relatively smaller than that between dissimilar examples. Although the distance metric can be a general function, the most prevalent one is the Mahalanobis metric defined by

$$D_A(x_i, x_j) = \sqrt{(x_i - x_j)^T A (x_i - x_j)}$$

where A is a positive semi-definite matrix and x_i and x_j are two instances in the data [YJ06]. While existing methods have been proven to be effective, what they fail to adequately address is how such side information is obtained in the absence of class label information. Indeed, the majority of methods found in the machine learning literature are not truly interactive, but instead simulate user interaction. In contrast, this dissertation is concerned with learning methods that can be used interactively. Chapter 3 includes a prototype with an interactive visualization method for observing which instances are considered similar based on the current distance metric, and a way to directly manipulate the visualization to redefine similarity.

2.2 Analytic Provenance

Analytic provenance in the visual analytics community broadly includes consideration for the history of how an analyst progressed through the various stages of his or her analytic

process, and encompasses the belief that the process is just as important as the product [NCE+11]. Through analyzing a user's interactions, researchers in analytic provenance seek to identify *how* a user discovers insight and how the same procedures can be stored and reapplied to automatically solve other similar problems [XGH06, KCD+09]. More complete descriptions and examples of analytic provenance can be found in surveys, including [FKSS08, dCCM09].

Many systems have been developed in the visual analytics community for logging, storing, and analyzing a user's interactions and activities. For example, the GlassBox system by Cowley et al. [CNS05] records low-level events generated by the interface (such as copy, paste, window activation, etc.). Sequences of visualization states can also be captured and visualized. For example, Graphical Histories captures and visualizes every new visualization produced by users over the course of the exploration [HMSA08]. Similarly, these sequences of visualizations or intermediate visualizations can be shown graphically to show branches and sequences during the exploration [SBE+11, DHRL+12]. At a process level, VisTrails captures the user's steps in a scientific workflow [BCC+05, CFS+06] and visualizes this history as graph of nodes depicting stages. Finally, at a model level, researchers, including in work included in Chapter 3 of this dissertation, have demonstrated that user interactions with representations of data can be used perform model-steering operations by inferring parameters of analytical models that can be exported and presented visually [BLBC12, EFN12b, GNRM08b, XGH06]. In Chapter 5 we discuss a method for visualizing the progress of a group of users through the space of possible analytical models.

The expanse of this body of work shows the value and extent of information encoded in interactions. Analyst strategies can be extracted from interaction logs, though it requires extensive manual examination [DJS+09]. Automatic analysis of interaction data can be used for authentication purposes [LB99, PB04]. Recent work included in this dissertation shows that higher-level information about users, like their performance and some aspects of their characteristics can be automatically predicted from this low-level interaction data, even hinting at human-readable models of their command sequences [BOZ+14] (see Chapter 6).

2.3 Visual Analytics and Machine Learning

There are many ways to take advantage of machine learning for interactive systems. Researchers in interactive machine learning strive to use human interaction with machine learning to improve machine learning results and improve user experiences, leveraging computers' raw analytical power and humans' reasoning skills to achieve results greater than either alone [SRL+09]. Systems have been built for grading [BJV], network alarm triage [ALK+11], building social network groups [AFW12], ranking search results with user context [ART06], managing overeating [CCR+13], and searching for images [AFKT11].

The visual analytics community focuses on applications to data analysis. One example is that techniques often considered in the realm of machine learning have been used to project high-dimensional data into 2D information visualization for data exploration. Jeong, et al. [JZF+09a] created a tool with a coordinated view between a projection of the data using principal component analysis (PCA) and parallel coordinates. The user can change the parameters of the projection interactively to explore the data. Similarly, Buja, et al. [BSL+04] created a tool with which a user can look at the data in a multi-dimensional scaling (MDS) projection and manipulate parameters directly to change the visualization. Dust and Magnets [YMSJ05] and RadViz [HGP99] layout high-dimensional points in a 2D visualization where the dimensions are anchored and their positions can be manipulated by the user to affect the display. These efforts demonstrate the effectiveness of combining interactive visualization with machine learning techniques. However, in these systems, the user's interaction is limited to modifying the parameters of the projection algorithm.

Several methods have been proposed that couple machine learning techniques with visualization to cluster or classify data. Nam, et al. [NHM+07] introduced ClusterSculptor, which allows the user to iteratively and interactively apply different clustering criteria to different parts of a dataset. Basu, et al. [BDL10], used metric learning to assist users in sorting items into clusters. Garg, et al. [GNRM08a] use Inductive Logic Programming to learn rules based on user inputs. These rules can be stored and reused in other parts of the data to identify repeating trends and patterns. Andrienko, et al. [AAR+09] allow expert users to build classifiers of trajectories from sampled data, and interactively modify

the parameters of the classifier at different stages in the analysis. Broekens, et al. [BCK06] propose a system that allows a user to explore data in an MDS projection by dragging points around to affect clustering and layout. DesJardins, et al. [DMF07] visualize data via a spring layout in which the user can interact with the visualization by pinning points in place. The pinned points are interpreted as constraints, and the constraints are used in a clustering analysis that results in a regenerated visualization that attempts to satisfy the constraints. Similarly, Endert, et al. [EFN12a] developed a spring-based system specific to text analysis, and developed a variety of interaction paradigms for affecting the layout. Choo, et al. [CLKP10] presented iVisClassifier, which is a system based on supervised linear discriminant analysis that allows the user to iteratively label data and recompute clusters and projections. In all these systems, the user works closely with an automated machine learning algorithm through a visual interface to explore and better understand the data, but none of these systems explicitly addresses learning a distance function as in Chapter 3.

There have been some methods designed specifically to learn a distance function and select features. The interactive tool proposed by Okabe and Yamada [OY11] learns a distance function by allowing a user to interact with a 2D projection of the data. However this tool is restricted to clustering, and supports only pairwise constraints that are formed by requiring users to select pairs of points and specify whether or not they are in the same cluster. Thus the user is forced to make these decisions purely based on the 2D projection. In contrast, our method as described in Chapter 3 provides several coordinated views of the data and does not restrict the user to formulate only pairwise constraints. May, et al. [MBD$^+$11] presented the SmartStripes system which assists the user in feature subset selection by visualizing the dependencies and interdependencies between different features and entity subsets. This work is similar to ours in that both methods seek to identify relevant dimensions in a high-dimensional dataset. However, unlike the SmartStripes system that directly represents low-level statistical information of each feature, our approach hides the complex mathematical relationships in the features and allows the user to interact directly with the visual interface.

Perhaps most conceptually similar to the work of Chapter 3 is that by Endert, et al. [EHM$^+$11a], which presents variations of three projection techniques, including MDS,

that can be updated based on user interaction. While their techniques are similar to ours, our approach emphasizes the externalization of a user's interactions to produce a useful, *exportable* distance function. Unlike the formulations of [EHM+11a], our system produces distance functions that are simple enough that the user can observe the relative importance of features while interacting with the software.

2.4 Visual Text Analytics

An important data type in analytics, one that we use as an example application in Chapter 5, is text. Visualizing text corpora entails showing relationships between documents and terms so that users can see topics, themes, trends, and other characteristics. For example, Stasko et al. developed Jigsaw which provides users with multiple linked views of document characteristics and relationships [SGL08]. Similarly, IN-SPIRE is a visual analytics system the shows visualizations of documents in many views, including the Galaxy View, where documents are shown spatially in a scatterplot [WTP+95]. Such visual metaphors encode relative similarity between documents visually as distance. As a result topics and themes in the documents become apparent as clusters of groups in the spatialization. Andrews et al. have shown that in fact being able to organize documents spatially helps analysts with sensemaking [AEN10].

The keywords frequently used in a document corpus can be shown using a visualization technique called word clouds [VWF09]. Word clouds consist of a spatial metaphor containing a collection of words from the document corpus. These words are often selected based on specific characteristics extracted from a dataset, including occurrence counts. Common visual encodings used include occurrence counts encoded using font size. Similarly, color can be used to encode word or term types or topics [SKK+08]. The layout algorithms for word clouds include space-filling algorithms that attempt to minimize whitespace [KL07], computing the grammatical similarity between terms [WZG+14], and determining the similarity score between terms to produce context-preserving word clouds [CWL+10]. More recently, techniques have extended the term-based word clouds to include semantic zoom functionality that enables users to retrieve detail (i.e., documents) in context of the

words being visualized [EBC+13].

Prior work exists that visualizes the context around words in text corpora. For example, Word Trees are an effective technique for representing the context of how terms are used given the adjacency of other terms [WV08]. Similarly, FeatureLens enables visual exploration of patterns in the text that stem from the saliency computed algorithmically [DZG+07]. Endert et al. presented ForceSPIRE, which enables users to directly reposition and regroup subsets of documents [EFN12c]. ForceSPIRE learns from these interactions, and adjusts the spatial layout of the remaining documents accordingly.

The work presented in this dissertation (Chapter 5) presents a technique that enables interactivity and visual exploration at the keyword-centric level. The visualization shows keywords spatially, encoding relative distance as the similarity between the keywords. Further, users are able to interact directly with these keywords to train and steer the underlying computation based on interactive model learning approaches.

2.5 Interactive Model Learning for Visual Analytics

In recent years, the visual analytics community has developed numerous tools and techniques that allow the users to interactive explore and create machine learning models through a visual interface. These interactions can utilize direct manipulation in which the user would directly modify and adjust the parameters of the model while examining the visual output, or the models can be implicitly learned based on the user's actions with different aspects of the visualization.

For directly manipulating model parameters, the iPCA system by Jeong et al. [JZF+09b] projects high-dimensional data points into 2D using principal component analysis (PCA) and allows the user to directly modify the weights of original data dimensions. Using a similar concept, Turkay et al. presented a dual-space approach for exploring and analyzing data in high-dimensional space [TFH11]. More recently, van den Elzen and van Wijk developed the BaobabView system that allows the user to semi-automatically refine a decision tree [vdEvW11], Muhlbacher et al. proposed a tool for generating regression models using a partition-based approach [MP13], and Gleicher presented the Explainers system that al-

lows the user to interactively control the outcomes of a support vector machine classifier [Gle13]. While these projects and systems focus on different types of machine learning methods, they share the common principle of model learning by allowing users to interactively manipulate parameters in a visual interface.

More related to the approach championed by this dissertation are the visual analytics systems that implicitly learn models by passively observing the user's interactions with data. In particular, the ForceSPIRE system by Endert et al. [EFN12c] allows the user to position documents in a spatial layout where the distance between two documents represent the similarity between them. Based on the user's positioning of documents, the ForceSPIRE system would update the underlying spring-based model and dynamically update the visualization. Using a similar spatialization approach, we presented the Dis-Function system [BLBC12], discussed in Chapter 3, that allows a user to manipulate high-dimensional data projected into 2D (using multidimensional scaling). The system would then learn the weights of a distance function by observing the relative distances between these data points in 2D. In Chapter 5 the proposed keyword-centric approach is inspired by these earlier works. However, our approach differs in that instead of directly visualizing the documents (as in the ForceSPIRE system) or the data points (as in the Dis-Function system), we choose to visualize the keywords (features) within the data. In Chapter 5, we discuss the advantages that this approach has over the previous methods, including providing a more scalable framework for large-scale data analysis.

2.6 Inferring Cognitive Traits and Strategies

Understanding the user is critical to creating the systems of the future that automatically customize the user experience for maximum comfort and efficiency. Much of the existing work in the visual analytics community on connecting the ways users solve problems with their cognitive abilities has been based on eye tracker data [AABW12, LME10, SCC13]. For example, Lu et al. demonstrated how eye gaze data can be used to determine important or interesting areas of renderings and automatically select parameters to improve the usability of a visualization system [LME10]. Steichen et al. explored the use of eye track-

ing data to predict visualization and task type [TCSC13, SCC13]. With varying degrees of accuracy they were able to predict: (1) a user's cognitive traits: personality, perceptual speed and visual working memory, (2) the difficulty of the task, and (3) the visualization type. These findings are particularly important for visual analytics tasks as previous research has shown that users' cognitive traits can be used as predictors of speed and accuracy [GF10, ZOC+13]. Although researchers have demonstrated the utility of eye gaze data, its collection is often not suitable for dynamic or interactive systems where what the user sees is not static. In our work [BOZ+14], described in Chapter 6, we forgo specialized sensors and analyze mouse interactions[1].

Cognitive traits can also be correlated with proficiency in certain domains. For instance, Ziemkiewicz et al. [ZOC+13], Green and Fisher [GF10], and Ottley et al. [OCZC13] demonstrate a significant correlation between the personality trait locus of control (a measure of perceived control over external events) and speed and accuracy on complex visualization tasks. Though more subtle, they also found significant effects with the personality traits extraversion and neuroticism. Other cognitive traits such as perceptual speed [All00, CM08] and spatial ability [Che00, ZK09] have also been shown to affect performance on visual analytics task.

Other types of traits can be used to adapt systems as well. In the HCI community, Gajos et al. developed the SUPPLE system that can learn the type and degree of a user's disability by analyzing mouse interaction data and generate dynamic and personalized interfaces for each specific user [GW04]. Although the intended scenario is in the domain of accessibility, the approach and methods developed by Gajos et al. can be generalized to other interfaces as well.

In the web usage mining community, researchers have used click stream data for modeling and predicting users' web surfing patterns [EV03, KJ04, KB00, SCDT00]. Some of the techniques developed for these web mining applications could be adapted to extend work like ours. However, we focus on a more general but complex visual task, and on learning about the users themselves as they complete the task.

[1] Recent work suggests that mouse movements in some interfaces are strongly correlated with eye movements [Coo06, HWB12]

2.7 Active Learning

Active learning is a form of semi-supervised machine learning in which the learner iteratively queries the user for additional information while building its model. The key idea behind active learning is that an algorithm can achieve greater accuracy or performance with fewer training labels if it is allowed to choose the most helpful labels [Set10].

A common approach is to select the data points that are most uncertain to classify. Different measures of the uncertainty are based on the disagreement in the class labels predicted by an ensemble of classification models [AM98, MM04, SOS92], by distance to the decision boundary [CCS00, RM01, TK01a], by the uncertainty of an unlabeled example's projection using the Fisher information matrix [Mac92, ZO00], or with Bayesian analysis that takes into account the model distribution [FSST97, JS04, TK01b, ZXC03].

Active learning and metric learning come together in several recent works, where authors determine what the user should see based on uncertainty of labels and coverage of the dataset [EFS12], or the median points in groups with the same label [WSLA09]. Yang and Jin select pairs of points for feedback based on the uncertainty of deciding their closeness [YJ07].

In a sub-category of active learning algorithms called active clustering, the end goal is a clustering instead of classification, and the common approach is to gather constraints by iteratively querying the user about pairs of points. Points are chosen by uncertainty [HB97], or by most informative example pairs [BBM04a]. One work by Xu et al. is especially related to that of Chapter 4. The authors learn a two-class spectral clustering with active learning by examining the eigenvectors of the pairwise distance matrix to find points on the boundary of being put in either cluster [XdW05].

Generally, active learning methods are based on querying the user for one unit of feedback at a time. In our approach in Chapter 4 the user plays an active role in deciding what feedback to provide: no suggestions are given without an initial seed point of interest from the user, and then, several suggestions are provided for the user to peruse.

Chapter 3

Learning Distance Functions from Visual User Interactions

This chapter is based on the paper:

Eli T. Brown, Jingjing Liu, Carla E. Brodley, and Remco Chang. Dis-Function: Learning Distance Functions Interactively. In *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on*, pages 83-92. IEEE, 2012.

3.1 Introduction

As discussed in the introduction to this dissertation, a method for bridging the space between data domain experts and the analysis tools they use is needed to allow those dependent on data to make full use of the combined power of their own abilities and those of computers. In this chapter we introduce an approach and prototype implementation, which we name Dis-Function, that allows experts to leverage their knowledge about data to define a distance metric. Using our system, an expert interacts directly with a visual representation of the data to define an appropriate distance function, thus avoiding direct manipulation of obtuse model parameters. The system first presents the user with a scatterplot of a projection of the data using an initial distance function. During each subsequent iteration, the expert finds points that are not positioned in accordance with his or her understanding of the data, and moves them interactively. Dis-Function learns a new distance function which

incorporates the new interaction and the previous interactions, and then redisplays the data using the updated distance function.

In the remainder of this chapter we present the proposed approach which allows a user to implicitly describe a distance function over high-dimensional data by interacting with a visualization of the data. We present the results of experiments on a machine learning benchmark dataset with our prototype system to assess Dis-Function's ability to learn a distance metric for classification. In addition, we evaluate the system's ability to provide interactive or near-interactive speed and conclude that performance scales linearly in the number of dimensions and quadratically in the number of data points. We finish with a discussion of the potential of Dis-Function and future directions of research.

3.2 Learning a Distance Function Interactively

Our approach to learning a distance function is both interactive and iterative. The user follows the procedure below until satisfied with the visualization, and thus with the *learned* underlying distance function.

1. Based on the current distance metric, we provide a two-dimensional scatterplot visualization of the data as well as other coordinated views (see Section 3.3).

2. The expert user observes and explores the provided visualizations and finds inconsistencies between the visualizations and his or her knowledge of the data. The user interacts with the scatterplot visualization via drag/drop and selection operations on data points with the mouse.

3. Dis-Function calculates a new distance function based on the feedback from the previous step. The new distance function is used to re-start the process at Step 1.

Figure 3.1 illustrates the process of iterating these three steps starting with the data as input, then making updates to the distance function until the user is satisfied with the 2D projection. In this section we describe our approach to each of these steps. We leave the details of the visualizations to the following section.

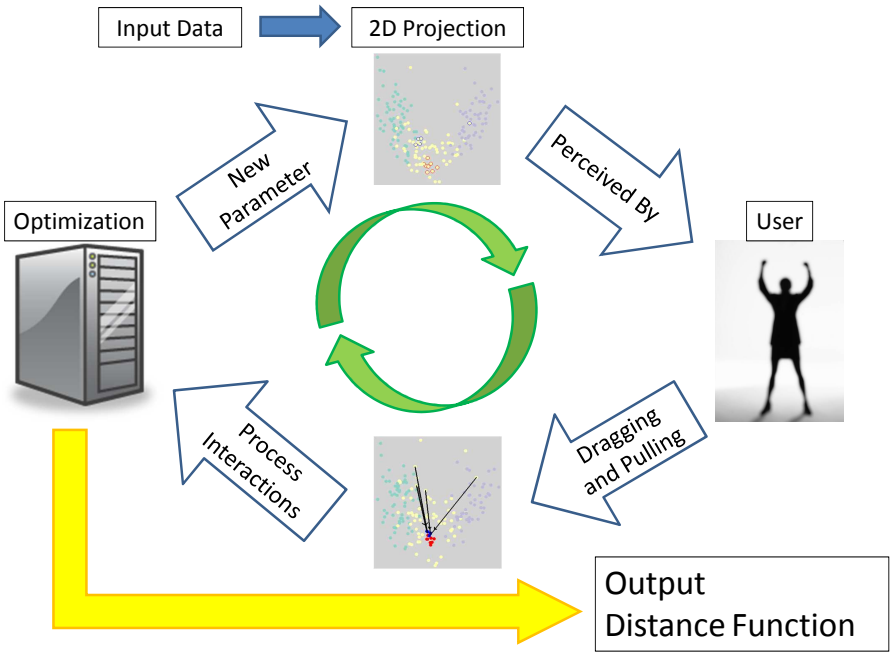


Figure 3.1: Flow chart showing the interactive process of using Dis-Function.

3.2.1 Producing a 2-D Scatterplot of the Data

To produce the two-dimensional scatterplot, we project the original (potentially high-dimensional) data to two dimensions via Multi-Dimensional Scaling (MDS) [BG05]. MDS has the property that when mapping from a high- to low-dimensional space, it preserves the relative distances between points. Thus, when a user looks to see if two points are the correct distance apart relative to others in the 2D projection, the relative distances between pairs of points observed in the projection correspond to their relative distance in the full-dimensional space as calculated with the current distance metric. The MDS projection is dependent on a distance function. The input is an $N \times N$ matrix D where each i, j entry contains the distance between points x_i and x_j from the set of all N data points in $\mathbb{R}^M$. (All notation used in this section is shown in Table 3.1).

Specifically, the projection algorithm accepts a pairwise distance matrix D covering all points, calculated with the "current" distance function. Note that in our experiments we set the initial distance function to be a uniformly-weighted Euclidean distance function

across all possible features. Given the matrix D, we compute an eigenvector decomposition in order to get the *principal components*, a ranked set of orthogonal vectors.[1] The top-ranked vector is the direction of highest variance in the distance matrix, and the second-ranked is the orthogonal vector that describes the next-most amount of variance. The data points, represented as vectors, are projected onto those two top-ranking principal components [Jol86]. The final result is a set of N vectors in $\mathbb{R}^2$, one for each original data point. Using these new vectors we display the data as a scatterplot visualization.

3.2.2 User Input

In Section 3.3 we describe the expert's interaction with the data in more detail after we have presented the details of the visualization system. For now, we ask the reader to assume that the user is able to directly manipulate the scatterplot to define sets of points that should be nearer to one another or further apart. To this end, let us define two sets of data points Y_1 and Y_2, selected by the user, as sets of points which should be moved relative to each other. We then calculate a matrix U that will represent the user input when calculating an updated distance function as described in Section 3.2.3, where U is defined as follows:

$$U_{ij} = \begin{cases} \frac{intended\_distance}{original\_projected\_distance} & \text{if } (x_i, x_j) \in Y_1 \times Y_2, \\ 1 & \text{otherwise.} \end{cases} \tag{3.1}$$

where $original\_projected\_distance$ is computed as the Euclidean distance between points x_i and x_j in the scatterplot before the user moved them, and $intended\_distance$ is their Euclidean distance in the scatterplot after. Thus dragging data points x_i and x_j *closer* results in $U_{ij} < 1$, whereas dragging them *further* apart would result in $U_{ij} > 1$. These values will be used to compute a new distance function as described in the next section. Note that the majority of the values for $U_{i,j}$ will be equal to 1 because the interaction paradigm is that the user wants to change the relative distances between points in Y_1 and Y_2 *only*, wishing to maintain the relative distances of all other data points.

[1] That is, we calculate an MDS projection by looking at the principle components, as in principle component analysis (PCA), of the pairwise distance matrix [GKWZ07].

3.2.3 Updating the Distance Function

We incorporate user input to create a new distance function by solving an optimization problem over the space of possible distance functions. We use a weighted Euclidean distance function, i.e., Euclidean distance with each dimension of the data weighted by a coefficient. Although there are many other possibilities, we chose weighted Euclidean distance because it is easy for a human to map the magnitude of the weight of each feature to its relative importance. We describe in Section 3.3 how we present a visualization of the weights of the distance function to further help the user understand the data.

The distance between two data points $x_i, x_j \in \mathbb{R}^M$ is given by:

$$D(x_i, x_j|\Theta) = \sum_{k=1}^{M} \theta_k (x_{ik} - x_{jk})^2 \tag{3.2}$$

where M is the number of original dimensions in the data, Θ is the vector of feature weights, and θ_k is the weight for feature k. We initialize with all weights equal, i.e., $\theta_k = 1/M$.

To update Θ after a user interaction at time t, we seek to find the Θ^t that maintains the relative distances of points the user did not select while encouraging changes that affect the selected points in the desired direction. We formalize this intuition with the following optimization criterion:

$$\Theta^t = \arg\min_{\Theta^t} \sum_{i<j\leq N} L_{ij}^t \left(D(x_i, x_j|\Theta^t) - U_{ij}^t \cdot D(x_i, x_j|\Theta^{t-1}) \right)^2 \tag{3.3}$$

where U_{ij}^t is defined in Equation 3.1 and is the result of the user's interactions at round t based on the projection using the distance function defined by Θ^{t-1}. The term L_{ij}^t, defined in Equation 3.4, is a scalar weight that is greater than one when the points x_i and x_j are in Y_1^t and Y_2^t, and one otherwise. In the summation over all points in the objective function of Equation 3.3, this increases the value of terms corresponding to points the user moved. We define L_{ij} at time t as:

$$L_{ij}^t = \begin{cases} \frac{N(N-1)}{|Y_1^t||Y_2^t|} - 1 & \text{if } (x_i, x_j) \in Y_1^t \times Y_2^t, \\ 1 & \text{otherwise.} \end{cases} \tag{3.4}$$

Definitions used in describing our methods	
N, M	Number of points, number of dimensions
$x_i \in \mathbb{R}^M$	Point i of the data
x_{ik}	Value of feature k of data point x_i
Θ	Vector in $\mathbb{R}^M$ containing the weight of each dimension for a distance function
θ_k	Weight of feature k in Θ
Θ^t and Θ^{t-1}	Indicate Θ values from before $(t-1)$ and after an optimization step
$D(x_i, x_j \mid \Theta)$	Distance between x_i and x_j given parameters (dimension weight vector) Θ
δ_{ijk}	Abbreviation used in the gradient of the objective function as a stand-in for $(x_{ik} - x_{jk})^2$
$\mathbb{O}_{ijt}$	Abbreviation used in the gradient of the objective function for the square root of a term of the full objective function
L_{ij}	The impact coefficient in the objective function
U_{ij}	Entry in matrix U containing the user feedback information for the pair (x_i, x_j)

Table 3.1: Definitions of the symbols described in our methods.

where Y_1^t and Y_2^t are the sets of points in each user interaction set at iteration t. The value of the coefficient is the ratio of the number of unchanged pairs of points to the number of changed pairs. This heuristic and somewhat ad hoc weight is to ensure that the points the user selected have impact in the overall value of the objective function, even though the function is a sum over all points in the dataset, and Y_1^t and Y_2^t could be relatively small.

Our objective is to incorporate new user feedback at iteration t, while preserving the user's previous interactions. Previous iterations of feedback are not explicity represented. Instead, Equation 3.3 minimizes the difference, over all pairs of points, between the new distance and a multiple (U_{ij} from the user input) of the old distance. By including the old distance in the function and summing over all points, we provide some inertia against the user's updates. This was an important design decision, as machine learning methods for finding distance functions generally focus on a single set of constraints from the user and optimize once (with the exception of [BHHSW05], which has an online version of the RCA algorithm).

To find a solution to this optimization problem we use nonlinear conjugate gradient descent [HZ06], which is an iterative algorithm. Starting from an initial guess, the solver moves in steps toward a minimum of the objective function by walking along the gradient. At each step, the gradient is evaluated at the current guess, and a new guess is generated by

moving in the direction of the gradient some small amount. This process continues until it converges. Although versions of the algorithm exist that determine step directions without the gradient, we provided the following gradient function to the solver for efficiency:

$$\nabla objective(\Theta) = \begin{bmatrix} \frac{\partial \Theta}{\partial \Theta_1} \\ \vdots \\ \frac{\partial \Theta}{\partial \Theta_M} \end{bmatrix} = \begin{bmatrix} 2\sum_{i<j\leq N} \delta_{ij1}\mathbb{O}_{ijt} \\ \vdots \\ 2\sum_{i<j\leq N} \delta_{ijM}\mathbb{O}_{ijt} \end{bmatrix} \tag{3.5}$$

where

$$\delta_{ijk} = (x_{ik} - x_{jk})^2$$

and

$$\mathbb{O}_{ijt} = L_{ij}\left(D(x_i, x_j|\Theta^t) - U_{ij}^t \cdot D(x_i, x_j|\Theta^{t-1})\right).$$

3.3 Visualization and User Interaction

Figure 3.2 shows Dis-Function, the prototype system introduced in this work. Dis-Function presents the user with three coordinated views of the data to aid in data exploration. Along the bottom of the window, seen in Figure 3.2E, user can see the raw data in a table with column labels. In Figure 3.2A, the interactive scatterplot visualization both displays data and captures user interaction. In 3.2C, a view we call *parallel bars* shows the user how the values of all the points in the dataset are distributed in each dimension. It appears as a bar graph with one bar for each dimension. The bars are each colored with a heat map to show how common each range of values along the bar is.

The three views are coordinated, which facilitates exploration [Rob07]: selecting a point on the scatterplot causes the point to be moved into view in the data table and highlighted, as well as highlighted on the parallel bars view. The point is highlighted by a black line across each bar at the height corresponding to that point's value in the bar's dimension. Placing the mouse over an element in the data table causes the point to be highlighted in the scatterplot and parallel bars.

Together, these views allow a user to explore the data in order to provide more

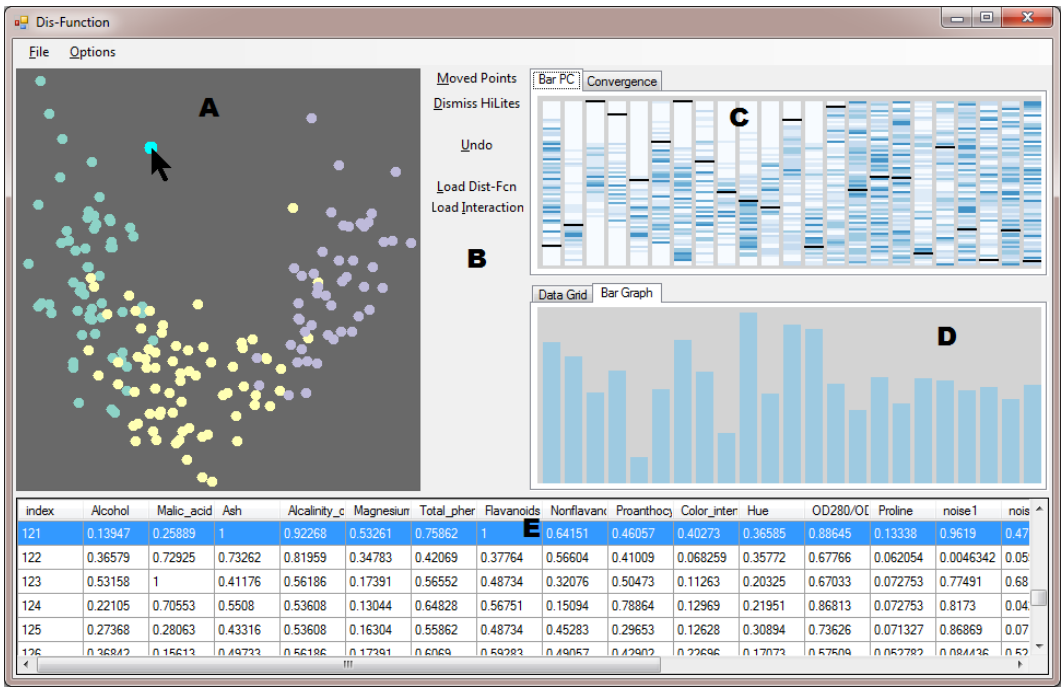


Figure 3.2: This screenshot shows Dis-Function comprising A) the MDS scatterplot visualization of the data; B) the buttons for recalculating the projection, undoing unsatisfying input, loading custom distance functions and user input data, etc.; C) the Parallel Bars visualization described in Section 3.3; D) a bar graph of the current distance function (obscured 'Data Grid' tab shows a tabular version); and E) the original data. All these views are tightly coordinated such that interactions with one view are immediately reflected on the others. For example, in the figure above, the mouse cursor is over a point in the scatterplot, and thus the corresponding point in the data table at the bottom is highlighted and the black lines on (C) highlight the values of the data point in each dimension as they relate to other data points.

useful feedback to Dis-Function. Aside from just the relative distances among the points as shown in the scatterplot of the projection, the user can see the actual data in the original data space. Assuming the user has some domain knowledge, he or she will likely understand the implications of certain ranges of values in certain dimensions. The user can also observe from the parallel bars visualization how any data point fits into the scheme of the data on a dimensional basis. If a given point is an outlier in one or all dimensions, for example, that will be clear from the parallel bars visualization.

