

Explainers: Expert Explorations with Crafted Projections

Michael Gleicher, *Member, IEEE*

Abstract— This paper introduces an approach to exploration and discovery in high-dimensional data that incorporates a user’s knowledge and questions to craft sets of projection functions meaningful to them. Unlike most prior work that defines projections based on their statistical properties, our approach creates projection functions that align with user-specified annotations. Therefore, the resulting derived dimensions represent concepts defined by the user’s examples. These especially crafted projection functions, or *explainers*, can help find and explain relationships between the data variables and user-designated concepts. They can organize the data according to these concepts. Sets of explainers can provide multiple perspectives on the data. Our approach considers tradeoffs in choosing these projection functions, including their simplicity, expressive power, alignment with prior knowledge, and diversity. We provide techniques for creating collections of explainers. The methods, based on machine learning optimization frameworks, allow exploring the tradeoffs. We demonstrate our approach on model problems and applications in text analysis.

Index Terms—high-dimensional spaces, exploration, support vector machines



1 INTRODUCTION

High dimensional data are typically explored using methods that select a smaller set of dimensions which are then examined with analytic and visualization tools amenable to small numbers of dimensions. These smaller sets are either chosen from the original variables using feature selection methods or generated by techniques that create new dimensions with mathematically desirable properties. Unfortunately, such approaches do not consider the background knowledge and preexisting questions about the data that the user may have. An emerging class of methods permits viewers to use their prior knowledge to organize the data. However, these methods do not help in understanding the connection between the results and data.

In this paper, we explore an analytical approach that supports exploration and discovery in high-dimensional spaces guided by the user’s knowledge and questions of the data. The core idea is to enable the user to craft sets of projection functions that reflect their expert knowledge and fit their needs by making simple annotations of the data. Central to our approach is to consider the different roles that projection functions may serve in exploration, the qualities that projection functions need to have in order to serve these roles, and the kinds of methods available to create and assess the functions. We term such crafted projection functions *explainers* because they encode a user-comprehensible relationship between properties that the user specifies and the underlying data. The user specifies relationships among the data objects, (e.g. certain documents are comedies or one city is more interesting than another), and the system constructs explanations of how this is supported in the data.

Projection functions map high dimensional data points into simpler spaces. They may be as simple as selecting one of the original dimensions (or variables) in the data, or may involve a complex relationship. Projection functions can serve two main purposes: they organize the data along an axis, and they describe the connection between the axis and the variables. When projection functions align with user knowledge, both roles are enhanced: views are organized in meaningful ways and illuminate connections between variables and known properties. These correlations between data variables and known facts provide evidence and theories for explaining knowledge in terms of the data. Sets of projection functions can provide a diversity of viewpoints, or multiple explanations of the same phenomena.

The central contribution of this paper is to introduce a new approach to the exploration of high-dimensional spaces that allows a viewer

to create sets of projection functions that define dimensions that are meaningful to them, have useful relationships with the variables, and work together to provide diversity or agreement as necessary. We provide a consideration of what qualities may be useful in projection functions for them to serve as explainers and a set of tools crafting them based on user-specified annotations.

There are tradeoffs in selecting projection functions. Users may prefer simpler functions (to enhance comprehensibility), better alignment with specified annotations, or more diversity among the selected functions. These tradeoffs, as well as the diversity of user needs and knowledge, demand an approach that involves the user in crafting the sets of functions. Because of the multiple challenges in specifying user goals and computing functions that meet those goals, we use a “generate and test” approach. Based on initial user specifications, our techniques generate a collection of candidate functions and provide options for sorting, filtering and selecting. The methods for candidate function generation build on techniques from machine learning, while the sorting and sifting strategy is akin to rank-by-feature methods. Our approach uses linear functions, as they offer a good tradeoff of expressiveness, comprehensibility, and computational efficiency.

Unlike methods that select axis-parallel projections, our approach can use more complex functions to provide flexibility in the kinds of dimensions created, allowing it to represent user-defined properties. Unlike standard dimensionality reduction techniques or random projections, our approach creates derived dimensions that are aligned with the user’s specifications, so that they have meaning to the user. Unlike machine learning approaches, our methods provide a richer output that is compatible with a wide range of visualization and analytics tools, and consider tradeoffs in usefulness for understanding.

We follow this introduction with a case study to make the concepts concrete, allowing a comparison with related work. The paper then discusses the qualities projections need to succeed in their multiple roles. We describe techniques to generate collections of projection functions, as well as ranking and filtering techniques to create interesting sets from these larger collections. We conclude with case studies exploring collections of historical texts. Further details on all examples in the paper and additional case studies are provided on the accompanying web page¹.

1.1 Case Study: City Livability

A toy dataset provides an example to make our approach concrete. Other case studies are provided in §6. The city livability data set [5] rates 140 cities on 40 categories; each city is represented by a point in a 40 dimensional space. We refer to these dimensions of the data as *variables*, and in this case they are measurements of quantifiable aspects of the cities, such as crime rate and quality of health care.

• Michael Gleicher is with the Department of Computer Sciences, University of Wisconsin - Madison. E-mail: gleicher@cs.wisc.edu.

Author’s preprint version.

This paper will be published by the IEEE, who will hold the copyright to the final version.

¹<http://www.cs.wisc.edu/graphics/Vis/Explainers>

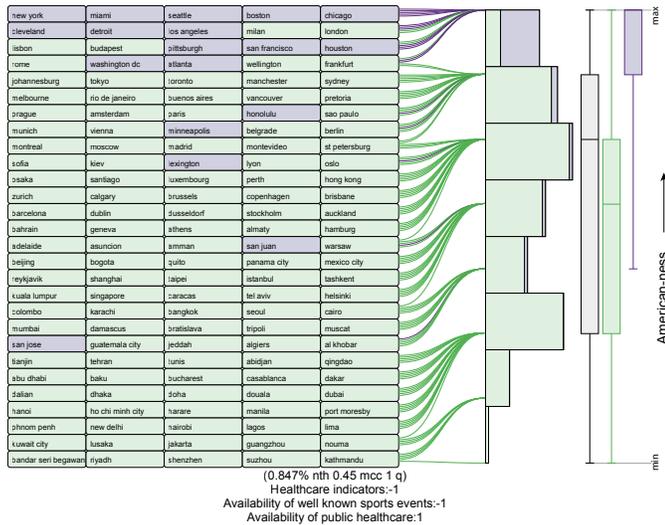


Fig. 1. A visualization of a projection of the city data that explains “American-ness” in terms of livability indices. The function is: $x_1 - x_2 - x_3$, where x_1 is availability of public health care, x_2 is availability of sporting events, and x_3 is a measure of healthcare quality (lower numbers are better). The function is correct quite often (scoring well on the metrics in §3.1). The visualization (described in §5.2) shows the rank order list of cities (in reading order), the mapping from cities to the projected dimension (thin lines connect from a rank scale to the value scale, which is highly quantized due to the data), a stacked histogram with uniform bin size, and box plots of the whole data (gray) and the two classes (USA and not). Blue cities have been marked as in the U.S.

American-Ness: Location is not encoded in the data, it is “expert” user knowledge. The user can mark the cities in the U.S. and the system generates derived dimensions that correspond to “American-ness.” Given the specified binary predicate, the methods of this paper can generate a set of projection functions that are aligned with it. The property of American-ness is defined by *example* in object space: cities in the US should have more of it than cities elsewhere.

If the functions that determine American-ness from the data are simple, they can expose relationships between the measured quantities and the user-specified property. Unfortunately, there is no way to know *a priori* how simple a function can be and still adequately capture the property, nor is there a clear limit of how simple a function must be to be comprehensible. Our approach generates many alternatives and allows the user to choose the appropriate tradeoffs for their needs. For the present example, a 3 variable linear function provided a good tradeoff: one or two variable functions do not achieve good correctness, while larger numbers of variables provide diminishing returns (the best 7 variable classifier still makes at least one mistake, despite being hard to understand). To further improve comprehensibility, our approach allows restricting the weights of the linear function to small integers. While the best (in terms of correctness) of these three variable functions still misorder some cities, the cities in the US are almost always scored higher (see Figure 1 for an example). The “mistakes” are sometimes interesting: for example, San Jose is often a negative outlier (Figure 2), is it San Jose, Costa Rica, not California?