In addition to the views of the data, we provide two views of the distance function and the user's progress toward finding it. Figure 3.2D shows two tabs. The one visible in the figure shows a bar graph representation of the current distance function. Each bar

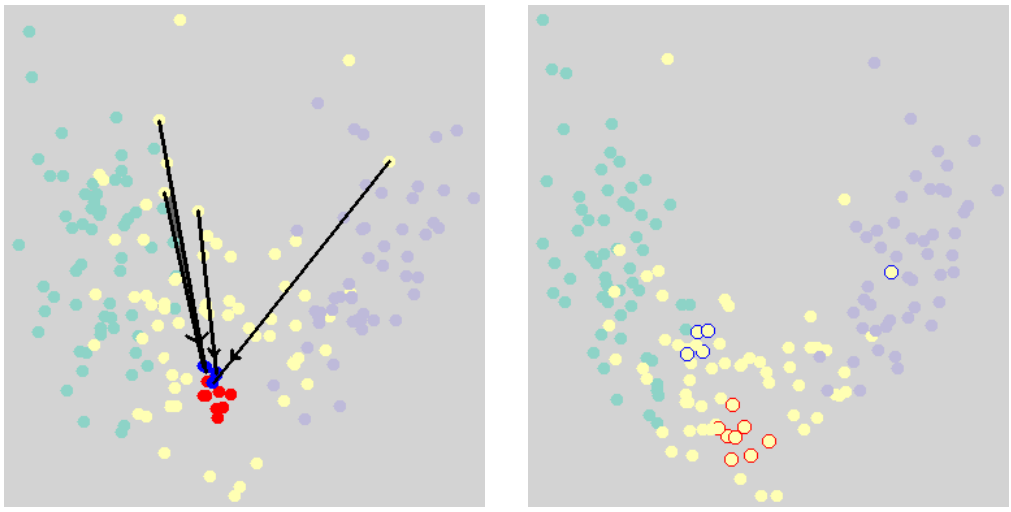


Figure 3.3: These images show an example of how a user manipulates the visualization. A handful of points have been marked in blue and dragged closer to another set of points, marked in red. After the update (on the right), the points in those groups are closer together, and the clustering with respect to different colors is more compact. The same red and blue points marked on the left are indicated in their new positions on the right with red and blue halos.

represents a dimension, and the bar height encodes the weight of that dimension. Using the bar graph, the user can watch the distance function change after each feedback iteration. This allows the user to observe the relative importance of the different dimensions in the current distance function used to display the data in the scatterplot to the left. The hidden tab in Figure 3.2D contains a data table version of the same information, but includes history, and makes it easy to export the distance function from any iteration.

Having described how the data is visualized we now turn to how the user can interact with the data through this interface. Recall that the goal of the interaction is to define two sets of points that should be closer to one another, or further apart. To this end, the user can select points and drag-and-drop points to mark them as members of either set and to move them some amount closer together or further apart. The points in the two sets are marked by different colors in the scatterplot visualization, and they correspond to using the left or right mouse button when clicking or dragging points. These two sets of points, which we indicate by red and blue in the visualization, correspond to the two sets, Y_1^t and Y_2^t respectively. During the feedback step of each iteration, the user can select and unselect points, and repeatedly move points around. To signal completing one round of interaction, the expert

clicks the *Moved Points* button (see 3.2B). At this point a new distance metric is learned from the feedback and the data is then reprojected using the new metric. Currently, the scatterplot and bar graph update as quickly as possible, without animation or optimizing for rotation. To provide context between iterations, after each iteration the user can see where the points in his or her input sets have been placed in the new projection via highlighting with colored rings (we illustrate this process in detail in the next section).

In the next section, we present empirical results of ten subjects interacting with Dis-Function and we provide preliminary experiments to assess its interactive speed. Our results show that our system is interactive or near-interactive for a standard machine leaning testing dataset.

3.4 Experiments and Results

In this section, we describe our evaluation of the effectiveness of Dis-Function at finding distance functions, the quality of distance functions learned by Dis-Function, and the time taken to perform each update as a function of the input. We begin with a presentation of the empirical results that demonstrate the efficacy of the proposed interaction method for defining sets of points to learn a distance metric.

3.4.1 Empirical Results

We had ten subjects from Tufts University (undergraduate and graduate students including six males and four females from Electrical Engineering, Visualization and Human-Computer Interaction) evaluate our software. In order to test software meant for experts in the absence of experts, we simulate the experience by coloring the data points in the scatter plot based on the known classes of the points; i.e., when the correct class membership of each point is visible, any user is an "expert" on the data.[2] We showed each participant how to use the software and let each perform as many iterations as desired. We performed our experiments on a modified version of the Wine dataset from the UCI Machine Learning repository [MFL88] as this has been used in prior studies of defining a distance metric. The

[2]Note that because the subjects interact based on class information, our experiments do not explicitly evaluate the efficacy of the coordinated visualizations.

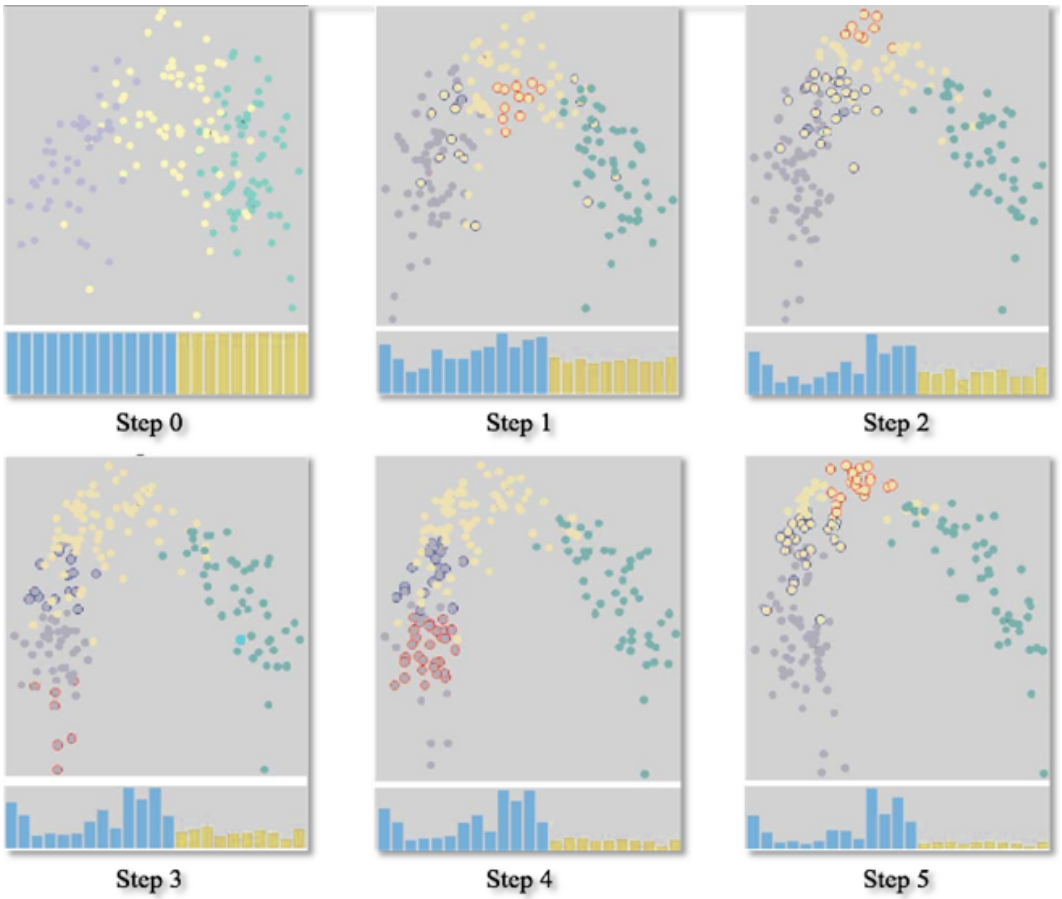


Figure 3.4: While Figure 3.3 demonstrates one step of feedback, this figure shows how the scatterplot visualization improves through a number of iterations of feedback (matching those of Figure 3.5). Each scatterplot shows the visualization after a round of feedback. The bar graph below each plot shows the distance function used to create the projection shown above it. Each bar represents a different dimension, and collectively they show the relative weights of the dimensions in the distance function. In each frame, the sets Y_1 and Y_2 from the previous interaction are highlighted with red and blue halos.

original Wine dataset has thirteen features describing chemical components of wine, and 178 instances, each representing a specific wine, and each labeled as one of three classes. We modified the Wine dataset as follows: we added ten noise features, each of which we generated by randomly choosing values from a uniform distribution over the range [0,1], matching the range of the data itself, which is normalized. We introduced these features in order to know exactly which features in the data were uninformative. We hypothesized that the user's interactions would result in a distance function giving these "useless" features a weight close to zero.

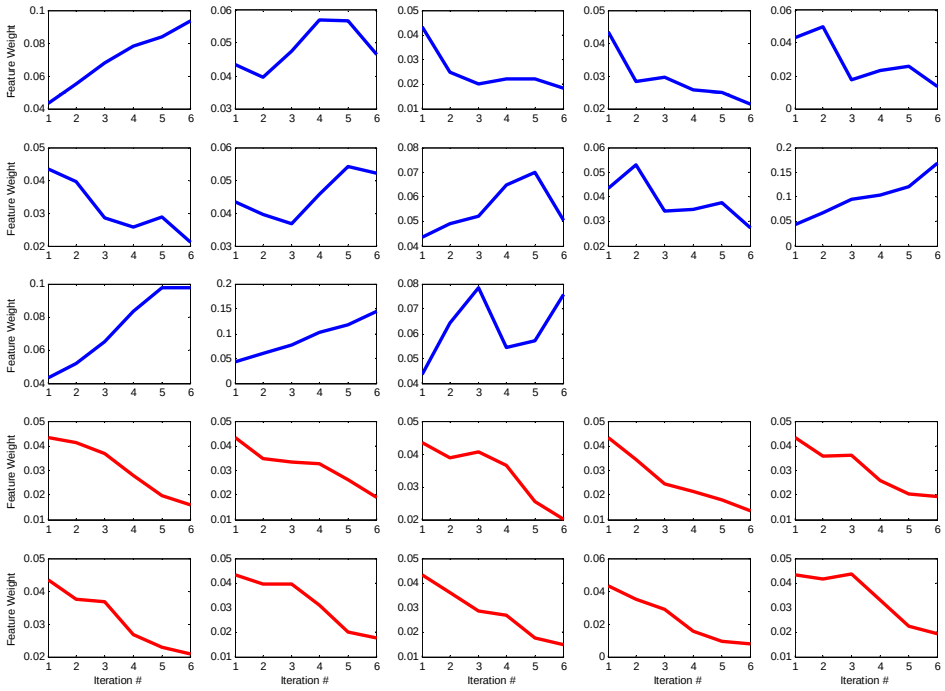


Figure 3.5: This figure shows the weight of each feature after each of User 10's five interactions. Each sub-graph shows a single feature. The x-axis gives the iteration number and the y-axis, the weight. The top thirteen correspond to the features in the original wine data and the bottom ten show the weights for the added noise features. Note that the weights of the added noise features quickly decrease and approach zero within a few iterations.

Because our users were given instant expertise in the form of data colored with class labels, we instructed them to provide feedback by moving points closer together that are in the same class (i.e., of the same color). In our experiments, we observed that all users quickly figured out that moving only a few points at a time did not result in significant changes to the distance function and further that moving points from class x that are far away from a class x cluster[3] to its center allows the system to converge more quickly. An example of a typical interaction is shown in Figure 3.3. The left side shows the original positions of data points with arrows indicating the user interaction; the user dragged the points from the start to the end of the arrow. The red and blue circles show the two sets of selected points. The right side shows the result of the reprojection of the data using the new

[3]Note that there may be more than one cluster per class.

distance function. The selected points have moved closer together and the clusters are more cohesive.

Our user study found all users were satisfied with the separation of different classes after 4–12 ($\mu = 7.3, \sigma = 2.5$) feedback updates. Figure 3.4 shows a sequence of updates by one of the participants where the augmented Wine dataset transitions from scattered to compact. Each step shown is after feedback (we do not show the user feedback step explicitly). The figure illustrates how the visualization changes with more user input. Note that the bar graph accompanying each scatterplot shows the weights of the dimensions in the distance function associated with the plot. Figure 3.5 shows the values of the dimension weights changing with each iteration for the same user as was used to generate Figure 3.4. Each sub-graph in Figure 3.5 shows the weight of a different dimension; the x-axis gives the iteration number and the y-axis shows the magnitude of the weight. Notice that the weights of the noisy features (the bottom ten) plunge steadily downward as was hypothesized; recall that these features were generated uniformly at random and thus provide no information about the classes in the data. In our experiment, all ten participants generated distance functions with low weights on these noisy features.

We evaluated the users' learned distance functions using a k-nearest-neighbor (k-NN) classifier. Recall that a k-NN classifier classifies a previously unseen (test) instance by taking the majority vote of the instance's k nearest neighbors, where "nearest" is calculated using a (weighted) Euclidean distance function. Thus we can evaluate the quality of the learned distance function using a leave-one-out cross-validation (LOOCV)[4] over the training data.

We show the results for $k = 1, 3, 5$ and 7 in Table 3.2. We note three observations from these results. First, all user-guided distance functions perform better than using the original unweighted Euclidean distance function. Second, performance is also a function of the user's ability as can be seen by the fact that users 2 and 3 performed worse than everyone else despite having the same directions. Finally, the Wine dataset is a relatively "easy" classification task in that our baseline accuracy is already 90%. We anticipate that

[4]In an LOOCV we hold out each instance one at a time, and use the rest of the data to form our k-NN classifier.

User	k-NN Accuracy 1	3	5	7
Even Weight	0.89	0.91	0.91	0.91
1	0.97	0.97	0.97	0.97
2	0.92	0.93	0.95	0.96
3	0.92	0.93	0.95	0.96
4	0.94	0.97	0.98	0.97
5	0.96	0.97	0.97	0.97
6	0.95	0.96	0.98	0.96
7	0.95	0.95	0.97	0.97
8	0.94	0.96	0.96	0.97
9	0.94	0.96	0.96	0.97
10	0.94	0.97	0.98	0.98

Table 3.2: Results of a leave-one-out cross-validation (LOOCV) for the Wine data using k-NN for $k = 1,3,5,7$. "Even Weight" is the baseline condition, i.e., an evenly-weighted Euclidean distance function without user interaction.

for "harder" classification tasks we will see even more of a performance increase after user interaction.

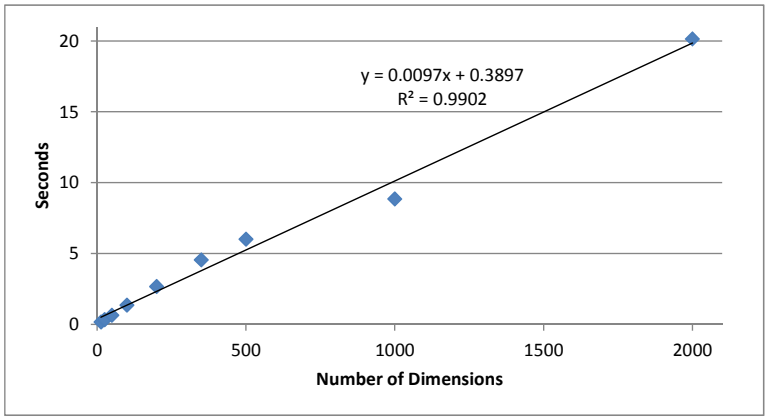
3.4.2 Interactive Speed Performance

Using the Wine dataset, we find that user interactions with the visualization are fluid, and that updates based on user feedback take on the order of a second. In this section, we describe additional experiments to evaluate the scalability of Dis-Function in a controlled manner. Specifically, we examine the performance of Dis-Function as the dataset grows in size (in terms of number of rows) and in complexity (in number of dimensions) independently. Our experiment was conducted on a desktop computer with an AMD Phenom X3 processor and eight gigabytes of memory, running Windows 7 Home Premium. Our implementation of Dis-Function is in C#, using Windows Forms. The rendering is done in software using GDI+ (without using GPU hardware support), the PCA computation is done using the *PricipalComponentAnalysis.Compute* function in the Accord.NET Framework library,[5] and conjugate gradient is done using the *mincgoptimize* function from the C# ALGLIB library version 3.5.0.[6] At the time of the experiment, no other applications were

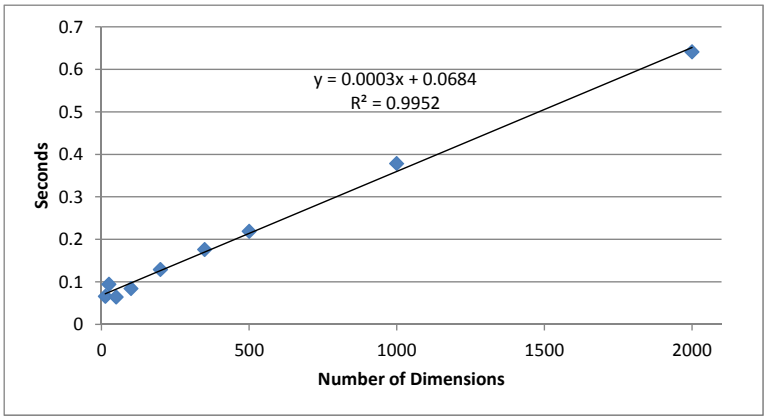
[5] http://code.google.com/p/accord/

[6] www.alglib.net

running on the computer except for basic Windows services running in the background. In the remainder of this discussion, the reported performance is based on the amount of time required for Dis-Function to perform the optimization and re-projection, independent of the interface.



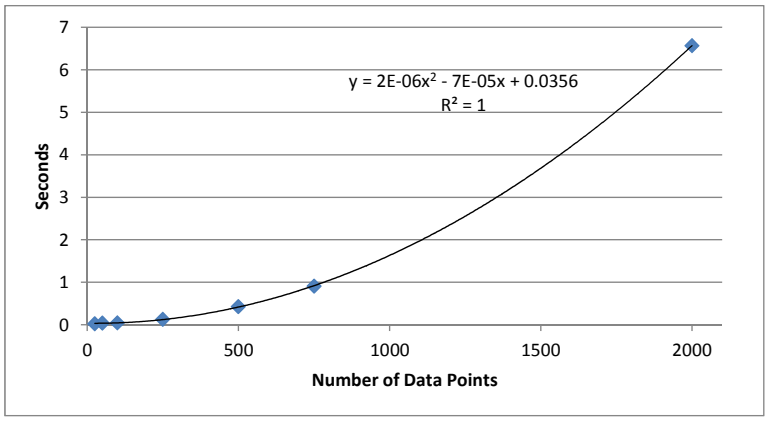
(a) Optimization



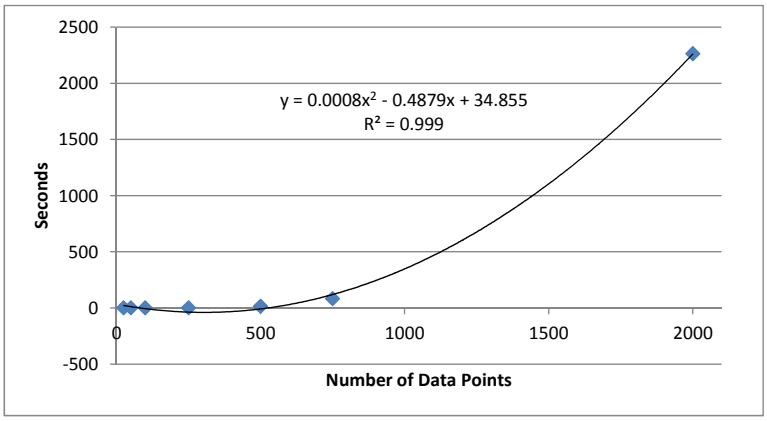
(b) Projection

Figure 3.6: Performance, as affected by *data complexity* (number of dimensions), of processing user feedback for one iteration by (a) running optimization to find a new distance function and (b) re-projecting data for the scatterplot. Notice that both operations scale linearly in data dimensionality.

In the Dis-Function prototype, we include a stand-alone command-line executable that links against Dis-Function. This program allows us to write scripts that test different types of input and collect performance data. To test the dependence on data dimensionality, we extended the Wine dataset, which has 178 data points and 13 dimensions, up to 2000 dimensions. Those extra dimensions were filled with random numbers drawn from a uniform distribution over the range $[0, 1]$, the same range as the original, normalized data. We



(a) Optimization



(b) Projection

Figure 3.7: Performance, as affected by *data size* (number of points), of processing user feedback for one iteration by (a) running optimization to find a new distance function and (b) re-projecting data for the scatterplot. Notice that both operations scale quadratically in data size.

ran our performance test repeatedly with all the data points, starting with only the real data dimensions (13), and cumulatively growing to the full 2000 dimensions. Figure 3.6 shows the results of this experiment: the dependence of the *optimization* time on the number of dimensions (Figure 3.6 (a)), and the dependence of the *re-projection* time on the number of dimensions (Figure 3.6 (b)).

To evaluate the performance in data size, we randomly generated a 2000-element dataset with two dimensions, and used sequential subsets of it to create datasets of different sizes. Figure 3.7 (a) shows the time taken by the *optimization* as a function of the number of data points, and Figure 3.7 (b) shows the time taken by the *re-projection* as a function of the number of data points.

Both optimization and projection scale the same way: linearly in the number of dimensions and quadratically in the number of data points. The graphs include trend lines fit by Microsoft Excel, and in all cases the correlation is high, as seen in the figures. These results are aligned with our expectations because the conjugate gradient method can be expected to converge in as many steps as there are dimensions. In terms of number of data points, the calculations are dependent on pairwise distances, which number $O(N^2)$.

Although the performance as it stands makes Dis-Function comfortable to use, we believe the performance of the re-projection step can be improved substantially by introducing online singular value decomposition (SVD) into our PCA calculation, similar to the approach of Jeong, et al. [JZF+09a]. Using *online* SVD would allow us to calculate the eigenvalues at each projection step incrementally. Other options for fast eigenvalue calculation include iterative algorithms [Hea01]. Separately, we could improve the performance of the optimization step by stopping it early: empirically we have noticed a good solution is reached in only a few steps. Truncating the number of steps the optimizer is allowed would sacrifice only a small amount of precision and speed up the software's response to user input. Finally, a refactoring of the objective function could reveal applicability of alternative, higher-performance optimization methods.

3.5 Discussion

In this section we discuss Dis-Function as a general purpose data analytics tool, propose future work, and provide some usage guidelines.

3.5.1 Broad and Flexible Use

What we have presented in Dis-Function is a prototype for a widely-applicable data analytics tool. The distance functions produced by Dis-Function provide a form of knowledge externalization that quantifies expert notions of a data space. By assigning numerical weights to each dimension indicating relative importance, the learned distance function can also serve the purpose of feature selection. A user may discard features with a relatively low weight, thereby reducing the dimensionality of a large and complex dataset in order to

make it easier for a user to explore and analyze.

Because a distance function is a representation of an expert's intention, if the expert has more than one intention, he or she can use Dis-Function to create multiple distance functions, each reflecting a different analysis hypothesis. For example, if a researcher wants to study subjects in two different contexts such as socioeconomic similarity and physiological similarity, he or she can run Dis-Function twice to produce two distance functions. The first time, the researcher moves points with similar socioeconomic background closer; the second time, the researcher drag points with similar physiological makeup together. Both resulting distance functions can be used in additional computational analysis, perhaps comparing how each clusters the data. (Recall that one can use the learned distance function with clustering algorithms such as *k*-means [Mac67] or EM [DLR77]).

3.5.2 Possible Extensions

Thinking of Dis-Function as a framework instead of just a prototype opens some exciting possibilities for capturing different types of expertise and exploring ways to express knowledge by interacting directly with a visualization. We have provided only one simple mechanism for capturing user input.

More techniques for incorporating user input will be tied to introducing different visualizations amenable to similar "semantic interactions" [EFN12a]. The goal is to find visualizations where we can discover a mapping between some manipulation of the view and a semantic meaning for the user, and where that meaning can be translated into mathematics for adjusting the generation of the visualization. Not only could we offer different types of projections, but we can learn distance functions for other types of data. For example, when representing hierarchical data using a phylogenetic tree, the framework of Dis-Function can be immediately applied because a phylogenetic tree is also generated from pairwise distance data.

We can experiment with completely different classes of visualization like parallel coordinates [ID90], RadViz [HGP99], and Dust and Magnets [YMSJ05], for which tools exist for exploring data by manipulating the parameters. Dis-Function could allow an expert to use those tools to discover similar data points, and then model that feedback to stretch

dimensions for improved visualization.

3.5.3 Usage Tips

Our own informal experimentation revealed some best-practice ways of interacting with Dis-Function. While the semantic meaning of separating dissimilar points is clear, the optimization we have used to learn a distance function is not designed for such feedback. As an example, consider moving two points together: they can only move in one direction: toward each other. On the other hand, when specifying that two points should be further apart, the two points can be moved in any direction. Indeed, when separating groups of points, Dis-Function occasionally introduces re-orientation of all data points in a way that is difficult to correlate to the previous layout. In some cases, this behavior is desirable – for example to separate tightly overlapping clusters. However, in most cases, it makes sense to perform the transformation "move set A further from set B" as two iterations of feedback by moving points closer: move A closer to points far from B, then B closer to points far from A. This way it is clearer in which direction to spread sets A and B.

3.6 Summary

In this chapter we presented a prototype implementation, named Dis-Function, that allows a user to interact with a visualization to define a custom distance function. In particular, through a set of coordinated views, the user can explore data and find points to drag closer together. Based on a series of these interactions, the system learns a weighted Euclidean distance function that can be used in any data analysis algorithm requiring the definition of a distance function. The weights are human-readable as importance ratings of each dimension, giving the user a way to understand what facets of the data are most relevant. We demonstrated the scalability of Dis-Function in both data size and complexity, and illustrated empirically by using a well-known dataset that an expert user could use Dis-Function to build a distance function that can be used to improve classification or clustering.

Chapter 4

EigenSense: Saving User Effort with Active Metric Learning

This chapter is based on the work-in-progress workshop paper:

Eli T. Brown and Remco Chang. EigenSense: Saving User Effort with Active Metric Learning. *20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining workshop on Interactive Data Exploration and Analytics (IDEA)*. 2014.

4.1 Introduction

Learning distance functions from user interactions with a visualization is a compelling mechanism for empowering users to create helpful data models without understanding complicated algorithms. However, in the prototype introduced in Chapter 3, we have left open the question of how exactly a user would choose with which points to interact. In order to provide feedback about what points should be grouped closer together, a user might have to compare large, even infeasible, numbers of points. User feedback is expensive, since human throughput at reviewing data is far lower than a computer's throughput at analysis. What is needed is a mechanism to use computing throughput to guide the user's investigation. Ideally, to the user this looks like making sure that his or her efforts are leveraged as well as possible. One possiblity is to program the machine learning back-end to help the user identify what data points are important.

The machine learning community of active learning researchers develops techniques to query users for feedback in ways that will most help the machine learner. A common approach is to query users about points chosen so that the user feedback will resolve uncertainty in the model. Emphasizing the interactive learning perspective, our approach keeps the user in control, providing suggestions only when the user initiates a direction of inquiry. We target our active learning method toward predicting the impact of any given user input. Using our method, a user can judiciously spend the effort of developing feedback on data that will affect the underlying model as much as possible.

In this chapter, which describes work in progress, we first introduce EigenScore, a measure that leverages "eigenvector sensitivity" to predict how much a potential user input will change an underlying metric learning model. We then propose EigenSense, which uses EigenScores to guide a user toward making the most productive feedback while minimizing his or her effort (in terms of data points examined). Finally we provide two types of evidence of the efficacy of this algorithm. First, we compare EigenScores to the ground-truth of what they estimate: the amount that particular constraints would change the metric learning model. Second, we show with simulations that the few points selected for user review by EigenSense often include the best possible choices as evaluated by an oracle.

4.2 Motivation: Eigenvector Sensitivity to Find Critical Points

The eigenvectors of a matrix have been used to represent its underlying structure for applications in many domains including connectivity graph analysis [PBMW99], face recognition [TP91], and clustering [Wei99]. The eigenvectors of symmetric matrices A for which entry A_{ij} represents some measure of distance between objects i and j is of particular relevance. For example, the PageRank [PBMW99] algorithm uses an $n \times n$ pairwise matrix to represent the transition probabilities between pairs of the n webpages. Here entry A_{ij} corresponds to the probability of landing on node (page) j during a length-one random walk, having started at node i. Raising that matrix to the power k gives a matrix of the probabilities of landing on node j having started a *length-k* random walk at node i. Increasing powers of A^k will show the asymptotic behavior of flow through the graph. Conveniently, following from the

definition and orthogonality of eigenvectors, the dominant eigenvector approximates this quantity. For a real, symmetric matrix A, suppose we have the eigenvalues $\lambda_1, \ldots, \lambda_n$ sorted in order of decreasing magnitude, and their corresponding eigenvectors $\mathbf{v_1}, \ldots, \mathbf{v_n}$. Because the eigenvectors are orthogonal, any vector $\mathbf{x} \in \mathbb{R}^n$ can be written as a linear combination of the eigenvectors, with coefficients α_i. We can observe the asymptotic behavior:

$$\begin{aligned} x &= \alpha_1 \mathbf{v_1} + \alpha_2 \mathbf{v_2} + \ldots + \alpha_n \mathbf{v_n} \\ Ax &= \alpha_1 A\mathbf{v_1} + \alpha_2 A\mathbf{v_2} + \ldots + \alpha_n A\mathbf{v_n} \\ A^k x &= \alpha_1 \lambda_1^k \mathbf{v_1} + \alpha_2 \lambda_2^k \mathbf{v_2} + \ldots + \alpha_n \lambda_n^k \mathbf{v_n} \\ &= \alpha_1 \lambda_1^k \left(\mathbf{v_1} + \frac{\alpha_2}{\alpha_1} \left(\frac{\lambda_2}{\lambda_1} \right)^k \mathbf{v_2} + \ldots + \frac{\alpha_n}{\alpha_1} \left(\frac{\lambda_n}{\lambda_1} \right)^k \mathbf{v_n} \right) \end{aligned}$$

Note that because the dominant eigenvalue, $\lambda_1 \geq \lambda_i, i = 2 \ldots n$ in the final sum, the $\mathbf{v_1}$ term dominates.

When studying population dynamics, biologists take advantage of this fact with a matrix L, called a "Leslie" matrix, where each element L_{ij} represents an organism's survival prospects to age i from age j [1]. To see the equilibrium point of a population, biologists study the dominant eigenvector of this matrix [EG11].

Extending this technique to see how the population can be affected by environmental factors, biologists adjust survival rates at different times in the lifecycle by editing the matrix, and reconsider the new dominant eigenvalue. This sensitivity of the eigenvalue to change in particular matrix entries is the eigenvalue sensitivity [EG11].

Motivated by biologists' successes with eigenvalue sensitivity in Leslie matrices, we consider the behavior of the dominant eigenvector of our related $n \times n$ pairwise distance matrices, and we adapt the concept of sensitivity to the context of active metric learning. We will use the dominant eigenvector of a pairwise distance matrix as a stand-in for its overall structure, and calculate the sensitivy of that eigenvector with respect to changes in entries of that matrix. Since each entry corresponds to a pair of data points, we will use this approach to estimate how individual user inputs, i.e. user constraints that certain pairs

[1] The matrix represents different age groups' rates of survival and reproduction by setting the first row to the fertility rate of each age group, and the lower off-diagonal entries to organism survival probabilities from one age group to the next.

of points have small distances between them, will affect the structure of the distance matrix and the underlying distance metric.

4.3 Application Context

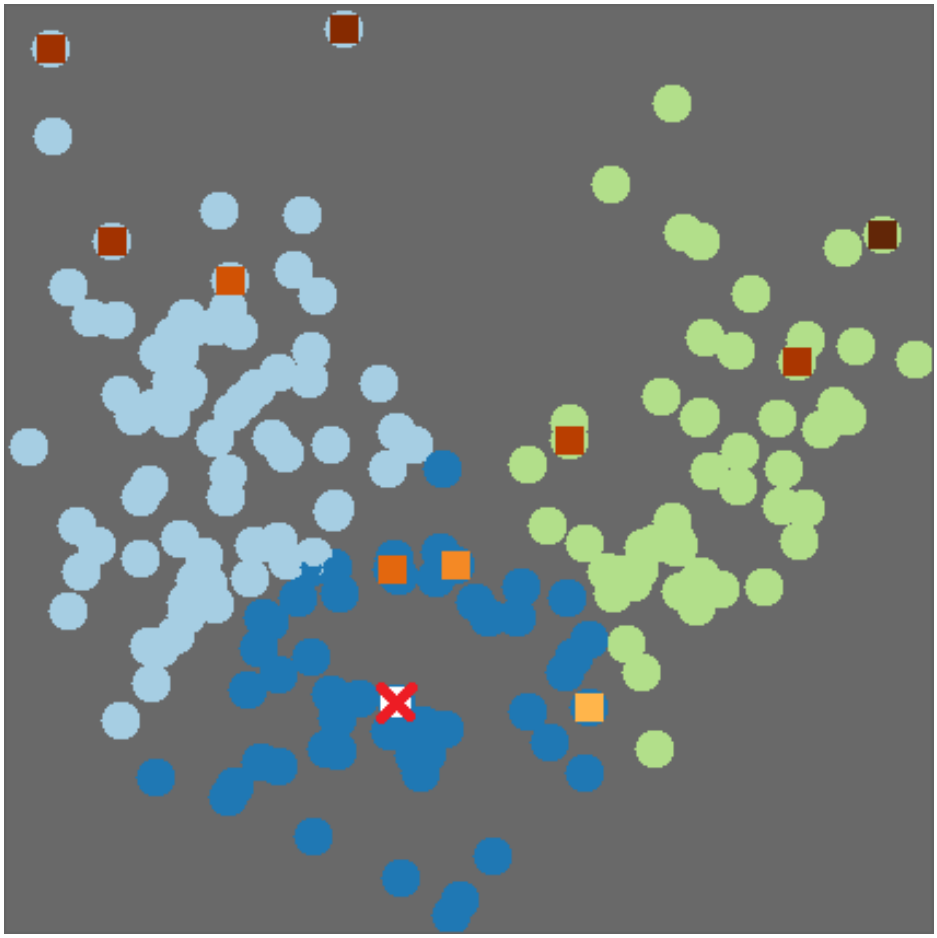


Figure 4.1: EigenSense demonstrated on an interactive scatterplot of projected data – all data points are laid out with multidimensional scaling (MDS) and colored by a spectral clustering. The point with a red X is the one the user clicked, asking what other data should be considered in relation to that point. The colored squares show the EigenSense response, with darker colors indicating higher EigenScores (see Section 4.4.1). Only the top five percent of scores from each cluster are highlighted, helping the user target the most fruitful data to examine.

The EigenSense method is best understood within a real interactive learning context. In Chapter 3 we described Dis-Function, a system that shows an analyst high-dimensional real-valued data in a 2D projection, and learns a distance function iteratively though user

feedback. Feedback is provided by dragging together points that should be closer together. The method is effective with appropriate user feedback, but the user is given no information about what points would be helpful to the metric learning backend.

To illustrate how EigenSense can help to close this information gap, we have integrated EigenSense into Dis-Function. When a user clicks a point of interest, EigenSense responds by showing several points that may be relevant to the first. Figure 4.1 presents a screenshot of this modified Dis-Function, specifically the data projection portion. The data have been arranged using multidimensional scaling (MDS), and colored based on a spectral clustering. The user has clicked the point marked by a red X. In response, EigenSense adds colored squares showing which points may be of interest relative to that X. Darker colors indicate a higher eigenvector sensitivity score, or EigenScore (see Section 4.4.1).

A user might pick a point of inquiry for many reasons, e.g. looking at outliers to find false ones or looking at the most familiar data first (perhaps a doctor examining a data point representing an especially familiar patient record). Alternatively, as pictured in Figure 4.1, a user could decide to look near the center of a cluster. As shown, clicking such a point near the middle of the blue cluster marks it with an X and causes EigenSense to respond by marking with colored squares some points that the user should compare to X. In the example, most points that EigenSense identifies as potentially interesting are in different clusters from X, or are outliers in the data. It makes sense that EigenSense would mark those points because if the user provided feedback that tightly linked any of these faraway points to X, the model would have to change significantly to account for such a large alteration.

The example also includes points recommended by EigenSense that are within the same cluster as X (drawn with the same color circle). They are in the same cluster as X because the clustering algorithm concluded they are similar to X, but that is not necessarily correct. If the user were to specify that these points are in truth very similar to X, that would tighten the cluster by improving the underlying distance function. The specific points EigenSense has highlighted in the cluster are those that are expected to result in the strongest updates.

In the example of Figure 4.1, the three highlighted points in the same cluster as X

are in fact in the same class as X in the ground truth labels for these data. The next-closest point, which has been placed in the green cluster, also belongs to the same actual class as X. Since the clustering is tightly connected to the underlying distance function, following the recommendation from EigenSense to consider this point and then fixing its association to X by correcting its distance from X could improve the model substantially.

These predictions of which points could provide the strongest update to the model are intended to guide the user towards giving the machine learner fruitful feedback. Looking at the highlighted points from EigenSense could prevent wasted time looking at less productive comparisons, and help to take best advantage of precious expert user time.

4.4 The EigenSense Method

In our interactive machine learning context, we have presented a user with data and need useful side information to improve our learned model. More specifically, in the context provided by Section 4.3, we assume a user examines a visualization of data and notices points of interest, perhaps outliers, cluster exemplars, or points aligned with personal expertise. We aim to answer, given one point of interest selected by the user, which are other points that the user should examine. We chose this interaction paradigm for two reasons: first, the user guides the process as opposed to simply being used for point comparisons, and second, having an initial point sharply reduces the computational complexity (see Section 4.4.2). In deciding which points to suggest for user examination, our ideal is to uncover the point that would make the strongest update to the model with the user's feedback, leveraging expertise efficiently to minimize effort.

In this section, we introduce a technique using eigenvector sensitivity on a pairwise distance matrix to provide these predictions of strong model updates. First we associate a score (called the EigenScore) with any pair of data points. The EigenScore of a pair is designed to predict the strength of a model update corresponding to user feedback about that pair. We then present the EigenSense algorithm, which uses EigenScores to recommend top candidate points to the user.

4.4.1 Calculating EigenScores

The EigenScore between two points represents our prediction for how strongly a change in distance between them would affect the underlying structure of the pairwise distance matrix. Specifically, it is a measure of the sensitivity of the dominant eigenvector of that matrix to changes in its elements, which correspond to pairs of data points.

Given a distance function and a data set with N points, we calculate the pairwise distance matrix

$$D \in \mathbb{R}^{N \times N} \text{ where } D_{ij} = distance(x_i, x_j)$$

Note that no specific type of distance function is required. To model how that matrix changes with specific x_i and x_j assumed to be perfectly close together, i.e. because the user specified so with feedback, we construct a new distance matrix D' which is identical to D, except that we set $D_{ij} = D_{ji} = 0$. These entries now reflect that x_i and x_j should be close to one another.

We next compute the dominant eigenvector for D, called v_1, and for D', which we indicate with v_1'. We compute the cosine similarity between v_1 and v_1'. Note that we desire a dissimilarity metric, showing how much v_1' is different from v_1, so we define

$$EigenScore(x_i, x_j) = 1 - CosineSimilarity(v_1, v_1')$$

Algorithm 1 summarizes this process.

Note that computing a function $eig(D)$ that returns all the eigenvectors, e.g. using a factorization method like SVD or the Cholesky decomposition, is computationally expensive [Hea01]. However, because computing the EigenScore requires only the dominant eigenvector (and because we are restricted to a real-valued symmetric matrix), we can dramatically improve performance by using the Lanczos method [Hea01], which returns only the dominant eigenvector and which we denote $eigs(D, 1)$ as in MATLAB.

Algorithm 1: EigenScore

Input: Data points x_i, x_j, distance matrix D
Output: $ES_{ij} \in [0,1]$

Calculate $v_1 = eigs(D,1)$ [dominant eigenvector]
Let $D' = D$
Set $D'_{ij} = D'_{ji} = 0$
Calculate $v'_1 = eigs(D',1)$
Set $ES_{ij} = 1 - CosineSimilarity(v_1, v'_1)$
return ES_{ij}

4.4.2 Using EigenScores to make EigenSense

The EigenScore algorithm maps a pair of points to a scalar value representing potential impact on the distance metric, and thus implicitly provides a ranking over pairs of points. This section addresses how to use this ranking with the goal of reducing user effort.

Calculating EigenScores over all pairs of points is prohibitively expensive for an interactive context. However, recall that in our usage context, the user has selected one point of interest and we must suggest options for a second point to pair with the first for a potential user constraint. Evaluating possibilities for just the choices of a second point requires only $(N-1)$ evaluations of EigenScore. We further limit the number of suggestions the user sees to some proportion $k \in (0,1]$ of the total data to save the user from examining every point. Rather than returning a fully ranked list of the top $k*(N-1)$ of the $(N-1)$ total points, we want to choose a *diverse* set of points for consideration. Our rationale is that we expect high EigenScores to correspond to pairs of points where user constraints would cause big updates to the model, but these may not be good updates. For example, outliers in the dataset will often contribute to high EigenScores, but should not necessarily be used in constraints.

To create the desired set of suggestions, we first cluster the data (using the current learned distance function), then sort the points in each cluster c by their EigenScore and return the top $k*|c|$ points within each cluster. This process is detailed in Algorithm 2.

The performance of our implementation is critical to demonstrating the feasibility of this technique for interactive systems. Our prototype system provides EigenSense recommendations on demand as response to interaction with a visualization. The current

implementation connects to MATLAB from C# via a COM interface to take advantage of the Lanczos algorithm for quickly calculating the dominant eigenvector. Still, as an example of performance capability, on a laptop with an Intel i5 480M processor, for a dataset of about 200 points with about 20 dimensions, an EigenSense response takes about one second.

Algorithm 2: EigenSense

Input: Initial point x_i, distance matrix D, set of clusters C, threshold k
Output: S, a set of model-critical points

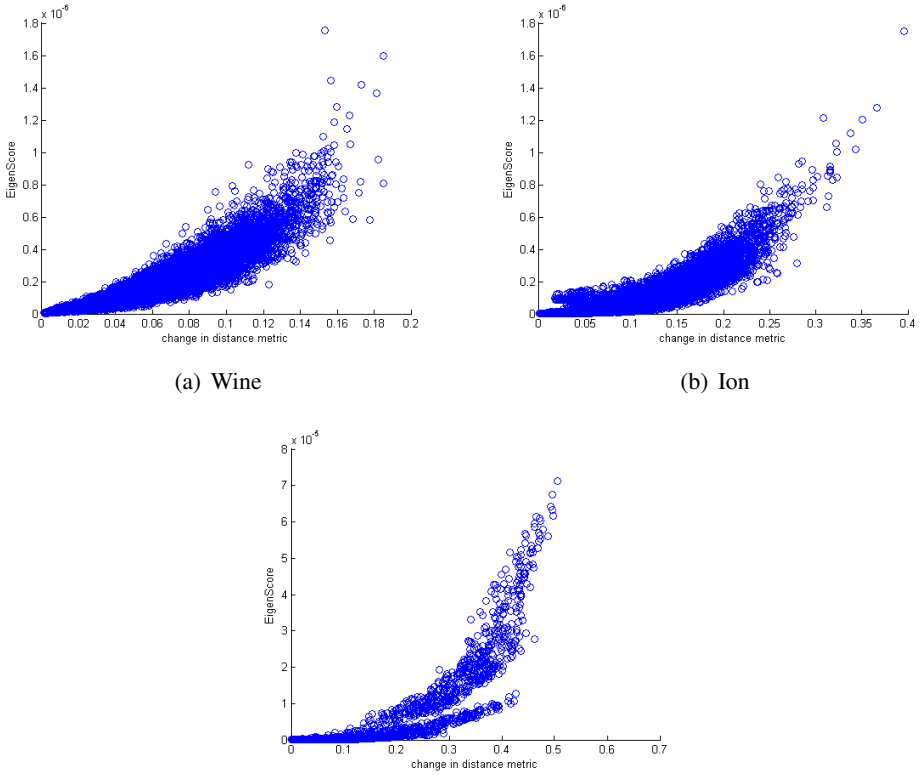
```
foreach cluster c ∈ C do
    foreach point x_j ∈ c do
        Compute ES_ij = EigenScore(x_i, x_j, D)
    Let S_c be the set of k × |c| points with the highest ES_ij
Let S = ⋃ S_c
return S
```

4.5 Experiments and Results

We validate the accuracy and effectiveness of our proposed method through two experiments on test datasets from the UCI Machine Learning Repository[BL13]. First, we compare EigenScores against actual values of the quantity they estimate and see that they could be an effective low-cost estimator of model change. Second, we evaluate the accuracy of EigenSense by considering the quality of the sets of points it offers to users compared against the ground-truth best points. We show that guided by EigenSense, a user could pick high-quality inputs while reviewing small amounts of data.

4.5.1 Experiment 1: Compare To Ground Truth

In this experiment we evaluate the EigenScores by comparing them directly to the value they are attempting to estimate. Recall that in our interactive metric learning context, EigenScores are an estimate of how much the distance matrix, as a stand-in for the distance metric itself, would be changed by constraining a given pair of points. The ground truth is prohibitively expensive to calculate for an interactive system, but can be prepared



(a) Wine

(b) Ion

(c) Parkinsons

Figure 4.2: Experiment 1 – In this comparison between EigenScores and the quantity they estimate, each point in each graph represents a pair of data points from the appropriate dataset. The horizontal axis shows the actual amount the underlying distance function changes when a given pair of points is constrained together. The vertical axis shows the EigenScore for that pair of points.

offline.

For three datasets, starting from scratch with no constraints, we used our prototype system (with interactive metric learning based on Brown et al. [BLBC12]) to calculate for each possible pair of points the actual change in distance function resulting from constraining the pair. The graphs in Figure 4.2 show the comparison of these values to the EigenScores. We use weighted Euclidean distance functions, thus the initial distance function is parameterized by the vector $\Theta_{init} = (1/M, ..., 1/M)$ of length M. In the graphs, the horizontal axis is the change in the distance metric from applying the constraint and the vertical axis is the EigenScore:

$$1 - \text{CosineSimilarity}(\Theta_{init}, \Theta_{post\_constraint})$$

Although the correlations between EigenScores and actual distance metric change are not obvious linear relationships, it is apparent from visual inspection that the quantities are related. This first pass evaluation shows the promise of EigenScores as an estimate of distance metric change, which implies that it could be an inexpensive way to predict model change for interactive machine learning.

4.5.2 Experiment 2: Evaluate Suggestion Quality

The goal of this experiment is to determine the quality of EigenSense recommendations by comparing them to the choices an oracle would make. Given an oracle that can rank all user feedback options in terms of which yield the best distance functions, we look to see how the EigenSense recommendations rank in that list.

We simulate choices of a point of interest x_i by the user, and then compute both the oracle's ranking of all possible constraint pairs with x_i, and the set of EigenSense options that would be presented to the user. Specifically, the oracle takes advantage of the labels for our test datasets to calculate, for all pairs of constraints that include x_i, the accuracy (with k-NN) of the distance metric resulting from an update with the given constraint. That is, given one point x_i, the oracle applies the system's metric learning algorithm with each constraint pair $(x_i, x_j)\ \forall x_j$, and evaluates each resulting distance function at classifying the data with k-NN. The accuracy scores of these evaluations provide a ranking over the constraint pairs. We compare the EigenSense options against the oracle ranking by finding the EigenSense recommendation with minimum oracle rank.

Figure 4.3 shows the results of our experiment. Each graph line corresponds to a different dataset, and each plotted point represents an average over ten simulations, each of which simulated ten user inputs. Simulated users picked a first point randomly then some (not necessarily optimal) EigenSense recommendation for the second. In total, each plotted point represents 100 uses of EigenSense. The horizontal axis is the k parameter of EigenSense (see Algorithm 2 and Section 4.4.2), which determines how many points will be shown to the user. Because the vertical axis shows the best oracle ranking of the EigenSense points, lower scores are better. It is no surprise that with larger values of k, where the user is being shown more points, the opportunity for the best-ranked points to be

included is higher. Using a low value of k means showing the user few points and saving effort, whereas using a high value means showing more points but having a better chance to show the absolute best ones. The results of this experiment suggest that, depending on the dataset, a user could give strong feedback to a metric learner while only reviewing less than ten percent of the data, or in some cases, substantially less.

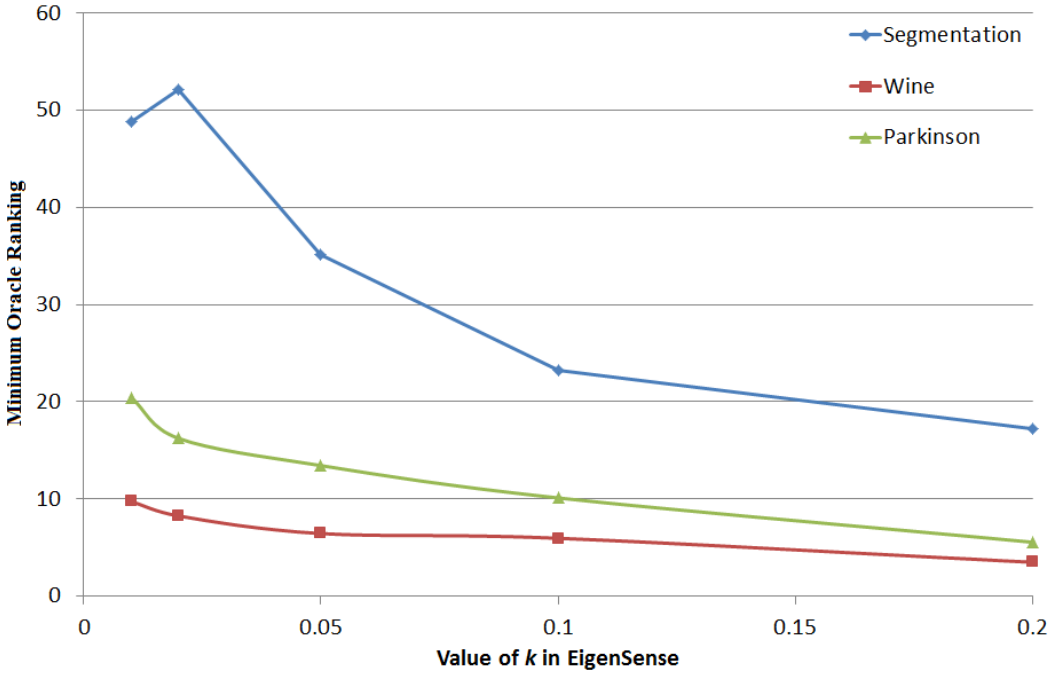


Figure 4.3: Experiment 2 – The horizontal axis shows values of the k parameter to EigenSense, i.e. how much data is shown to the user. The vertical axis shows the minimum (best) rank of the EigenSense recommendations in the oracle's ordering of all possible point pairs. Note that, as expected, as more data is shown to the user (k increases), there is more chance of the best possible options being revealed (rank decreases). Even with a small amount of data revealed, the EigenSense suggestions provide strong options.

4.6 Future Work

Although we have collected the presented evidence of EigenSense's effectiveness, there are opportunities for improving the algorithm itself. For example, there are several variations on how to generate pairwise distance or similarity matrices. Further, the performance of the implementation could be improved by using a library implementation of the Lanczos method for calculating the dominant eigenvector, instead of using MATLAB via COM calls.

The performance improvement is critical for the main thrust of future work, which is to complete the evaluation of the technique by testing it with human subjects. In particular, participants in a user study will use the tool to cluster some images with known classes. We can then evaluate their comfort with the tool, confidence in the recommendations, and progressive accuracy of the distance metrics learned from their inputs to see if they do better with or without EigenSense.

4.7 Summary

This chapter contributes to the study of interactive metric learning by applying active learning to reduce the workload of the human actor. We introduced the concept of EigenScores based on eigenvector sensitivity of distance matrices, and then applied these to create the EigenSense algorithm, which identifies and recommends points for user consideration given an initial exploratory direction. We presented evidence of the effectiveness of the algorithm by demonstrating its correlation with ground-truth values of the quantity it estimates, and then by showing the frequency with which EigenSense presents the best possible option to users. Our results indicate that EigenSense could help save human workload by vastly reducing the number of data points to be considered while maintaining near-optimal metric learning results.

Chapter 5

Doc-Function: Visual Text Analytics with Interactive Model Learning in Keyword-Centric Spatializations

This work is based on a paper submitted to *Visualization and Graphics, Transactions on (TVCG) 2015*:

Eli T. Brown, Kris Cook, Remco Chang and Alex Endert. Doc-Function: Visual Text Analytics with Interactive Model Learning in Keyword-Centric Spatializations.

5.1 Introduction

The approach explored through this dissertation focuses on capturing user interactions and codifying the generated hypotheses numerically by learning data models using back-end machine learning. Chapter 3 demonstrates, via the Dis-Function prototype, the potential of such systems, and Chapter 4 shows how to improve their efficiency with respect to users' time. However, the techniques presented so far are only applicable to numerical data. A different kind altogether, text data abounds and is of critical interest to multiple parties, including for intelligence and anti-terrorism purposes. Fortunately, text data lends itself well to this type of implicit model-building.

One effective visual metaphor for exploratory sensemaking of text corpora is spatial

layout. The use of space as a means for users to express meaning without requiring formal reasons, annotations, or other direct descriptions has been shown to be beneficial to analytical reasoning and the discovery processes [SIM99]. Further, Andrews et al. showed how the ability for analysts to organize documents spatially enabled them to maintain and build insights over the duration of a sensemaking task [AEN10]. Similarly, computationally-generated spatializations exist that create layouts where similarity is visually encoded by distance. For example, the IN-SPIRE Galaxy View shows documents in a spatialization, from which analysts can observe groups and regions of documents that pertain to a specific data-driven concept of set of topics [WTP+95]. Such visual metaphors are effective for sensemaking of text corpora as they enable inherent flexibility and subjectivity on the part of the analysts. This flexibility of spatializations provides an effective bi-directional medium for user interactions to couple with visual outputs of data analytic models. The challenge, then, is in how to combine the user's domain expertise and feedback into the computational processes.

Model steering has been used with text applications in previous work. For example, in IN-SPIRE's Galaxy View, users can focus on some subset of topics in a document corpus by emphasizing specific keywords, then re-run the model to create a new spatialization, and generate an updated view [BCBN11]. Directly interacting with such model parameters requires high formality and expressiveness in context of the analytic model being used [ENCZ]. Depending on the domain expertise of the analyst, and the stage of the analytic process, this may not be desired [SIM99].

Text data has also been tackled with automatic model-steering, i.e. performing updates based on user interactions with visual forms as opposed to parameters [EHR+14]. For instance, re-positioning documents in a spatialization is interpreted by the system as updating the parameters of the distance function representing similarity (as opposed to requiring users to update those parameters directly) [EFN12b, BNH14]. These methods create bi-directional spatializations, where the visual metaphor is used not only for communication of the model's approximation of the data relationships, but also to enable users to communicate their expertise and prior knowledge of these relationships. For example, users can create groups and clusters of documents in the spatialization, from which the system solves

for the distance function or dimension weighting that corresponds to those relationships. In contrast to direct manipulation methods above, such automatic model-steering approaches require less formality and knowledge of the underlying models being used.

The approach presented in this chapter falls into this automatic category of model steering. However, instead of directly manipulating the location of documents (or observations [EHM+11b]) in spatializations, our technique enables users to re-arrange terms extracted from the documents. In "document-centric" approaches, analysts must first read some of the documents to discover the relevant terms, and decide which documents to reposition spatially. As dataset sizes increase, it may take longer for analysts to locate two or more documents for which a similarity assessment can be made.

In this chapter, we present a novel visual analytic technique for "keyword-centric" spatializations for visual data exploration of text. As opposed to the traditional methods of visualizing a spatial layout of *documents*, our approach first extracts a set of *keywords* from the documents and encodes the similarity between keywords spatially. The keyword-centric spatialization is similar to context-preserving word clouds [CWL+10], in that pairwise distances between keywords encodes their relative similarity. As a result, clusters of keywords represent related concepts. Analysts interact with the keywords using the same implicit model steering technique. However, the keyword-centric approach bypasses the requirement for the analyst to read the documents before being able to manipulate the spatialization. Instead, by reading the keywords on the screen, the analyst can immediately begin to form hypotheses and gain a high-level gist of the concepts within the documents. We present an implementation of our technique in the prototype, Doc-Function, designed to enable sensemaking of text corpora.

We evaluate the utility of Doc-Function through an exploratory user study consisting of analysts performing a sensemaking task. We show that users were successfully able to use Doc-Function to perform a text sensemaking task. Finally, leveraging the conceptualization and the high-dimensional space created using our keyword-centric approach, we demonstrate that each analyst's interaction trails can be visualized and compared. We call this high-dimensional space "Model Space" and show that this approach represents an effective method for encoding and visualizing multiple users' analytic provenance. visually

explore and compare the incremental adaptation of the data model over time for each analyst. We show that this method effectively shows the analytic provenance of the analysts in our user study.

The primary contributions to visual analytics described in this chapter are:

- A visual analytic technique for creating and interactively steering keyword-centric spatializations
- A prototype application, Doc-Function, demonstrating the utility of our technique for visual text analysis
- An evaluation of our prototype, showing the effectiveness of our approach for performing a sensemaking task
- A novel visualization called ModelSpace for visualizing users' interaction history and analytic provenance based on the implicit, keyword-centric model steering technique

5.2 Learning Inverted Document/Keyword Models

Similar to previous work on learning models from user input we use an iterative interaction process [BLBC12], as illustrated in Figure 5.1. We present a visualization of data and ask users to manipulate data directly. A back-end machine learning algorithm processes the input to learn a new model and the system uses that model to present a new visualization that is closer to the user's intended mental model. This iterative process allows incremental improvement, and when the user is content with the visual representation, the model representation can be used for analytical purposes.

In this particular work, the data is a text corpus, the data points users interact with are keywords extracted from the documents. The features of information corresponding to each keyword, i.e. the dimensions of the dataset, are the documents in which the keywords appear. The visualization is a multidimensional scaling (MDS) [BG05] of keywords extracted from the document collection, and the back-end machine learning algorithm is based on the work of Brown et al. [BLBC12] (discussed in Chapter 3), producing a distance function based on the user changing distances between groups of data points. This

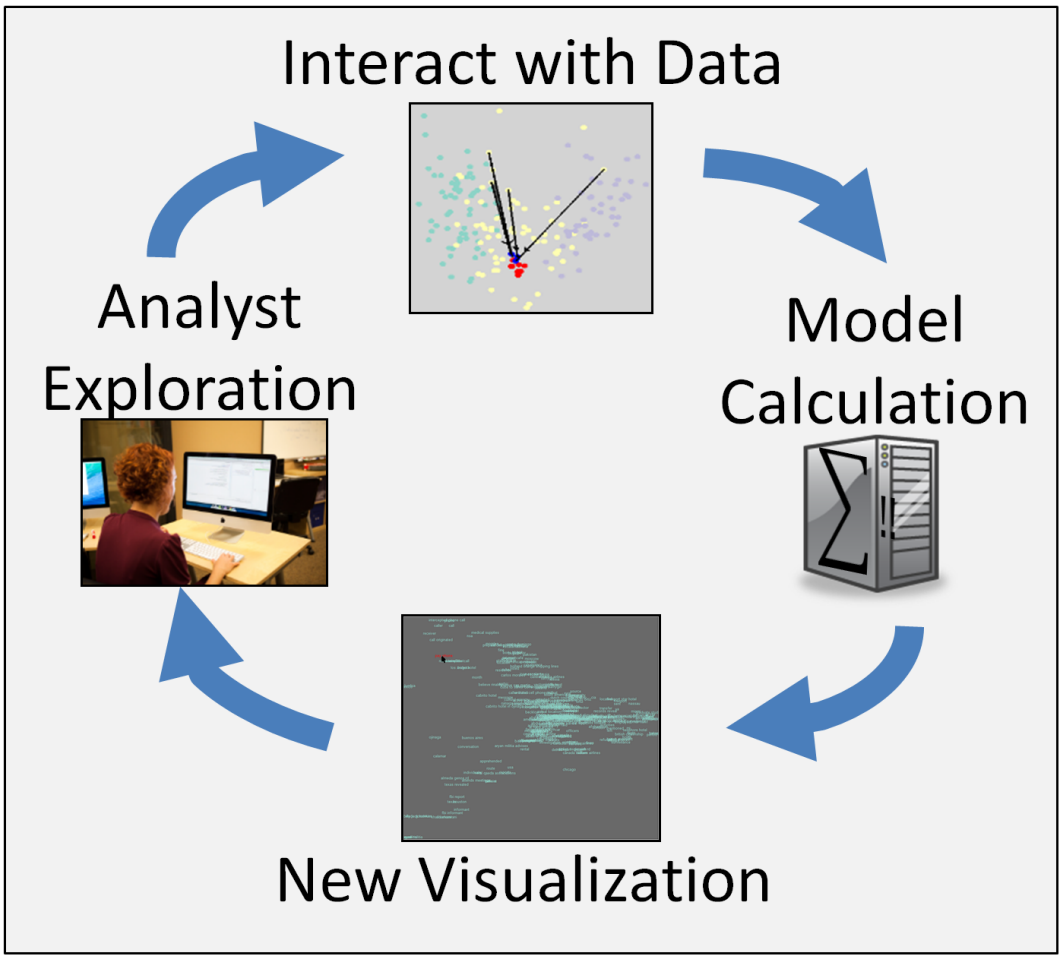


Figure 5.1: This figure shows the iterative feedback model. The first time through, a user views and explores an initial visualization. His or her manipulations are fed to a machine learner in the back-end, which produces a new model and thus a new visualization for further exploration. Iteratively, the user approaches a visualization that corresponds to his or her mental model, and can use the corresponding machine learning model.

section first covers the specifics of the keyword extraction and projection, and then explains how user feedback is used to update the model.

5.2.1 Extracting Keywords from the Document Corpus

Traditionally, layout mechanisms for analysts to review bodies of text have been based on showing representations of documents [EFN12b, WTP+95, CLKP10]. Instead, we show the user a layout of keywords extracted from the documents. Similar to the concept behind word clouds [VWF09], we believe that these keywords are more likely to map to users'

concept maps than the full documents, and thus the layout of the keywords will be a better proxy for the user to express his or her mental model.

In order to extract keywords from the documents, we apply Rapid Automatic Keyword Extraction (RAKE) and Computation and Analysis of Significant Themes (CAST) [RECC10, RBC+09]. First, RAKE scans each document and extracts short sequences of words separated by stop words and phrase delimiters (e.g. punctuation). A graph of co-occurrence of constituent words within the candidate keywords is computed and statistics over that graph are used to score keywords. Those keywords with high enough score are retained. Note that in order to preserve keywords that may include stop words, like the phrase *time of day*, RAKE creates extra possible keywords to test out pairs of extracted keywords when they appear at least twice in the same order within a document [RECC10].

The CAST algorithm [RBC+09] is applied to the result of RAKE. The keywords are evaluated to discover keywords, or *themes*, in the text corpus, which can envelop multiple keywords. The CAST algorithm scans all keywords from all documents and builds a hierarchical agglomerative clustering based on the idea that two keywords are similar if they co-occur in documents. Themes are chosen from among the clusters based on an evaluation formula detailed in Rose et al. [RBC+09], and the themes are named after their keyword with the highest association to the documents covered by the theme. These themes then include sets of keywords that tend to appear in the same sets of documents [RBC+09]. Using RAKE and CAST allows us to show the user higher-level concepts like *denver bank accounts* or *egyptian passport* as opposed to just single words or even single keywords. We continue to call these extracted themes keywords, $k_1, \ldots, k_N$, and use them as the entities in our spatial text visualization.