Crafted projections, such as American-ness, serve two main roles. First, they help organize the data, providing a nameable axis that can be used in either standard displays, such as parallel coordinates and scatterplots, or in specialized views (Figure 1). We can use these to identify exemplars, outliers, and trends. Second, because the projection functions are simple, they provide ways to explain the connection between the property (American-ness) and the variables. Examining a function (Figure 1), we can see that American cities are distinguished by having good health care indicators and availability of sporting events, but poor public health care. Because our approach can generate many projection functions, we can find multiple *explanations* of American-ness according to the data (see Figure 2). If the user’s goal is explanation, they may stress simplicity over correctness.

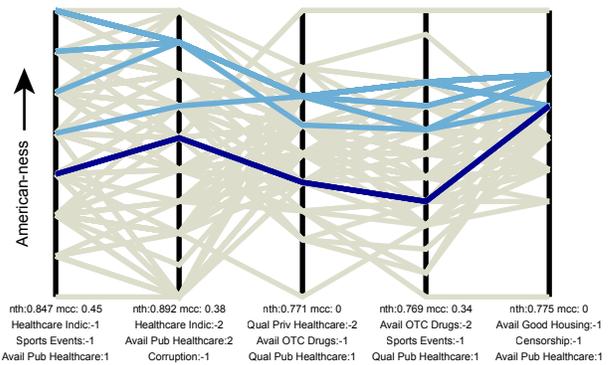


Fig. 2. A parallel coordinates plot showing 5 diverse explainers of American-ness. The leftmost axis is Figure 1. Cities marked as being in the U.S. are colored blue. The U.S. city that consistently has the lowest American-ness (San Jose) is marked in dark blue.

Another advantage to our approach is that we can create dimensions to meet specifications. We can easily specify different properties to create (such as “European-ness” or “has-hosted-the-olympics-ness” or “contrast Western Europe and North America”). We can also add additional constraints beyond just the groupings, such as insisting that particular cities appear in specific places in the list. We can create views of “American-ness” that require New York to be on top.

Exemplars: We can also use our approach to understand exemplars. For example, to understand “What makes Paris different from other cities?” one can define dimensions of “Paris-ness,” where Paris scores higher than other cities. While no single variable distinguished Paris well, two and three variable functions perform well, with the diversity of functions providing a variety of perspectives on what makes Paris unique. We can also use such “exemplar dimensions” as a redundant basis for looking at the data. For example, graphing Paris-ness vs. New-York-ness shows that while the two are correlated ($R = .63$), the data still distributes interestingly in a scatterplot (Figure 3). Unlike other projections (like PCA), the axes of the graph are namable properties. If orthogonality of axes were important, we could look for cities whose axes were uncorrelated to define a basis, for example Paris and Shanghai have axes that are uncorrelated ($R = .03$), even though Paris and Shanghai score highly on each other’s scales.

To further understand “Paris-ness” we might want to distinguish what is specific to Paris, versus what is French. As there is one other French city in the data (Lyon), we can look for explanations of Paris-ness that score Lyon highly (some projections have Lyon tied with Paris for the most Parisian city in the world), and explanations that do not. Examining these simple functions, we see that some explanations of Paris-ness consider its climate and public transport (which Lyon does not share according to the data), while others focus on housing and health care, where Lyon scores similarly.

Approach Summary: Our approach creates *explainers*, projection functions crafted to encode the relationship between a user-specified property and the data variables. The projections are sparse, quantized linear functions. Our approach generates sets of functions, allowing the user to make tradeoffs between correctness, simplicity, and diversity. Our methods generate these sets by first creating a large collection of candidate functions using feature selection to choose sparse subsets of the variables, computing the best function using each of these subsets, and generating different quantizations of each. The user can then select appropriate functions by filtering and sorting based on a variety of properties of these functions. For example, to create the American-ness example, feature selection (§4.2) generated over 700 3-variable functions, each of which was quantized (§4.4) to several different levels creating a collection of several thousand functions. These were filtered (§5.1) to remove ones that did not perform sufficiently well (§3.1) and then a subset was chosen that included the highest correctness and others that were as uncorrelated as possible (§4.2).

various constraints, integrate well with feature selection, are widely studied, and have excellent implementations available. Other candidate approaches, such as logistic regression or random forests, do not appear to offer all of these advantages.

Exemplars: The idea of exemplar SVMs, which inspire our exemplar functions, was first developed by Malisiewicz et al. [32]. Their exploration of the visual uniqueness of Paris [10] inspired our Paris example. This work points out the advantages of exemplar classifiers, over the more obvious approaches (such as measuring the distance from the example). These advantages include not needing a good distance metric and allowing the exemplar not to be the extrema (e.g. a city could be more Parisian than Paris).

Distance Metrics: Projection functions, which map high-dimensional to lower dimensional points, are different from distance functions, which measure the distance between a pair of points. There is an emerging field of metric learning that is developing method for determining these functions from specifications, see Yang et al. [51] for a survey. Brown et al. [3] use metric learning to provide an interactive dimensionality reduction. Our approach is useful for determining distance metrics indirectly, as will be discussed in §5.3.

3 PROPERTIES OF EXPLAINER FUNCTIONS

For any given specification, there are an infinite variety of projection functions that might be considered. This section enumerates these various tradeoffs, and provides our metrics for assessing projection functions according to them. Subsequent sections introduce methods that generate collections functions that span these tradeoff and filtering and sorting methods that use the metrics to select appropriate ones.

3.1 Correctness

A core idea of explainers is that they are generated based on user-specified annotations of the data, used to convey prior knowledge and questions of interest. The primary form that these specifications take is the differentiation between two groups, such that members of one group have more of the resulting property than members of the other (e.g. defining “American-ness”). The ability to choose sets provides a flexible mechanism to specify prior knowledge and comparison questions. The sets may be partially specified (some elements are unspecified). Specifications may also include other types of constraints that relate objects. For example a user may specify that an element has the most or least of a value (e.g. New York is more American than other cities), or relate objects (e.g. Boston is more American than Miami). In the future, we hope to consider other ways to specify properties.

The property of *correctness* measures how well a function represents the specified property. There are many ways to assess the correctness of a projection function. In general, it is difficult to capture the performance of the function over the range of the data in a single number, see [47] for a survey of the issues. For the binary set property definitions, we might begin with the definition that elements of the positive set (denoted \mathcal{P}) should have higher values than elements of the negative set (\mathcal{N}), so

$$\forall i \in \mathcal{P} \forall j \in \mathcal{N} f(i) > f(j).$$

For a given classifier, we can compute the percentage of time this is correct over the data (out of the $i*j$ possible), which we call the Non-Threshold Metric (*nth*).

Most correctness metrics involve a threshold. That is, $\forall i \in \mathcal{P} f(i) > th$ and $\forall j \in \mathcal{N} f(j) < th$. When considered this way, the projection function serves as a classifier: it can predict for any data element which set it is in. The choice of threshold affects the error profile. For prediction applications, determining the classification threshold is a subtle decision as it must weigh the costs of false positives and negatives in the unseen data. For data explanation applications, we can choose the optimal threshold by examining the data. Generally, we set the threshold for a classifier by finding the value that achieves the highest accuracy.

Once the threshold is set, there are a variety of ways to quantify performance [37]. Basic metrics such as accuracy, precision, recall and F1 do not capture the error profile, and are problematic when the

sets have different sizes. While we provide all of these options to the user, the most often used is the Matthews correlation coefficient (*mcc*) which accounts for the relative sizes of the classes to measure how much better a classifier is than chance.

The metrics mentioned so far are binary: they only consider whether each decision is made correctly, counting the number of true and false results. Metrics can consider how bad the mistakes are – the loss – by accumulating the amount of error (either sum of squares or sum of absolute values). While such metrics are central to determining projection functions, they are less useful for interpretation. For the examples in this paper, loss functions are considered only internally to the optimization algorithms for determining projection functions.

Achieving perfect correctness is not always possible, or desirable. The data may not actually provide sufficient information to distinguish the groups, or tradeoffs with other qualities may make it advantageous to sacrifice some correctness in order to achieve other goals. Getting some portion of things wrong is often not a problem: indeed, we find that classification “mistakes” are often interesting data elements to examine (e.g. San Jose in the example).

Unlike machine learning, which seeks to make predictions about unseen data, our approach seeks to explain the given data. Therefore, correctness metrics are applied data itself, rather than using schemes such as cross-validation to predict performance on unseen data.

3.2 Simplicity and Explanatory Power

While *simplicity* is an intuitive property of a function, basically measuring how complicated it is to represent or compute, it is difficult to measure precisely.

Simplicity of projection functions takes many forms. First, is the class of function. We limit our approach to linear functions. Such functions are easy to interpret and afford efficient algorithms. Not all linear functions are equally simple. We consider three factors: sparsity, quantization, and familiarity of the variables. *Sparsity* describes how many variables the linear combination has. Linear combinations of smaller number of variables are simpler than those with more. Sparsity is measured by the number of non-zero weights. *Quantization* considers the complexity due to the weights themselves. A simple combination of weights is easier to interpret (e.g. $x + y - 2z$ rather than $3.654x + 3.234y - 6.43z$). We prefer weights to be integers and small. We measure the quantization of a linear combination as the largest absolute value of a weight when the linear combination is represented with integer coefficients. *Familiarity* considers that not all variables are equally easy for the viewer to understand. Our collaborators repeatedly make statements like “I’d rather see two variables I am familiar with than one that I am not.” Our methods allow familiarity metrics by adding user-specified weighting to the sparsity metric.

Simpler classes of functions are less expressive than more complex ones. Therefore, choosing a simpler function may cost the ability to accurately meet a specification but provides two key benefits. First, the loss of expressive power may be gained in explanatory power. Simpler functions are easier for a viewer to understand, allowing them to make inferences and build theories from the observed correlations. Second, requiring simple functions are a way to avoid over-fitting. As a sufficiently expressive class of functions can encode any relationship, the existence of a complex relationship between a property and the data may say little about that relationship.

Seo and Schneiderman [39] argue that linear functions (beyond single variables) are too complicated for viewers to interpret. However, the choice is not binary (single variables vs. arbitrary linear combinations), as they suggest. Sparse and quantized linear combinations provide useful tradeoffs of simplicity and complexity. The ability to express useful, viewer specified concepts make the tradeoff worthwhile.

3.3 Statistical Distributions

A function that provides a perfect classifier, returning 1 or -1 for each data element, may not be useful for organizing data. If our goal is to organize the data, projection functions that spread the elements along the dimension are valuable. The idea of using the quality of the distribution as a measure of interestingness is central to the projection

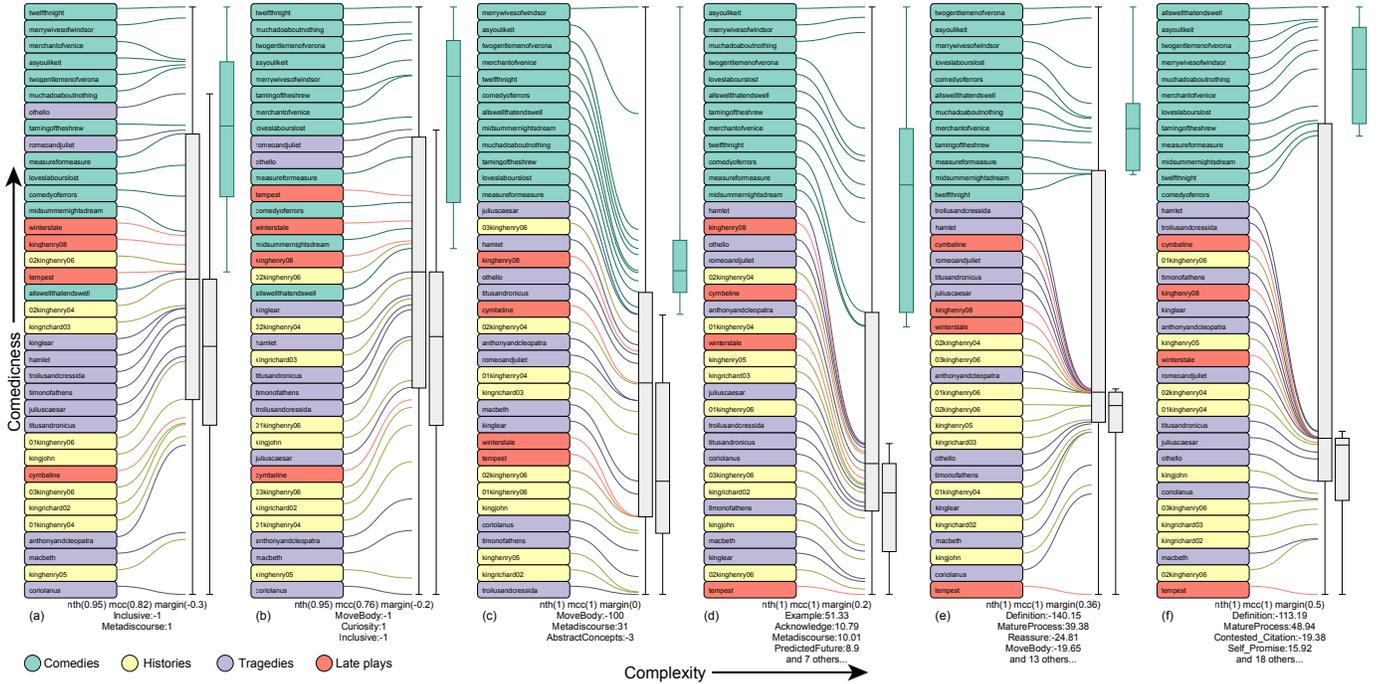


Fig. 4. A set of different explainers for comediness in Shakespeare’s plays (§6.1) visualized as described in §5.2 or Figure 1. Some very simple functions with 2 or 3 unit weights get good performance (a,b) and make “interesting” mistakes. Perfect correctness can be achieved with 3 variables (c), requiring very specific weights. Support vector machines can generate larger margins (d,e,f), but more complicated functions.

pursuit concept [16]. Various notions of interestingness are provided by the ranking frameworks [25, 39, 44]. These metrics can be used together with the other kinds of metrics we discuss in this section.

When working with collections, there is often a desire to identify individuals that can summarize the collection. Having a meaningful projection function allows us to identify epitomes (the elements that have the most of the property), the typical (e.g. the mean, median, or mode), and limits (the elements that define class boundaries).

3.4 Diversity

The property of *diversity* measures how similar functions are. Having a diverse set of projections, allowing for some measure of redundancy, provides alternative viewpoints, suggests different theories, and can improve robustness. We consider two major types of diversity between projection functions: diversity of inputs (variables each consider), and diversity of outputs (the distribution of values produced when the function is applied to the data set). The two are independent: it is possible to obtain the same outputs with different inputs, and two functions can combine the same variables in different ways to achieve very different outputs. To measure the difference in outputs of two functions, we use the correlation coefficient (Pearson’s R). To measure the difference in the inputs of two functions, we compute a variable overlap metric: the portion of each classifier’s variables that are used by the other.