5.2.2 Generating the Text Layout

In a model-steering system, the visualization used to gather feedback from the user should reflect the model so the user can see his or her model-steering influencing the visual representation of the data. In this case we are building a distance-metric model, which means the visualization must reflect the distances between data points (i.e. keywords). To determine the distances between these keywords, we use a modified Bray-Curtis dissimilarity metric

[BC57]. The original Bray-Curtis formula defines the dissimilarity between two keywords k_i and k_j based on their co-occurrences in documents and is defined as:

$$BC(k_i, k_j) = 1 - \frac{2|docs(k_i) \bigcap docs(k_j)|}{|docs(k_i)| + |docs(k_j)|} \tag{5.1}$$

where $docs(k)$ is the set of documents that include keyword k. This function indicates what proportion of the total possible documents in common the two keywords share. Thus the Bray-Curtis dissimilarity measure ranges from 1 (the keywords are used in none of the same documents) to 0 (the keywords are used in all of the same documents).

The original Bray-Curtis formula in (5.1) is based on sets of documents that contain keywords. In order to convert this formula into a vector formulation that better fits our needs, we adopt the equation provided in the CAST algorithm [RBC$^+$09]. First, consider each keyword to be represented by a vector with length equal to the number of documents in the corpus. Thus each keyword's vector, i.e. k_i for keyword i, has an element k_{il} corresponding to keyword i's relationship to document l for all M documents. That relationship is encoded by the inverse document frequency (IDF) [MRS08a]:

$$idf(k_i) = \log \frac{M}{|docs(k_i)|} \tag{5.2}$$

$$k_{il} = \begin{cases} idf(k_i) \text{ if } k_i \text{ is in } doc_l \\ 0 \text{ otherwise} \end{cases} \tag{5.3}$$

Note that the denominator is never zero because there are no keywords that do not appear in a document. Then using the vector specification of 5.3, the vector-based Bray-Curtis is:

$$vectorBC(k_i, k_j) = \frac{\sum_{l=1}^{M} |k_{il} - k_{jl}|}{\sum_{l=1}^{M} k_{il} + k_{jl}} \tag{5.4}$$

Finally, in order to use this equation in a model-steering context, we require that it be dependent on a model. Similar to Brown et al. [BLBC12], our model is a weight for each

dimension of the data, i.e. for each document, expressing the relative importance of that document in determining the similarities of keywords. Therefore we propose a version of the Bray-Curtis dissimilarity that applies a vector $\Theta = [\theta_1, \ldots, \theta_M]$ of weights, one for each document. When a system is initialized, Θ will be initialized to $\theta_i = 1/M \ \forall i$, but updates from the user will change these values to incorporate knowledge about the data. Using the vector representation and applying the weight vector, we rewrite Bray-Curtis:

$$weightedBC(k_i, k_j, \Theta) = \frac{\sum_{l=1}^{M} \theta_l |k_{il} - k_{jl}|}{\sum_{l=1}^{M} \theta_l (k_{il} + k_{jl})} \tag{5.5}$$

One further adjustment is required for this application. Due to the sparseness of the data from CAST, we augment the space of documents with dummy documents that each represent one keyword. We define the document $kwdoc_{k_i}$ for any keyword k_i as a document containing all the keywords that co-occur with k_i in any document, i.e. the document $kwdoc_{k_i}$ contains the set of keywords:

$$\{k_j : |docs(k_i) \bigcap docs(k_j)| > 0\} \tag{5.6}$$

With the appended document set, the vector for keyword k_i has an entry associated with each real document, and then an entry associated with each dummy document The entries for the dummy documents are calculated as for normal documents, following Equation 5.3. For vector elements corresponding to dummy documents, e.g. document $kwdoc_{k_j}$, the value is nonzero only when keyword k_i co-occurs with keyword k_j in some document. In total, the vector corresponding to a keyword has elements representing documents with the following structure:

$$k_i = [doc_1, \ldots, doc_M, kwdoc_{k_1}, \ldots, kwdoc_{k_N}]$$

These adjusted vectors have length $M+N$, but because these vector elements corresponding to keywords are dummy documents, we now redefine the value of the symbol for the number of documents, M, to be this new value that includes the count of the dummy documents, and use M to mean this quantity going forward.

Using the modified Bray-Curtis distance between these keyword vectors, we cre-

ate an MDS projection of the keywords for a two-dimensional visualization by creating a matrix of pairwise dissimilarities between keywords, applying Principal Component Analysis (PCA), and projecting onto the two strongest principal components. This projection space optimizes for maintaining the similarities from the Bray-Curtis space down to the two-dimensional viewing space.

5.2.3 Interpreting User Interaction

Creating a visualization of the text data is only the first step. The projection described is intended to represent a learned model, Θ. To create a model-building experience, we must be able to convert user feedback with the visualization into a model update. We can refer to the model-parametrized Bray-Curtis distance of Equation 5.5 as a parametrized distance function:

$$D(k_i, k_j|\Theta) = weightedBC(k_i, k_j, \Theta)$$

In this form, we use it in the machine learning back-end formulation from Brown et al.'s work, Dis-Function [BLBC12], by using optimization over the objective function

$$\Theta^t = \arg\min_{\Theta^t} \sum_{i<j\leq N} L_{ij}^t \left(D(k_i, k_j|\Theta^t) - U_{ij}^t \cdot D(k_i, k_j|\Theta^{t-1})\right)^2 \tag{5.7}$$

After a user has interacted with a visualization that encodes an underlying model $\Theta(t-1)$, we learn a new model Θ^t that encapsulates the feedback from the user. This feedback is encoded in the form of the matrix U where each entry U_{ij} shows the amount of change the user made to pairs of keywords (i, j).

In our approach, we support two methods for a user to update the model. Either the user provides two groups of points $Y1$ and $Y2$ and requests the groups be moved relative to each other (this is referred to as a "*two-group update*"), or the user provides a single group of points, requesting that the points in the group be updated relative to each other (referred to as "*one-group update*"). In both cases, entries in U corresponding to keywords the user did not include in any group are set equal to 1, while those involved in the update are set

equal to the ratio

$$\frac{\text{user adjusted distance}}{\text{original distance}} \text{ (for distances in the projected space)}$$

More specifically, in the case of the two-group update, the entries U_{ij} corresponding to points where $x_i \in Y1$ and $x_j \in Y2$. For the one-group update, there is only one group, $Y1$ and the entries of U are updated where $x_i \in Y1$ and $x_{j\neq i} \in Y1$. Note that the L^t_{ij} coefficient simply encodes a scaling factor to ensure terms of the summation that include interaction points have higher weighting. For a more detailed explanation of Equation 5.7, refer to Chapter 3, Figure 3.3.

The optimization seeks to minimize change to the overall system while taking into account changes to relative distances imposed by users [BLBC12]. The algorithm produces a new value of Θ that includes the feedback, and this new model is used to regenerate the visualization as well as becoming the new representation of the user's mental model.

5.3 The Prototype

To evaluate the efficacy of keyword-centric model-steering sensemaking, we built the Doc-Function prototype. While Section 5.2 explains the technical aspects underpinning the system, this section describes the Doc-Function prototype and shows the interface that participants in our experiment used to uncover a fictitious terrorist plot (see Section 6.2).

The Doc-Function system is designed around the concept of keyword spatialization, with additional features to facilitate exploration and discovery. The main interface is shown in Figure 5.2. Occupying most of the window is the projection of the keywords. In this region, a user can drag and drop keywords to explore. When the mouse cursor is over one or more keywords, the corresponding documents are shown in the panel on the right for perusal. This way, the user can immediately relate the concepts he or she is considering to the content that is responsible for their relationship and that underscores the objective. Further exploration or comparison of document sets can be achieved using alt+click on a keyword, bringing up a window containing the related documents. Multiple windows can

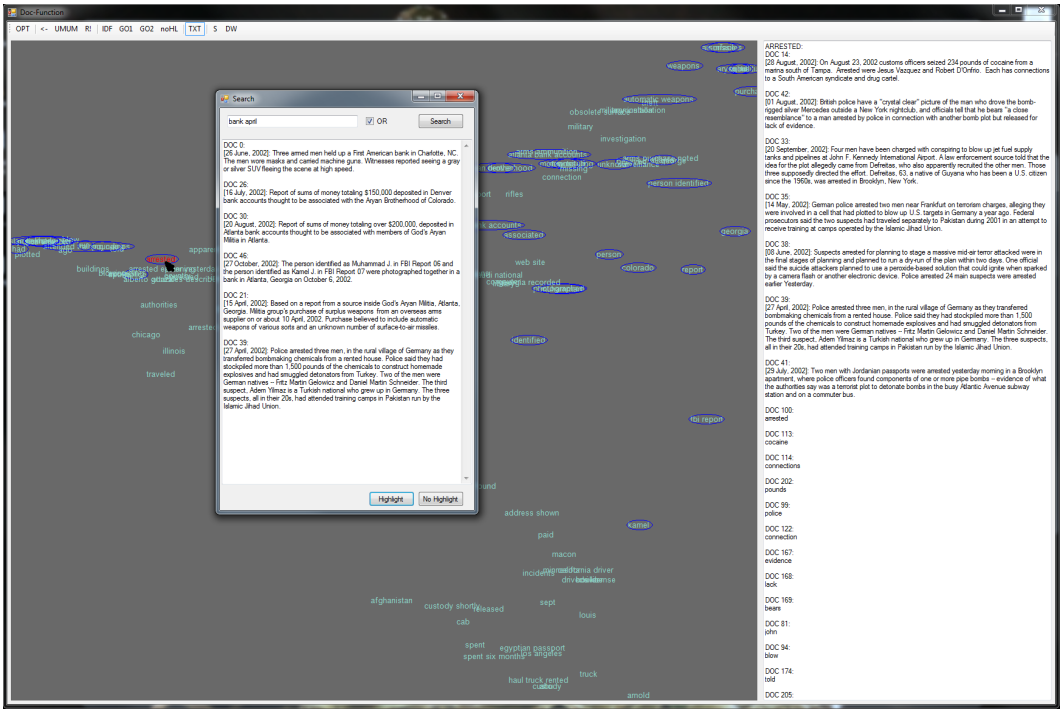


Figure 5.2: The Doc-Function prototype shows a projection of the important keywords from the collection of documents (see Section 5.2.1), and allows a user to move them around and provide feedback on their placement to a machine learning back-end (see Section 5.2.3). The right column shows the documents that use the keyword the mouse cursor is over ("*arrested*"), and the buttons along the top provide helpful functionality like undo, reset and search. This figure also illustrates the search capability. The pop-up window allows a user to search for one or more string tokens, and shows all the documents that include those tokens. The *Highlight* button draws circles around all of the keywords that contain those documents, as seen in the figure.

be opened this way, allowed comparison of document sets side-by-side.

To communicate to the system that certain groups of keywords should be closer or further away, a user holds the control button while moving them. The keywords are written in red or blue based on which mouse button was used (left or right respectively). There are two ways of using this point-marking mechanism to update the model. The user can order a one-group update or a two-group update. As explained in Section 5.2.3, these two types of update are encoded differently for the back-end machine learning, but both results in a model update that takes user feedback into account. The separate groups of keywords marked in red and blue with the two different mouse buttons are used as the two different groups, $Y1$ and $Y2$ for the learner, or just $Y1$ in the case of a single-group update.

Either update is ordered with a button on the toolbar, after which Doc-Function learns a new model based on the feedback. Through an optimization, this process results in a new underlying model, i.e. a new distance function that takes into account the changes to relative positioning of keywords given by the user's feedback feedback. The new distance function is used to regenerate the visualization, which is directly dependent on the distances between points (see Section 5.2 and Figure 5.1). After updating the model, Doc-Function highlights the moved keywords in the new projection so the user can see how the changes affected those points. Those highlights, as well other types, can be turned off with a toolbar button. Another toolbar button gives the user a view of the model directly, showing which documents are weighted most highly by providing a popup window with all the documents ordered from most strongly weighted to weakest, along with the actual weight values.

If the user is unhappy with the visualization given by a new model after an update, the second-from-left section of the toolbar provides features for undoing the changes. There is an undo button and a reset button. While the first undoes only one step, returning the internal model to its previous state and re-rendering the visualization accordingly, the reset button returns the model and visualization to the original state, i.e. with an evenly-weighted distance function. Finally, the user also has the ability to undo only the interaction performed since the last update. This option does not change or update the model, but it only puts anything the user has moved or highlighted back to its original location and state.

The Doc-Function prototype also includes features to facilitate the process of sense-making. Most importantly, there is a search function seen in use in Figure 5.2. The small text box on the pop-up window can be used to search for a phrase, or the presence of any of the included strings (based on the option button beside it). The larger box shows all the documents that were found in the search. On the bottom of the window, the Highlight (and corresponding No Highlight) buttons toggle circles around keywords in the main visualization window that appear in the documents that responded to the search. In the figure, highlights have been turned on, and several circled keywords are visible on the keyword spatialization in the main window, helping the user see where the search could direct them to look in the spatial layout.

The prototype system Doc-Function is built in C# (.NET Framework version 4.5)

using Windows Forms and GDI+ with the ALGLIB library [BB13] for optimization and the Accord [dS12] library for calculating principal component projections.

5.4 Evaluation

To evaluate the the proposed prototype, Doc-Function, and its underlying model construction, as well as to collect data on how users took advantage of model-steering and spatialization, we conducted a user study. In this section, we explain the source data, the characteristics of the groups of participants, the task they performed, our procedure for the experiment, and the data we collected.

5.4.1 Data

Using a dataset designed for intelligence training, we asked our participants to find a fictitious terrorist threat embedded in the data. Specifically, the data consists of 49 documents and the keyword extraction identified 165 keywords (see Section 5.2). Each document contains on average 7.7 of the keywords ($\sigma = 3.5$, mode tied for 4 and 10). Correspondingly, each keyword is found in 2.3 documents ($\sigma = 1.5$, mode = 2).

Information about the threat is spread out over several documents. To make the task more difficult, the corpus contains distracting documents that are superficially suspicious but are not actually helpful for the task. In total, the task is reasonably difficult, but can certainly be performed successfully without intelligence training.

5.4.2 Participants

Our study population consisted of 13 participants at the Pacific Northwest National Laboratory (PNNL) in Richland, WA. They included 7 males and 6 females. We chose our participant group to include varied levels of experience with intelligence and text analysis to see how a spectrum of expertise would be applied to Doc-Function. Our participant group includes personnel who are professional analysts (2), scientists and engineers (5), interns (5), and administrative staff (1). The scientists and engineers were from varying research backgrounds including Mathematics, Biology and Computer Science.

5.4.3 Task

We asked each participant to use the Doc-Function prototype to uncover a fictitious terrorist plot threaded through a document corpus. These documents include mock field reports and mock reports from other intelligence gathering and collection. The threat is not written in a single document, but rather must be pieced together from other intelligence and extracted from false leads. The participants could use all the features of Doc-Function described in Section 5.3. The scheduled experiment periods were one hour, but the participants were given flexibility. They all continued until satisfied that they had discovered the plot or found everything they would be able to, using no more than an extra twenty minutes. Using a talk-aloud study, we encouraged the participants to explain their reasoning processes as they worked.

5.4.4 Procedure

After filling out a participation agreement, each participant was given a writing utensil and a sheet of scrap paper (and told they could have more). The administrator of the task, who was the same for all participants, then provided a tour of the Doc-Function system and its features. We explained to all participants the nature of the talk-aloud study and continued to encourage them to talk during the course of the task. We also continued to encourage them to use the model update mechanism to maximize our data collection, especially when they were unsure what step to take for their analysis. Whenever asked, we provided help with how to use any of Doc-Function's features. The time required to calculated model updates varied, and so we took advantage of extra time when possible to probe participants for extra information about their though process and direction.

Finally, when the participant was finished, we administered a survey to check that the plot was uncovered and learn what features were useful and what could be improved. We asked for a summary of findings, evidence to support those, and organizations, people and places involved. We also asked for hypotheses considered but not pursued, and examples of backing up (often corresponding to when the participant used the undo or reset features) during analysis. We also asked for feedback with the tool, including favorite functionality

and proposed missing functionality.

5.4.5 Data Collection

With their consent, all participants' efforts at the task were recorded on video (with audio). The Doc-Function system logs all types of interactions: mouse-over document viewing, alt+click document viewing, searching, search highlighting, viewing document weights, reset, undo, undo since previous model, and both types of model update, including information about which keywords were moved to where.

Those records are accompanied by copious notes taken by the administrator. These notes include comments of participants about their processes and descriptions of times when participants had certain insights about the plot. Further they show when participants took advantage of the spatialization, used the reset feature, or discussed people or certain places. All these observations were recorded with timestamps and have been coordinated with the software logs.

5.5 Results

In this section, we discuss the results of our evaluation with our keyword-centric, model-steering text analytics prototype for sensemaking. We partition the results into two sections. First we cover the participants' success with the prototype tool, and second, we discuss their feedback.

5.5.1 Participant Success with the Tool

Perhaps most important to note is that nearly all of the participants discovered the fictitious plot in the document corpus. Because we had a range of skill levels involved in the task, there is certainly a range of depth of understanding of the plot, but every participant but one was able to identify elements of the threat for further consideration. This proportion yields a success rate of 92 percent.

More interesting are the ways in which we can characterize the different uses of the tool and utility of the different features. The spatial layout was useful to almost every-

one. Participants were not always explicit upfront that they were taking advantage of the layout. Rather they were often able to leverage the spatial layout apparently through intuition and based on their understanding and our explanation of what the layout of keywords represented and how it is made (i.e. similar or related keywords should be closer to each other). Because of the talk-aloud nature of the experiment, we were able to ask participants what influenced their choices of which documents to read and recorded that their responses involved reasoning about the layout. For example one participant mentioned, "*[These] are all about intercepting plans,*" and when asked how he came to find that set of documents he pointed, "*They were all crammed in there together.*" As one analyst started reading, the administrator asked, "How did you decide to look at that keyword," and in response the analyst pointed to part of the screen and said simply, "*I just decided I was spending too much time over there.*" Another example was a participant who pointed at the screen and explained, "*Seems like the weapons stuff is all over in this area now, which is good*." Overall, in our observations of the participants and discussions during their reasoning process, we found that all but one of them took advantage of the spatial layout in deciding what to read or what to interact with. The one participant who did not use the spatialization is also the one who did not satisfactorily uncover the plot. Of course, this is not enough evidence to conclude that the spatialization was strictly necessary for the task, but it was certainly part of the process of successful investigations.

Some features of Doc-Function were more popular than others. For example, only one participant used the two-group model update. In our previous work with model-steering systems, we have seen that even when a model update requires specifying two groups, users may think about providing feedback in terms of grouping one set of data points together. The single-group update may have been more popular because it was more intuitive. The one participant who did use the two-group update liked the feature and explained using it like an OR operator among the keywords in each group. This is a semantic meaning of the operation that the authors had not considered. Further, the new semantic intent of this participant is a reminder of the rich possibilities for further interaction techniques that are possible with keywords. For example, performing a search has been investigated as semantically useful in related model-steering work with documents, [EFN12b], but with

the keyword-centric approach and the search-and-highlight functionality of Doc-Function, we could leverage the connection between the search and the set of highlighted keywords to ask the user which were the most useful highlights, and update the model by reinforcing those highlighted keywords' connections to any search keywords.

Most participants took advantage of either the undo or reset function. In total, five participants used undo and two used reset, with no overlap between the two groups. They used it for different reasons, but overall it was because the model built from their interactions was not helping. For example, one participant said he had made a mistake by, "Collecting too many [keywords] at once." Giving participants the ability to undo or start over gave them more control over their investigations.

Every single participant made use of not only search but search highlighting. For example, participants would notice people or places in documents and then search for their names to see what other keywords were relevant to those entities. Participant 12 described searching for a person's name and then using the highlight feature to see a useful grouping in the spatial layout. The name appeared in documents whose keywords were grouped together in the spatialization. The search feature connects the keyword space to the document space, and the highlight connects the document space back to the keyword space. Together, these features helped people understand connections between one keyword and others through documents.

5.5.2 Participant Feedback

In the course of discussions with the study participants and in the post-task survey, we collected feedback on Doc-Function itself. Participants had overall good impressions, but we also uncovered potential areas for improvements and features to test in future research.

In the survey at the end of the study we asked participants what features they liked and what features may have been missing. Again, search was popular: "I liked a lot. [I] Liked the search..., the search is nice because I can just pull up a keyword thing and then... being able to read the documents in that format helps." Another participant, "[I liked] that I could do a search and highlight all the [keywords] that contained documents that had that search [keyword] in it (sic)." And another included the pop-up showing all documents

associated with a keyword, saying, “”[I liked] both the search and the alt-click, those things were excellent, that helped a lot.” Participants went further and suggested ways to make search more powerful, like an option to select keywords in the layout and have all the corresponding documents show.

One common theme participants thought could be improved was the visual layout of keywords. One explained, for example, wanting to “be able to spread the keywords enough to see them.” Keywords were drawn with overlap, making it difficult to see some words before items had been moved around. Based on this feedback, as part of future iterations of the Doc-Function system, we aim to add *jitter* to the keywords to remove overlap between the text. In addition, to further encode more information into the visualization, we will integrate Word Cloud techniques including using font size and color to indicate such properties as read vs. unread, keyword frequency, of the keywords and color to indicate

While participants found the spatial layout of keywords useful, several suggested that the next step toward helping them build mental models would be tools or even automation that could help keep track of people, places and dates. We aim to include a mechanism for indicating when keywords belong to these categories so that they can be made prominent or even visualized separately, i.e. with coordinated views of a graph for connected people, a map for places, and a calendar for dates.

Overall we are encouraged by the reactions of study participants to our prototype. They took advantage of most of the features provided and gave us helpful insight into how the software could be improved to better enable their analysis experience in the future.

5.6 Numerical Provenance

In this section, we explain ModelSpace, our novel method for leveraging the numerical models produced during model steering to examine the provenance of users’ analysis (see Figure 5.3). We first explain our specific instance of numerical provenance. Next we describe how we leverage that provenance information along with other data collected during the Doc-Function evaluation to build a visualization. Finally, we discuss findings about our participants’ use of the software uncovered by this visual analysis.

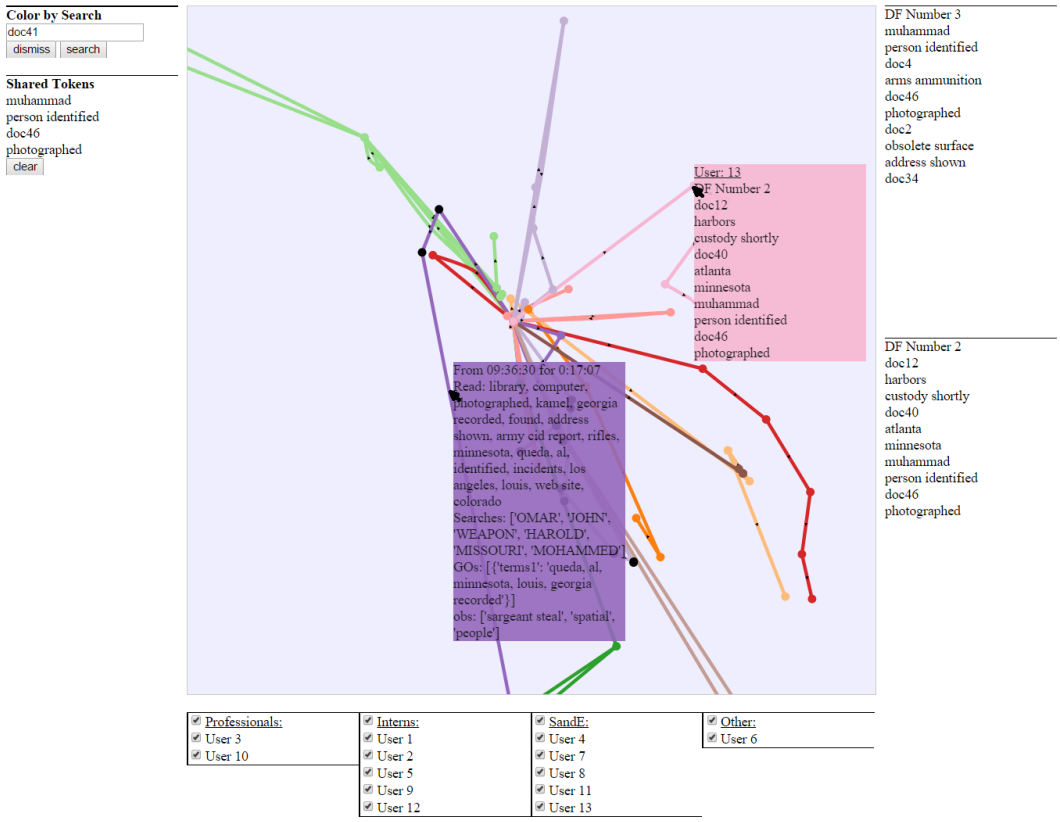


Figure 5.3: This figure shows ModelSpace visualizing the paths multiple participants took through the space of possible models during the course of the Doc-Function evaluation. The dots represent models, i.e. some Θ_u^t achieved by some participant at some point in the analysis task. The lines connect the models and represent the time in between model updates. User identity is encoded with color. Arrows on the lines indicate the ordering of the models. An example of mouseover text is included for both a dot and a line. For a dot, this shows the ten most highly-weighted documents, and for a line, this shows all the activity that occurred between the models that the line connects. Note that this view is zoomed in and some of the lines connect to dots outside the viewing window.

5.6.1 Participant Exploration Space

During the course of their explorations, our participants saw several spatial layouts of the data. Each layout was a manifestation of the model built from their model-steering inputs, and each model is represented by a vector in $\mathbb{R}^M$. The models are a special form of provenance information - a set of numerically encoded models in sequence that track the user's progress through the space of possible models, i.e. the "*model space*". We sought to take advantage of this unique, numerical provenance data by examining and comparing the paths

different participants took, providing insight into how participants used Doc-Function differently, and how their provenance paths and features of the software brought them to their conclusions.

The particular numerical provenance notation in our case is the series of model vectors that participants encountered during the model-steering operations using Doc-Function. Recall that for each time a Doc-Function user completes a model update, we learn a weight vector $\Theta \in \mathbb{R}^M$ explaining the importance of each of the documents to the user's current understanding (see Section 5.2.3). Each model corresponds to some time t and some user u, and is denoted Θ_u^t. Thus a user u has a model trail through model space that is the sequence $\{\Theta_u^1, \ldots, \Theta_u^t, \ldots, \Theta_u^T\}$ based on his or her interactions. In similar fashion to Mao et al. [MDL07], we take advantage of the real-valued vector nature of this model sequence and create a projection of the models themselves. Specifically, we calculate pairwise Euclidean distances between all the model vectors across all users and all time steps and apply multidimensional scaling (MDS) to build a spatial layout of the models themselves, showing all their relationships to each other. Figure 5.5 (a) illustrates how the models are related to each other in a projection.

5.6.2 Exploring Model Space with ModelSpace

Figure 5.3 shows ModelSpace, the interactive visualization we created for visualizing the type of numerical provenance data produced by the Doc-Function prototype. The dots represent models, i.e. vectors Θ_u^t, achieved by some participant at some point in the analysis task. The lines connect the models and represent the time in between model updates. User identity is encoded with color. All the participants[1] start with the same unweighted model, so there is a line in each color emanating from the middle. In Figure 5.3, the view is zoomed-in, and so some of the lines connect to dots that are outside the viewing window. The small arrows on the lines indicate the ordering of the models. ModelSpace shows undo and reset with a curved line pointing from the model the participant saw when requesting the undo or reset back to a previous model, either one step back or to the beginning,

[1]Note that two participants' data was excluded from the visualization because they had been forced to take a break during the task and this compromises the timestamp information used to build the visualization.

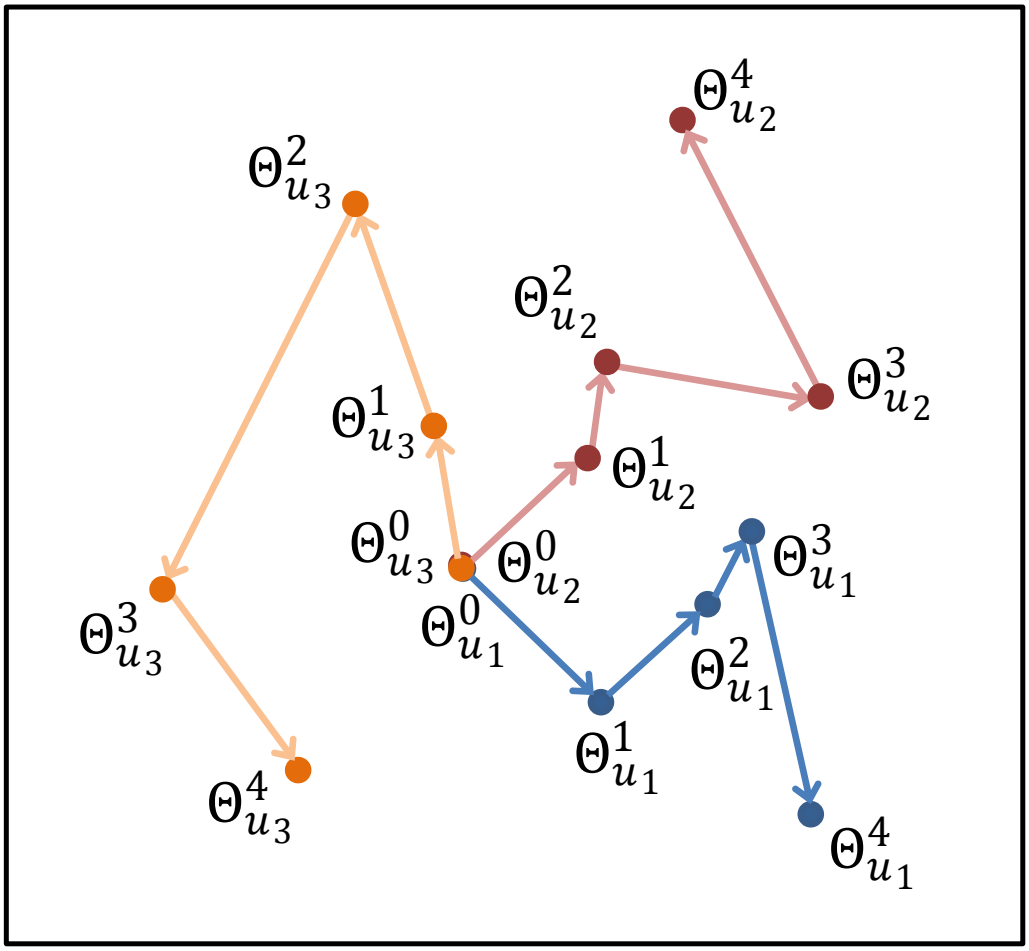
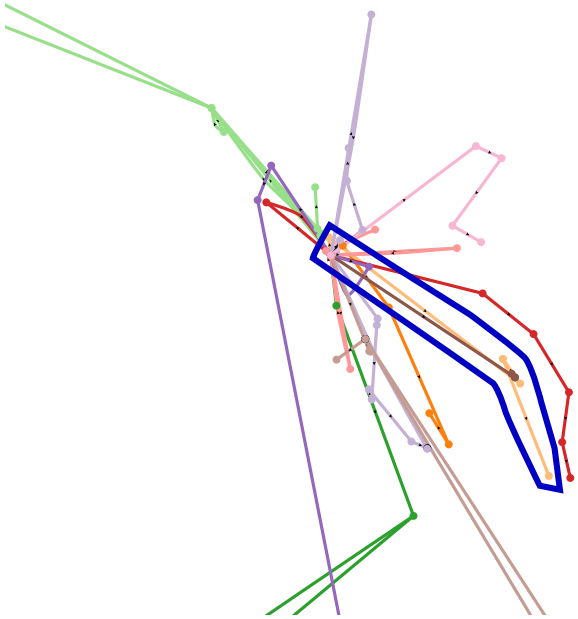
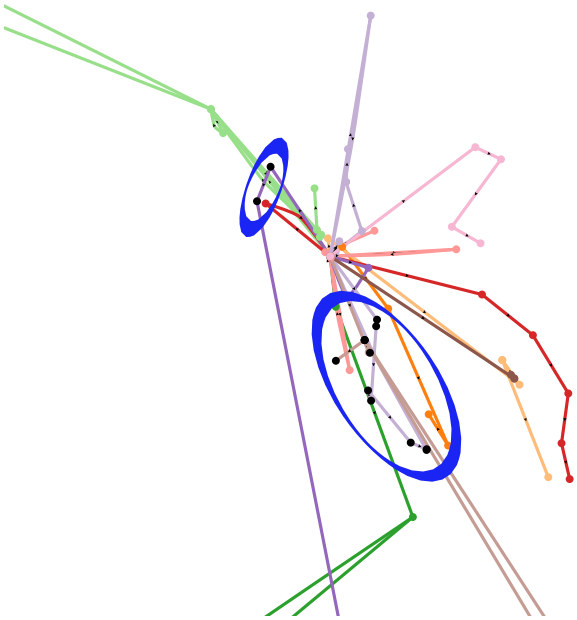


Figure 5.4: This figure illustrates how the series of models created by a user's path of exploration can be written as a series of functions that can be represented by vectors and thus projected into two-dimensional space for examination. Each dot represents a vector Θ^t_u which specifies the internal model of Doc-Function for one user, u, at one timestep, t, of the analysis, i.e. one model the user built with feedback in Doc-Function.