4 GENERATING PROJECTION FUNCTIONS

We seek methods that create a variety of functions that allow for control of the tradeoffs among correctness, simplicity, statistical interest, and diversity. We first consider how to create correct functions, and then examine how to use this machinery to create a range of tradeoffs. Because it is difficult to provide exact control of some of the properties, we instead prefer techniques that sample the space of functions: producing collections of functions each representing different tradeoffs. Given this collection, we can apply sorting and filtering methods to select appropriate ones.

In our approach, projections are linear functions denoted as $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$, where \mathbf{x} is a vector containing a data element with m variables. The function is encoded by a weight vector \mathbf{w} and a scalar offset b . We will sometimes write this using homogeneous coordinates,

concatenating a 1 to the \mathbf{x} vector and the offset to the \mathbf{w} vector, e.g. $f(\mathbf{x}) = \tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}}$. We denote data element i as \mathbf{x}_i .

4.1 Function Creation as Constrained Optimization

The basic form of explainer specification gives a binary predicate of some elements being in the positive set \mathcal{P} and others to be in the negative set \mathcal{N} . The machine learning literature provides a wide variety of methods for determining classification functions for such problems. We build on the well-known Support Vector Machine (SVM) method (for an introduction, see books such as [47] or [41], or tutorials such as [20]). We provide a brief review here to allow us to introduce notation so we can discuss issues later.

An explainer function can be shifted such that 0 divides the classes, leading to a constraint on each data element i :

$$\forall i \in \mathcal{P} f(\mathbf{x}_i) \geq 0 \text{ and } \forall i \in \mathcal{N} f(\mathbf{x}_i) \leq 0. \quad (1)$$

For notational convenience, we define a label vector \mathbf{y} , such that y_i is the label associated with object i . We use the convention that 1 means it is in the \mathcal{P} , -1 if it is in \mathcal{N} , and 0 if the item is unlabeled. This allows re-writing Equation 1 as:

$$y_i \tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}}_i \geq 0. \quad (2)$$

Of the possible solutions for $\tilde{\mathbf{w}}$, the smallest one is chosen (for reasons below). Minimizing the $L1$ (sum of absolute values) norm of $\tilde{\mathbf{w}}$ or the $L2$ norm both lead to optimization problems that can be solved efficiently. The $L1$ norm is often used as an approximation to the $L0$ (number of variables) norm, and leads to sparse solutions.

The linear inequalities allow the trivial solution ($\tilde{\mathbf{w}} = \mathbf{0}$) to the minimization. To avoid this, rather than simply require that the classes be separate, we try to create a non-zero margin between them by replacing the 0 in Equation 2 with a 1. This gives the minimization problem

$$\tilde{\mathbf{w}} = \operatorname{argmin} \|\mathbf{w}\|^r \text{ subject to } y_i \tilde{\mathbf{w}} \cdot \tilde{\mathbf{x}}_i >= 1 \quad (3)$$

where r is the regularization norm (1 or 2). This minimization is known as a Support Vector Machine (SVM). Because SVMs attempt to minimize \mathbf{w} , they also minimize the range of the resulting values.

Because the absolute size of the margin is fixed, its relative size (to the overall range) is maximized. SVMs are maximum margin classifiers. Figure 4 illustrates a range of margins.

Allowing for some error in the constraints makes them always feasible, and permits controlling a tradeoff in the amount of error. We add to each constraint an error term e_i , yielding

$$y_i \bar{\mathbf{w}} \cdot \bar{\mathbf{x}}_i + e_i \geq 1, \quad (4)$$

and add constraints to enforce $e_i \geq 0$. The magnitude of these errors is called the loss and should be minimized. The complete SVM is therefore

$$\bar{\mathbf{w}} = \operatorname{argmin} \alpha \|\mathbf{w}\|^r + \beta \sum e_i^2 \text{ subject to } y_i \bar{\mathbf{w}} \cdot \bar{\mathbf{x}}_i + e_i \geq 1, e_i \geq 0 \quad (5)$$

Weighting terms α and β allow for a tradeoff between keeping the weights small and keeping the errors small. As written, this provides a single knob, as it is the ratio of the two that controls the tradeoff between correctness and (a form of) simplicity. We denote this parameter as c , where $c = \beta/\alpha$. We avoid exposing these parameters to the user, selecting them automatically (§4.2).

Solving: Equation 5 is a quadratic program. Standard SVM solvers (our implementation uses LibLinear [14]) can solve these problems efficiently by using specialized techniques.

If other relationships between between data elements are specified, these too can be posed as linear inequalities (e.g. $f(\mathbf{x}_i) > f(\mathbf{x}_j)$) and added to Equation 5. However, this is no longer a standard SVM, and requires using a general purpose quadratic programming solver (our system uses the CVXOPT module for Python). In our experience, such solvers are considerably slower and less robust than the specialized SVM solvers. All examples in this paper use LibLinear.

The SVM provides the offset (b) that centers the margin. However, because we seek to optimize performance on the known data (rather than generalization), we can re-compute it to produce an optimal result. After computing $\bar{\mathbf{w}}$, we replace the offset with the value that achieves the highest possible accuracy. This can be done in a linear scan of the sorted data. All examples in this paper do this post-hoc optimization of accuracy, although we typically prefer to use non-threshold metrics so the offset does not matter.

Families of Functions: Central to our approach is to create sets of functions that capture different tradeoffs. Using the SVM (or equivalent) machinery, the two primary ways we generate families of functions are to vary the parameter (c , as discussed in the section on L1 feature selection below), or to select different subsets of the columns of the data matrix and compute SVMs for each (as used in the section on exhaustive feature selection, below). A related approach is to apply a projection pursuit style algorithm: computing a linear function, factoring it out of the data matrix, and then repeating the process on the rank-reduced matrix. We term such a method *support vector pursuit*. In practice, we find it ineffective. First, orthogonality of the functions generated is rarely a principal goal. Second, finding good functions in the increasingly rank-deficient matrices becomes poorly conditioned. Third, SVMs are most effective if the data matrix is whitened, which can undo the rank reduction. Instead, we almost always use the feature selection methods below to generate a family of classifiers, and then filter this list to remove redundancy.

4.2 Feature Selection

The problem of finding \mathbf{w} can be viewed as having two parts: selecting which weights will be non-zero (e.g. which variables will participate), and then determining the values of these weights. The former task is known as *feature selection*. See [19] for a survey of the issues and main approaches.

To provide control over the tradeoff between simplicity (in terms of sparsity) and correctness, we use the different results of feature selection, and compute the weights that achieve the best correctness given each set of variables. This step can be accomplished by solving the SVM with a subset of the variables (i.e. selecting a subset of the columns of the data matrix).

We have found that the naïve strategy of computing dense functions and choosing a subset of variables by picking the ones with the highest loadings performs poorly. Instead, we employ two other strategies for feature selection to produce collections of candidate feature sets. The first is to select features as part of the SVM solution. The second is to exhaustively create candidate feature sets, and then solve an SVM over each of these to determine the candidate functions. These methods have complementary advantages, so we usually combine their results.

L1 SVMs: As mentioned previously, when we use L1 ($r = 1$), low values of c will tend to produce sparser results, at the expense of greater error and smaller margins. Our strategy samples a variety of values of c to generate a collection of functions that have different tradeoffs. We use a binary search to find different values of c that produce functions. We begin by bracketing c with the minimum value (0, which produces the “null” function of zero variables) and some large value. Intervals of c are recursively subdivided if the functions generated for the upper and lower levels are different, or a specified recursion depth limit is reached. After computing each SVM to determine the active variables, we solve a second SVM over the reduced set of variables with a larger value of c to find correctness levels at the same level of simplicity. This technique, known as *debiasing* [49], improves the results. Because this second solution occurs over a smaller set of variables, its computational cost is low.

The repeated use of L1 SVMs generates a family of functions with a tradeoff between simplicity and correctness/margin. This can be useful in determining how many variables are required to adequately express a specification. However, it suffers from two major flaws. First, it provides only a single function for a given level of sparsity. Second, while the varying c is usually effective at helping to find sets of variables capable of achieving high levels of correctness (potentially using many variables), empirically, it is less effective for extreme tradeoffs where we are willing to accept larger amounts of error to achieve very sparse classifiers. This is related to the fact that the L1 norm is only a heuristic approximation for our L0 goal. The next method is good for finding small feature sets.