(a) Highlighting User 9, Document 41



(b) Highlighting Document 41

Figure 5.5: Subfigures (a) and (b) show views of ModelSpace for our numeric provenance data. The dots represent models, i.e. some Θ_u^t achieved by some participant at some point in the analysis task. The specific user is encoded by the color. Each line connects two dots and represents the time between the two models. Arrows on the lines indicate the ordering of the models. In (b), we have indicated in blue the trails through model space of the two professional analysts. Note they are quite similar. In (c), we show a search for "*doc41*". Models in which Document 41 is one of the top ten most important documents (i.e. has one of the highest weights) are thus highlighted in black. Note that Document 41 is only important in a couple regions of the ModelSpace, illustrating how the technique can help find such regions.

respectively.

The visualization shows more than just the spatial relationships and the sequence of events. The information used to create it includes the full logs from Doc-Function, which contain the keywords included in each model update, all searches conducted, and every instance of the participant placing the mouse cursor over a keyword (as a proxy for what documents were seen). In addition, the user study administrator of the user study took careful notes about what insights the participants discovered a what times, including when they discussed certain people and places and when they took advantage of the spatialization. These observations are digitized and collated by timestamp. They are coordinated with the logs so that ModelSpace can provide a fuller picture of the participants' analysis processes and show not only what models they created, but how they forged their paths to these models.

The information from the administrator's observations is associated with the lines, telling the story of what participants did between model updates. While mouse-over text for the dots indicates the ten most important documents for the corresponding model, the mouse-over text for a line shows everything that happened during the interval between the two dots the line connects. Events between model updates include information from the logs (keywords moused-over, searches conducted, keywords in applied mode updates) and observations from the administrator's notes (insights discovered). With all this information compiled, coordinated based on timestamps, and associated with lines connecting models in the visualization, ModelSpace can provide a detailed impression of what a given paticipant was doing that lead to different models.

Two additional features to this exploratory visualization facilitate finding the patterns we discuss in the next section. First, a search feature that highlights all lines and dots containing the search text with black makes it easy to examine where in the model space different documents were important and when participants had certain insights or read certain documents. Second, a token set intersection feature shows a list of all the tokens in common among the most recent dots and lines clicked. The list resets with a *clear* button and then continues to update the intersection as items are clicked. Using this feature, we can see what keywords several models have in common just by clicking the dots in sequence.

5.6.3 ModelSpace Insights

ModelSpace provides ample opportunities for exploration of the interaction data. It becomes clear immediately from this view that while all the users start in the same place, they diverge in different directions, with some directions more similar than others. This is difficult to ascertain without such a numerical provenance representation and accompanying visualization.

The data are not large enough to provide significant statistical results about how different types of participant differed (e.g. interns vs. professionals). However, with exploration, interesting patterns emerge. First, the two professional analysts had remarkably similar paths, as seen in Figure 5.5 (b). Of course, two paths are not enough to prove a pattern, but we can see potential for this analysis technique to unveil such similarities.

Following each participant's trail using the ModelSpace feature that shows documents in common between models, (see the token intersection feature explained in Section 5.6.2), we can see what the models along the individual trails have in common. In particular, we see the intersection of the sets of documents that were most important at each model. We found that for all participants included in ModelSpace, in each participant's final trail, i.e. the one that ended with the close of the evaluation as opposed to with an undo or reset, there is at least one document in common along the whole trail. That means that once a participant went down a path of analysis, some elements of the participant's initial model update carried through the rest of the process. This could be a visual manifestation of the inertia inherent in the learning mechanism. Whether it is intentional on the part of the participants or not, we clearly see through ModelSpace that without using undo or reset, participants do not escape the effects of their initial inquiries.

For a further example of how ModelSpace reveals patterns in our provenance data, consider Figure 5.5 (c). In this figure, a search has been performed for "*doc41*", the name of a document. The black dots, circled in blue, highlight the models in which Document 41 was weighted as one of the most important. The figure shows how the visualization can reveal that models where Document 41 was important appear in a limited number of areas in this spatial layout of the model space.

We believe this numerical provenance opens up new avenues for using visualization to explore users' interaction patterns as model-steering gains a place in common applications. Unlike traditional visualizations of user's interaction logs, the use of ModelSpace allows immediate comparison of the analysis trails between multiple participants. Although a thorough examination of the benefits and limits of this approach is out of the scope of this work, we believe that the use of this broader technique for visualizing and analyzing users' analytic provenance can lead to new techniques and evaluation methods. In section 6.9 we will further discuss the potential implications of evaluation and analysis using this approach.

5.7 Discussion

One of the grounding principles of the work presented in this chapter is the concept that enabling interactions on spatializations of keywords is an effective approach for sensemaking tasks of text corpora. Our user study showed the utility of this approach, and also raised interesting and important questions that can further extend our knowledge of the role (or science of) user interaction for visual analytics [PSCO09, YKSJ07]. Additionally, the integrated technique for visualizing and evaluating the implicit model steering in our technique allows for a novel method of understanding how the system adapts over the course of the investigation. These points are discussed in the subsections below.

5.7.1 Document Weighting and Query Refinement

The user feedback and interaction on the keyword positioning enables our technique to solve for weights on documents. Such keyword weighting mechanisms are also used in the information retrieval and query refinement. For example, Ruotsalo et al. [RJMK14] show how providing user feedback on terms via relevance feedback techniques [Roc71] enables query refinement based on the user's interest. Similarly, the work presented in this chapter makes use of the document weighting to compute updated spatial layouts of the keywords. However, the ability to refine and filter the document set based on the weighting could lead to a more scalable approach for visual text analytics. In theory, the amount of words in the

English dictionary has a lower upper limit than the number of unique documents that one can create with those words. As such, the keyword-centric approach to model-steering and visual analytics may provide a scalable approach to large-scale document analysis.

Additionally, prior studies have shown that analysts reason about text data during sensemaking tasks using concepts and insights that are typically single words [EFN12a]. However, these studies further showed that the spatial constructs created by users throughout their sensemaking tasks tend to be described by them using multiple words or short phrases. Together, these findings suggest that our keyword-centric approach that makes use of compound words and phrases can better support the users' analyses as they better reflect the users' insights and hypotheses about a topic or a thread of investigation.

5.7.2 Does Insight Result from the Visual Analytic Process, or a Single Visual?

Studying and developing the science of user interaction for visual analytics is becoming increasingly important. There has been more literature recently that discuss the potential ways to describe insight in the context of visual data exploration [CZGR09]. In general, two schools of thought include that insight is either the *process*, or the *product* of the analysis [NCE+11]. The method of visualizing and analyzing the analytic provenance in this chapter shows the model evolution over time. Each user interaction perturbs the model, and this update creates a new spatial representation of the information. Thus, we can observe more directly if the paths or sequences of these states (caused by user interactions) lead to similar insights, or if any specific points or regions in the visualization attribute to similar insights or findings.

In our analysis of the users' analytic provenance using ModelSpace, we discovered evidence that support both the *process* and the *product* arguments. As noted in Section 5.6.3, we observed that two expert analysts trails coincide with each other, suggesting that the *process* itself is important. However, at the same time, we also discovered certain documents that appear near each other in the spatialization, thus indicating that certain *products* are particularly relevant to the users' understanding. While we unfortunately are

not able to resolve this debate in this study, the successful use of the ModelSpace approach indicates that numerical provenance can be a rich direction for future research in analytic provenance and the evaluation of visual analytics systems.

5.8 Future Work

As interactive model steering continues to become a more popular method for integrating user feedback into visual analytic systems, the challenge of evaluating these techniques will become more critical. It is understood that researchers in this area see the potential for increasing automation and computation. However, there is the realization and open challenge that maintaining user control is also critical [ENCZ, KAF+08]. At a more holistic level, studies have tested the utility for successfully performing sensemaking tasks (e.g., [EFN12b, BLBC12]).

While the Doc-Function system and its evaluation adds to our understanding of how keyword-centric model steering can be used in analyzing a text corpus, it remains difficult to objectively determine how the Doc-Function system effectively integrates user-driven and automated analyses. Our ModelSpace approach has shed some light on how analysts, with the help of automated analysis, explore keywords and documents to discover plots and patterns in the corpus. However, even with the use of ModelSpace, it is still difficult to discern which of the analysts were more effective and why. Traditional evaluation methods based on task completion and performance metrics and the more modern approaches based on user-reported insights are both insufficient in determining the effectiveness of a series of a user's interactions and analysis steps. As the the visual analytics community continues to grapple with balancing user-driven analysis with automated techniques, it becomes increasingly important for the community to develop additional methods and metrics for evaluating how these model steering techniques support and improve sensemaking.

5.9 Summary

In this chapter, we presented a prototype visual analytics tool, Doc-Function, for interactive model-steering and analysis of document corpora. Unlike traditional model-steering methods that require the users to directly manipulate the text documents in a spatialization, the Doc-Function system takes a keyword-centric approach in which the user interacts with the (compound) keywords extracted from these documents. We evaluated the Doc-Function system with participants who range from having little or no analysis training to professional analysts and found that almost all participants were able to successfully use the system to discover topics within the given document corpus. In addition, with the use of the keyword-centric approach, we were able to capture the participants' interaction trails as numerical models. When viewed with a visualization, these interaction trails reveal interesting patterns about each participant's analysis pattern and allow us to further investigate the process and products generated by these participants during their investigations.

Chapter 6

Finding Waldo: Learning about Users from their Interactions

This chapter is based on the paper:

Eli T. Brown, Alvitta Ottley, Helen Zhao, Quan Lin, Richard Souvenir, Alex Endert, Remco Chang. Finding Waldo: Learning about Users from their Interactions. *Transactions on Visualization and Computer Graphics (TVCG)*, 2014. (Presented at VAST 2014)

6.1 Introduction

With ModelSpace in Chapter 5, we showed a technique for analyzing users' analytic trails through the space of possible machine learning models in a text analysis task. In doing so, we saw a hint of the potential for using computational methods to examine numerical representations of user activity. However, ModelSpace makes use of the numerical representation to create an interactive visualization of the users' provenance. A human must still be involved to gain knowledge from the other users' interaction trails. In considering what a computer-only approach can do with such numerical provenance data, we transition to the second prong of this dissertation as described in the Introduction (Chapter 1) with Figure 1.1. We turn the model-learning around and use the user's interaction data to learn about the *user* as opposed to the data. We work towards future systems that can learn about their users and adapt appropriately.

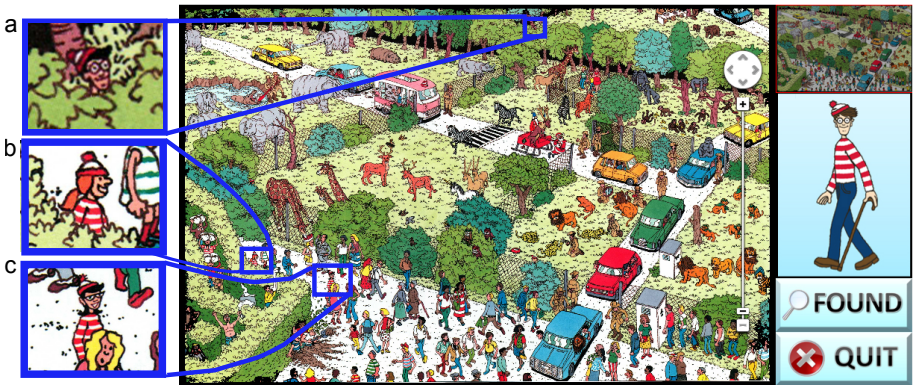


Figure 6.1: The interface from our user study in which participants found Waldo while we recorded their mouse interactions. Inset (a) shows Waldo himself, hidden among the trees near the top of the image. Distractors such as the ones shown in inset (b) and (c) help make the task difficult.

Since visual analytics fundamentally requires the close collaboration of human and computer [TC05], it is critical that we enable such partnerships by empowering the computer to understand the human more effectively. While the computer can communicate large amounts of information on screen via visualization, the human's input to an analytic system is still largely limited to mouse and keyboard [LIRC12]. This human-to-computer connection provides limited bandwidth [JLMP93] and no means for the human to express analytical needs and intentions, other than to explicitly request the computer to perform specific operations.

Researchers have demonstrated that although the mouse and keyboard appear to be limiting, a great deal of a user's analytical intent and strategy, reasoning processes, and even personal identity can be recovered from this interaction data. Machine learning researchers have recovered identity for re-authenticating specific users in real time using statistics over raw mouse interactions [Max03, PB04, RM06, Yu10] and keyboard inputs [LB99], but classified only identity, no user traits or strategies. In visual analytics, Dou et al. [DJS+09] have shown that strategies can be extracted from interaction logs alone, but at the cost of many hours of tedious labor. Unfortunately these manual methods are not feasible for real-time systems to adapt to users. The techniques needed to learn about users and their strategies and traits in real time do not exist to our knowledge.

In this chapter, we demonstrate on a small visual analytics subtask that it is indeed

possible to automatically extract high-level semantic information about users and their analysis processes. Specifically, by using well-known machine learning techniques, we show that we can: (1) predict a user's task performance, and (2) infer some user personality traits. Further (3), we establish that these results can be achieved quickly enough that they could be applied to real-time systems.

Our conclusions draw from an online experiment we conducted to simulate a challenging visual search task that one might encounter as a component of a visual analytics application with the game *Where's Waldo* (see Figure 6.1). The participants were given a visualization enabling a set of interactions (panning and zooming) to explore the image and find the character Waldo. During the participants' search process, we collect a wide range of information about their interactions, including the state of the visualization, and the time and location of all mouse events.

Inspired partly by our visualization of the user paths through the search image, we used this low-level interaction data to create three encodings that capture three major aspects of visual analytics: data, user and interface. The encodings are: (1) *state-based*, which captures the total state of the software based on what data (portion of the image) is showing, (2) *event-based*, which captures the user's actions through statistics of the raw mouse activity, and (3) *sequence-based*, which encodes sequences of clicks on the interface's buttons. The encoded information is then analyzed using well-known machine learning techniques such as support vector machines (SVM) and decision trees to classify groups of users with performance outcomes and individual differences.

The results of our analyses demonstrate that we can indeed automatically extract users' task performance, and infer aspects of their personality traits from interaction data alone. Further, task performance can be estimated quickly enough to be used in a real-time system. Depending on which data encoding with its corresponding machine learning algorithm, we attain between 62% and 83% accuracy at differentiating participants who completed the task quickly versus slowly, with state-based yielding up to 83%, event-based up to 79% accuracy, and sequence-based 79%.

With the goal of uncovering more intrinsic user factors, we applied the same techniques to classify participants on personality traits, and found promising signal. In partic-

ular, we can classify users based on three of their personality traits: locus of control, extraversion, and neuroticism with 61% to 67% accuracy. The correlation between these three personality traits and the participants' performance are consistent with previous findings in the visual analytics community on individual differences [GF10, OCZC13, ZOC+13].

Finally, on applying the techniques in real-time, we show that accurate prediction of the user's task performance and personality traits can be achieved after observing users for a limited time period. Using the same encoding and analysis techniques described above, we build classifiers based on a limited amount of the user's interaction logs. We demonstrate encouraging results for employing this technology in real-time systems, e.g. with only two minutes of observation on a task that requires an average of nearly eight minutes to complete, we can correctly classify the users with an average of 84% of the final accuracy.

Overall, our contributions to visual analytics are that we:

- Show that participants can be classified as fast or slow at the visual search task by applying machine learning to three encodings of participants' interaction data: (1) state-based, (2) event-based, and (3) sequence-based.
- Apply these same techniques to classify participants based on personality traits and demonstrate success for the traits locus of control, extraversion and neuroticism.
- Evaluate the plausibility of applying this work to real-time systems by providing results using shorter timespans of data collection.

6.2 Experiment

To investigate what interaction data encodes about users of a system, we sought a task that would simulate realistic tasks, and be difficult enough that people would have to think about how to solve it (engage strategies). Adopting a large visual search task satisfies our criteria: it is easy to explain to participants, but not easy to do, and it is a basic component of typical visual analytics tasks. Specifically, we chose *Where's Waldo* [Han87], a famous children's game consisting of illustration spreads in which children are asked to locate the character Waldo. Finding Waldo is not easy thanks to the size of the image, which is large enough to

require panning and zooming, and the fact that it is craftily drawn to provide a challenge. However, the target is known and there is a guarantee that the task is possible.

We performed an online experiment, collecting interaction data as our participants searched for Waldo in a large image (for our interface, see Figure 6.1). While Waldo is attired in a distinct red and white striped pattern (see Figure 6.1: his full image appears in the panel on the right and his placement in the full spread is shown in inset (a)), he is sometimes hidden behind objects, and the illustrations are filled with distractors specifically designed to mislead the user (e.g., scene elements covered in red and white stripes or the characters shown in Figure 6.1 insets (b) and (c)). To locate Waldo, users have to visually filter unimportant data, making him sometimes difficult to find. This difficulty is also analogous to real-life applications of visual search, where the target item may be partly occluded or obscured by objects of similar color, shape or size.

6.2.1 Task

In the main task, participants were presented with a Where's Waldo poster and were asked to navigate the image by clicking the interface's control bar (Figure 6.1). The control bar was designed to resemble Google Maps' interface and afforded six interactions: *zoom in*, *zoom out*, *pan left*, *pan right*, *pan up* and *pan down*. However, unlike Google Maps, our interface does not allow dragging, rather all actions occur through mouse clicks only.

The zoom levels for the interface range from 1 to 7 (level 1 being no zoom and level 7 being the highest magnification possible). The full image has resolution 5646 by 3607 pixels. At zoom level 1, the full image is shown. At zoom level k, the user sees proportion $1/k$ of the image. Panning moves the display by increments of $1/2k$ pixels.

The interface also includes two buttons not used for navigation: *Found* and *Quit*. When the target is found, the participant is instructed to first click *Found* then click on the target. The user must then confirm the submission on a pop-up alert. We require multiple clicks to indicate Waldo has been found to encourage participants to actively search for the target instead of repeatedly testing many random guesses. If the participant clicks *Found* but does not click on the correct location of Waldo, the click is logged, but nothing happens visually. Unless the participant quits the application, the experiment does not terminate

until Waldo is found correctly.

6.2.2 Data Collection

For our analysis, we recorded as much mouse activity as possible, including both mouse click and mouse move events. Mouse click events on interface buttons were logged with a record of the specific button pressed and a time stamp. Similarly, we recorded the interface coordinates of the mouse cursor and the timestamp for every mouse move event.

To establish labels for our machine learning analysis of performance outcomes and personality traits, we recorded both completion time and personality survey scores for each participant. Because researchers have shown [GF10, OCZC13, ZOC+13] that the personality factors locus of control (LOC), a measure of perceived control over external events, and neuroticism and extraversion are correlated with performance on complex visualization tasks, the survey was chosen to collect those traits. Specifically, we use a twenty-seven-question survey which includes the Locus of Control (LOC) Inventory (five questions) [GJE+06] and the Big Five Personality Inventory (twenty questions) [DOBL06] intermingled. The survey also includes two attention checks which require participants to give an obvious and precise response. These were used to filter participants who did not pay attention while completing the survey.

6.2.3 Participants

We recruited online unpaid volunteers, totaling 118 who successfully completed the task by finding Waldo, of whom 90 successfully completed a personality survey and passed an attention check. Women comprised 39 percent, and men 61 percent. Each participant used his or her own computer and completed the task via the Internet. They were required to have basic computer skills and to have never seen the poster in the experiment before. The participants had a median education level of a master's degree. Ages range from 18 to 45 ($\mu = 24$ and $\sigma = 2.8$). Average task completion time was 469.5 seconds ($\sigma = 351.9$).

6.2.4 Procedure

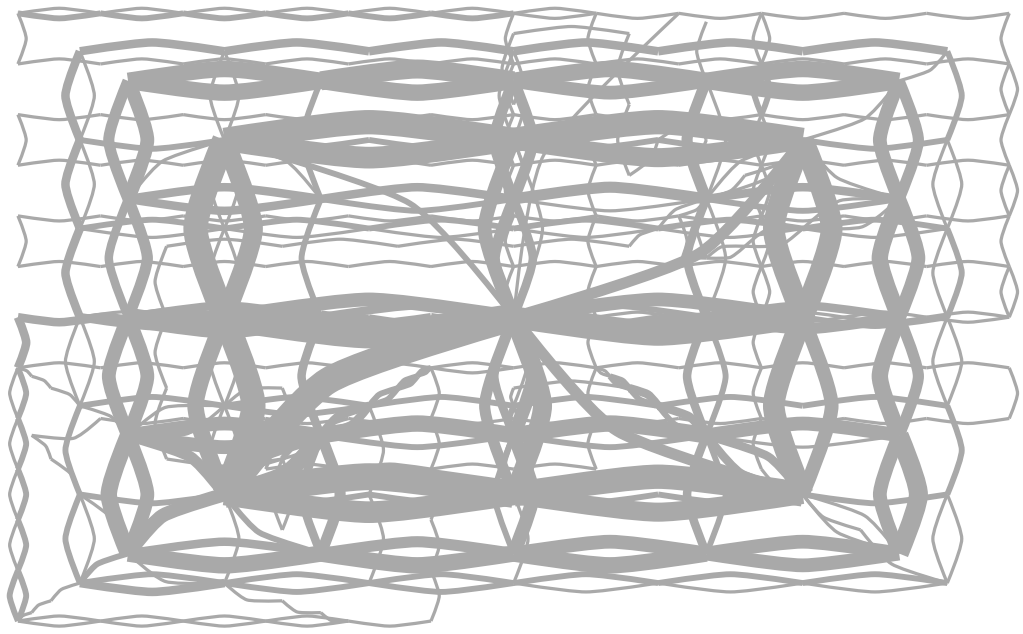
Participants were first asked to complete the personality surveys by rating a series of Likert scale questions on a scale of 1 (strongly disagree) to 5 (strongly agree). Next, participants read the instructions for the main portion of the experiment and were shown the main interface (Figure 6.1). They were instructed to manipulate the image by using the six buttons on the control bar to find Waldo using as much time as needed and told their completion time would be recorded. Once they had successfully found the target, they completed a basic demographic survey.

6.3 Hypotheses

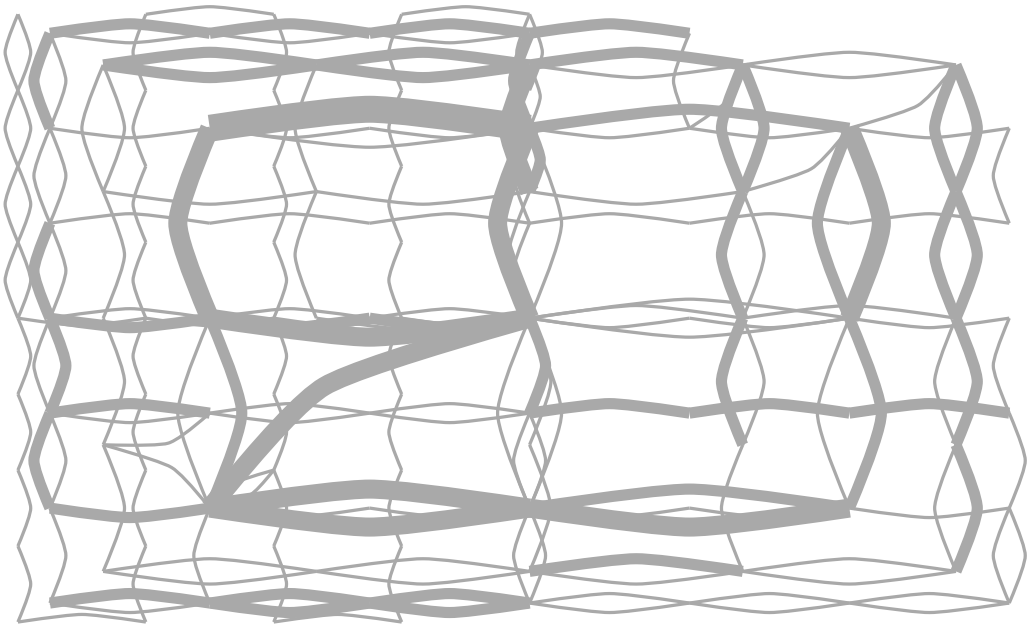
We collected data at the lowest possible level to ensure that we captured as much information about the participants' analysis process as possible. Over the next four sections we discuss how we first visualize this data, then create encodings to capture different aspects of the participants' interactions based on three core aspects of visual analytics: data, user, and interface. Specifically we encode (1) the portion of the data being displayed, as high-level changes in program state, (2) low-level user interactions, in the form of complete mouse-event data, and (3) interface-level interactions, as sequences of button clicks on the interface's controls. We analyze our data with these encodings with machine learning to evaluate the following hypotheses. First, we hypothesize that participants who are quick at completing the task employ different search strategies from those who are slow, and that these differences are encoded in a recoverable way in the interactions; second, that we can analytically differentiate users' personality traits based on interactions; and third, that these differentiations can be detected without collecting data for the entire timespan of the task, but instead can be found using a fraction of the observation time.

6.4 Visualizing User Interactions

To explore our hypothesis that we can detect strategies employed by different groups of participants, we first visualize their interactions. Figures 6.2 and 6.3 show example visu-

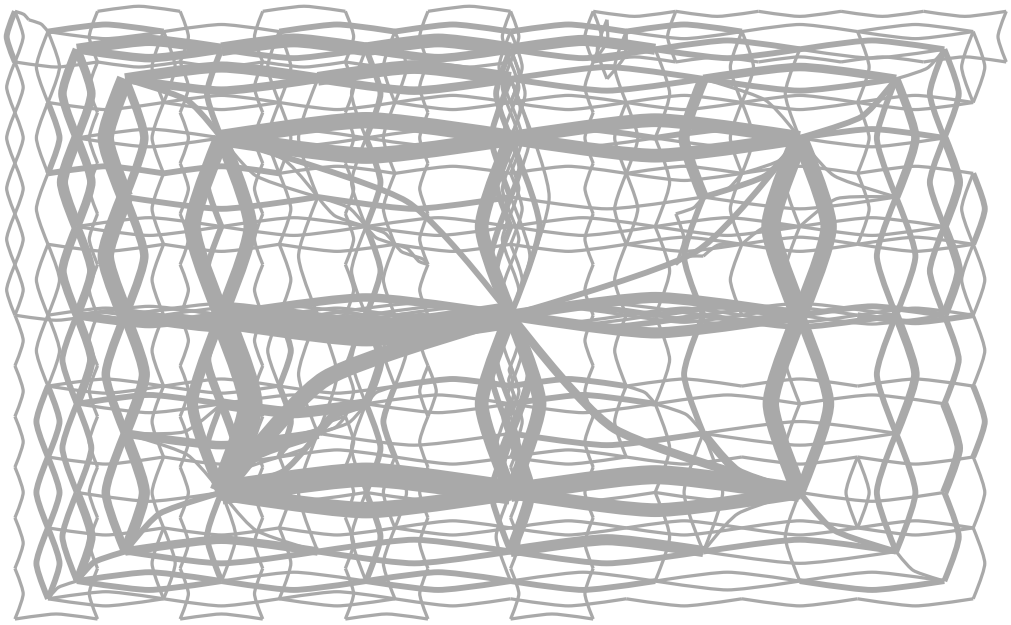


(a) Slow

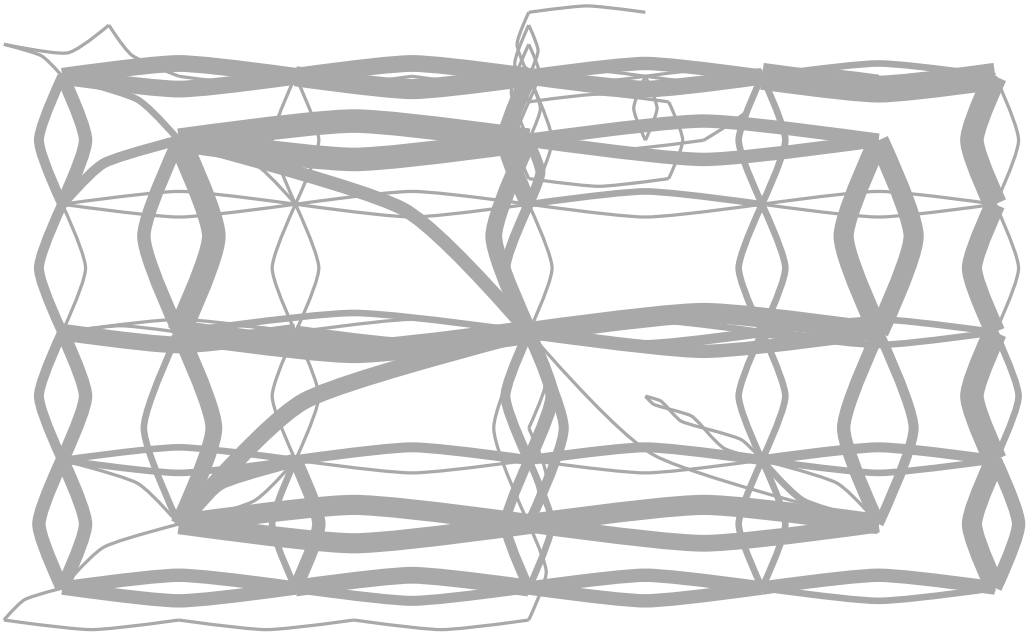


(b) Fast

Figure 6.2: Visualizations of transitions between viewpoints seen by participants during the study (see Section 6.4). Subfigures (a) and (b) show slow and fast users respectively, as determined by the mean_nomed splitting method (see Section 6.5).



(a) External LOC



(b) Internal LOC

Figure 6.3: Visualizations of transitions between viewpoints seen by participants during the study (see Section 6.4). Subfigures (a) and (b) are split with the mean_nomed method (see Section 6.5) based on locus of control, a personality measure of a person's perceived control over external events on a scale from externally controlled to internally controlled.

alizations of user movement around the Waldo image. The area of the visualization maps to the Waldo image. Each elbow-shaped line segment represents a transition from one user view of the image to another, i.e. from a view centered on one end-point of the line to the other. Where these lines intersect with common end-points are viewpoints of the image experienced by the participant while panning and zooming. The lines are bowed (elbow shaped) to show the direction of movement from one viewpoint to the next. Lines curving below their endpoints indicate movement toward the left, and those curving above indicate movement to the right. Bowing to the right of the viewpoints indicates movement toward the bottom, and bowing left indicates movement toward the top.