Exhaustive Search: To complement the searching c approach to feature selection, we also use an approach with complementary features: exhaustive search of the combinations of variables. Because we are often interested exploring the entire space of linear functions with small numbers of variables, such a brute force approach can be attractive. While such an approach may need to solve a very large number of SVM problems (to determine the weights for each combination of variables), each of these problems is very small (just a few variables) and can be solved quickly. The examples from §1.1 (cities in the USA) have 40 variables, so there are $\binom{40}{3} = 9880$ sets of variables to try. This takes about 30 seconds on a Dell XPS12 Laptop in our non-optimized Python implementation.

The combinatorial explosion precludes using the totally exhaustive approach for data with more variables, or if we want to consider functions with more than 3 variables. We therefore use a heuristic to make the search more tractable. We apply a greedy approach: to combine the simplest functions that perform best. For example, rather than considering all 40 variables to combine, we consider only the 20 that perform best as single variable functions to create $20 \times 19 = 380$ pairs. In creating triples, rather than adding to all of these pairs, we consider only the top 40 of them, combining them with the 20 best variables to create approximately 800 three variable functions (the number is approximate since there will be redundancy - the 20 variables already appear in the 2 variable functions). Our default settings use the top 25 variables to determine the best pairs, and the top 30 pairs and 30 variables to determine the triples to consider. These settings were not chosen in a principled way and may need to be adapted for problems with different numbers of variables (we have tried it on examples with between 40 and 150). We use the n th metric to select the “best” functions.

The greedy heuristic does not guarantee the optimal solution [19] as weak variables can combine to make good combinations. We justify our use of the greedy approach in several ways. First, we are not necessarily seeking the optimal combination of variables, but rather, to find a diverse sampling of good classifiers. Second, while weak func-

tions (or variables) may combine in ways that perform well in terms of correctness, this is often counter-intuitive or involves less familiar variables, and therefore may be more challenging in terms of explanatory power (although, there is an argument for exposing a viewer to variables they may not have thought about).

Empirically, this method works well, even without any tuning of the default parameters. For the American-ness example, heuristic culling considers 706 three variable functions (rather than 9880), and takes about a second. The best function achieves a 92.3% *nth* score, while the exhaustive approach is able to find one with a 92.9% score. The function found by our approach is the second best one found exhaustively. However, the greedy approach allows us to explore explainers with more variables, finding a 4 variable function scoring over 96%, a 5 variable function scoring over 98%, and several 7 variable functions that score over 99%. Computing all these functions (up to 7 variables) takes about as long as the fully exhaustive search, but has the advantage that it also produces the best classifiers with smaller numbers of variables in the process, allowing the user to make a decision as to how much of a tradeoff to make between correctness and simplicity. All examples in this paper with 3 or fewer variables use this approach with the default parameters, unless specified.

The greedy heuristic only works for a few variables. The SVMs have more variables so performance becomes an issue, and it exhausts its supply of useful variables. However, for these more complex classifiers, the L1 methods above work well.

4.3 Additional Constraints

Additional constraints, such as requiring a particular element to have the highest value, can be handled in two ways. First, they can be treated as hard constraints and added to Equation 5. This gives the additional constraints priority over everything else. However, this approach has a number of disadvantages. First, it provides no way to relax the constraints to provide tradeoffs with other aspects, including the correctness of the other parts of the specification. For instance, it could be that allowing the element to be the second highest value (rather than the highest) can be accomplished with a far simpler function. Second, this approach produces a single function, not a diversity of solutions. Third, this approach requires the use of a general optimization solver, rather than the specialized SVM solvers.

A second approach to the additional constraints is to use them as soft constraints. We generate a collection of possible functions, and then either choose those that best satisfy the constraints, or use a threshold to cull those that do not meet the constraints well enough. This approach has complementary properties to the first: while there is no guarantee that it will find a function that meets the constraints exactly, it can find multiple functions that meet the constraints in various ways, and it is agnostic to the way the initial functions are generated.

4.4 Quantization

The quantization level of a linear function offers another simplicity tradeoff. As we restrict the range of values we permit for the weights, the potential expressivity of the functions becomes smaller, and the correctness performance may decrease accordingly. We are unaware of any practical method for directly optimizing integer weights as part of the SVM computation. Therefore, our strategy is to compute functions with continuous variables, and then to try a variety of different quantization levels.

To quantize a linear function, we begin with the quantization level one (e.g. each weight must either be 0, 1 or -1), and consider higher levels of quantization until we reach a level at which the quantized function achieves the same performance as the real-valued one. This provides a set of functions with different quantization level vs. correctness tradeoffs. This process is applied to each member of the families of classifiers produced during the feature selection process.

5 USING PROJECTION FUNCTIONS

The methods of the previous section generate a large collection of projection functions for a given specification. Combining parameter search for L1 SVMs, near-exhaustive search of 1- 2- and 3-variable

functions, and possibly solving for several different combinations of additional constraints, leads to a collection of a few hundred candidate functions. Quantizing each to several different levels yields up to a few thousand candidates. While many are very similar, the subtle differences represent different tradeoffs. From this large collection, smaller subsets can be selected.

5.1 Filtering and Sorting

Rank-by-feature and scagnostics approaches allow interactive exploration of a large collection of potential views. We apply this approach with a few alterations. First, we consider more than just the properties of the distribution, we consider all of the tradeoffs discussed in Section 3. Each projection function can be ranked by its properties including the number of variables, the quantization level, as well as a variety of metrics of correctness (*nth*, *mcc*, accuracy, margin, t-test confidence level). Second, we generally have a much larger number of projections to consider.

Our tools can create a scagnostics style scatterplot matrix, where each point is a projection and the variables are the various metrics. We select a subset of the metrics (to keep the scatterplot matrices reasonable). Such views are useful in determining what tradeoffs need to be made for a given specification. They allow determining what quantization levels and numbers of variables are required to achieve desired levels of performance, and how many different functions are likely to achieve desired levels. This allows the user to make informed decisions for filtering and selection.

After viewing the range of possibilities, the typical workflow uses filtering to restrict the collections to functions that are acceptable. This new list is then sorted by whatever metric is most important to the user. Correctness measures are the most common sorting metric, but sometimes ad hoc metrics are valuable, such as the ranking of a particular data element (e.g. Lyon in the introductory example).

From this ordered list of acceptable projections, our methods select a set that is as diverse as possible. As mentioned in Section 3.4, we use one of two metrics: the correlation between projections, or the overlap between their variable sets. For a selected metric, we apply a greedy algorithm: selecting the first function in the list (which is the one rated best according to the current sort); choosing the function that is most different from the ones selected, determined by summing the difference metric with all elements of the selected set; and repeating this process, adding new members to the selection set, until the desired size is reached. A different greedy algorithm variant selects the first function on the list, removes all functions considered to be too similar to it (by thresholding the metric), and then adding the first remaining function to the selection set. This process prefers functions that score higher on whatever ranking criterion is used, but it does require choosing a hard threshold for similarity.

5.2 Visualizing Function Sets

Visualizing the projection functions is a standard problem in displaying distributions for comparison. We rely on conventional methods including scatterplot matrices, heatmaps, and parallel coordinates. One potentially unique challenge with a crafted projection function is that we may be interested in the specific items - they are not anonymous points. While we augment our graphs with mouseover popups, this does not allow for quick assessment of where particular elements are.

Our standard presentation for inspecting a projection applies a number of visual encodings (Figures 1, 4). First, we use position to encode rank, which is useful to get a sense of the ordering given by a projection, and to assess its correctness (when color coding is applied). Second, we encode value with position, akin to a parallel coordinates display. Splines connecting the rank view to the value view give a sense of the density. Third, we show a histogram of the density, using stacked bars to give class distinctions. Finally, we add a boxplot to help convey a sense of how well the classes are separated.

Unfortunately, our standard presentation does not scale: even at the scale of 140 cities (Figure 1), the list view begins to break down. In such larger cases, we switch to a colorfield representation where we use position to encode data element (for example, in sorted order),

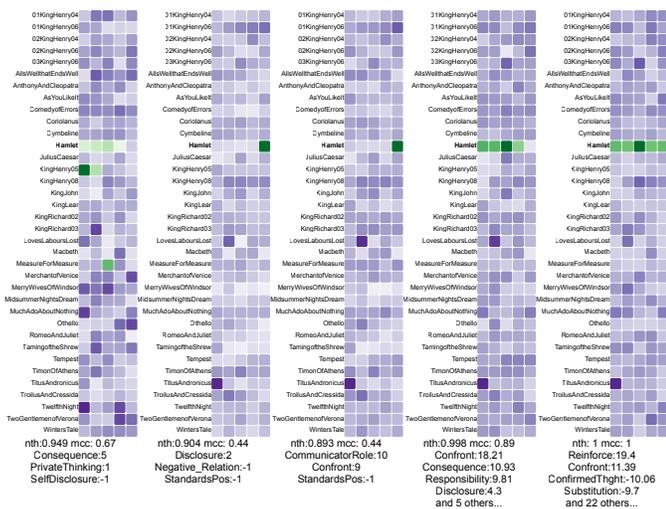


Fig. 5. Projection functions for dimensions of “Hamlet-ness” displayed using a color field visualization. For each of the 5 acts of the 36 plays, the colored square encodes the value using a purple-green diverging scale. Shades of green represent positive amounts of Hamlet-ness, while purples represent negative amounts.

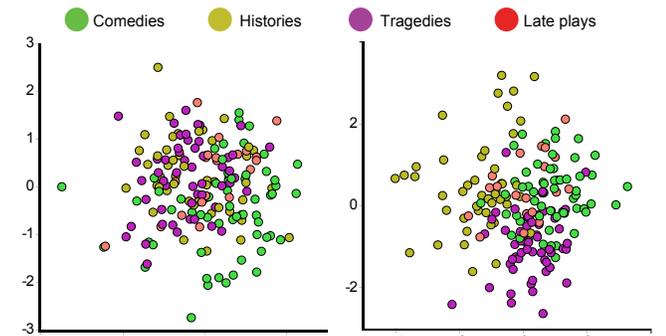


Fig. 6. Using derived axes as for a distance metric. Left: a multi-dimensional scaling (MDS) of the distance matrix (Euclidean in 115-dimensions) of the original data points in the acts of Shakespeare data set. Right: an MDS of the distance matrix using the 4 genre dimensions.

and use color to encode value (see Figure 5). Such ordered colorfields allow for quick location of specific elements. Summary judgments can be made quickly over colorfields [7].

5.3 Generating Distance Metrics

Creating effective distance metrics for high dimensional spaces is difficult. Having a large number of dimensions makes a distance ambiguous, as there are many different directions this distance could be in. The different dimensions may require different scalings, and cross-correlations may require scaling as well. We can use our approach as a user-guided dimensionality reduction to create distance metrics that reflect the users knowledge.

Figure 6 shows an example of using our approach to generate a distance metric for the Shakespeare acts data set (§6.1). We begin with 180 points (36 plays, each with 5 acts) in 115 dimensions. We create new dimensions that correspond to the four genres of the plays, and then use Euclidean distance in this space. This effectively creates a weighting matrix (akin to the Mahalanobis distance), similar to what a metric learning process would have produced. The distance matrix from our approach is not simply a diagonal matrix (as in [3]), but has sparsity (the dimensions that it was built from only consider 3 variables each). While it should be expected that the classes are better separated with our distance metric, the patterns and outlier that emerge fit with what is known about the plays, helping to confirm that the differences in genre are reflected in the data (see §6.1).

6 CASE STUDY

Literary scholarship has been a motivating domain for the development of our approach. To “distantly read” a large corpora, scholars abstract each text as a vector of numbers. This allows analytical techniques to be applied across a corpus. Our collaborators seek to consider the stylistic variation in writing, without regard to the content. To do this, they process texts with a text tagger that counts the number of occurrences of words of various types. The particular text-tagging scheme used in our examples is Docuscope [24, 28], which groups words into about 100 rhetorical categories. The categories were devised to capture common kinds of writing in the twentieth century. For example, “sense objects,” “motion,” “positivity,” “argument,” “emotion,” “first person,” and “common authorities” stand among the list of Docuscope’s categories. The categories are not precisely rhetorical labels, nor are they precisely all from the same register in the hierarchy of speech analysis but they distribute words in modern English such that each word can be said to be belong most strongly to one of these groupings. We have created our own version of the Docuscope tagger that is able to operate on larger corpora and longer texts.

The result of Docuscope processing is that any text is represented by a vector of measurements (115 for the version of Docuscope we use) that measure the amount of usage of various word types. Many of the scholarly questions considered involve establishing the relationships between “high-level” concepts (such as the genre) and the “low-level” details of word usage. The individual Docuscope variables are not of direct interest - unless they happen to relate to some high-level concept of interest. That is, we may be less interested in the low-level measurement like “sense objects” (how often that verbal activity is used), and would prefer features that are aligned with concepts of interest, such as genre. Prior work has explored the collections of points in 115 dimensions using factor analysis [6], principal component analysis [46], or clustering [22]. Specific tools have been built for these explorations (e.g. [8]), but they generally apply standard statistical methods (like PCA) and focus on workflow issues.

Our approach allows a scholar to use Docuscope data to find connections between high-level concepts (their expert knowledge) and the word usage statistics, as well as to use the Docuscope data to organize corpora according to concepts of interest.

6.1 Shakespeare’s Plays

Shakespeare’s 36 plays provide a familiar testing ground for our approach. Docuscope has proven to be a useful tool in studying them [22, 46]. Concepts in Shakespeare are well known; however, recent numerical scholarship has attempted to connect this knowledge with the specific details of word usage, such as the Docuscope data.

The connection between genre² and word usage is a common topic for numerical scholarship. It is not obvious that different genres would be written differently: is a tragedy just a comedy with an unhappy ending? Witmore and Hope [46] have shown that the different genres have different word usages, but their PCA analysis does not explain these differences.

To create a dimension of “comedicness” (or another genre), we create projection functions where the comedies score higher than the non-comedies. Several such projections are shown in Figure 4. The simplest classifiers get some plays “wrong” in an interesting way: for example, Romeo and Juliet scores high on the comedicness scale — with a different last few pages, Romeo may have awakened giving a non-tragic ending. Similarly, scholars have noted that Othello is like a comedy until the murdering begins [46]. These simple projection functions give specific patterns of rhetorical forms that lead to comedic-ness, allowing a scholar to develop theories of causality. The four genres form a non-orthogonal, but interesting, set of basis dimensions for examining the data (Figure 7). Significantly, the projection functions are simple enough to allow scholars to form causal theories, and the ability to generate multiple projections (not just one best one) lets scholars select ones that aid in their theory building.

²Shakespeare’s plays are commonly grouped into 4 genres, comedies, histories, tragedies, and late plays.