Zoom levels of viewpoints are not explicitly encoded, but the set of possible center points is determined by the zoom level. High zoom levels mean center points are closer together, so shorter-length lines in the visualization indicate the user was exploring while zoomed in. Note that diagonal movement through the Waldo image is not possible directly with the participants' controls. Instead, diagonal lines in the visualization are created because of zooming, i.e. when zooming out requires a shift in the center point.

This visualization can be used to show the movements made by a whole group of users by counting, for each flow line, the number of users who made the transition between the two corresponding viewpoints in the correct direction. In our examples, we are showing such aggregations for four different groups of users. In each case, the thickness of the lines encodes how many users in the group made that transition.

The two sub-figures of Figure 6.2 compare users who were fast versus slow at completing the task. Users were considered fast if their completion time was more than one standard deviation lower than the mean completion time, and correspondingly considered slow with completion times more than one standard deviation above the mean (for further explanation see Section 6.5). Users who were slow produce a finer-grain set of lines, indicating they made more small movements through the image using a higher zoom level and saw more of the Waldo image in higher detail. Further, the extra lines in the lower left of Figure 6.2 (a) as compared to Figure 6.2 (b) suggest that these slower participants were led astray by the distractors in the image, e.g. the people wearing similar clothing to Waldo seen in Figure 6.1, insets (b) and (c).

Evidence of different strategies is also salient when visualizing results based on some personality factors. The personality trait locus of control (LOC) has been shown to affect interaction with visualization systems [GF10, OCZC13, ZOC+13]. Figures 6.3 (a) and (b) visualize differences between participants with external (low) versus internal (high) LOC. In these subfigures, we see that the external group zoomed in much further on average, while the internal group performed more like the fast group and was able to find Waldo with a smaller set of viewpoints.

These observations are readily seen through these visualizations, but cannot be seen from inspection of the data, nor from machine learning results. Encouragingly, these visualizations hint that there are patterns to uncover in the data. The rest of this work explains our analytical results in extracting them automatically with machine learning.

6.5 Completion Time Findings

In Section 6.4, we presented visual evidence that our collected interaction data encodes differences between groups of participants. However, being able to tell fast users from slow is more useful if it can be done automatically. In this section, we delve into the data with analytical methods, using machine learning to build predictors of task performance outcomes. In particular, we adopt two common machine learning algorithms, decision trees [Mit97], which learn hierarchical sets of rules for differentiating data, and support vector machines (SVMs) [HCL10], which learn hyperplanes that separate data points of different classes in the data space. We apply these, to three representations of the interaction data, created to capture different aspects of how users interacted with the system.

Specifically, we tested three categories of representations of the participants' interactions, corresponding to some core aspects of visual analytics (data, user, and interface): the views of the image data participants encountered during their task (state-based), their low-level mouse events (event-based), and their clicks on interface controls (sequence-based). In this section we briefly explain how we derive the target participant groups used for our machine learning results, then show, for each representation of the data, our results at predicting if a given user would be fast or slow in completing the task.

Table 6.1: Completion Time Classifiers - results for state space, edge space and mouse events were achieved using support vector machines. The n-gram space results use decision trees. These results were calculated using leave-one-out cross validation.

Data Representation	Class Split	Accuracy (%)
state space	mean_nomed	83
	mean	79
edge space	mean_nomed	83
	mean	63
mouse events	mean_nomed	79
	mean	62
n-gram space	mean_nomed	79
	mean	77

We establish two different methods for labelling our participants based on the collected data. Our analyses aim to classify participants into discrete classes, fast and slow, but our recorded data includes only each participant's actual completion time. The first discretization method is to apply the mean completion time (469.5 seconds) as a splitting point: participants with a completion time lower than the mean are assigned to the 'fast' group, and higher to 'slow'. Participants with scores exactly equal to the mean are excluded from the data. In our results, this splitting method is indicated as *mean*. In the second method, we assume that participants whose scores are within one standard deviation of the mean have 'average' performance and we exclude them from the study, labelling the rest as above. We refer to this approach as the 'no-medium' splitting method, indicated in results tables as *mean_nomed*. The no-medium method allows us to see that stronger patterns emerge for participants with more extreme performance.

6.5.1 State-Based Analysis

In the visualization of participants' movement through the Waldo image (see Section 6.4), differences across groups of participants in how they examine the data become salient. This discovery would be more broadly applicable if the differences could be determined automatically. We create two data representations emulating these visual forms to search for patterns that differentiate users based on what parts of the image they chose to look at. In the "state space" encoding, we capture the portion of the data viewed as each participant navigated the Waldo image. In the "edge space" encoding, we capture transitions partici-

pants made between viewpoints of the image. Applying support vector machines (SVM) yields high-accuracy classifiers of completion time with both representations.

The state space encoding can be represented by a vector space. We consider the set $s \in S$ of all visual states (given by view position and zoom) that were observed by any user during completing the task. We create a set of vectors u_i, one representing each user, such that $u_i = (count_i(s_1), count_i(s_2), \ldots, count_i(s_{|S|}))$, where $count_i(s_j)$ indicates the number of times user i landed on state j. For the data from the Waldo task, this process yields a vector space in 364 dimensions.

A similar vector space expresses the transitions between viewpoints of the visualization, encoding how participants moved the viewpoint around the image in their search for Waldo. Their strategies may be encapsulated by how they directed the view during their search. In this vector space, the set $t \in T$ consists of all transitions made between any viewpoints by any participant while completing the task. If each viewpoint is represented by the location of its center, x, then $T = \{(k,m)\}$ where any participant made the transition $x_k \rightarrow x_m$ from position x_k to position x_m while searching for Waldo. Each individual user's vector is constructed as $v_i = (count_i(t_1), count_i(t_2), \ldots, count_i(t_{|T|})$, where $count_i(t_j)$ indicates the number of times user i made transition t_j. The dimensionality of our derived transition-based vector space (edge space) is 1134. The zoom levels are not explicitly encoded, but the set of possible center points is determined by the zoom level. This feature space is most closely related to the visualization described in Section 6.4 and seen in Figure 6.2.

The calculated vectors are used as a set of data features for input to an SVM [Vap98], a widely-applied machine learning method that works on vector space data. SVMs are both powerful and generic, and work by discovering an optimal hyperplane to separate the data by class. For this work we focus on results from the default implementation in the machine learning software package Weka [HFH$^+$09], which means a linear hyperplane, and slack parameter $c = 1$. This choice of an out-of-the-box classifier is intended to demonstrate that these results can be achieved in a straightforward manner.

Table 6.1 shows the accuracy of our completion time predictions, calculated via leave-one-out cross validation. Both state and edge space provide strong completion-time

prediction results, with maximum accuracies of 83%. However, these classifiers can only take into account high-level changes in the software, as opposed to the lower-level physical actions that may characterize different participants, which leads us to investigate different encodings for further analyses.

6.5.2 Event-Based Analysis

Users move their mouse throughout the process of working with a visual analytic system. Sometimes they move the mouse purposefully, e.g. to click on a control, other times they hover over regions of interest, and sometimes they make idle movements. Where the state and edge space encodings fail to make use of this information, the event-based data encoding described in this section derives from the most raw interaction information available to capture innate behavioral differences.

Previous machine learning work has shown that mouse event data contains enough information to re-authenticate users for security purposes [Max03, PB04, RM06, Yu10]. We adapted the data representation of Pusara et al. [PB04] for our interaction data by calculating their set of statistics over event information. Because we are predicting completion time, we removed any statistics that we found to be correlated with completion time. Table 6.2 shows the set of functions we used to encapsulate the participants' mouse movements and raw clicks. This set includes statistics on click information (number of clicks and time between clicks), raw button click information (percentage of clicks on a particular button, e.g., "% Left" refers to the percentage of button clicks on the "Pan Left" button), and derived mouse movement information (such as the number of moves, and the mean, standard deviation and third statistical moment of the distance and angle between them). The set does not include total counts of clicks on different buttons or the total number of mouse movement events, because those were strongly correlated with the total completion time. In total, we use twenty-seven features, listed across the two columns of Table 6.2.

As with the state-space representations, we apply SVMs to the mouse-event data. Table 6.1 shows the accuracy achieved with the mouse-event data using SVM classifiers, calculated using leave-one-out cross-validation. This approach manages a maximum score of 79%, which shows that there is strong signal in this low-level mouse data. The input fea-

Table 6.2: Features calculated for SVM analysis of mouse movement and raw mouse click data. μ, σ, and μ_3' refer to the mean, standard deviation, and third statistical moment. Pairwise indicates functions of pairs of consecutive events.

Click Event Features	Move Event Features
Clicks per second	Movements per second
Avg. time between clicks	Pairwise Euclidean distance (μ, σ, μ_3')
% Left, %Right	Pairwise x distance (μ, σ, μ_3')
% Up, % Down	Pairwise y distance (μ, σ, μ_3')
% Zoom in, % Zoom out	Pairwise speed (μ, σ, μ_3')
% Found, % Quit	Pairwise angle (μ, σ, μ_3')
% Clicks on Image	

tures may reflect subconscious mouse movement habits more than actual intended actions, so the results indicate that the differences between populations may be driven by innate differences in approach or cognitive traits. Even though none of the features is individually correlated with the completion time, these low-level interaction statistics taken together are enough to analytically separate fast from slow users.

6.5.3 Sequence-Based Analysis

The most direct representation of a user's process may be the sequence of direct interactions with software. Clicks are conscious actions that represent the user's intentions, and thus building classifiers based only on these semantically relevant interactions may provide more insight into why and how participants' analytical strategies differ. For our sequence-based analysis, we examine the sequences of button clicks used by participants to achieve the task of finding Waldo. We connect n-grams, a method from information retrieval for extracting short subsequences of words from collections of documents, to decision trees, a class of machine learning algorithms that produces human-readable classifiers.

6.5.3.1 N-Grams and Decision Trees

The n-gram method from information retrieval is intended for text, so an n-gram feature space must start with a string representation of data. We assign a unique symbol to each of the seven buttons in the interface: 'L' for pan left, 'R' for right, 'U' for up, 'D' for down, 'I' for zoom in, 'O' for out, and 'F' for declaring Waldo found. Each participant's

total interactions are thus given by an ordered string of symbols. We derive an n-gram vector space by considering each symbol a word, and each participant's sequence of words a document. Each dimension in the vector space then corresponds to one n-gram (i.e. one short sequence of user actions). Participants are represented by a vector of counts of the appearances of each n-gram in their interaction strings.

In our analyses we apply the J48 decision tree algorithm and NGramTokenizer from Weka [HFH+09] to classify participants based on task performance, and report accuracy scores from leave-one-out cross validation. The effectiveness of n-grams is sensitive to the choice of n. We empirically chose a combination of 2- and 3-grams as we found that to best balance accuracy and expressiveness of our eventual analytic output. Our results at classifying participants on completion time are shown in Table 6.1, revealing a top accuracy of 77% calculated with leave-one-out cross validation.

6.5.3.2 Decision Tree Interpretation

One advantage to using a decision tree with n-grams is that the resulting classifier is human-readable. Figure 6.4 shows the decision tree produced for the completion time data in n-gram space, using a mean split for classes. Each internal node shows a sequence of button clicks and the branches are labeled with the number of occurrences needed of that n-gram to take that branch. We can make several inferences about strategy from this tree. The root node indicates the strongest splitting criteria for the data. In this case, that node contains "L D", the n-gram corresponding to a participant clicking "Pan Left" then "Pan Down". If that sequence was clicked more than three times by anyone, that indicated the person would finish slowly. This makes sense because Waldo is in the upper right of the image. Moving in the wrong direction too many times can be expected to slow down progress at the task.

The "F U" and "D F R" nodes are also revealing. The "F" corresponds to telling the program that Waldo is found. These "F" button presses are not the last action, meaning they do not correspond to correctly finding Waldo. Instead, these sequences show participants' false guesses. Thus the tree suggests that participants who made several false guesses finished the task more slowly.

Finally, the "O O I" and "L O I" nodes correspond to behavior where the partici-

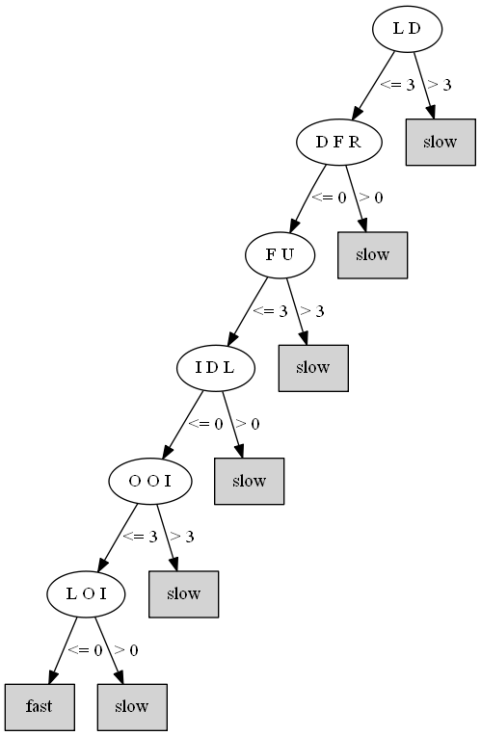


Figure 6.4: This is the decision tree generated as a classifier for fast versus slow completion time with mean class splitting. Each internal node represents an individual decision to be made about a data point. The text within the node is the n-gram used to make the choice, and the labels on the out-edges indicate how to make the choice based on the count for a given data point of the n-gram specified. Leaf nodes indicate that a decision is made and are marked with the decided class.

pant zoomed out and then back in again. The "O I" component could indicate participants zooming out to gain context before zooming in again. Alternatively, the same subsequence could indicate participants zooming out and immediately back in, wasting time.

The readability of this technique shows promise for identifying trends in strategies and behaviors. We cannot guarantee that these interpretations reflect the participants' actual intentions, but rather submit these as possible reasons for what is shown in the tree. The real power of using n-grams and decision trees on interaction sequences is that it makes this type of hypothesizing possible, leading to deeper investigation when it is beneficial to understand how people are solving a given task.

Table 6.3: Personality Classifiers - all of these results are with SVM except when using n-grams, which we pair only with decision trees

Data Representation	Class Split	Accuracy (%)
LOC		
n-gram	mean	67
Neuroticism		
mouse events	mean_nomed	62
edge space	mean_nomed	64
Extraversion		
edge space	mean	61

6.6 Personality Findings

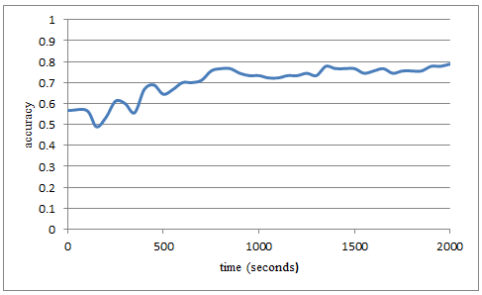
Prior research by Ziemkiewicz et al. [ZOC+13] and Green and Fisher [GF10] suggests that users will use a visualization system differently based on their personality traits. Motivated by these findings, we explore the efficacy of extracting personality traits from interactions. Specifically, we apply the same data encodings and machine learning algorithms used for the completion time analyses to predict users based on their personality traits.

Instead of classes derived from completion time, we separate users into *low* and *high* groups based on their scores on each personality inventory: locus of control, extraversion, agreeableness, conscientiousness, neuroticism and openness to experience. Consistent with our completion time analysis, we test both mean and mean_nomed splits (see Section 6.5). Table 6.3 summarizes our analysis results.

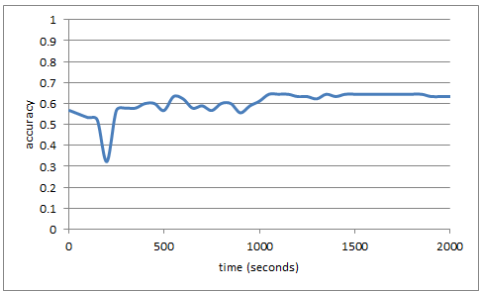
Across several techniques, we successfully classified users based on their LOC, neuroticism, and extraversion scores. Of the personality traits, our techniques were best with LOC, yielding classification accuracies as high as 67%. This supports the findings of Ziemkiewicz et al. [ZOC+13] that of the personality traits, LOC was the strongest predictor of users' performance on visualization search tasks. Consistent with our findings, prior work also found significant effects with neuroticism and extraversion [GF10, ZOC+13].

6.7 Limited Observation Time

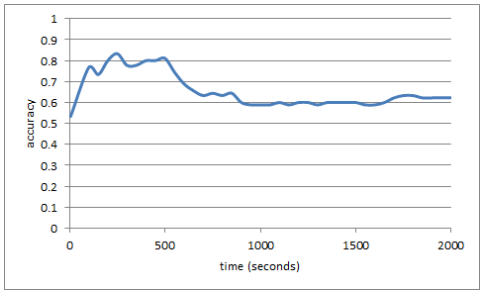
The participants in our study were given as much time as they needed to complete the Waldo task. So far, the presented results have taken advantage of the full timespan of the collected



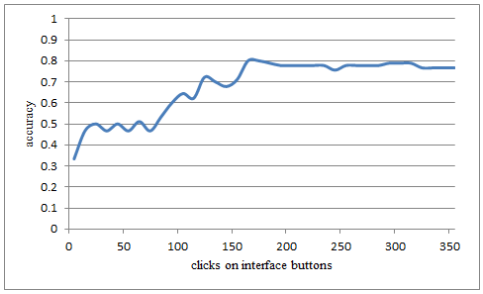
(a) State Based



(b) Edge Based



(c) Mouse Events



(d) Sequence Based

Figure 6.5: Graphs showing the ability to classify participants' completion time as a function of the extent of data collected. The x-axis represents the number of seconds of observation, or number of clicks for the sequence based data. The y-axis is the accuracy achieved after that amount of observation. Accuracy values are calculated with leave-one-out cross validation, and use the mean splitting method (see Section 6.5).

data from their interactions to classify them. Investigating the minimal timespan required for this type of analysis is crucial for potential real-time applications, so we evaluated our classifiers' performance as a function of the data collection time.

Figure 6.5 shows, for each of the different data representations, graphs of how task performance classification improves (on trend) with more observation time, i.e. more information available. Figure 6.6 shows one example of this behavior from personality trait classifiers. The x-axis is the amount of data collected, and the y-axis is the accuracy achieved by training and classifying with that amount of data. For all but the sequence-based analysis, the x-axis represents time. For the button click sequences, the x-axis is based on the number of clicks instead. Leave-one-out cross validation (LOOCV) and the mean-based class definition are used for all these results.

These graphs demonstrate two things. First, accuracy scores comparable to the final score can be achieved with much less than the maximum time. Note that close to

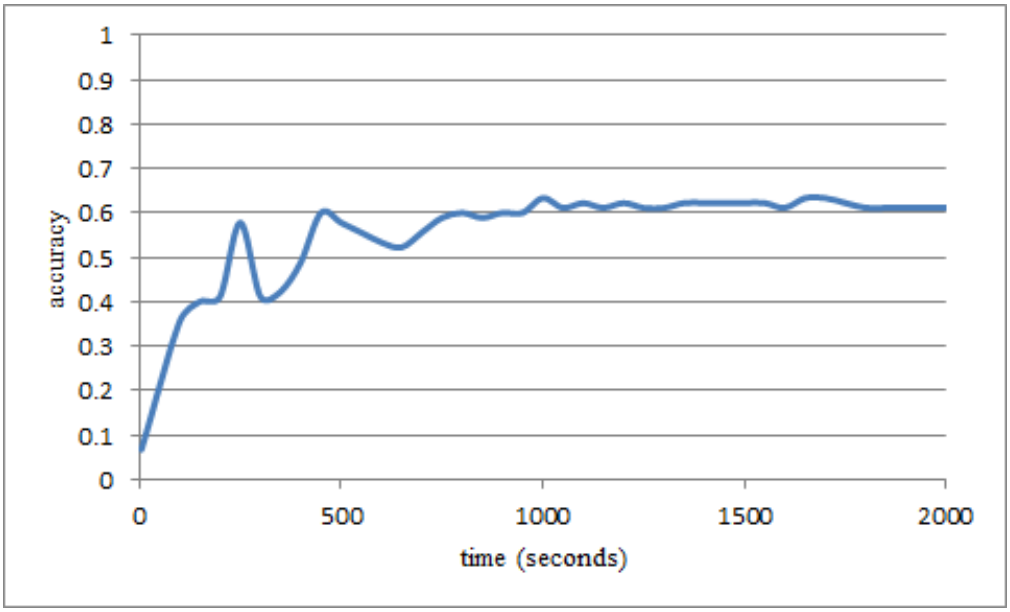


Figure 6.6: This graph shows the dependence of the ability to classify the personality trait extraversion on the amount of time the participants are observed. The x-axis represents the number of seconds of observation. The y-axis is the accuracy achieved after that amount of time. This example uses the edge space encoding and the mean splitting method (see Section 6.5). Accuracy values are calculated with leave-one-out cross validation.

the mean completion time, the encodings are achieving much of their eventual accuracy scores: state-based, 64% instead of its eventual 83%; edge-based, 60% compared to 63%; and sequence-based, 61% as opposed to 77%. These correspond to 77%, 95% and 79% of their final accuracy percentage scores, respectively.

Second, as expected, in most cases using more data allows for better results. In the case of the mouse event data, the accuracy peaks before reaching the average participant's finishing time, about 470 seconds.

6.8 Extended Results

In this work, we focused on machine learning results produced with off-the-shelf algorithms to emphasize that they could be re-applied in a straightforward way. However, in the course of our investigation, we applied a number of additional customizations to find the most accurate classifiers possible with our data representations. These extended results can be found in Appendix 6.11. In the appendix, we explain the additional methods we used and

show the results we achieved by customizing the classifiers. We show cases in which our tuning produced higher-accuracy classifiers, and revealed signal with feature spaces or class splitting criteria that otherwise could not encode certain traits.

6.9 Discussion and Future Work

In this work, we have shown, via three encodings, that interaction data can be used to predict performance for real-time systems, and to infer personality traits. Our performance predictions ranged in accuracy from 62% to 83%. On personality traits, we were able to predict locus of control, extraversion, and neuroticism with 61% up to 67% accuracy. Further, we found that with only two minutes of observation, i.e. a quarter of the average task completion time, we can correctly classify participants on performance at up to 95% of the final accuracy.

Given the above results, there are some fascinating implications and opportunities for future work. In this section, we discuss the choice of task and data representations, and how they may be generalized, differences between the personality results versus those for completion time, and future work.

6.9.1 The Waldo Task and Our Encodings

The Where's Waldo task was chosen because it is a generic visual search task. It is a simple example of an elementary sub-task that comes up often in visual analytics: looking for a needle in a haystack. The user can manipulate the view, in this case with simple controls, and employ strategy to meet a specific task objective. In this section we address how this experiment and our analyses may scale to other systems. Because our set of encodings is based on three core aspects of visual analytics, data, user, and interface, we frame the extensibility of our approach in terms of data and interface.

The data in our experiment is the image in which participants search for Waldo. At a data scale of twenty-megapixels, our state-based interaction encodings, which are closely tied to the data because they capture what parts of the image a participant sees, reach hundreds of features to over 1000 features. As the size of the data (image) increases, the state

space and edge space may not scale. However, the event-based and sequence-based encodings depend only on the interface, and thus could scale with larger image data.

Conversely, the interface in our experiment is a simple set of seven buttons. Increasing the complexity of the interface affects the event-based and sequence-based encodings. The mouse-event features include statistics about how often each button is pressed, and the sequence-based encoding requires a different symbol for each button. While these two encodings may not be able to scale to meet increased interface complexity, the state-based encoding is unaffected by the interface and thus could scale with the number of controls.

The three encodings we used in this chapter can all be extracted from the same interaction logs. Each one of them provides enough information to recover task performance efficiently. Because of their complementary relationships with the core concepts of interface and data, their strength, as an ensemble, at learning from interaction data is not strictly constrained by either interface or data complexity.

The scalability of the ensemble of encodings raises the possibility that our approach could be generalized to other visual search tasks and to complex visual analytics tasks. In particular, since users' interactions in visual analytics tasks have been shown to encode higher-level reasoning [DJS+09], we envision that our technique could be applied to other sub-tasks in visual analytics as well. Specifically, we consider the Waldo task as a form of the data search-and-filter task in the Pirolli and Card Sensemaking Loop [PC05]. We plan on extending our technique to analyzing user's interactions during other phases of the analytic process such as information foraging, evidence gathering and hypothesis generation.

6.9.2 Personality

Being able to demonstrate that there is signal in this interaction data that encodes personality factors is exciting. However, none of the results for personality factors are as strong as those for completion time. Not only are the overall accuracy scores lower, but we found that in examining the time-based scores (as in Section 6.7), for many personality factors, there was not a persistent trend that more data helped the machine learning (Figure 6.6 shows one of the stronger examples where there is a trend).

While the prediction accuracies are low, our results are consistent with prior find-

ings [SCC13] in the human-computer interaction community on individual differences research. Taken together, this suggests that although personality and cognitive traits can be recovered from users' interactions, the signals can be noisy and inconsistent. In order to better detect these signals, we plan to: (1) explore additional machine learning techniques, like boosting [Sch90] for leveraging multiple learners together, and (2) apply our techniques to examine interactions from more complex visual analytics tasks. We expect the latter to amplify results as Ziemkiewicz et al. [ZOC+13] have shown that personality trait effects are dampened when the task is simple. In their work, for complex inferential tasks, the effects were more pronounced and potentially easier to detect [ZOC+13].

6.9.3 Future Work

This work is a first step in learning about users live from their interactions, and leaves many exciting questions to be answered with further research. The ability to classify users is interesting on its own, but an adaptive system could test the feasibility of applying this type of results in real time. For example, since locus of control can affect how people interact with visual representations of data [ZOC+12], a system that could detect this personality trait could adapt by offering the visualization expected to be most effective for the individual user. Different cognitive traits may prove more fruitful for adaptation, but even completion time could be used to adapt, by giving new users advice if they start to follow strategies that would lead to their classification as slow.

Further, of the data representations we evaluated, only the mouse events, the lowest-level interactions, encode any information about time passing during the task. The other representations do not encode the time between states or button presses, but that information could be useful for a future study. For our sequence-based analysis, our approach was to pair n-grams with decision trees for readability, but there are plenty of existing treatments of sequence data that remain to be tried for this type of data classification on visual analytic tasks, including sequence alignment algorithms, and random process models, e.g. Markov models. Finally, in this work we focused on encoding one aspect of data or interface at a time, but combining feature spaces could be powerful. In fact, in experimenting with a feature space that leverages multiple types of encodings, we achieved 96% accuracy on

completion time with mean_nomed splitting[1].

The experimental task was a simple version of a basic visual analytics sub-task. Our results could be strengthened by expanding the experiment to test Waldo in different locations, or different stimuli like maps with buildings and cars. The breadth of applicability could be evaluated by testing other elementary visual analytics tasks such as using tables to find data or comparing values through visual forms.

Our plans to extend this work expand on three fronts: (1) evaluating additional personal traits, like cognitive factors such as working memory, to our analyses, (2) trying further machine learning algorithms and encodings to learn from more of the information being collected, like the times of the interactions and (3) extending experiments with different tasks including deeper exploration of visual search. We believe there are many opportunities to extend this work, both experimentally and analytically.

6.10 Summary

In this chapter, we presented results of an online experiment we conducted where we recorded participants' mouse interactions as they played the game Where's Waldo. We broke the users into groups by how long it took them to find Waldo (completion time) and their personality traits. Visualizing the participants views of the data, we showed that there are differences in strategies across groups of users. We then applied machine learning techniques, and demonstrated that we can accurately classify the participants based on their completion time using multiple representations of their interactions: visualization states, low-level mouse events, and sequences of interface button clicks. By examining artifacts of our machine learning work with these sequences, we were able to identify short subsequences of interactions that identify groups of users. These human-readable classifier results hint at user strategies across groups. We were also able to detect and classify the participants based on some personality factors: locus of control, extraversion, and neuroticism. Finally, we showed the dependence of the machine learning results on the observation time of the participants.

[1] Specifically, we tested a modified state space encoding where the zoom level information is replaced by an identifier of the button click that caused the state.

6.11 Appendix: Extended Results

Though we demonstrated that completion time and personality can be modeled from raw interactions can be done directly with off-the-shelf tools using default settings, further attention to detail can yield stronger classifiers. In this appendix, we discuss some additional results that we achieved by tuning the algorithms, including applying principal component analysis (PCA), and optimizing the parameters of support vector machines (SVMs).

Table 6.4: Additional SVM Results - all results are calculated using leave-one-out cross validation.

Data Representation	Class Split	Classifier	Accuracy (%)
Completion Time			
edge space	mean_nomed	SVM_{poly}	87
	mean	SVM_{poly}	72
mouse events	mean_nomed	SVM_{poly}	88
	mean	SVM_{poly}	82
LOC			
edge space	mean	SVM_{poly}	62
state space	mean_nomed	SVM_{poly}	63
state $space_{PCA}$	mean_nomed	SVM	63
Neuroticism			
edge space	mean_nomed	SVM_{poly}	68
state $space_{PCA}$	mean_nomed	SVM	68

The SVM algorithm is sensitive to a slack parameter [CV95] and to the choice of kernel. Common practice is to address this by using a parameter search to find the best parameter values [HCL10]. In the context of deploying the best possible classifier for a given dataset, that entails simply trying different choices of the parameter (or sets of parameters) and evaluating the classifiers until the best can be reported. Since our goal is instead to evaluate the classifiers and encodings themselves for this type of data, we take the approach of validating the algorithm of classifier+param-search. As usual for cross validation, the data is split into k folds. Each fold takes a turn as the test data, while the other folds are used for training, providing k samples of accuracy to be averaged for a total score. In testing a classifier+param-search algorithm, the algorithm being evaluated on one fold is one that chooses a parameter by testing which value produces the best classification result. To evaluate "best classification result", another (nested) cross validation is needed. The

original fold's training data is split into folds again and cross validation is used over those inner folds to pick the optimal parameter. Weka implements a more sophisticated version of this practice that allows optimizing two parameters at once (generally referred to as grid search) and uses optimizing heuristics to limit the number of evaluations [PHf14]. We have used this implementation to run a grid search that optimizes over (1) slack parameter and (2) degree of polynomial for kernel (1 or 2, i.e. linear or quadratic). In Table 6.4, this classifier is called SVM_{poly}. This table shows highlights of the results that we produced with this technique.

Another helpful factor in working with SVMs on high-dimensional data is principal component analysis. PCA projects the high-dimensional data into a lower-dimensional space defined by the eigenvectors of the original data. The number of eigenvectors is chosen to make sure that 95% of the variance in the data is accounted for in the low-dimensional approximation[2]. Applying PCA to the data space was particularly helpful in data representations like state space, which has a high degree of dimensionality. In Table 6.4, data representations with PCA applied are indicated by the subscript PCA.

Overall, the results in Table 6.4 show cases in which our tuning produced higher-accuracy classifiers, and revealed signal with feature spaces or class splitting criteria that otherwise could not encode certain traits. The completion time results for the edge space and mouse event feature spaces are improvements of up to 32%. Specifically with edge space encoding and mean split, SVM_{poly} offers 82% accuracy instead of 62% with off-the-shelf SVM. In our earlier analyses, we did not find sufficient signal to report on LOC with any state-based encodings, but using PCA or parameter search makes that possible. Through applying standard methods for tuning SVM, we gained higher accuracy over our existing results, and demonstrated connections between encodings and traits that were otherwise obscured.

[2]We used the Weka filter implementation with this option enabled.

Chapter 7

Discussion

The way forward in interactive systems that leverage machine learning behind-the-scenes will include innovation on both sides of the equation: (1) new interaction paradigms to best capture the semantics of user intent and (2) new machine learning algorithms and adaptations to make learning in this context plausible for more types of models. We will need to leverage information gathered from across a wide variety of tasks, from exploring data to reading email, and make use of a broad array of algorithms to model efficiently what we learn. More broadly, what is needed is a framework for what is required of a machine learning algorithm and what is required of a human interface in order for this paradigm to be efficacious. In particular, we must bound what features of a machine learning algorithm are necessary to be effective for learning in this interactive context, and what features of a human-computer interface are required to collect valid information for a machine learner effectively.

Such a framework, complete with examples of successes and the knowledge gained in building them, is future work. In this section, I will lay out the groundwork of this project: a discussion of machine learning for use in interactive systems. First, I provide a sketch of a framework for how such systems can be put together, including what types of user interaction can be supported. Second, I review the broad categories of machine learning, focusing on explaining how different types of algorithm are suited to the interactive context and fit into the framework. I add a discussion of some research areas within machine learning that have specific advantages for application to an interactive setting. Finally,

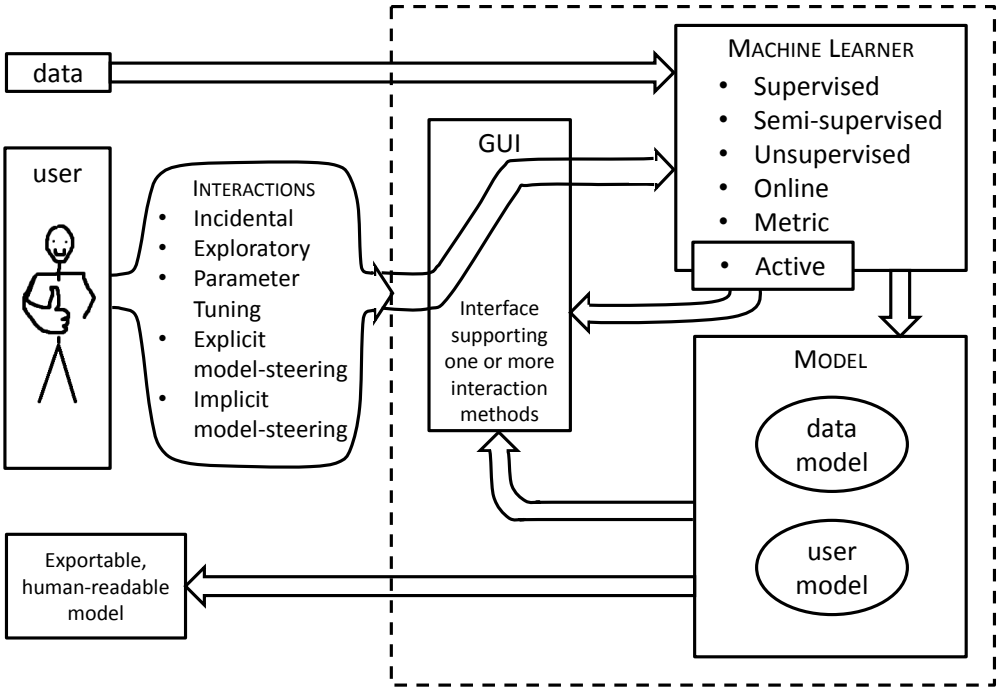


Figure 7.1: This framework diagram sketches a framework for interactive model-learning systems. It includes types of learning algorithm and types of interaction as described in Section 7.1.

I connect the framework and the machine learning techniques to produce a table showing how combinations of interaction types and machine learning algorithms can lead to a wide variety of systems and applications. This discussion will include examples of how machine learning has been and could be used in interactive systems in both HCI and visual analytics, as well as my thoughts from experience gained so far in conducting research on this type of system.

7.1 The Beginnings of a Framework

If the way forward in interactive systems is to have interaction between human and computer that maximizes the strengths of each, using machine learning back-ends to aide and amplify user efforts at analysis, then innovation is needed in both machine learning and HCI. In this section, I sketch a framework for such systems as groundwork for future progress. A full framework explaining all possible uses of machine learning and visualization or HCI together is beyond the scope of this dissertation. Instead, I will include some

thoughts about what is needed from a machine learning algorithm, and what is needed from a graphical user interface or visualization to make these systems effective. I begin with a diagram (Figure 7.1) that shows how the components of such a system fit together, what types of interactions mechanisms can be used, and what types of machine learning algorithm. In the rest of this section, I provide a discussion of the diagram, explain the types of interactions, and overview the machine learning concepts included in the diagram along with notes on their applications.

Figure 7.1 illustrates that all user interactions are affected through the visual interface, but the information is prepared and forwarded to the machine learning algorithm. As described in the rest of this chapter, there are a variety of types of algorithm to be used in this component. A special note is that active learning, which uses the model to help guide the user's analysis, has a direct line to affect the GUI without affecting the model. Otherwise, the learners take into account the data and feedback from the user to build models, and those models are used to update the interface. Note that there are two types of model: (1) models of the data constitute the analytical results and affect how the data is displayed as well as the final analytic product, and (2) models of users can be learned from their interactions and used to optimize the interface for that type of user. Once a set of interactions has been used to update the model(s), the visualization can be updated to reflect the new model information, and the user is thus presented with a new opportunity to explore the data and provide further feedback.

The framework shows several types of machine learning algorithm; in fact the machine learning community has produced myriad algorithms to choose from for creating integrated systems. While there is certainly room to invent new algorithms or adapt existing ones to better fit this purpose, we start with the premise that we can pick from among those available. The most important feature is that the algorithm be able to support fast updates when new information from the user is available. The update must happen quickly enough that the result can affect the visual interface without breaking the user's focus. In order to update quickly, some learners can support incremental updates. These algorithms can accept a single piece of feedback to update the model without recalculating everything. If the algorithm cannot perform an update for a single piece of feedback, the entire model

can be relearned as long as the process is fast enough. Incremental updates are critical to the exploratory, sensemaking and model-building process because in order to shield a user from reviewing all possible instances, the user must be able to supply feedback as it suits his or her process. Ideally, when the user is finished, the model can be exported as a product of the analysis and can itself be instructive. This is not a requirement for the learning process, but to get the best analytical results out of a human-computer partnership, the human must understand the result. Comprehensible machine learning includes a variety of model types like metric learners, which can produce weightings of data dimensions, and decision trees and rule learners [LRMM12]. It is, of course, necessary, as with choosing visualizations, to pick an algorithm that uses the same data type as the data at hand (e.g. nominal, numerical, etc.). Finally, it is important, as with any machine learning application, to try to choose a learner that builds a type of model that can handle the kinds of patterns expected in the actual data.

The requirements for the visualization component are more vague because the human-facing component can be flexible. Most important is that the interface or visualization reflects the model. If updates to the model cannot be seen, the feedback loop of the user's iterative improvement will be broken. Further, though the learner is responsible for the update time of the model, any calculations for updates to the visualization are part of the limited time-frame available for showing the user the result of feedback, and must be capable of interactive speed. Finally, the most flexible constraint, there must some mechanism in the interface for collecting information that can be useful to the machine learning back-end. There are many forms from raw interaction logs to specific buttons for labeling data points.

Overall, it is important that the visualization and machine learning be chosen together so that they match up on data types and they can communicate with each other. Figure 7.1 shows a sketch of how the components of and interactive learning system fit together. The data is input to the system through the learner, so it can be part of the model that gets shown to the user (certainly the data could also bypass the model as some part of the display, but we concentrate here on the interactive system flow). The user can observe the GUI and interacts with the system via certain types of interactions:

1. *Incidental* — interactions that are not part of the data exploration or model-steering, i.e. interactions with other parts of the interface. They can be used to learn models of the user.

2. *Exploratory* — interactions involved in exploring the data, but not those that carry a semantic meaning from the user to update the model explicitly or implicitly.

3. *Parameter Tuning* — interactions that change the parameters of a learning algorithm as opposed to providing model-steering feedback.

4. *Explicit model steering* — interactions in which the user is intentionally updating the data model.

5. *Implicit model steering* — interactions that are part of a primary task but used for model steering, as opposed to explicitly for steering the model (e.g. a spam filter).

These types of interactions encompass all the types seen throughout this dissertation and more. In Chapter 6 with *Finding Waldo*, we address the use of *incidental* and *exploratory* interactions. All of the interactions with the visual search task of the participants in our study were recorded and used to create models of the user but there was no analytical model steering, explicit nor implicit. *Parameter tuning* is a common usage of machine learning in visualization, and includes systems like iPCA [JZF+09a], where the principle component analysis algorithm is controlled through a visualization. During the intelligence analysis task of Chapter 5, participants using Doc-Function could move keywords around the canvas without affecting the model. The associated interaction data is considered *exploratory*. That application also involves *explicit model steering*, in the same fashion as Dis-Function from Chapter 3, when participants in the studies mark and move data points (Dis-Function) or keywords (Doc-Function) and then click an update button. The user understands that those changes are interpreted as feedback on the data model, and used to update it for another iteration. In contrast to those analytic systems, *implicit model steering* is used for applications where the machine learning back-end is assisting the user with a primary task. One classic example is spam filtering, in which a user cleaning his or her inbox by marking some emails as spam will also be helping to train an algorithm

that detects such emails automatically. The important difference between the explicit and implicit cases is subtle, but for the former, the user is engaged with the task of building the model, as opposed to providing feedback during some other primary task. The explicit case may be preferred for professional contexts, where the implicit one may be for end-user software. There is room for applications to contest this distinction one way or the other.

These five interaction types characterize how a user can interact with a visual analytics system that is integrated with machine learning techniques. However, much like interface and interaction design patterns and techniques, there is a wide range of machine learning algorithms. The appropriate pairing and combination of the interaction types and machine learning techniques can lead to effective collaborations between the human user and the computational system. In the section below, I will outline machine learning techniques and describe their potential use in visual analytics systems following my proposed framework.

7.2 Machine Learning in Broad Strokes

Machine learning is a massive field, growing in popularity, perhaps largely because its graduates are so sought after by companies looking to make the gains that their data collection undertakings promise. For a given dataset where a number of data points or *instances* are each represented by some *features* (other names include dimensions and variables), the goal of machine learning is to discover a pattern. These patterns broadly fall into two categories: a set of groups of similar instances (i.e. clusters), or a function that maps an instance to some target *label*. Though it is not usually explained this way, the top-level distinction between machine learning algorithms is actually how much human intervention is required. Algorithms that require the every single instance be provided with a label as part of the input are categorized as *supervised* learning. At the other end of the spectrum is *unsupervised* learning, in which no labels are required at all. Clustering algorithms fall into this latter category, and work to create structure where none is supplied. In-between, there is *semi-supervised* machine learning, which can take advantage of limited human intervention. Interactive systems can take advantage of all three of these branches.

7.2.1 Unsupervised Learning

Clustering is already a useful part of many systems, providing a mechanism for automatically grouping data for user consideration. Classic algorithms include k-means, expectation maximization (EM), hierarchical clustering, [JMF99] and DBSCAN [EKS+96]. Clustering can be used explicitly, e.g. showing a user groups of data points. With a visual layout, cluster information can be used to assign colors to data points. It can also be used behind-the-scenes to evaluate or group data automatically to determine what to show the user, e.g. WireVis [CLG+08]. Without any intervention from a user about what is expected in the data, unsupervised learning can detect structures that can guide further exploration. Note that unsupervised learning cannot use data feedback from the user, so these algorithms cannot be used as the primary learner for model steering applications. They can, however, be guided by users through *parameter tuning*, as is the case with iPCA [JZF+09a], in which principle component analysis is controlled with a visual interface.

7.2.2 Supervised Learning

Supervised machine learning encompasses a broad range of algorithms and is a powerful tool for a broad range of applications. When 'ground-truth', i.e. a label of the correct *class* or group, is available for every data point, this type of algorithm learns a function that maps a data instance to its appropriate class. The algorithm inputs a set of labeled data instances called *training data* and outputs the above function, called a *classifier*. While that classifier is expected to perform well on the training data, i.e. map an already-seen instance to the correct label, its real power is mapping previously unseen data instances to their correct label. The usual way of evaluating a classifier is *k-fold cross validation*, wherein a set of data with labels is broken randomly into k chunks (called *folds*). The machine learner is trained (i.e. the algorithm is used to build a classifier) with each possible set of $k-1$ chunks, leaving out the ith chunk. The classifier is then queried for labels of the instances of the ith chunk, whose actual labels it has not seen, and the accuracy percentage is recorded. The overall accuracy of the classifier can be determined by the average score across the different chunks. Cross validation is the baseline standard for accuracy in the machine

learning community where one often wants to know how well to expect the classifier to do on unseen data. Further evaluation is usually required to test how an algorithm responds to certain parameters for a given dataset, how confident the classifier is about its results, and how well the classifier does at retrieval metrics beyond accuracy (e.g. recall). Testing the statistical significance of the classifier's results requires separate evaluation as well (look at Bilal's reference). Further caution is required to put the accuracy results in context. If there is a class imbalance, i.e. more data points of one class than another, than the accuracy scores are affected. As an example, it means nothing to get seventy percent accuracy on a data set where seventy percent of the instances belong to one class, because guessing the majority class would be just as effective a classifier.

Classic examples of supervised machine learning algorithms include neural networks, support vector machines (SVM), logistic regression, naive bayes classifiers, and decision trees. These each have different performance characteristics and learn different types of classifying functions. While a decision tree carves out chunks of the possible input space around known examples, [AW97] an SVM discovers a hyperplane through the input space where examples on either side are assigned different class [CV95]. Different shapes of data can be more effectively modeled by different types of classifier.

Supervised machine learning algorithms can be applied to a wide variety of systems, whether for directly working with a user's data, or working behind the scenes to improve performance. Because classifiers are already so broadly applied in data analysis, and produce important but sometimes complex results, one way that visualization cooperates with classifiers is to help visualize their output. Alsallakh et el. proposed a set of visualizations that help analyze the performance of classifiers that provide classification confidence scores for multiple classes of outcome [AHH$^+$14]. *Parameter tuning* interactions can be useful in helping analysts get the most out of supervised algorithms. Gleicher [Gle13] presented Explainers that allows a user to tune readability vs. accuracy tradeoffs in SVMs. The BaobabView system by van den Elzen and van Wijk [vdEvW11] allows the user to semi-automatically refine a decision tree. There are also numerous behind-the-scenes possibilities for these algorithms. It is important to note that for both *implicit* and *explicit model steering*, using a supervised algorithm requires caution to ensure that the

assumption of having all the needed labels is appropriate. One example of how to do this comes from ReGroup by Amershi et al. [AFW12], in which a Naive Bayes classifier is continually used to help people build groups of their Facebook friends. In Chapter 6, we discuss our work [BOZ+14] in which we used supervised machine learning on *incidental* and *exploratory* user interaction data to build a classifier that can predict the performance of a user on a visual search task. That work demonstrates the possibility of detecting facets of a user automatically from interactions, but does not close the loop by following the arrow that updates the user interface based on the model. Instead, the work paves the way for future work to use such classifiers to optimize comfort and performance by tailoring data views or interface features to a class of user.

7.2.3 Semi-Supervised

In between supervised and unsupervised sits *semi-supervised* learning. The main idea is to leverage large amounts of unlabeled data to improve classification when labels are scarce (see Zhu's work for a survey [Zhu05]). However, we do not necessarily expect users of an interactive system to continually apply labels, so we pay special attention to a subgroup of these algorithms that deal with learning environments in which some "side information" beyond the data itself is available only for a subset of the data points. In systems that need human and machine cooperation, it makes sense to emphasize algorithms that can make use of limited information as it gets revealed by the user.

This latter group of semi-supervised algorithms generally rely not on labels for data instances but on a different type of ground-truth altogether. The usual form is constraints over pairs of instances that specify not what class the pair belongs to, but whether the pair's instances belong together or separate. The notion of togetherness is somewhat fluid in that the same notion applies to points belonging in the same cluster and points having the same class. In this way, the techniques of semi-supervised learning fill a special niche between supervised learning of labels and unsupervised learning of clusters. This niche may be a sweet spot for interactive systems, because these algorithms can use partial ground-truth in a way that can be more convenient for users to apply than knowing actual labels for data instances. For example, when examining a data instance, a user need not know specifically

what class that datum belongs to, but rather need only connect it to something similar or dissimilar in her or her mental model. Known classes for groups of points can certainly be applied, but the process of adding constraints can be performed while the user is working through sensemaking and may not be sure of exactly what the final data groups will be. In this way, semi-supervised algorithms can be powerful tools for leveraging *explicit model steering* interactions in complex applications. The ability to take advantage of limited labels also makes them extremely well suited to *implicit model steering* applications, where the labels are provided as part of a primary task and, not being a key priority for the user, may be scarce.

Two major categories of the semi-supervised algorithms described above are constrained clustering and metric learning. Examples of constrained clustering include COP K-means [WCR+01] modeled off k-means, and Penalized Probabilistic Clustering (PPC) [LL04], modeled off expectation maximization clustering [DLR77]. Metric learning has a broad range of literature, which is covered in Section 2.1. In visual analytics and HCI, metric learning has already been put to good use. For example, we demonstrated in DisFunction [BLBC12], discussed in Chapter 3, that certain user interactions with projected data points can be used to build a distance function that models how important different data dimensions are to a user's mental model. In iCluster [BDL10], Basu et al. provide a grouping tool that uses a learned distance function to make suggestions. Amershi et al. have used metric learning for assisting users with network alarm triage [ALK+11], and for improving image search by allowing example-based visual concepts [AFKT11].

7.2.4 Feature Creation

One important consideration for all machine learning methods is the features used to represent the data. Certainly the data types matter. As in data visualization, some techniques can accept nominal data and some are meant only for real-valued data. But beyond that, the choice of representation is part of the modeling process from the beginning and can be at least as challenging and nuanced as picking the appropriate algorithm. For example, in Chapter 6, we learn models of users from their interaction data. We tested four different encodings to turn interaction logging into vector data: (1) generate a string that contains a

symbol for every button press, treat the symbols as words, and create a vector space using standard mechanisms for text, i.e. a vector space of the n-grams that appear [CT+94]; (2) calculate statistics of the low-level mouse movements, like the average angle of motion or clicks per second; (3) and (4) model the transitions between states of the interface that participants in the study saw, and build a vector space out of the set of states and one out of the set of transitions between states. Evaluating multiple encodings meant we could search for signal from different meaningful representations. The choice of features has great impact on the effectiveness of the classifiers. In terms of accuracy, no single encoding was most effective for all target classifiers. Further, despite trying several encodings, we still were unable to get as strong of a result for personality prediction as we initially expected given the immediately salient differences in our visualizations of different personality groups. This cautionary example reveals two points - (1) it illustrates the need for using visualization and machine learning in tandem since visualization can make some patterns more apparent, and (2) it makes clear that just because a pattern is visually distinct for some representation does not mean a machine learning technique can trivially build a strong classifier. Encodings should be built with some understanding of what meaning will be modeled from the data, and as with machine learning in general, the only way to know what is actually best is to evaluate multiple methods side by side.

7.3 Metric Learning

Semi-supervised learning is especially adaptable to interactive systems, as described above. Within semi-supervised, however, metric learning deserves special consideration[1]. Several examples of its use have already been discussed in Section 7.2.3. While constrained clustering uses feedback from a user to build an optimal clustering, that clustering is the final output. Metric learning builds a distance function over the data that is consistent with the constraints. This function that maps any pair of data points to a scalar representing their dissimilarity takes into account all the feedback provided by the user. Pairs of points that the user specified belong together will map to low distances, and pairs specified as belonging

[1]Metric learning can also be used as a supervised algorithm, but here we discuss its special strengths in its semi-supervised capacity.

apart will map to large distances. The function that is output by the metric learning process can be used in clustering, to improve the notion of similarity between points, in classification, for algorithms like k-Nearest Neighbors that depend on distance, and even information retrieval, where knowing what points are most similar to a query enables a system to find the right data [JKDG09].

In addition, metric learning models have the capacity to be human-readable. In particular, the type used in Chapter 3 is a weighted Euclidean distance function, meaning there is a weight assigned to each dimension of the data, indicating its importance. The distance function can be used analytically, but a person can easily read from the weights what the relative importance of the actual, meaningful data dimensions are. Though the readability has many potential applications, this feature is especially helpful in systems designed around an *explicit model steering* environment. Since part of the user's goal is to build a strong model, they can gain extra insight by being able to understand the result.

A further advantage of metric learning algorithms is that they can begin learning with a small amount of user guidance. This is important in both *explicit model steering* and *implicit model steering* contexts, where a user needs to see the fruits of his or her labels in immediate feedback. One potential drawback is that the techniques may not be the highest performance, owing to the need for most algorithms to compute many pairwise distance calculations. However, there are a number of algorithms with different performance characteristics depending on what is desired of the distance metric. For example, Dis-Function [BLBC12] is intended for an interactive context and minimizes overall change to the model at each interactive step. On the other hand, much metric learning work, including foundational work by Xing [XNJR02], optimizes for as strong as possible a result with the given labels, ignoring unlabeled data. In between, Leman et al. [LHM+13] introduced a formulation that allows adjusting the strength of previous user constraints. Perhaps most promising in terms of speed performance is the LEGO [JKDG09] algorithm, which is specialized for quick updates (it is an online algorithm, see Section 7.4). For any algorithm, there are substantial gains that can be made with careful engineering.

Finally, pairwise distance functions are directly applicable to visualization. In visualizing high dimensional data, including text data, one common approach is to project

the data into a two-dimensional plane. The method for calculating this projection is often an optimization that attempts to minimize the differences between the pairwise distances in high dimensional space and low-dimensional space, e.g. multidimensional scaling (MDS) [BSL+08]. When the distances in high-dimensional space are governed by a well-crafted distance function, the visual results can be much improved. Thanks to their role in high-dimensional visualization, metric learning algorithms can be helpful not only for model steering, but for aiding the *exploratory* interactions.

7.4 Online and Active Learning

As the previous sections described, there are applications of all categories of machine learning to interactive systems. There are two smaller sub-fields of machine learning that have additional promise for interactivity. The first tackles one of the main challenges of interactive machine learning: the learning algorithms must react quickly to a small amount of feedback. The usual case of a supervised learning algorithm may not be suited to taking a small piece of information and adjusting the model. Some algorithms take the full set of points into account all at once. The simplest approach is to re-learn the model with the new feedback appended to the previous feedback, but that is wasteful and can become computationally prohibitively expensive, especially for the interactive timeframe. There is, however, a class of algorithm designed to make small changes quickly. *Online* algorithms are a subset of supervised algorithms that are built to accept one label at a time [LGKM06]. One example application is a security system that recognizes people in a video feed. It may be expensive to train it to recognize a single person by digesting many images. However, it is still important that the learner be able to improve when new labeled images of that person are available, providing examples of what he or she looks like in different light or clothing. An online algorithm can be provided one additional labeled picture and take that into account without having to reconsider all the previous training. Though this type of learner is generally discussed in a supervised context, it can be applied to interactivity. If we remove the assumption that we have labels for all data points but are gradually encountering more, we can just take advantage of the fact that we have a classifier that can adjust quickly with

new information. In an interactive context, such behavior is critical, since a user needs to see quick updates based on incremental information. In fact, online algorithms are excellent candidates to be used in interactive settings, especially with *implicit* and *explicit model steering* types of interactions. When a supervised algorithm creates a desirable model but is too slow for interactive use, an online implementation can sometimes address that concern.

The second pertinent subfield of machine learning is *active learning*, which falls under semi-supervised learning. Researchers in active learning study ways to work most effectively with a limited number of labels by taking control over which labels are seen. Given the semi-supervised assumption that there is a large collection of unlabeled data, these algorithms incrementally query users for labels on data points that will be the most helpful to the learning algorithm. There are a number of mechanisms for choosing such points, perhaps the most common of which is querying for labels of data points about which the current model is least certain (uncertainty sampling) [Set10]. Active learning algorithms are poised to be incredibly useful in visual analytics because when an actual human is engaging with a system, one must maximize use of valuable expert time. In such a system, keeping the user in control is important to his or her exploration, so techniques are needed that can guide the user to fruitful exploration without controlling the process altogether. Regrettably, in active learning literature, the human expert providing labels is generally simulated by a label-proving oracle. The oracle is always correct, and by virtue of being digital does not mind providing labels one at a time as an algorithm chooses. Researchers look at how the cross-validation accuracy improves as a function of how many labels have been applied by the simulated user. Chapter 4 is work-in-progress but provides an approach for active learning that keeps a human user in control. Only after a user begins an inquiry by choosing some data point, the algorithm reveals points of interest relative to that chosen point. The points are chosen to maximize change to the model. For a type of system like this that keeps the user in control, evaluation is more complicated. Simulating a user who is actually engaged in an analysis process is different from just applying labels when asked. Simulating with an oracle that is too strong makes all algorithms look the same because all can be perfect. Simulating a weak human means that while we may be making some strong suggestions, the human may be picking the worst among them. In the particular

case of Chapter 4, we are recommending strong updates to the model, but we can only estimate the strength of the change, not its quality. The worst of the suggestions could be strongly negative. Active learning can play a powerful role in future interactive systems by most effectively leveraging user efforts, but techniques must be developed and properly vetted that can keep take best advantage of users' time while keeping them in control. By guiding exploration, these techniques will link the *exploratory* and *implicit* or *explicit model steering* interactions to amplify the user's efforts.

7.5 Summary

As the size and complexity of data continue to increase, utilizing machine learning techniques in visual analytics systems will be even more critical for assisting the user in exploring and analyzing the data. However, there has yet to be a systematic guideline for integrating user interfaces and interaction techniques with back-end machine learning techniques. Although such a guideline is beyond the scope of this thesis, in this chapter I sketched a framework that categorizes interaction types (incidental, exploratory, parameter tuning, implicit and explicit model steering), and explains how they fit with a variety of the most effective machine learning algorithms for this interactive context. In particular, in Sections 7.2 – 7.4, I broadly categorized machine learning algorithms, and provided examples of their relationships to the interaction types.

The complexity and depth of the framework lies in the potential to combine the interaction types and techniques in different ways. In Figure 7.1, I present a table showing how combinations of interaction types and machine learning algorithms can lead to very different systems and applications, resulting in a wide range of possibilities. Some of these combinations would be helped by innovation in algorithms by the machine learning community. Currently, there is not a strong interest in human users in that community, but those of us who see the potential for working together must continue to make the case. One compelling but simple example of why this is necessary comes from the well-known illustration given by statistician Francis Anscombe [Ans73]. He created four small datasets, each with two variables. Visually, as seen in ordinary scatter plots of the data (Figure 7.2), it is im-

	Incidental	*Exploratory*	*Parameter Tuning*	*Explicit Model Steering*	*Implicit Model Steering*
Supervised	Learn models for differentiating between groups of users from their interactions, e.g. between time periods or between types of users as in Chapter 6.	Applies to visualizations of supervised models, i.e. the exploration is of the model itself, OR visualization of data that provide means for users to decide on labels.	The output of the machine learner is being changed in response to the user adjusting the parameters, directly or indirectly, as opposed to providing labels.	User is using a visualization to control a supervised model, perhaps in tandem with using exploratory interactions to learn about it.	User engaged in a primary task that somehow involves applying a label to every data point, e.g. an application for users to sort photographs where they must file every one.
Semi-supervised	Applicable if building user models but can only determine labels, e.g. real-time via heuristic, for some instances.	Applies to visualization of semi-supervised models, i.e. the exploration is of the model OR visualization of data that provide means for users to decide on feedback.		User is supplying labels or feedback about some of the data points, e.g. the work of Chapters 3 and 5.	User engaged in a primary task that involves applying labels or feedback on a subset of the data points.
Unsupervised	Applies only to unsupervised analysis of groups of users.	Applies to visualization of unsupervised models, i.e. the exploration of the model.		Does not apply since the unsupervised model cannot accept feedback.	
Online	Specialization of supervised algorithms that can accept labels for data points one at a time.			Online versions of supervised algorithms can make those powerful algorithms plausible for use in these interactive scenarios.	
Metric	Can be used as a supervised or semi-supervised algorithm and used as explained above.	A type of learner, thus not directly part of exploration, except that metric models can improve the visualization of high-dimensional data, e.g. with enhanced projections.	As with other supervised and semi-supervised algorithms, parameters can greatly affect the output of a metric learner.	Useful type of model that can be supervised or semi-supervised and produces an exportable model.	
Active	Does not apply to interactions with data, but back-end could potentially try to setup the interface to get user to react.	Can affect the exploratory environment by introducing information about how to change the model most effectively.	User could tune parameters of an active learning algorithm to affect the recommendations provided to the user.	Can help improve the effectiveness of the user at providing model steering inputs.	Only applies if the primary task can be altered to get the user to provide labels that are more helpful to the learner.

Table 7.1: This table explains how the different types of interactions match up with the different types of machine learning algorithms.

mediately apparent that these are very different data. However, all four sets in fact have the same mean, standard deviation, correlation, and linear regression parameters, so summary statistics would miss the difference completely. Certainly machine learning can offer deeper analysis than summary statistics, but there are limitations to what can be perceived without human review of results. As machine-learning projects without appropriate visualization to verify the results begin to cause problems in industry and science, we may see more interest in a blended approach.

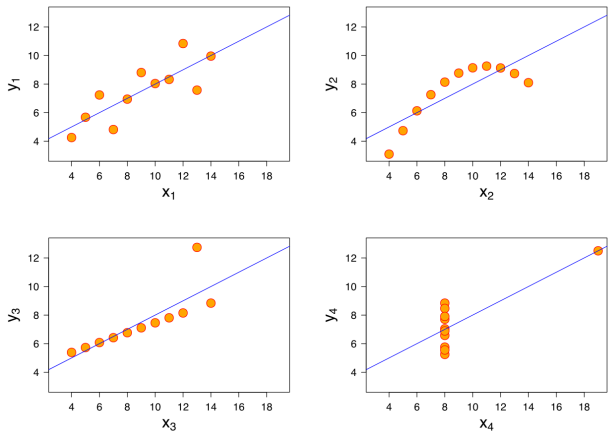


Figure 7.2: Anscombe's quartet [Ans73], a collection of four datasets that illustrate the need for visualization. Each scatter plot is a different dataset, but all share the same statistical properties including mean, variance and regression parameters.

Figure 7.1 and the corresponding framework are based on my observations and experience working on Dis-Function, Doc-Function, EigenSense, and Finding Waldo. Neither is meant to be definitive, but they are instead meant to be the start of a systematic guideline for advancing machine learning and visual analytics research. Although a more thorough investigation into the validity and completeness of this framework is necessary, it hopefully can serve as a starting point for developing a much needed guideline for the field of visual analytics.

Chapter 8

Conclusion

In this dissertation, I have discussed my work towards analytics systems that integrate visualization and machine learning to allow data stakeholders to reap the benefits of high-end model learning techniques while neither having to turn the process over to people outside their field nor gain separate expertise in data science. First, I described a prototype system for learning distance functions by interacting directly with a visualization of high-dimensional numerical data. Despite the relative convenience of learning models by interacting with visualizations of data, the user may still find the number of data points to examine burdensome. In order to provide the kind of feedback described, many comparisons between data points may be required. To facilitate this process and maximize the effects of the user's hard work, I introduced a work-in-progress system that helps the user focus inquiries on the most fruitful parts of the data. Next, I provided an extension to this automatic model-building work that shows the same model-building can be performed for text data. The models being built are weighted Euclidean distance functions, meaning that (1) they are human-readable (they tell the relative importance of each data dimension), and (2) they are represented as vectors. Because of the latter, I showed the possibilities for reasoning about users' analytical processes through a numerical provenance that considers movement through the (vector) space of possible models. In a continued vein of improving the analyst's experience, I provided a proof-of-concept experiment showing that it is plausible to learn about users automatically from their interactions. This technology could someday be used to create systems that optimize their interfaces to fit with the type of user at the

controls. Finally, I presented the sketch of a framework for how these integrated machine learning and human computer interface systems can be designed, including the types of interaction and machine learning mechanisms, and a discourse on machine learning methods and how they fit this framework.