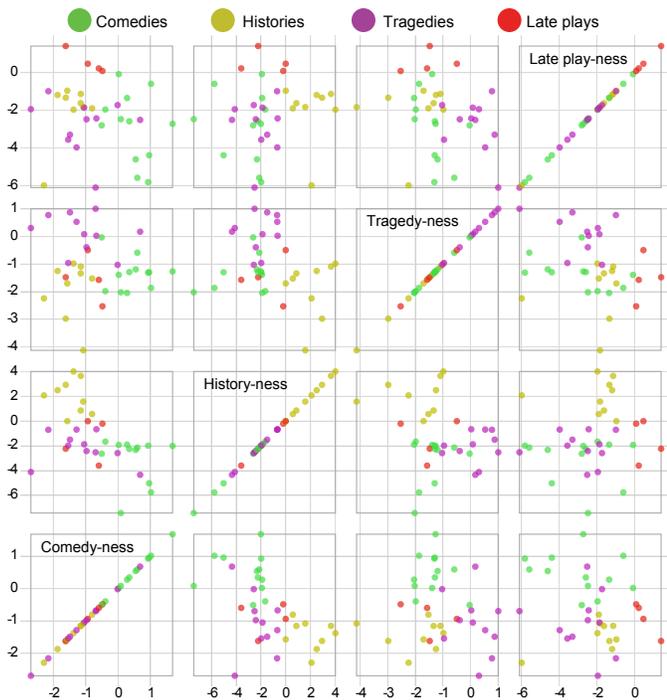


Fig. 7. A scatterplot matrix plotting Shakespeare's 36 plays using the four genres as dimensions using 2-variable projection functions.

We can also consider the plays broken into their constituent acts. Does the genre of a play manifest itself in the acts individually? We can create projection functions designed to map the 180 acts ($36 * 5$) to their constituent genres. Again, simple functions can perform well, with outliers that provide food for thought for scholars. Not all concepts lead to workable explainers. For example, we tried to create projection functions for specific acts (e.g. are Act 5s distinctive from other acts). The only projections that our methods were able to find were far too complex to provide any meaningful insight.

The distance metric example of Figure 6 (§5.3) also illustrates how the genre is reflected in rhetorical usage. The four projections, corresponding to the four genres were used to create the distance metric, so it should not be surprising that the distances reflect the genres. However, in Figure 6 we can observe many facts that were not provided to the system. For example, the late plays mix elements of all genres, and tend to cluster towards the center, although some acts are quite comedic (they are sometimes referred to as tragi-comedies). The tragedy outliers in the comedy region are the expected acts where comic events occur (e.g. the courtship of Romeo and Juliet in Act 2).

Our approach allows us to consider the uniqueness of individual plays. For example, we can define a Hamlet-ness metric by identifying projections that score the 5 acts of Hamlet above the 175 other acts (Figure 5). Interestingly, simple projections either score well on the first acts, or the last act. To create a Hamlet function that is correct over all acts requires quite a complex function. This suggests that there is no common distinctiveness to the acts of Hamlet, in terms of their rhetorical form usage.

6.2 Novels

We applied our system to the results of Docuscope tagging a collection of 343 18th- and 19th- century novels. One question to ask is whether the works of a particular author are distinctive in terms of their rhetorical style. For some authors, such as Defoe and Austen, there are 3-variable, quantized functions that achieve perfect correctness. For Charles Dickens, even the best 3-variable classifiers put a few novels as being “more Dickensian than Dickens.” Examining these outliers shows that they are authors that were thought emulate Dickens’ style.

The fact that Austen’s style can be observed so readily using the limited measurements of Docuscope data is interesting to our litera-

ture scholar collaborators. Factor analysis could have been used to determine that these works are distinct. However, explainers provide simpler projections (one has 3 variables with unit weights) that more easily lead to theories, and a diverse set of projections (there are 7 that perfectly discriminate) permits the scholar to consider multiple ideas to see which is likely to have meaning.

7 DISCUSSION

This paper described an approach for exploration and discovery in high-dimensional data using the idea of crafting projection functions to serve the users needs. By generating projection functions that align with user specifications, and by considering tradeoffs in correctness, simplicity, and diversity, our approach creates derived dimensions that serve to both organize the data, as well as show its connection to properties of interest. We enhance the rank-by-feature framework to consider new types of properties, and a richer set of functions.

We have limited the approach to linear projection functions. Arguments against non-linear functions include computational practicality, understandability, and control of over-fitting. However, these are tradeoffs, similar to those made with linear functions. Just as a user may sometimes be willing to trade the simplicity of an axis aligned projection for the expressiveness of a more general linear function, non-linear functions may be warranted. Non-linearity is just another simplicity tradeoff: is a simple non-linear function (e.g. $x^2 + y^2$) more difficult to interpret than a linear one involving more variables and having obscure weights? In the future, we hope to make a more rigorous study of the tradeoff between function complexity and interpretability.

At present, our experiments have been on relatively small scale examples, 40-140 variables and 30-1500 objects. The scalability of our approach is a concern. Less because of computation time, but more in terms of the presentation and interpretation of the results. The visualizations we provide already break down at current scales. Future work will consider more scalable visualizations and interactions. With more variables, there is more potential for redundancy. Our present methods avoid redundant variables as they minimize the number of variables used. However, we hope to combine our approach with the methods such as [43] that help identify redundancy. Similarly, larger numbers of elements increase the likelihood that more complex classifiers may be necessary to capture more complex phenomena. We seek to adapt our approach to help users make sense of these large and complex classifiers.

Another limitation in our work is understanding the statistical significance of the projections. Intuitively, the existence of a simpler projection seems to be a stronger indication that the property is really represented in the data, however, we seek to formalize this.

Our current approach considers user knowledge and needs only through specified partial order relationships that define properties. We are exploring extensions to other forms of specifications. In particular, we do not provide a clean way to consider multi-class separation, only considering the classes independently. And, our approach considers a single dimension at a time. While this has advantages in simplicity, control over redundancy, and the availability of numerical techniques, it can be limiting. Our present implementation does not provide a unified user experience, however we are developing one.

Even with its limitations, our approach to crafting projection functions has proven useful to our collaborators, and shows promise in other application domains. The ability to define dimensions of interest, such as comedic-ness or Paris-ness, rather than being required to show data variables or statistically defined dimensions (such as variance maximal), permits creating viewpoints that are meaningful and relevant. Control over the tradeoffs allows for identifying functions that help explain relations between variables and properties. We believe that our approach will be the basis for a variety of useful tools.

Acknowledgements: This work was supported in part by NSF awards IIS-1162037, DRL-0918409, DRL-1247262 and CMMI-0941013, and a grant from the Andrew Mellon Foundation. We thank our collaborators on the Visualizing English Print project for providing problems that motivated the approach, especially Robin Valenza for her help in articulating it.

REFERENCES

- [1] A. Anand, L. Wilkinson, and T. N. Dang. Visual Pattern Discovery using Random Projections. In *IEEE VAST*, 2012.
- [2] M. Ankerst, S. Berchtold, and D. Keim. Similarity clustering of dimensions for an enhanced visualization of multidimensional data. In *IEEE InfoVis*, 1998.
- [3] E. T. Brown, J. Liu, C. E. Brodley, and R. Chang. Dis-function: Learning distance functions interactively. In *IEEE VAST*, 2012.
- [4] M. W. Browne. An Overview of Analytic Rotation in Exploratory Factor Analysis. *Multivariate Behavioral Research*, 36(1):111–150, Jan. 2001.
- [5] Buzzdata. Best City Contest, 2012.
- [6] J. Collins, D. Kaufer, P. Vlachos, and S. Ishizaki. Detecting collaborations in text comparing the authors’ rhetorical language choices in the Federalist Papers. *Computers and the Humanities*, 38(1):15–36, 2004.
- [7] M. Correll, D. Albers, S. Franconeri, and M. Gleicher. Comparing averages in time series data. In *CHI*, pages 1095–1104, May 2012.
- [8] M. Correll, M. Witmore, and M. Gleicher. Exploring Collections of Tagged Text for Literary Scholarship. *Computer Graphics Forum*, 30(3):731–740, June 2011.
- [9] I. S. Dhillon, D. S. Modha, and W. Spangler. Class visualization of high-dimensional data with applications. *Computational Statistics & Data Analysis*, 41(1):59–90, Nov. 2002.
- [10] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. A. Efros. What makes Paris look like Paris? *ACM Trans. on Graphics*, 31(4):101, July 2012.
- [11] A. Endert, C. Han, D. Maiti, L. House, S. Leman, and C. North. Observation-Level Models for Visual Analytics. In *IEEE VAST*, 2011.
- [12] L. R. Fabrigar, D. T. Wegener, R. C. MacCallum, and E. J. Strahan. Evaluating the Use of Exploratory Factor Analysis in Psychological Research. *Psychological Methods*, 4(3):272–299, 1999.
- [13] J. Faith. Targeted Projection Pursuit for Interactive Exploration of High-Dimensional Data Sets. In *2007 11th International Conference Information Visualization (IV ’07)*, pages 286–292. IEEE, July 2007.
- [14] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research*, 9:1871–1874, June 2008.
- [15] S. J. Fernstad, J. Shaw, and J. Johansson. Quality-based guidance for exploratory dimensionality reduction. *Information Visualization*, 12(1):44–64, Oct. 2012.
- [16] J. Friedman and J. Tukey. A Projection Pursuit Algorithm for Exploratory Data Analysis. *IEEE Trans. on Computers*, C-23(9):881–890, Sept. 1974.
- [17] R. Gnandesikan, J. Kettnering, and J. Landwehr. Projection plots for displaying clusters. In *Statistics and Probability: Essays in Honor of C.R. Rao*, pages 269–280. 1982.
- [18] D. Guo. Coordinating computational and visual approaches for interactive feature selection and multivariate clustering. *Information Visualization*, 2(4):232–246, Dec. 2003.
- [19] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, Mar. 2003.
- [20] M. Hearst, B. Schölkopf, S. Dumais, E. Osuna, and J. Platt. Support Vector Machines. *IEEE Intelligent Systems*, (July/August):18–28, 1998.
- [21] F. Heimerl, S. Koch, H. Bosch, and T. Ertl. Visual Classifier Training for Text Document Retrieval. *IEEE Trans. on Vis. and Comp. Graphics*, 18(12):2839–2848, Dec. 2012.
- [22] J. Hope and M. Witmore. The Very Large Textual Object: A Prosthetic Reading of Shakespeare. *Early Modern Literary Studies*, 9(3):1–36, 2004.
- [23] S. Ingram, T. Munzner, V. Irvine, M. Tory, S. Bergner, and T. Moller. DimStiller: Workflows for dimensional analysis and reduction. In *IEEE VAST*, 2010.
- [24] S. Ishizaki and D. Kaufer. DocuScope: Computer-aided rhetorical analysis. In P. McCarthy and C. Boonthum, editors, *Applied Natural Language Processing and Content Analysis: Advances in Identification, Investigation, and Resolution*. IGI Global, 2011.
- [25] S. Johansson and J. Johansson. Interactive dimensionality reduction through user-defined combinations of quality metrics. *IEEE Trans. on Vis. and Comp. Graphics*, 15(6):993–1000, 2009.
- [26] P. Joia, F. V. Paulovich, D. Coimbra, J. A. Cuminato, and L. G. Nonato. Local Affine Multidimensional Projection. *IEEE Trans. on Vis. and Comp. Graphics*, 17(12):2563–71, Dec. 2011.
- [27] E. Kandogan. Just-in-Time Annotation of Clusters, Outliers, and Trends in Point-based Data Visualizations. In *IEEE VAST*, 2012.
- [28] D. Kaufer, C. Geisler, P. Vlachos, and S. Ishizaki. Mining Textual Knowledge for Writing Research and Education. In L. V. Waes, M. Leijten, and C. Neuwirth, editors, *Writing & Digital Media*, pages 115–129. 2006.
- [29] Y. Koren and L. Carmel. Robust linear dimensionality reduction. *IEEE Trans. on Vis. and Comp. Graphics*, 10(4):459–70, Jan. 2004.
- [30] S. Lespinats and M. Aupetit. CheckViz: Sanity Check and Topological Clues for Linear and Non-Linear Mappings. *Computer Graphics Forum*, 30(1):113–125, Mar. 2011.
- [31] J. Lewis, L. van der Maaten, and V. de Sa. A Psychophysical Investigation of Dimensionality Reduction. In *NIPS Workshop on the Challenges of Data Visualization*, page NP, 2010.
- [32] T. Malisiewicz and A. A. Efros. Beyond Categories: The Visual Memex Model for Reasoning About Object Relationships. In *NIPS*, 2009.
- [33] T. May, A. Bannach, J. Davey, T. Ruppert, and J. Kohlhammer. Guiding feature subset selection with an interactive visualization. In *IEEE VAST*, 2011.
- [34] A. Patro, N. Mehta, M. Ward, and E. Rundensteiner. Value and Relation Display for Interactive Exploration of High Dimensional Datasets. In *IEEE InfoVis*, 2004.
- [35] F. V. Paulovich, L. G. Nonato, R. Minghim, and H. Levkowitz. Least square projection: a fast high-precision multidimensional projection technique and its application to document mapping. *IEEE Trans. on Vis. and Comp. Graphics*, 14(3):564–75, 2008.
- [36] H. Piringer, W. Berger, and H. Hauser. Quantifying and Comparing Features in High-Dimensional Datasets. In *2008 12th International Conference Information Visualisation*, pages 240–245. IEEE, July 2008.
- [37] D. Powers. Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011.
- [38] M. Sedlmair, A. Tatu, T. Munzner, and M. Tory. A Taxonomy of Visual Cluster Separation Factors. *Computer Graphics Forum*, 31(3):1335–1344, June 2012.
- [39] J. Seo and B. Shneiderman. A rank-by-feature framework for interactive exploration of multidimensional data. *Information Visualization*, 4(2):96–113, May 2005.
- [40] M. Sips, B. Neubert, J. P. Lewis, and P. Hanrahan. Selecting good views of high-dimensional data using class consistency. *Computer Graphics Forum*, 28(3):831–838, June 2009.
- [41] S. Sra, S. Nowozin, and S. Wright, editors. *Optimization for Machine Learning*. MIT Press, Cambridge, 2012.
- [42] E. Tejada, R. Minghim, and L. Gustavo Nonato. On improved projection techniques to support visual exploration of multi-dimensional data sets. *Information Visualization*, 2(4):218–231, Dec. 2003.
- [43] C. Turkay, A. Lundervold, A. J. Lundervold, and H. Hauser. Representative Factor Generation for the Interactive Visual Analysis of High-Dimensional Data. *IEEE Trans. on Vis. and Comp. Graphics*, 18(12):2621–2630, 2012.
- [44] L. Wilkinson, A. Anand, and R. Grossman. Graph-theoretic scagnostics. In *IEEE InfoVis*, pages 157–164, 2005.
- [45] L. Wilkinson, A. Anand, and R. Grossman. High-dimensional visual analytics: interactive exploration guided by pairwise views of point distributions. *IEEE Trans. on Vis. and Comp. Graphics*, 12(6):1363–72, 2006.
- [46] M. Witmore and J. Hope. The Hundredth Psalm to the Tune of “Green Sleeves”: Digital Approaches to Shakespeare’s Language of Genre. *Shakespeare Quarterly*, 61(3):357–390, 2010.
- [47] I. H. Witten, E. Frank, and M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*, 3e. Morgan Kaufmann, 2011.
- [48] P. C. Wong and R. D. Bergeron. 30 Years of Multidimensional Multivariate Visualization. In *Scientific Visualization, Overviews, Methodologies, and Techniques*, pages 3–33. IEEE Computer Society, May 1997.
- [49] S. Wright, R. Nowak, and M. Figueiredo. Sparse Reconstruction by Separable Approximation. *IEEE Trans. on Signal Processing*, 57(7):2479–2493, July 2009.
- [50] J. Yang, M. O. Ward, E. A. Rundensteiner, and S. Huang. Visual hierarchical dimension reduction for exploration of high dimensional datasets. In *VISSYM ’03 Proceedings of the symposium on Data visualisation*, pages 19–28, May 2003.
- [51] L. Yang. Distance Metric Learning: A Comprehensive Survey. 2005.