Bibliography

[AABW12] Gennady Andrienko, Natalia Andrienko, Michael Burch, and Daniel Weiskopf. Visual analytics methodology for eye movement studies. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2889–2898, 2012.

[AAR+09] G. Andrienko, N. Andrienko, S. Rinzivillo, M. Nanni, D. Pedreschi, and F. Giannotti. Interactive visual clustering of large collections of trajectories. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 3–10. IEEE, 2009.

[AEN10] Christopher Andrews, Alex Endert, and Chris North. Space to think: large high-resolution displays for sensemaking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 55–64. ACM, 2010.

[AFKT11] Saleema Amershi, James Fogarty, Ashish Kapoor, and Desney S Tan. Effective end-user interaction with machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1529–1532, 2011.

[AFW12] Saleema Amershi, James Fogarty, and Daniel Weld. Regroup: Interactive machine learning for on-demand group creation in social networks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 21–30. ACM, 2012.

[AHH+14] Bilal Alsallakh, Allan Hanbury, Helwig Hauser, Silvia Miksch, and Andreas Rauber. Visual methods for analyzing probabilistic classification data. *Vi-*

sualization and Computer Graphics, IEEE Transactions on, 20(12):1703–1712, 2014.

[ALK+11] Saleema Amershi, Bongshin Lee, Ashish Kapoor, Ratul Mahajan, and Blaine Christian. Human-guided machine learning for fast and accurate network alarm triage. In *Proceedings of the International Joint Conference on Artifical Intelligence (IJCAI)*, pages 2564–2569, 2011.

[All00] Bryce Allen. Individual differences and the conundrums of user-centered design: Two experiments. *Journal of the american society for information science*, 51(6):508–520, 2000.

[AM98] N. Abe and H. Mamitsuka. Query learning strategies using boosting and bagging. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML)*, pages 1–9, 1998.

[Ans73] Francis J Anscombe. Graphs in statistical analysis. *The American Statistician*, 27(1):17–21, 1973.

[ART06] Rakesh Agrawal, Ralf Rantzau, and Evimaria Terzi. Context-sensitive ranking. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, pages 383–394. ACM, 2006.

[AW97] Chidanand Apté and Sholom Weiss. Data mining with decision trees and decision rules. *Future generation computer systems*, 13(2):197–210, 1997.

[BB13] Sergey Bochkanov and Vladimir Bystritsky. ALGLIB, 2013. http://www.alglib.net.

[BBM04a] S. Basu, A. Banerjee, and R.J. Mooney. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the IEEE International Conference on Data Mining*, pages 333–344, 2004.

[BBM04b] M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pages 81–88, 2004.

[BC57] J Roger Bray and John T Curtis. An ordination of the upland forest communities of southern wisconsin. *Ecological monographs*, 27(4):325–349, 1957.

[BCBN11] Shawn J Bohn, Augustin J Calapristi, Shyretha D Brown, and Grant C Nakamura. Analytic steering: Inserting context into the informaion dialog. In *IEEE Workshop on Interactive Visual Text Analytics for Decision Making at VisWeek 2011*, 2011.

[BCC+05] Louis Bavoil, Steven P Callahan, Patricia J Crossno, Juliana Freire, Carlos E Scheidegger, Cláudio T Silva, and Huy T Vo. Vistrails: Enabling interactive multiple-view visualizations. In *Visualization, 2005. VIS 05. IEEE*, pages 135–142. IEEE, 2005.

[BCK06] J. Broekens, T. Cocx, and W.A. Kosters. Object-centered interactive multi-dimensional scaling: Ask the expert. In *Proceedings of the Eighteenth Belgium-Netherlands Conference on Artificial Intelligence (BNAIC)*, pages 59–66, 2006.

[BDL10] Sumit Basu, Steven M Drucker, and Hao Lu. Assisting Users with Clustering Tasks by Combining Metric Learning and Classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 394–400, 2010.

[BG05] I. Borg and P.J.F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer Verlag, 2005.

[BHHSW05] Aharon Bar-Hillel, Tomer Hertz, Noam Shental, and Daphna Weinshall. Learning a mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, 6(6):937–965, 2005.

[BJV] Sumit Basu, Chuck Jacobs, and Lucy Vanderwende. Powergrading: a clustering approach to amplify human effort for short answer grading.

[BL13] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[BLBC12] Eli T Brown, Jingjing Liu, Carla E Brodley, and Remco Chang. Dis-function: Learning distance functions interactively. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 83–92. IEEE, 2012.

[BNH14] Lauren Bradel, Chris North, and Leanna House. Multi-model semantic interaction for text analytics. In *Visual Analytics Science and Technology (VAST), 2014 IEEE Conference on*, pages 163–172. IEEE, 2014.

[BOZ+14] Eli T Brown, Alvitta Ottley, Helen Zhao, Quan Lin, Richard Souvenir, Alex Endert, and Remco Chang. Finding waldo: Learning about users from their interactions. 2014.

[BSL+04] A. Buja, D. Swayne, M. Littman, N. Dean, and H. Hofmann. Interactive data visualization with multidimensional scaling. *Report, University of Pennsylvania*, 2004.

[BSL+08] Andreas Buja, Deborah F Swayne, Michael L Littman, Nathaniel Dean, Heike Hofmann, and Lisha Chen. Data visualization with multidimensional scaling. *Journal of Computational and Graphical Statistics*, 17(2):444–472, 2008.

[CCR+13] E.A. Carroll, M. Czerwinski, A. Roseway, A. Kapoor, P. Johns, K. Rowan, and M.C. Schraefel. Food and mood: Just-in-time support for emotional eating. In *Proceedings or the Humaine Association Conference on Affective Computing and Intelligent Interaction (ACII)*, pages 252–257, Sept 2013.

[CCS00] C. Campbell, N. Cristianini, and A. Smola. Query learning with large margin classifiers. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pages 111–118, 2000.

[CFS+06] Steven P Callahan, Juliana Freire, Emanuele Santos, Carlos E Scheidegger, Cláudio T Silva, and Huy T Vo. Vistrails: visualization meets data manage-

ment. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 745–747. ACM, 2006.

[Che00] Chaomei Chen. Individual differences in a spatial-semantic virtual environment. *Journal of the American Society for Information Science*, 51(6):529–542, 2000.

[CHL05] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 539–546, 2005.

[CLG+08] Remco Chang, Alvin Lee, Mohammad Ghoniem, Robert Kosara, William Ribarsky, Jing Yang, Evan Suma, Caroline Ziemkiewicz, Daniel Kern, and Agus Sudjianto. Scalable and interactive visual analysis of financial wire transactions for fraud detection. *Information visualization*, 7(1):63–76, 2008.

[CLKP10] J. Choo, H. Lee, J. Kihm, and H. Park. iVisClassifier: An interactive visual analytics system for classification based on supervised dimension reduction. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 27–34. IEEE, 2010.

[CM08] Cristina Conati and Heather Maclaren. Exploring the role of individual differences in information visualization. In *Proceedings of the working conference on Advanced visual interfaces*, pages 199–206. ACM, 2008.

[CNS05] Paula Cowley, Lucy Nowell, and Jean Scholtz. Glass box: An instrumented infrastructure for supporting human interaction with information. In *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on*, pages 296c–296c. IEEE, 2005.

[Coo06] Lynne Cooke. Is the mouse a "poor man's eye tracker"? In *ANNUAL CONFERENCE-SOCIETY FOR TECHNICAL COMMUNICATION*, volume 53, page 252, 2006.

[CORE08] Murat Cokol, Fatih Ozbay, and Raul Rodriguez-Esteban. Retraction rates are on the rise. *EMBO reports*, 9(1):2–2, 2008.

[CT+94] William B Cavnar, John M Trenkle, et al. N-gram-based text categorization. *Ann Arbor MI*, 48113(2):161–175, 1994.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[CWL+10] Weiwei Cui, Yingcai Wu, Shixia Liu, Furu Wei, Michelle X Zhou, and Huamin Qu. Context preserving dynamic word cloud visualization. In *Pacific Visualization Symposium (PacificVis), 2010 IEEE*, pages 121–128. IEEE, 2010.

[CZGR09] Remco Chang, Caroline Ziemkiewicz, Tera Marie Green, and William Ribarsky. Defining insight for visual analytics. *Computer Graphics and Applications, IEEE*, 29(2):14–17, 2009.

[dCCM09] Sérgio Manuel Serra da Cruz, Maria Luiza Machado Campos, and Marta Mattoso. Towards a taxonomy of provenance in scientific workflow management systems. In *Services-I, 2009 World Conference on*, pages 259–266. IEEE, 2009.

[DHRL+12] Cody Dunne, Nathalie Henry Riche, Bongshin Lee, Ronald Metoyer, and George Robertson. GraphTrail: Analyzing large multivariate, heterogeneous networks while supporting exploration history. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1663–1672. ACM, 2012.

[DJS+09] Wenwen Dou, Dong Hyun Jeong, Felesia Stukes, William Ribarsky, Heather Richter Lipford, and Remco Chang. Recovering reasoning processes from user interactions. *IEEE Computer Graphics and Applications*, (3):52–61, 2009.

[DKJ+07] J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon. Information-theoretic metric learning. In *Proceedings of Twenty-Fourth International Conference on Machine Learning (ICML)*, pages 209–216, 2007.

[DLR77] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, pages 1–38, 1977.

[DMF07] M. Desjardins, J. MacGlashan, and J. Ferraioli. Interactive visual clustering. In *Proceedings of the Twelfth International Conference on Intelligent User Interfaces*, pages 361–364. ACM, 2007.

[DOBL06] M Brent Donnellan, Frederick L Oswald, Brendan M Baird, and Richard E Lucas. The mini-ipip scales: tiny-yet-effective measures of the big five factors of personality. *Psychological assessment*, 18(2):192, 2006.

[DP12] Thomas H. Davenport and D.J. Patil. Data scientist: The sexiest job of the 21st century. October, 2012.

[dS12] César Roberto de Souza. A tutorial on principal component analysis with the accord. net framework. *arXiv preprint arXiv:1210.7463*, 2012.

[DZG+07] Anthony Don, Elena Zheleva, Machon Gregory, Sureyya Tarkan, Loretta Auvil, Tanya Clement, Ben Shneiderman, and Catherine Plaisant. Discovering interesting usage patterns in text collections: integrating text mining with visualization. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 213–222. ACM, 2007.

[EBC+13] Alex Endert, Russ Burtner, Nick Cramer, Ralph Perko, Shawn Hampton, and Kristin Cook. Typograph: Multiscale spatial exploration of text documents. In *Big Data, 2013 IEEE International Conference on*, pages 17–24. IEEE, 2013.

[Eco13] The Economist. Trouble at the lab. October 19th, 2013.

[EFN12a] A. Endert, P. Fiaux, and C. North. Semantic interaction for visual text analytics. In *Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems (CHI)*, pages 473–482. ACM, 2012.

[EFN12b] Alex Endert, Patrick Fiaux, and Chris North. Semantic interaction for visual text analytics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 473–482. ACM, 2012.

[EFN12c] Alex Endert, Patrick Fiaux, and Chris North. Semantic interaction for visual text analytics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 473–482. ACM, 2012.

[EFS12] Sandra Ebert, Mario Fritz, and Bernt Schiele. Active metric learning for object recognition. In Axel Pinz, Thomas Pock, Horst Bischof, and Franz Leberl, editors, *Pattern Recognition*, volume 7476 of *Lecture Notes in Computer Science*, pages 327–336. Springer Berlin Heidelberg, 2012.

[EG11] Stephen P Ellner and John Guckenheimer. *Dynamic models in biology*. Princeton University Press, 2011.

[EHM+11a] A. Endert, C. Han, D. Maiti, L. House, and C. North. Observation-level interaction with statistical models for visual analytics. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 121–130. IEEE, 2011.

[EHM+11b] Alex Endert, Chao Han, Dipayan Maiti, Leanna House, Scotland Leman, and Chris North. Observation-level interaction with statistical models for visual analytics. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*, pages 121–130. IEEE, 2011.

[EHR+14] Alex Endert, M Shahriar Hossain, Naren Ramakrishnan, Chris North, Patrick Fiaux, and Christopher Andrews. The human is the loop: new directions for visual analytics. *Journal of Intelligent Information Systems*, 43(3):411–435, 2014.

[EKS+96] Martin Ester, Hans-Peter Kriegel, Jrg Sander, Xiaowei Xu, Evangelos Simoudis, Jiawei Han, and ed. Fayyad, Usama M. A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231, 1996.

[ENCZ] Alex Endert, Chris North, Remco Chang, and Michelle Zhou. Toward usable interactive analytics: Coupling cognition and computation.

[EV03] Magdalini Eirinaki and Michalis Vazirgiannis. Web mining for web personalization. *ACM Transactions on Internet Technology (TOIT)*, 3(1):1–27, 2003.

[FKSS08] Juliana Freire, David Koop, Emanuele Santos, and Cláudio T Silva. Provenance for computational tasks: A survey. *Computing in Science & Engineering*, 10(3):11–21, 2008.

[FSST97] Y. Freund, H. Seung, Sebastian, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning Journal*, pages 133–168, 1997.

[GF10] Tera Marie Green and Brian Fisher. Towards the personal equation of interaction: The impact of personality factors on visual analytics interface interaction. In *Visual Analytics Science and Technology (VAST), 2010 IEEE Symposium on*, pages 203–210. IEEE, 2010.

[GJE+06] Lewis R Goldberg, John A Johnson, Herbert W Eber, Robert Hogan, Michael C Ashton, C Robert Cloninger, and Harrison G Gough. The international personality item pool and the future of public-domain personality measures. *Journal of Research in Personality*, 40(1):84–96, 2006.

[GKWZ07] Alexander N. Gorban, Balzs Kgl, Donald C. Wunsch, and Andrei Zinovyev. *Principal Manifolds for Data Visualization and Dimension Reduction*. Springer Publishing Company, Incorporated, 2007.

[Gle13] Michael Gleicher. Explainers: Expert explorations with crafted projections. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2042–2051, 2013.

[GNRM08a] S. Garg, J.E. Nam, IV Ramakrishnan, and K. Mueller. Model-driven visual analytics. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 19–26. IEEE, 2008.

[GNRM08b] Supriya Garg, Julia Eunju Nam, IV Ramakrishnan, and Klaus Mueller. Model-driven visual analytics. In *Visual Analytics Science and Technology, 2008. VAST'08. IEEE Symposium on*, pages 19–26. IEEE, 2008.

[GRHS04] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems 19*, pages 513–520, 2004.

[GVS09] M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? Metric learning approaches for face identification. In *Proceedings of the International Conference on Computer Vision*, pages 498–505, 2009.

[GW04] Krzysztof Gajos and Daniel S Weld. Supple: automatically generating user interfaces. In *Proceedings of the 9th international conference on Intelligent user interfaces*, pages 93–100. ACM, 2004.

[Han87] Martin Handford. *Where's Waldo?* Little, Brown Boston, 1987.

[HB97] T. Hofmann and J.M. Buhmann. Active data clustering. In *Advances in Neural Information Processing Systems 12*, pages 528–534, 1997.

[HCL10] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. Technical report, National Taiwan University, Taipei 106, Taiwan, April 2010.

[Hea01] M.T. Heath. *Scientific Computing*. The McGraw-Hill Companies, Incorporated, 2001.

[HFH+09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[HGP99] P. Hoffman, G. Grinstein, and D. Pinkney. Dimensional anchors: A graphic primitive for multidimensional multivariate information visualizations. In *Proceedings of the 1999 Workshop on New Paradigms in Information Visualization and Manipulation in Conjunction With the Eighth ACM International Conference on Information and Knowledge Management*, pages 9–16. ACM, 1999.

[HLLM06] S.C.H. Hoi, W. Liu, M.R. Lyu, and W. Ma. Learning distance metrics with contextual constraints for image retrieval. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2072–2078, 2006.

[HMSA08] Jeffrey Heer, Jock Mackinlay, Chris Stolte, and Maneesh Agrawala. Graphical histories for visualization: Supporting analysis, communication, and evaluation. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1189–1196, 2008.

[HWB12] Jeff Huang, Ryen White, and Georg Buscher. User see, user point: gaze and cursor alignment in web search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 1341–1350, New York, NY, USA, 2012. ACM.

[HZ06] William W Hager and Hongchao Zhang. A survey of nonlinear conjugate gradient methods. *Pacific journal of Optimization*, 2(1):35–58, 2006.

[ID90] A. Inselberg and B. Dimsdale. Parallel coordinates: A tool for visualizing multi-dimensional geometry. In *Proceedings of the 1st Conference on Visualization*, pages 361–378. IEEE, 1990.

[JKDG09] Prateek Jain, Brian Kulis, Inderjit S Dhillon, and Kristen Grauman. Online metric learning and fast similarity search. In *Advances in neural information processing systems*, pages 761–768, 2009.

[JLMP93] Robert J.K. Jacob, John J Leggett, Brad A Myers, and Randy Pausch. Interaction styles and input/output devices. *Behaviour & Information Technology*, 12(2):69–79, 1993.

[JMF99] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[Jol86] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986.

[JS04] R. Jin and L. Si. A bayesian approach toward active learning for collaborative filtering. In *Uncertainty in Artificial Intelligence*, pages 278–285, 2004.

[JZF+09a] D. Jeong, C. Ziemkiewicz, B. Fisher, W. Ribarsky, and R. Chang. iPCA: An interactive system for PCA-based visual analytics. *Computer Graphics Forum*, pages 767–774, 2009.

[JZF+09b] Dong Hyun Jeong, Caroline Ziemkiewicz, Brian Fisher, William Ribarsky, and Remco Chang. ipca: An interactive system for pca-based visual analytics. In *Computer Graphics Forum*, volume 28, pages 767–774. Wiley Online Library, 2009.

[KAF+08] Daniel Keim, Gennady Andrienko, Jean-Daniel Fekete, Carsten Görg, Jörn Kohlhammer, and Guy Melançon. *Visual analytics: Definition, process, and challenges*. Springer, 2008.

[KB00] Raymond Kosala and Hendrik Blockeel. Web mining research: A survey. *ACM Sigkdd Explorations Newsletter*, 2(1):1–15, 2000.

[KCD+09] Nazanin Kadivar, Victor Chen, Dustin Dunsmuir, Eric Lee, Cheryl Qian, John Dill, Christopher Shaw, and R Woodbury. Capturing and supporting the analysis process. In *Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on*, pages 131–138. IEEE, 2009.

[KJ04] Pranam Kolari and Anupam Joshi. Web mining: Research and practice. *Computing in science & engineering*, 6(4):49–53, 2004.

[KL07] Owen Kaser and Daniel Lemire. Tag-cloud drawing: Algorithms for cloud visualization. *arXiv preprint cs/0703109*, 2007.

[LB99] Terran Lane and Carla E. Brodley. Temporal sequence learning and data reduction for anomaly detection. *ACM Transactions on Information and System Security*, 2(3):295–331, August 1999.

[LGKM06] Pavel Laskov, Christian Gehl, Stefan Krüger, and Klaus-Robert Müller. Incremental support vector learning: Analysis, implementation and applications. *The Journal of Machine Learning Research*, 7:1909–1936, 2006.

[LHM+13] SC Leman, L House, D Maiti, A Endert, and C North. Visual to parametric interaction (v2pi). *PLoS ONE*, 8(3):e50474, 2013.

[LIRC12] Bongshin Lee, P. Isenberg, N.H. Riche, and S. Carpendale. Beyond mouse and keyboard: Expanding design considerations for information visualization interactions. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 18(12):2689–2698, Dec 2012.

[LL04] Zhengdong Lu and Todd K Leen. Semi-supervised learning with penalized probabilistic clustering. In *Advances in neural information processing systems*, pages 849–856, 2004.

[LME10] Aidong Lu, Ross Maciejewski, and David S Ebert. Volume composition and evaluation using eye-tracking data. *ACM Transactions on Applied Perception (TAP)*, 7(1):4, 2010.

[LRMM12] Benjamin Letham, Cynthia Rudin, Tyler H McCormick, and David Madigan. Building interpretable classifiers with rules using bayesian analysis. *Department of Statistics Technical Report tr609, University of Washington*, 2012.

[Mac67] J. B. Macqueen. Some methods of classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[Mac92] D.J.C. MacKay. Information-based objective functions for active data selection. *Neural Computation*, pages 590–604, 1992.

[Max03] R.A. Maxion. Masquerade detection using enriched command lines. In *Dependable Systems and Networks, 2003. Proceedings. 2003 International Conference on*, pages 5–14. IEEE, 2003.

[MBD+11] T. May, A. Bannach, J. Davey, T. Ruppert, and J. Kohlhammer. Guiding feature subset selection with an interactive visualization. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 111–120. IEEE, 2011.

[MDL07] Yi Mao, Joshua V Dillon, and Guy Lebanon. Sequential document visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1208–1215, 2007.

[MFL88] C. Armanino M. Forina, R. Leardi and S. Lanteri. PARVUS: An extendable package of programs for data exploration, classification and correlation. *Journal of Chemometrics*, pages 191–193, 1988.

[Mit97] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[MM04] P. Melville and R. J. Mooney. Diverse ensembles for active learning. In *Proceedings of the Twenty-First International Conference on Machine Learning (ICML)*, pages 74–83, 2004.

[MP13] Thomas Muhlbacher and Harald Piringer. A partition-based framework for building and validating regression models. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):1962–1971, 2013.

[MRS08a] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. Scoring, term weighting and the vector space model. *Introduction to Information Retrieval*, 100, 2008.

[MRS+08b] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[NCE+11] Chris North, Remco Chang, Alex Endert, Wenwen Dou, Richard May, Bill Pike, and Glenn Fink. Analytic provenance: process+ interaction+ insight. In *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems*, pages 33–36. ACM, 2011.

[NHM+07] E.J. Nam, Y. Han, K. Mueller, A. Zelenyuk, and D. Imre. ClusterSculptor: A visual analytics tool for high-dimensional data. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 75–82. IEEE, 2007.

[OCZC13] Alvitta Ottley, R Jordan Crouser, Caroline Ziemkiewicz, and Remco Chang. Manipulating and controlling for personality effects on visualization tasks. *SAGE Publications Information Visualization*, 2013.

[OY11] M. Okabe and S. Yamada. An interactive tool for human active learning in constrained clustering. *Journal of Emerging Technologies in Web Intelligence*, 3(1):20–27, February 2011.

[PB04] M. Pusara and C.E. Brodley. User re-authentication via mouse movements. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 1–8. ACM, 2004.

[PBMW99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: bringing order to the web. *Stanford InfoLab*, 1999.

[PC05] Peter Pirolli and Stuart Card. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Pro-*

ceedings of International Conference on Intelligence Analysis, volume 5, pages 2–4, 2005.

[PHf14] Bernhard Pfahringer, Geoff Holmes, and fracpete. Class gridsearch, revision 9733. http://weka.sourceforge.net/doc.stable/weka/classifiers/meta/GridSearch.html, 2014.

[PSCO09] William A Pike, John Stasko, Remco Chang, and Theresa A O'Connell. The science of interaction. *Information Visualization*, 8(4):263–274, 2009.

[RBC+09] Stuart Rose, Scott Butner, Wendy Cowley, Michelle Gregory, and Julia Walker. Describing story evolution from dynamic information streams. In *Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on*, pages 99–106. IEEE, 2009.

[RECC10] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. Automatic keyword extraction from individual documents. *Text Mining*, pages 1–20, 2010.

[RF06] R. Rosales and G. Fung. Learning sparse metrics via linear programming. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 367–373, 2006.

[RJMK14] Tuukka Ruotsalo, Giulio Jacucci, Petri Myllymäki, and Samuel Kaski. Interactive intent modeling: information discovery beyond search. *Communications of the ACM*, 58(1):86–92, 2014.

[RM01] N. Roy and A. Mccallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*, pages 441–448, 2001.

[RM06] R.W. Reeder and R.A. Maxion. User interface defect detection by hesitation analysis. In *Dependable Systems and Networks, 2006. DSN 2006. International Conference on*, pages 61–72. IEEE, 2006.

[Rob07] J.C. Roberts. State of the art: Coordinated multiple views in exploratory visualization. In *Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV)*, pages 61–71, July 2007.

[Roc71] Joseph John Rocchio. Relevance feedback in information retrieval. 1971.

[SBE+11] Ankit Singh, Lauren Bradel, Alex Endert, Robert Kincaid, Christopher Andrews, and Chris North. Supporting the cyber analytic process using visual history on large displays. In *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, page 3. ACM, 2011.

[SCC13] Ben Steichen, Giuseppe Carenini, and Cristina Conati. User-adaptive information visualization: using eye gaze data to infer visualization tasks and user cognitive abilities. In *Proceedings of the 2013 international conference on Intelligent user interfaces*, pages 317–328. ACM, 2013.

[SCDT00] Jaideep Srivastava, Robert Cooley, Mukund Deshpande, and Pang-Ning Tan. Web usage mining: Discovery and applications of usage patterns from web data. *ACM SIGKDD Explorations Newsletter*, 1(2):12–23, 2000.

[Sch90] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[Set10] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 2010.

[SGL08] John Stasko, Carsten Görg, and Zhicheng Liu. Jigsaw: supporting investigative analysis through interactive visualization. *Information Visualization*, 7(2):118–132, 2008.

[SIM99] Frank M Shipman III and Catherine C Marshall. Formality considered harmful: Experiences, emerging themes, and directions on the use of formal representations in interactive systems. *Computer Supported Cooperative Work (CSCW)*, 8(4):333–352, 1999.

[SKK+08] Christin Seifert, Barbara Kump, Wolfgang Kienreich, Gisela Granitzer, and Michael Granitzer. On the beauty and usability of tag clouds. In *Information Visualisation, 2008. IV'08. 12th International Conference*, pages 17–25. IEEE, 2008.

[SKWH09] C. Shen, J. Kim, L. Wang, and A. Hengel. Positive semidefinite metric learning with boosting. In *Advances in Neural Information Processing Systems 22*, pages 1651–1659, 2009.

[SOS92] H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 287–294. ACM, 1992.

[SRL+09] Simone Stumpf, Vidya Rajaram, Lida Li, Weng-Keen Wong, Margaret Burnett, Thomas Dietterich, Erin Sullivan, and Jonathan Herlocker. Interacting meaningfully with machine learning systems: Three experiments. *International Journal of Human-Computer Studies*, 67(8):639–662, 2009.

[TC05] James J Thomas and Kristin A Cook. *Illuminating the path: The research and development agenda for visual analytics*. IEEE Computer Society Press, 2005.

[TCSC13] Dereck Toker, Cristina Conati, Ben Steichen, and Giuseppe Carenini. Individual user characteristics and information visualization: Connecting the dots through eye tracking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 295–304. ACM, 2013.

[TFH11] Cagatay Turkay, Peter Filzmoser, and Helwig Hauser. Brushing dimensions-a dual visual analysis model for high-dimensional data. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2591–2599, 2011.

[TK01a] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In *Journal Of Machine Learning Research*, pages 999–1006, 2001.

[TK01b] Simon Tong and Daphne Koller. Active learning for parameter estimation in bayesian networks. In *In Advances in Neural Information Processing Systems 14*, pages 647–653, 2001.

[TL07] L. Torresani and K. C. Lee. Large margin component analysis. In *Advances in Neural Information Processing Systems 20*, pages 1385–1392, 2007.

[TP91] Matthew A Turk and Alex P Pentland. Face recognition using eigenfaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–591. IEEE, 1991.

[Vap98] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[vdEvW11] Stef van den Elzen and Jarke J van Wijk. Baobabview: Interactive construction and analysis of decision trees. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*, pages 151–160. IEEE, 2011.

[VWF09] Fernanda B Viegas, Martin Wattenberg, and Jonathan Feinberg. Participatory visualization with wordle. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):1137–1144, 2009.

[WBS06] K.Q. Weinberger, J. Blitzer, and L.K. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems 19*, pages 10:207–244, 2006.

[WCR+01] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. Constrained k-means clustering with background knowledge. 2001.

[Wei99] Yair Weiss. Segmentation using eigenvectors: a unifying view. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 975–982. IEEE, 1999.

[WS08] K.Q. Weinberger and L.K. Saul. Fast solvers and efficient implementations for distance metric learning. In *Proceedings of the Twenty-Fifth International Conference on Machine Learning (ICML)*, pages 1160–1167, 2008.

[WSLA09] Fei Wang, Jimeng Sun, Tao Li, and N. Anerousis. Two heads better than one: Metric+active learning and its applications for it service classification. In *Ninth IEEE International Conference on Data Mining (ICDM)*, pages 1022–1027, Dec 2009.

[WTP+95] James A Wise, James J Thomas, Kelly Pennock, David Lantrip, Marc Pottier, Anne Schur, and Vern Crow. Visualizing the non-visual: spatial analysis and interaction with information from text documents. In *Information Visualization, 1995. Proceedings.*, pages 51–58. IEEE, 1995.

[WV08] Martin Wattenberg and Fernanda B Viégas. The word tree, an interactive visual concordance. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1221–1228, 2008.

[WZG+14] Ji Wang, Jian Zhao, Sheng Guo, Chris North, and Naren Ramakrishnan. Recloud: semantics-based word cloud visualization of user reviews. In *Proceedings of the 2014 Graphics Interface Conference*, pages 151–158. Canadian Information Processing Society, 2014.

[XdW05] Q. Xu, M. desJardins, and K.L. Wagstaff. Active constrained clustering by examining spectral eigenvectors. In *Discovery Science*, pages 294–307, 2005.

[XGH06] Ling Xiao, John Gerth, and Pat Hanrahan. Enhancing visual analysis of network traffic using a knowledge representation. In *Visual Analytics Science And Technology, 2006 IEEE Symposium On*, pages 107–114. IEEE, 2006.

[XNJR02] E.P. Xing, A.Y. Ng, M.I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*, pages 505–512, 2002.

[YHC09] Y. Ying, K. Huang, and C. Campbell. Sparse metric learning via smooth optimization. In *Advances in Neural Information Processing Systems 22*, pages 2214–2222, 2009.

[YJ06] L. Yang and R. Jin. Distance metric learning: A comprehensive survey. *Michigan State Universiy*, pages 1–51, 2006.

[YJ07] L. Yang and R. Jin. Bayesian active distance metric learning. *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence (UAI)*, 2007.

[YKSJ07] Ji Soo Yi, Youn ah Kang, John Stasko, and Julie Jacko. Toward a Deeper Understanding of the Role of Interaction in Information Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13:1224–1231, 2007. 6.

[YL12] Y. Ying and P. Li. Distance metric learning with eigenvalue optimization. In *Journal of Machine Learning Research*, pages 1–26, 2012.

[YMSJ05] J. S. Yi, R. Melton, J. Stasko, and J. A. Jacko. Dust and Magnet: Multivariate information visualization using a magnet metaphor. *Information Visualization*, pages 239–256, 2005.

[Yu10] Y. Yu. Anomaly detection of masqueraders based upon typing biometrics and probabilistic neural network. *Journal of Computing Sciences in Colleges*, 25(5):147–153, 2010.

[Zhu05] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

[ZK09] Caroline Ziemkiewicz and Robert Kosara. Preconceptions and individual differences in understanding visual metaphors. In *Computer Graphics Forum*, volume 28, pages 911–918. Wiley Online Library, 2009.

[ZO00] T. Zhang and F.J. Oles. A probability analysis on the value of unlabeled data for classification problems. *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pages 1191–1198, 2000.

[ZOC+12] Caroline Ziemkiewicz, Alvitta Ottley, R Crouser, A Yauilla, S Su, William Ribarsky, and Remco Chang. How visualization layout relates to locus of control and other personality factors. 2012.

[ZOC+13] Caroline Ziemkiewicz, Alvitta Ottley, R Jordan Crouser, Ashley Rye Yauilla, Sara L Su, William Ribarsky, and Remco Chang. How visualization layout relates to locus of control and other personality factors. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 19(7):1109–1121, 2013.

[ZXC03] Y. Zhang, W. Xu, and J. Callan. Exploration and exploitation in adaptive filtering based on bayesian active learning. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*, pages 896–903, 2003.